Architectures pour les logiciels de Réalité Augmentée : des concepts aux applications

Jean-Yves Didier

UNIVERSITÉ D'ÉVRY – VAL D'ESSONNE
Laboratoire d'Informatique, Biologie Intégrative et Systèmes Complexes

Habilitation à diriger des recherches

Spécialité : ROBOTIQUE

Architectures pour les logiciels de Réalité Augmentée : des concepts aux applications

---

Jean-Yves Didier

Date prévue : 10 décembre 2018

JURY

M. David Fofi,	Professeur, Université de Bourgogne	Rapporteur
M. Guillaume Moreau,	Professeur, Ecole Centrale de Nantes	Rapporteur
M. Olivier H. Roux,	Professeur, Ecole Centrale de Nantes	Rapporteur
M. Jean-Pierre Jessel	Professeur, Université Paul Sabatier (Toulouse)	Examinateur
M. Frédéric Mérienne	Professeur, Arts et métiers Paris-tech	Examinateur
M. Samir Otmane,	Professeur, Université d'Evry val d'Essonne,	Examinateur
M. Malik Mallem,	Professeur, Université d'Evry val d'Essonne,	Examinateur

Remerciements

Mener à bien un projet de recherche est quelquechose qui ne se fait pas tout seul. Le chercheur, par la nature même de son travail, est amené à confronter ce qu'il a réalisé avec ce que d'autres ont déjà développé, à s'en nourrir et à l'utiliser pour enrichir son propre travail. Comme Sir Isaac Newton l'a si bien résumé : « *If I have seen further, it is by standing on the shoulders of giants.* » ; le processus visant à repousser les limites des connaissances humaines est essentiellement incrémental. Que les personnes qui ont écrit les articles qui m'ont inspiré ou poussé dans une direction ou une autre soient remerciées.

Je tiens à exprimer ma reconnaissance auprès de
Monsieur David Fofi, Professeur à l'Université de Bourgogne,
Monsieur Guillaume Moreau, Professeur à l'Ecole Centrale de Nantes,
Monsieur Olivier H. Roux, Professeur à l'Ecole Centrale de Nantes,
Monsieur Jean-Pierre Jessel, Professeur à l'Université Paul Sabatier (Toulouse),
Monsieur Frédéric Mérienne, Professeur aux Arts et métiers Paris-Tech,
Monsieur Samir Otmane, Professeur à l'Université d'Evry val d'Essonne,
Monsieur Malik Mallem, Professeur à l'Université d'Evry val d'Essonne,
pour avoir accepté d'évaluer ces travaux. Je remercie plus particulièrement Messieurs David Fofi, Guillaume Moreau et Olivier H. Roux d'avoir accepté la lourde tâche de rapporter cette habilitation. Qu'ils puissent me pardonner son volume conséquent.

Je pense également à mes collaborateurs les plus proches au cours de ces années passées au laboratoire IBISC. Que Malik, Hanna, Samir, David, Fakhr-eddine soient remerciés. Je n'oublierai pas non plus les doctorants avec lesquels j'ai travaillé et que j'ai encadré et sans qui la présente habilitation n'aurait pu voir le jour : merci donc à Benjamin, Imane, Mehdi, Alia et Insaf. De nombreux stagiaires les ont épaulés également dans leur travaux et les miens. La place manquerait pour les citer tous, mais cela ne m'empêchera pas de leur être également reconnaissant.

Un chercheur ne peut s'épanouir que dans un environnement qui lui est propice. De nombreux facteurs entrent en jeux. Tout d'abord à l'échelle du laboratoire, où ses directeurs successifs : Etienne, Jean-Louis, Saïd et Franck ont contribués au développement de cet environnement en maintenant parfois la barque en des temps difficiles. L'ambiance dans l'équipe de travail est également prépondérante, et les échanges sont également importants, merci donc à Christophe (le grand) et Christophe (le petit), tous les doctorants de l'équipe RATC (Réalité Augmentée et Travail Collaboratif) puis IRA2 (Interactions, Réalité Augmentée et Robotique Ambiante) avec lesquels j'ai pu échanger et les autres membres du laboratoire. Qu'ils me pardonnent de ne pas tous les citer nommément.

Enfin, comment ne pas oublier ma famille : avec tout d'abord mes parents qui m'ont inculqué le goût du travail bien fait et la persévérance, qualités qui sont nécessaires à un chercheur aujourd'hui. Finalement, il me faut aussi remercier ma femme Kaoli pour son soutien durant cette période.

Table des matières

Table des figures

Liste des tableaux

Introduction

La réalité augmentée (RA) est une technique qui rend possible le mixage d'entités virtuelles corrélées avec des informations issues du monde réel dans le but d'enrichir l'expérience utilisateur de la réalité. Dans sa forme la plus commune, le monde réel est filmé par une caméra et le flux vidéo produit est ensuite traité pour y incruster des images virtuelles. Selon l'une des définitions couramment donnée dans la littérature [AZUMA 1997], un système de réalité augmentée se doit de combiner des objets réels et virtuels dans un environnement réel, d'être temps réel (c'est à dire dans ce contexte avec une latence peu ou pas perceptible pour l'utilisateur) et de recaler (c'est à dire aligner ou mettre en place une cohérence spatiale et temporelle) entre les objets réels et virtuels.

De manière large, cela implique qu'au niveau des flux de données, un système de réalité augmentée commence par acquérir des informations sur le monde réel par le biais de capteurs, puis traite ces informations afin de les mettre en correspondance avec des connaissances *a priori* que le système possède du monde réel. Ceci est ensuite restitué à l'opérateur qui peut agir sur l'application en retour.

Développer une application de réalité augmentée nécessite un certain nombre de compétences. Des connaissances en traitement d'images et en traitement du signal sont nécessaires de manière à extraire les informations pertinentes pour l'application dans les flux de données issus des capteurs. De plus, des compétences en synthèse d'image sont nécessaires afin de réaliser la composition entre les augmentations virtuelles et les images du monde réel. Cela peut-être couplé avec des connaissances plus approfondies en réalité virtuelle et faire appel à des techniques d'interaction 3D et d'interaction en réalité mixte. Ce sont les compétences nécessaires pour traiter les *exigences dites fonctionnelles* (proches du « coeur de métier » du concepteur de systèmes de réalité augmentée).

Des compétences transversales en programmation et en génie logiciel sont également nécessaires et sont appelées *exigences non fonctionnelles*. La tendance est également à réaliser des applications connectées et/ou distribuées, tirant partie, quand cela est possible, des ressources processeurs lorsqu'elles sont multiples. Enfin des problématiques de temps réel mou viennent se greffer dessus puisque ces applications coordonnent des capteurs et des dispostifs de rendu fonctionnant à différentes échelles de temps (du GPS fonctionnant à un hertz à la boucle haptique qui, pour simuler des contacts rigides efficacement, doit fonctionner à une fréquence proche du kilohertz). De plus, le domaine de la réalité étant un domaine jeune, ce dernier évolue rapidement (au niveau matériel et logiciel), ce qui amène un besoin de standardisation des parties afin de pouvoir les remplacer plus facilement. Enfin, dans la perspective de pouvoir obtenir des configurations opérationnelles rapidement, il importe de fournir des solution de prototypage rapide d'application de réalité augmentée.

L'objectif général de cette habilitation est de s'intéresser à ces exigences non fonctionnelles, l'idée étant de décharger les concepteurs d'application de réalité augmentée de ces considérations pour qu'ils se concentrent sur leur coeur de métier. Toutefois, proposer des architectures est une chose, les valider en est une autre. Pour cela, il faut aussi transposer ces concepts dans le cadre d'applications réelles. C'est pourquoi, cette habilitation porte le tite : *Architectures pour les logiciels de Réalité Augmentée : des concepts aux applications*. Elle résume près de douze années de recherches consécutives à ma prise de fonction en septembre 2006 et s'inscrit dans la lignée des travaux entamés au cours de ma thèse. Le plan des recherches entreprises correspond peu ou prou à celui de cette habilitation

même si leur présentation ne correspond pas à l'ordre chronologique.

Ce mémoire est articulé en deux parties :

— **La partie I**, ne contenant qu'un seul chapitre, présente non seulement mon *curriculum vitae* mais aussi mes activités de recherche, de valorisation, d'enseignement et d'administration ;

— **La partie II**, plus conséquente et composée de trois chapitres, présente les travaux de recherche entrepris.

Le chapitre 2 pose le cadre des travaux de recherche. Nous y présenterons le contexte, à savoir le domaine de la réalité augmentée, ses concepts et ses applications. Ce tour d'horizon nous permettra de poser les exigences logicielles du domaine, qu'elles soient fonctionnelles lorsqu'elles touchent au coeur de métier ou non-fonctionnelle, c'est à dire transversales. Cela nous permettra de préciser notre plan de recherche et de remettre nos productions scientifiques dans leur contexte.

Le chapitre 3 détaillera nos propositions d'architectures logicielles pour les applications de réalité augmentée. Il traitera essentiellement des exigences non fonctionnelles. Nous verrons quelles sont nos contributions en termes d'architecture modulaire et reconfigurables afin de répondre aux besoins d'un domaine considéré comme étant encore jeune. Nous y traiterons des solutions utilisées telle que la programmation orientée composant et comment elles sont mises en oeuvre dans un modéle d'application permettant, à l'exécution, une reconfiguration des chemins de données, rendant l'architecture adaptable à des aléas tels qu'une panne de capteur. Nous verrons comment nous étendons cette architecture pour y intégrer la problématique de la distribution des traitements au sein d'un réseau, que ce soit par le biais d'un protocole dédié ou au travers de solutions standardisées telles que les *webservices*. Ceci est particulièrement valable dans le cadre de terminaux à faible puissance embarquée qui peuvent malgré tout bénéficier d'une puissance de calcul distante. A ce sujet, nous verrons quelles sont les recommandations que nous donnons en termes de scénario de distribution. Enfin, nous traiterons des aspects parallèle et des contraintes temporelles dans les applications de réalité augmentée. De part leur nature et l'utilisation de données provenant de plusieurs capteurs ou, pour celles combinant modalités visuelles et haptiques, le lancement de plusieurs fils d'exécution se justifie. Toutefois cela introduit des problèmes en terme de concurrence pour lesquels nous proposons des solutions. Dans le même ordre d'idée ces systèmes utilisent des capteurs et des dispositifs de restitution aux échelles de temps diverses, ce qui rend leur synchronisation délicate. L'idée derrière l'analyse des contraintes temporelles est de pouvoir anticiper à l'avance un dysfonctionnement afin de le corriger avant même l'implémentation sur la plate-forme physique, ce qui accroit la sûreté de fonctionnement de telles applications. Ces contributions sont valorisées au travers de deux *framework*, ARCS (pour *Augmented Reality Component System*) et MIRELA (pour *MIxed REality LAnguage*) dont nous ferons une présentation détaillée.

Le chapitre 4 présente, dans un ordre thématique, les diverses exigences fonctionnelles et applications sur lesquelles j'ai pu travailler depuis ma prise de fonction. Elles présentent une vue complémentaires au chapitre précédent. En effet, elles sont aussi une application des concepts architecturaux et, par les problématiques réelles qu'elles soulèvent, servent à les enrichir. Sans ces dernières les architectures présentées n'auraient pas de substance. Nous commencerons par le sujet de la localisation, omni-présent dans les applications de réalité augmentée et verrons comment nous avons mis au point un capteur hybride pour la localisation à grande échelle en extérieur. Ensuite, nous nous intéresserons à l'interprétation des processus et des scènes observées par le système dans le but de fournir une assistance à l'opérateur. Dans le premier cas, cela sera appliqué au suivi de procédure de maintenance, sujet que nous connaissons bien puisqu'il fût au coeur des problématiques de ma thèse. Dans le deuxième cas, ce sera pour reconnaître les gestes et l'état émotionnel d'un utilisateur dans le cadre de la téléopération, ce qui ouvre la voie à de nouvelles interactions entre l'homme et

la machine. Enfin, nous nous intéresserons à la problématique de la multi-modalité et verrons comment croiser interactions visuelles et haptiques. Tout au long de ce chapitre, même si nous présentons essentiellement des applications, nous veillerons à expliquer les liens qu'elles entretiennent avec les architectures logicielles proposées dans le chapitre 3.

Enfin, nous dresserons le bilan de ces douze années d'activités et verront quelles sont les perspectives que nous pouvons donner à ces travaux.

Première partie

Présentation du candidat

Chapitre 1

Curriculum Vitae étendu

1.1 Présentation succincte

1.1.1 Biographie

J'ai obtenu mon DEA RVMSC (Réalité Virtuelle et Maîtrise des Systèmes Complexes, devenu depuis le Master Réalité Virtuelle et Systèmes Intelligents) en 2002, la même année que mon diplôme d'ingénieur à l'IIE (Institut d'Informatique d'Entreprise, devenue depuis ENSIIE). Récipiendaire d'un doctorat en robotique en 2005, je me suis orienté vers une carrière d'enseignant chercheur. En effet, depuis septembre 2006, je suis en poste en tant que Maître de Conférences à l'Université d'Evry-Val d'Essonne. J'effectue mes enseignements au sein du département Génie Informatique de l'UFR Sciences et Technologies. Dans le même temps, je suis rattaché au laboratoire IBISC (Informatique, Biologie Intégrative et Système Complexe) dans l'équipe de recherches IRA2 (Interaction, Réalité Augmentée et Robotique Ambiante). Dans la droite ligne de mes travaux initiés en thèse, je me suis consacré à l'étude et à la conception de systèmes de Réalité Augmentée du point de vue logiciel.

1.1.2 Etat civil

Nom et prénom :	**Jean-Yves DIDIER**
Date et lieu de naissance :	2 février 1980 à Épinal (Vosges)
Situation familiale :	Marié
Coordonnées professionnelles :	Laboratoire IBISC 40, rue du Pelvoux CE 1455 Courcouronnes 91020 Evry Cedex Tél : +33 (0) 1 69 47 75 74 Courriel : jeanyves.didier@univ-evry.fr

1.1.3 Cursus académique

2002 – 2005 Préparation de thèse de doctorat au Laboratoire des Systèmes Complexes (LSC, devenu IBISC en 2006) rattaché à l'université d'Evry-Val d'Essonne. Sujet : *Contributions à la dextérité d'un système de réalité augmentée mobile appliqué à la maintenance industrielle*. Thèse soutenue publiquement le 12 décembre 2005.

2001 – 2002 DEA RVMSC (Réalité Virtuelle et Maîtrise des Systèmes Complexes) au CEA de Saclay, mention Bien.

1999 – 2002 Elève ingénieur à l'IIE (Institut d'Informatique d'Entreprise, Evry 91)
Options de deuxième année : Robotique, SEA-LAN (Système d'exploitation, archi-

tecture matérielle et administration de réseaux).
Option de troisième année : Optimisation mathématique.

1997 – 1999 Classe Préparatoire aux Grandes Ecoles, filière MP (Mathématiques, Physique) au lycée Claude Gellée, Epinal (88)

1997 Baccalauréat série S mention Très Bien.

1.1.4 Parcours professionnel

09/2006 – maintenant Maître de conférences en Génie Informatique à l'UFR-ST d'Évry (91), membre du laboratoire (IBISC)

07/2009 – 01/2010 Obtention d'un CRCT de 6 mois

10/2005 – 08/2006 Attaché Temporaire à l'Enseignement et à la Recherche (ATER) à l'Institut d'Informatique d'Entreprise (IIE - CNAM)

10/2004 – 09/2005 ATER à l'UFR-ST d'Évry

09/2002 – 08/2004 Ingénieur de recherche contractuel au LSC sur le Projet RNTL (Réseau National des Technologies Logicielles) AMRA (Assistance à la Maintenance en Réalité Augmentée).

09/2002 – 08/2004 Monitorat à l'UFR-ST d'Évry.

01/2002 – 06/2002 Stage de six mois au LSC. Sujet : *le recalage dynamique dans un système de réalité augmentée en vision indirecte.*

06/2001 – 08/2001 Stage de trois mois au LSC. Sujet : *le recalage d'objets en trois dimensions de formes libres à partir d'une analyse photoclinométrique de ces derniers.*

06/2000 – 08/2000 Stage de trois mois à PCA Informatique, Gérardmer (88).
Sujet : *Réalisation d'outils améliorant le confort d'utilisation des logiciels développés par l'entreprise*.

1.2 Activités de recherche

1.2.1 Encadrement de la recherche

1.2.1.1 Thèses

Doctorant	Financement	Taux	Années	Mots clés
Insaf Ajili	MESR	50 %	2015 - 2018	Reconnaissance du geste et des émotions, interaction
Alia Rukubayihunga	CIFRE	50 %	2013 - 2016	Réalité augmentée, suivi, maintenance industrielle
Mehdi Chouiten	MESR	50 %	2008 - 2013	Réalité augmentée, distribution, web service
Imane Zendjebil	ANR	40 % [1]	2006 - 2010	Réalité augmentée, capteur hybride, recalage basé modèle
Benjamin Bayart	MESR	40 % [2]	2006 - 2007	Réalité augmentée, haptique, interaction

TABLE 1.1 – Taux d'encadrement des thèses

La table 1.1 page ci-contre introduit les taux d'encadrement, par année, des thèses que j'ai suivies. Les titres et la composition des jurys associés à ces dernières sont données ci-après. Depuis 2015, les écoles doctorales de l'université d'Evry-val d'Essonne sont intégrées à celles de Paris-Saclay qui ne délivre pas de mention.

Insaf Ajili

Analyse et reconnaissance de gestes expressifs en se basant sur la méthode LMA, soutenue le 3 décembre 2018, devant le jury suivant :

— Mme Catherine Achard, MdC HDR Sorbonne Université, Rapporteure ;

— M. Fakhr-Eddine Ababsa, Prof. Arts et Métiers ParisTech, Rapporteur ;

— M. Titus Zaharia, Prof. Telecom SudParis, Examinateur ;

— Mme Indira Thouvenin, Prof. Université de technologie de Compiègne, Examinatrice ;

— M. Jean-Yves Didier, MdC Université d'Evry, Encadrant ;

— M. Malik MALLEM, Prof. Université d'Evry, Directeur de thèse.

Alia Rukubayihunga

Vers un système interactif de réalité augmentée mobile pour la supervision de scénarios de maintenance industrielle, thèse CIFRE (convention n° 2013/0692) effectuée en partenariat avec l'entreprise Wassa, société spécialisée dans le Web et la mobilité et soutenue le 15 décembre 2016, devant le jury suivant :

— M. Guillaume Moreau, Prof. Ecole Centrale de Nantes, Rapporteur ;

— M. Jean-Marc Cieutat, Chercheur associé (HDR), ESTIA, Bidart, Rapporteur ;

— M. Frédéric Mérienne, Prof. ENSAM, Institut Image, Examinateur ;

— M. Frédéric Sommerlat, Directeur du développement, WASSA SAS, Co-encadrant ;

— M. Jean-Yves Didier, MdC Université d'Evry, Co-encadrant ;

— M. Samir Otmane, Prof. Université d'Evry-val-d'Essonne, Directeur de thèse.

Mehdi Chouiten

Architecture distribuée dédiée aux applications de Réalité Augmentée mobile, thèse soutenue le 31 janvier 2013, obtenue avec la mention très honorable, devant le jury suivant :

— Mme Laurence Duchien, Prof. Université Lille 1, Rapporteur ;

— M. Guillaume Moreau, Prof. Ecole Centrale de Nantes, Rapporteur ;

— M. Pascal Poizat, Prof. Université Paris Ouest Nanterre la Défense, Examinateur ;

— M. Romain Rouvoy, MdC Université Lille 1, Examinateur ;

— M. Jean-Yves Didier, MdC Université d'Evry, Encadrant ;

— M. Malik Mallem, Prof. Université d'Evry, Directeur de thèse.

1. A partir de la deuxième année
2. A partir de la deuxième moitié de la thèse

Imane Zendjebil

Localisation 3D basée sur une approche de suppléance multi-capteurs pour la Réalité Augmentée Mobile en Milieu Extérieur, thèse soutenue le 1er octobre 2010, obtenue avec mention très honorable, devant le jury suivant :

— M. David Fofi, Prof. Université de Bourgogne, Rapporteur ;

— M. Eric Marchand, Prof. Université de Rennes 1, Rapporteur ;

— M. Pascal Guitton, Prof. Université de Bordeaux 1, Examinateur ;

— M. Fakhreddine Ababsa, MdC Université d'Evry, Encadrant ;

— M. Jean-Yves Didier, MdC Université d'Evry, Co-encadrant ;

— M. Malik Mallem, Prof. Université d'Evry, Directeur de thèse.

Benjamin Bayart

Réalité Augmentée haptique : théorie et applications, thèse soutenue le 7 décembre 2007, obtenue avec la mention très honorable, devant le jury suivant :

— M. Jacques Tisseau, Prof. Ecole Nationale d'Ingénieur de Brest, Rapporteur ;

— M. André Crosnier, Prof. Université Montpellier II, Rapporteur ;

— M. Christophe Chaillou , Prof. Université Lille 1, Examinateur ;

— Mme Marie-Odile Berger, Chargé de Recherche/HDR Loria – INRIA Nancy, Examinatrice ;

— M. Jean-Yves Didier, MdC Université d'Evry, Encadrant ;

— M. Abderrahmane Kheddar, Directeur de Recherche CNRS, Directeur de thèse ;

— M. Alan Savary, Ingénieur chez Total Immersion, Invité.

1.2.1.2 Projets de fin d'études

1. Christopher De Barros, étudiant de Master GEII à l'Université d'Evry, *Étude et développement d'une application de geocaching en réalité augmentée*, février-juillet 2015 ;
2. Abir Chebli, étudiante de Master GEII à l'Université d'Evry, *Étude des dysfonctionnements des applications de réalité mixte dus aux contraintes de temps*, février-juillet 2014 ;
3. Belaïd Arib, étudiant de Master RVSI à l'Université d'Evry, *Prototypage rapide des applications de réalité augmentée : implémentation générique des automates temporisés*, février-juillet 2010 ;
4. Geoffroy Gley, étudiant de Master MOPS à l'Université d'Evry, *Prototypage rapide de systèmes de réalité virtuelle*, février-juin 2008 ;
5. Julien Soc, étudiant de Master MOPS à l'Université d'Evry, *Caractérisation et modélisation d'un système multimodal : Modélisation compositionnelle et vérification de systèmes de réalité augmentée*, février-juin 2007 ;
6. Jean-Pierre Soc, étudiant de Master MOPS à l'Université d'Evry, *Caractérisation et modélisation d'un système multimodal : transposition d'un modèle formel vers la plate-forme ARCS*, février-juin 2007 ;
7. Philippe Gagnières, étudiant de Master RVSI à l'Université d'Evry, *Markerless tracking initialisé avec une cible*, février-juillet 2006 ;
8. Aïcha Tazi, étudiante de Master RVSI à l'Université d'Evry, *Suivi multi-capteurs dans un système de réalité augmentée*, février-juillet 2006 ;

9. André Laurie, élève ingénieur en troisième année à l'ENSIIE, *Étude des modalités d'interactions Homme/Machine pour un système de Réalité Augmentée*, février-juillet 2004 ;
10. Madjid Maidi, étudiant de DEA RVMSC à l'Université d'Evry, *Calibration d'un système multi-capteurs mobile*, février-juillet 2003.

1.2.1.3 Stages intermédiaires

1. Victor Leclerc, élève ingénieur ENSIIE deuxième année, *Architecture sur ORBSlam2*, juin-août 2018 ;
2. Kévin Goilard, élève ingénieur ENSIIE deuxième année, *Implémentation de méthodes de reconnaissance d'objets peu texturés*, juin-août 2016 ;
3. Sylvain Degeorges, élève ingénieur ENSIIE deuxième année, *Optimisation d'un système de reconnaissance de mouvement contraint en réalité augmentée*, juin-août 2016 ;
4. Johan Arcile, étudiant en troisième année de licence, *Implémentation d'un outil de compilation des spécifications MIRELA vers les automates temporisés au format UPPAAL (XML)*, juin 2014 ;
5. Mathieu Moine, élève ingénieur ENSIIE deuxième année, *Implementation tool of Timed Automata specifications*, juin-août 2013 ;
6. Jonathann Jacobs-Van Poucke, étudiant en troisième année de licence, *Développement de composants logiciels : intégration de VRPN dans le framework de réalité augmentée ARCS*, juin 2013 ;
7. Marouane Hannafi, élève ingénieur ENSIIE deuxième année, *Conception et développement de composants logiciels pour traiter le flux vidéo d'un ARDrone*, juin-août 2012 ;
8. Pierre Mauvy, élève ingénieur ENSIIE deuxième année, *Algorithmes d'optimisation pour la vision par ordinateur*, juin-août 2011 ;
9. Martin Hild, élève ingénieur ENSIIE première année, *Conception et réalisation d'un environnement de programmation visuelle*, juin-septembre 2009 ;
10. Pierre Etienne Bougué, élève ingénieur ENSIIE deuxième année, *Formatage de modèles 3D maillés en vue d'appliquer des algorithmes de vision par ordinateur*, juin-août 2007 ;
11. Cyril Lanquetuit, élève ingénieur ENSIIE deuxième année, *Implémentation de l'algorithme de l'itération orthogonale - Application à la réalité augmentée sur un système de stéréovision*, juin-août 2005 ;
12. Yohann Petit, élève ingénieur maître IUP GEII Evry, *Conception et développement d'un environnement intégré dédié au prototypage rapide d'applications de réalité augmentée*, mars-juillet 2005 ;

1.2.2 Projets et contrats de recherche

Ci-après, la liste des projets et contrats de recherche auxquels j'ai participé est donnée. La table 1.2 page suivante en fait la synthèse en indiquant lesquels ont relevé de ma responsabilité. Pour deux d'entre eux, j'ai joué le rôle de facilitateur, étant le contact préliminaire au sein laboratoire pour le montage du projet. En effet, dans le cadre de la CIFRE avec Wassa, celle-ci m'a été proposée par un de mes anciens doctorants (M. Mehdi Chouiten). Quant au projet RAXENV, un de mes anciens camarades de promotion de DEA (M. Luc Frauciel), alors au Bureau des Ressources Géologiques et Minières (BRGM) en tant qu'ingénieur R&D, m'avait contacté.

Nom	Type	Rôle	Années	Montant (k€)
LOCA-3D	Challenge ANR	–	2017 - 2020	240,0
EZWheel	Contrat industriel	✔	2018	2,7
ICAM	Prestation de conseil	✔	2017	1,3
Wassa	Contrat CIFRE	⇄	2014 - 2016	24,0
Polonium	Partenariat Hubert Curien	–	2014	–
MIRELA	Action incitative (local)	✔	2012	2,7
RAXENV	ANR - TL	⇄	2007 - 2010	106,0
VarSCW	Action incitative (local)	–	2006 - 2009	24,0
Evr@	ASTRE (région)	–	2002 - 2005	372,0
AMRA	RNTL	–	2002 - 2004	177,0

Rôles – ✔ : responsable, ⇄ : facilitateur et participant, – : participant.

TABLE 1.2 – Projets et contrats de recherche

Challenge ANR MALIN – LOCA-3D

Responsable scientifique (côté IBISC) : Samia Bouchafa ;

Participants (côté IBISC) : Fabien Bonardi, **Jean-Yves Didier**, Hicham Hadj-Abdelkhader, Viachaslau Kachurka, David Roussel.

Budget demandé pour IBISC : 240 000 €

Type de projet : Challenge ANR MALIN

Années : 2018 – 2021

Ce challenge de l'ANR vise à produire un système de localisation en intérieur à destination d'agents civils et militaire en opération dans des bâtiments. Ce système doit aussi pouvoir effectuer la transition lors d'un passage de l'extérieur vers l'intérieur et inversement. LOCA-3D, pour Localisation, Orientation et CArtographie 3D, est une des réponses à ce challenge. IBISC s'attache à développer des méthodes de localisation par la vision, ultérieurement fusionnées avec une centrale inertielle, tout en maintenant une architecture logicielle cohérente et capable de distribuer les traitement sur des dispositifs de calcul de nature hétérogène.

Prestation de conseil pour EZ-Wheel

Responsable scientifique : **Jean-Yves Didier** ;

Montant de la prestation : 2 772 €

Type de projet : prestation de conseil

Année : 2018 (juillet)

La société EZ-Wheel est spécialisée dans la création de roues intelligentes pour équiper des chariots de transport et de manutention. Elle m'a sollicitée pour l'accompagner sur un de ses projets nécessitant la mise en oeuvre de la bibliothèque de traitement d'images OpenCV.

Prestation de conseil pour l'ICAM-Sénart (Institut Catholique des Arts et Métiers)

Responsable scientifique : **Jean-Yves Didier** ;

Montant de la prestation : 1 320 €

Type de projet : prestation de conseil

Année : 2017 (juin)

Cette prestation de conseil était lié à une problématique de transfert de connaissances et de compétences liées à l'utilisation de WebGL, bibliothèque javascript permettant de bénéficier de l'accélération matérielle 3D dans le navigateur web. L'ICAM l'a utilisé dans le cadre d'un de ses contrats industriels.

Contrat CIFRE IBISC – Wassa

Responsable scientifique : Samir Otmane ;

Participants (côté IBISC) : **Jean-Yves Didier**, Samir Otmane ;

Type de projet : Contrat CIFRE ;

Années : 2014-2016

Ce contrat a été signé entre IBISC et la société Wassa, dont une branche des activités est en lien avec le traitement d'images et l'apprentissage automatique. Le montant précisé ici est le montant touché par IBISC en plus du salaire versé dans le cadre de la thèse CIFRE de Mme Alia Rukubayihunga. C'est dans ce cadre que nous avons posé les premiers jalons pour un système de suivi automatique de procédure de maintenance.

PHC Polonium 2014

Responsable scientifique : Hanna Klaudel ;

Participants (côté IBISC) : Nazim Agoulmine, Serenella Cerrito, **Jean-Yves Didier**, Bachir Djafri, Lukasz Fronc, Hanna Klaudel, Tarek Melliti, Franck Pommereau, Viet Van Pham ;

Type de projet : Partenariats Hubert Curien (PHC) – Campus France

Années : 2014 – 2015

Il s'agit d'un projet d'échange et de coopération scientifique entre la France et la Pologne. Dans ce cadre, nous souhaitons affiner notre compréhension des dysfonctionnements associés aux contraintes temporelles dans des applications informatiques destinées au contrôle et à la commande de systèmes à base de capteurs et d'effecteurs. En plus d'une modélisation à base d'automates temporisés, nous souhaitons introduire une dimension probabiliste afin de pouvoir déterminer la fréquence ou la probabilité des dysfonctionnements rencontrés. Le partenariat avec l'Institut d'Informatique Théorique et Appliquée de Gliwice (Pologne), disposant d'une expérience dans le domaine probabiliste, nous rend à même de concevoir et mettre en oeuvre les outils permettant l'analyse fine de ces dysfonctionnements.

Action Incitative locale – MIRELA

Responsable scientifique : **Jean-Yves Didier**

Participants : Hanna Klaudel, Bachir Djafri, Mehdi Chouiten.

Budget alloué : 2 700 €

Type de projet : action interne au laboratoire

Année : 2012

Dans le cadre de l'action incitative VarSCW (voir plus loin), nous avions pu identifier un certain nombre de verrous associés à l'analyse des contraintes temporelles des applications de réalité augmentée. En particulier, les questions suivantes avaient été levées et nécessitent une réponse

— Comment réduire l'impact de l'explosion combinatoire de l'espace des états possibles des systèmes lors de la vérification des propriétés temporelles de ces derniers ?

— Comment déterminer si le système est réalisable ou non étant données sa spécification formelle et une infrastructure concrète fixée ?

— Comment assurer la validité de l'implémentation, c'est-à-dire la cohérence de la réalisation par rapport au modèle, éventuellement sous certaines hypothèses ?

Cette action incitative de durée courte a permis de redémarrer cette activité et d'apporter les premières réponses sous la forme d'un modèle dérivé des automates temporisés facilitant l'analyse de certaines propriétés temporelle d'un système de réalité augmentée.

Projet ANR – RAXENV (Réalité Augmentée en eXtérieur appliquée à l'ENVironnement), ANR06-TLOG14

Responsable scientifique, côté IBISC : Fakhreddine Ababsa

Participants : Malik Mallem, **Jean-Yves Didier**, Imane Zendjebil.

Budget alloué pour IBISC : 106 000 €

Type de projet : national (ANR)

Années : 2007 – 2010

L'objectif du projet RAXENV est de démontrer la faisabilité d'un système de réalité augmentée en extérieur dans le domaine des sciences et techniques de l'environnement, que ce soit en termes de technologie ou d'adoption par les utilisateurs finaux. Il associe 5 partenaires : le BRGM et la Lyonnaise des Eaux (comme utilisateurs finaux), le laboratoire Ibisc (Réalité augmentée), l'équipe-projet Iparla (Labri, INRIA (visualisation et interactions sur terminaux mobiles communicants)) et la société Archividéo (modèle urbain 3D, visualisation sur Internet). Le projet a démarré en février 2007 pour une durée de 3 ans et bénéficie d'une aide de l'Agence Nationale de la Recherche (ANR). Outre la part active que j'ai prise au montage du projet, mes contributions se situent au niveau de la conception et du prototypage de la chaîne d'acquisition et de traitement des données en vue de localiser le système par rapport à son environnement.

Action incitative locale – Projet VARSCW

Responsable scientifique : Samir Otmane

Participants : Malik Mallem, **Jean-Yves Didier**, Guillaume Hutzler, Hanna Klaudel, Paul Richard (Université d'Angers), Frédéric Davesne, Bachir Djafri

Budget alloué : 24 000 €

Type de projet : action interne au laboratoire

Années : 2006 – 2009

Le projet VARSCW (*Virtual and Augmented Reality Supported Collaborative Work*) avait pour objectif de développer de nouvelles architectures logicielles pour une nouvelle génération de collecticiels distribués, multimodaux et adaptatifs supportant des interfaces homme-machine multisensorielles. Une approche consiste à combiner astucieusement les techniques et les outils de RA/RV avec les modèles d'architecture logicielle des collecticiels basées sur des approches multi-agents et sur les systèmes distribués temps-réels. Ce projet renforce la collaboration scientifique entre les deux équipes RATC et LIS (devenues depuis IRA2 et COSMO) du laboratoire IBISC, dans le cadre de la réunion des deux laboratoires (LaMI et LSC) au 1er janvier 2006. Il s'agit en fait de faire vivre le nouveau pôle Interface et Interaction créé suite à cette réunion en fortifiant les liens entre les deux équipes LIS et RATC. Dans le cadre de ce projet inter-équipe, j'ai collaboré avec Hanna Klaudel (PU) et Bachir Djafri (MCF) sur la modélisation et l'analyse des contraintes temporelles dans les applications de réalité augmentée.

Réseau d'excellence européen INTUITION

Réseau d'excellence INTUITION : participation aux groupes de discussion Haptique (WG 2.10) et Réalité Augmentée (WG 2.2) (septembre 2006 - 2008). Au sein du second groupe, j'ai contribué à la rédaction du ToR (*Term of Reference*) : document qui permet de définir et de cadrer le domaine de recherche qu'est la réalité augmentée. Ce document dresse un inventaire des laboratoires et des plate-formes européennes qui sont associées à ces activités.

Projet ASTRE – Evr@

Responsable scientifique : Malik Mallem

Participants : David Roussel, Samir Otmane, **Jean-Yves Didier**, Abderrahmane Kheddar, Etienne Colle

Budget alloué : 372 000 € (ASTRE 76k€, MENRT 227k€, CNRS 60k€, RNTL AMRA 9k€)

Type de projet : régional

Années : 2002 – 2005

L'objectif de ce projet a été l'acquisition d'une plate-forme matérielle destinée aux expérimentation en réalité virtuelle et en réalité augmentée à destination des activités pédagogiques et de recherche au sein de l'Université d'Evry val d'Essonne. Cette plate-forme porte le nom d'Evr@ et dispose d'un site internet qui lui est dédié : http://evra.ibisc.univ-evry.fr

J'ai participé à la rédaction des spécifications techniques du Cahier des Clauses Techniques Particulières de l'appel d'offre 03.003 pour une marché public lancé par l'université d'Évry-Val-d'Essonne paru au BOMP B/0077-284. J'ai été plus particulièrement amené à rédiger les spécifications pour les lots 2 et 3. Celles-ci concernaient l'acquisition d'un casque de réalité semi-transparent pour la réalité virtuelle ainsi que les machines du parc informatique de la plate-forme (serveurs et stations de travail).

Projet RNTL – AMRA

Responsable scientifique côté IBISC : Malik Mallem

Participants : Samir Otmane, **Jean-Yves Didier**, David Roussel, Florent Chavand.

Budget alloué pour IBISC : 177 000 €

Type de projet : national

Années : 2002 – 2004

Le projet AMRA est un projet RNTL (Réseau National des Technologies Logicielles) placé sous la tutelle du ministère de la Recherche. Ce dernier a commencé en 2002 et s'est achevé en mai 2004. Le projet était mené par un consortium de partenaires qui sont :

— Alstom Transport, industriel apportant la problématique de travail,

— Le Commissariat à l'Énergie Atomique (CEA), plus particulièrement le Laboratoire d'Intégration des Systèmes et des Technologies (LIST), qui a développé un système de vision pour le projet, et qui a réalisé l'intégration finale du prototype,

— Le Laboratoire Systèmes Complexes (LSC) devenu depuis IBISC,

— Acti-CM, une *start-up* issue du CEA spécialisée dans la métrologie.

Le but était d'implémenter un système de Réalité Augmentée à usage mobile pour une utilisation en milieu industriel, et plus spécifiquement dans le domaine de la maintenance industrielle. Ce projet a poursuivi plusieurs objectifs :

— Fournir une aide contextuelle à des mainteneurs inexpérimentés, les formant sur site,

— Apporter aux agents de maintenance un accès, depuis leur poste de travail, à des informations pertinentes (documentation de maintenance, modes opératoires, films de montage),

— Augmenter la disponibilité de l'information sur le lieu de maintenance en utilisant les techniques de Réalité Augmentée.

Le prototype AMRA est un système de réalité augmentée en vision indirecte constitué d'un tablette-PC (un ordinateur portable allégé pourvu d'un écran tactile) pour la visualisation des informations, qui agit comme une fenêtre augmentée sur le monde réel, grâce à la caméra embarquée sur ce dernier. Ce type de système aborde plusieurs problématique : celle de l'informatique nomade (en anglais *mobile computing*), celle du recalage temps réel des entités virtuelles sur les images du monde réel, et enfin celle du développement d'une aide graphique contextuelle adaptée. C'est dans ce dernier volet que je suis principalement intervenu.

1.2.3 Dissémination scientifique

1.2.3.1 Liste des publications

La table ci-dessous regroupe des indicateurs chiffrés sur la production scientifique depuis le début de ma carrière. Elle contient également un indicateur bibliométrique : le *h-index* pour la période 2013-2018. La liste de publications donnée ci-après est classée par catégories. Lorsqu'ils sont connus, les indicateurs tels que l'*impact factor* et le *scientific journal ranking (SJR)* sont communiqués pour les journaux. De même, le rang des conférences scientifiques est indiqué [3]. En outre, diverses sources d'information agrègent cette même liste (parfois avec quelques différences). En particulier :

— Google scholar : https://scholar.google.fr/citations?user=H-yKCo8AAAAJ&hl=fr&oi=ao ;

— ORCID : http://orcid.org/0000-0002-9863-5471 ;

— Research Gate : https://www.researchgate.net/profile/Jean-Yves_Didier ;

— HAL : https://hal.archives-ouvertes.fr/search/index/q/*/authFullName_s/Jean-Yves+Didier.

Revues internationales avec comité de lecture	8
Revues nationales avec comité de lecture	2
Chapitres d'ouvrages	1
Conférences internationales avec comités de lecture	27
Workshops internationaux avec comités de lecture	11
Communications nationales	6
Délivrables de projets	6
H-index (sur la période 2013-2018)	6

TABLE 1.3 – Indicateurs bibliométriques

Revues internationales avec comités de lecture

1. AJILI, I., MALLEM, M. et **DIDIER, J.** (A paraître[a]). « Human motions and emotions recognition inspired by LMA qualities ». In : *The Visual Computer*.
2. AJILI, I., RAMEZANPANAH, Z., MALLEM, M. et DIDIER, J. (A paraître[b]). « Expressive motions recognition and analysis with learning and statistical methods ». In : *Multimedia Tools and Applications*.

3. source : https://www.conferenceranks.com

3. AJILI, I., MALLEM, M. et **DIDIER, J.-Y.** (2018). « An Efficient Motion Recognition System Based on LMA Technique and a Discrete Hidden Markov Model ». In : *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 12.9, p. 707–713. ISSN : eISSN :1307-6892.
4. CHOUITEN, M., DOMINGUES, C., **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2014). « Distributed Mixed reality for diving and underwater tasks using Remotely Operated Vehicles ». In : *International Journal on Computational Sciences & Applications* 5.4, elec–proc.
5. ABABSA, F., ZENDJEBIL, I. M., **DIDIER, J.** et MALLEM, M. (2012). « Smart Localization Using a New Sensor Association Framework for Outdoor Augmented Reality Systems ». In : *Journal of Robotics* 2012, 634758 :1–634758 :15. DOI : 10.1155/2012/634758. URL : https://doi.org/10.1155/2012/634758.
6. MAIDI, M., **DIDIER, J.-Y.**, ABABSA, F. et MALLEM, M. (2010). « A Performance Study for Camera Pose Estimation using Visual Marker based Tracking ». In : *Machine Vision and Applications, IAPR International Journal, Springer* 21.3, p. 365–376.
7. **DIDIER, J.-Y.**, DJAFRI, B. et KLAUDEL, H. (2009). « The MIRELA framework : modeling and analyzing mixed reality applications using timed automata ». In : *Journal of Virtual Reality and Broadcasting*. VRIC 2008 (Laval Virtual) Special Issue 6.1. Sous la dir. de J. HERDER, S. RICHIR et I. THOUVENIN. t urn :nbn :de :0009-6-17423,, ISSN 1860-2037. URL : http://www.jvrb.org/archiv/1742/.
8. **DIDIER, J.-Y.**, ABABSA, F.-e. et MALLEM, M. (2008). « Hybrid Camera Pose Estimation Combining Square Fiducials Localization Technique and Orthogonal Iteration Algorithm ». In : *International Journal of Image and Graphics (IJIG)* 8.1, p. 169–188.

Revues nationales avec comités de lecture

1. **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2009). « ARCS : Une Architecture Logicielle Reconfigurable pour la conception des Applications de Réalité Augmentée ». In : *Technique et Science Informatiques (TSI), Réalité Virtuelle - Réalité Augmentée* 28.6-7/2009. Numéro spécial, p. 891–919.
2. ZENDJEBIL, I. M., ABABSA, F., **DIDIER, J.-Y.**, LALAGÜE, E., DECLE, F., DELMONT, R., FRAUCIEL, L. et VAIRON, J. (2009). « Réalité Augmentée en Extérieur : Etat de l'Art ». In : *Technique et Science Informatiques (TSI), Réalité Virtuelle - Réalité Augmentée* 28.6-7/2009. Numéro spécial, p. 857–890.

Chapitres d'ouvrages

1. ABABSA, F., MAIDI, M., **DIDIER, J.-Y.** et MALLEM, M. (2008). « Vision-Based Tracking for Mobile Augmented Reality ». In : *Multimedia Services in Intelligent Environments, Springer*. P. 297–326.

Conférences internationales avec comités de lecture

1. AJILI, I., MALLEM, M. et **DIDIER, J.-Y.** (2018). « Relevant LMA Features for Human Motion Recognition ». In : *ICIAP 2018 : International Conference on Image Analysis and Processing, Paris, France, (Oct 29-30, 2018)*. T. 12. 10, p. 2899.
2. AJILI, I., **DIDIER, J.-Y.** et MALLEM, M. (2017a). « Gesture recognition for humanoid robot teleoperation ». In : *26th IEEE International Symposium on Robot and Human Interactive Communication (RO'MAN 2017)*. Lisbon, Portugal.

3. AJILI, I., MALLEM, M. et **DIDIER, J.** (2017b). « Robust human action recognition system using Laban Movement Analysis ». In : *Knowledge-Based and Intelligent Information & Engineering Systems : Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*. P. 554–563. DOI : 10.1016/j.procs.2017.08.168. URL : https://doi.org/10.1016/j.procs.2017.08.168.
4. RUKUBAYIHUNGA, A., **DIDIER, J.-Y.** et OTMANE, S. (2016a). « Real Time Noise Reduction to Identify Motion Parameters in AR Maintenance Scenario ». In : *Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on*. IEEE, p. 27–30.
5. — (2016b). « Towards assembly steps recognition in augmented reality ». In : *Proceedings of the 2016 Virtual Reality International Conference*. ACM, p. 17.
6. ARCILE, J., CZACHÓRSKI, T., DEVILLERS, R. R., **DIDIER, J.**, KLAUDEL, H. et RATAJ, A. (2015). « Modelling and Analysing Mixed Reality Applications ». In : *Man-Machine Interactions 4 - 4th International Conference on Man-Machine Interactions, ICMMI 2015, Kocierz Pass, Poland, October 6-9, 2015*, p. 3–17. DOI : 10.1007/978-3-319-23437-3_1. URL : http://dx.doi.org/10.1007/978-3-319-23437-3_1.
7. RUKUBAYIHUNGA, A., **DIDIER, J.-Y.** et OTMANE, S. (2015). « Reprojection error as a new metric to detect assembly/disassembly maintenance tasks ». In : *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*. IEEE, p. 513–518.
8. DEVILLERS, R. R., **DIDIER, J.**, KLAUDEL, H. et ARCILE, J. (2014). « Deadlock and Temporal Properties Analysis in Mixed Reality Applications ». In : *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, p. 55–65. DOI : 10.1109/ISSRE.2014.33. URL : http://dx.doi.org/10.1109/ISSRE.2014.33.
9. **DIDIER, J.-Y.** et MALLEM, M. (2014). « A new approach to detect potential race conditions in component-based systems ». In : *Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*. ACM, p. 97–106.
10. DEVILLERS, R., **DIDIER, J.-Y.** et KLAUDEL, H. (2013). « Implementing timed automata specifications : the "sandwich" approach ». In : *13th International Conference on Application of Concurrency to System Design (ACSD2013)*.
11. CHOUITEN, M., DOMINGUES, C., **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2012). « Distributed mixed reality for remote underwater telerobotics exploration ». In : *Proceedings of the 2012 Virtual Reality International Conference*. VRIC '12. Laval, France : ACM, 1 :1–1 :6. ISBN : 978-1-4503-1243-1. DOI : 10.1145/2331714.2331716. URL : http://doi.acm.org/10.1145/2331714.2331716.
12. CHOUITEN, M., **DIDIER, J.-Y.** et MALLEM, M. (2011). « Component-based middleware for distributed augmented reality applications ». In : *Proceedings of the 5th International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE 2011)*. Verona, Italy : ACM, p. 3.
13. ABABSA, F., **DIDIER, J.-Y.**, ZENDJEBIL, I. M. et MALLEM, M. (2008). « Markerless Vision-Based Tracking of Partially Known 3D Scenes for Outdoor Augmented Reality Applications ». In : *ISVC (1)*, p. 498–507.
14. BAYART, B., **DIDIER, J.-Y.** et KHEDDAR, A. (2008). « Force Feedback Virtual Painting on Real Objects : A Paradigm of Augmented Reality Haptics ». In : *Haptics : Perception, Devices and Scenarios (Proceedings of EuroHaptics 2008)*. T. 5024/2008. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 776–785.
15. **DIDIER, J.-Y.**, DJAFRI, B. et KLAUDEL, H. (2008a). « MIRELA : A Language for Modeling and Analyzing Mixed Reality Applications Using Timed Automata ». In : *IEEE Virtual Reality 08*. Sous la dir. de M. LIN, A. STEED et C. CRUZ-NEIRA. IEEE. Reno, Nevada, p. 249–250.

16. — (2008b). « Modeling and analyzing mixed reality applications using timed automata ». In : *1st Mediterranean Conference on Intelligent Systems and Automation (CISA ?08)*. Sous la dir. de H. ARIOUI, R. MERZOUKI et H. A. ABBASSI. AIP, p. 173–178. URL : http://evra.ibisc.univ-evry.fr/Proceedings/CISA08/cdr_pdfs/indexed/stage4_copyr/173_1.pdf.

17. — (2008c). « The MIRELA Framework : modeling and analyzing mixed reality applications using timed automata ». In : *10th Virtual Reality International Conference*. Laval, France, p. 189–199.

18. ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.** et MALLEM, M. (2008a). « On the Hybrid Aid-Localization for Outdoor Augmented Reality Applications ». In : *ACM Symposium on Virtual Reality Software and Technology*. Bordeaux, France.

19. ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.**, VAIRON, J., FRAUCIEL, L., HACHET, M., GUITTON, P. et DELMONT, R. (2008b). « Outdoor Augmented Reality : State of the Art and Issues ». In : *Virtual Reality International Conference*, p. 177–187.

20. BAYART, B., DRIF, A., KHEDDAR, A. et **DIDIER, J.-Y.** (2007). « Visuo-Haptic Blending Applied to a Tele-Touch-Diagnosis Application ». In : *HCI (14)*. Beijing, China, p. 617–626.

21. **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2006). « A Component Model for Augmented/-Mixed Reality Applications with Reconfigurable Data-flow ». In : *8th International Conference on Virtual Reality (VRIC 2006)*. Laval (France), p. 243–252.

22. MERAD, D., **DIDIER, J.-Y.** et SCUTURICI, M. (2006). « Tracking 3D free form object in video sequence ». In : *Third Canadian Conference on Computer and Robot Vision*. Quebec city (Canada), p. 50.

23. **DIDIER, J.-Y.**, ROUSSEL, D. et MALLEM, M. (2005). « A Time Delay Compensation Method Improving Registration for Augmented Reality ». In : *2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*. Barcelona (Spain), p. 3396–3400.

24. — (2004). « A Texture Based Time Delay Compensation Method for Augmented Reality. » In : *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004)*. Arlington (USA), p. 262–263.

25. ABABSA, F., **DIDIER, J.-Y.**, MALLEM, M. et ROUSSEL, D. (2003). « Head motion prediction in augmented reality systems using monte carlo particle filters ». In : *Proceedings of the 13th International Conference on Artificial Reality and Telexixtance (ICAT 2003)*. The Virtual Reality Society of Japan. Tokyo (Japan), p. 83–88.

26. ABABSA, F., ROUSSEL, D., MALLEM, M. et DIDIER, J.-Y. (2002). « 2D/3D automatic matching technique for 3D recovering of free form objects ». In : *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. T. 2. IEEE, p. 430–433.

Workshop internationaux et posters avec comités de lecture

1. ARCILE, J., **DIDIER, J.-Y.**, KLAUDEL, H., DEVILLERS, R. R. et RATAJ, A. (2015). « Indefinite waitings in MIRELA systems ». In : *Proceedings 4th International Workshop on Engineering Safety and Security Systems, ESSS 2015, Oslo, Norway, June 22, 2015*. P. 5–18. DOI : 10.4204/EPTCS.184.1. URL : http://dx.doi.org/10.4204/EPTCS.184.1.

2. CHOUITEN, M., **DIDIER, J.-Y.** et MALLEM, M. (2013). « A Framework for Service Based Composite Augmented Reality Applications ». In : *Ubiquitous Virtual Reality (ISUVR), 2013 International Symposium on*. IEEE, p. 19–22.

3. **DIDIER, J.-Y.**, KLAUDEL, H. et MOINE, M. (2013). « An Improved Approach to Build Safer Mixed Reality Systems by Analysing Time Constraints ». In : *Joint Virtual Reality Conference (JVRC) 2013 Poster, Demo and Industrial Track*, p. 87–90.

4. CHOUITEN, M., **DIDIER, J.-Y.** et MALLEM, M. (2012). « Distributed Augmented Reality Systems : How Much Performance is Enough ? » In : *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, p. 337–342. DOI : 10.1109/ICMEW.2012.64.
5. **DIDIER, J.-Y.**, CHOUITEN, M., MALLEM, M. et OTMANE, S. (2012). « ARCS : A framework with extended software integration capabilities to build Augmented Reality applications ». In : *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on*, p. 60–67. DOI : 10.1109/SEARIS.2012.6231170.
6. ZENDJEBIL, I., ABABSA, F.-E., **DIDIER, J.-Y.** et MALLEM, M. (2011). « Large Scale Localization - For Mobile Outdoor Augmented Reality Applications ». In : *International Conference on Computer Vision Theory and Applications (VISAPP 2011)*. Vilamoura, Algarve, Portugal, p. 492–501.
7. ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.** et MALLEM, M. (2010). « A GPS-IMU-camera modelization and calibration for 3D localization dedicated to outdoor mobile applications ». In : *Control Automation and Systems (ICCAS), 2010 International Conference on*. IEEE, p. 1580–1585.
8. BENJAMIN, B., **JEAN-YVES, D.** et ABDERRAHMANE, K. (2008). « Interacting with a virtual tool on a real object ». In : *Proceedings of 1st Mediterranean Conference on Intelligent Systems and Automation (CISA 2008)*. T. 1019. 1. AIP Conference Proceedings, p. 563–564. DOI : 10.1063/1.2953046.
9. ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.** et MALLEM, M. (2008a). « Hybrid Localization System for Mobile Outdoor Augmented Reality Applications ». In : *The First International Workshops on Image Processing Theory, Tools and Applications*. Sousse, Tunisia : IEEE.
10. — (2008b). « Toward an Inertial/Vision Sensor Calibration for Outdoor Augmented Reality Applications ». In : *the 2nd International Workshop on Mobile Geospatial Augmented Reality (REGARD)*. Sous la dir. de SPRINGER. Springer. Québec, Canada : Springer.
11. **DIDIER, J.-Y.**, ROUSSEL, D., MALLEM, M., OTMANE, S., NAUDET, S., PHAM, Q.-C., BOURGEOIS, S., MÉGARD, C., LEROUX, C. et HOCQUARD, A. (2005). « AMRA : Augmented Reality assistance in train maintenance tasks ». In : *Workshop on Industrial Augmented Reality (ISMAR'05)*. Vienna (Austria).

Communications nationales sans comités de lecture

1. AJILI, I., MALLEM, M. et **DIDIER, J.-Y.** (2016). « Gesture recognition for robot teleoperation ». In : *11ème journées de l'AFRV*.
2. RUKUBAYIHUNGA, A., **DIDIER, J.-Y.**, CHOUITEN, M. et OTMANE, S. (2014). « An overview of occlusion issues handling in Augmented Reality ». In : *8èmes journées de l'Association française de réalité virtuelle, augmentée, mixte et d'interaction 3D*, p. 83–95.
3. **DIDIER, J.-Y.** et MALLEM, M. (2012). « Automated Concurrent Access Management for Components of Augmented Reality Applications ». In : *7èmes journées de l'AFRV*. Strasbourg, France, p. 13–19.
4. ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.**, VAIRON, J., FRAUCIEL, L., HACHET, M., GUITTON, P. et DELMONT, R. (2007). « Réalité augmentée en exterieur : enjeux et état de l'art ». In : *2èmes journées de l'AFRV*. Marseille, France.
5. **DIDIER, J.-Y.**, OTMANE, S., ROUSSEL, D. et MALLEM, M. (2003a). « Architecture Logicielle Modulaire adaptée au Recalage Dynamique dans un Système de Réalité Augmentée en Vision Directe ». In : *17ème journée des Jeunes Chercheurs en Robotique*, p. 102–106.

6. **DIDIER, J.-Y.**, ROUSSEL, D., OTMANE, S. et MALLEM, M. (2003b). *Architecture informatique dédiée à l'estimation et à la prédiction du point de vue d'un opérateur dans un système de réalité augmentée multicapteurs en vision directe*. Communication au Journées RA-Temps Réel du GdrISIS.

Rapports et livrables de recherche

1. ZENDJEBIL, I., ABABSA, F.-E., **DIDIER, J.-Y.**, HACHET, M., GUITTON, P., DELMONT, R., FRAUCIEL, L., VAIRON, J. et al. (2007a). *Livrable projet ANR RAXENV, SP1. 1 La Réalité Augmentée en Extérieur-Etat de l'art*. Rapp. tech.
2. — (2007b). *Livrable projet ANR RAXENV, SP1. 3 Cahier des charges du système matériel*. Rapp. tech.
3. **DIDIER, J.-Y.** (2004). *Livrable projet RNTL AMRA, SP5. 4 Moteur multi-média pour la réalité augmentée – Spécifications et implémentation*. Rapp. tech.
4. **DIDIER, J.-Y.** et LAURIE, A. (2004). *Livrable projet RNTL AMRA, SP3. 4 Proposition d'un système de balisage XML décrivant une procédure de maintenance industrielle*. Rapp. tech.
5. **DIDIER, J.-Y.**, MALLEM, M., ROUSSEL, D. et OTMANE, S. (2003a). *Livrable projet RNTL AMRA, SP5. 1 Etude préliminaire sur les lunettes de visualisation*. Rapp. tech.
6. — (2003b). *Livrable projet RNTL AMRA, SP5. 2 Simulation d'une application de réalité augmentée en utilisant des capteurs de localisation*. Rapp. tech.

Mémoires

1. **DIDIER, J.-Y.** (2018). « Architectures pour les logiciels de réalité augmentée : des concepts aux applications ». Habilitation. Evry : Université d'Evry-Val d'Essonne.
2. — (2005). « Contributions à la dextérité d'un système de réalité augmentée mobile appliqué à la maintenance industrielle ». Thèse de doct. Evry : Université d'Evry-Val d'Essonne.
3. — (2002). « Recalage dynamique dans un système de réalité augmentée en vision indirecte ». Mém.de mast. Université d'Evry-Val d'Essone.

1.2.3.1.1 Logiciels

ARCS – Augmented Reality Component System : Application directe des mes travaux de recherche, c'est une collection d'outils et de bibliothèques logicielles destiné aux personnes qui souhaitent écrire des logiciels de réalité augmentée. Le premier prototype a vu le jour en 2004 et la première version finalisée est distribuée sous licence GNU–GPL (*GNU General Public Licence*) depuis 2011. Écrit à la base en C++, une autre version existe, écrite en Javascript et exécutable dans le contexte du navigateur.
Page du projet ARCS : http://arcs.ibisc.univ-evry.fr

MIRELA – MIxed REality LAnguage : MIRELA est un langage de haut niveau permettant de décrire les contraintes temporelles associées aux applications de réalité mixte afin de les analyser pour déterminer, avant réalisation, si un éventuel dysfonctionnement est à prévoir ou non afin de pouvoir le corriger en amont. Ce travail fait appel à une modélisation à l'aide d'automates temporisés et à des outils de vérification de modèle tels qu'UPPAAL ou PRISM. Les développements associés à MIRELA ne sont que partiellement rendus publics. Le point d'entrée est à l'adresse suivante : https://forge.ibisc.univ-evry.fr/groups/mirela.

Extension bibtex pour Mediawiki : Il s'agit d'une extension développée en 2006 permettant de stocker, de manipuler et d'effectuer le rendu de références bibliographiques au format bibtex au sein du moteur *Mediawiki* à destination des sites d'édition de contenu collaboratifs appelés wikis. Ce moteur est notoirement utilisé par le site *Wikipedia*. L'extension, sous une licence appartenant au domaine publique, est maintenue depuis 2012 par Simon L. Garfinkel, *associate professor* à la *Naval Postgraduate School* à Arlington en Virginie. La page actuelle de l'extension est la suivante : http://simson.net/page/Mediawiki_Bibtex_Extension

1.2.3.2 Rayonnement

1.2.3.2.1 Organisation de conférences

— **2018** : Membre du comité local d'organisation des journées de la réalité virtuelle (J•RV2018), co-portées par l'Association Française de Réalité Virtuelle (AFRV) et le GdR IG-RV ;

— **2012, puis 2013** : Co-chair de la session concernant la réalité augmentée avec Samir Otmane (PU) à la conférence VRIC – *Virtual Reality International Conference* ;

— **2008, puis 2009** : Co-organisateur de la conférence CISA – *Conference on Intelligent Systems and Automation*. J'ai été plus particulièrement en charge de l'infrastructure internet (site de la conférence et gestion électronique des soumissions d'articles) ;

— **2006** : Membre volontaire pour aider au déroulement de la conférence Eurohaptics 2006.

1.2.3.2.2 Keynote

Présentation d'une *keynote* pour la conférence ICTIA (1^{st} International Conference on Information and Communication Technologies Innovations and Applications) le 7 mars 2014 à l'ISET de Sousse sur le thème « Augmented Reality : Issues, Trends and Challenges ».

1.2.3.2.3 Relecture d'articles

Je suis régulièrement sollicité en tant que relecteur dans des conférences internationales :

1. Humanoids – *International Conference on Humanoid Robots*, IEEE, 2018 ;
2. IHCI – *International Conference on Intelligent Human Computer Interaction*, 2018 ;
3. VRIC – *Virtual Reality International Conference*, 2014, 2015, 2016, 2017 et 2018 ;
4. ICATS – *International Conference on Automatic, Telecommunication and Signals*, 2015 ;
5. ISMAR – *International Symposium on Mixed and Augmented Reality*, IEEE, en 2011, 2012, 2013 et 2014 ;
6. ICNSC – *International Conference on Networking, Sensing and Control*, IEEE, en 2013 ;
7. PNSE – *Petri Nets and Software Engineering*, 2012 ;
8. ICCAS – *International Conference on Control Automation and Systems*, en 2010 ;
9. ICRA – *International Conference on Robotics and Automation*, IEEE, en 2010 et 2016 ;
10. ICIRA – *International Conference on Intelligent Robotics and Application*, en 2010 ;
11. CISA – *mediterranean Conference on Intelligent Systems and Automation*, en 2008 et 2009 ;
12. IROS – *international conference on Intelligent RObots and Systems*,IEEE , en 2009 ;
13. CASE – *Conference on Automation Science and Engineering*, IEEE, en 2008 ;
14. IPTA – *International Conference on Image Processing Theory, Tools and Applications*, IEEE, en 2010 et en 2008 (l'évènement était à l'époque un *workshop*) ;

15. RO-MAN – *International Symposium on Robot and Human Interactive Communication*, IEEE, en 2008 ;
16. WHC – *World Haptics Conference*, en 2007 ;

Quelques relectures ponctuelles ont aussi été effectuées pour des revues :

1. IJCV – *Internation Journal of Computer Vision*, Springer, en 2018 ;
2. TSI – *Techniques et sciences informatiques*, Lavoisier en 2010.

1.2.3.2.4 Mobilité Erasmus

Durant le mois de septembre 2011, j'ai eu l'occasion d'effectuer une mobilité dans le cadre du projet Erasmus. Je suis ainsi allé durant quelques jours en Pologne, à l'Université de Technologie de Poznan. Cela a été l'occasion de présenter en séminaire mes travaux de recherche en vue de lancer de nouvelles collaborations et d'effectuer quelques heures d'enseignement à des étudiants de Master 1 sur les techniques pour fiabiliser les logiciels embarqués dans le secteur aéronautique.

1.2.3.2.5 Vulgarisation

Lors des éditions 2017 et 2018 de la *Fête de la science*, j'ai animé un atelier autour de la réalité augmentée et de son utilité dans l'exploitation des réseaux d'eau potable. L'application développée sur tablette utilise le logiciel ARCS.js et les données aimablement fournies par la communauté d'agglomération de Grand Paris Sud.

— Site de l'édition 2018 de la Fête de la Science à l'Université d'Evry : http://www.fetedelascience.univ-evry.fr/

— Lien vers la description des ateliers de l'édition 2017 : https://sortir.grandparissud.fr/evenements/stands

1.3 Activités d'enseignement

Voici une liste, non exhaustive, des différents types d'enseignement que j'ai pu dispenser depuis mon entrée en fonction en tant que Maître de conférences. La table 1.4 donne un récapitulatif année par année des heures effectuées. Ce volume d'heure a fortement augmenté depuis 2010-2011 en raisons de plusieurs facteurs allant de la sous-dotation en postes de l'Université à la croissance des effectifs étudiant (l'UFR S&T a absorbé une augmentation de plus de 40% sur la période) et d'une offre de formation renouvelée et accrue pour la période 2014-2019.

Année	CM	TD	TP	heq TD
2006 – 2007	28	58	139	192
2007 – 2008	66	64	92	224
2008 – 2009	66	48	113	222
2010 – 2011	91	126	107	369
2011 – 2012	89	141	107	381
2013 – 2014	65	168	74	315
2014 – 2015	74	121	83	315
2015 – 2016	78	120	229	437
2016 – 2017	66	161	227	487
2017 – 2018	58	204	173	465

TABLE 1.4 – Récapitulatif des heures d'enseignement effectuées

La table 1.5 indique les enseignements ou portions d'enseignement pour lesquels j'ai eu la responsabilité de la coordination du contenu pédagogique et des enseignants. Ces derniers vont de la Licence 2 au Master 2.

Pour s'y retrouver, voici les significations des différents acronymes des formations :

A2I Automatique et Informatique Industrielle ;

E3A Electronique, Energie Electrique et Automatique ;

ENSIIE Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise ;

GEII Génie Electrique et Informatique Industrielle ;

GSI Génie des Systèmes Industriels ;

IN Industries Numériques ;

ISAS Ingénierie des Systèmes Aéronautiques et Spatiaux ;

ISC Ingénierie des Systèmes Complexes ;

ISVD Intégration Systèmes Voix-Données ;

MIAW Métiers de l'Industrie, Applications Web (IRSSI lors de l'accréditation précédente) ;

PC Physique Chimie ;

RSTE Réseaux et Systèmes de Télécommunication pour l'Entreprise (ancienne dénomination d'ISVD) ;

SAAS *Smart Aerospace and Autonomous Systems* ;

SPI Sciences Pour l'Ingénieur.

Enseignement	Filière	CM	TD	TP	Années
Algorithmique - programmation	LP A2I	9	8.5	12	2008 - 2010
	L2 PC-SPI	0	15	15	2011 - 2013
Système d'exploitation	L3 SPI GI	8	8	12	2014 - 2016
Introduction aux réseaux	L3 GSI/GEII	6	6	8	2006 - 2008
	LP A2I	3	3	8	2010 - 2013
	LP RSTE/ISVD	16	12	12	2010 - 2015
Nouveaux standards pour le web (HTML5)	LP IRSII / MIAW	0	20	0	2014 - 2019
Ingénierie numérique et collaborative	M1 ISC	10	12	12	2015 - 2019
Ingéniere des systèmes	M1 ISC	10	12	12	2015 - 2019
Programmation orientée objet	M1 ISC	8	10	12	2015 - 2019
Réseaux, couches hautes	M1 GEII	6	4,5	9	2007 - 2012
Applications distribuées	M1 GEII	6	6	4	2012 - 2015
Synthèse d'images	M1 GEII	4	6	12	2009 - 2015
	M1 E3A	4	6	8	2015 - 2019
Génie logiciel	M2 GEII	20	0	0	2008 - 2015
	M2 IN/ISAS	8	10	12	2015 - 2019
Modélisation UML / Aeronautical software	M2 GSI	14	0	0	2008 - 2015
	M2 SAAS	10	12	8	2015 - 2019
Modélisation géométrique de robot	ENSIIE 2	0	0	21	2006 - 2012
Réalité augmentée	ENSIIE 2	3.5	0	24.5	2012 - 2019
Langage objet avancé	ENSIIE 2	5.25	0	13.5	2012 - 2019
Informatique graphique	ENSIIE 2	5.25	0	7	2012 - 2019

TABLE 1.5 – Aperçu des enseignements menés depuis la prise de poste

1.3.1 Au niveau Licence

Système d'exploitation

Ce cours est donné en troisième année de licence (SPI parcours Génie Informatique). Il permet de se familiariser avec les fonctionnalités principales d'un système d'exploitation qui sert de couche d'abstraction vis à vis du matériel en fournissant des services aux applications tels que les systèmes de fichiers, la gestion des processus, la gestion des utilisateurs. Cette matière est aussi l'occasion pour les étudiants de se familiariser avec le système d'exploitation GNU/Linux et de faire leurs premières armes avec les scripts *shell*.

Introduction aux réseaux

Ce cours est dispensé en troisième année de licence (filière GEII et GSI et Licence Professionnelle Réseaux et Télécommunications, spécialité Réseaux et Sécurité pour les Télécommunications dans l'Entreprise, devenue depuis Intégration Systèmes Voix-Données). Il démarre avec la description de la norme OSI pour décrire les diverses caractéristiques des réseaux. Les topologies des réseaux et les techniques de communication physique entre les machines sont abordées. Une étude des réseaux Ethernet et de la suite de protocoles IP est effectuée. Enfin, ce module se termine par la présentation du système de résolution des noms de domaine (DNS) au niveau de la couche applicative.

Algorithmique - Programmation

Cette matière s'adresse aux étudiants de la Licence Professionnelle Automatique et Informatique Industrielle (A2I) ainsi qu'aux étudiants de L2 PC-SPI. Ce cours définit ce qu'est un programme ainsi que ses constituants de base. L'accent est ensuite mis sur un pseudo-langage permettant de décrire un programme quelque soit le support choisi pour l'écrire par la suite. L'objectif est de familiariser les étudiants avec la logique de programmation. La mise en œuvre au travers des TPs emploie le langage C pour réaliser des programmes.

Nouveaux standards pour le Web

A destination des étudiants de Licence Professionnelle IRSII (Intégrateur de Réseaux et de Services Intranet-Internet, devenue depuis Métiers de l'Internet : Applications Web), cette matière donne les clés pour comprendre le futur standard HTML5 en cours d'élaboration. Sont explorées : la nouvelle syntaxe proposée par HTML5 ainsi que les APIs qui gravitent autour du nouveau standard : géolocalisation, stockage local du côté navigateur, base de données, etc.

1.3.2 Au niveau Master

Génie logiciel

Le génie logiciel désigne l'ensemble de méthodes, outils, techniques et activités participant à l'élaboration des logiciels. Ce cours, dispensé aux étudiants de M2 GEII puis des M2 Industries Numériques (IN) et Ingénierie des Systèmes Aéronautiques et Spatiaux (ISAS) aborde :

— Les cycles de vie des logiciels (de la conception au déploiement) ainsi que les méthodologies les accompagnant ;

— La norme UML2.0 (*Unified Modeling Language*) : un outil destiné principalement à la programmation orienté objet qui formalise l'analyse et la conception de systèmes informatiques ;

— Les patrons de conception et les architectures logicielles qui fournissent des solutions classiques à des problèmes récurrents de conception.

Modélisation UML / Aeronautical software

UML est un langage de modélisation des différents aspects concernant les systèmes d'informations et, de manière plus large, les systèmes informatiques. Chacun des ces aspects se traduit, dans UML, par un diagramme. Ce cours, s'adressant au élèves de dernière année de Master Pro (filière GSI), balaie dans un premier temps les notations associées aux différents diagrammes puis, dans un deuxième temps, les techniques de construction de ces derniers sont présentées. Cet enseignement est également dispensé en anglais dans le Master européen SAAS (*Smart Aerospace & Autonomous Systems*) au sein du module *Aeronautical Software*. Ce dernier ajoute une spécificité par rapport aux exigences métier spécifique au secteur de l'aéronautique.

Programmation orientée objet

Cette série de Travaux Pratiques aborde le paradigme de la programmation orientée objets. Le langage de programmation employé est Java. Les étudiants de Master 1 ISC apprennent également à concevoir et agencer des interfaces graphiques dans ce langage.

Réseaux (couches hautes)

L'objectif de ce module est d'étudier le fonctionnement des couches hautes concernant les réseaux informatiques pour les étudiant du M1 GEII qui souhaitent se spécialiser dans les réseaux et les télécommunications. Ce cours est articulé en trois axes :

— Fonctionnement général des réseaux informatiques : étude approfondie de la pile de protocoles TCP/IP ;

— Créations d'application distribuées dont le fonctionnement, par essence, nécessite un réseau. Protocoles abordés : RPC, CORBA, Java-RMI ;

— Sensibilisation à la sécurité des réseaux : études des attaques et des parades associées tant au niveau de la programmation des applications que de l'administration des réseaux.

Applications distribuées

Dispensé dans le cadre du parcours RMR (Réalité Mixte et Réseau) des étudiants de M1 GEII à l'Université d'Evry, ce cours s'intéresse aux applications distribuées (appelées aussi application réparties). Les étudiants découvre le type de répartition qui peut être fait : distribution des données ou des calculs au travers d'un réseau puis voient quelles utilisations concrètes en sont faites. Ce cours approfondit également les fonctionnalités des intergiciels tels que CORBA, RPC ou Java-RMI. L'accent est mis sur les difficultés de mise en oeuvre : séralisation/désérialisation, système de nommage cohérent, problèmes de sécurité.

Mathématiques pour la synthèse d'images et manipulation de graphes de scènes

Également dispensé dans le cadre du parcours RMR, ce cours présente aux étudiants quelques fondements mathématiques pour la synthèse d'image et comment cela est mis en application dans les cartes graphiques et pour programmer et interagir avec des scènes 3D. Sont ainsi présentés : les coordonnées homogènes, les différentes formalisations des rotations dans l'espace (angles de Cardan, d'Euler, quaternions et matrices de rotations), les modèles d'illumination, etc. De plus, afin de structurer tout cela, beaucoup d'API modernes de programmation 3D se basent sur la notion de graphes de scènes. Les étudiants sont amenés à mettre ces concepts en application en manipulant la bibliothèque OpenInventor.

Ingénierie numérique et collaborative

Dispensé aux étudiants du Master 1 Ingénierie des Systèmes Complexes, ce module a pour objectif de présenter les concepts, outils et méthodologies de l'ingénierie numérique et collaborative. Sont traitées les problématiques de la modélisation des structures produit et des processus métier, du référencement, de la gestion documentaire ainsi que de la gestion de configurations et de versions. Deux types de systèmes sont en particulier étudiés : les PLM (*Product Lifecycle Management*) associés à la gestion d'un portefeuille de produit et les SCM (*Software Configuration Management*) qui permettent le développement collaboratif de logiciels.

Ingénierie des systèmes

Les étudiants de Master 1 Ingénierie des Systèmes Complexes se voient inculqués une approche système au travers d'une démarche méthodologique générale et multidisciplinaire, s'appuyant sur le langage de modélisation SysML, qui englobe l'ensemble des activités visant à concevoir, faire évoluer et vérifier un système apportant une solutions aux besoins du client tout en étant acceptable par tous. L'enseignement se partage entre acquisition de méthode et application de celle-ci sur un objet technologique complexe.

1.3.3 En école d'ingénieurs

Ces enseignements sont dispensés à l'ENSIIE (Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise) à des élèves ingénieurs en deuxième année.

Modélisation géométrique et simulation d'un robot en réalité virtuelle

Ce projet consiste à modéliser la géométrie d'un robot industriel existant afin d'écrire ensuite l'application qui va commander ce dernier en réalité virtuelle. Ce projet permet à l'étudiant de se familiariser avec les bibliothèques de programmation graphique en 3D, tout particulièrement OpenGL et OpenInventor.

Réalité augmentée

Dispensé dans l'option Vision artificielle et réalité augmentée, ce cours commence par définir ce qu'est la réalité augmentée, les fonctionnalités associées à ce type de système et les applications qui en découlent. Des problématiques techniques sont ensuite abordées telles que le suivi, l'estimation de pose, le recalage et la composition de scènes mêlant réel et virtuel. Un aperçu des différentes architectures logicielles possibles est donné. Enfin, les étudiants concrétisent les notions apprises dans le cadre de ce cours par le biais d'un projet tutoré. A cette occasion, ils utilisent la version javascript d'un *framework* de réalité augmentée qui n'est autre que celui développé au cours de mes recherches, à savoir ARCS, pour *Augmented Reality Component System*.

Langage orienté objet avancé

Cette option dans le cursus des élèves ingénieur leur permet d'approfondir leurs connaissances en langage orienté objet. Une partie sur la *Standard Template Library* est effectuée par un collègue et j'effectue une partie concernant les derniers standards C++ en date, à savoir C++11, 14 et 17. La sémantique de déplacement est introduite ainsi que les problèmes annexes : élision de copie,

interprétation des *rvalue*. Les *templates* variadiques et les fonctions lambdas introduites par le standard sont également examinées. En parallèle, je fais également une introduction au *framework* de développement Qt.

Informatique graphique

Ce cours présente le fonctionnement d'une carte graphique et le *pipeline* graphique mis en œuvre sur cette dernière. Les fondements mathématiques sont également abordés avec l'ensemble des calculs à effectuer dans les espaces projectifs à l'aide des coordonnées homogènes. Enfin, les étudiants appréhendent la notion de graphe de scène et de *shaders*.

1.4 Activités d'intérêt collectif

La figure 1.1 regroupe les principales activités d'intérêt collectif que j'ai été amené à effectuer durant ces dernières années. Les sections hachurées représentent des responsabilités adjointes, des fonctions de suppléant ou un siège dans une instance résultant d'une prise de fonction. La figure ne représente pas les activités à venir (elles seront en revanche mentionnées dans le texte). Les pages suivantes donnent la liste complète de ces activités.

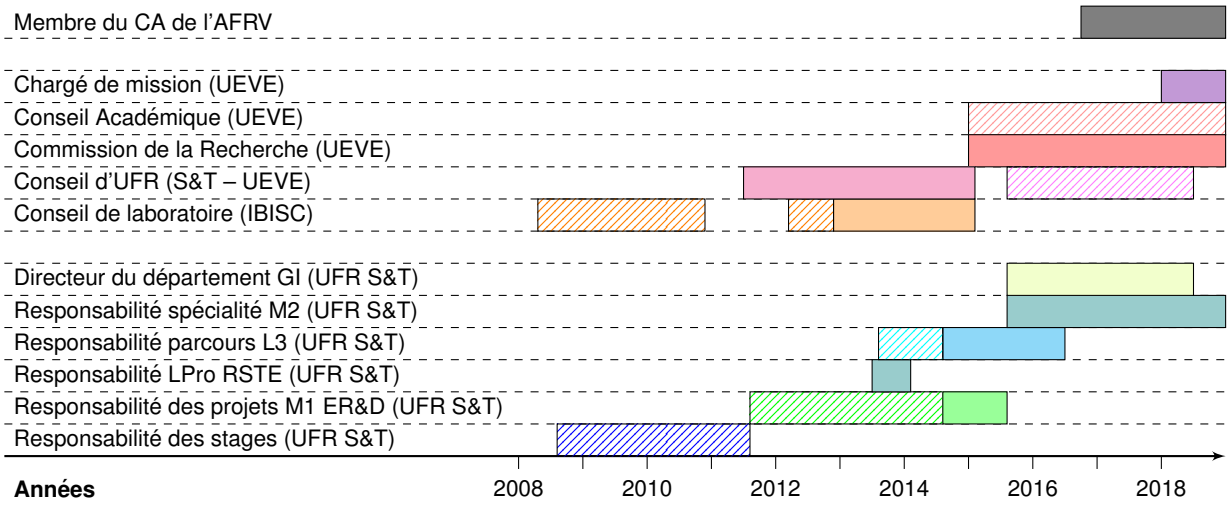


FIGURE 1.1 – Vue synthétique des principales activités d'intérêt collectif

1.4.1 Communauté et instances universitaires

Chargé de mission pour le déploiement du logiciel SysIPHE

Depuis janvier 2018, je suis en charge du déploiement, pour l'ensemble de l'université d'Evry-Val-d'Essonne, du logiciel SysIPHE (pour **Sys**tème d'**I**nformation dédié au **P**ilotage des **H**eures d'**E**nseignement). Ce dernier, développé par mes soins en 2015 lors de ma prise de fonction en tant que directeur de département pour au départ gérer les heures du département, a été étendu l'année suivante l'UFR Sciences et Technologies. Les expérimentations ayant été concluantes et l'Université recherchant à l'époque un logiciel pouvant accompagner une heure d'enseignement de sa création à sa mise en paiement a décidé de retenir ce logiciel pour gérer ces heures. 5 UFR bénéficient de son déploiement afin de traiter les heures d'enseignement effectuées par un peu plus de 2200 intervenants chaque année. Dans ce contexte, j'ai encadré un apprenti sur l'année 2017-2018 qui a ensuite été embauché par l'Université. L'adresse où le logiciel a été déployé est : https://sysiphe.univ-evry.fr.

Membre de la commission de la recherche

Je suis actuellement membre élu de la Commission de la Recherche de l'université d'Evry-Val-d'Essonne (depuis janvier 2015). Cette mandature prend fin en décembre 2018.

Membre du conseil académique

Je suis membre du Conseil Académique de l'université d'Evry-Val-d'Essonne depuis mon entrée dans la Commission de la Recherche.

Membre du conseil de laboratoire

J'ai eu l'occasion de faire partie du conseil de laboratoire (IBISC) à trois reprises depuis quelques années : les deux premières en tant que membre suppléant et la dernière en tant que membre élu. Les mandats ont été les suivants :

— décembre 2012 à janvier 2015 : membre élu ;

— février 2012 à décembre 2012 : membre suppléant ;

— avril 2008 à décembre 2010 : membre suppléant.

Membre du conseil d'UFR

De juin 2011 à janvier 2015, j'ai été membre élu du conseil de l'UFR Sciences et Technologies de l'université d'Evry-Val-d'Essonne (UFR S&T – UEVE).

Membre du conseil d'Administration de l'Université

A partir de janvier 2019, je siégerai en tant que membre élu au conseil d'administration de l'université d'Evry-val d'Essonne.

1.4.2 Vie du laboratoire et de l'équipe de recherche

Responsable adjoint de l'équipe de recherche IRA2

A partir de janvier 2019, je serai responsable adjoint de l'équipe de recherche IRA2 aux côtés du Professeur Samir Otmane. Il s'agit essentiellement d'un rôle d'animation scientifique et de vie de l'équipe. Cependant, nous comptons développer les activités de l'équipe et accroître le nombre de contrats industriels au cours du quinquennal qui débute.

Membres du conseil de laboratoire

De 2008 à 2010, puis en 2012, j'ai été suppléant au conseil du laboratoire IBISC. J'y ai été également membre élu de 2012-2015.

Administrateur de l'intranet de l'équipe IRA2

De 2005 à maintenant, je suis l'administrateur du serveur et du site internet dédié à la plate-forme Evr@. Ce serveur héberge également des services et des outils disponibles en intranet visant à faciliter l'écriture collaborative d'article scientifiques et de logiciels.

Gestion quotidienne de la plate-forme Evr@

De 2004 à 2006, j'ai été co-administrateur du parc informatique de la plate-forme (environ une dizaine de machines) ainsi que co-gestionnaire du matériel spécifique employé. La plate-forme dispose par ailleurs de son propre site web, déployé par mes soins en 2006 et encore en activité : http://evra.ibisc.univ-evry.fr.

1.4.3 Responsabilités pédagogiques

Directeur du département Génie Informatique

J'ai assuré la direction du département Génie Informatique à l'UFR S&T de septembre 2015 à août 2018. J'ai donc assuré la gestion d'une équipe pédagogique composée de 11 membres permanents pour une enveloppe annuelle d'environ 3500 heures d'enseignement. Dès que j'ai occupé ce poste, je suis également entré au conseil de l'UFR Sciences et Technologies en tant que membre invité permanent.

Responsable du parcours de Master 2 Industries Numériques

Dans le cadre du renouvellement de l'offre de formation à l'université d'Evry-Val d'Essonne, je suis le coordinateur pédagogique et responsable de la spécialité « Industries Numériques » dans la mention « Génie Industriel », maintenant « Ingénierie des Systèmes Complexes », portée par l'Université de Paris-Saclay. Cette spécialité a pour objectif de former des personnes compétentes dans le domaine du « progressive » ou « intelligent manufacturing », domaine identifié comme une technologie clé pour pour les années à venir. Dans ce cadre, la formation apporte les compétences scientifiques et technologiques nécessaires pour réussir, en l'accompagnant, la transition entre les méthodes traditionnelles de pilotage des chaînes de production et les nouvelles méthodes dites intelligentes. Cette transition peut se réaliser à deux niveaux : au niveau de la conception des chaînes de production ou au niveau de la supervision de ces dernières. Cela passe par l'automatisation et l'instrumentation de la chaîne de production avec des capteurs intelligents capables de remonter les informations en vue de les intégrer dans les systèmes de gestion des entreprises : systèmes d'information, ERP ou même PLM. Dans cette perspective, la formation apporte des connaissances et des compétences en organisation de la production, logistique, modélisation, optimisation, conception et supervision des systèmes d'informations, des systèmes informatiques distribués et des objets connectés. Le détail de la formation est indiqué à l'adresse suivante :
https://www.universite-paris-saclay.fr/fr/formation/master/m2-industries-numeriques-in

Responsable des projets d'Étude, Recherche et Développement de M1

Ce projet fait partie intégrante du tronc commun des spécialités du Master Sciences pour l'Ingénieur de l'UFR Sciences et Technologies de l'Université d'Evry-Val-d'Essonne. La responsabilité consiste à organiser et piloter ces projets découpés sur trois unités d'enseignement. Après avoir été responsable adjoint pendant quelques années (voir plus bas), j'ai pris la responsabilité principale de ces projets au mois de septembre 2014 pour une année (le renouvellement de l'offre de formation modifie les modalités du projet). Les effectifs actuels ayant augmentés, cette responsabilité implique de coordonner les interactions entre une trentaine de collègues et 140 étudiants.

Responsable du parcours Génie Informatique de Licence 3

Suite au montage du dossier pour la Licence Sciences Pour l'Ingénieur (voir plus bas), je suis devenu responsable du parcours Génie Informatique en Licence 3 en septembre 2014. Ce parcours

a pour objectif d'apporter et de consolider une base scientifique et technologique solide pour aborder sereinement une poursuite d'études en Master, intégrer une école d'ingénieur dans le domaine du génie informatique, ou de manière plus générale dans les domaines technologiques. Le parcours apporte également les compétences et connaissances nécessaires à l'exercice de fonctions techniques de l'ingénierie de conception et de réalisation de capteurs intelligents et dans leur intégration dans un système informatique. Le programme de la licence 3 est décrit sur le site internet de l'université.

Responsable par intérim du Master 1 Ingénierie des Systèmes Complexes

Après le montage du Master 1 Ingénierie des Systèmes (répertorié la première année sous le nom de Génie Industriel), le site d'Evry a connu une vacance de responsabilité. J'ai donc assuré, pour une durée d'un an (de septembre 2015 à août 2016) l'intérim pour la gestion quotidienne de cette filière. Le détail de la formation est accessible en cliquant sur ce lien.

Responsable par intérim de la Licence Professionnelle RSTE

De juillet 2013 à janvier 2014, j'ai été amené à prendre la responsabilité par intérim de la Licence Professionnelle RSTE (Réseaux et Sécurité pour les Télécommunications dans l'Entreprise), devenue depuis la licence professionnelle ISVD. Cette licence a pour spécificité d'être préparée en alternance, ce qui implique un partenariat important avec les entreprises, le Centre de Formation des Apprentis (CFA d'Evry-Val-d'Essonne) et l'UFR S&T.

Coordination pédagogique d'un parcours de licence et d'une spécialité de master

Dans le cadre de la préparation de l'offre de formation pour le plan quinquennal de la vague E (2015-2019), j'ai eu la charge de coordonner la construction de la maquette d'enseignement du parcours Génie Informatique de la licence Sciences Pour l'Ingénieur (UFR Sciences et Technologies de l'Université d'Evry-Val-d'Essonne) et de la spécialité de Master intitulée « Industries Numériques ». Cette dernière fait partie de la mention « Génie Industriel » puis « Ingénierie des Systèmes Complexes » rattachée à la *school* « Ingénierie, STI » de la nouvelle Université de Paris-Saclay.

Responsable adjoint des projets d'Étude, Recherche et Développement de M1

De septembre 2011 à juillet 2014, j'ai fait partie du comité de pilotage des projets d'Étude, Recherche et Développement de Master 1. Il s'agit d'organiser et veiller au bon déroulement des projets pluridisciplinaires proposés (qui sont sur 200h tutorées) pour une centaine d'étudiants par an. Depuis septembre 2012, je me suis plus particulièrement attaché à la conception et à la mise en oeuvre d'un système d'information pour piloter l'ensemble. Enfin, j'ai pris la responsabilité principale de ces projets en septembre 2014. Durant cette période, j'ai également développé un portail web, toujours en activité, permettant pour les enseignants de déposer des notes de centrage associées au projet. Ce site est également consulté par les étudiants pour choisir leur sujet d'étude. L'adresse de ce portail est la suivante : https://erd.ufrst.univ-evry.fr.

Responsable adjoint des stages

Durant une période de trois ans (de septembre 2008 à septembre 2011), j'ai assuré la tâche de responsable adjoint des stages. Plus particulièrement, j'ai veillé à la cohérence pédagogique et administrative de ces derniers pour la licence professionnelle A2I (Automatique et Informatique Industrielle) et le DEUST Maintenance Aéronautique (Diplôme d'Études Universitaire Scientifique et

Techniques). Cela a du se faire tout en maintenant la liaison avec la PAE de l'Université d'Evry (Plate-forme d'accès à l'emploi).

1.4.4 Expertises

1. AAP projets numériques, Paris-Saclay (2017) ;
2. Expertise thèse CIFRE 2016-0139 (2016) ;
3. IRT Jules Vernes, Appel à Projet (2015) ;
4. Expertise thèse CIFRE 2014-1503 (2014).

1.4.5 Autres tâches d'intérêt collectif

Membre du conseil d'administration de l'Association Française de Réalité Virtuelle (AFRV)

L'Association Française de Réalité Virtuelle regroupe des membres issus du monde académique et de l'industrie. Son action vise à promouvoir et développer l'utilisation de la réalité virtuelle et de structurer cette communauté afin d'augmenter sa visibilité auprès des instances nationales et européennes. Depuis novembre 2016, je siège au conseil d'administration de cette association.

Comités de sélection/recrutement

— Membre de comité de sélection : avril 2017, avril 2018 et décembre 2018 ;

— Président de commission ad-hoc pour recrutement d'ATER en section 61 : avril 2018 ;

— Président de commission ad-hoc pour recrutement d'un Maître de conférence associé (PAST) : mai 2017 ;

— Membre de commission ad-hoc pour recrutement d'ATER en section 61 : avril 2016 et avril 2017.

Deuxième partie

Synthèse des travaux de recherche

Chapitre 2

Cadre des travaux de recherche

Ce présent chapitre a pour objectif de poser le contexte des recherches effectuées. Nous commencerons tout d'abord par cerner le sujet de recherche, en faisant un tour d'horizon du type d'applications visées, à savoir la réalité augmentée et la manière dont elle est utilisée. Puis, en termes d'architecture logicielle, nous nous dégagerons les exigences fonctionnelles (le coeur de métier pour le concepteur d'applications actuelles de réalité augmentée) et les exigences non-fonctionnelles pour lesquelles nous pouvons apporter des solutions au niveau architectural. L'objectif est, nous le rappelons, de proposer des architectures logicielles qui permettent aux concepteurs d'application de se concentrer sur les verrous inhérents au domaine de la réalité augmentée et de les décharger au maximum des considérations secondaires, telles que la gestion du temps réel, de la concurrence, de la modularité ou des communications réseau par exemple. Enfin, nous dégagerons les grandes thématiques de notre plan de recherche, puis offrirons une vision globale de ce dernier.

2.1 La réalité augmentée et ses applications

2.1.1 Concepts

La paternité du terme réalité augmentée (RA), ou plus exactement de sa contre-partie anglaise *augmented reality* est attribuée à [CAUDELL et MIZELL 1992] et est apparu au début des années 1990. Désignant d'abord une technologie, l'une des premières définitions données est la suivante :

Définition 2.1

La réalité augmentée une variante de la réalité virtuelle qui utilise des casques de visualisation tête haute semi-transparents pour superposer des images générées par ordinateur à la vue réelle de l'utilisateur [REKIMOTO et NAGAO 1995].

Le concept a ensuite évolué avant de se stabiliser. Plusieurs définitions intermédiaires sont apparues, centrées sur la technologie d'abord, puis sur les besoins de l'utilisateur, comme le prouvent les deux caractérisations suivantes :

Définition 2.2

La réalité augmentée se rapporte aux périphériques d'affichage qui ajoutent des informations virtuelles à la perception sensorielle d'un utilisateur [FEINER 2002].

Définition 2.3

La réalité augmentée améliore les interactions entre l'utilisateur et son environnement réel en fournissant des capacités ou des informations additionnelles [DUBOIS et al. 2003].

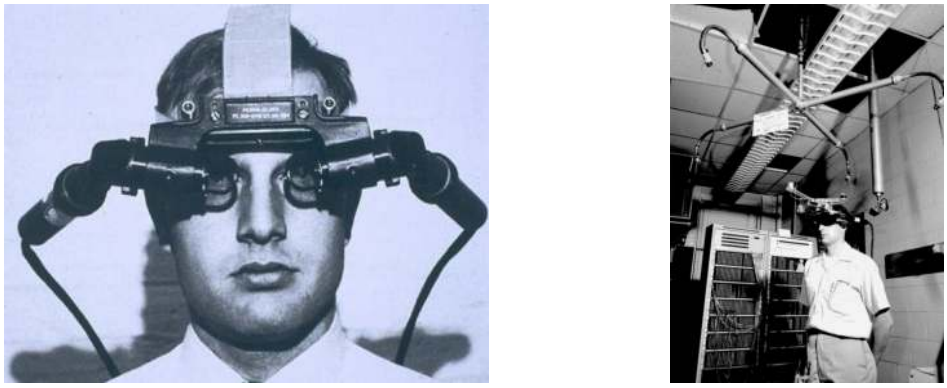


FIGURE 2.1 – *Sword of Damocles*, un des premiers prototypes de système de réalité virtuelle et augmentée [SUTHERLAND 1968]

Il est à noter que la définition 2.2 est devenue avec le temps restrictive car la réalité augmentée, à l'image de la réalité virtuelle, ne se cantonne pas qu'à l'augmentation d'un seul sens [LINDEMAN et NOMA 2007]. Nous noterons qu'à ce jour, la modalité sensorielle privilégiée majoritairement par la RA est la vision. Dans ce cas, elle permet de superposer des images de synthèse à des images prises par caméra(s) (Réalité Augmentée en Vision Indirecte - RAVI) ou directement à la vue de l'opérateur (Réalité Augmentée en Vision Directe - RAVD) suivant le type de dispositif utilisé. Néanmoins, la RA est potentiellement applicable aux autres sens de l'utilisateur. L'augmentation (voire pour certains sens la diminution) peut concerner différentes modalités sensorielles (vision, toucher, audition, odorat ou goût).

À ce jour, la définition technique la plus aboutie est sans doute celle que l'on trouvera ci-dessous :

Définition 2.4 : Réalité augmentée

La Réalité augmentée caractérise les interfaces qui [AZUMA 1997] :

1. superposent des informations virtuelles au monde réel (c'est à dire combinent objets virtuels et physiques dans le même espace d'interaction) ;
2. sont en temps-réel interactif ;
3. sont spatialisées – les objets virtuels sont recalés et interactifs dans l'espace 3D.

Nous reviendrons sur cette dernière définition, qui introduit un certain nombre d'exigences liées aux applications (au sens informatique) de réalité augmentée. Par ailleurs, la définition 2.1 établit un lien avec la réalité virtuelle. Ceci n'est pas un hasard, l'un premier dispositif reconnu de réalité virtuelle, la *Sword of Damocles* de [SUTHERLAND 1968] est en même temps le premier casque de réalité virtuelle et le premier dispositif de réalité augmentée connu. En effet, son système optique d'affichage était semi-transparent (voir figure 2.1) et permettait, de ce fait, de visualiser en même temps un environnement réel et l'espace de travail réel de l'utilisateur. Selon [MILGRAM et al. 1994], il convient de rappeler que la réalité augmentée est localisée sur une partie d'un continuum (représenté à la figure 2.2) liant environnements réel et virtuel et appelé réalité mixte.

Nous pouvons donc définir la réalité mixte comme suit :

Définition 2.5 : Réalité mixte

La réalité mixte est un ensemble de techniques permettant à l'utilisateur de percevoir la coexistance, spatiale et temporelle, de deux environnements, l'un réel et l'autre virtuel.

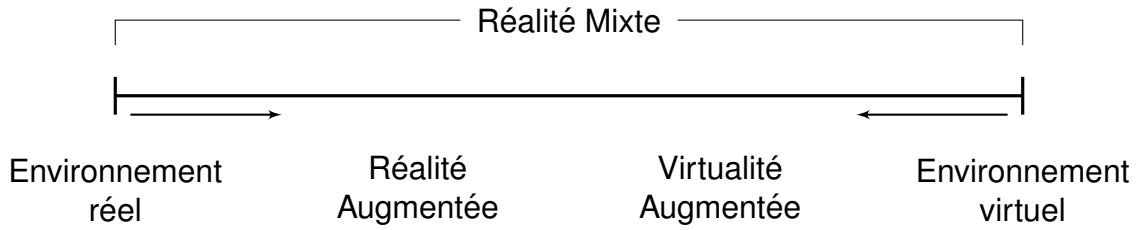


FIGURE 2.2 – Le continuum de Milgram [MILGRAM et al. 1994]

2.1.2 Système de réalité augmentée

Afin d'établir la cohérence spatiale et temporelle entre les environnements réels et virtuels, un système de réalité augmentée, tel que représenté à la figure 2.3, doit acquérir des informations sur le premier. Ces dernières sont ensuite confrontées à la connaissance (*a priori* ou constituée en ligne) dont il dispose de cet environnement et permettent de faire le lien avec les entités virtuelles. Les traitements effectués par le logiciel supportant le système sont présentés à l'utilisateur par le biais de dispositifs de restitution. À son tour, l'utilisateur peut agir sur l'interface homme-machine elle-même constituée de capteurs spécifiques ou non (le clavier et la souris font par exemple partie de cette catégorie), ce qui constitue la boucle d'interaction entre le système et l'utilisateur.

Dans cette partie, nous traiterons principalement du matériel pouvant être utilisé pour construire un système de réalité augmentée. L'objectif sera non pas d'être exhaustif, mais de présenter la variété des périphériques employés.

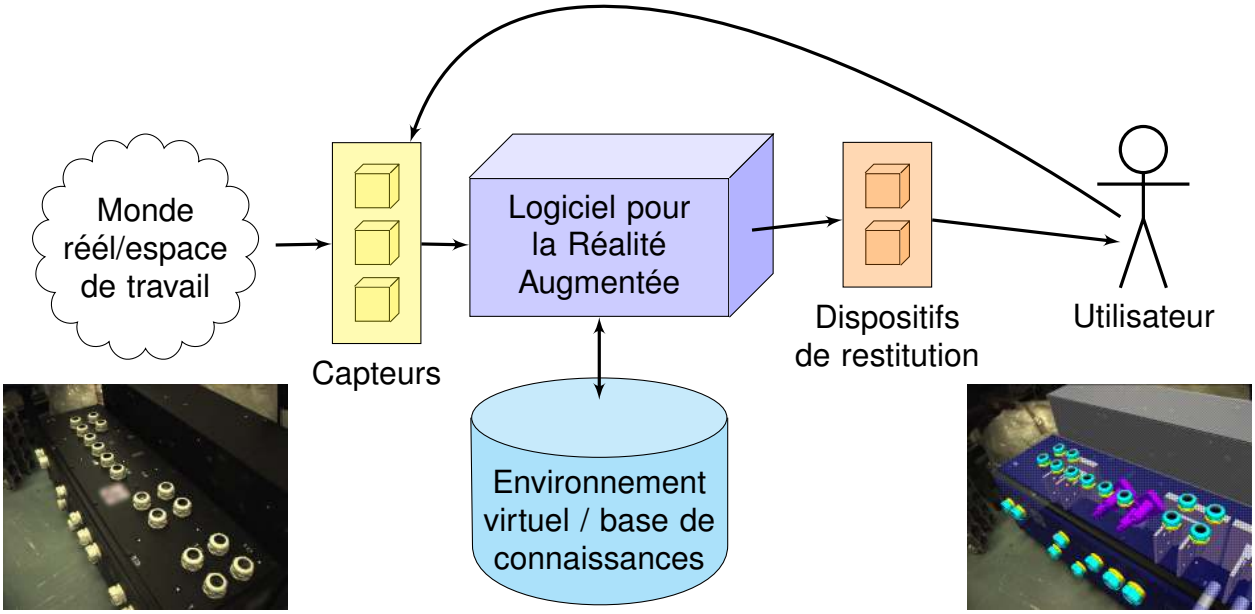


FIGURE 2.3 – Vue globale d'un système de Réalité Augmentée

2.1.2.1 Les périphériques d'entrée

Comme nous l'avons vu, un système de réalité augmentée doit présenter deux caractéristiques qui sont d'être capable de se localiser et de fournir des mécanismes d'interaction avec l'utilisateur. Par voie de conséquence, deux types de périphériques d'entrée sont à présenter.

Tout d'abord, un des problèmes classiques de la RA est de pouvoir suivre, en position et en orientation, le système ou le point de vue de l'utilisateur par rapport à son espace de travail (ce que l'on appelle estimer la pose). Ceci permet ensuite d'aligner (recaler) l'environnement virtuel sur l'espace de travail réel. Différentes technologies existent pour pouvoir y parvenir. [WELSH et FOXLIN 2002]

et [BILLINGHURST et al. 2015] ont tour à tour dressé une liste des solutions existantes. Nous les regrouperont par grandes catégories. La figure 2.4 page suivante donne quelques exemples de capteurs utilisés.

Les capteurs à positionnement local

Dans cette catégorie, nous trouverons les capteurs magnétiques, à ultrasons ou à infrarouges, dédiés à la localisation et opérant dans des espaces restreints et contrôlés. Si nous prenons le cas de la première famille de capteurs, ces derniers sont sensibles aux perturbations du champ magnétiques et n'ont qu'une portée restreinte. D'autres inconvénients accompagnent les autres familles de capteurs dont une généralement non négligeable, leur coût. Beaucoup utilisés dans les années 1990 et au début des années 2000, leur utilisation s'est peu à peu amoindrie.

Les capteurs inertiels

Ils permettent de mesurer accélération, vitesse angulaire et parfois la direction du nord magnétique. Ils sont utilisés de deux manières, soit en exploitant les données brutes pour estimer la position et l'orientation du système de réalité augmentée (ce qui est susceptible de dériver avec le temps), soit en exploitant l'orientation globale (par rapport à la direction de la gravité et au nord magnétique) donnée par ces derniers. Leur zone d'utilisation n'est pas restreinte, néanmoins ces capteurs peuvent s'avérer sensibles aux perturbations du champ magnétique.

Les systèmes de positionnement globaux

Les représentants les plus connus sont le GPS (*Global Positioning System*) ou le GPS différentiel [REITMAYR et DRUMMOND 2006] qui peuvent être couplés à d'autres technologies telles que celles désormais intégrées dans nos téléphones portable (géolocalisation GSM – *Global System for Mobile communications*, ou par proximité avec les réseaux WiFi). Ces solutions ne localisent qu'en position et sont opérationnelles à l'échelle planétaire. Toutefois, leur précision est fortement dégradée en intérieur (voir inexistante dans le cas du GPS) et leur précision est relativement faible par rapport à d'autres technologies.

Le suivi basé vision

Il combine une ou plusieurs caméras (capteur privilégié dans le cas de la RAVI car il réalise l'acquisition d'images de l'environnement réel) aux capacités simples telle que la prise d'image ou étendues lorsqu'elles sont en mesures de restituer les profondeurs auxquelles se situent les éléments qui composent la scène réelle (à l'instar de la Kinect de Microsoft et appelées également caméras RGB-D). L'utilisation conjointe de ces caméras, avec des opérations de traitement d'image permettent d'estimer la position et l'orientation du système de réalité augmentée. Les méthodes utilisées sont variées : utilisation de cibles codées (par exemple celles de l'ARToolkit [KATO et BILLINGHURST 1999]) ou de descripteurs locaux (tels que SIFT [LOWE 1999], SURF [BAY et al. 2006], ...), suivi basé modèle, méthodes de localisation et cartographie simultanée (appelées SLAM pour *Simultaneous Localization And Mapping*) développées au départ pour la robotique et introduites en réalité augmentée par [DAVISON et al. 2007], ou suivi de structure 3D en exploitant les caméras RGB-D (tel l'algorithme *KinectFusion* [NEWCOMBE et al. 2011]). En fonction des algorithmes de traitement d'image et du capteur retenus, la qualité de l'estimation du point de vue peut grandement varier. De plus, des facteurs externes tels que les variations soudaines de luminosité (en particulier en extérieur), les mouvements brusques appliqués sur le capteur, les occultations partielles ou totales de la scène peuvent dégrader la qualité de l'estimation. Nous retiendrons toutefois que, dans la cas de la RAVI,

(a) Capteur magnétique
(Fastrak – Polhemus)

(b) Centrale inertielle
(MTi – Xsens)

(c) GPS
(Pro XT – Trimble)

(d) Camera
(Ueye – IDS)

(e) Camera RGB-D
(Kinect – Microsoft)

(f) Capteur hybride caméra-centrale inertielle
(Matris – Xsens)

(g) Contrôleur sans-fil
(Flystick – ART)

(h) Interface naturelle
(LeapMotion)

(i) Prototypage d'interface
(Phidgets)

FIGURE 2.4 – Exemples de capteurs et périphériques utilisés pour la localisation et l'interaction en réalité augmentée

l'image de l'environnement réel présentée à l'utilisateur est celle captée et traitée et permet, dans des conditions normales, un recalage optimal.

Les capteurs hybrides

Ce sont en réalité des assemblages de capteurs combinés avec des méthodes de traitement du signal et des données qui permettent de compenser les défauts de l'un ou l'autre des capteurs pris individuellement. Les stratégies d'hybridation des périphériques dépendent étroitement du type d'application développée. Il est à noter que maintenant les *smartphones* embarquent simultanément au moins trois types différents de capteurs : inertiels, à positionnement global et une caméra standard voire RGB-D (à l'instar du projet Tango de Google [KERALIA et al. 2014]). À ces choix d'hybridation correspondront des stratégies différentes de fusion des données, en fonction de la nature des capteurs : coopérative, complémentaire ou compétitive [ELMENREICH 2002].

Périphériques d'interaction

Ensuite, l'utilisateur doit pouvoir interagir avec le système de réalité augmentée. Ses ordres sont transmis par le biais de périphériques conventionnels (claviers, souris) ou non (dispositifs de commande sans fil, gants de réalité virtuelle, bras à retour d'effort, etc).

De manière générale, les périphériques d'interaction qui sont utilisés dans les applications de réalité virtuelle sont aussi exploitable en réalité augmentée. [BOWMAN et al. 2004] donne, bien que la référence ne soit pas récente, une idée de l'étendue des périphériques d'interaction pouvant exis-

ter en réalité virtuelle. L'ensemble de ces périphériques est par ailleurs rangée dans une catégorie « contrôleur » par [BILLINGHURST et al. 2015]. Celle-ci peut inclure les périphériques conçus au cas par cas dans certains laboratoires ou à l'aide de kits de prototypage rapide d'interface tels que les *phidgets* [GREENBERG et FITCHETT 2001] ou les accessoires venant se greffer sur les récents nano-ordinateurs ou micro-contrôleurs de type *Raspberry Pi* ou *ARDuino*.

D'autres façons d'interagir avec un système de réalité augmentée existent. En particulier, nous pouvons penser aux interfaces naturelles, exploitant les gestes du corps en partie ou dans sa globalité (par exemple via les routines de détection de squelette de la Kinect de Microsoft ou celles permettant d'effectuer le suivi de la main par le Leap Motion) ou aux interfaces tangibles qui exploitent des objets directement présents dans la scène. Toutefois, ces interfaces particulières exploitent les capteurs ou les périphériques que nous avons déjà mentionné.

2.1.2.2 Les dispositifs de restitution

Les dispositifs de restitution vont permettre à l'utilisateur du système de percevoir simultanément les environnements virtuels et réels recalés. Pour rester totalement neutre sur les modalités sensorielles en sortie du système et à destination de l'utilisateur, les dispositifs de restitution sont des affichages (nous pouvons, à l'instar de l'affichage visuel, parler d'affichage haptique ou auditif) pilotés logiciellement par des boucles de rendu. Toutefois, il faut reconnaître que la réalité augmentée fait la part belle aux dispositifs de restitution visuel. La taxonomie de [MILGRAM et al. 1994] leur est, par ailleurs, exclusivement consacrée. Dans cette section, nous aborderons donc essentiellement cette catégorie sans pour autant rejeter les autres modalités sensorielles. La planche donnée à la figure 2.5 présente quelques dispositifs de restitution qui peuvent être utilisés dans les applications de réalité augmentée.

Les lunettes de réalité augmentée

Dispositif rêvé pour faire de la réalité augmentée, les lunettes ou casques semi-transparents libèrent les mains de l'utilisateur, affichent les entités virtuelles dans le champ de vision de l'utilisateur et permettent à ce dernier de voir directement son espace de travail. Souvent considéré comme le dispositif ultime pour la réalité augmentée, sa mise en oeuvre est néanmoins complexe. En effet, des phénomènes tels que la latence de bout en bout (c'est à dire le temps entre l'acquisition de la donnée par les capteurs et l'affichage des entités virtuelles après traitement de cette dernière) peuvent causer des décalages perceptibles par l'utilisateur entre les environnements virtuels et réels [AZUMA 1995]. Une autre critique souvent faite à l'égard de ces dispositif est le faible champ de vision qui peut être augmenté (de l'ordre d'une trentaine de degrés) en raison des possibilités des systèmes optiques actuels. Toutefois, leur utilisation connaît un regain d'intérêt depuis que deux des GAFAM [1] proposent leurs dispositifs : les *Google Glasses*, dont le projet a été suspendu un temps et qui semble être relancé, et le casque *Hololens* de Microsoft [KRESS et CUMMINGS 2017].

Les casques de réalité virtuelle

Par opposition aux lunettes de réalité augmentée, les casques de réalité virtuelle ne permettent pas de voir directement l'environnement réel. Celui-ci est capturé en vidéo par une ou plusieurs caméras, puis mixé avec les augmentations pour être restituées à l'opérateur. Par rapport aux lunettes, les applications tolèrent davantage de latence de bout en bout, celle-ci n'influant plus sur le décalage visuel entre le réel et le virtuel. Toutefois, une latence importante peut se traduire par un inconfort important pour l'utilisateur, voire être source du mal des simulateurs (en anglais *Virtual Reality Sickness*)

1. GAFAM : Google, Apple, Facebook, Amazon et Microsoft

(a) Lunettes de réalité augmentée (Hololens - Microsoft)

(b) Casque de réalité virtuelle (Oculus Rift)

(c) Ecran portable – Tablette-PC (Latitude XT2 - Dell)

(d) Ecran portable – téléphone équipé de la technologie Google Tango

(e) Ecran fixe – salle de réalité virtuelle du laboratoire IBISC

(f) Ecran fixe – abbaye de Cluny

(g) Réalité augmentée spatialisée – Volkswagen

(h) Restitution haptique (Phantom - Sensable)

(i) Restitution olfactive (Metacookie+)

FIGURE 2.5 – Exemples de dispositifs de restitution sensorielle en réalité augmentée

[HETTINGER et RICCIO 1992], comparable au mal de mer. Les casques peuvent avoir un champ visuel plus important pour visualiser l'environnement virtuel. La contrepartie est l'absence de vision périphérique pour l'utilisateur qui évolue dans l'environnement réel.

Les écrans portables à la main

Ces dispositifs de restitution visuelle sont ceux des ordinateurs, tablettes et téléphones portables. De part la nature nomade de ces systèmes, ils permettent d'emporter un écran et donc d'augmenter une portion de l'environnement réel qui aura été filmée par une caméra et mixée avant affichage. Ils présentent l'intérêt d'être peu sensible à la latence de bout en bout, et de ne pas encombrer la vision périphérique. Le champ de vision pour l'augmentation est de taille variable. Le principal inconvénient vient du fait que cet affichage ne libère pas au moins l'une des mains, ce qui peut être une gêne pour l'opérateur en fonction des tâches à accomplir. Enfin, par rapport aux dispositifs précédent, ce dernier est de nature légèrement différente, les deux premiers étant destinés principalement à un affichage égocentrique (le point de vue utilisé pour le rendu est une estimation de celui de l'utilisateur) et cette catégorie étant destinée à un affichage exocentrique (le point de vue est celui de la caméra).

Les écrans de taille plus conséquente

Il s'agit d'écrans de taille plus importante qui permettent d'augmenter une portion de l'environnement réel plus conséquente. Ces écrans de grande taille se déplacent plus difficilement et sont destinés

à être posés à un endroit particulier, l'utilisateur pouvant s'y référer à tout moment et ayant les mains libres pour accomplir ces tâches. Ces écrans peuvent être parfois montés sur des charnières mobiles pour leur permettre de pivoter afin de couvrir une portion plus importante de l'environnement réel. Dans cette catégorie, nous pouvons aussi trouver les écrans des salles de réalité virtuelle où l'augmentation se fera en fonction de ce qui est filmé par la caméra sur un site potentiellement distant.

La réalité augmentée spatialisée

Cette variante de la réalité augmentée exploite l'utilisation de vidéo-projecteurs, fixes ou mobiles pour projeter les augmentations sur l'environnement réel. L'utilisateur voit alors le mélange produit de réel et de virtuel. Les solutions de ce type garantissent que l'utilisateur aura les mains libre. En fonction du type d'installation, seule une portion fixe de l'environnement réel pourra être augmentée. Il est également à noter qu'un environnement avec une forte luminosité peut être une source de perturbation non négligeable.

Les autres dispositifs de restitution sensorielle

Comme nous l'avons écrit, la réalité augmentée, bien qu'essentiellement déclinée pour augmenter la perception visuelle, ne se borne pas qu'à ce seul sens. En effet, en complément du rendu visuel, d'autres applications utilisent des dispositifs de rendu haptique [BAYART et KHEDDAR 2006a ; BIANCHI et al. 2006], auditif [LI et al. 2004 ; LINDEMAN et al. 2007] ou olfactif [NARUMI et al. 2011] pour parachever l'effet d'immersion de l'utilisateur dans un environnement virtuel concomitant à la réalité. A ceci, se rajoutent les dispositifs de réalité virtuelle qui permettent d'augmenter les différents sens et dont on pourra trouver une liste plus détaillée dans [BOWMAN et al. 2004].

Comme nous pouvons le constater, les systèmes de réalité augmentée peuvent employer de multiples dispositifs de restitution tout comme de nombreux périphériques et capteurs différents. Cela n'est pas sans conséquence pour les parties logicielles de ces mêmes systèmes comme nous le verrons plus tard.

2.1.3 Applications

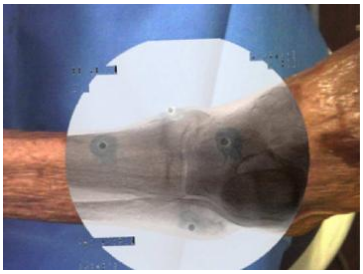
En dehors du développement de concepts fondamentaux et de la technologie, les laboratoires de recherche se sont intéressés très tôt aux finalités et aux applications possibles de la réalité augmentée. Nous en recensons ici quelques unes pour montrer l'étendue des possibilités offertes. La planche donnée à la figure 2.6 page ci-contre illustre et mentionne quelques uns des projets évoqués ci-dessous.

Divertissement

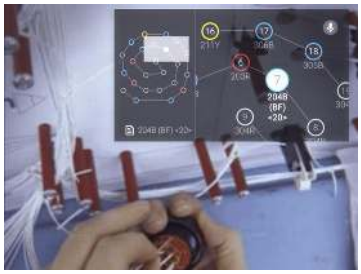
La récente sortie du jeu Pokemon Go au cours de l'été 2016 a mis la réalité augmentée au premier plan. En utilisant les capacités de géolocalisation des *smartphones* et leur caméra, le jeu place les *pokemons* dans le monde réel. Les utilisateurs peuvent alors les capturer et les dresser. Il est à noter qu'il ne s'agit pas du premier jeu utilisant la réalité augmentée. Ainsi, en 2010, Sony avait déjà sorti un titre utilisant le même principe : *Invizimals*. Si l'on remonte plus loin, [THOMAS et al. 2000] proposait déjà un prototype du jeu vidéo Quake en réalité augmentée.

L'intérêt de la réalité augmentée ne s'arrête pas au seul jeu vidéo, les retransmissions sportives utilisent de plus en plus les techniques de réalité augmentée, soit pour améliorer la compréhension du spectateur quant à l'action de jeu se déroulant sous ses yeux, soit pour incruster de la publicité directement sur les pelouses des terrains.

(a) Divertissement : capture d'écran du jeu vidéo Pokemon Go !

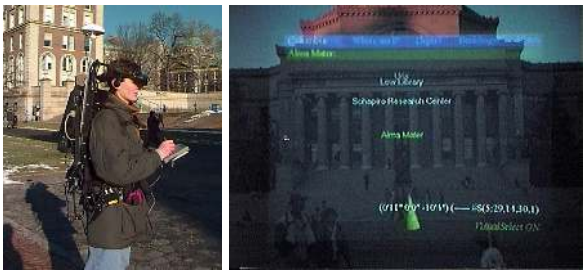


(b) Médecine : Cam-C, déployé dans les salles d'opération



(c) Industrie : mise en place de la connectique d'un avion (Boeing)

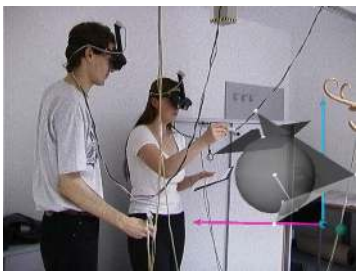
(d) Armement : écran tactique individuel, prototype TAR



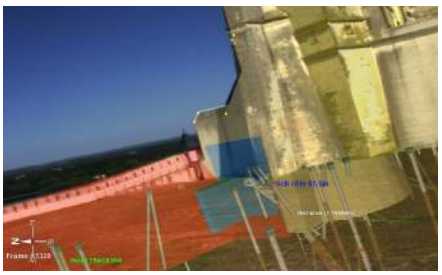
(e) Navigation : système de visite guidée MARS

(f) Valorisation du patrimoine : découverte du site d'Olympie (ARCHEOGUIDE)

(g) Commerce en ligne : catalogue IKEA en réalité augmentée



(h) Education : résolution de problème d'analyse vectorielle (Construct3D - StudierStube)



(i) Génie civil : suivi de chantier géotechnique (Raxenv)

FIGURE 2.6 – Exemples d'utilisation de la réalité augmentée

Industrie

L'industrie est sans doute l'un des premiers secteur historique d'application de la réalité augmentée. [CAUDELL et MIZELL 1992] qui introduit le terme de réalité augmentée dans la littérature scientifique, développe également un cas d'utilisation pour les techniciens en charge du montage et du branchement des câbles électriques dans les avions de la firme Boeing. Curieusement, c'est cette même tâche qui a été retenue un peu plus de 20 ans plus tard par la même entreprise pour bénéficier d'un des programmes d'utilisation des *Google Glass*, l'utilisation de ces dernières et du logiciel approprié permettant un gain de temps de l'ordre de 30% pour l'assemblage [TITA 2015].

De nombreux projets de réalité augmentée se sont intéressés à son utilisation dans le cadre de l'industrie, en particulier dans le cadre de la maintenance. Nous pouvons citer les projets STARMATE [SCHWALD et al. 2001] et AMRA [DIDIER et al. 2005b] qui restent des projets de recherche, ARVIKA (1999-2003) [FRIEDRICH et al. 2002] ainsi que ses successeurs ARTESAS (2004-2006) et AVILUS (2008-2011) qui sont des initiatives allemandes pour fédérer des consortiums d'industriels et de laboratoires autour de l'utilisation de la réalité augmentée dans l'industrie.

[OLIVEIRA et al. 2014] et [PALMARINI et al. 2018] permettent de se faire une idée de la place prise par l'industrie, et en particulier par le domaine de la maintenance, au sein des applications possibles de la réalité augmentée.

Médecine

La réalité augmentée est également utilisée en médecine. Cela peut-être dans un cadre de formation au geste médical, pour préparer une opération ou pour l'effectuer. Ainsi, [FUCHS et al. 1998] fait état d'un premier prototype utilisant la réalité augmentée pour la chirurgie laparoscopique. En effet, dans ce cadre, le chirurgien pratique de petites incisions pour passer les instruments (dont une « caméra embarquée » appelée laparoscope), puis opère en utilisant le point de vue associé à ces derniers. L'objectif du démonstrateur, qui n'a pas été déployé en salle d'opération, était de montrer que l'on pouvait offrir une autre vue, plus globale au chirurgien afin de faciliter l'opération.

Dans le cadre de la formation au geste chirurgical, [BARSOM et al. 2016] balaye une partie de la littérature consacrée à ce sujet. De manière anecdotique, et pour donner une idée du type de formation dispensée, nous pouvons citer [SIELHORST et al. 2004] qui est un simulateur permettant de former le personnel soignant à l'utilisation des forceps dans le cas d'un accouchement difficile.

Certaines des technologies mises au point, sont à présent suffisamment matures pour être déployées dans les salles d'opération [NAVAB et al. 2012] et assister le chirurgien dans l'accomplissement de ses actes de soin.

Nous noterons toutefois que de nombreuses pistes de recherche sont encore ouvertes, le problème principal étant que les modèles numériques obtenu par imagerie médicale sur un patient peuvent différer de la réalité en raison de la nature même des organes qui sont des corps déformables. Des travaux sont donc en cours pour améliorer le recalage entre le modèle virtuel l'organe réel subissant des déformations [HAOUCHINE et al. 2013].

Défense et armement

Le secteur de la défense et de l'armement est intéressé par la réalité augmentée car elle permet d'agréger les informations tactiques et de les restituer aux soldats sur le terrain. Ainsi, elle permet de donner une vue globale du contexte d'un engagement tout en réduisant la charge cognitive du soldat. Les premiers systèmes que l'on peut qualifier comme appartenant à la réalité augmentée sont sans conteste les affichages tête haute dans les cockpit.

Ces projets relevant aussi du secret militaire, il est parfois difficile de collecter des informations sur ces derniers. Toutefois, nous pouvons citer le projet BARS (pour *Battefield Augmented Reality*

System) dont l'objectif était d'aider les soldats d'infanterie à apprécier la situation dans le cas d'un engagement dans un milieu urbain [JULIER et al. 2000].

L'armée américaine a récemment fait la démonstration d'un prototype de casque équipé d'un affichage tête haute nommé TAR pour *Tactical Augmented Reality*, utilisable de jour comme de nuit et permettant d'afficher des informations tactiques contextualisée en fonction de la géolocalisation du militaire [GALLAGHER 2017].

Navigation et tourisme

Depuis l'apparition du GPS sur le marché grand public, la façon de naviguer, de conduire ou de se déplacer en environnement inconnu a été bouleversée. Ces appareils, couplés à des logiciels mettant en relation la position de l'utilisateur avec des données géolocalisées permet l'affichage contextuel de cartes, de points d'intérêts avoisinants et d'indiquer les prochaines directions à suivre.

De tels dispositifs ont aussi été proposés en réalité augmentée. Parmi ces derniers, nous pouvons citer MARS, pour *Mobile Augmented Reality System* [FEINER et al. 1997], qui est un système « portable » de réalité augmentée et qui permettait de réaliser une visite interactive du campus de l'Université de Columbia.

Plus récent et moins encombrant car tenant dans un *smartphone*, nous pouvons citer l'application Wikitude qui a démarré en mettant en évidence les points d'intérêts géolocalisés dans l'encyclopédie en ligne Wikipedia se situant à proximité de l'utilisateur, puis s'est transformé en kit de développement. L'un des produits phares actuels réalisé est un GPS en réalité augmentée qui superpose les directions à prendre à une vue réelle de l'environnement [*Wikitude* 2017].

Valorisation du patrimoine

De part la possibilité de voir ce qui est caché ou ce qui a existé à une autre époque et de le superposer à l'environnement directement visible, la réalité augmentée est aussi un outil pour la valorisation du patrimoine ou l'héritage culturel. Ainsi, le projet ARCHEOGUIDE [VLAHAKIS et al. 2002] permet aux visiteurs du site d'Olympie de découvrir les bâtiments tels qu'ils étaient construits à l'époque antique sur le support de leur choix : *smartphone*, PDA ou lunettes de réalité augmentée. D'autres projets, installés sur des sites tels que le château de Nottingham (projet Augurscope [SCHNÄDELBACH et al. 2002]) ou la cité de Heidelberg (projet Geist [HOLWEG et SCHNEIDER 2004]) permettent de découvrir et de revivre, sous une forme ou une autre, une partie du passé des lieux visités. Très régulièrement, de telles applications font l'objet de publication. L'une des dernières en date fait état d'un déploiement au sein du *Royal Ontario Museum* à Toronto. Celle-ci fait découvrir l'histoire attachée à une pièce archéologique qui est la chambre de *Kitines* [PEDERSEN et al. 2017].

Vente en ligne et commerce

La réalité augmentée trouve également des applications dans la vente en ligne ou le commerce. Elle permet, par exemple, à l'opticien Atol de faire tester en ligne à ses clients les différentes montures disponibles dans son catalogue [*Atol les Opticiens lance l'application pour essayer ses lunettes grâce à la réalité augmentée* 2011]. De même, à partir de l'année 2014, le catalogue IKEA, utilisé conjointement avec une application sur tablette ou *smartphone*, permet d'aménager en réalité augmentée son habitation avant de passer commande [TWEEDIE 2013]. Enfin, un certain nombre d'industriels de l'automobile commencent à utiliser la réalité augmentée pour permettre au client de personnaliser les voitures directement chez le concessionnaire et à taille réelle sans que ce dernier ait à posséder toutes les configurations possibles. Ce système testé par Renault et Diotasoft [LUCAS 2013] est suscite aussi l'intérêt d'Audi et de Jaguar.

Education

En tant que media particulier, la réalité augmentée permet d'expérimenter de nouvelles manières d'apprendre ou d'appréhender en trois dimensions des problèmes qui n'étaient qu'exprimés en deux dimensions. C'est dans cette optique que *Construct3D* [KAUFMANN et SCHMALSTIEG 2003], un projet réalisé avec le *framework Studierstube* a été développé. Le but de ce premier prototype éducatif était de faciliter l'appropriation spatiale sur des problèmes de géométrie. D'autres projets utilisant la réalité augmentée ont été réalisés depuis, comme l'indique [BACCA et al. 2014]. Il est à noter que l'un des effets non négligeables de l'utilisation de la réalité augmentée dans l'éducation est que la motivation des étudiants et leur implication dans le processus d'apprentissage se voit accrue.

Génie civil, construction et architecture

Les domaines de l'architecture, de la construction et du génie civil génèrent de grands volumes de données géolocalisées, en trois dimensions, qui doivent être accessibles par les différents acteurs impliqués dans ces projets. La réalité augmentée permet d'y accéder tout en les contextualisant, ce qui facilite leur compréhension dans toutes les étapes du cycle de vie d'une construction. Ainsi, la réalité augmentée permet de se rendre compte sur site de la future emprise d'un bâtiment et des modifications que cela va apporter sur l'environnement, cela permet la planification en bureau d'étude et le suivi des chantiers.

De manière plus anecdotique, la réalité augmentée a été utilisée pour montrer comment une carte peu être augmentée dans le cadre de la préparation et de la planification d'intervention en cas de crue [REITMAYR et al. 2005]. De même, elle peut être utilisée pour localiser et visualiser le passage de réseaux souterrains dans un chantier urbain afin d'éviter de percer une canalisation (projet VIDENTE [SCHALL et al. 2013]) ou dans le cadre du suivi d'un chantier géotechnique (projet RAXENV [ABABSA et al. 2012]).

[CHI et al. 2013] donne un aperçu d'un certain nombre de projets impliquant la réalité augmentée dans le domaine et discute de l'opportunité que représente cette technologie.

Comme nous avons pu le voir, la réalité augmentée peut être exploitée dans de nombreux domaines et ses applications, au service de l'utilisateur du système sont conséquentes. De plus, les technologies qui peuvent être employées sont de natures très différentes, que ce soit au niveau des capteurs ou des dispositifs de restitution. Cette variété et cette richesse des applications se reflète nécessairement dans les logiciels qui sont au cœur des systèmes de réalité augmentée. Cela va être l'objet de la partie qui suit.

2.2 Spécificité des logiciels pour la réalité augmentée

Un système de réalité augmentée nécessite une partie matérielle et une partie logicielle. Cette dernière, en vertu du domaine pour lequel elle est exploitée, renferme certaines spécificités. En premier lieu, nous ferons un état des lieux des exigences attachées aux logiciels pour la réalité augmentée, puis nous constaterons que cela motive le besoin de créer des architectures logicielles particulières. Cela nous permettra d'introduire nos travaux de recherche et de présenter leur organisation.

2.2.1 Exigences logicielles

Comme pour tout logiciel, les exigences qui y sont rattachées se subdivisent en deux catégories :

— Les **exigences fonctionnelles** qui sont les fonctions que le logiciel doit exécuter. C'est aussi ce que l'on appelle parfois les règles métier ;

— Les **exigences non-fonctionnelles** qui sont des caractéristiques ou des propriétés recherchées pour le logiciel.

Cette distinction est importante car c'est sur cette dernière que s'appuie notre travail : en tâche de fond nous travaillons sur les exigences non fonctionnelles car ce sont ces dernières qui vont fortement contraindre les architectures logicielles que nous présentons. Toutefois, pour vérifier le bien fondé et la validité d'une architecture nous devons pouvoir la déployer dans des cas d'étude réels et y incorporer les règles métiers.

La figure 2.7 synthétise en quelques mots-clés et sous la forme de carte heuristique (*mindmap*) les différentes exigences qui pèsent sur le logiciel. Elle nous servira de canevas pour la partie qui suit.

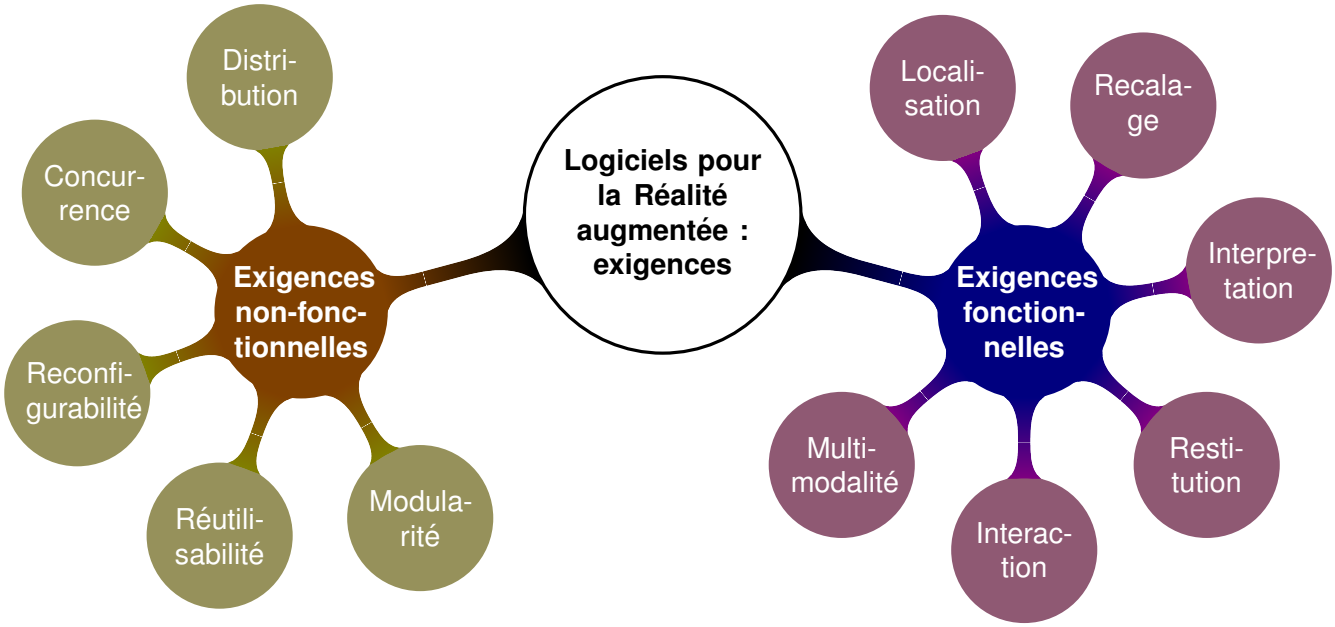


FIGURE 2.7 – Carte heuristique des exigences logicielles pour la réalité augmentée

2.2.1.1 Exigences fonctionnelles

Les exigences fonctionnelles sont, comme nous l'avons écrit, les règles métier auxquelles doivent se plier les logiciels pour la réalité augmentée. Ces règles dépendent fortement des applications rencontrées, ces dernières étant variées comme nous avons pu le constater à la section 2.1.3 page 56. Nous nous concentrerons donc sur les fonctionnalités transversales aux applications de réalité augmentée qui restent propres à ce domaine.

A cet égard, si nous repartons de la définition 2.4 page 50 de la réalité augmentée donnée par Azuma, nous pouvons extraire un certain nombre d'exigences fonctionnelles, liées en premier lieu au système de réalité augmentée, puis, par extension, à son cœur logiciel. Ces deux exigences principales sont :

— Le **recalage** qui permet d'aligner le virtuel sur le réel ;

— L'**interaction** entre l'utilisateur et le système.

De ces exigences en découlent d'autres :

— La **localisation** du système est induite car sans elle le recalage n'est pas possible ;

— La **restitution**, ou rendu des informations virtuelles est également nécessaire, sans quoi l'interaction ne peut exister. De plus, cette restitution, si nous considérons qu'elle ne se limite pas au seul sens de la vue, peut être **multi-modale**.

Enfin, la mise en relation des environnements virtuels et réels passe par une **interprétation** de la réalité (sous des formes plus ou moins abouties) faite par le système.

2.2.1.2 Exigences non-fonctionnelles

Les exigences non-fonctionnelles sont essentiellement des exigences architecturales. Elles ne portent donc pas sur le cœur des fonctionnalités de la réalité augmentée mais peuvent grandement contribuer à la supporter.

Nous avons pu constater, aux sections 2.1.2.1 page 51 et 2.1.2.2 page 54, la diversité des capteurs, des dispositifs d'interaction et de restitution employés dans les systèmes de réalité augmentée. Le domaine de la réalité augmentée étant relativement neuf, les périphériques, parfois tout d'abord des prototypes de laboratoire, évoluent. De même, les techniques et algorithmes employés sont sans cesse renouvelés. Cela implique donc de standardiser certaines parties du logiciel de manière à pouvoir les remplacer (substituer par exemple un composant gérant un capteur donné par un autre). De cette contrainte vient l'exigence de **modularité**.

De plus, un fois certains traitements développés, il peut être opportun de les conserver pour une ultérieure **réutilisation**, dans le cadre d'une évolution du logiciel ou dans le cadre du développement d'une nouvelle application.

Dans les cas impliquant plusieurs capteurs élémentaires et en fonction des stratégies d'hybridation ou de la disponibilité de l'un ou de l'autre, il peut être nécessaire de modifier, à l'exécution, une partie du chemin de traitement de données. Cela aboutit à l'exigence de **reconfigurabilité**.

Les différents capteurs utilisés peuvent fonctionner suivant des cycles d'acquisition liés à des échelles de temps variables. Ainsi, un GPS délivrera une donnée à la seconde, là où la centrale inertielle pourra en envoyer une centaine. Si à cela nous ajoutons que les logiciels de réalité augmentée sont soumis à des contraintes de performance (liées au « temps réel interactif »), il est tentant d'exploiter les performances des processeurs multi-cœur des appareils et de développer des applications faisant appel aux techniques de *multi-threading* où le logiciel est subdivisé en « fils d'exécution » opérant en parallèle et partageant la mêmoire mémoire. Ainsi, des exigences en terme de gestion de la **concurrence** et de respect des délais opérationnels apparaissent.

Enfin, en raison de la capacité de calcul et/ou de mémoire limitée de certains terminaux (tels que les téléphones portables) et des traitements parfois intensifs que demandent les applications de réalité augmentée (en particulier lorsque l'on applique certains algorithme de traitement d'image), il peut être avantageux de procéder à une **distribution** des traitements ou des données entre plusieurs noeuds de calcul, le dispositif utilisé par l'utilisateur n'étant qu'un noeud particulier dans cette infrastructure. Pour terminer sur ce point, dans le cadre de scénarios applicatifs de type travail collaboratif, la gestion de la distribution leur fournit également un support.

Nous verrons, à la fin de ce chapitre, comment les exigences détaillées ici fournissent un canevas à nos travaux de recherche. Avant toutefois de les esquisser, nous allons aborder le problème des compétences et connaissances mobilisées dans le cadre du développement de logiciels pour la réalité augmentée.

2.2.2 Compétences et connaissances mobilisées

Développer du logiciel pour la réalité augmentée de manière *ad-hoc*, mobilise un nombre important de connaissances et de compétences. Si nous faisons abstraction des règles métier spécifiques à chacune des applications développées, il n'en reste pas moins que ce socle de savoir reste conséquent et est relié au exigences mentionnées ci-dessus.

Le tutorial présenté par [PIEKARSKI 2004] à la *Linux conf Australia* en offre une démonstration intéressante puisqu'elle est suffisamment ancienne pour qu'un aperçu des problèmes techniques inhérents soit donné (dans le cadre restreint de la réalité augmentée en vision indirecte) et que s'esquissent des solutions sur étagère.

Si nous devions dresser une liste des savoir-faire mobilisés par l'écriture de tels logiciels, nous aurions deux catégories :

1. Savoir-faire en rapport avec les exigences non-fonctionnelles et l'écriture des logiciels en général :
   — programmation impérative ou mieux, orientée objets ;
   — programmation système, pour l'interfaçage des capteurs et périphériques ;
   — programmation concurrente et/ou temps réel ;
   — programmation réseau et/ou d'applications distribuées ;
   — techniques de génie logiciel en rapport avec la spécification, la conception et la validation des programmes.
2. Savoir-faire en rapport avec les exigences fonctionnelles du domaine de la réalité augmentée :
   — traitement d'image et vision par ordinateur (dans le cas où le système utilise une ou plusieurs caméras) ;
   — traitement du signal et/ou fusion de données (dans le cas du traitement bas niveau des données des capteurs simples ou hybrides) ;
   — synthèse d'images et/ou rendu multimodal ;
   — techniques d'interactions en réalité virtuelle/mixte/augmentée ;
   — géométrie spatiale, nécessaire pour le rendu mais aussi la localisation et le recalage (cela peut aller jusqu'à la connaissance et l'utilisation des techniques de cartographie dans le cadre de données géolocalisées) ;
   — apprentissage automatique si l'on veut aller jusqu'à la reconnaissance ou l'interprétation de phénomènes se déroulant dans l'environnement réel.

Une telle liste de connaissances et de compétences mobilisées illustrent à quel point créer un système de réalité augmentée à partir de rien peut s'avérer une tâche complexe. Ce constat est donc le point de départ de nos travaux.

2.3 Plan synthétique de recherche

Cette section est l'occasion d'introduire l'objectif des recherches menées et leur positionnement. Elle sera aussi l'occasion de les catégoriser au niveau thématique en fonction du type d'exigence traitée. Nous donnerons également une vision d'ensemble du plan de recherches suivies.

2.3.1 Objectif des recherches menées

Comme mentionné dans la section précédente, écrire un logiciel pour la réalité augmentée mobilise un certain nombre de compétences et de connaissances. Fort heureusement, dans le monde du logiciel, il est possible de capitaliser sur l'expérience acquise par soi-même ou par de tierces parties. C'est ce que permettent tour à tour :

— Les *bibliothèques logicielles*, qui permettent d'agréger un ensemble de fonctions ou de classes pour une utilisation ultérieure ;
— Les *frameworks* ou *cadriciels* qui ont pour vocation à accélérer les développements de logiciel dans un contexte donné en imposant un cadre architectural à la manière dont sont structurés les logiciels produits.

La seconde catégorie, dont la première n'est finalement qu'une sous-partie, est celle qui nous intéresse. En effet, un *framework* soulage le développeur d'un certain nombre de contraintes logicielles (les exigences non-fonctionnelles) pour se consacrer principalement sur l'implémentation des règles métiers propres à son domaine (les exigences fonctionnelles). Il se conforme à un modèle

architectural pré-établi. Ceci nous amène naturellement à détailler notre positionnement qui est le suivant :

Positionnement 2.1

Nos travaux de recherche s'articulent autour des architectures logicielles pour la réalité augmentée. En particulier, nous nous attachons à concevoir des modèles d'architecture répondant aux exigences non-fonctionnelles des logiciels pour la réalité augmentée. Dans le même temps, cette recherche étant appliquée, nous souhaitons confronter ces architectures à la réalité du terrain en vérifiant qu'elles sont aptes à être intégrées dans des systèmes de réalité augmentée et facilitent l'implémentation des exigences fonctionnelles.

2.3.2 Synthèse thématique

Les recherches menées s'articulent dont autour de deux thématiques :

1. la thématique principale concerne les **architectures logicielles** pour la réalité augmentée et s'attache particulièrement à fournir des outils permettant aux développeurs de s'affranchir des exigences non-fonctionnelles ;
2. une thématique secondaire concerne les **exigences fonctionnelles de la réalité augmentée** dans laquelle nous développons des applications de réalité augmentée avec les architectures mentionnées.

La table 2.1 page ci-contre trie les références que nous avons publiées suivant ces thématiques. En parallèle, la figure 2.8 page 66 réalise une chronologie comparée entre les thèses encadrées, les projets de recherche et les logiciels développés, tout en les associant aux thématiques de nos recherches.

2.3.2.1 Architectures logicielles pour la réalité augmentée

Les sous-thématiques abordées dans cette partie sont les suivantes :

1. La proposition d'une **architecture logicielle modulaire et reconfigurable** adaptée à la réalité augmentée. La solution retenue est une architecture orientée composants reposant sur une généralisation du patron de conception *observateur* pour la communication entre ces derniers ;
2. Des solutions pour que cette architecture soit transparente quand à la **distribution** et aux connections au réseau ainsi qu'une étude théorique permettant de déterminer quand une stratégie de distribution est pertinente à appliquer pour répartir les traitements ;
3. Une méthodologie et des outils théoriques et pratiques pour résoudre les problèmes des délais opérationnels, de la synchronisation et plus généralement des **contraintes temporelles** dans le cadre de logiciels faisant appel aux techniques de programmation **concurrente**. Cette dernière partie fait appel aux techniques vues et connues dans le domaine des méthodes formelles et de la conception des systèmes temps-réels. En particulier, nous nous appuyons sur le formalisme et l'analyse de réseaux d'automates temporisés.

Cette recherche étant **appliquée**, nous nous sommes attachés à implémenter les solutions proposées dans deux *frameworks* aux objectifs complémentaires :

ARCS, pour *Augmented Reality Component System*, se focalise sur les problèmes de modularité et de reconfigurabilité. Ses évolutions successives ont aussi permis de traiter la problématique de la distribution (ARCS2), jusqu'à en faire un *framework* utilisable directement dans le navigateur web en tirant partie des récentes technologies HTML-5 (arcs.js) ;

MIRELA, pour *MIxed REality LAnguage*, et son successeur MIRELA-NG, proposent un langage de haut niveau pour décrire les applications de réalité mixte ainsi que des outils permettant de traduire ce langage dans des spécifications formelles à base d'automates temporisés et d'analyser,

Thématique 1 : architectures logicielles	
Architecture modulaire et reconfigurable	DIDIER et al. 2009b ; DIDIER et al. 2006 ; DIDIER et al. 2012 ; DIDIER et al. 2003 ; DIDIER 2005
Distribution	CHOUITEN et al. 2014 ; CHOUITEN 2013 ; CHOUITEN et al. 2012b ; CHOUITEN et al. 2011 ; CHOUITEN et al. 2013 ; CHOUITEN et al. 2012a
Contraintes temporelles et concurrence	DIDIER et al. 2009a ; ARCILE et al. 2015a ; DEVILLERS et al. 2014 ; DIDIER et MALLEM 2014 ; DEVILLERS et al. 2013 ; DIDIER et al. 2008c ; DIDIER et al. 2008d ; DIDIER et al. 2008b ; ARCILE et al. 2015b ; DIDIER et al. 2013 ; DIDIER et MALLEM 2012
Thématique 2 : applications de réalité augmentée	
Localisation et recalage	MAIDI et al. 2010 ; ZENDJEBIL et al. 2010 ; DIDIER et al. 2008a ; ABABSA et al. 2008b ; ZENDJEBIL et al. 2008b ; ABABSA et al. 2008a ; MERAD et al. 2006 ; ZENDJEBIL et al. 2008a ; ZENDJEBIL et al. 2008c ; DIDIER 2002
Interprétation	RUKUBAYIHUNGA et al. 2016a ; RUKUBAYIHUNGA et al. 2016b ; RUKUBAYIHUNGA 2016 ; RUKUBAYIHUNGA et al. 2015
Interaction	BAYART et al. 2008 ; BAYART et al. 2007 ; AJILI et al. 2016 ; AJILI et al. 2017a ; AJILI et al. 2017b
Divers : sujets connexes à la réalité augmentée	
Etats de l'art	ZENDJEBIL et al. 2009 ; ZENDJEBIL et al. 2008d ; RUKUBAYIHUNGA et al. 2014 ; ZENDJEBIL et al. 2007
Autres	DIDIER et al. 2005a ; DIDIER et al. 2004 ; ABABSA et al. 2003 ; DIDIER et al. 2005b

TABLE 2.1 – Publications triées selon les thématiques du plan de recherche

avant implémentation réelle de l'application, si cette dernière ne souffre pas de dysfonctionnements liés à ses contraintes temporelles.

Ces frameworks ont ensuite été exploités dans les projets de recherche ou utilisés dans les thèses que j'ai encadrées.

2.3.2.2 Applications de réalité augmentée

Ces applications, utilisant par ailleurs les architectures proposées ou servant d'étude de cas concrètes pour mettre en évidence des problèmes fondamentaux à propos des exigences non fonctionnelles, implémentent des fonctionnalités intrinsèques à la réalité augmentées pour lesquelles nous avons proposés des solutions en particulier :

1. La **localisation** et le **recalage** avec un travail particulier sur les capteurs hybrides et les stratégies de fusion mises au point ainsi que la chaîne de traitement utilisée pour le recalage basé modèle ;
2. La détection et l'**interprétation** d'évènements particuliers dans le cadre de suivi de scénarios de maintenance industrielle ;
3. Les **interactions** homme-machine, avec des travaux sur la reconnaissance du geste pour commander les applications de réalité augmentée ainsi que des travaux sur la **multi-modalité**, en

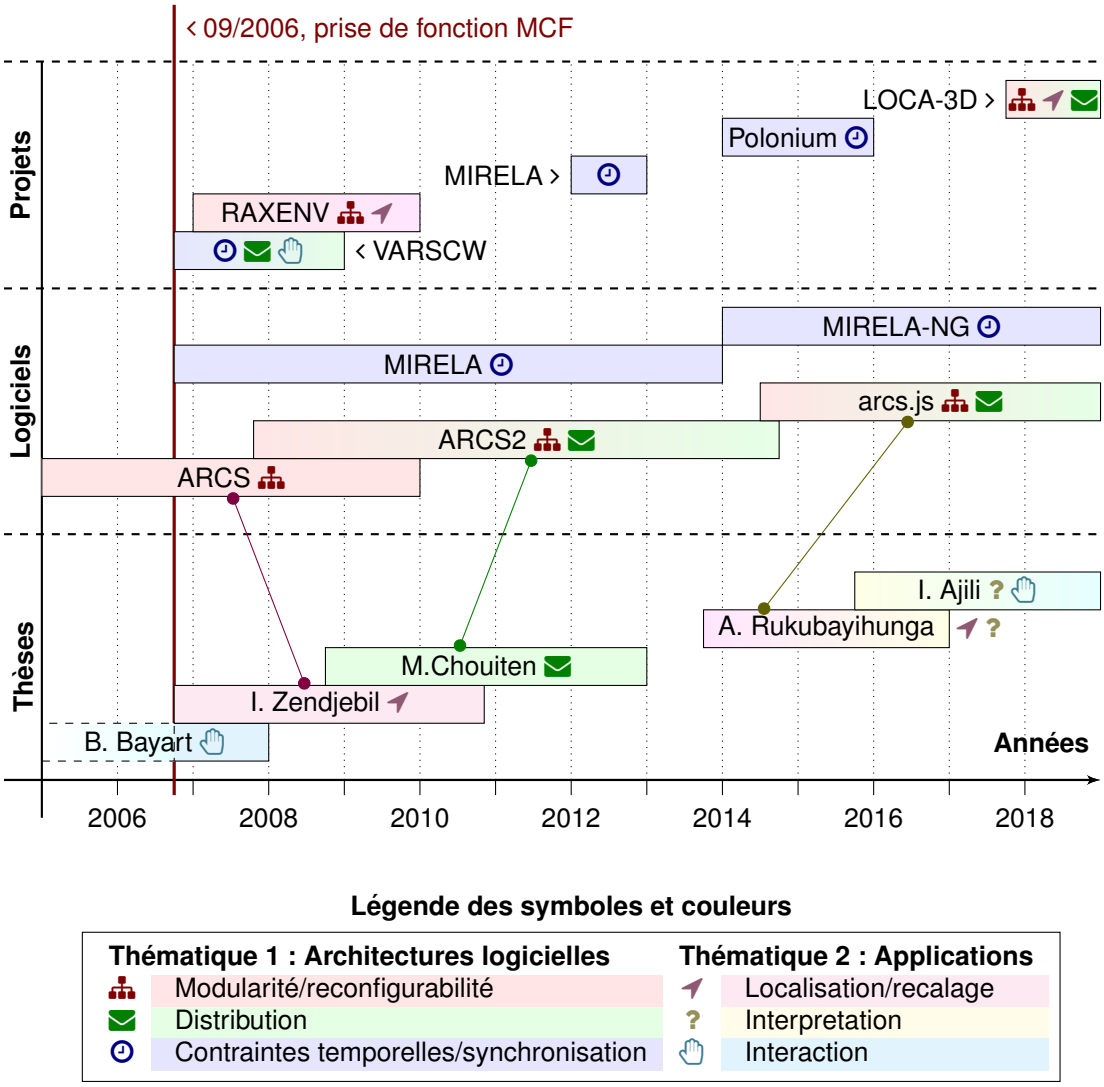


FIGURE 2.8 – Chronologie comparée des thèses encadrées, des logiciels et des projets de recherche développés

particulier la réalité augmentée haptique.

Les recherches et les résultats obtenus sur chacun de ces points seront développés dans les chapitres qui suivent. Nous commencerons donc tout d'abord par traiter les exigences non-fonctionnelles associées aux logiciels pour la réalité augmentée, puis nous aborderons les applications et comment celles-ci combinent exigences fonctionnelles et non fonctionnelles.

Chapitre 3

Architecture logicielles pour la Réalité Augmentée

Ce chapitre présente les travaux effectués dans le domaine des architectures logicielles pour la réalité augmentée, en particulier en ce qui concerne les exigences non fonctionnelles associées aux applications. Nous verrons tout d'abord comment nous avons posé les bases pour une architecture modulaire et reconfigurable dont les concepts ont été implémentés dans un *framework* appelé ARCS (pour *Augmented Reality Component System*). Ensuite, il nous a paru nécessaire d'inclure dans ces travaux d'autres exigences afin de gérer la distribution des traitements au travers d'un réseau, puis la concurrence lorsque ces applications sont pourvues de fils d'exécution parallèles (*threads*). Enfin, le dernier volet de ce chapitre introduira un second *framework* dont l'objectif est de proposer une chaîne de prototypage d'applications de réalité augmentée permettant tout d'abord de les décrire dans un langage de haut niveau, puis de les traduire dans une spécification formelle afin d'analyser le comportement de l'application avant implémentation.

3.1 Architecture modulaire et reconfigurable

3.1.1 Cadre et définitions

Une des motivations profondes de nos travaux étant la réutilisabilité, il convient de définir formellement ce qu'est une architecture logicielle.

Définition 3.1 : Architecture logicielle

L'architecture logicielle d'un programme ou d'un système informatique est la ou les structures du système, ce qui comprend les éléments logiciels, les propriétés externes visibles de ces éléments et leurs relations [BASS et al. 2003].

Ainsi, une architecture peut expliquer comment les éléments du système qui la composent sont réutilisables. Dans le domaine du génie logiciel, cela intervient à plusieurs niveaux :

— Au niveau du langage de programmation, la réutilisabilité est, par exemple, au centre des préoccupations de la programmation orientée objets ;

— Au niveau des fonctions et bibliothèques, qui mettent à disposition par le biais d'interfaces de programmation (appelées aussi API *Application Programming Interfaces*) la partie publique de leur code en vue de l'appeler dans un programme. Sur ce point, la réutilisabilité peut aller plus loin avec les composants ;

— Au niveau des concepts : à un niveau intermédiaire pour décrire comment organiser localement un ensemble de classes ou de fonctions (c'est ce que l'on appelle les patrons de conception)

ou au niveau architectural avec l'utilisation de modèle récurrents d'architectures (à ce sujet, on pourra se référer à GARLAN et SHAW 1993 et à RICHARDS 2015).

Pour lever toute ambiguïté, nous donnons ci-dessous les définitions communément acceptées pour les composants et les patrons de conception.

Définition 3.2 : Composant logiciel

Un composant logiciel est une unité de composition avec une interface spécifiée contractuellement et uniquement des dépendances contextuelles explicites. Il peut être déployé indépendamment et est sujet à composition avec des tierces parties [SZYPERSKI 2002].

Un composant est donc sujet à composition pour produire un résultat, avec d'autres, c'est à dire une application logicielle. Un composant est une unité physiquement réutilisable. Les patrons de conception seront, en revanche, des concepts réutilisables.

Définition 3.3 : Patron de conception

Les patrons de conception constituent une forme capitalisation de l'expérience pour construire des logiciels réutilisables. Ils fournissent un moyen de réutiliser la connaissance gagnée par des concepteurs chevronnés. Enfin, ils se comportent comme des briques pour construire des systèmes plus complexes ; en cela ils peuvent être considérés comme des micro-architectures qui contribuent à l'architecture globale du système [GAMMA et al. 1993].

Les patrons de conception s'organisent en plusieurs familles en raison des types de problèmes qu'ils sont amenés à résoudre. [GAMMA et al. 1999] organise ainsi 23 patrons de conception en 3 familles : les patrons de conception structurels, comportementaux et créationnels. Ce catalogue s'étend sans arrêt puisqu'il est alimenté par les retours d'expérience en génie logiciel comme en témoigne [CUNNINGHAM 2013].

Il est à noter que certains modes de réutilisation combinent deux précédemment évoqués ; il s'agit des *frameworks* [JOHNSON 1997].

Définition 3.4 : *Framework* ou Cadriciel

Un *framework* décrit l'architecture d'une application informatique, la nature de ses constituants et comment ils interagissent. C'est également un programme réutilisable destiné à intégrer du code ou des composants développés par une tierce partie et qui détermine le structure global et les flux de contrôle du logiciel [JOHNSON 1997].

En cela, un *framework* constitue la contrepartie physique d'une architecture, puisqu'il cadre le développement, l'utilisation et l'intégration de composants logiciels en son sein. Il n'est pas à confondre avec ce que l'on appelle les kits de développement [1] qui peuvent venir en complément :

Définition 3.5 : Kit de développement logiciel

Appelé aussi SDK (pour *Software development Kit*), il regroupe un ensemble d'outils de développement qui permettent la création d'applications pour une plate-forme (logicielle ou matérielle) cible.

Ainsi, la plateforme cible d'un kit de développement peut-être un *framework* et les outils mis à disposition peuvent être des bibliothèques, leurs API, des composants ou des outils pour les produire.

Enfin, dernière catégorie de concept que nous allons aborder, il s'agit des intergiciels :

1. Par abus de langage, il nous arrivera aussi de nommer *framework* le kit de développement et sa cible lorsque cette dernière est un *framework*.

Définition 3.6 : *Middleware* ou Intergiciel

Un *Middleware*[a] est défini comme une couche logicielle au dessus du système d'exploitation mais en dessous de l'application qui fournit une abstraction commune pour la programmation d'un système distribué [BAKKEN 2001].

a. Parfois appelé en français intergiciel.

Ces derniers peuvent donc être vus, dans une certaine mesure, comme des *frameworks* spécialisés dans la distribution des données ou des calculs, car, par nature, un intergiciel impose un style architectural distribué.

Ces concepts reviendront tout au long de l'exposé de ces recherches, en particulier au cours des deux premières parties de ce chapitre concernant les architectures modulaires et reconfigurable pour la réalité augmentée et la distribution.

3.1.2 Revue des architectures existantes pour la réalité augmentée

Dans cette partie, nous restreindront notre champ d'étude aux architecture appartenant aux domaines de la réalité augmentée et de l'informatique ubiquitaire, par ailleurs prolifiques. En effet, [ENDRES et al. 2005] dénombrait pas moins de 30 propositions d'architecture pour la réalité augmentée et/ou la réalité mixte.

La raison invoquée par [PONDER et al. 2003] est que, créer un système de réalité augmentée étant complexe, les développeurs se sont peu à peu tournés vers des solutions permettant de combiner et d'intégrer dans leurs logiciels des *frameworks* ou des composants réutilisables, la réponse naturelle étant le foisonnement des solutions architecturales et intergiciels. C'est un constat similaire à celui que nous avons écrit au chapitre précédent (section 2.3.1 page 63).

L'objectif n'est donc pas de citer l'intégralité de ces architectures, concrétisées par des *frameworks* ou des intergiciels mais de donner un aperçu des solutions proposées afin de discuter des caractéristiques souhaitables pour une architecture dédiée.

3.1.2.1 Etat de l'art

Avant de passer en revue ces architectures, il convient de remarquer que le domaine de la réalité augmentée rassemble deux communautés scientifiques aux vues différentes. La première, issue du monde de la vision par ordinateur, se concentre davantage sur le traitement des images et les informations que l'on peut en extraire. La deuxième, issue de la communauté de réalité virtuelle et des interfaces homme-machine, se concentre davantage sur le rendu et l'interaction. Ces deux communautés proposent également des architectures organisées différemment : les spécialistes de la vision par ordinateur se concentrent davantage sur les flux de données qui traversent et sont traités par les composants, ceux de réalité virtuelle utilisent davantage des organisations hiérarchique des composants destinés au rendu (ce que l'on appelle aussi un graphe de scène) qui devient alors le coeur de l'architecture. Les deux visions sont complémentaires (certaines des architectures proposées mêlent les deux), au contraire. Toutefois, chaque communauté met en avant plutôt un aspect qu'un autre.

De plus, le mode de communication entre les sous-systèmes d'une architecture exerce une influence : cette dernière sera considérée comme intrinsèquement distribuée si, lors du lancement d'une instance, cette dernière dépend d'un support réseau ou d'un intergiciel pour s'exécuter. D'autres architectures se contenteront, au contraire, de ne fournir ces capacités de distribution qu'à la demande. Enfin, nous pouvons rencontrer une dernière catégorie d'application qui sont basées sur des sous-systèmes fonctionnels dédiés. La limite est parfois assez floue, certaines architectures exhibant des caractéristiques provenant de ces différentes catégories.

Architecture en flots de données

Le projet TINMITH [PIEKARSKI et THOMAS 2003], démarré en 1998, est axé sur la description des flux de données. C'est une bibliothèque d'objets hiérarchisés conçue pour gérer les flux de données issus des capteurs, les différentes opérations de filtrage des données ainsi que le rendu final des applications. Écrite en C++, elle s'appuie sur un système de *callback* et un mécanisme de sérialisation du flux de données employant le XML. Les communications entre les objets sont gérées par un graphe de flux de données qui traverse différentes couches, de l'acquisition à l'étape de mixage du réel et du virtuel. Cette architecture, résolument orientée sur une vue de développement, est adaptée aux systèmes de réalité augmentée mobile, et embarqués grâce à l'effort d'optimisation des briques élémentaires du système.

IMAGETCLAR [OWEN et al. 2003] fournit un environnement de prototypage rapide pour concevoir et tester des applications de réalité augmentée. Les développeurs peuvent utiliser les composants fournis avec le moteur ou en développer de nouveaux C++. L'environnement proposé contient des scripts pour créer les squelettes des nouveaux composants ainsi qu'un éditeur graphique pour connecter visuellement les composants entre eux et générer automatiquement la glu logique associée, sous la forme de scripts en langage TCL.

VHD++ [PONDER et al. 2003] est un *framework* orienté objet développé en C++. Son architecture se rapproche de celle des micro-noyaux. En effet, une application contient un moteur (*vhdRuntimeEngine*) qui a pour rôle d'instancier et de gérer des services (*vhdService*) prenant en charge les diverses fonctionnalités liées à l'exécution d'une application de réalité augmentée. Le moteur gère des propriétés (*vhdProperty*) décrivant l'état de système et sa configuration (une propriété peut-être, par exemple, un graphe de scène). VHD++ est extensible, chaque nouveau composant devant être déclaré comme un service. Ces derniers, s'ils sont gérés par le même moteur, échangent directement par appel de fonction. Enfin, des services gérés par des moteurs différent peuvent communiquer entre eux par le biais d'une version C++ du protocole de distribution RMI (*Remote Method Invocation*). Enfin, le moteur est configurable à l'aide de fichiers XML.

La solution commerciale D-FUSION, produit de Total-Immersion [TOTALIMMERSION 2008], est un kit de développement comprenant divers composants pour l'acquisition, le suivi, l'interaction et le rendu. La plateforme permet de développer des greffons et elle peut-être configurée à l'aide scripts écrits dans le langage Lua.

Architectures en flots de données distribués

Initié en 2000 à l'université de Munich, le projet DWARF [BAUER et al. 2001] (pour *Distributed Wearable Augmented Reality Framework*) est un système de composants distribués décentralisé. Il est constitué d'un ensemble de services basés sur CORBA (*Common Object Request Broker Architecture*). Chacun est décrit par un fichier XML et peut être sollicité dynamiquement à l'exécution, l'ensemble formant un graphe de flux de données distribué. Ainsi, chaque capteur, algorithme de suivi, filtre ou boucle de rendu est encapsulé dans un service. Pour démarrer, le système lance un gestionnaire qui explore le réseau pour les détecter et les intégrer en cours d'exécution.

AMIRE [DÖRNER et al. 2002 ; ABAWI et al. 2004] (pour *Authoring MIxed REality*) est issu d'un projet européen qui s'est déroulé de 2002 à 2004 et dont le but était de définir et développer une architecture pour prototyper rapidement des applications de réalité mixte sans connaître les détails des technologies sous-jacentes employées. Cette architecture orientée composants comporte deux niveaux de granularité : les gemmes (des micro-composants) et les composants (agrégation de plusieurs gemmes). Les deux comportent une interface de configuration, des « slots » d'entrée, et de sortie, points d'échange avec d'autres composants. Par ailleurs, cette architecture possède une ouverture sur les systèmes distribués par le biais de l'utilisation de CORBA. Enfin, ce système propose une interface de programmation visuelle par bloc pour connecter les divers composants tout en étant guidé pas à pas

afin de suivre des règles génériques de conception des applications de réalité mixte. La configuration des applications obtenue est alors stockée dans un fichier XML.

MORGAN [OHLENBURG et al. 2004] fournit un ensemble de bibliothèques pour prototyper rapidement des applications distribuées de réalité virtuelle et augmentée. Elles incorporent des composants qui interfacent des périphériques de suivi et de capture de mouvement ainsi que divers moteurs de gestion de graphes de scène. MORGAN utilise CORBA pour résoudre le problème de la distribution et s'appuie sur un certain nombre de patrons de conception connus afin de faciliter la compréhension et l'extension de son fonctionnement interne.

Graphes de scène

VIRTOOLS [FUCHS et MOREAU 2006], dont certains concepts ont été transposé à 3DVia, est une solution propriétaire maintenue par Dassault Systèmes. Ce logiciel produit des applications 3D interactives et dispose d'extensions pour les applications de réalité virtuelle et, dans une moindre mesure, de réalité augmentée. L'outil utilise un langage de programmation visuel pour attribuer des comportements aux objets (nommés, dans sa terminologie, *Building blocks*) qui composent l'application. L'outil propose plusieurs vues de l'application qui permettent, entre autre, de gérer l'organisation hiérarchique des *Building blocks* (leur graphe de scène) et le flux de donnée les traversant. Des modules supplémentaires permettent la distribution des applications développées sur des grappes d'ordinateurs.

Plus récent, ARTIFICE [KAUFMANN et al. 2012] est un *framework* développé en utilisant le moteur de jeu Unity3D en tant que base (et qui est en réalité le véritable *framework*). Il y ajoute trois sous-systèmes spécifiques pour la réalité augmentée qui gèrent le suivi, l'interaction et la distribution. Le suivi est géré par deux intergiciels qui sont *OpenVideo* et *OpenTracker*. La distribution s'appuie sur les mécanismes réseau présents dans Unity3D qui permet à des *GameObject*[2] de communiquer entre eux et entre instances différentes du moteur. La configuration et le développement se fait donc en utilisant les moyens mis à disposition avec le moteur de jeu, à savoir UnityScript[3] et C# comme langages de programmation.

Graphes de scène distribués

Développée par l'Université Technique de Vienne et de Graz (Autriche) depuis 1997, la *StudierStube* [SCHMALSTIEG et al. 2002] est l'architecture spécifique à la réalité augmentée la plus ancienne encore en activité. Chaque utilisateur dispose d'un espace de travail décrit par le biais d'un graphe de scène distribué (basé sur l'API d'OpenInventor) avec une partie privative et une partie commune. Un gestionnaire de sessions centralisé orchestre les échanges entre espaces de travail. Pour gérer les capteurs ainsi que les périphériques d'entrée, ce projet intègre *OpenTracker* [REITMAYR et SCHMALSTIEG 2001], un sous-système s'appuyant sur une description XML (eXtensible Markup Language) des flux de données. Le développeur peut programmer entièrement à l'aide de scripts en utilisant des noeuds pré-définis ou développer ses propres noeuds en C++. Il est à noter que ce système convient particulièrement pour les projets de RA collaboratifs et distribués.

AVANGO, et son successeur AVANGO-NG [KUCK et al. 2008], sont orientés vers les environnements interactifs distribués pour les réalités virtuelles et augmentées. Son architecture se calque sur celle des graphes de scène, les noeuds étant des composants qui communiquent entre eux par des routes entre les champs (propriétés des noeuds). La dernière version d'AVANGO utilise OpenSceneGraph pour la gestion des graphes de scène et rajoute une couche pour permettre la communication entre les noeuds d'une instance à une autre du moteur via le réseau. Le *framework* interprète les lan-

2. Composants de base de Unity3D organisés hiérarchiquement suivant un graphe de scène.
3. Souvent abusivement appelé JavaScript alors que les deux langages ont peu de points communs.

gages de script Python ou Scheme. De nouveaux composants peuvent être ajoutés par programmation en C++.

Sous-système fonctionnels

VARU [IRAWATI et al. 2008] s'attache à combiner trois espaces d'interaction : réalité virtuelle, réalité augmentée et informatique ubiquitaire. Un utilisateur peut passer de l'un à l'autre et collaborer avec les autres sans être dans le même espace. Il s'agit d'une architecture client-serveur où le serveur entrepose les objets communs et gère la simulation pour l'ensemble des clients, les notifiant dès lors qu'un changement survient sur un objet commun. Chaque client est composé d'un noyau en charge de la communication et la gestion des composants. Un certain nombre de ces derniers sont prédéfinies pour gérer les espaces d'interaction (au choix parmi les trois), les périphériques, l'affichage et la récupération des flux audio et vidéo. De nouveaux composants peuvent être développés en utilisant une API programmée en C++. Enfin, le serveur et chaque client est configurable en utilisant un fichier de description XML.

Pensé pour l'expérience utilisateur et l'accroche narrative en réalité mixte, MRSS (pour *Mixed Reality Software Suite*) [HUGHES et al. 2005] intègre quatre sous-systèmes aux fonctionnalités dédiées. Le sous-système principal, le *story engine* a pour fonction de charger des scripts et suivre une scénarisation. Les trois autres sous-système gèrent les rendus graphique, audio et les effets spéciaux. Ils communiquent ainsi avec les données issues des capteurs, des périphériques ainsi que de différents greffons (*plugins*) prenant en charge la simulation physique ou l'intelligence artificielle d'agents virtuels. La communication est assurée entre les moteurs et les différents modules du système via un protocole dédié développé sur la base de TCP/IP. Les instances de MRSS peuvent également dialoguer entre elle. Enfin, la configuration se fait par le biais de scénarios décrits en utilisant XML et des scripts écrits en langage C ou Java.

3.1.2.2 Caractéristiques d'une architecture dédiée pour la réalité augmentée

Architecture	Granularité	Flexibilité	Reconfigurabilité	Distribution
TINMITH	**Variable**	**Forte**	Non	**A la demande**
IMAGETCLAR	**Variable**	Faible	Non	Déléguée
VHD++	**Variable**	**Forte**	Non	**A la demande**
D-FUSION	**Variable**	Moyenne	Non	Déléguée
DWARF	Grossière	**Forte**	**Oui**	Intrinsèque
AMIRE	**Variable**	**Forte**	Non	Déléguée
MORGAN	Grossière	**Forte**	Non	Intrinsèque
VIRTOOLS	**Variable**	Moyenne	Non	**A la demande**
ARTIFICE	**Variable**	Moyenne	Non	**A la demande**
STUDIERSTUBE	**Variable**	Moyenne	Non	Intrinsèque
AVANGO-NG	**Variable**	**Forte**	Non	Intrinsèque
VARU	Grossière	Faible	**Oui**	Intrinsèque
MRSS	Grossière	Faible	Non	**A la demande**
Cible	**Variable**	**Forte**	**Oui**	**A la demande**

TABLE 3.1 – Comparaison des caractéristiques des architectures

Nous venons de le voir, la littérature au sujet des architectures logicielles pour la réalité augmentée est riche. De nombreuses options existent et le choix entre les différentes solution difficile. De plus, concevoir un modèle architectural stable et solide est un processus long et itératif [JOHNSON 1997]

qui nécessite, pour faire évoluer convenablement le *framework* résultant, d'étudier et d'être confronté à de nombreuses applications du domaine.

Nous remarquerons que la plupart de ces architectures ont un point commun : elles sont à base de composants tels que définis par [SZYPERSKI 2002] qui impliquent *de facto*, modularité et réutilisabilité. Toutefois, cette réutilisabilité peut rencontrer certaines limites en fonction de l'architecture choisie. Ainsi, les composants possèdent une *granularité*. Elle est fine lorsqu'un module implémente un algorithme simple. À l'inverse, nous trouvons les systèmes monolithiques. Une granularité trop fine peut engendrer des inconvénients en termes de performances, le coût en temps de calcul de la communication prenant le pas sur le coût des actions effectuées par les composants. Une granularité trop grossière handicapera la réutilisabilité du composant. On parlera parfois de *blob* ou de *god object* pour qualifier cette situation, qui est connue pour être un anti-patron de conception, soit une pratique à éviter dans le domaine du génie logiciel [CUNNINGHAM 2013]. De plus, il existe des moyens de faire varier cette granularité en jouant sur des possibilités d'abstraction telles que les offre le patron de conception *Composite* permettant d'agréger des composants pour en former de nouveaux, de granularité plus grossière. Ce mécanisme est, par exemple, présent dans les architectures à base de graphe de scène.

En vue de la pérenniser, une architecture doit permettre de produire des applications *flexibles*. Cela se manifeste par une capacité d'adaptation par rapport à des besoins ou à des degrés de libertés variables [MARLET 2011]. Ainsi, une architecture telle que celle de MRSS a tendance à figer les capacités d'évolution de l'application (par exemple, le rendu haptique n'est pas envisageable). A ce niveau, une autre propriété associable à la flexibilité est l'indépendance par rapport à un moteur de rendu [KUCK et al. 2008]. Ceci veut donc dire que les choix architecturaux ne doivent pas orienter les choix fonctionnels mais plutôt les accompagner.

De plus, et c'est peut-être l'une des caractéristiques les moins répandues, nous souhaitions, pour notre architecture, disposer d'un mécanisme de *reconfiguration* à l'exécution. Ici, nous nous intéresserons plus particulièrement à la reconfiguration dynamique du chemin de données de l'application. Ce mécanisme permet à l'exécution de pallier les défaillances d'un capteur ou de parer à des scénarios de type débranchement à chaud d'un ou plusieurs capteurs. Cette capacité a été notamment utilisée dans le cadre du projet RAXENV (voir section **??**).

Caractéristique supplémentaire, les modules peuvent être *distribués* sur un réseau de machines ou situés sur un seul ordinateur. Pour nous, la distribution sera *intrinsèque* dans une architecture lorsque toutes les communications inter-modules font appel à des mécanismes de distribution : ces derniers font partie intégrante du moteur qui gère les composants ainsi que leurs connexions. Comme [PIEKARSKI et THOMAS 2003] le souligne, cette caractéristique n'est pas forcément souhaitable, car ce mode de communication impose que les données soient sérialisées et désérialisées au cours de la transmission, ce qui occasionne un coût de communication non négligeable en termes de performances de l'architecture. À l'inverse, d'autres architectures permettent une distribution à la demande ou *délèguent* la distribution au niveau des composants : ces derniers peuvent contenir ou non du code permettant de distribuer l'application.

Enfin, un examen de l'évolution des architectures montrent que celles-ci intègrent davantage de composants et de technologies hétérogènes qui font elles-même appel à des architectures orientées composants (VARU,ARTIFICE et MRSS en sont des exemples). Il nous paraît intéressant de pouvoir faciliter l'intégration de ces dernières. Cette problématique de la complexité croissante des systèmes de réalité augmentée incite également à séparer les rôles des personnes intervenant dans le développement et la conception des applications de réalité augmentée afin de maîtriser la complexité [ABAWI et al. 2004]. Une façon de l'envisager est de séparer les rôles de développeur de composant (qui travaillent davantage sur la technologie et les algorithmes) et les développeurs d'application qui se concentrent sur l'intégration des composants.

Pour résumer, nous avons souhaité adopter le positionnement suivant pour notre architecture :

Positionnement 3.1 : Architecture souhaitée

L'architecture visée doit exhiber les propriétés suivantes :

1. Orientée composants ;
2. Reconfigurable à l'exécution ;
3. Distribuable à la demande ;
4. Granularité variable ;
5. Flexible ;
6. Intégration étendue.

Le tableau 3.1 page 72 compare les systèmes étudiés sur une partie des critères énoncés. Les architectures présentées ne pouvant pas satisfaire les critères choisis, nous nous sommes attelés à la tâche de construire une architecture dédiée qui y correspond. La partie distribution fera l'objet d'une partie séparée.

3.1.3 Modèle d'architecture proposé

Le modèle d'architecture présenté ici est la synthèse de quinze années d'évolution, les premiers fondements ayant été esquissés au cours de ma thèse [DIDIER 2005] puis successivement développés dans [DIDIER et al. 2006 ; DIDIER et al. 2009b ; DIDIER et al. 2012].

Pour décrire une architecture orientée composant (caractéristique 1 de notre positionnement), nous devons décrire le modèle de composant ainsi que le modèle d'application que nous proposons. Ceci nous amènera naturellement à traiter de la question de la reconfiguration, puis des capacités d'intégration de l'architecture.

Les modèles proposés ont été implémentés dans notre *framework* ARCS (pour *Augmented Reality Component System*) qui a connu plusieurs évolutions de 2003 à maintenant.

3.1.3.1 Modèle de composant

Selon la définition de [SZYPERSKI 2002], un composant doit être capable de décrire son interface. Dans notre cas, cette dernière est constituée d'entrées, les *slots*, et de sorties, les *signaux* (figure 3.1). Cette terminologie est empruntée à celle de plusieurs bibliothèques d'interfaces graphiques qui emploient ce type de communication entre des objets déclenchant des évènements et les objets les réceptionnant. Ce mécanisme, voisin du patron de conception *Observateur*, permet aux composants de communiquer entre eux de manière synchrone sans se connaître *a priori*. Ces connexions sont gérées, en ce sens qu'elles peuvent être établies et coupées à l'exécution avec les contrôles de rigueur concernant l'existence et la destruction d'une des deux parties de la communication. Du point de vue de l'environnement ARCS, un composant est un objet qui contient des signaux, des slots ainsi que des mécanismes d'introspection permettant d'explorer son interface.

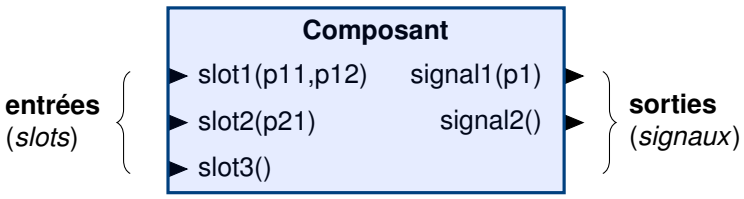


FIGURE 3.1 – Représentation graphique d'un composant

La communication entre deux composants n'est possible que lorsqu'un signal est connecté à un slot. A l'intérieur d'un composant, l'activation d'un slot (assimilable à un appel de méthode ou de

Pseudo-code du slot du composant `a` :

```
function mySlotA()
{
    emit mySignal1();
    emit mySignal2();
}
```

Liste de connections :

```
a.mySignal1() --> b.mySlotB()
a.mySignal2() --> c.mySlotC()
```

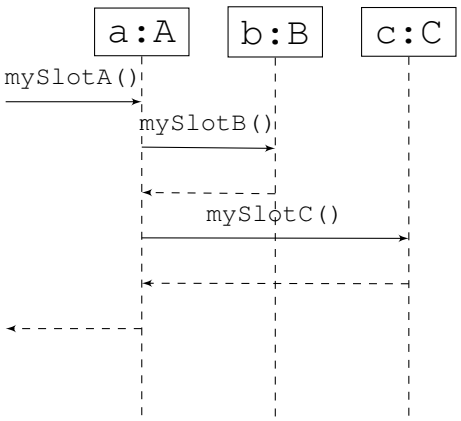
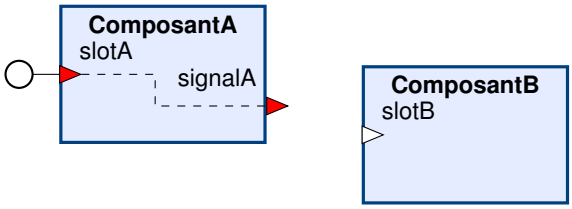


FIGURE 3.2 – Modèle de communication synchrone entre composants

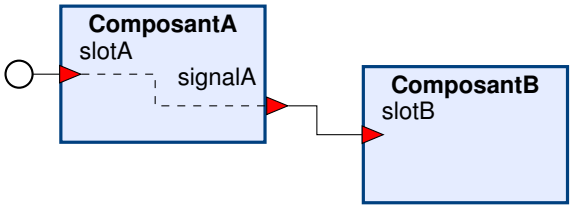
fonction) peut déclencher en retour un ou plusieurs signaux. La communication étant synchrone, le déclenchement d'un signal active immédiatement le slot qui lui est éventuellement connecté. Le composant ayant émis le signal ne reprend ses actions que lorsque le slot appelé a achevé son exécution. La figure 3.2 montre, pour un exemple avec trois composants, comment les signaux activent les slots ainsi que la séquence d'exécution associée (représentée à l'aide d'un diagramme de séquence UML).

Pour terminer, il convient d'expliquer comment les composants sont configurés. Cela est effectué au travers de deux mécanismes. Le premier, classique, consiste à le faire au moment de son instanciation, le second, en invoquant des slots. Ces derniers ne devront alors comporter que des paramètres dont le type est sérialisable. Par la suite, nous distinguerons deux types d'invocation : les *invocations de pré-connexion* et les *invocations de post-connexion*. La figure 3.3 explique les raisons de cette différenciation.



Soient deux composants A et B dont les connexions ne sont pas encore réalisées. A contient un slot *slotA* qui active un signal *signalA* et B possède un slot *slotB*. Si nous invoquons *slotA*, celui-ci active uniquement *signalA*.

(a) Invocation de pré-connexion



Les connexions sont maintenant réalisées. Si nous invoquons *slotA*, celui-ci active *signalA*. Ce dernier, à présent connecté au *slotB*, est à son tour activé. Les invocations se propagent alors le long des connexions.

(b) Invocation de post-connexion

FIGURE 3.3 – Mécanisme de propagation des invocations

3.1.3.2 Modèle d'application et mécanisme de reconfiguration

Le mécanisme de communication entre les composants s'effectue par des connexions signal-slot. Nous appelons une *feuille* (ou *sheet*) un ensemble de composants et de connexions complété par la liste des invocations, de pré-connexion ou de post-connexion. Elle constitue le résultat de la composition de nos composants. Une feuille représentée graphiquement aura l'aspect de la figure 3.4.

Une feuille n'est finalement que la représentation de l'état d'un processus à une période donnée. Pour décrire un processus dans sa totalité, un ensemble de plusieurs feuilles est nécessaire (voir fi-

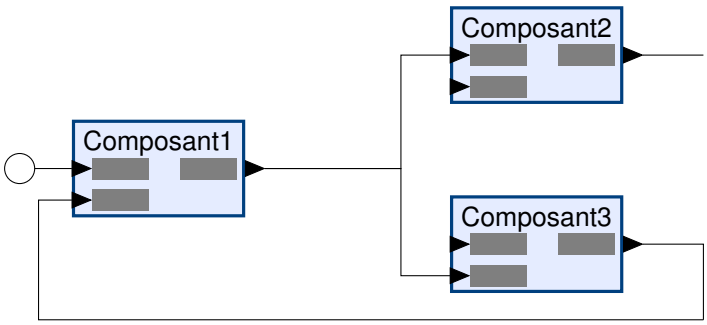


FIGURE 3.4 – Une feuille est le résultat de la composition de plusieurs composants.

gure 3.5). Sachant qu'une seule feuille est opérationnelle à un instant donné, il est nécessaire d'y adjoindre un mécanisme de gestion des feuilles nommé *contrôleur*.

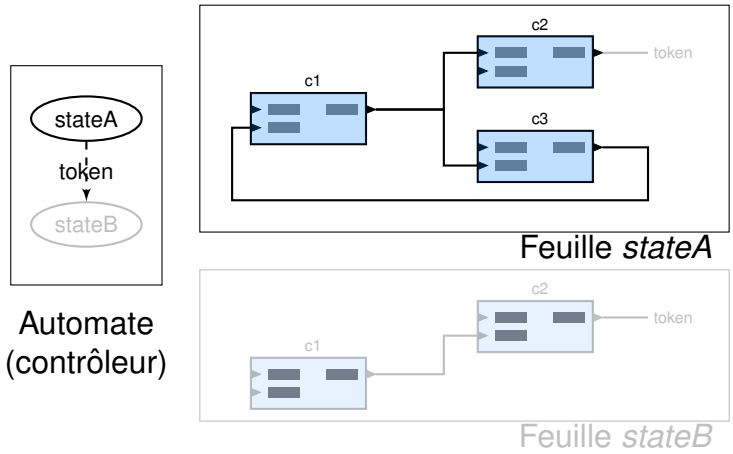


FIGURE 3.5 – Vue d'un processus : un automate gérant plusieurs feuilles

Le contrôleur est un automate de [MOORE 1956] donc à états finis reliés par un ensemble de transitions. En fonction de l'état courant de l'automate va correspondre une feuille décrivant les interactions entre les composants. le changement d'état est déclenché quand l'automate reçoit un jeton de la part de la feuille courante. Enfin, notre automate contient un état terminal qui, lorsqu'il est atteint, arrête l'application.

Les feuilles communiquent avec l'automate par le biais d'un de ses composants en charge d'émettre un jeton. Nous allons à présent détailler le mécanisme de passage d'un état de l'automate à un autre ce qui introduit la notion de reconfigurabilité de notre système.

Remarque 3.1 : Cycle de vie des composants

La feuille, en tant que configuration opérationnelle d'une application, gère le cycle de vie des composants, ces derniers pouvant être partagés entre les feuilles ou, au contraire, propre à une seule. Deux mécanismes additionnels sont donc introduits :

— A une feuille est associé un *contexte* regroupant les composants propres à cette dernière et différent du contexte global du processus ;

— Des invocations dites de *nettoyage* sont proposées pour restaurer l'état d'un composant avant le passage à une autre feuille.

Les modifications décrites ici sont abordées dans [DIDIER et al. 2012] et succèdent à d'autres mécanismes présentés dans des publications antérieures.

À chaque état d'un processus correspond une feuille. *À tout moment, il n'y a qu'une feuille active*

c'est à dire qui décrit la configuration dans laquelle les composants œuvrent de concert. En supposant qu'une feuille est active et que cette dernière envoie un jeton à l'automate, si ce jeton et cette feuille correspondent à une transition, alors le changement d'état est déclenché. Celui-ci s'effectue en plusieurs étapes :

1. *Déconnexions :* les liens signaux/slots décrits par l'état initial sont déconnectés. À la fin de cette étape, toute communication entre composant est impossible ;
2. *Invocations de nettoyage :* les composants peuvent être configurés pour, par exemple, restaurer un état de départ ;
3. *Changement de la feuille courante :* la feuille courante change et devient celle décrite par l'état final de la transition qui a été déclenchée. Les composants dépendant du contexte de l'ancienne feuille sont détruits, ceux propres au nouveau contexte sont instanciés ;
4. *Invocations de pré-connexion :* avant de connecter les différents liens signaux/slots de la feuille courante, cette étape initialise les objets ;
5. *Connexions :* les différents liens signaux/slots décrits dans la feuille courante sont activés. À la fin de cette étape, les composants communiquent à nouveau ;
6. *Invocations de post-connexions :* ces dernières peuvent se propager au travers de la feuille par le biais des communications signaux/slots.

À l'issue de ce cycle, la feuille courante a changé, ce qui a reconfiguré les communications existantes entre les composants du processus. Ce mécanisme permet de gérer le cycle de vie des composants puisque certains ne seront pas utilisés sur toutes les feuilles.

Jusqu'à présent, nous avons évoqué des processus. Pour parachever notre modèle d'application, il nous faut ajouter qu'une application est un ensemble de processus concurrents, chaque processus étant plus exactement un fil d'exécution (ou *thread*).

Pour terminer sur cette première partie, l'architecture proposée a été implémentée dans un *framework* nommé ARCS pour (*Augmented Reality Component System*) [4]. Ce dernier est actuellement disponible dans deux versions, l'une, écrite dans le langage C++, est destinée aux applications nécessitant de bonnes performances en raison d'algorithmes coûteux en temps de calcul et l'autre écrite dans le langage Javascript, ciblant une exécution dans le navigateur web. La plupart des concepts présentés jusqu'ici sont présent sur les deux implémentation à l'exclusion du parallélisme entre les processus pour la version Javascript. La section suivante concerne donc plus particulièrement la version C++. Comme il s'agit d'un *framework*, ce dernier se manifeste par un moteur d'exécution prenant en charge la gestion des applications à partir d'une description XML de cette dernière. Les détails pratiques sur l'implémentation de l'architecture seront donnés à la section 3.1.3.4 page 80.

3.1.3.3 Capacité étendue d'intégration

Le modèle d'architecture présenté est relativement flexible dans la mesure où il n'oriente pas les choix fonctionnels (choix métier) des applications développées sur cette base. Nous avons vu qu'une bonne pratique est également d'être indépendant des moteurs de rendu. Toutefois, nous souhaitons pouvoir les intégrer. Certaines API de graphes de scène, telles qu'*Inventor* ou *OpenSceneGraph* décrivent une hiérarchie d'objets qui constituent la scène et qui sont appelés nœuds, ces derniers possédant des champs permettant de les configurer. Ces bibliothèques exposent également les propriétés des nœuds. En cela, leur fonctionnement proche de celui des composants. Par ailleurs, un certain nombre d'autres *frameworks* utilisent également une approche orientée composants.

4. Voir le site http://arcs.ibisc.fr

Notion de familles de composants

Nous nous sommes donc demandé s'il n'était pas possible de faire en sorte que l'architecture soit capable d'accueillir d'autres familles de composants (que nous appellerons des composants *exogènes*), ces derniers pouvant alors être manipulés comme s'ils faisaient partie des composants de l'architecture (considérés comme des composants *natif*). Ces aspects sont essentiellement présentés dans DIDIER et al. 2012 et modifient le modèle de composant de l'architecture.

Ce dernier autorise l'introduction de nouveaux types de composants dans la mesure où ils respectent le paradigme signal/slot. Au niveau implémentation, nous avons recours à une classe abstraite représentant les composants (nommée *AbstractComponent*). Intégrer dans l'architecture une nouvelle famille de composants passe par la spécialisation de cette classe qui doit définir :

— Les mécanismes d'instanciation et de destruction du composant concret : *AbstractComponent* est donc une interface qui agit en tant qu'adaptateur et gère la référence à ce dernier ;

— La gestion des signaux et des slots : les entrées et sorties du composant concret doivent être exportée et exploitable de cette manière ;

— La gestion des connexions : l'utilisation du patron de conception *adaptateur* est connue pour introduire un surcoût en temps de calcul. Dans le cas où deux composants proviennent de la même famille, il est intéressant d'optimiser la connexion entre les deux en utilisant le mécanisme prévu à l'origine dans la famille. Cela en évite en effet de traduire une sortie en signal, puis de la transmettre à un slot qui effectuera la transformation inverse à destination de l'entrée du composant concret. Dans le cas où les composants ne font pas partie de la même famille, un mécanisme standardisé de communication est mis en place ;

— Des fonctions de sérialisation/désérialisation : pour l'instanciation et la configuration des composants.

Puisque l'idée est de rendre possible l'extension d'ARCS en interfaçant d'autres familles de composants. Le moteur contient une fabrique (*Factory*) extensible pour introduire de nouvelles familles de composants. Cette dernière fait appel à des classes dérivant d'*AbstractFamily*, qui est l'interface pour déclarer des fabriques supplémentaires de composants associées à la nouvelle famille. Son rôle est de gérer ces fabriques ainsi que la gestion de l'instanciation des composants de la famille.

Remarque 3.2

A notre connaissance, il s'agit de la seule architecture dédiées aux applications de réalité augmentée permettant d'intégrer d'autres familles de composants de cette manière.

La figure 3.6 montre le diagramme de classe d'une partie du moteur où la fabrique intervient. Elle se base sur le patron de conception *fabrique abstraite* (ou *Abstract Factory*). Dans la figure, deux familles concrètes de composants sont introduites :

— La famille native (*NativeFamily*) à ARCS et qui instancie des fabriques de composants natifs (*NativeComponent*) adaptant des objets de type *QObjet* dont nous verrons la raison lorsque nous aborderons la question de l'implémentation ;

— La famille *inventor* (*InventorFamily*) qui instancie des composants de type *InventorComponent*, en réalité des noeuds de graphe de scène dérivés de la classe *SoFieldContainer*, les rendant accessibles directement par le moteur d'ARCS.

De la même manière, d'autres familles de composants ont été développées, notamment pour gérer la distribution à la demande (voir section 3.2 page 88), pour scripter une partie du comportement du moteur ou encore pour interfacer *OpenSceneGraph*, une autre bibliothèque de graphe de scène.

Pour compléter l'intégration de famille de composants exogènes, le *framework* incorpore également des fabriques de type pour sérialiser, désérialiser et instancier des objets manipulés par les familles.

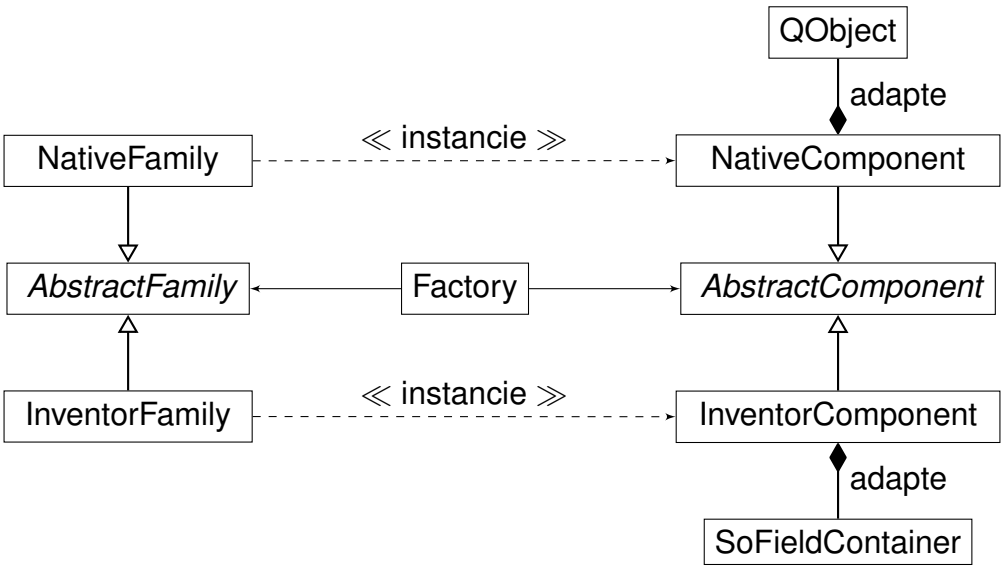


FIGURE 3.6 – Diagramme de classes UML de la fabrique de composant du moteur d'ARCS.

Exemple – intégration d'Inventor

Afin d'éprouver notre architecture, nous avons mis au point les classes permettant d'accéder aux nœuds des graphes de scène *Inventor* comme s'ils étaient des composants d'ARCS. Si nous considérons l'une des implémentations actuelles, à savoir Coin3D [*SIM – Coin3D 3D Graphics Development Kit* 2014], cette dernière est composée d'environ 350 000 lignes de codes et donne accès à pas moins de 300 différents types de noeuds.

Les noeuds dans Inventor peuvent être vus comme des composants qui dérivent d'une classe nommée *SoFieldContainer*, chargée de gérer des propriétés (les champs) dont les valeurs modifient l'état interne des noeuds et/ou leur comportement. L'API d'Inventor possède des facultés d'introspection en ce qui concerne ces classes : elle permet de déterminer, à l'exécution, leur type et la liste des champs (leurs type et leur valeurs). Inventor étant une bibliothèque de gestion de graphes de scènes, ses noeuds sont organisés hiérarchiquement afin de décrire une scène 3D selon des relations père – fils. Dans le même temps, un champ d'un noeud peut-être connecté à un autre champ afin de propager des modifications dans le graphe de scène.

L'adaptation des nœuds en composants pour ARCS est alors directe : les nœuds deviennent des composants et les champs sont transformés en signaux et slots en fonction de leurs capacités. Inventor possédant des mécanismes de sérialisation (la bibliothèque dispose d'un format natif de description de graphes de scène auquel VRML est apparenté), nous pouvons les intégrer dans ARCS.

```
<component id="cube" type="Cube"/>
<component id="mat"  type="Material">Material { diffuseColor 1.0 0.0 0.0}</component>
<component id="scene" type="Separator">
    Separator {
        RotationXYZ { axis X  angle 0.707 }
        RotationXYZ {
            axis Y
            angle 0 = ElapsedTime { speed 0.6 } . timeOut
        }
        Material { diffuseColor 1.0 0.025 0 }
        Cube { width 2 height 1 }
    }
</component>
```

Listing 3.1 – Déclaration de noeuds inventor en tant que composants

Cela nous permet de manipuler les noeuds Inventor et des composants natifs comme on peut le

constater au listing 3.1 page précédente qui est un extrait de configuration XML du moteur. Nous pouvons ainsi constater qu'aux lignes 4 à 12, le code écrit est en réalité dans le format textuel de description d'un graphe de scène Inventor. Les lignes 1 et 2 ne sont pas en reste car les composants manipulés, de type *Cube* et *Material* sont des nœuds Inventor. La capture d'écran reproduite à la figure 3.7 montre l'intégration des noeuds Inventor dans une application basée marqueur. Les lecteurs familiers d'Inventor auront remarqué que l'un des manipulateurs caractéristique de cette bibliothèque est affiché autour du personnage virtuel.

FIGURE 3.7 – Capture d'écran d'une application basée marqueur

Pour terminer, l'intégration d'Inventor a été réalisée à l'aide d'environ 2000 lignes de codes et permet un accès complet aux 300 nœuds et aux différents types de champs. L'implémentation se concrétise par l'écriture de deux classes principales, *InventorFamily* et *InventorComponent*, ainsi que quelques classes satellites agissant comme des intermédiaires entre les composants natifs et les composants Inventor.

3.1.3.4 Implémentation des concepts

ARCS désigne à la fois une bibliothèque de classes, un *framework* et un kit de développement. Écrit en C++, son coeur comporte environ une trentaine de classes concrétisant cette architecture. De plus, en raison de facilités techniques, ARCS s'appuie sur le kit de développement *Qt* (dans ses versions 4 et 5), utilisé pour le développement d'interfaces graphiques THE QT COMPANY 2017, qui fournit des bibliothèques facilitant la portabilité d'un système d'exploitation à un autre (ARCS est ainsi exécutable sur différentes plate-formes : windows, linux, ...) et qui propose une implémentation du mécanisme signal/slot.

Structure de la bibliothèque

Nous subdiviserons la modélisation de la structure de la bibliothèque en trois volets. Le premier concerne les concepts de base de l'architecture. Comme nous pouvons le constater à la figure 3.8 page ci-contre, une application sous ARCS comporte un contexte partagé par des processus, ou plus exactement des *threads*. Chaque processus est piloté par un contrôleur (une machine à états finis) et contient plusieurs feuilles (*sheet*) qui représentent chacune une configuration opérationnelle d'un processus et contiennent un contexte propre, une liste d'invocations pour configurer les composants ainsi que la liste des connexions qui représentent les communications entre composants. Il est à noter que l'application et le contrôleur sont eux-même considérés comme des instances de composants, ce

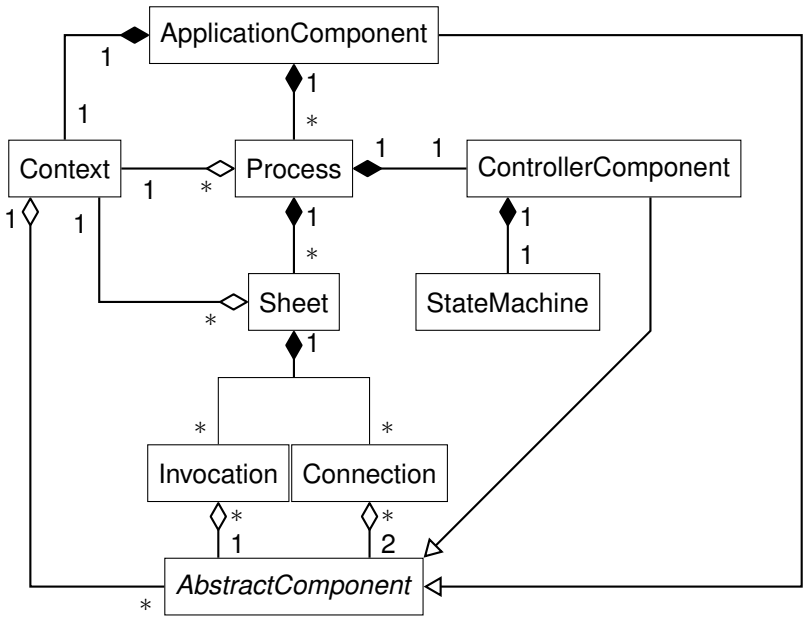


FIGURE 3.8 – Diagramme de classe des concepts architecturaux d'ARCS.

qui leur permet d'exhiber les mêmes comportements, entre autres, la sérialisation/désérialisation, et l'utilisation de signaux et de slots.

Un deuxième volet de la modélisation porte sur l'instanciation des composants et du mécanisme de fabrique qui lui est associé (voir figure 3.9 page 83). La classe centrale du dispositif (*Factory*) est implémentée en utilisant le patron de conception *Singleton* afin de s'assurer qu'une et une seule instance de cette dernière existe dans le *framework*. Celle-ci pilote des gestionnaires de bibliothèques de composants. Ces dernières sont de type dynamique, c'est à dire chargeables au cours de l'exécution d'un programme [5]. Elles possèdent un point d'entrée pour exporter des définitions de familles exogènes (*AbstractFamily*), de composants natifs (*NativeComponent*) héritant de *AbstractComponent* et de types. Les contextes que nous avions vu précédemment gèrent donc les composants par le biais de *Factory*. Une dernière classe s'occupe de lire et d'exploiter les formats XML utilisés par le *framework*.

Enfin, au niveau de la modélisation, la possibilité est offerte de créer de nouveaux composants par composition. Ce mécanisme est calqué sur le patron de conception composite (voir figure 3.10) auquel un niveau intermédiaire est rajouté : un *CompositeComponent* hérite d'*AbstractComponent* et contient un contexte, une feuille indiquant comment les composants sont connectés et configurés et des adaptateurs (*Method*) qui assurent le routage des informations des signaux et slots généraux de la structure composite aux méthodes des sous-composants concernés. Cela permet d'obtenir une granularité variable, non seulement en fonction de la taille du composant développé, mais aussi par composition de ces derniers (qui peuvent être eux-même composites, ...).

Comportement du moteur

ARCS est un *framework*, ce qui veut dire qu'il fournit un cadre architectural. Il prend en charge les flux de contrôle de l'application, et dans notre cas, gère aussi les cycles de vie des composants. Une application est donc décrite dans un fichier XML dont la structure est calquée sur celle de l'architecture comme le listing 3.2 page suivante permet de le constater. Une explication détaillée du format de ces fichiers a été reléguée à l'annexe **??** page ??.

Le moteur d'exécution, à son démarrage, lit une description XML d'une application. Puis, en fonction des informations consignées dans cette dernière, charge les bibliothèques dynamiques contenant

5. Les bibliothèques dynamiques sous Windows utilisent l'extension *dll*, pour *dynamic link library*. Sous les systèmes de type Unix, ce seront des extensions *so*, pour *shared objects*.

```
<application mode="event">
  <context>
    <libraries>
      <library path="../../sample/sample"/>
    </libraries>
    <components>
      <component id="b" type="Loop" />
      <component id="d" type="DisplayInt" />
      <component id="s" type="StateMachine">
        <statemachine>
          <first name="start"/>
          <last name="end"/>
          <transitions>
            <transition source="start" token="end" destination="end"/>
          </transitions>
        </statemachine>
      </component>
    </components>
    <constants>
      <constant id="iterations" type="int">5</constant>
    </constants>
  </context>

  <processes>
    <process controller="s">
      <sheet id="start">
        <connections>
          <link source="b" signal="newIteration(int)" destination="d"
              slot="display(int)"/>
          <link source="b" signal="sendToken(QString)" destination="s"
              slot="setToken(QString)" />
        </connections>
        <postconnections>
          <invoke destination="b" slot="setIterations(int)"
              type="constant">iterations</invoke>
        </postconnections>
      </sheet>
      <sheet id="end"/>
    </process>
  </processes>
</application>
```

Listing 3.2 – Description d'une application ARCS au format XML

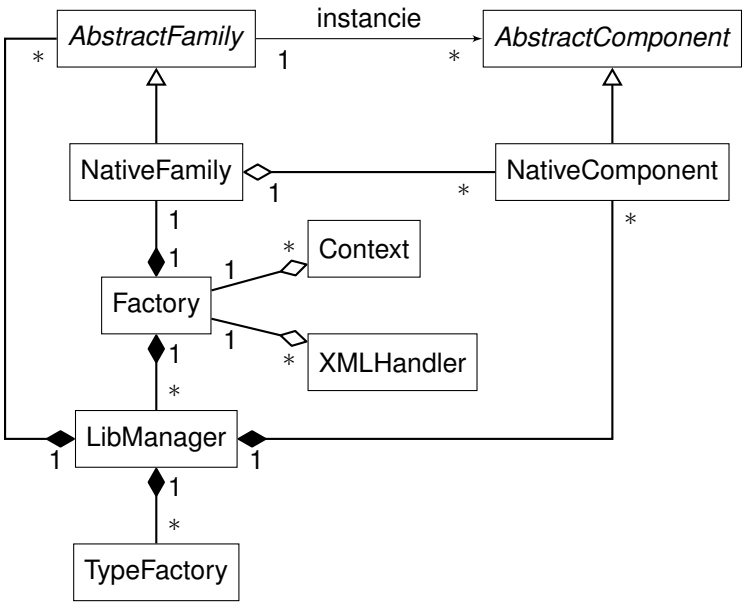


FIGURE 3.9 – Diagramme de classe de la fabrique

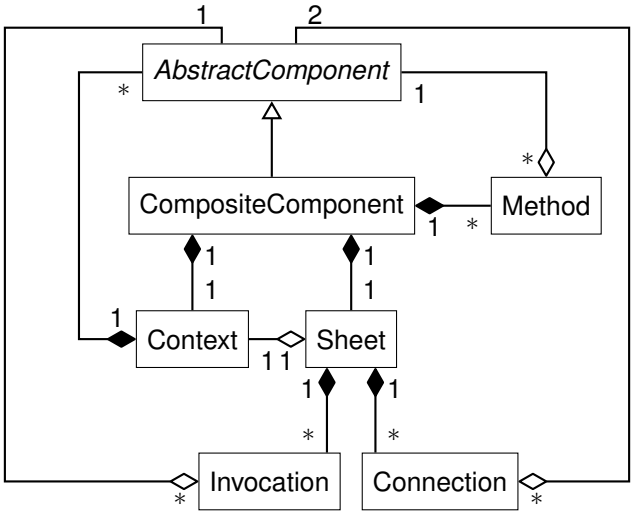


FIGURE 3.10 – Diagramme de classe de composition

des composants compilés (figure 3.11). Il instancie les composants et assure leur communication conformément à la description fournie. Le moteur d'exécution est composé de quatre éléments :

— un *parseur XML* : afin de lire et d'interpréter la description de l'application ;

— une *fabrique* : cet élément charge les familles, les fabriques de composants et de types, à partir des bibliothèques, les instancie et les détruit à la demande ;

— un *gestionnaire d'application* : les chargés d'instancier les processus, de leur fournir un contexte commun et de fermer l'application dès lors que tous les processus ont fini de s'exécuter ;

— un *contrôleur* par processus : chargé d'effectuer le basculement des feuilles lorsque ce dernier reçoit un jeton. Cela implique la reconfiguration du chemin de données pour le processus.

Lors du lancement de l'application, le moteur d'exécution lance, pour chaque processus, la feuille associée à son état initial. Ceci correspond aux étapes 3 à 6 du mécanisme de changement de feuilles détaillé à la page 77. L'arrêt est effectué lorsque, pour chaque processus, un état terminal est atteint dans son contrôleur.

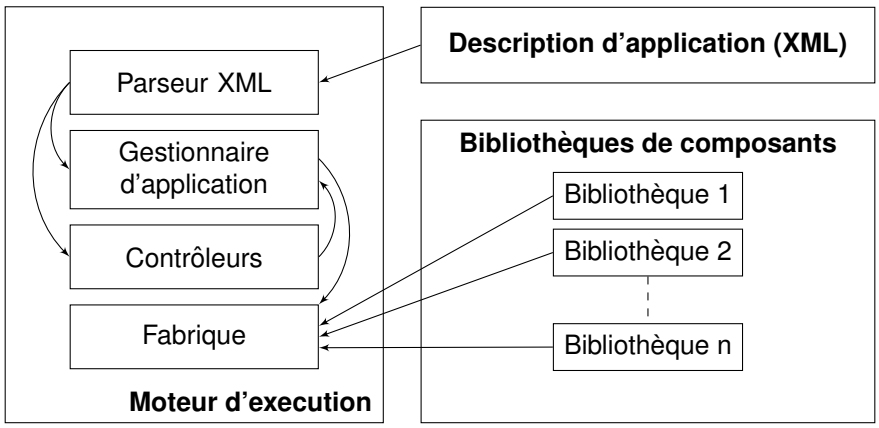


FIGURE 3.11 – Moteur d'exécution avec ses bibliothèques de composants périphériques

Remarque 3.3 : Différences entre variantes C++ et Javascript du *framework*

Le comportement du moteur décrit ici ne concerne que la variante C++ du *framework*. La version Javascript, conçue pour un environnement web, a un comportement similaire mais pas strictement identique. En particulier, elle charge non pas une description XML mais JSON (*JavaScript Object Notation*), plus adaptée à cet environnement. Les bibliothèques de composants se manifestent sous la forme de modules Javascript.

Pour terminer, ce moteur peut être exploité de deux manières. La première est sous la forme d'un exécutable qui charge la description et les bibliothèques, la deuxième est par le biais d'une bibliothèque intégrée à un projet de développement. Dans ce cas, une solution architecturale a été prévue (voir figure 3.12). Celle-ci permet à une application tierce (*ClientApp*) d'exploiter une *Interface* convenue à l'avance et implémentée par l'un des composants (ici *ComposantA*). Une référence à ce dernier est exportée par la classe (*ApplicationComponent*), ce qui permet à l'application tierce de la manipuler.

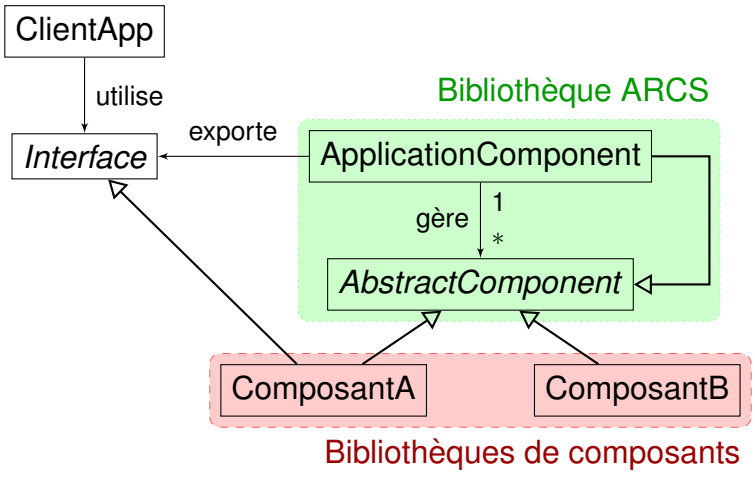


FIGURE 3.12 – Solution architecturale pour intégrer le moteur d'ARCS dans une application tierce.

Ce dernier mécanisme a été employé dans le cadre du projet RAXENV. En effet, ce projet ANR, concrétisant une collaboration entre plusieurs partenaires, a intégré différentes technologies. Le laboratoire IBISC, en charge de la partie de développement et d'intégration d'un capteur de localisation hybride a utilisé ARCS pour prototyper sa partie qui s'est interfacée avec trois capteurs de nature hétérogènes : une caméra, un GPS et une centrale inertielle. Le résultat de la localisation ainsi que les données brutes provenant des capteurs ont été remontées au moteur de rendu, nommé *Elkano*,

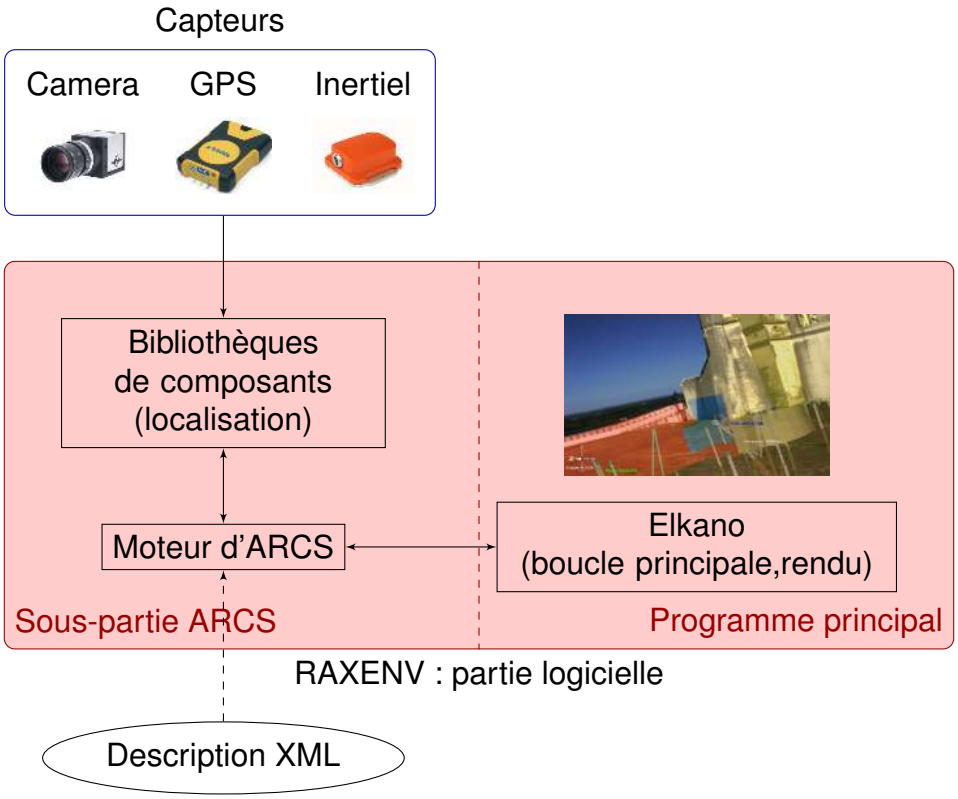


FIGURE 3.13 – Intégration du moteur d'ARCS dans la partie logicielle du projet RAXENV.

développé par le LABRI (Laboratoire Bordelais de Recherche en Informatique) qui a pris en charge la boucle principale du programme.

Outils du kit de développement

Comme nous l'avons écrit plus haut, ARCS est non seulement une bibliothèque et un *framework*, mais aussi un kit de développement fournissant des outils pour faciliter la réalisation d'une application de réalité augmentée tels que :

— Un programme en ligne de commande pour lancer le moteur d'ARCS à l'aide d'une description XML d'application ;

— Une interface graphique pour paramétrer le programme précédent ;

— Un outil pour créer et assister la compilation de bibliothèques de composants ;

— Un éditeur graphique qui génère une description XML d'application à partir d'un langage graphique (voir figure 3.14 page suivante).

Ce dernier outil est particulièrement intéressant : il permet de séparer, dans le processus de développement d'une application de réalité augmentée, les rôles des différents intervenants. Ainsi le développement de composants est bel et bien séparé de la création d'application en elle-même, les connaissances pour réaliser cette dernière tâche étant liées à l'intégration des composants sans nécessairement maîtriser les technologies et algorithmes associés. Nous atteignons ainsi l'un des objectifs que nous avions préalablement fixés.

3.1.3.5 Discussion

A l'issue de cette proposition d'architecture, trois points nous paraissent importants à discuter. Certains d'entre eux ouvrent des perspectives d'extension des travaux.

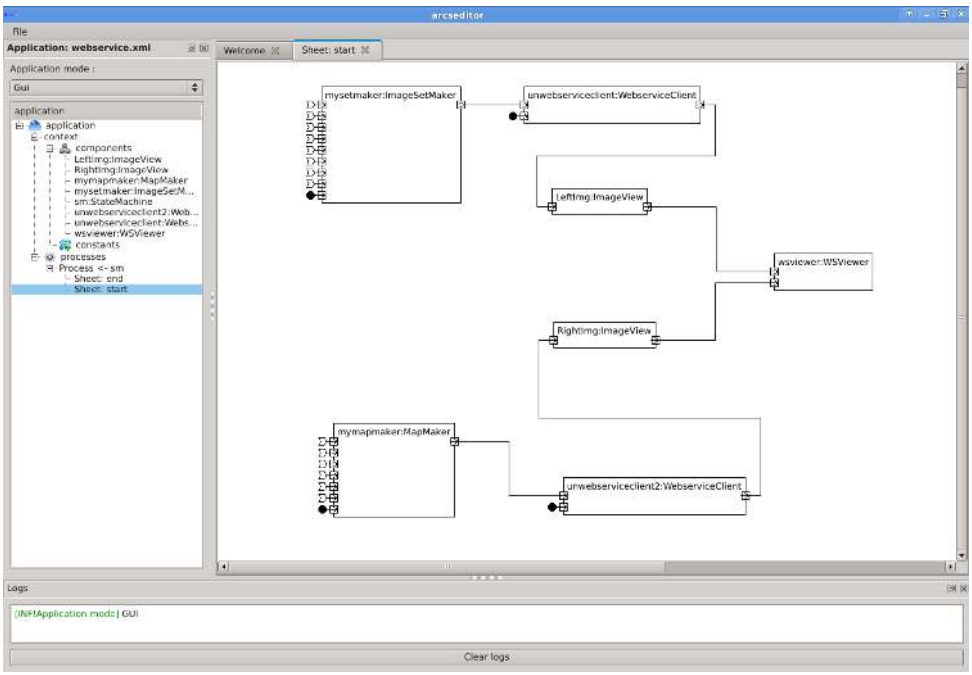


FIGURE 3.14 – Capture d'écran de l'éditeur graphique

Compromis performance / généricité

Tout d'abord, nous avons insisté sur les performances escomptées d'une telle architecture. Toutefois, certaines des autres caractéristiques recherchées telles que l'organisation en composants, la reconfigurabilité et une intégration étendue peuvent induire un effet contraire et dégrader les performances globales du système [MARLET 2011]. Nous nous sommes donc attachés dans un premier temps, à réaliser une implémentation la plus efficace possible en ayant recours à des techniques de mise en cache pour accélérer les accès aux composants et aux données utilisées lors des invocations, puis, dans un deuxième temps, à mesurer le surcoût associé aux mécanismes introduits.

Nous avons donc utilisé une plate-forme de test constitué d'un processeur Pentium IV cadencé à 3,2 GHz et de 1 Go de mémoire vive sur laquelle était installé le système d'exploitation Linux. L'objectif a été de déterminer 1) le coût lié à l'activation d'une feuille, qui établit des connections et des invocations et 2) le coût lié à la communication entre composants au travers du mécanisme signal/slot.

Nous avons donc lancé plusieurs séries de tests conçus pour mettre le moteur à rude épreuve. Ces derniers ont consisté à construire des feuilles avec un grand nombre d'invocations et de connections (jusqu'à 10 000 de chaque) et de mesurer les temps nécessaire pour les effectuer. De ces mesures, il en ressort que 14 microsecondes sont nécessaires pour effectuer une invocation, 12 microsecondes sont consommées pour mettre en place une connexion et enfin, moins d'1 microseconde est nécessaire pour effectuer une communication entre composants via le mécanisme signal/slot.

Si nous transposons ces résultats à une application telle que celle de RAXENV, le coût moyen estimé pour basculer entre deux feuilles est d'environ 650 microsecondes, ce qui est négligeable en comparaison des temps nécessaires pour traiter par exemple une image et rend ce compromis acceptable.

Flexibilité et exigences fonctionnelles

L'architecture finalement proposée est très fortement orientée sur les exigences non fonctionnelles des logiciels pour la réalité augmentée. Ceci est un aspect souhaitable en vue de renforcer sa flexibilité. Toutefois, ce qui va intéresser principalement un concepteur d'application, ce sont les exigences

fonctionnelles de la réalité augmentée pour lesquelles notre architecture n'apporte pas de réponse directe.

Une des solutions peut alors consister à mettre en place des modèles préconfigurés que l'utilisateur peut enrichir à sa guise pour développer des applications. De ce fait, l'architecture reste suffisamment flexible car la possibilité reste offerte de démarrer sans les modèles ou d'en adopter un afin de gagner du temps dans la phase de prototypage d'une application.

Évaluation d'une architecture logicielle

Lorsqu'une architecture logicielle est proposée, l'une des questions est de savoir comment l'évaluer. Cette question est ouverte car elle dépend des critères mis en place. Par exemple, elle pourra être évaluée par rapport à ses qualités intrinsèques, la façon dont elle remplit les exigences pour lesquelles elle a été conçue, par rapport à ses performances en terme de temps de calcul ou d'occupation de l'espace mémoire ou par les services qu'elle rend par exemple en terme de gain de temps de développement.

Nous avons apporté plus haut quelques éléments de réponse relatif à ses performances. De part sa construction, notre architecture satisfait les exigences ciblées. Pour les services rendus aux utilisateurs, nous n'avons pas d'évaluation formelle. La question reste donc à ce jour ouverte et offre des perspectives de travail par la suite. Des éléments de réponse sont toutefois apportés dans la partie suivante sur la distribution, en adaptant une méthodologie nommée *Software Performance Evaluation*.

Dans cette première partie, nous avons proposé une architecture pour les applications de réalité augmentée se basant, entre autres, sur des exigences de modularités, de réutilisabilité, de reconfigurabilité et de flexibilité. Cette dernière a été implémentée pour donner le jour au *framework* ARCS utilisé au sein de l'équipe IRA2 pour développer des applications de réalité augmentée. Il introduit une redistribution au niveau des rôles dans une équipe de développement, une partie étant dédié au développement de composants et l'autre à la conception et à l'écriture d'application, les niveaux de spécialisation étant différents. Enfin, nous avons vu que l'un des traits de cette architecture est qu'elle met en place des mécanismes de distribution des traitements à la demande afin de conserver ses performances. Nous allons voir comment nous avons mis au point cette solution particulière et ce que nous entendons exactement par distribution dans ce contexte.

3.2 Distribution et applications de réalité augmentée

L'état de l'art des solutions de distribution utilisées en réalité augmentée étant déjà couvert (section 3.1.2.1 page 69), nous nous attacherons à le compléter par un tour d'horizon des solutions génériques de distribution d'application existantes. Nous exposerons ensuite la solution proposée puis nous la compléterons avec une évaluation des performances requises pour une architecture de réalité augmentée distribuée. Enfin, nous verrons quelques applications de ces travaux.

3.2.1 Tour d'horizon des solutions de distribution

La définition 3.6 introduit le terme de *middleware* en tant que couche logicielle intermédiaire permettant la programmation d'un système distribué dont nous donnons la définition ci-dessous :

Définition 3.7 : Système distribué

Un système distribué est constitué d'une collection de processus distincts séparés spatialement et communiquant entre eux par échanges de messages [LAMPORT 1978a].

La finalité d'un tel système est d'exploiter les ressources d'un ensemble ou d'un parc de machine comme si elles n'en formaient qu'une de manière transparente pour l'utilisateur. Deux types de distribution existent :

— Le premier distribue des données. C'est le cas par exemple de la base de données associée au système de nommage des machines sur internet (le DNS, pour *Domain Name System*) ;

— Le deuxième distribue des traitements de manière à bénéficier d'un ensemble de ressources de calcul plus important pour une application. C'est sur ce deuxième point que nous nous positionnons.

A la section 3.1.2 page 69, nous avons vu que certaines architectures proposées pour la réalité augmentée incluaient des mécanismes de distribution des traitements. Nous allons tout d'abord présenter quelques technologies qui permettent de la réaliser. Puis nous reviendrons sur les techniques utilisées plus particulièrement en réalité augmentée.

3.2.1.1 Technologies pour la distribution

Un certain nombre de technologies génériques ont été mises au point pour la distribution. Deux paradigmes ont essentiellement émergé au cours des trente dernières années :

— Les intergiciels permettant l'appel de procédure, l'utilisation d'objets ou de composants à distance basés sur des protocoles de communication dédiés ;

— Les webservices qui exploitent internet et plus particulièrement le protocole de transfert des pages web en tant qu'infrastructure.

Avant de voir quelques exemples de ces technologies, nous allons discuter des principes communs à ces solutions.

3.2.1.1.1 Principes communs aux architectures de distribution

Les architectures de distribution s'appuient sur une brique fondamentale dans le contexte des réseaux informatiques : la communication client-serveur.

Définition 3.8 : Serveur

Le serveur est un processus fournissant des services à d'autres processus (usuellement appelés client). Dans la majorité des cas, le nombre de clients et l'identités de ces derniers n'est pas déterminée à l'avance par le serveur [GARLAN et SHAW 1993]. Par extension, la machine hébergeant le processus est également appelée serveur.

Définition 3.9 : Client

Le client est la contrepartie du serveur ; il s'agit d'un processus requérant des services auprès d'un ou plusieurs serveurs. L'identité des serveurs est connue des clients ou ils disposent, à tout le moins de capacités pour les découvrir. Par extension, le terme client s'applique aussi à la machine d'hébergement.

Le service fourni dans notre contexte est une ressource de traitement, appelée par le client et exécutée par le serveur. Qui plus est, la programmation des communications client-serveur faisant généralement appel à des primitives systèmes de bas niveau, dans le cadre d'une architecture de distribution, une surcouche est conçue pour masquer ces particularités et ce d'autant plus que le client peut-être, dans certains cas, développé à l'aide d'un langage de programmation différent de celui utilisé par le serveur, possiblement sur des architectures matérielles hétérogènes.

Les architectures distribuées reposent sur deux principes :

1. La description d'une interface commune entre le client et le serveur, le client faisant appel aux services décrits par l'interface et le serveur réalisant (fournissant) ces derniers ;
2. L'utilisation du patron de conception *Proxy* [GAMMA et al. 1993] qui permet de manipuler une entité logicielle se substituant à une autre (par exemple un objet local qui redirigera les requêtes vers un objet distant).

Dans certaines architectures de distribution, les objets servant de proxy ont un rôle davantage formalisé. Ainsi, celui situé du côté du client est appelé *souche* (en anglais *stub*) et celui du côté du serveur est appelé *squelette*. La figure 3.15 montre la manière dont les différents constituants de l'architecture sont agencés. L'interface est commune entre les parties client et serveur. Elle servira de référence pour l'implémentation de la souche, du squelette et de la ressource de traitement distante. Idéalement, la souche et le squelette sont automatiquement générés à partir de l'interface. L'entité requérante accédera à la partie client par le biais de la souche qui agit en tant que *proxy* et communique, par le biais de protocoles dédiés, avec le squelette, agissant lui-même comme *proxy* pour la ressource de traitement. La souche et le squelette prendront en charge les fonctionnalités de sérialisation et désérialisation, opérations visant à traduire les messages à échanger sous la forme de paquets d'octets transitant au travers du réseau.

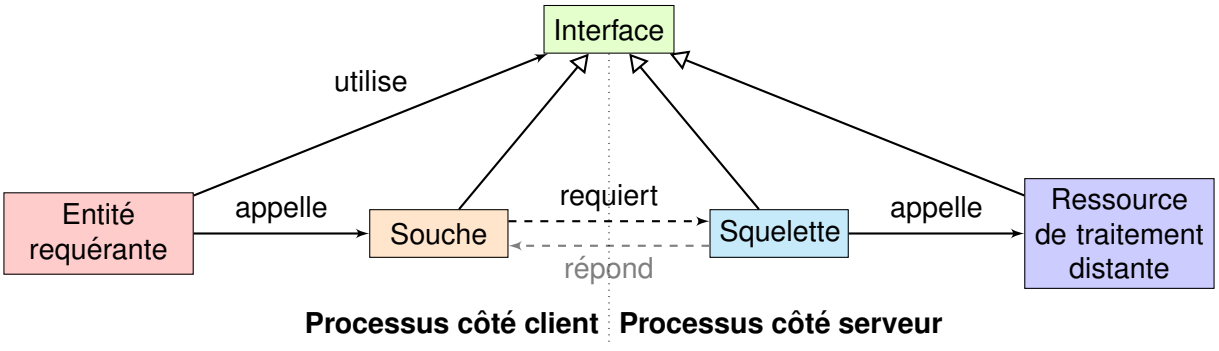


FIGURE 3.15 – Schéma de communication dans une application distribuée

3.2.1.1.2 Intergiciels génériques à protocoles dédiés

Dans cette section, nous présentons quelques intergiciels dédiés à la distribution d'application.

RPC, pour *Remote Procedure Call*, est un mécanisme permettant d'appeler une fonction distante. Depuis les bases jetées par Xerox [BIRRELL et NELSON 1984], ce type d'intergiciel a connu plusieurs implémentations dont la plus courante est ONC RPC (ONC signifiant *Open Network Computing*) originellement développée par Sun Microsystems.

Dans son principe de fonctionnement, les services fournis par le serveur, c'est à dire les fonctions appelables à distance, sont décrites dans le langage XDR (pour *eXternal Data Representation*). Ce dernier est traduit dans le langage ciblé par un programme spécial qui génère la souche et le squelette à partir de la description. Le développeur a alors, à sa charge, l'écriture de l'entité requérante et de la ressource de traitement distante.

Les services, à l'exécution, s'enregistrent sur un serveur d'enregistrement nommé *portmapper* qui permet d'indiquer aux clients le port à utiliser pour appeler une fonction à partir de son nom.

CORBA, pour *Common Object Request Broker Architecture*, est sans doute l'un des intergiciels orienté objet parmi les plus renommés. Sa spécification [OMG 1991] est activement maintenue depuis sa création en 1991 par l'*Object Management Group*. Elle a connu plusieurs évolution et sous-spécifications dont une concernant les composants logiciels.

Dans sa version actuelle, CORBA définit le langage IDL (*Interface Description Language*) pour spécifier des interfaces. Elles sont ensuite traduites pour générer les souches et les squelettes par des outils spécialisés vers la plupart des langages de programmation courants : C, C++, Java, Lisp, Ruby, Python...

Techniquement, CORBA définit aussi un bus de communication virtuel, régi par le protocole abstrait GIOP (*General Inter-ORB Protocol*) dont l'implémentation standard est IIOP (*Internet IOP*). Cette dernière s'appuie sur le protocole de transport TCP, couramment utilisé de nos jours sur le réseau Internet. Le bus de communication prendra en charge le transport des informations sérialisées. Le squelette et la souche assurent la sérialisation/désérialisation des données et la remontée d'erreur en cas de problème de communication. La figure 3.16 page suivante représente l'agencement de ces différents éléments dans l'architecture CORBA.

Java RMI (*Remote Method Invocation*) est un intergiciel spécifique à l'API Java. Il retranscrit les concepts de RPC en l'étendant à un modèle orienté objet [GROSSO 2001]. L'interface commune entre le client et le serveur est décrite en utilisant le langage Java. RMI a subit plusieurs évolution en parallèle de l'API Java :

— depuis la version 3, son protocole de communication est compatible avec IIOP (CORBA) ;

— depuis la version 5, le squelette et la souche peuvent être générés automatiquement à l'exécution.

Pour être utilisables, les services doivent se déclarer dans un registre qui associe la localisation du service à son nom.

Enfin, Java RMI possède un mécanisme supplémentaire par rapport à CORBA, c'est le support de la mobilité du code. En effet, le code compilé d'une souche peut être stocké sur un dépôt (en réalité, un site web) et importé par n'importe quelle machine virtuelle, ce qui offre des possibilités accrues en matière de distribution.

Apache Thrift est une solution développée par Facebook [SLEE et al. 2007] et maintenue par la fondation Apache. Elle repose sur la définition d'une interface dans le langage Thrift-IDL, ensuite traduite par un compilateur dans le langage de programmation de choix du développeur (Java, C++, Python, ...). Cette étape produit deux classes : une classe cliente et une interface pour le service dont le développeur aura à charge l'implémentation dans un fichier séparé. Théoriquement, Thrift supporte

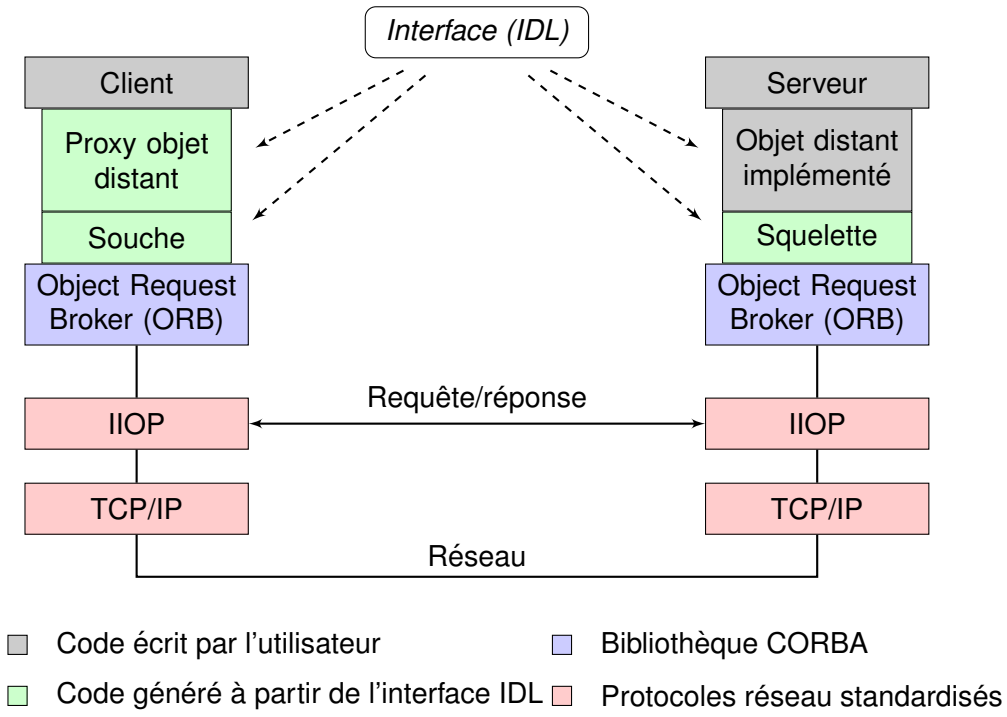


FIGURE 3.16 – Schéma directeur de l'architecture CORBA

plusieurs mode de communication entre le client et le serveur. Toutefois, dans sa version standard, il s'agit d'un protocole de communication binaire s'appuyant sur TCP.

Enfin, il convient de citer la longue lignée des intergiciels Microsoft qui commencent avec **DCOM** (pour *Distributed Component Object Model*) et dont le dernier représentant en date est **WCF** (*Windows Communication Foundation*), partie intégrante de Microsoft.Net [CHAPPELL 2009]. Dans ce cadre, une ressource distante nécessite trois composants : un classe décrivant le service, un processus hôte hébergeant le service et un ou plusieurs points d'échange. Pour chacun, trois éléments sont requis : une adresse le localisant, le type de combinaison de protocoles utilisés pour la communication et l'interface à laquelle il se conforme. La particularité de cet intergiciel réside dans la variété des protocoles supportés, ceux-ci allant d'une mouture binaire compact et propriétaire à des modes de communication que nous retrouverons dans les webservices au paragraphe suivant. Au besoin, les classes clientes peuvent être automatiquement générée à partir de l'interface spécifiée par les points d'échange.

Les intergiciels cités ici, à part quelques rares exceptions, utilisent des protocoles d'échange dédiés. En raison du développement d'internet et la complexité de l'intrication de ses différents sous-réseaux, il peut être intéressant d'utiliser des protocoles plus répandus, et donc plus à même d'être reconnus sans effort particulier par les pare-feux et autres dispositifs de sécurité, tel que le protocole HTTP (*Hyper Text Transfer Protocol*) à l'origine dédié aux contenus multimédias des pages Web. Nous verrons, dans la section suivante, que celui-ci peut être utilisé en tant que protocole de communication et sera l'un des fondements des webservices.

3.2.1.1.3 Les webservices

Il existe plusieurs définitions des webservices. Celle que nous donnerons est adaptée d'un rapport technique publié sous le couvert du W3C (*World Wide Web Consortium*) qui publie les standards en rapport avec le Web. Dans sa version originelle, la définition précise davantage les technologies utilisées. Toutefois, ces dernières ont évolué depuis la sortie de la définition en 2004.

Définition 3.10 : Webservice

Un webservice est un système logiciel conçu pour permettre des interactions de machines à machines au travers d'un réseau. Il possède une interface décrite dans un format compréhensible par la machine. Les autres systèmes interagissent avec le webservice en se conformant à sa description en utilisant des messages transportés et sérialisés via le protocole HTTP en conjonction avec d'autres standards associés au Web [HAAS et BROWN 2004].

Le W3C identifie deux catégories de Webservice comme étant les plus courantes :

— les webservices conformes à l'architecture REST, dont le principe est de manipuler des représentations sérialisées de ressources web par le biais de transactions sans état ou session ;

— les webservices arbitraires, ces derniers pouvant exposer un ensemble arbitraire d'opérations.

Les deux classes de webservice ont en commun d'utiliser des adresses de type URI (*Uniform Resource Identifier*), chaînes de caractères identifiant une ressource sur un réseau, des protocoles et des formats de sérialisation faisant partie des standards du web.

Les **formats de sérialisation** utilisés sont majoritairement **XML** (pour *eXtensible Markup Language*), langage à balises extensible à la demande, et **JSON** (pour *JavaScript Object Notation*), deux formats texte structurés et lisibles à la fois par la machine et l'homme.

Au niveau des **protocoles de communication**, ces derniers sont, au choix, **HTTP** (ou sa variante sécurisée – **HTTPS**), ce qui est suffisant pour les architectures REST, ou des protocoles s'appuyant sur ce dernier. Ainsi, **XML-RPC** [WINER 1999], est l'un des protocoles les plus anciens, permettant de réaliser des requêtes RPC auprès d'un webservice, formattées en utilisant le langage XML. Cette idée a, par la suite, été reprise dans le protocole **SOAP** (signifiant originellement *Simple Object Access Protocol*) [GUDGIN et al. 2007]. Les messages échangés par ce dernier sont structurés sous la forme d'une « enveloppe » contenant un en-tête (méta-données exploitables et modifiables par les noeuds acheminant le message) et un corps (contenant les données, soit en vue de réaliser un appel distant de procédure, soit pour réceptionner un document plus conséquent).

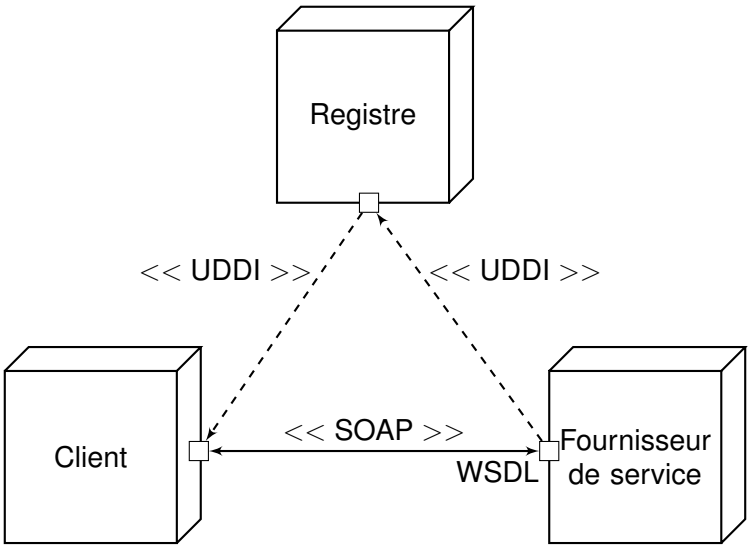


FIGURE 3.17 – Architecture des webservice arbitraire selon NEWCOMER et LOMOW 2004.

En ce qui concerne les **webservices arbitraires**, ils combinent souvent plusieurs technologies en vue d'être pleinement opérationnels. Pour se conformer à leur définition, ils doivent être capable de décrire leur interface. Cela peut-être réalisé à l'aide du format **WSDL** (pour *Web Services Description*

Language), un dialecte spécialisé de XML standardisé par le W3C. De plus, ces webservices peuvent s'enregistrer auprès de noeuds **UDDI** (pour *Universal Description Discovery and Integration*) qui est également un protocole de communication utilisant le langage XML comme format de sérialisation. Ainsi, les webservices arbitraires peuvent s'inscrire dans une infrastructure complexe intégrant plusieurs technologies. Une architecture basée sur ces derniers s'appuie sur des spécifications (SOAP, WSDL et UDDI) qui permettent l'interaction entre un client et un fournisseur de service, combinées avec des mécanismes potentiel de découverte de service [NEWCOMER et LOMOW 2004]. Le fournisseur publie une description WSDL de son service et le client y accède par le biais d'un registre UDDI, puis sollicite l'exécution d'une transaction auprès du fournisseur en échangeant des messages SOAP comme l'illustre la figure 3.17 page ci-contre. Il est à noter que le passage par le registre UDDI est optionnel dans le cas où le client a la connaissance *a priori* du webservice auquel il doit se connecter.

Pour terminer, l'architecture **REST** (pour *REpresentational State Transfer*) est une alternative que peuvent employer les webservices [FIELDING 2000]. Ils doivent alors se conformer à 6 règles :

1. Une séparation des responsabilités entre clients et serveurs (couplage faible) ;
2. Une interaction sans état persistant : les échanges entre clients et serveur doivent être indépendant de tout contexte conservé entre les deux ou de toute session ;
3. Une gestion du cache : chaque réponse du serveur doit indiquer si elle peut être mise en cache ou non afin d'améliorer les performances ;
4. Une uniformité de l'interface : les ressources possèdent un identifiant et une représentation clairement définis. Les messages sont auto-descriptifs et indiquent également, par le biais d'hyperliens, comment accéder aux états suivants de l'application ;
5. Une architecture en couche : les composants ne peuvent voir que les couches avec lesquelles ils interagissent directement ;
6. Un accès au code (facultatif) ; cela permet au serveur de transférer du code exécutable par le client afin d'étendre ses fonctionnalités. De ce fait, l'implémentation d'un client pourra se retrouver réduite d'autant.

Si REST est essentiellement une collection de principe architecturaux, en théorie découplée des protocoles de communication et des formats d'échange de données, il n'en demeure pas moins que ses implémentations sont basées essentiellement sur les capactités du protocole HTTP.

3.2.1.1.4 Discussion

Comme nous l'avons vu dans ce tour d'horizon des solutions pour la distribution des traitements, les intergiciels génériques et les webservices couvrent un panel important de solutions. En fonction du type d'application souhaitée, l'une ou l'autre des solutions sera davantage adaptée. Ainsi, la table 3.2 page suivante dresse une brève comparaison des caractéristiques discriminantes entre les deux. Nous retiendrons essentiellement que les webservices ont pour avantage principal de donner une importante scalabilité à une application (pour faire simple, l'application peut répondre au requêtes d'un nombre important d'utilisateurs) et l'usage de protocoles standards capables de passer au travers de parefeux sans configuration additionnelle. Les intergiciels génériques ont, quand à eux, une taille de message souvent réduite et optimisée, ce qui allège l'encombrement du réseau et rend la sérialisation / désérialisation potentiellement plus rapide.

Pour replacer les choses dans leur contexte, car nous nous intéressons avant tout aux applications de réalité augmentée, il est important de mettre en relation les architectures présentées à la section 3.1.2.1 page 69 et leurs solutions de distribution. Comme nous pouvons le voir à la table 3.3 page suivante, relativement peu de solutions s'appuient sur des intergiciels génériques et aucune sur des webservices. Les raisons qui dictent ces choix sont probablement les suivantes :

Critères de comparaison	Intergiciels génériques	Webservices
Modèle de données	Modèle objet	Message structuré
Couplage client-serveur	Réduit	Faible
Scalabilité	Limitée	Forte
Enveloppe	Réduite	Verbeux
Langage d'accès	Selon IDL disponible	Tout langage Web
Passage de pare-feux	Dépend du protocole	HTTP/HTTPS connus

TABLE 3.2 – Comparaison entre intergiciels génériques et webservices

1. Les applications développées avec ces architectures s'adressant *in fine* à un nombre restreint d'utilisateurs, l'aspect scalabilité, favorable aux webservices, n'a donc pas été considéré comme un critère déterminant ;
2. Les performances en termes de temps de calcul constituent un point important pour les applications de réalité augmentées, en raison des algorithmes utilisés (traitement d'image, reconnaissance de primitives...) ; un protocole de communication plus verbeux et complexe à lire coûte davantage de temps de calcul également, ce qui dissuade leur utilisation ;
3. A moins de développer des serveurs http sur mesure, les webservices fonctionnent sur un mode requête/réponse, ce qui peut nécessité une adaptation importante pour des architectures où les traitements sont organisés en flux de données.

Architecture	Solution de distribution
TINMITH	Sur mesure, sérialisation XML
VHD++	C++ RMI
DWARF	CORBA orienté composant
MORGAN	CORBA
ARTIFICE	Unity Game Objects
STUDIERSTUBE	Distributed Inventor
AVANGO	Sur mesure
VARU	Sur mesure, articulé autours de 3 services prédéfinis par noeuds
MRSS	Sur mesure, articulé autours de 4 services prédéfinis par noeuds

TABLE 3.3 – Solutions de distribution proposées dans les architectures de réalité augmentée

En conséquence, notre positionnement, par rapport à la distribution, sera le suivant :

Positionnement 3.2 : Architecture distribuée pour la RA

Notre architecture pour la distribution doit offrir certaines garanties de performance en cours d'utilisation. Pour cette raison, nous nous tournons vers une solution similaire aux intergiciels génériques.
Toutefois, les webservices restent intéressants pour délivrer et manipuler des données donc nous verrons également comment intégrer notre architecture dans un contexte web. Cela constitue un premier pas vers les webservices, possibilité sous explorée à l'époque où nos travaux sur le sujet a commencé.

3.2.2 Extension de l'architecture proposée

Les extensions que nous proposons pour notre architecture seront communiquées en deux temps :

1. dans un premier temps, nous lui conférons un caractère distribué ;
2. dans un deuxième temps, nous explorons l'utilisation potentielle des webservices dans les applications de réalité augmentée et esquissons quelques solutions architecturales.

3.2.2.1 Proposition d'un modèle d'intergiciel

Comme nous l'avons vu, certaines solutions architecturales développées s'appuient sur des intergiciels génériques pour gérer la distribution. C'est par exemple le cas de DWARF ou de MORGAN qui s'appuient sur CORBA. La plupart implémentent un choix différent, qui est celui de l'intergiciel développé sur mesure.

Dans notre cas, nous avons choisi une architecture spécifique. En effet, nous souhaitions préserver le paradigme de communication signal/slot et une organisation des traitements structurées autour des flux de données entre composant. De cela découle un rôle ambivalent pour nos composants, c'est à dire qu'ils peuvent être à la fois clients et serveurs. Cela constitue une originalité forte de notre architecture telle que nous avons pu la présenter dans [CHOUITEN et al. 2011] et [DIDIER et al. 2012].

Au niveau des choix effectués, nous adoptons une architecture où la **description d'une application est centralisée**. Cette dernière est localisée sur un des noeuds qui aura le rôle d'**application maître** et qui établira les connexions entre les différents services, appelés dans ce contexte des **composants esclaves**. Elles respectent la sémantique signal/slot et se comportent de manière similaire aux appels distants de procédures (RPC). À l'exécution, 3 types de connexion signal/slot surviennent :

— les connexions maître – esclave,

— les connexions esclave – esclave,

— les connexions esclave – maître.

Chaque noeud (application ou composant distant) peut, suivant la situation, se comporter en service, c'est à dire offrir ses *slots* en vue de répondre à un signal. Au niveau de l'application, un composant distant est manipulé par le biais d'un intermédiaire local qui réalisera la sérialisation et la désérialisation des données.

Enfin, dans la description d'une application, nous considérons les composants distants comme des composants standards. C'est à dire que le développeur d'une application n'a pas à se soucier des mécanismes réseaux qui sous-tendent la distribution tout comme un développeur de composant peut développer ses composants indépendamment d'un contexte réseau.

Solution architecturale

Pour mettre au point une solution pour notre architecture, nous exploitons les capacités étendues d'intégration de notre *framework* (voir section 3.1.3.3 page 77).

Comme nous pouvons le voir à la figure 3.18 page suivante, nous ajoutons une nouvelle famille de composants pour gérer la distribution des traitements (*NetworkFamily*). Elle instancie des composants de type *NetworkComponent*. Ils représentent un composant distant capable de générer, à la volée, des objets en charge de la sérialisation (*NetworkProxySlot* – qui se comporte également comme un client) et de la désérialisation (*NetworkProxySignal*) des données pour chaque signal et slot distant. Le composant se conforme au patron de conception *Proxy*. Pour parachever l'implémentation, l'application locale maître et les composants distants esclaves embarquent un objet dont le comportement est celui d'un service (*NetworkServer*). Enfin, du côté des composants esclaves, l'objet de type *NetworkServer* couplé au véritable composant (à noter que ce composant peut-lui même être de nature composite, ce

qui offre une granularité variable), génère à la volée les objets en charge de la sérialisation et de la désérialisation et comporte un objet (*NetworkConfigurator*) qui associe une adresse sur le réseau au composant distant.

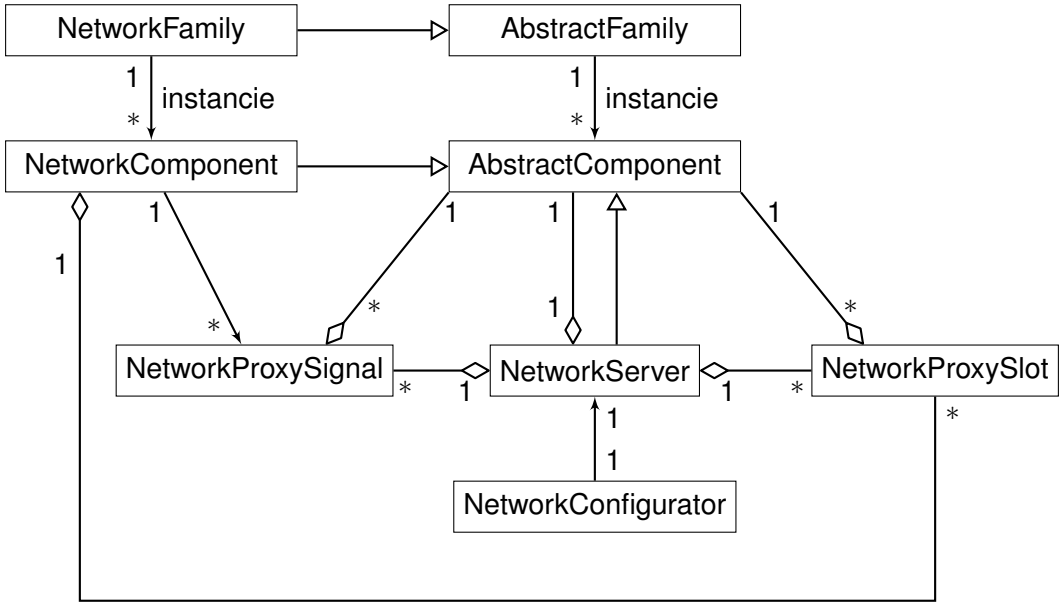


FIGURE 3.18 – Diagramme de classe des concepts architecturaux d'ARCS.

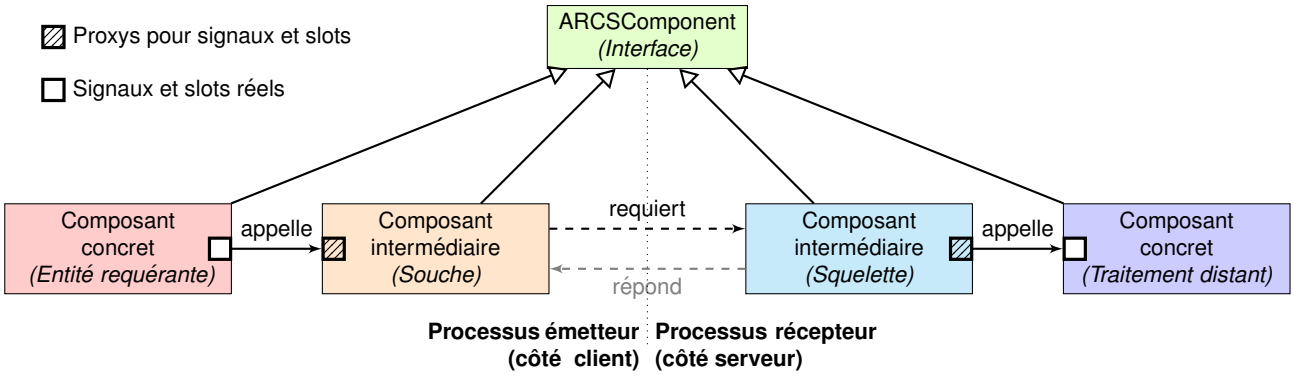
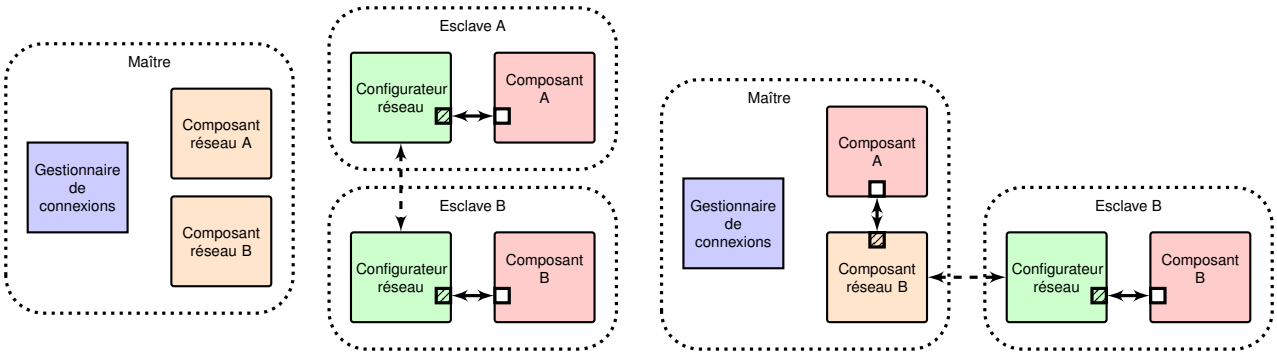


FIGURE 3.19 – Principe général de la distribution transposé à notre intergiciel

Comportement général

La transposition de la figure 3.15 page 89 à notre intergiciel produit la figure 3.19. Les composants intermédiaires prennent en charge les fonctionnalités dévolues à la souche et au squelette et génèrent automatiquement les *proxy* correspondants aux signaux et aux slots utilisés dans l'application. La nature du composant intermédiaire dépend du type de noeud. S'il est maître, il sera un composant réseau (*NetworkComponent*) ; à l'inverse, il sera un configurateur réseau (*NetworkConfigurator*).

D'autres particularités naissent de la description centralisée des applications et des échanges de données décentralisés. Ainsi, le noeud maître gère les descriptions des connexions et instancie des composants (*NetworkComponents*) représentant leurs contre-parties distantes. Cela permet ainsi de les contrôler à partir du noeud maître. Ceci permet au maître de gérer des connexions distantes entre deux noeuds (voir figure 3.20(a) page ci-contre) sans pour autant que les données ne transitent par le noeud maître. Dans le cas d'une connexion entre un maître et un esclave, nous retrouverons le schéma de communication client-serveur (voir figure 3.20(b) page suivante).



(a) Connexion esclave – esclave (b) Connexion maître – esclave

FIGURE 3.20 – Types de connections

Cette solution nous a permis d'écrire des applications distribuées pour la réalité augmentée tout en préservant la sémantique de la connexion signal/slot pour nos composants comme nous le présenterons à la section 3.2.4 page 104. L'un des intérêts de ce mode de fonctionnement est que le développeur de composants n'a pas à se soucier du contexte d'utilisation de ce dernier ni d'écrire spécifiquement du code pour pouvoir communiquer au travers du réseau. Un tel mécanisme permet, en effet, de distribuer n'importe quel composant écrit au préalable.

3.2.2.2 Webservices et applications de réalité augmentée

Cette partie contient essentiellement quelques considérations concernant le couplage du moteur d'exécution aux *webservices*. Nous envisageons deux comportements différents pour une application prototypée avec ARCS :

— l'application est client d'un *webservice* ;

— l'application est un *webservice*.

Application cliente

La gestion d'un client repose sur les mêmes principes que la gestion de la distribution. Il nous est possible de réaliser un composant générique capable de s'interfacer avec un *webservice* en utilisant le protocole HTTP, à minima un encodage générique des méthodes proposées par le protocole (méthodes *GET* et *POST*) et des formats de sérialisation répandus tels que XML et JSON. Une bonne partie de ces fonctionnalités sont disponibles via la bibliothèque Qt sur laquelle est développé ARCS. Dès lors, nous avons pu développer un composant générique appelé *ARCSServiceClient* (héritant de *ARCSAbstractComponent*) qui permet de traduire les signaux émis par les autres composants en requêtes vers un *webservice* et de retranscrire les réponses en signaux à destination des autres composants.

Application en tant que service

Nous avons développé des composants dédiés pour chaque type de service que nous avions à mettre en place. Ils ont néanmoins un fonctionnement commun :

— ils adoptent le comportement d'un serveur Web ;

— ils traitent des données reçues au format XML ou JSON ;

— ils sont capable de sérialiser/désérialiser ces requêtes et de les coupler à des signaux ou à des slots.

De plus, pour certaines applications, nous « externalisons » la gestion de certaines fonctionnalités. Ainsi, dans le cas où nous avons besoin de *streaming* vidéo (fonctionnalité permettant de diffuser une vidéo au travers d'un réseau), notre *webservice* communique l'adresse où se trouve ce flux.

A ce stade, nous souhaiterions élargir la réflexion et faire en sorte que nous puissions adapter à la volée des composants pour les transformer en *webservices* et exporter leur description dans un format standardisé.

3.2.3 Évaluation des performances

Notre architecture mise au point, nous souhaitions pouvoir évaluer ses performances. Nous avons constaté l'absence de méthodologie permettant cette évaluation pour des systèmes complexes tels que ceux de réalité augmentée.

Positionnement 3.3 : Méthodologie d'évaluation des performances

Nous proposons dans cette section une méthode originale d'évaluation des performances des applications de réalité augmentée distribuées qui prend en compte les spécificités de ce type de système. Cette dernière s'appuie sur la méthode générique SPE (*Software Performance Engineering*) et introduit les spécificités associées à la Réalité Augmentée et aux enjeux de la distribution de ce type d'application.

Dans les grandes lignes, la méthodologie proposée s'appuie sur trois piliers :

— une architecture fonctionnelle de référence. Nous avons choisi celle de [MacWilliams et al. 2004], elle même issue des retours d'expérience de la communauté de réalité augmentée ;

— la détermination des critères de performance liés à la qualité d'une expérience de réalité augmentée (critères qualitatifs et métriques) ;

— une méthodologie générique d'évaluation d'architectures logicielles appelée SPE (pour *Software Performance Engineering*) qui tient compte de l'aspect distribué des applications.

Les objectifs de cette méthodologie sont multiples :

— savoir si un environnement de développement supporte les performances requises pour une application donnée et les évaluer une fois cette dernière développée ;

— déterminer, pour un environnement donné, quelle architecture offre les meilleures performances ;

— aboutir à des recommandations concernant l'environnement de développement ou l'architecture à mettre en place pour améliorer les performances.

3.2.3.1 Architecture de référence

Pour l'architecture de référence, nous nous appuyons sur la structuration en paquetage proposée par MacWilliams et al. 2004 (voir figure 3.21 page suivante). Cette dernière identifie 6 regroupements fonctionnels dans les applications de réalité augmentée :

— Le *tracking* prend en charge la localisation du système de RA par rapport à son environnement ;

— Le modèle monde (*World model*) décrit la connaissance *a priori* de l'environnement. Ce dernier peut-être enrichi en ligne, ce qui est le cas dans les applications de type SLAM ;

— La partie présentation se charge du formattage des données pour les restituer à l'utilisateur ;

— La partie interaction interprète les actions utilisateurs sur l'interface homme-machine, afin de modifier le comportement du système ;

— Le contexte mémorise l'état de l'application et pilote la partie *tracking* afin d'optimiser la partie reconnaissance de l'objet suivi ;

— Enfin, l'application sert de chef d'orchestre et coordonne les différents coeurs fonctionnels.

Cette architecture, transposée à ARCS, voit disparaître les paquetages. En revanche, les différentes fonctionnalités sont assurées par des composants ainsi que des parties du moteur d'exécution. Sa transposition peut se manifester sous la forme décrite à la figure 3.22 page suivante, comme nous l'avons montré dans CHOUITEN et al. 2012a.

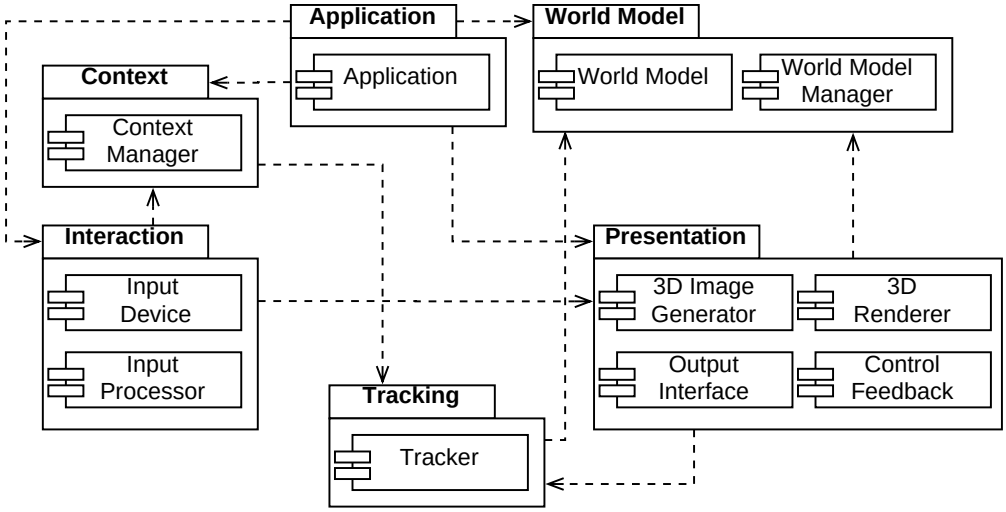


FIGURE 3.21 – Architecture fonctionnelle proposée par MACWILLIAMS et al. 2004

3.2.3.2 Critères d'évaluation

L'évaluation d'une architecture distribuée s'effectue suivant deux types de critères. Les uns, qualitatifs, permettent de vérifier que l'architecture est compatible avec certaines exigences fonctionnelles. Les autres, quantitatifs, permettent d'évaluer plus finement les performances ainsi que d'édicter des recommandations quant aux pratiques à employer pour optimiser ces dernières.

Critères qualitatifs

Les critères que nous retenons sont employés pour évaluer les intergiciels génériques [TOKUNAGA et al. 2003] ainsi que sur certaines exigences spécifiques à la convergence numérique et aux applications de RA. Rapidement, ces critères sont les suivants :

1. *Rôle ambivalent des composants distribués* : de part la possibilité de les déployer comme serveur ou comme client, cela facilite la conception des applications, augmente la scalabilité de l'architecture et flexibilise le fonctionnement de l'application. Cette fonctionnalité facultative est présente dans ARCS ;
2. *Haut niveau d'abstraction* : au sens de la programmation orientée objet, il implique le découplage entre interfaces implémentations afin de disposer de plusieurs alternatives pour un comportement donné. Dans notre architecture, ce rôle est assuré par les familles de composants ;
3. *Transparence et sensibilité au contexte* : cette notion implique que le développeur d'application n'ait plus à traiter d'exigences liées à la distribution telles que la sérialisation/désérialisation ou la prise en charge de l'écriture de code spécifique aux communications sur un réseau. Dans le même temps, ce comportement doit être adaptable pour tenir compte des spécificités de l'application. La solution décrite en 3.2.2.1 page 95 abonde dans le sens de ce critère ;

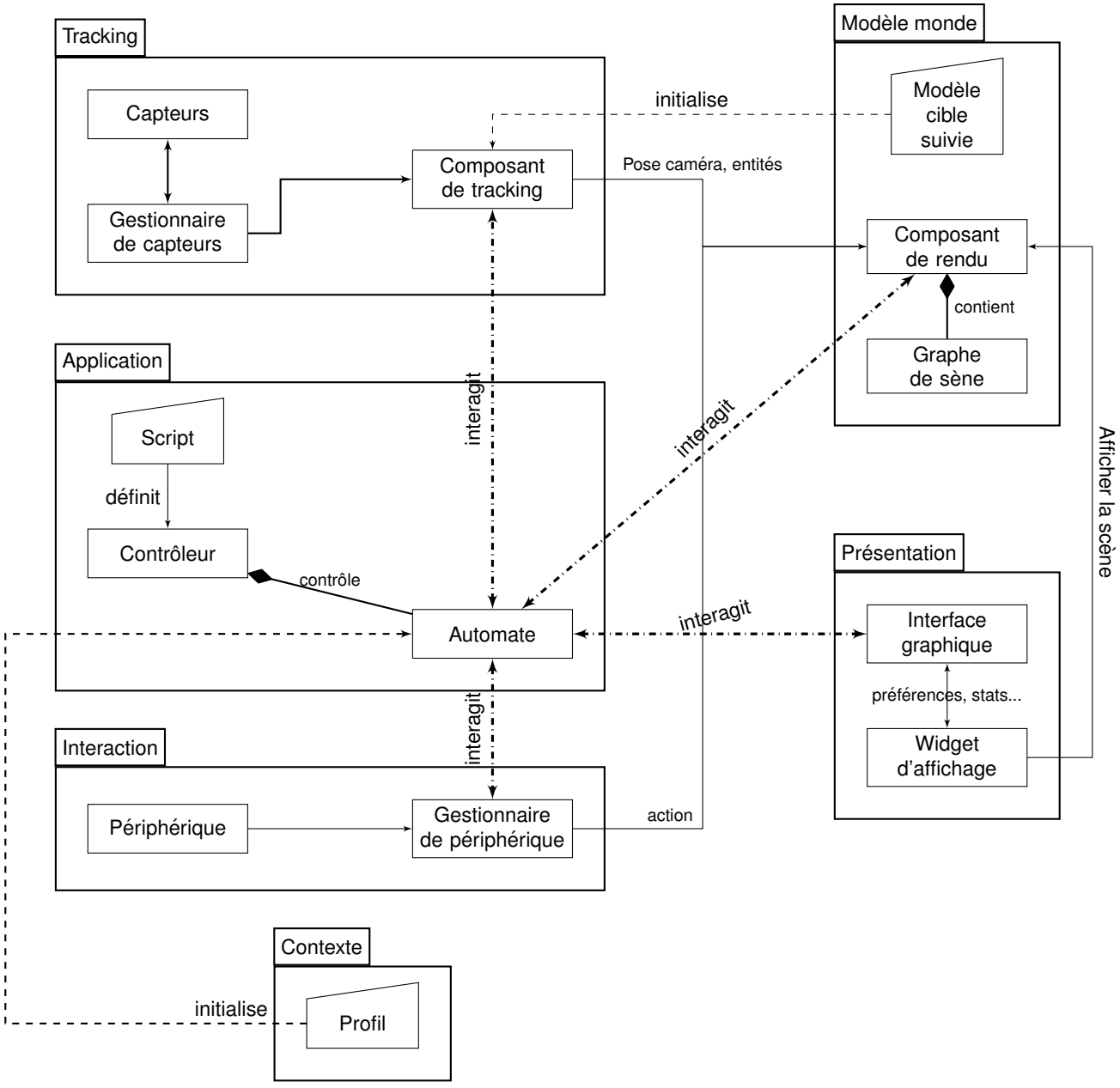


FIGURE 3.22 – Architecture fonctionnelle de MacWilliams transposée à une application développée avec ARCS

4. *Convention de nommage sémantique* : elle devrait indiquer la localisation d'un composant sur le réseau, si possible de manière compréhensible pour l'humain. Pour ce faire, nous empruntons une formalisation similaire aux URL (par exemple `arcs://localhost:2030/` qui indique le nom de machine ainsi que le port sur lequel se connecter pour accéder à un composant distant). Cet aspect est sans doute celui sur lequel nous pouvons progresser, l'aspect sémantique étant absent ;
5. *Facilité de programmation et réutilisabilité* : il est difficile de répondre complétement à ce critère subjectif. Dans le cas de notre architecture, un nombre important de composants ont été développés et sont prêts à l'emploi. Un pas supplémentaire pourrait être franchi en proposant des canevas d'application pré-remplis et paramétrables. Seules les parties spécifiques à son application seraient à développer. Toutefois, nous rentrons dans des considérations qui relèvent plus de l'ingénierie et de l'industrialisation d'une solution logicielle ;
6. *Flexibilité et interopérabilité* : la première fait référence à la faculté d'un environnement logiciel à supporter différents scénarios applicatifs s'exécutant sur des plates-formes matérielles hétérogènes. La deuxième est identifiée à la capacité de collaboration ou d'intégration d'autres applications et/ou composants logiciels appartenant à d'autres environnements de développement. ARCS, étant basé sur Qt, un *framework* multi-plateformes (Windows, Linux, MacOS, Android,...), cela assure un large support de configurations matérielles et de systèmes d'exploitation. Ce critère est pris en compte et a fait l'objet de la section 3.1.3.3 page 77 ;
7. *Multicast* : Il s'agit de la diffusion des informations au travers d'un réseau via un système d'abonnement. Cette technique permet d'optimiser la bande passante, notamment quand plusieurs clients doivent accéder à la même source d'information en même temps. Notre architecture ne supporte pas ce type de diffusion. Toutefois, certains composants spécifiques, notamment pour la diffusion de vidéo au travers du réseau implémentent cette technique.

Critères quantitatifs

Ces critères varient en fonction de l'application ciblée. Nous identifions ceux qui nous semblent importants dans le cas de réalité augmentée distribuée, à savoir :

1. *Temps-réel interactif* : par définition (voir définition 2.4 page 50), il s'agit d'un des critères intrinsèques à la réalité augmentée. Cela implique, au niveau communication, que ces dernières s'effectuent dans des délais raisonnables qui sont à la fois dépendants du volume et du temps de latence de la transmission ;
2. *Scalabilité* : cette spécificité touche les applications massivement multi-utilisateurs. Si les applications professionnelles visent quelques dizaines de collaborateurs, les jeux vidéos opèrent à une échelle plus vaste de l'ordre de la dizaine de milliers d'utilisateur à l'instar d'Ingress [NIANTIC 2012] ou de Pokemon Go [NIANTIC 2015]. Cela a alors des répercutions techniques sur l'architecture et la politique de répartition de charge (*load balancing*) ;
3. *Tolérance aux pannes* : il s'agit de la capacité du système à réagir et à continuer de fonctionner en cas de défaillance d'une partie de ce dernier. L'un des moyens pour y parvenir est de proposer des modes dégradés.

Ces critères doivent donc être considérés lors de la conception de l'architecture logicielle.

3.2.3.3 Scénarios de répartition

Dans l'objectif d'évaluer les performances requises pour une application de réalité augmentée, nous devons identifier les scénarios de distribution possibles. C'est, par ailleurs, la première étape de la méthodologie *Software Performance Engineering* (SPE) proposée par SMITH et WILLIAMS 2002. Elle revient à identifier quels composants seront déployés sur un site distant (notés R pour *remote*),

sur le dispositif terminal de l'utilisateur (notés L pour *local*) ou distribués sur les deux sites (notés B – *both*).

Les composants distribuables appartiennent à trois blocs fonctionnels en RA : acquisition de données, traitement des données acquises et visualisation. Cela concerne donc essentiellement les paquetages *tracking*, *interaction* et *présentation* de l'architecture de MacWilliams.

L'objectif étant d'identifier les scénarios possibles, et au vu des critères retenus, 27 alternatives se présentent théoriquement. Toutefois, certaines ne sont pas distribuées (si tout se déroule sur le site distant ou sur le terminal local). De plus, certains scénarios impliquent que les données qui transitent via le réseau soient de la même nature et peuvent donc être regroupés, en tenant compte des variantes symétriques du point de vue des échanges (ainsi, si dans un scénario, un ensemble de données passe du local au distant et dans un autre scénario, ce même ensemble utilise le chemin inverse, les deux alternatives seront regroupées). Au final, seuls 8 scénarios différent du point de vue des performances de distribution comme nous pouvons le constater à la table 3.4.

	Acquisition L			Acquisition R			Acquisition B		
	L Trait.	R Trait.	B Trait.	L Trait.	R Trait.	B Trait.	L Trait.	R Trait.	B Trait.
L Vis.	✖	2	3	1	5	4	1	2	7
R Vis.	5	1	4	2	✖	3	2	1	7
B Vis.	5	2	6	2	5	6	2	2	8

TABLE 3.4 – Scénarios de distribution des applications de réalité augmentée

3.2.3.4 Métriques et indicateurs de performance

L'enjeu est d'identifier, à fonctionnalités équivalentes, quels sont les scénarios à privilégier. Pour cela, nous devons nous attacher à quantifier les données échangées lorsqu'un scénario est appliqué. Nous nous appuierons sur une représentation simplifiée du *workflow* des applications de réalité augmentée. Celui-ci se présente sous la forme décrite à la figure 3.23 et met en évidence les étapes entre lesquelles des données sont susceptibles d'être échangées au travers d'un réseau.

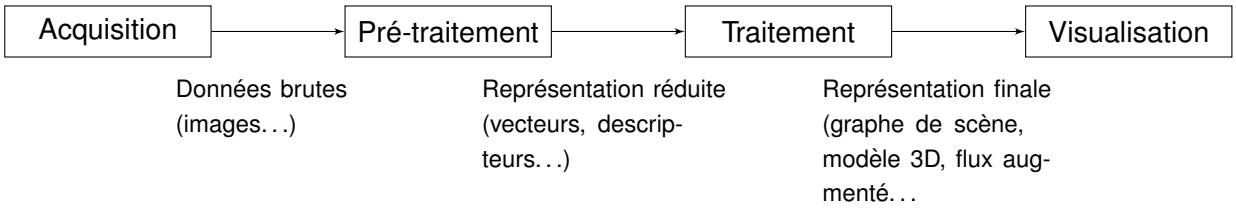


FIGURE 3.23 – *Workflow* d'une application de réalité augmentée

Le débit requis pour les données à transmettre est exprimé sous la forme du *goodput*. Il mesure la bande passante requise par les données applicatives en faisant abstraction des en-têtes requis par les protocoles de transport de l'information. La table 3.5 page ci-contre fournit des indications volumétriques pour quelques capteurs usuellement exploités dans les applications de réalité augmentée, à savoir la caméra (stéréo ou non), le GPS et la centrale inertielle (CI). Ces dernières seront fonction de la fréquence d'acquisition des capteurs et de la taille unitaire de leur trames de données, ce qui permet d'indiquer la fourchette de variation possible.

Un autre indicateur est le temps de latence de bout en bout de l'application, c'est à dire le temps nécessaire pour accomplir l'ensemble des étapes du *workflow* présenté plus haut. Les exigences à propos de cette durée varient en fonction de l'application et du type de rendu (visuel ou haptique, par

	Faible débit			Haut débit		
	Fréquence	Taille unitaire	*Goodput* requis	Fréquence	Taille unitaire	*Goodput* requis
Caméra	15Hz	20,7 ko (320x200, JPG90)	309,9 ko/s	60Hz (stereo)	347,4 ko (1600×1200)	20,35 Mo/s
GPS	1Hz	<0,5 ko	0,5 ko/s	100 Hz	<0,5 ko	<0,05 Mo/s
CI	20 Hz	<1 ko	20 ko/s	100 Hz	<1 ko	0,1 Mo/s

TABLE 3.5 – *Goodput* requis par quelques capteurs usuels

exemple) effectué. Toutefois, des délais supérieurs à 40 ms entre l'acquisition des données et le rendu visuel commencent à dégrader perceptiblement la qualité de l'interaction [ELLIS et al. 1997].

En ce qui concerne les pré-traitements, l'idée est de transmettre, non pas les images, mais des représentations intermédiaires, afin de permettre le calcul de la pose (la position et l'orientation) du système par rapport à son environnement. C'est par exemple le cas si l'on s'appuie sur des représentations intermédiaires telles que les descripteurs locaux (SIFT, SURF, etc).

Dans un contexte où les traitements sont distribués sur un parc hétérogène d'unités de calcul, la localisation du traitement a une influence sur la latence globale. Pour cela, nous introduisons deux indicateurs supplémentaires, L_P et R_P qui représentent la durée totale de l'opération de pré-traitement et d'envoi des données, selon que les pré-traitements soient effectués en local ou à distance et qui sont obtenus comme suit :
L_P = *(taille des données de sortie / goodput) + temps de pré-traitement en local*
R_P = *(taille des données de entrée / goodput) + temps de pré-traitement sur site distant*

Enfin, dans un contexte de transmission des informations dans un réseau, nous devons aussi estimer le débit requis pour atteindre le *goodput*. Toutefois, celui-ci dépend des choix opérés au niveau du réseau physique et des protocoles mis en oeuvre. A titre d'exemple, avec un MSS (*Message Segment Size*, que l'on peut qualifier de taille de paquet de données) standard de 563 octets et un tête TCP (*Transmission Control Protocol*, protocole de transmission de données largement utilisé pour Internet), sans option, de 20 octets, ajouté lui-même à un en-tête IP (*Internet Protocol*) de 20 octets, nous obtenons un ratio de 93% de données utiles transportées.

Ces considérations apportent ensuite un éclairage sur les scénarios de distributions à privilégier.

3.2.3.5 De la sélection de scénario

La sélection d'un scénario dépendra du type d'application visée. Aussi, nous ne donnerons, pour clore cette partie, que des recommandations générales sur les scénarios à privilégier dans le cas où plusieurs alternatives sont possibles.

Par exemple, dans le cas où l'acquisition exploite une image de taille 640×480 pixels, son poids est de $921,6$ kO sans compression. La taille d'un descripteur SIFT pourra faire jusqu'à 1 kO en fonction de l'implémentation qui en est faite, ce qui veut dire qu'un poids équivalent au données brutes est atteint aux alentours d'un millier de points d'intérêt détectés et décrits. Cela est approximativement le double du nombre de points détectés dans l'image considéré. Dans la plupart des cas, si la capacité de calcul des deux sites diffèrent peu, il est intéressant de conserver le pré-traitement en local. Dans ce cas, le scénario 4 sera à privilégier par rapport au scénario 1.

Nos indicateurs L_P et R_P aident à choisir un scénario. Ainsi, si $L_P > R_P$, nous privilégierons le scénario 1, sinon le 4. Si le site hébergeant dispose de meilleures capacités de calcul que le site de visualisation, et si l'algorithme n'est pas parallélisable entre les deux sites, le scénario 5 peut être préféré en fonction des volumes de données requis pour la visualisation. En revanche, si l'algorithme est parallélisable, le scénario 3 devrait être préféré.

3.2.4 Applications de la distribution

Plusieurs applications ont été développées en exploitant les concepts énoncés plus haut en rapport avec la distribution. Une description exhaustive de ces dernières pourra être trouvée dans CHOUITEN 2013. Elles nous ont permis de valider :

— notre protocole *ad-hoc* de distribution pour un processus de localisation et de cartographie simultanée (SLAM) utilisant une seule caméra, où les étapes de la capture à l'appariement de points s'effectuent en local alors que le reste des traitements sont effectués sur un site distant (figure 3.24) ;

— une application écrite avec ARCS en tant que cliente de web-services standardisés, à l'instar de celui de Google Maps (figure 3.25 page ci-contre), ou non tel que le web-service des reconstruction 3D mis à disposition par le *Center for Machine Perception* de l'université technique de Prague (figure 3.26 page suivante) ;

— un service écrit avec ARCS *ad-hoc* pour une application externe que nous allons présenter en détail et qui a fait l'objet d'une valorisation au travers de plusieurs publications [CHOUITEN et al. 2012a ; CHOUITEN et al. 2014].

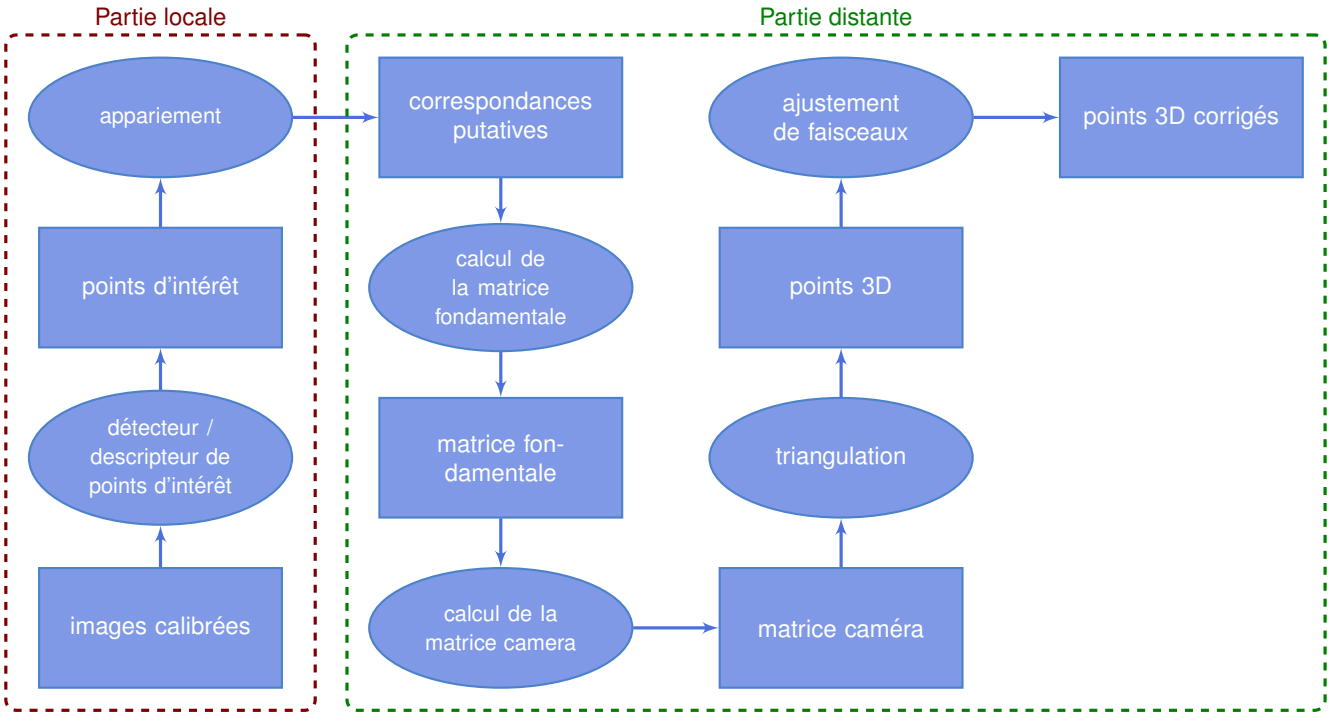


FIGURE 3.24 – Logigramme des étapes de l'application SLAM monoculaire

Dans cette dernière application, nous avons utilisé ARCS pour développer une application de réalité mixte sous-marine. L'idée était qu'à travers d'internet, n'importe quel utilisateur puisse découvrir le monde sous-marin en temps réel par le biais d'un véhicule téléguidé appelé ROV (pour *Remotely Operated Vehicule* – voir figure 3.29(b) page 108). L'un des défis était de parvenir à afficher un flux vidéo temps réel mêlant entités 3D et vidéos 2D filmées en direct par le ROV. Cette application a été développée dans le cadre du projet européen Digital Ocean (FP7 262160).

L'architecture principale du système était répartie sur cinq noeuds comme le montre la figure 3.27 page 106, à savoir :

— Le lieu d'opération du ROV (en mer ou fosse de plongée) ;

— L'application ARCS (site de déploiement de l'application de réalité mixte) ;

— Une base de contenus 3D (modèle 3D de la faune et de la flore sous-marine) ;

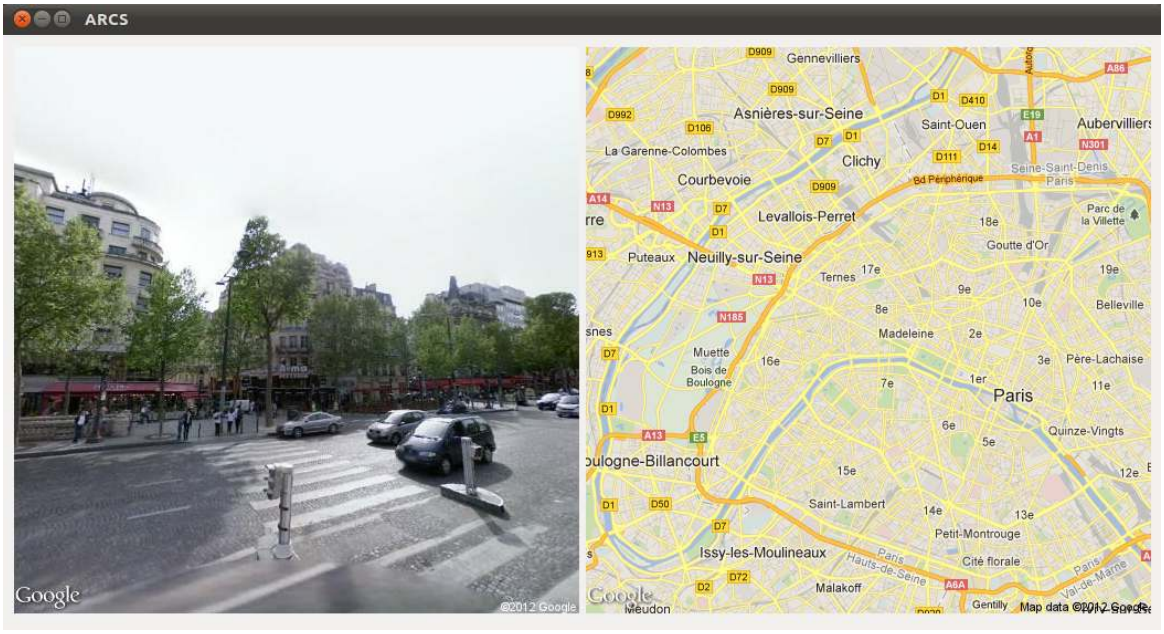


FIGURE 3.25 – Exemple d'une application ARCS utilisant les services de google

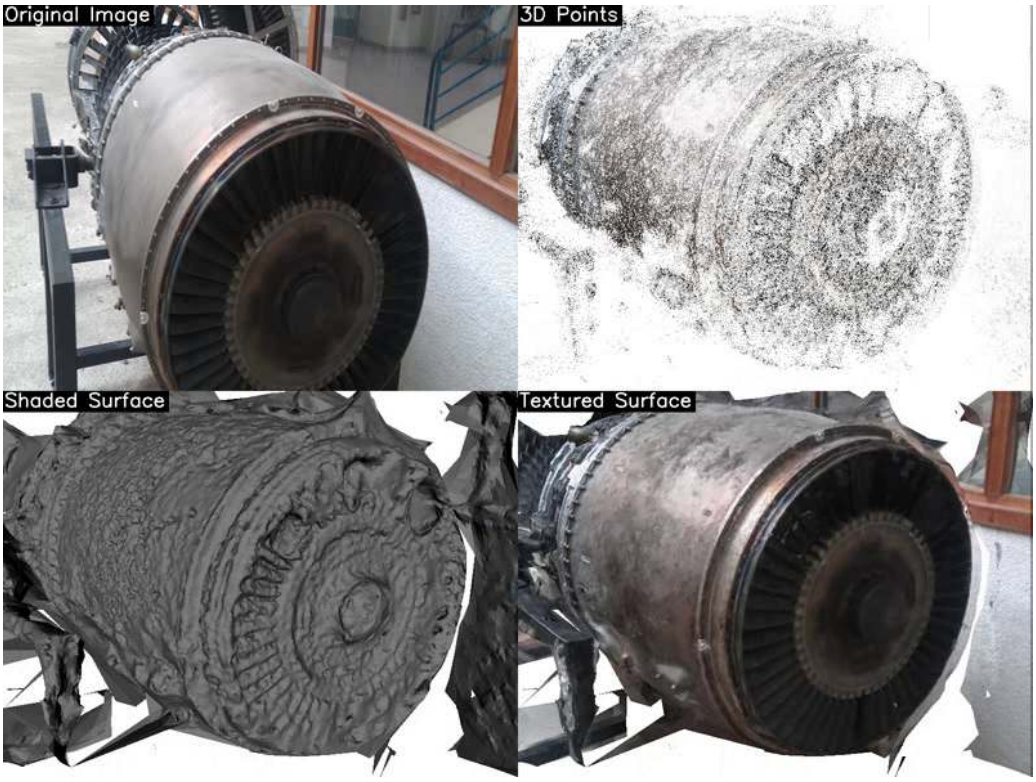


FIGURE 3.26 – Résultat de l'utilisation du webservice de reconstruction 3D de l'université technique de Prague

— Le serveur web hébergeant l'application ;

— Le site utilisateur.

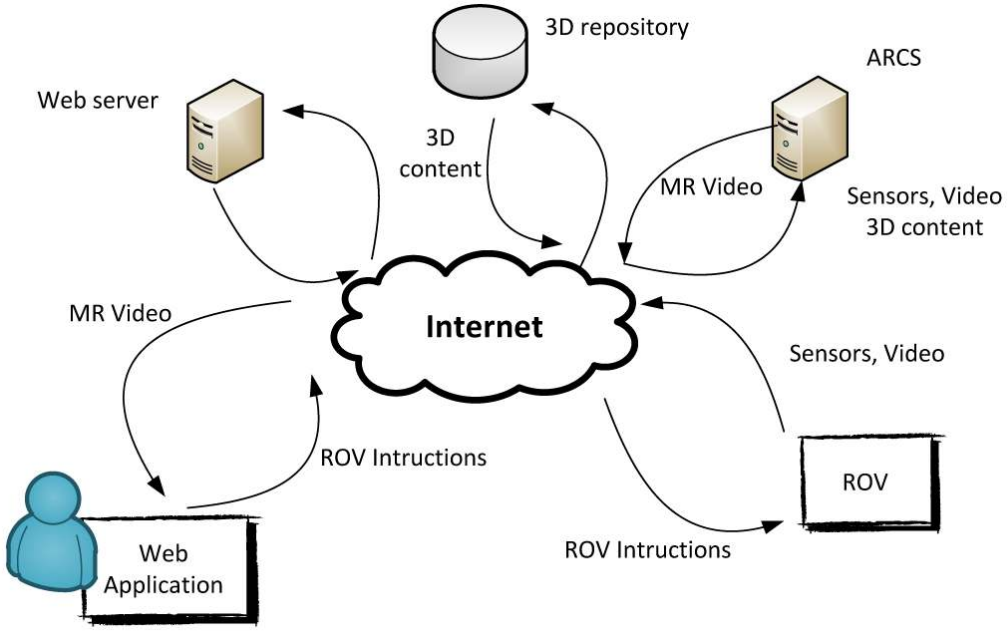


FIGURE 3.27 – Flux de données échangés dans l'application de réalité mixte sous-marine

L'utilisateur contrôle le robot en lui envoyant des commandes de haut niveau via une interface web. L'application les décompose ensuite en instructions bas niveau pour le ROV. Ce dernier est équipé d'un ensemble de capteurs dont une caméra qui envoie un flux vidéo continu vers le serveur Web. L'application ARCS récupère ce flux pour générer la scène augmentée et la mettre à disposition de l'application Web.

L'architecture interne de l'application web s'appuie sur les technologies HTML5 pour le rendu et Php pour contrôler le ROV via le protocole TCP ModBus. Elle coordonne donc, comme nous pouvons le voir à la figure 3.28 page ci-contre :

— les commandes à envoyer au ROV ;

— les flux vidéos augmentés issus de l'application ARCS ;

— la vue personnalisée de l'utilisateur.

L'application ARCS, quant à elle, reprend, dans son organisation interne, le modèle d'architecture que nous avions présenté à la figure 3.22 page 100 et qui découle de l'application directe de la proposition de MACWILLIAMS et al. 2004. Techniquement, le rendu s'appuie s'appuie sur le moteur 3D OGRE, distribué sous licence libre et la diffusion de flux s'effectue en utilisant l'API de VLC (lecteur multimédia) et le format libre Ogg s'appuyant sur les codes Theora (video) et Vorbis (audio). Au niveau applicatif, deux scénarios d'augmentation différents sont proposées :

— Une augmentation contextuelle de la scène en ajoutant, par exemples, des poissons typiques de l'écosystème dans lequel le ROV est en immersion ;

— La reconnaissance et le suivi de marqueur pour afficher des informations dans une zone ou sur un objet afin de faciliter son identification (comme le suggère la figure 3.29(c) page 108).

Ce flux vidéo est ensuite rediffusé et affiché via l'interface graphique de l'utilisateur qui présente, comme nous pouvons le constater à la figure 3.29(a) page 108, six parties :

1. Le retour vidéo en réalité mixte provenant de l'application ARCS ;
2. Un panneau de navigation et de contrôle du ROV ;

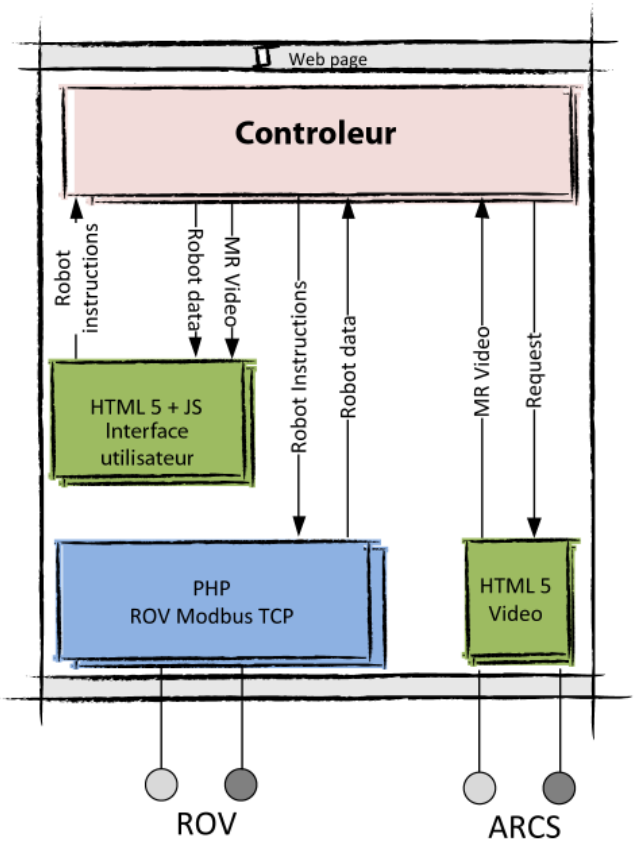


FIGURE 3.28 – Architecture logicielle de l'application web

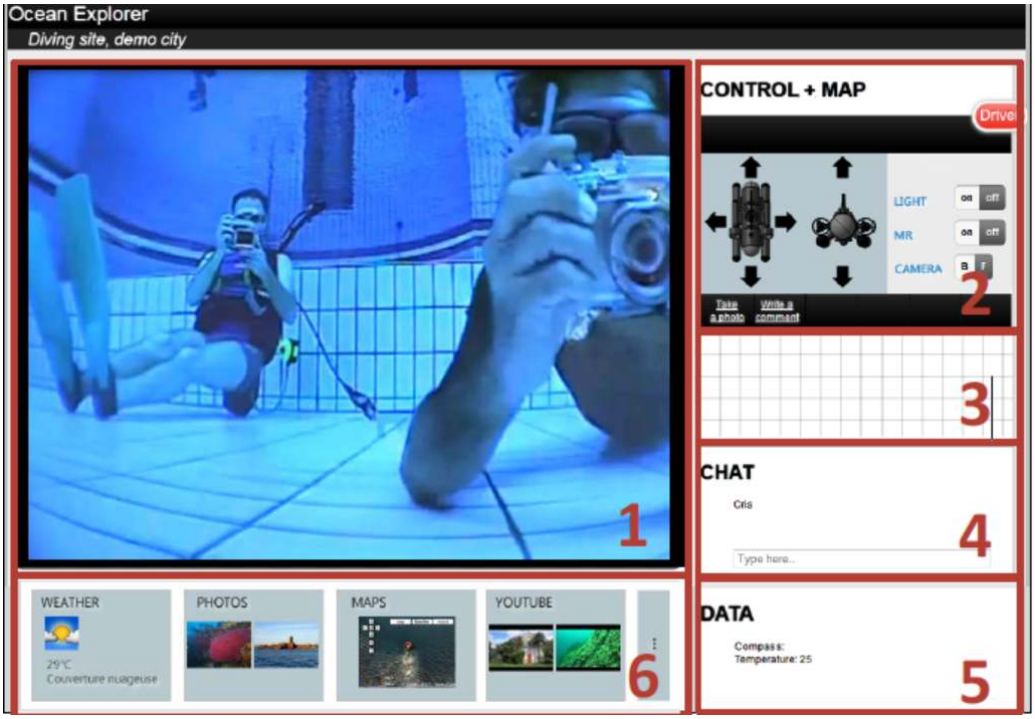
3. Le suivi de la trajectoire du ROV ;
4. Une messagerie instantannée pour communiquer avec d'autres utilisateurs ;
5. Un tableau de bord remontant de manière synthétique les données renvoyées par les capteurs du ROV ;
6. Une zone d'information pour afficher les données relatives au site de plongée.

Le système complet a par ailleurs fait l'objet d'une validation en piscine, en lac puis en mer.

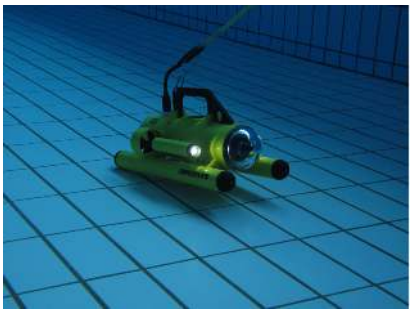
3.2.5 Perspectives sur ces travaux

Ces travaux exposent ce que nous avons pu faire en matière de distribution, au sens répartition des traitements, des applications de réalité augmentée. Nous avons proposé un protocole *ad-hoc* de distribution adapté à la sémantique signal-slot de notre *framework*, des pistes concernant l'adaptation à des *webservices* et une méthodologie d'évaluation des performances.

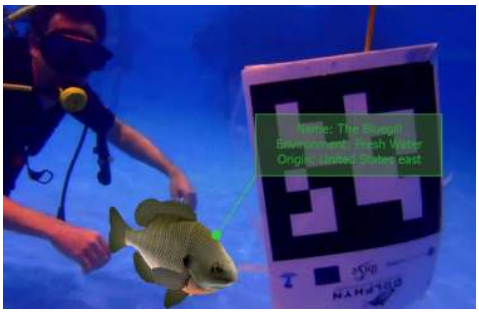
Nous aurions toutefois souhaité aller plus loin en terme d'interfaçage avec les *webservices* en proposant des méthodes automatiques de génération à la volée de composée, reposant à nouveau sur la notion de famille de composants. Cette piste, qui est encore à explorer peut s'avérer profitable pour la mouture javascript d'ARCS, spécialement dédiée aux environnements web. Enfin, une autre problématique n'est ici pas traitée, c'est celle consistant à combiner et répartir les traitements sur des ressources de calcul hétérogènes (par exemple processeurs de carte graphiques – GPU – et processeurs génériques – CPU).



(a) Interface web de l'utilisateur dans le projet Digital Ocean



(b) ROV à l'UCPA Aqua 92 à 5 mètres de profondeur



(c) Bouée augmentée avec modèle 3D et texte

FIGURE 3.29 – Résultats du projet Digital Ocean

3.3 Concurrence et contraintes temporelles

Comme anoncé dans la section 2.3.2 page 64, nous nous nous intéressons aux problématiques associées à la concurrence et aux contraintes temporelles dans les applications de réalité augmentée. Ceci est du à deux raisons distinctes :

1. Les algorithmes (notamment de traitement d'image) mis en jeu dans les applications de réalité sont généralement gourmands en ressources de calcul. Une des solutions pour satisfaire à la demande est de répartir les traitements non pas sur différents noeuds mais sur différents processus (lourds ou légers) exécutables en parallèle. Ceci est rendu d'autant plus possible avec l'émergence graduelle des processeurs multi-cœur au cours des 20 dernières années. Ainsi, la plate-forme SnapDragon 845 pour téléphone mobile produite en 2017 par Qualcomm, renferme 8 coeurs [2017]. Des architectures logicielles parallèles et concurrentes sont donc possibles, y compris sur des terminaux mobiles ;
2. Par définition, l'une des caractéristique des applications de réalité augmentée est d'être en temps-réel interactif (voir la définition 2.4 page 50 donnée par Azuma). Autrement dit, la perception du décalage entre le retour augmenté fourni par le système et la réalité du phénomène se déroulant sous les yeux de l'utilisateur doit être la moins perceptible possible. Ces seuils varient en fonction du type d'application, de système et de modalité d'augmentation sensorielle. Toutefois, ils se traduisent par la notion de respect de contraintes temporelles proches du temps-réel tel qu'il est considéré dans la discipline informatique. Cela classe les applications de réalité augmentée dans la catégorie des systèmes temps-réel dit « mous » ou « souples » (où les contraintes temporelles peuvent être relâchées).

Remarque 3.4 : Notion de temps-réel

Le terme temps-réel recouvre plusieurs notions dans le domaine de la réalité virtuelle et/ou augmentée, qu'il convient de distinguer :

1. Le *temps-réel interactif* est atteint lorsque l'utilisateur ne perçoit pas ou très faiblement la latence entre ses actions sur le système et la réponse de ce dernier ;
2. Les *systèmes temps-réel* sont des systèmes capables de fournir un résultat correct dans les délais qui leur sont impartis. Le respect des contraintes temporelles est donc aussi important que l'exactitude du résultat ;
3. Les *simulations en temps-réel* sont des simulations virtuelles dans lesquelles le temps de la simulation s'écoule de la même manière que le temps physique. Ainsi, une seconde de simulation équivaut à une seconde de temps « réel ».

Dans le cadre de nos recherches, nous nous intéresserons aux deux premières notions.

Les contributions rassemblées dans cette partie sont de deux natures :

— La première indique, comment, dans le cadre logiciel que nous avons déjà posé, nous gérons la concurrence pour nos composants. En particulier, nous montrerons qu'il est possible, de manière raisonnable, de gérer cette dernière en lieu et place du développeur. L'un des avantages est que les composants peuvent être programmés sans se soucier de la gestion de la concurrence et permet d'élargir le cadre de leur réutilisation. Nous montrerons comment nous envisageons d'intégrer ces résultats dans le *framework ARCS* ;

— La deuxième expose les recherches que nous avons entreprises dans le domaine des méthodes formelles afin d'exploiter des formalismes permettant de modéliser, analyser et vérifier le respect des contraintes temporelles associées aux applications de réalité augmentée. Cela nous permettra d'introduire en particulier le langage de haut niveau MIRELA (pour *Mixed Reality Language*) servant à décrire des applications de réalité mixte, la modélisation sous la forme

d'automates temporisés à tâches synchronisées et enfin l'approche sandwich permettant d'anticiper sur le relâchement des contraintes temporelles lorsque l'on implémente une application en se basant sur les automates précédents.

3.3.1 Gestion de la concurrence dans un système orienté composant

Les résultats présentés ici proviennent essentiellement de [DIDIER et MALLEM 2012 ; DIDIER et MALLEM 2014]. Ils montrent comment la concurrence peut-être gérée au sein du *framework* ARCS présenté à la section 3.1.3 page 74. Le modèle d'application proposé se décompose sous forme de processus ou plus exactement de fils d'exécution (*thread*).

Dans ce cadre, nous devons nous assurer que le comportement et les résultats des composants ne s'en retrouvent pas altérés lors d'une exécution parallèle (propriété dite de *thread-safety*). Trois solutions peuvent alors être proposées :

1. L'implémentation des composants comporte systématiquement des instructions visant à garantir la propriété. Cela peut-être réalisé par exemple dans une classe mère dont hériteraient les composants. Toutefois, ces vérifications sont généralement coûteuses à l'exécution et sont inutiles si le contexte d'exécution du composant ne requiert pas cette propriété ;
2. La gestion de la propriété est déléguée au développeur alors en charge de sécuriser les composants susceptibles d'être utilisés dans un environnement concurrent. Toutefois, de par leur nature réutilisable, il est difficile de prévoir à terme dans quel type d'environnement sera utilisé un composant ;
3. L'environnement d'exécution est muni d'heuristiques spécialisées permettant de détecter ces situations et de prendre les mesures adaptées par rapport aux composants s'exécutant dans un environnement concurrent. C'est cette voie que nous explorerons.

Nous allons à présent décrire les hypothèses sur lesquelles nous nous basons pour proposer cette heuristique. Nous esquisserons une solution s'appuyant sur le patron de conception *moniteur* [SCHMIDT et al. 2000] et verrons comment détecter les situations dans lesquelles nous devons y avoir recours. Enfin, nous présenterons succinctement les résulats obtenus.

3.3.1.1 Hypothèses sur l'environnement d'exécution

Pour garantir le fonctionnement de l'heuristique que nous présenterons, le système de programmation par composant doit disposer des caractéristiques suivantes (respectées dans notre cas) :

— Les composants ont des entrées et des sorties séparées ;

— Les composants ont un comportement interne synchrone ; une sortie n'est susceptible d'être déclenchée que si une sollicitation a été reçue en entrée. Toutefois, ici il ne sera pas nécessaire de savoir précisément quelle entrée déclenche quelle sortie ;

— La communication entre composants est synchrone ;

— Les composants sont configurés et initialisés par des invocations sur leurs entrées. Ces invocations peuvent servir à lancer le système ;

— La liste des composants susceptibles de communiquer entre eux est connue à l'avance.

L'environnement d'exécution doit aussi exhiber quelques caractéristiques qui sont :

— son parallélisme ;

— les invocations de départ peuvent être réalisées dans différents *threads*.

3.3.1.2 Esquisse de la solution

Une esquisse de la solution consiste à exploiter l'idée du patron de conception *moniteur*. Celle-ci est exploitée dans certains langage tels que Java et se manifeste par l'utilisation du mot-clé `synchronized`. Dans notre cas, le composant se voit « enveloppé » par un moniteur qui gérera ses accès concurrents. Cette gestion est alors effectuée à l'aide d'une exclusion mutuelle (*mutex*). Le moniteur se substitue au composant. Dès lors qu'un de ses *slots* est appelé, il verrouille l'exclusion mutuelle, ce qui bloque l'accès au composant par d'autres *threads*, y compris par le biais d'autres *slots*. Dans le même temps, le *slot* originel du composant est exécuté. Lorsque l'appel est terminé, l'exclusion mutuelle est dévérrouillée comme nous pouvons le voir à la figure 3.30.

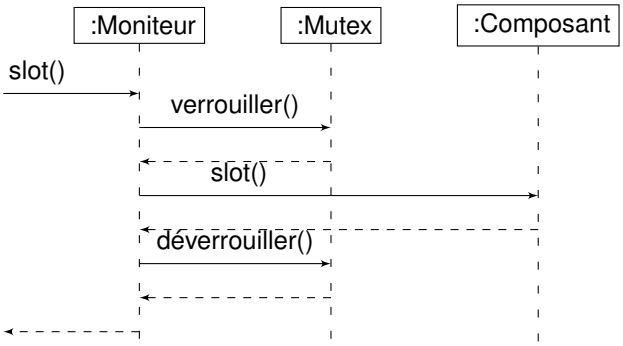


FIGURE 3.30 – Comportement d'un moniteur de composant implicant une exclusion mutuelle

L'utilisation des exclusions mutuelles n'est pas exempte de défauts. Par exemple, leur utilisation peut déboucher sur d'autres problèmes tels que les situations d'impasse (*deadlock*) dès lors que deux *threads* ou plus sont mutuellement verrouillés ou les situations de famine où un composant est favorisé au détriment d'un autre. Cela fait l'objet d'autres types d'analyses qui sortent du cadre de cette solution.

Une fois la solution esquissée, il nous reste à déterminer quels sont les composants à placer sous le contrôle d'un moniteur. C'est l'objet de l'heuristique que nous présenterons après avoir examiné les alternatives proposées dans la littérature.

3.3.1.3 Détection des situations de compétition

Une exclusion mutuelle est apposée pour éviter les situations de compétition lorsque deux *threads* exécutent simultanément les instructions contenues dans un même composant. Dans la littérature les problèmes associés sont divisés en deux :

1. La détection des situation de compétition ;
2. Le problème de l'allocation minimale des exclusions mutuelles afin de prévenir ces situations.

Différentes approches existent pour détecter les situations de compétition. L'approche statique effectue des vérifications sur les systèmes de type [ABADI et al. 2006], passe par une analyse du code [STERLING 1993 ; PRATIKAKIS et al. 2006 ; VOUNG et al. 2007] ou du flux de donnée [ENGLER et ASHCRAFT 2003]. Cette approche ne peut se transposer à notre cas puisque nous considérons que nous n'avons pas accès au code des composants. L'approche dynamique analyse les traces d'exécution des programmes. Les algorithmes et outils développés [SAVAGE et al. 1997 ; BANERJEE et al. 2006 ; POZNIANSKY et SCHUSTER 2007 ; FLANAGAN et FREUND 2009] s'appuient sur la relation « arrivé avant » (ou *happens-before*) [LAMPORT 1978b] ou utilisent une analyse des ensembles bloquants constitués des exclusions mutuelles destinés à protéger l'accès à une variable partagée. Ces outils doivent avoir accès au contexte du programme au cours de l'exécution, ce qui ralentit ces derniers. D'autres techniques telles que l'analyse *post mortem* existent également mais ne correspondent pas à nos besoins.

Le problème de l'allocation minimale des exclusions mutuelles (en anglais MLA, pour *Minimal Lock Assignments*) cherche à déterminer un placement optimal de ces dernières. Les techniques résultantes sont appliquées sur des portions de code jugées critiques et non pas des composants tels que nous les envisageons [HALPERT et al. 2007 ; EMMI et al. 2007 ; ZHANG et al. 2008]. Ici nous ne sommes pas dans le cadre d'une optimisation globale, mais nous cherchons à obtenir une méthode automatisée pour placer ces exclusions à l'exécution. De manière intéressante, les conclusions de HALPERT et al. 2007 indiquent qu'une optimisation fine n'est pas nécessaire pour obtenir des performances raisonnables. C'est ce qui nous a permis de nous orienter vers une heuristique déterminant quels composants sont à placer sous le contrôle d'un moniteur.

3.3.1.4 Heuristique de recherche des composants à monitorer

La solution passe par la construction d'un graphe orienté dont les sommets représentent les composants et les arcs les communications reliant un signal à un slot.

Notations

Nous notons :

— $\mathcal{C}$, l'ensemble des composants du système et donc de sommets du graphe ;

— $\mathcal{M}$, l'ensemble de composants à monitorer ;

— $\mathcal{L}$, un ensemble d'étiquettes. À l'initialisation, nous avons autant d'étiquettes que de *threads* ;

— $\mathcal{E}$, l'ensemble des arcs du graphe dirigé ;

— Chaque composant c de $\mathcal{C}$ a une propriété notée $c.labels$. Elle correspond aux étiquettes attachées au composant et est un sous-ensemble de $\mathcal{L}$;

— Chaque arc e de $\mathcal{E}$ possède deux propriétés : $e.head$ est la tête de l'arcs et $e.tail$ sa queue. Les deux sont des éléments de $\mathcal{C}$;

— $|\mathcal{S}|$ est le cardinal de $\mathcal{S}$;

Heuristique de recherche

L'algorithme 1 page ci-contre donne la formalisation complète de notre heuristique. Si nous mettons de côté l'initialisation, elle itère suivant une boucle comportant trois phases jusqu'à ce que l'ensemble des composants à monitorer soit déterminé :

1. La propagation des étiquettes le long des arcs (lignes 14 à 20) ;
2. Le marquage, où les composants à monitorer sont détectés (lignes 22 à 24) ;
3. La réduction du graphe et la génération de nouvelles étiquettes (lignes 25 à 39).

L'initialisation consiste à générer le premier ensemble d'étiquettes et les attache aux composants considérés comme étant des *threads*. Les autres composants ne comportent pas d'étiquette.

La propagation indique quels composants sont atteignables par quels threads. Cela est équivalent à générer un graphe d'atteignabilité auquel nous ajoutons des étiquettes.

La phase de marquage indique quels composants sont à monitorer. Elle se base sur l'analyse des arcs incidents d'un sommet. Si le sommet à l'origine de l'arc ne comporte pas le même ensemble d'étiquettes que celui d'arrivée, alors ce dernier est marqué.

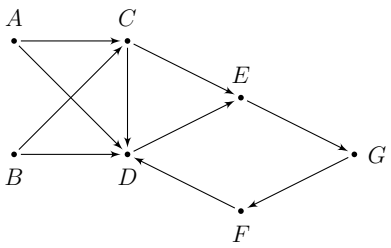
La phase de réduction de graphe supprime les sommets associés à au plus une étiquette, c'est à dire qui ne sont atteints que par un seul *thread* ainsi que les arcs associés. À Chaque composant marqué, une nouvelle étiquette est associée alors que les autres sommets verront leurs étiquettes effacées.

Ces phases sont réitérées jusqu'à ce que plus un seul composant ne puisse être marqué ou éliminé.

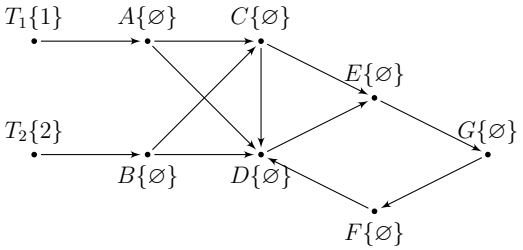
Algorithme 1 : Heuristique de découverte de composants à monitorer

Données : $\mathcal{C}$ – ensemble de composants, $\mathcal{E}$ – ensemble d'arcs
Résultat : $\mathcal{M}$ – ensemble de composants à monitorer

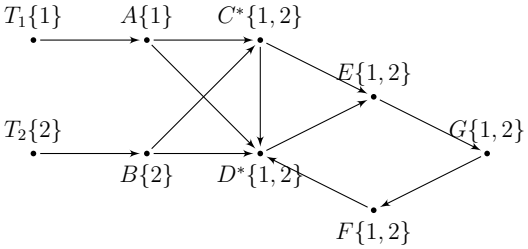
```
   // Initialisation
1  M ← ∅;
2  L ← ∅;
3  l ← 1;
4  pour chaque c ∈ C faire
5      si c représente un thread alors
6          l ← l + 1;
7          L ← L ∪ {l};
8          c.labels ← {l};
9      sinon c.labels ← ∅;
10 fin
11 répéter
       // Propagation
12     répéter
13         applied ← false;
14         pour chaque e ∈ E faire
15             si e.head.labels ≠ e.head.labels ∪ e.tail.labels alors
16                 e.head.labels ← e.head.labels ∪ e.tail.labels;
17                 applied ← true;
18             fin
19         fin
20     jusqu'à applied = false;
       // Marquage
21     M_c ← ∅;
22     pour chaque e ∈ E faire
23         si e.head.labels ≠ e.tail.labels alors M_c ← M_c ∪ {e.head} ;
24     fin
       // Réduction du graphe et génération des étiquettes
25     pour chaque c ∈ C faire
26         si |c.labels| ≤ 1 alors C ← C \ {c} ;
27         sinon
28             si c ∈ M_c alors
29                 c.labels ← {|L| + 1};
30                 L ← L ∪ c.labels;
31             sinon c.labels = ∅;
32         fin
33     fin
34     pour chaque e ∈ E faire
35         si e.head ∉ C ou e.tail ∉ C ou e.head ∈ M_c alors
36             E ← E \ {e}
37         fin
38     fin
39     M ← M ∪ M_c;
40 jusqu'à C = ∅;
```



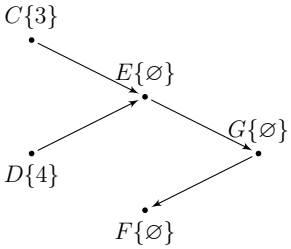
(a) Ensemble initial de composants



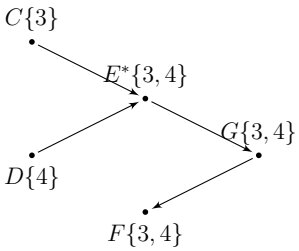
(b) Initialisation de l'heuristique



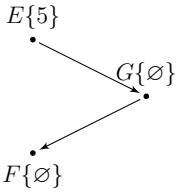
(c) Propagation et marquage(1ère itération)



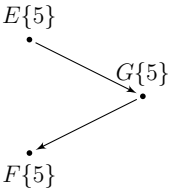
(d) Réduction et génération d'étiquettes (1ère itération)



(e) Propagation et marquage (2ème itération)



(f) Réduction et génération d'étiquettes (2ème itération)



(g) Propagation et marquage (3ème iteration)

FIGURE 3.31 – Fonctionnement de l'heuristique au travers d'un exemple.

Remarque 3.5 : Condition de convergence

La version de l'algorithme présentée ici présente une modification par rapport aux version publiées. Cette dernière porte sur la condition d'élimination des arcs à la ligne 35. La dernière partie, $e.head \in \mathcal{M}_c$, qui a été ajoutée ultérieurement, garantit non seulement la convergence de l'heuristique mais en plus accélère cette dernière.

Fonctionnement de l'heuristique

Le fonctionnement de l'heuristique est présenté sur un exemple abstrait à la figure 3.31. Nous considérons au départ les composants donnés à la figure 3.31(a), puis nous introduisons deux nouveaux composants correspondant à deux *threads*, T_1 et T_2 qui démarrent l'application en accédant à A et B. Les étiquettes associées à chaque composant sont notées entre accolades. Seuls T_1 et T_2 se voient affectés respectivement les étiquettes 1 et 2 (figure 3.31(b)). Lors de la première phase de propagation et de marquage (figure 3.31(c)), deux composants sont marqués, C et D (le marquage est noté par une étoile) puisque les noeuds les précédant sur les arcs ne partagent pas le même ensemble d'étiquette qu'eux. Ensuite, dans la phase de réduction (figure 3.31(d)), les composants T_1, T_2, A et B sont supprimés puisqu'il ne sont associés qu'à une seule étiquette. Les arcs correspondants sont

supprimés ainsi que ceux qui reviennent vers les composants marqués. De nouvelles étiquettes sont générées pour C et D pendant que les autres noeuds sont réinitialisés. A la deuxième itération, E sera marqué (figure 3.31(e)). La troisième itération (figure 3.31(g)) voit la fin de la boucle. En effet, la phase de réduction supprime tous les composants restant.

Au final, les sommets marqués et qui doivent être monitorés sont C, D et E. Au passage, nous constatons que l'exclusion mutuelle doit permettre la réentrance, sinon nous pourrions nous retrouver avec un blocage du moniteur affecté à D en raison du cycle D, E, G, F.

3.3.1.5 Application de l'heuristique

Dans [DIDIER et MALLEM 2014], nous avons présenté une preuve de concept de la faisabilité de l'implémentation de l'heuristique sur un cas simple conçu afin de mettre en évidence une situation de compétition résolue automatiquement. Plus intéressant, nous avons souhaité appliquer cette heuristique sur les développements réalisés pour le projet RAXENV (section 4.1.1 page 135). En effet, dans ce projet, utilisant la première version d'ARCS, nous avions du nous prémunir à la main contre les situations de compétition, ce qui s'est avéré fastidieux. Cela nous permet donc de disposer d'une base de comparaison avec les résultats fournis par notre heuristique.

Concrètement, l'application emploie 4 *threads* : 3 pour traiter les données des capteurs (un pour chaque capteur, à savoir le GPS, la centrale inertielle et la caméra) et 1 pour l'interface graphique. De plus, elle comporte trois états nominaux de fonctionnement (Initialisation manuelle (MI), Réinitialisation (RI) et Prédominance Visuelle (PV)). En fonction de la manière dont sont gérés les composants composites, nous les envisageons comme des boîtes noires (BN) ou des boîtes blanches (BB). Dans le premier cas, nous ne nous intéressons pas à leur structure interne, dans le second nous le faisons, ce qui nous permet de détecter plus finement quels sont les composants à monitorer. Ceci produit donc six jeux de données que nous allons analyser.

Jeu de données	**Composants**	**Arcs**	$\lvert\mathcal{L}\rvert$	**Itérations**	**Durée**	$\lvert\mathcal{M}\rvert$
Réinitialisation (BN) – RIBN	28	34	14	4	4 ms	5
Initialisation manuelle (BN) – IMBN	40	62	27	5	4 ms	9
Prédominance visuelle (BN) – PVBN	38	53	29	5	4 ms	10
Réinitialisation (BB)	35	50	14	4	4 ms	5
Initialisation manuelle (BB)	72	145	71	7	20 ms	21
Prédominance visuelle (BB)	59	112	44	7	12 ms	19

TABLE 3.6 – Jeux de données et résultats produits par notre heuristique.

La table 3.6 renferme les résultats de l'exécution de notre heuristique sur les différents jeux de données. Elle indique le nombre de composants et d'arcs des descriptions initiales ainsi que le nombre d'itérations et la durée nécessaires à l'algorithme pour calculer la solution. Ces données sont complétées par les nombres d'étiquettes générées et de composants à monitorer. Pour avoir une idée du graphe associé à de tels jeux de données, nous avons tracé à la figure 3.32 page suivante celui qui est associé à IMBN, dans lequel les composants composite sont considérés comme des boîtes noires. Nous relevons que finalement peu d'itérations sont nécessaires pour que l'algorithme converge, même si le jeu de données comporte jusqu'à 70 composants et 140 arcs.

La table 3.7 page suivante met en valeur un autre phénomène. Elle dresse la liste des composants à monitorer dans les 3 jeux de données analysés, chacun représentant un état donné de l'application. Il en ressort que certains composants sont monitorés dans certains états de l'application et pas dans d'autres (indiqué par oui ou non dans la colonne correspondante. La croix signifie que le composant n'est pas actif dans l'état considéré). Cela confirme au passage qu'une délégation de la gestion de la propriété de *thread-safety* au développeur est une solution qui ne permet pas de couvrir toutes les réutilisations d'un composant.

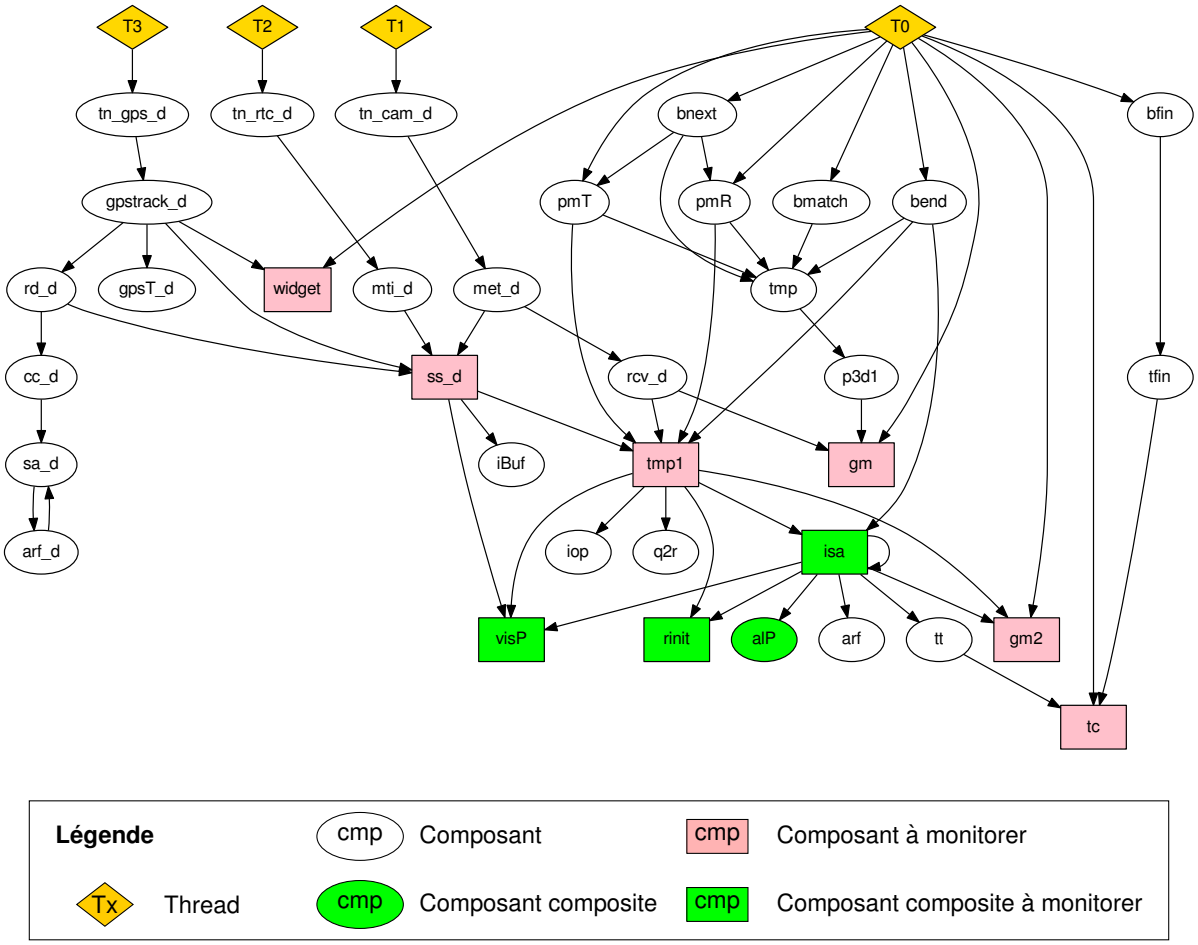


FIGURE 3.32 – Graphe associé au jeu de données IMBN

Nom	**In $\mathcal{M}_{\mathbf{RIBN}}$**	**In $\mathcal{M}_{\mathbf{IMBN}}$**	**In $\mathcal{M}_{\mathbf{PVBN}}$**
ss_d	Oui	Oui	Oui
widget	Oui	Oui	Oui
iBuf	**Oui**	**Non**	**Oui**
gm	Oui	Oui	Oui
tc	Oui	Oui	Oui
tmp1	×	Oui	Oui
visP	×	Oui	Oui
isa	×	Oui	×
gm2	×	Oui	×
rinit	**Non**	**Oui**	**Oui**
gm3	×	×	Oui
alP	**Non**	**Non**	**Oui**
chrono	×	×	Oui

TABLE 3.7 – Composants à monitorer dans les jeux de données RIBN, IMBN et PVBN.

Jeu de données	RIBN	IMBN	PVBN
Composants sous-protégés	2	5	6
Components sur-protégés	2	5	2
Différences : total	4	10	8
Différences rapport	14%	25%	21%

TABLE 3.8 – Comparaison avec une implémentation manuelle des exclusions mutuelles

Enfin, la table 3.8 indique quels sont les écarts relevés entre les résultats produits par notre heuristique et l'implémentation réalisée par le développeur de l'application. L'un des résultats les plus frappants est que si l'on mesure l'écart entre le résultat de l'heuristique et l'implémentation manuelle normalisé par le nombre total de composants, nous constatons un écart de 25% entre les deux. À la décharge du développeur, une des explications est que ce dernier a accès au code interne de chaque composant (qui sont alors des boites blanches) alors que notre heuristique ne se base que sur la description des flux de données dans l'application. Toutefois, les composants comptabilisés comme « surprotégés » relèvent à coup sûr d'une erreur de la part du développeur.

Au niveau des perspectives, certains travaux restent à réaliser. En particulier, la détection automatique des composants à monitorer peut, dans certains cas, s'avérer néfaste à l'exécution. En effet, cela peut aussi aboutir à des situations d'impasses (*deadlock*) pour lesquelles le programme ou l'application pourrait se retrouver bloquée. Une des solutions envisagée est de considérer l'ensemble des composants composites comme des boites blanches et de lancer au préalable une détection de cycle (ces derniers sont connus pour favoriser les situations d'impasse). Cela permettrait de repartitionner les composants et d'isoler les cycles dans des composants composites qui seraient alors à monitorer. Toutefois cette méthode n'est pas, à l'heure actuelle, validée expérimentalement.

Pour terminer, cette heuristique ne résout à elle seule pas tous les problèmes. En effet, elle ne tient pas compte des contraintes temporelles qui peuvent peser sur une application de réalité augmentée. Pour cela, nous nous sommes tournés vers une autre famille de méthodes que nous présentons maintenant.

3.3.2 Gestion des contraintes temporelles

Le domaine de la réalité mixte ne fait que peu appel aux méthodes formelles ou aux méthodes issues du temps-réel au sens du champ de discipline informatique. Pourtant, la complexité de ces systèmes exploitant d'une part les données fournies par divers capteurs opérant à des échelles de temps parfois très différentes (ainsi, un GPS fournis une mesure là où la centrale inertielle en produit une centaine dans le même intervalle de temps) et d'autre part des dispositifs de restitution tout aussi variable au niveau de leur délais opératoires (de quelques dizaines d'images par seconde pour une boucle de rendu visuel à une boucle de rendu haptique cadencé à 1 kHz pour simuler les contacts rigides) en fait une catégorie se prêtant à des études poussées au niveau du respect des contraintes opérationnelles.

Les architectures et *frameworks* cités à la section 3.1.2 page 69 ne s'appuient pas sur les méthodes formelles pour valider le comportement des applications produites. Toutefois, quelques travaux, de façon notable, les exploitent. Ils mettent en avant l'emploi de descriptions formelles des composants dans le but de faciliter leur recomposition dans une application plus large, parfois avec l'aide d'une chaîne d'outils pour produire l'application finale [SANDOR et REICHER 2001 ; LATOSCHIK 2002], pour faciliter l'écriture d'extensions ultérieures [NAVARRE et al. 2005] ou substituer un module par un autre à l'instar d'InTML [FIGUEROA et al. 2004 ; FIGUEROA et al. 2008]. En revanche, aucun de ces derniers ne se concentrent sur la problématique des contraintes temporelles à respecter.

Le paradoxe est donc qu'en pratique, pour les applications de réalité mixte, nous avons tendance à recourir à du matériel performant alors même que cela ne garantit pas forcément que les contraintes

opérationnelles en termes de délais soient respectées. L'idée est donc, pour notre part, de déterminer en amont si l'application est susceptible de poser ce type de problème avant l'implémentation physique sur le matériel. C'est dans ce cadre que nous proposons une méthodologie pour gérer les contraintes temporelles.

Positionnement 3.4 : Proposition d'une méthodologie pour gérer les contraintes temporelles

Dans le cadre du développement d'applications de réalité mixte régies par différentes contraintes temporelles, nous proposons une méthodologie permettant de valider avant implémentation de l'application sur le matériel son comportement selon les étapes suivantes :

1. Décrire une application dans un langage de haut niveau appelé MIRELA ;
2. Traduire la description dans une spécification formelle (dans notre cas, ce sera une variante des automates temporisés spécifiquement conçue pour cet usage) ;
3. Analyser la spécification afin de s'assurer du comportement de l'application ;
4. Générer un squelette d'application pour lequel les propriétés vérifiées sont préservées.

3.3.2.1 Décrire les applications de réalité mixte dans un langage de haut niveau

Nous souhaitions obtenir un langage compositionnel dans lequel les constituants de base qui représentent les différents composants d'une application de réalité mixte.

Le langage MIRELA qui en découle a connu plusieurs évolutions afin de simplifier sa traduction en une spécification formelle, comme peut en témoigner l'écart entre nos publication sur le sujet qui introduisaient la première version de MIRELA [DIDIER et al. 2008b ; DIDIER et al. 2008c ; DIDIER et al. 2008d ; DIDIER et al. 2009a] et la plus récente mise à jour du langage, MIRELA-NG (pour *MIRELA - Next Generation*) [ARCILE et al. 2015b].

Le principe est d'identifier les grandes familles de constituants d'une application de réalité mixte, fondé sur l'analyse de ces dernières. Nous avons ainsi dégagé quatre catégories d'éléments qui correspondent chacune à une étape de traitement des données (voir figure 3.33 page ci-contre) :

— Les capteurs, qui produisent les données exploitées par l'application. Ils sont décomposés en deux sous-catégories :
  — Les capteurs périodiques (*periodic*) qui produisent périodiquement des données ;
  — Les capteurs apériodiques (*aperiodic*) qui représentent les données transmises sur une base évènementielle (le clavier et la souris font partie de cette catégorie).
— Les unités de traitement en charge de transformer ces données. Pour ces dernières, nous les avons catégorisées en unités élémentaires dont la composition peut produire des comportements plus complexes :
  — Les unités *first* attendent qu'au moins une donnée soit présentée sur l'une de leurs entrées pour produire, quelques instants plus tard, une information en sortie ;
  — Les unités *both* attendent que toutes les données soient présentées sur toutes les entrées pour produire en sortie ;
  — Les unites « prioritaires » (*priority*) produisent des données dès lors que l'entrée prioritaire est activée. Le traitement prend en compte les données des autres entrées si elles sont présentes.
— Les boucles de rendu (*rendering*) qui restituent les informations aux utilisateurs ;
— Les zones mémoires (*shared memory*) partagées qui permettent aux unités de traitement de déposer des données à destination des boucles de rendu.

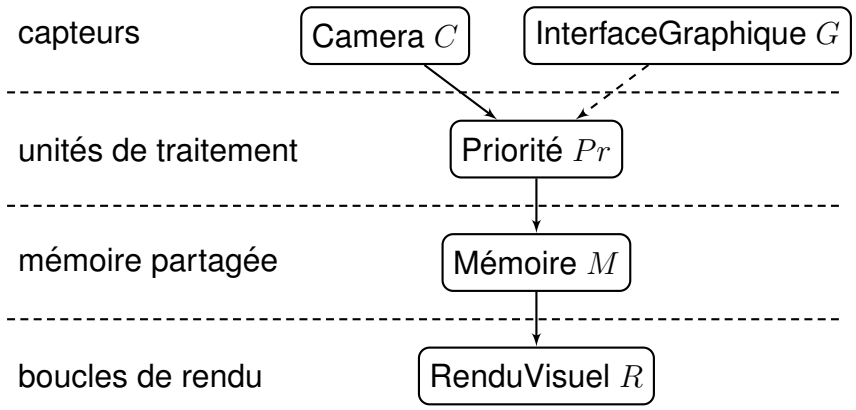


FIGURE 3.33 – Exemple de décomposition d'application de réalité mixte.

Une description dans le langage MIRELA suivra donc la forme suivante :

$$id = Comp \rightarrow TList; \ldots; \ id = Comp \rightarrow TList.$$

où $Comp$ représente un type de composant qui est soit un capteur ($Sensor$), une unité de traitement ($PUnit$), une mémoire partagée ($MUnit$) ou une boucle de rendu ($RLoop$). $TList$ est une liste optionnelle d'identifiants indiquant à quels composants sont envoyées les données et dans quel ordre. Chaque composant pourvu d'entrées indique quelles en sont les provenances par le même système d'identifiants.

Les types de composants sont décrits suivant la syntaxe suivante :

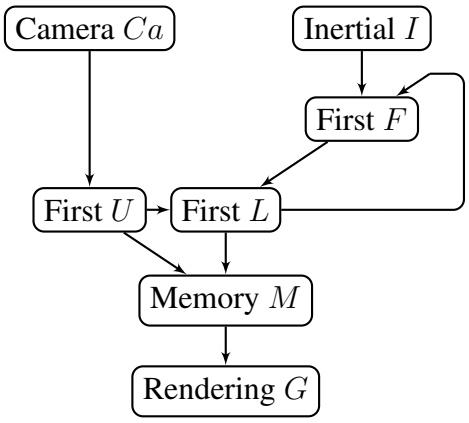
$$\begin{array}{lcl} Sensor & ::= & \mathsf{Periodic}(min\_start, max\_start)[min,max] \mid \\ & & \mathsf{Aperiodic}(min\_event) \\ PUnit & ::= & \mathsf{First}(SList) \mid \\ & & \mathsf{Both}(id, id)[min,max] \mid \\ & & \mathsf{Priority}(id[min,max], id[min,max]) \\ MUnit & ::= & \mathsf{Memory}(SList) \\ RLoop & ::= & \mathsf{Rendering}(min\_rg, max\_rg)(id[min,max]) \end{array}$$

Les indications de type min_* ou max_* donnent les durées temporelles associées aux actions de chacun des composants. Ainsi, dans le cas du capteur périodique, nous considérons qu'il prend entre min_{start} et max_{start} unités de temps pour s'initialiser, puis que sa boucle périodique d'envoi des données est comprise entre min et max unités de temps.

Ainsi, nous pouvons spécifier à haut niveau des applications telles que celles présentées à la figure 3.34 page suivante. Le système modélisé comporte une caméra C et une centrale inertielle I. Cette dernière est utilisée pour estimer la position et l'orientation de la caméra (appelée aussi pose) par rapport à son environnement, dans le cas où une estimation directe par traitement d'image (réalisée par U) échoue en raison de phénomènes tels que l'occultation de l'objet suivi, des mouvements brusques ou des changements importants et rapides de luminosité. Toutefois, la centrale inertielle aura aussi quelques inconvénients : calculer les positions et orientations à partir de ses données (tâche réalisée par F) au travers d'une double intégration des accélérations et d'une intégration des vitesses angulaire accumule les erreurs et dérive au bout d'un certain temps. Pour pallier cet inconvénient, les poses de la caméra, lorsqu'elles sont disponibles, servent à corriger le comportement de la centrale (rôle joué par L) ou pour interpoler la pose à partir des dernières données connues (ce qui explique le retour de L vers F). Le résultat de ces deux processus est stocké dans la mémoire partagée (M) afin d'être utilisé par un rendu graphique (G) effectué par dessus les images venant de la caméra. Ce modèle est extrait d'un système plus important comportant également un GPS non modélisé ici [ABABSA et al. 2012]. Les délais indiqués dans la description sont en dizaines de micro-secondes.

$$
\begin{aligned}
Ca &= \mathsf{Periodic}(2000, 3000)[3500, 4500];\\
I &= \mathsf{Periodic}(400, 600)[900, 1000];\\
U &= \mathsf{First}(Ca[7000, 9000]) \rightarrow (M, L);\\
F &= \mathsf{First}(L[40, 60], I[40, 60]);\\
L &= \mathsf{First}(F[20, 30], U[20, 30]) \rightarrow (M, F);\\
M &= \mathsf{Memory}(U[300, 400], L[10, 20]);\\
G &= \mathsf{Rendering}(500, 750)(M[310, 420])
\end{aligned}
$$

(a) Description MIRELA



(b) Structure associée

FIGURE 3.34 – Exemple de description MIRELA d'un système de réalité mixte

Une description MIRELA est ensuite traduite dans une ou plusieurs spécifications formelles en fonction des comportements à analyser. Dans notre cas, nous allons nous appuyer sur un sous-ensemble des automates temporisés, les TASTs (*Timed Automata with Synchronized Tasks*).

3.3.2.2 Spécification formelle d'une application de réalité mixte

Plusieurs types de spécifications formelles existent pour décrire et analyser le comportement des systèmes temps-réels. Certaines sont basées sur des notations semi-formelles associées à des automates à états finis décrivant le système et son comportement. Cela est le cas des diagrammes états-transitions (*statecharts*) [HAREL 1987b], ou d'UML-RT [DOUGLASS 1997]. Toutefois, ces représentations ne permettent pas à elles seules de vérifier les propriétés d'un système. Il leur est alors associé des représentations formelles basées sur des automates, des algèbres de processus [HAREL 1987a], des réseaux de Petri [REISIG 2012] ou la logique temporelle [PNUELI et MANNA 1992].

Très tôt, nous nous sommes intéressés aux automates temporisés [ALUR et DILL 1994] car ces derniers permettent d'exprimer les contraintes temporelles dans des systèmes concurrents. De plus, un ensemble d'outils matures et puissants (par exemple UPPAAL [LARSEN et al. 1997] ou PRISM [KWIATKOWSKA et al. 2011]) sont capables d'analyser et de simuler le modèle obtenu.

Brève présentation des automates temporisés

Nous ne reproduirons pas la définition formelle complète de ces derniers. En revanche, il convient d'expliquer leur fonctionnement.

Un automate temporisé est un automate à état finis auquel est adjoint une représentation continue du temps par le biais de variables réelles appelées *horloges* qui permettent d'exprimer les contraintes temporelles. Un automate temporisé est représenté sous la forme d'un graphe orienté dans lequel les noeuds correspondent aux localités dans lesquelles le système peut être et les arcs indiquent un possible changement de localité (voir figure 3.35 page ci-contre). Les contraintes temporelles sont formulées sous la forme de *contraintes d'horloge* et sont attachées soit aux localités, soit aux arcs. Elles s'expriment sous par le biais de conjonctions de contraintes atomiques qui comparent la valeur d'une horloge x à une constante rationnelle c. Chaque automate a un nombre fini de localités dont une seule étiquettée *initiale*. Pour chaque localité le temps progresse de manière uniforme tout comme les valeurs des horloges traduisant le temps écoulé depuis leur dernière remise à zéro. Les contraintes d'horloge associées aux localités, appelées aussi *invariants*, ne permettent au système de rester dans leur localité que si elles sont satisfaites. Les contraintes d'horloge associées aux arcs, appelées aussi

gardes indiquent les conditions portant sur l'horloge pour pouvoir franchir la transition. De plus les arcs peuvent être annotés avec des indications de remise à zéro d'horloge.

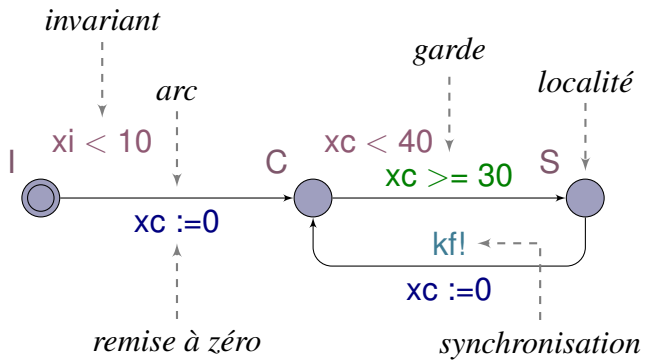


FIGURE 3.35 – Automate temporisé et notations associées

Un certain nombre d'extensions existent concernant le modèle des automates temporisés [WAEZ et al. 2013]. En particulier, les synchronisations permettent de créer des réseaux d'automates temporisés à même de se mettre à jour simultanément. Cela est réalisé par des annotations sur les arcs indiquant des émissions ($k!$) ou des réceptions ($k?$) sur des canaux de synchronisation. Lorsqu'une paire d'arcs annotée en réception et émission sur le même canal peut-être franchie simultanément, les deux automates progressent en même temps.

Dans un réseau d'automates, une transition concernera alors le franchissement simultané d'un ou plusieurs arcs. L'état d'un système sera composé de la liste des localités dans lesquelles se trouve le système auquel sont adjointes les valeurs des horloges.

Timed Automata with Synchronised Tasks

Les automates temporisés et leurs diverses extensions supportées par les outils de vérification ont une richesse certaine en termes d'expressivité de modèles. Il est ainsi possible d'exprimer des comportements qui peuvent n'avoir aucun sens une fois implémentés sur une machine réelle. Par exemple, il est possible de spécifier des systèmes ayant un comportement de type *Zenon*, c'est à dire ayant à réaliser une infinité d'action au cours d'un intervalle de temps fini, ce qui n'est pas possible sur une machine physique. Nous avons ainsi défini un sous ensemble d'automates temporisés appelé TAST (*Timed Automata with Synchronized Tasks*) qui garantissent un certain type de comportement implémentable. Pour sa définition formelle, nous nous rapporterons à [DEVILLERS et al. 2013]. Ils sont construits de la manière suivante :

1. Les localités sont séparées en deux sous-ensemble mutuellement exclusifs. L^+ contient les localités associées à des tâches à exécuter et L^- correspond à des localités où le système attend un évènement ou qu'une condition particulière soit réalisée ;
2. Les gardes ne peuvent contenir des conjonctions de prédicats élémentaires de la forme $x < e$ ou $x \geq e$ où x est une horloge et e une constante rationnelle positive ;
3. Les invariants, lorsqu'ils sont définis, ne contiennent qu'une clause de type $x < e$;
4. Les synchronisations sont de type urgente, c'est à dire qu'elles doivent être effectuées dès qu'elles sont possibles ;
5. Pour toute localité l de l'ensemble L^+, il lui est associée une horloge x_l qui est la seule à pouvoir être utilisée dans l'expression de l'invariant. De plus, tout arc entrant de l doit remettre à zéro x_l et tout arc sortant ne possède qu'une garde contenant l'expression $x_l \geq e'$;

6. Pout toute localité l de l'ensemble L^-, les mêmes règles s'appliqueront sur son invariant et les arcs entrants. Tout arc sortant sera soit pourvu d'une étiquette de synchronisation soit d'une garde équivalente à celle d'une localité de l'ensemble L^+ ;
7. Chaque TAST comporte au moins une garde de type $x \geq e$ et une remise à zéro de l'horloge x.

Cette structuration des automates nous garantit, par construction, que le système ne possèdera pas un comportement de type Zénon. Nous évoquerons d'autres propriétés à propos des TASTs lorsque nous aborderons l'analyse du modèle pour garantir son implémentabilité.

Passage de MIRELA aux automates temporisés

Le passage entre MIRELA et les automates temporisés se réalise par le biais d'un compilateur qui traduit la description en une ou plusieurs spécifications. Le principe est que pour chaque composant prévu par MIRELA, une contrepartie sous la forme de TAST lui est associée. Les communications sont ensuite symbolisées par des canaux de synchronisation reliant les automates. Ainsi la traduction de l'exemple que nous avions fourni (figure 3.34 page 120) devient le réseau de TASTs présenté à la figure 3.36. Les localités colorées en bleu pâles sont celles associées à une tâche. Autrement dit, elles appartiennent à l'ensemble L^+.

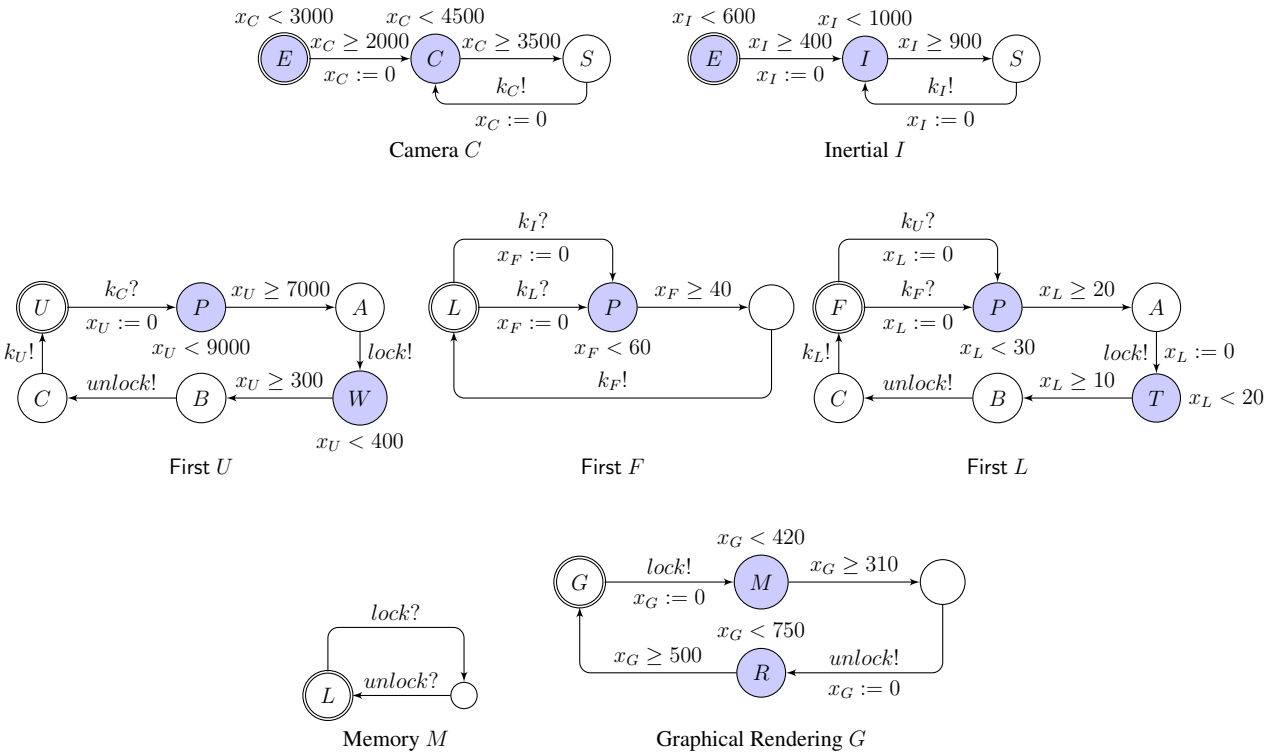


FIGURE 3.36 – Représentation sous forme de TASTs de notre exemple

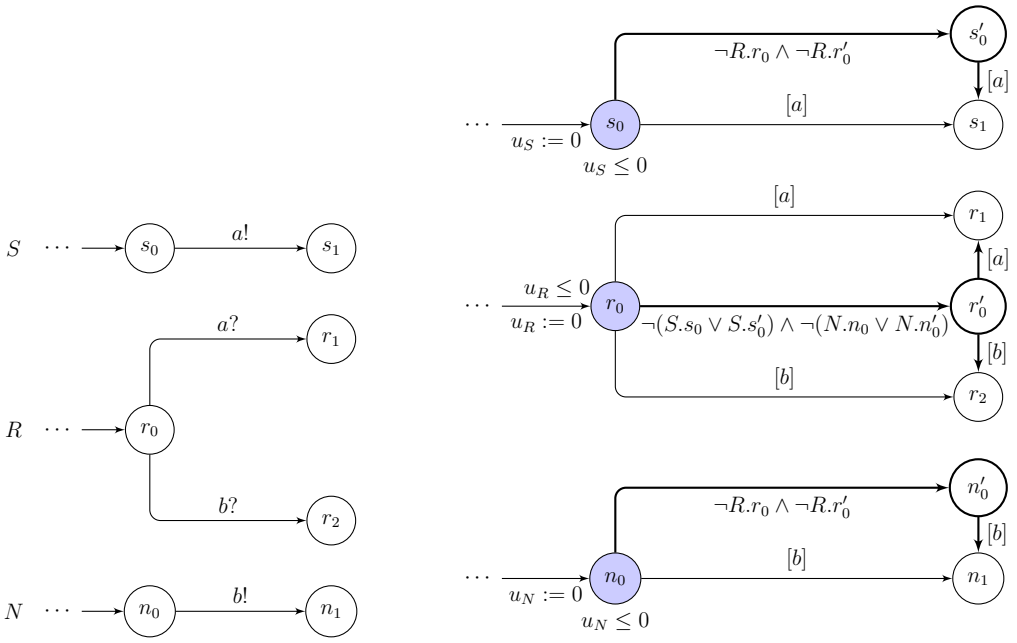
3.3.2.3 Analyser les spécifications

L'analyse d'une spécification construite à base d'automates temporisés fait partie de ce que l'on appelle le *model checking*. Pour la représentation formelle que nous utilisons, des outils permettent d'automatiser et de systématiser la recherche des propriétés souhaitables de notre système ou de s'assurer que des comportements indésirables ne se produisent pas.

Dans nos travaux, nous avons utilisé deux outils spécialisés :

— UPPAAL [LARSEN et al. 1997] dédié en premier lieu à l'analyse des réseaux d'automates temporisés. Ce dernier possède également les extensions utilisées par les TASTs telles que les communications urgentes ;

— PRISM [KWIATKOWSKA et al. 2011], plus généraliste, est davantage orienté vers les modèles probabilistes. Il permet également de modéliser les communications urgentes. Toutefois, cela nécessite une adaptation comme on peut le voir à la figure 3.37 et que nous avons proposé dans [ARCILE et al. 2015b].



(a) Réseau de TAST avec deux canaux de communication urgente a et b modélisé avec UPPAAL.

(b) Réseau correspondant modélisé avec PRISM

FIGURE 3.37 – Modélisation des communications urgentes dans PRISM. Les ajouts se matérialisent par des traits épais.

Ces outils, une fois nos modèles chargés, pourront vérifier des propriétés écrites dans une logique temporelle. Pour UPPAAL, ce sera la logique du temps arborescent CTL (*computational tree logic*) et pour PRISM, ce sera la variante CTL* qui combine à la fois CTL et la logique temporelle linéaire (LTL).

Langage de requête

Pour donner une idée de l'expression de ces requêtes, ces dernières se présentent sous la forme suivante (une représentation graphique des arbres logiques correspondant à ces requêtes est donnée à la figure 3.38 page suivante) :

— `AG not deadlock` est vraie s'il y a toujours une transition sortante franchissable ;

— `AG p` (appelée sûreté ou *safety*) est vraie si la propriété p est invariante ;

— `EG p` (appelée aussi sûreté) est vraie si p est vérifiée pour un chemin d'exécution (liste d'états consécutifs démarrant de l'état initial et se prolongeant jusqu'à l'état final) ;

— `EF p` (appelée accessibilité ou *reachability*) est vraie si le système peut atteindre, depuis le départ, un état qui vérifie la propriété p ;

— `AF p` (appelée vivacité ou *liveness*) est vraie si tout chemin d'exécution possède un état qui vérifie la propriété p ;

— `AG (p ⇒ AF q)` (appelée aussi vivacité) est vraie si lorsque p est vérifiée alors q sera vérifiée ultérieurement. Cette relation, dans l'outil UPPAAL, est aussi appelée *leads to* et est notée `p --> q`.

D'autres propriétés plus spécifiques permettent de s'enquérir du maximum ou du minimum d'une expression donnée sur l'ensemble du système ou pour les états vérifiant une certaine propriété.

Les outils de *model checking*, une fois la requête formulée, vont analyser le système pour statuer sur le résultat. Certaines d'entre elles vont, par ailleurs, nécessiter de parcourir l'ensemble des états du système (Il s'agira, en réalité, du parcours un espace d'état intermédiaire « compacté », celui exprimé par les automates de région).

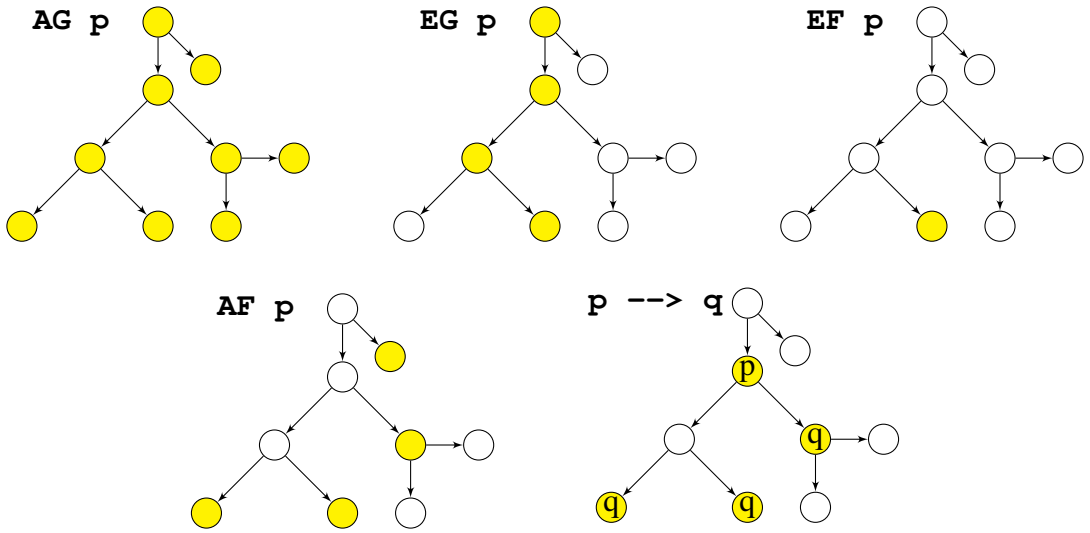


FIGURE 3.38 – Requêtes CTL et représentation des arbres logiques correspondants. Les localités en jaune sont celles qui vérifient la propriété p

Quelles propriétés examiner dans une application de réalité mixte ?

De par sa nature exhaustive, le *model checking* permet de vérifier tous les comportements possibles du système, y compris ceux qui ont une faible chance de survenir. Dans [DIDIER et al. 2009a], nous avions commencé à écrire quelques requêtes permettant d'analyser le comportement et qui permettent de répondre à des questions telles que « est-ce que la boucle de rendu est capable d'effectuer sa tâche y compris lorsqu'un des capteurs est désactivé ? ».

Dans [DEVILLERS et al. 2014 ; ARCILE et al. 2015b], nous réalisons une analyse plus systématique de l'application en commençant par catégoriser les comportements néfastes tels que :

— Le blocage complet (*complete blocking*), qui indique que le système spécifié se retrouve dans un état dans lequel il ne peut plus progresser au niveau temporel. Cela s'interprète par une violation des contraintes temporelles dans le système physique ;

— La situation d'impasse globale (*global deadlock*), qui indique que le système est bloqué dans un état et que la seule action possible est de laisser le temps s'écouler ;

— Les comportements de type Zénon (*Zeno*) quand l'état du système peut passer par une infinité de localité dans une période de temps bornée (possiblement nulle) ;

Au niveau des réseaux d'automates se rajoutent d'autres comportements néfastes :

— La situation d'impasse locale (*local deadlock*), qui ne se produit que dans un composant (un des automates) où la localité est figée pendant que les autres automates progressent normalement ;

— Une situation de famine (*starvation*) se caractérise par le fait qu'un composant peut progresser mais que le temps qui s'écoule avant que cela soit possible peut être infini, parce que d'autres composants le retardent indéfiniment ;

— L'attente locale non bornée (*local unbouded waiting*) qui traduit qu'un composant peut progresser mais que la durée permettant d'observer ce changement n'est pas limitée. Il est à noter que ce cas n'est pas toujours la manifestation d'un mauvais comportement quand il s'agit par exemple d'un état de récupération d'erreur.

Une autre catégorie de propriétés qui peuvent être vérifiées sont les valeurs temporelles telles que la durée d'un cycle dans un composant. De ce fait, nous pouvons analyser localement les durées de ces derniers en prenant en compte les temps d'attente qui sont, par définition, variables.

Analyse des comportements indésirables

Par construction, les réseaux de TAST produits à partir d'une description MIRELA possèdent un certain nombre de propriétés que nous avons pu démontrer dans [DEVILLERS et al. 2014] telles que :

— Aucun comportement de type Zénon (fort ou faible) ne peut survenir ;

— Une impasse locale ne peut survenir que lorsqu'une localité d'attente est atteinte dans un composant ;

— Une zone de mémoire partagée ne peut se retrouver en situation d'impasse que si tous les composants qui y sont connectés sont en situation d'impasse ;

— Une boucle de rendu ne peut se retrouver en situation d'impasse. Un système en possédant une ne peut donc être en impasse globale.

De ces propriétés, il en découle que seuls les problèmes survenant localement dans chaque composant nécessitent d'être détectés. De plus, il n'est pas nécessaire d'analyser ce qui se produit dans chacune des localités d'attente. En effet, celles qui se synchronisent sur la libération de la mémoire partagée et celles de la mémoire partagée en elle-même ne sont pas concernées ou leur comportement est analysé par le biais d'autres localités.

Nous pouvons donc classer les localités d'attente en trois sous-ensemble qui sont :

— $\mathcal{N}$, l'ensemble des localités d'attente à l'origine des transitions étiquetées *unlock ?* et *unlock !*. Celles-ci ne sont donc pas concernées par les comportements indésirables ;

— $\mathcal{O}$, l'ensemble des localités d'attente à l'origine des transitions étiquetées *lock !*. Elles ne peuvent être concernées que par les situations de famine ;

— $\mathcal{W}$, l'ensemble des autres localités d'attente.

L'analyse systématique permet de détecter un éventuel dysfonctionnement. Dans ce cas, on s'attachera à diagnostiquer également sa nature. C'est dans cet esprit que nous avons construit l'algorithme 2 page suivante permettant une analyse systématique des spécifications TASTs obtenues à partir d'une description MIRELA [ARCILE et al. 2015b]. De cette manière, nous pouvons ensuite modifier la spécification afin de fiabiliser l'application. Quelques exemples de ces modifications sont proposées dans [DEVILLERS et al. 2014].

De l'utilisabilité de la vérification de modèle

La procédure proposée nous a permis de vérifier divers modèles pour, au besoin, les adapter. Toutefois, nous n'avons pas abordé l'une des problématiques de la vérification qui concerne l'espace d'état généré par une spécification. En effet, et nous avons pu l'apprendre à nos dépends lors des premières traduction de MIRELA en automates temporisés (avant donc de formaliser les TASTs), plus le modèle

Algorithme 2 : Statut d'une localité d'attente

Données : $\mathcal{W}$, $\mathcal{O}$ – ensembles de localités d'attente dans une spécification MIRELA
Résultat : Déterminer, pour chaque localité, s'il s'agit d'une famine, d'une attente non bornée, une situation d'impasse (locale) ou une combinaison d'entre elles.

pour chaque $w \in \mathcal{W} \cup \mathcal{O}$ **faire**
  Vérifier $\phi_w \leftarrow$ EF EG w;
  si $\phi_w =$ *faux* **alors**
    w n'est ni une famine, une attente non bornée, ni une sitatuation d'impasse ;
  si $w \in \mathcal{O}$ **alors**
    w est une localité de famine
  Vérifier $\psi_w \leftarrow$ EF AG w;
  si $\psi_w =$ *faux* **alors**
    w est une famine et/ou une attente non bornée
  Vérifier $\rho_w \leftarrow$ EF EG $(w \wedge (\text{EF } \neg w))$;
  si $\rho_w =$ *faux* **alors**
    w est une situation d'impasse
  w est une situation d'impasse, une famine et/ou une attente non bornée
fin

est complexe, plus sa vérification est longue à réaliser. En réalité, le *model checking* employant les logiques temporelles LTL et CTL* fait partie des problèmes PSPACE-complets [SCHNOEBELEN 2002], c'est à dire qui peuvent être résolu en exploitant un volume de mémoire polynomial par rapport au nombre de données d'entrées.

Nous avons ainsi reportés qu'en changeant la manière de modéliser pour avoir des abstractions qui ne dépendent réellement que du temps, et donc moins proches de l'implémentation détaillée nous a permis de diviser le nombre d'état à parcourir pour vérifier certaines propriétés d'un facteur 100 [DIDIER et al. 2013]. Ce résultat a été rendu possible, entre autre, grâce à l'utilisation des TASTs.

Il n'en demeure pas moins que certaines analyse, en particulier lorsqu'elles impliquent de parcourir la totalité de l'espace des états restent complexes à réaliser, voir presque impossibles à analyser, le temps pris alors pour vérifier une requête se comptant parfois en jours. Dans ce cas, d'autres techniques doivent être utilisées pour parvenir à analyser le modèle. Cela a motivé des travaux dérivés s'attachant à découper les spécifications pour en faire des analyses par morceaux [DEVILLERS et KLAUDEL 2016].

3.3.2.4 Implémenter une spécification

Nous nous proposons d'élaborer une chaîne complète de prototypage des applications. L'idée est, une fois la spécification validée, de démarrer la réalisation du système de réalité mixte en fournissant un squelette d'application à partir des automates.

Toutefois, l'implémentation d'une telle application pose quelques problèmes. En effet, l'application physique étant cadencée sur une horloge physique, nous passons du domaine du temps continu tel que représenté par les automates temporisés au monde discret des circuits numériques.

Nous nous sommes attelés à ce problème dans [DEVILLERS et al. 2013]. Nous y avons proposé :

— Une architecture de plate-forme d'exécution d'une spécification à base de TASTs séparée en deux couches, une d'exécution des tâches et une de supervision par le biais d'un contrôleur ;

— Un algorithme de fonctionnement pour le contrôleur ;

— Une analyse des distortions possibles des horloges et une méthode pour analyser les propriétés du système telles qu'elles se présentent lors de son exécution.

Architecture pour l'exécution d'une spécification TAST

Nous avons proposé l'architecture représentée à la figure 3.39. Elle est divisée en deux couches, la plus basse est constituée de composants implémentant les tâches de l'application (associés aux localités de L^+), la plus haute renferme un contrôleur en charge d'interpréter la spécification TAST en mettant à jour le système de façon périodique et régulière (l'intervalle de temps entre les mises à jour étant appelé *quantum* de temps). Cela consiste à exécuter les transitions, mettre à jour les horloges et démarrer les tâches.

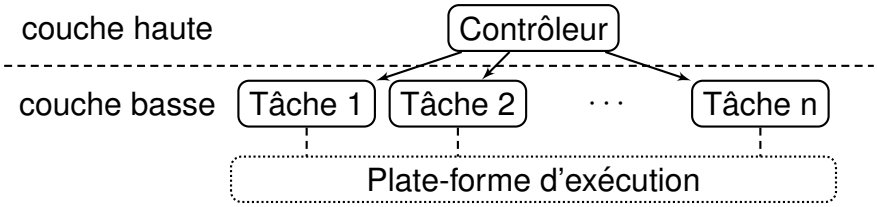


FIGURE 3.39 – Architecture de l'environnement d'exécution

Algorithme d'exécution du contrôleur

Pour une spécification TAST ($\mathcal{A}$) et un *quantum* de temps (Δ), le contrôleur résultant ($\mathcal{C}_{\mathcal{A},\cdot}$) se conformera au schéma d'exécution décrit par l'algorithme 3. Pour que l'exécution d'une spécification ait un sens, elle ne doit ni exhiber de comportement de type Zénon ni déboucher sur une situation d'impasse. Le contrôleur maintient sa propre version, discrète, des horloges. Elles sont initialisées à 0 et mises à jour à chaque boucle correspondant à un *quantum* de temps. Le contrôleur débute à l'état initial donné par la spécification et suit les localités atteintes dans chaque automate dans chaque tour de boucle. Une transition est considérée comme *exécutable* dès lors qu'elle est composé d'un arcs portant une garde satisfaite ou d'une paire d'arcs dont la synchronisation est permise. L'exécution d'une telle transition, si elle débouche sur des localités auxquelles une tâche est associée, démarre ces dernières. Pour terminer, chaque itération correspondant à un *quantum* est décomposée en deux phases. La première, dite de progression, met à jour le suivi de la spécification. Le seconde consiste à attendre la fin du *quantum* de temps.

Algorithme 3 : Algorithme du contrôleur gérant une spécification de TAST selon un *quantum* de temps donné.

Données : $\mathcal{A}$: spécification de TASTS, $\mathcal{A} = \{A_1, \ldots, A_n\}$, chaque $\mathcal{A}_i$ étant un TAST ; $\mathcal{A}$ ne débouche pas sur une situation d'impasse
Δ : *quantum* de temps
$\tilde{x}$: valeur de chaque horloge x de $\mathcal{A}$, dont les valeurs sont exprimées sous la forme $m\Delta$ avec $m \in \mathbb{N}$

```
1 Démarrer de l'état initial
2 pour tous les x̃ faire x̃ := 0;
3 pour chaque TAST A_i faire lancer la tâche ou attente associée à la localité initiale de A_i;
4 pour chaque pas de temps Δ                                   // Un quantum de temps Δ
5  faire
6     tant que des transitions sont exécutables faire en choisir une et l'exécuter;
7     pour tous les x̃ faire x̃ := x̃ + Δ;
8     attendre la fin du quantum de temps courant ;
9 fin
```

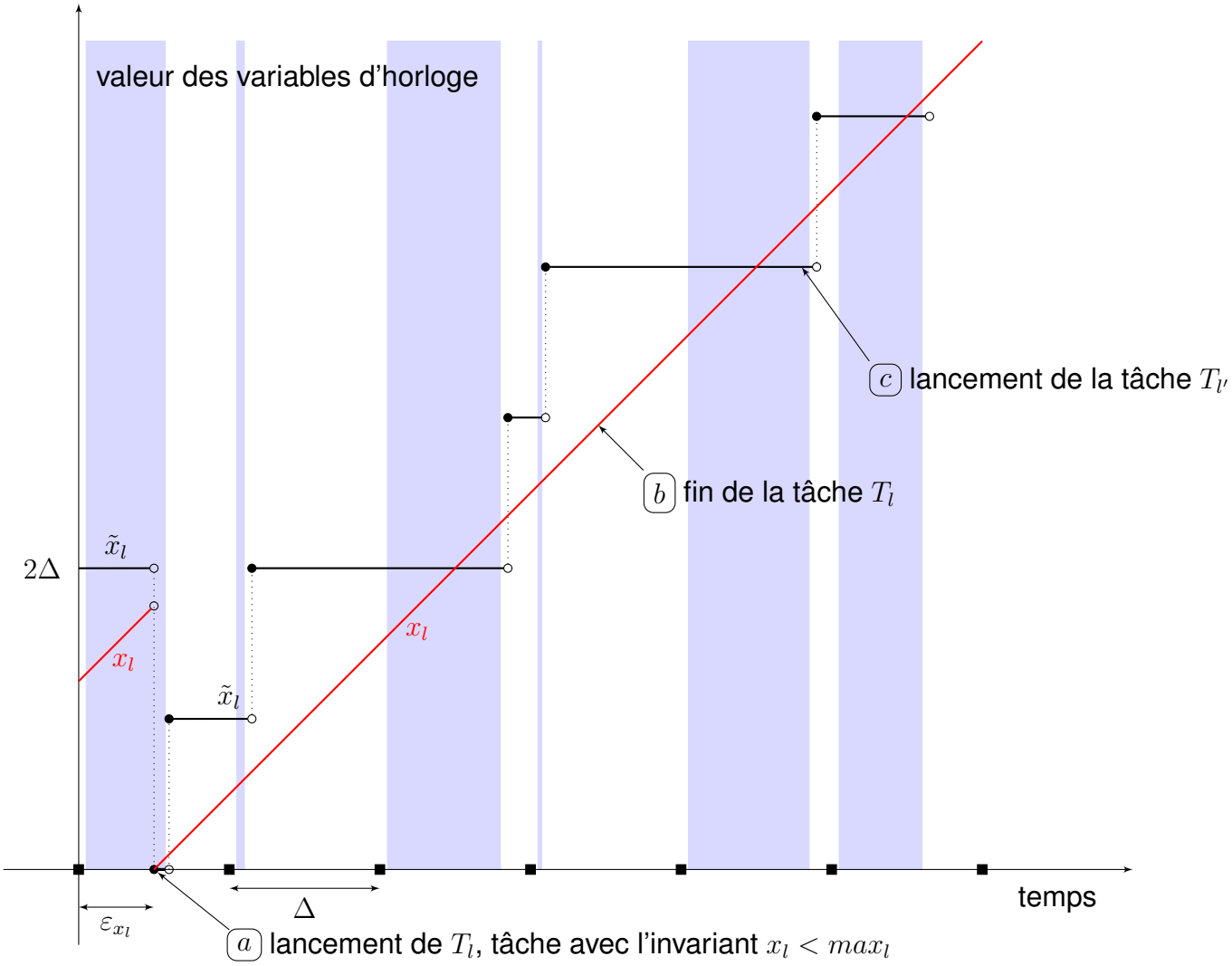


FIGURE 3.40 – Evolution de l'horloge discrète $\tilde{x}_l$ et de sa contrepartie continue x_l. Les zones colorés correspondent aux phases actives du contrôleur.

Altérations de comportement introduites par le contrôleur

Le contrôleur proposé, de part la nature discrète des variables associées au temps qu'il manipule, introduit des altérations par rapport à la spécification originelle $\mathcal{A}$. Certaines d'entre-elles sont représentées à la figure 3.40 qui prend pour exemple l'évolution d'une horloge x_l et de sa contrepartie discrète $\tilde{x}_l$ pour un chemin d'exécution commun de la spécification.

Le scénario présenté suppose, qu'après l'écoulement de quelques *quanta* de temps Δ, la variable d'horloge $\tilde{x}_l$ vaut 2Δ et x_l est également proche de cette valeur. Au cours de la phase de progression, après un délai de ϵ_{x_l}, nous entrons dans une localité l associée à une tâche. $\tilde{x}_l$ et x_l sont donc remises à zéro, et nous lançons la tâche T_l associée à l (point (a) du graphe). Après quelques *quanta* de temps, la tâche T_l se termine (point (b)) avec un délai proche de celui donné dans l'invariant (max_l). La détection de la fin d'exécution de la tâche et les actions correspondantes ne peuvent cependant s'exécuter que lors de l'itération suivante puisque l'évènement survient dans la phase d'attente et non pas de progression. Paradoxalement, alors que l'invariant est respecté par l'horloge physique x_l dans la spécification $\mathcal{A}$, le même invariant ne l'est pas dans le contrôleur par rapport à la variable discrète $\tilde{x}_l$. De plus, si la tâche T_l' est censé suivre immédiatement la tâche T_l, nous nous apercevons qu'un délai maximal de 2Δ peut survenir entre les deux. Ceci indique donc que le comportement de l'exécution de la spécification est altéré par rapport à celui de la spécification en elle-même.

L'approche « sandwich »

Comme nous l'avons vu, le comportement d'une spécification peut donc être différent de celui de son exécution. Le risque est que les résultats de l'analyse du comportement de l'un, ne puisse s'appliquer à l'autre.

L'idée de l'approche « sandwich » est que le comportement du contrôleur est encadré par la spécification originelle et une spécification élargie. Ainsi, si les comportements analysés sont équivalents dans les spécifications, alors nous pouvons garantir le comportement du contrôleur.

La spécification élargie, que l'on nommera $\overline{\mathcal{A}}$,consiste à modifier les gardes et invariants décrits par la spécification originelle. Cela dépend donc de la forme des prédicats employés dans les expressions utilisées :

— les prédicats de la forme $x \geq e$ sont transformés en $\tilde{x} > e - \Delta$, ce qui équivaut à $\tilde{x} \geq \Delta \lfloor \frac{e}{\Delta} \rfloor$, $\tilde{x}$ étant un multiple de Δ ;

— les prédicats de la forme $x < e$ sont transformés en $\tilde{x} < e + 2\Delta$, ce qui équivaut à $\tilde{x} \leq \Delta(\lceil \frac{e}{\Delta} \rceil + 1)$ pour la même raison que précédemment.

Il est également possible de produire une autre spécification, que l'on nommera $\widetilde{\mathcal{A}}$ et qui couvre le comportement du contrôleur. Elle suit les règles de transformation de la spécification $\mathcal{A}$, telles que décrites à la figure 3.41. Il est à noter que $\widetilde{\mathcal{A}}$ est construite à l'aide d'automates temporisés standards et non plus de TASTs. Cette dernière spécification est cependant plus longue à analyser, étant donné que le temps est « haché » en de très fines périodes Δ. Les outils de *model-checking* devant explorer l'espace des états générés par la spécification, ce nouvel espace est beaucoup plus vaste à explorer que le précédent.

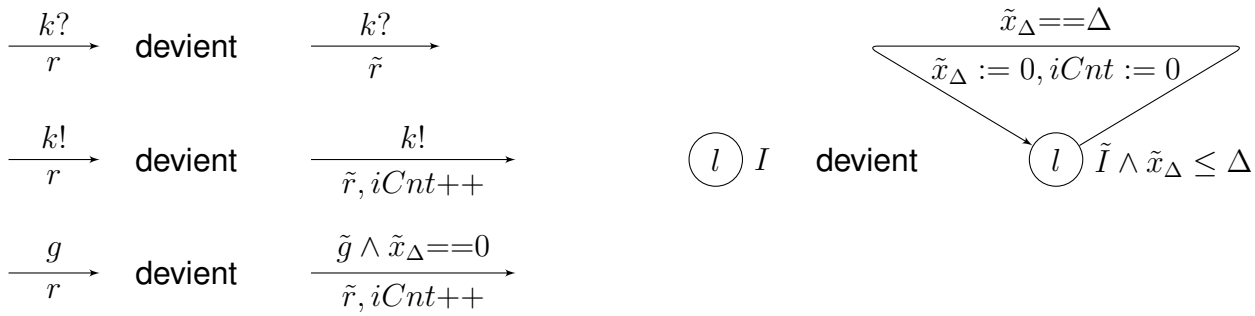
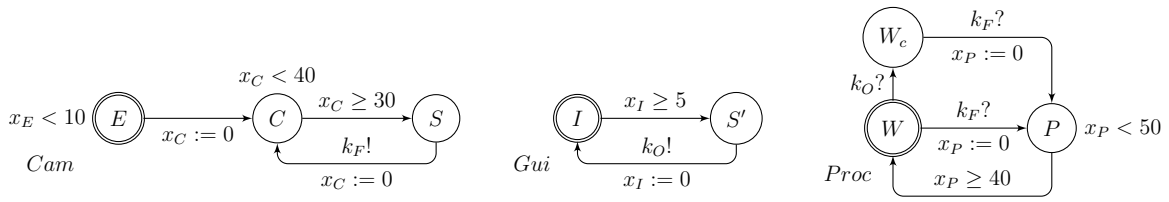


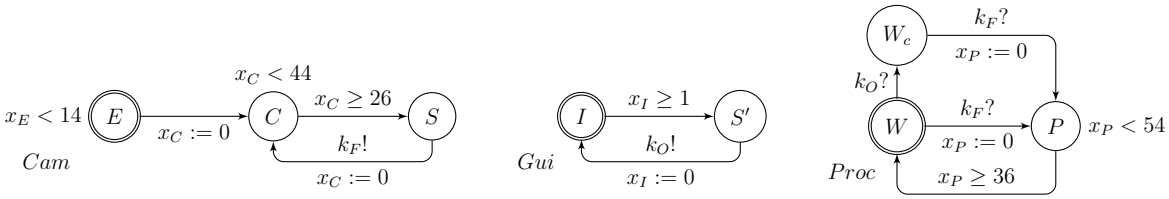
FIGURE 3.41 – Règle de transformation d'une spécification en vue de la sur-approximer.

Pour donner une vue d'ensemble des spécifications produites, à la figure 3.42 page suivante, nous partons d'un exemple réduit comportant une caméra, l'interception d'évènements provenant de l'interface graphique et une unité de traitement (ce qui ne constitue pas une application complète de réalité mixte). Nous en tirons deux spécifications pour analyser le comportement de l'exécution avec un pas de temps de 2ms, à savoir l'élargie $\overline{\mathcal{A}}$ (figure 3.42(b)) et le comportement du contrôleur (figure 3.42(c)).

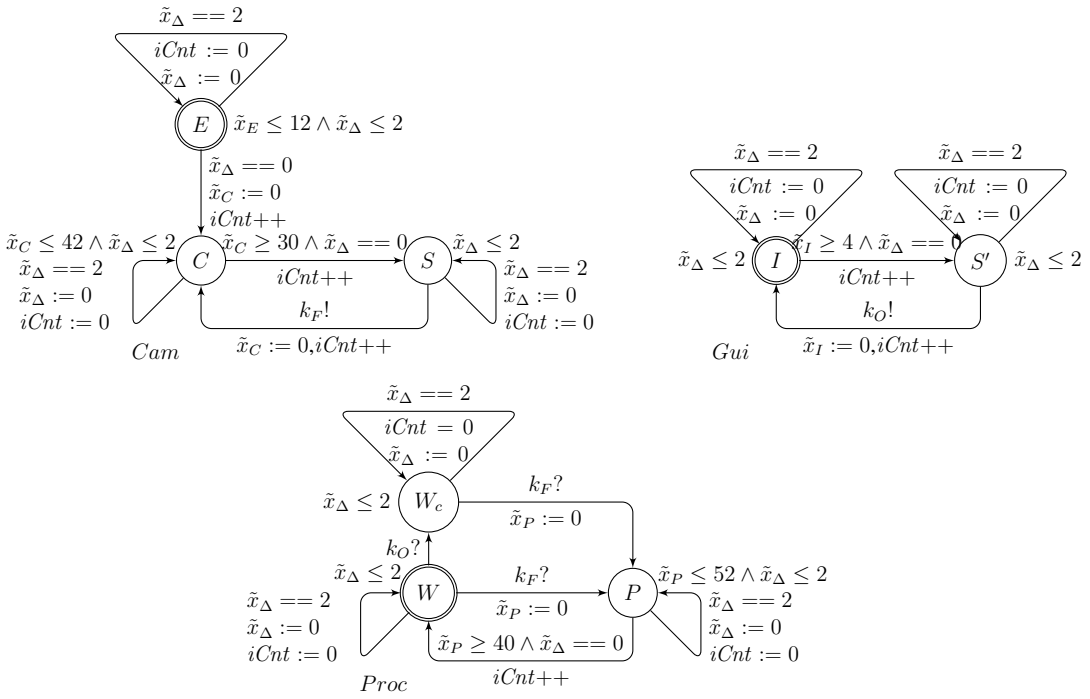
L'analyse des spécifications données en exemple permet ensuite de réaliser quelques vérifications (ici réalisées avec UPPAAL et non automatisées, ces travaux étant antérieurs à ceux présentés en section 3.3.2.3 page 122). Les résultats sont reportés dans la table 3.9 page 131. La spécification ne contenant ni mémoire partagée, ni boucle de rendu, nous vérifions si nous pouvons déboucher sur une situation d'impasse globale (requête `A[] not deadlock`). Nous vérifions également certaines propriétés d'atteignabilité (qui indiquent si la tâche de traitement est réellement effectuée ou non (requête `A<> Proc.P`) et si le composant de traitement est capable de répéter sa tâche à l'infini (requête `Proc.P --> Proc.W` et `Proc.W --> Proc.P`). On notera au passage que répondre à une requête sur la spécification $\widetilde{\mathcal{A}}$ nécessite d'explorer davantage d'états.



(a) Spécification de départ ($\mathcal{A}$)



(b) Spécification élargie pour Δ = 2ms ($\overline{\mathcal{A}}$)



(c) Spécification de l'exécution pour Δ=2ms ($\widetilde{\mathcal{A}}$)

FIGURE 3.42 – Spécifications d'origine, élargie et du comportement du contrôleur.

Du choix du quantum de temps

Pour le moment, nous avions occulté le problème du choix d'un paramètre : le *quantum* de temps (Δ). Notre étude dans [DEVILLERS et al. 2013] se complète par certaines règles (ainsi que les preuves associées) pour indiquer comment choisir ce dernier.

Tout d'abord, pour garantir une observabilité des phénomènes, Δ doit être plus petit que de moitié de la plus petite constante de temps du système [6].

En effet, la phase de progression dans chaque itération du contrôleur prend également du temps. Si la durée de cette phase est supérieure au *quantum* de temps choisi, le comportement du système n'est plus garanti.

6. Il est intéressant de remarquer que ce résultat est à rapprocher du théorème d'échantillonnage de Shannon, connu dans le domaine du traitement du signal. Les preuves que nous indiquons dans [DEVILLERS et al. 2013] sont toutefois d'une nature différente par rapport aux démonstrations usuelles de ce théorème.

Requête	Modèle	Etats explorés	Résultat
`A[] not deadlock`	$\mathcal{A}$	17	`true`
`A[] not deadlock`	$\widetilde{\mathcal{A}}$	271	`true`
`A<> Proc.P`	$\overline{\mathcal{A}}$	0	`true`
`Proc.P --> Proc.W`	$\overline{\mathcal{A}}$	19	`true`
`Proc.W --> Proc.P`	$\overline{\mathcal{A}}$	19	`true`
`sup: iCnt`	$\widetilde{\mathcal{A}}$	365	5

TABLE 3.9 – Requêtes utilisées sur l'exemple de la figure 3.42.

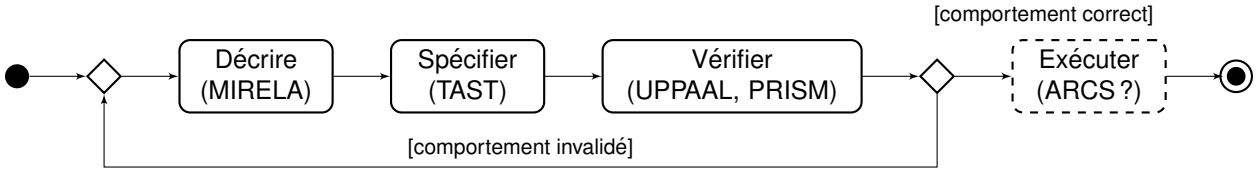


FIGURE 3.43 – *Workflow* proposé avec MIRELA

Ceci nous amène à l'inégalité permettant d'encadrer la valeur de Δ :

$$max_{iter}(n\Delta_g\delta_g + n(n-1)\Delta_c\delta_c^+\delta_c^- + 2\Delta_l) < \Delta < \frac{1}{2}min(D)$$

Où :

- D est l'ensemble des constantes de temps positives de la spécification ;
- n est le nombre d'automates de la spécification ;
- max_{iter} est le nombre maximal d'itérations dans la phase de progression ;
- Δ_g est le temps nécessaire à déterminer si une garde est satisfaite et δ_g est le nombre maximal d'arcs sortants d'une localité et munis d'une garde ;
- Δ_c est le temps nécessaire pour vérifier si une synchronisation est possible, δ_c^+ est le nombre maximal d'arcs sortant de la localité étiquetés pour une émission et δ_c^-, le nombre maximal d'arcs sortant étiquetés pour une réception ;
- Δ_l est le temps maximal nécessaire pour choisir une transition exécutable et lancer la tâche associée à la nouvelle localité si nécessaire.

Certaines de ces constantes dépendent de la plate-forme d'exécution, telles que Δ_g, Δ_c et Δ_l. Les autres, n, δ_g, δ_c^+, δ_c^- ainsi que max_{iter} dépendent de la spécification originelle. max_{iter} peut être borné par l'inégalité $max_{iter} < \sum d'_{ii} + 1$ où d'_i est la longueur du chemin pouvant être exécuté en temps nul dans l'automate $\mathcal{A}_i$ de la spécification $\mathcal{A}$. Toutefois, si cette borne est trop large, il est possible de l'obtenir par l'instrumentation de la spécification $\widetilde{\mathcal{A}}$ pour compter le nombre d'itération maximum possible dans un *quantum* de temps donné (c'est le sens de la requête `sup: iCnt` du tableau 3.9 qui indique $max_{iter} = 5$ dans notre exemple).

3.3.2.5 Vue d'ensemble du framework MIRELA

Ainsi, nous avons vu les différentes étapes du *framework* MIRELA. L'idée est ainsi de proposer un *workflow* qui part d'une description de haut niveau d'une application de réalité mixte jusqu'à son implémentation sur la plate-forme ciblée, en passant par des phases successives d'analyse afin de vérifier et garantir le comportement de l'application, du point de vue temporel.

Les étapes suivent celles représentées à la figure 3.43 auxquelles nous avons ajouté les outils présentés dans cette partie. Ainsi, la description se fait avec le langage MIRELA qui est ensuite traduite en une spécification à base de TAST. Cette dernière suit plusieurs étapes de vérification. Tout

d'abord pour analyser le comportement de la spécification à l'aide des outils UPPAAL et PRISM, puis pour vérifier le comportement de l'implémentation associée. Si l'une de ces étapes invalide le comportement, nous reprenons la description par exemple en relaxant certaines contraintes temporelles (il est parfois aussi possible d'altérer directement la spécification, ce qui est moins souhaitable dans une pratique d'ingénierie dirigée par les modèles). Enfin, l'objectif est de l'exécuter sur une plate-forme qui pourrait être ARCS.

Au niveau des travaux réalisés, les fondements théoriques sont posés. Les éventuelles difficultés peuvent venir du problème de l'espace d'état généré par les spécifications, ce qui peut compromettre la possibilité d'analyser le comportement. Quelques pistes ont été amorcées qui ne sont pas encore finalisées. Enfin, il reste à concrétiser la partie concernant l'implémentation afin de disposer d'une chaîne complète.

Sur cette partie, une autre piste de recherche consiste à mettre au point une méthode de rétro-ingénierie. L'idée serait d'écrire des outils d'interprétation du code qui permettraient de construire directement des réseaux de TAST à analyser. Ce serait plus dans une logique de traitement curatif de l'application lorsqu'un dysfonctionnement a été détecté dont l'origine, du aux contraintes temporelles n'est pas forcément bien comprise. En effet, l'analyse de mauvais comportement permet aussi, dans certains cas, d'obtenir une trace d'exécution indiquant ce qui a pu se passer dans l'application.

3.4 Résumé des contributions du chapitre

Dans ce chapitre, nous avons traité trois volets associés aux exigences non fonctionnelles des applications de réalité augmentée.

Nous avons ainsi proposé une architecture basée composants qui permet de résoudre les problèmes de modularité, est reconfigurable à l'exécution et qui possède des capacités d'intégration étendue. Cela a permis de poser les concepts qui ont été implémentés dans le *framework* ARCS.

Nous avons ensuite montré comment nous pouvions tirer partie des capacités du *framework* pour étendre ses capacités afin de gérer la distribution des traitements de deux manières : par l'utilisation d'un protocole *ad-hoc* ou par l'utilisation de *webservices*. Nous nous sommes aussi attelés à proposer une méthodologie, inspirée de SPE (*software performance engineering*) qui nous a permis de dénombrer les scénarios possibles de distribution et donner des recommandations générales sur les pratiques à favoriser. Quelques exemples d'application viennent valider expérimentalement les concepts explorés.

Enfin, nous avons traité des aspects concernant la concurrence et la gestion des contraintes temporelles. Nous avons vu comment intégrer une gestion automatique des situations de compétition entre composants au niveau du moteur du *framework* afin de décharger le développeur de cet aspect, tout en garantissant un certain niveau de performances. Puis, par la chaîne d'outils associées au langage MIRELA, nous avons vu comment décrire, spécifier et analyser les propriétés temporelles d'une application de réalité mixte, ce qui permet de détecter et de corriger les éventuels comportement indésirables de cette dernière avant de passer à son implémentation. Les outils introduits consistent en un langage de haut niveau, un modèle formel d'automates temporisés à l'expressivité restreinte afin de garantir certaines propriétés, les TASTs, et une méthodologie d'analyse de comportements indésirables.

Nous allons à présent nous intéresser davantage aux exigences fonctionnelles de la réalité augmentée qui seront abordées au cours du chapitre suivant.

Publications associées à ce chapitre

Architecture modulaire et reconfigurable

DIDIER, J.-Y. et al. [2012]. « ARCS : A framework with extended software integration capabilities to build Augmented Reality applications ». In : *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on*, p. 60–67. DOI : 10.1109/SEARIS.2012.6231170

DIDIER, J.-Y. et al. [2009b]. « ARCS : Une Architecture Logicielle Reconfigurable pour la conception des Applications de Réalité Augmentée ». In : *Technique et Science Informatiques (TSI), Réalité Virtuelle - Réalité Augmentée* 28.6-7/2009. Numéro spécial, p. 891–919

DIDIER, J.-Y. et al. [2006]. « A Component Model for Augmented/Mixed Reality Applications with Reconfigurable Data-flow ». In : *8th International Conference on Virtual Reality (VRIC 2006)*. Laval (France), p. 243–252

Distribution et applications de réalité augmentée

CHOUITEN, M. et al. [2014]. « Distributed Mixed reality for diving and underwater tasks using Remotely Operated Vehicles ». In : *International Journal on Computational Sciences & Applications* 5.4, elec–proc

CHOUITEN, M. [2013]. « Architecture distribuée dédiée aux applications de Réalité Augmentée mobile ». Thèse de doct. Université d'Evry val d'Essonne

CHOUITEN, M. et al. [2012a]. « Distributed Augmented Reality Systems : How Much Performance is Enough ? » In : *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, p. 337–342. DOI : 10.1109/ICMEW.2012.64

CHOUITEN, M. et al. [2012b]. « Distributed mixed reality for remote underwater telerobotics exploration ». In : *Proceedings of the 2012 Virtual Reality International Conference*. VRIC '12. Laval, France : ACM, 1 :1–1 :6. ISBN : 978-1-4503-1243-1. DOI : 10.1145/2331714.2331716. URL : http://doi.acm.org/10.1145/2331714.2331716

CHOUITEN, M. et al. [2011]. « Component-based middleware for distributed augmented reality applications ». In : *Proceedings of the 5th International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE 2011)*. Verona, Italy : ACM, p. 3

Concurrence et contraintes temporelles

ARCILE, J. et al. [2015a]. « Modelling and Analysing Mixed Reality Applications ». In : *Man-Machine Interactions 4 - 4th International Conference on Man-Machine Interactions, ICMMI 2015, Kocierz Pass, Poland, October 6-9, 2015*, p. 3–17. DOI : 10.1007/978-3-319-23437-3_1. URL : http://dx.doi.org/10.1007/978-3-319-23437-3_1

ARCILE, J. et al. [2015b]. « Indefinite waitings in MIRELA systems ». In : *Proceedings 4th International Workshop on Engineering Safety and Security Systems, ESSS 2015, Oslo, Norway, June 22, 2015*. P. 5–18. DOI : 10.4204/EPTCS.184.1. URL : http://dx.doi.org/10.4204/EPTCS.184.1

DEVILLERS, R. R. et al. [2014]. « Deadlock and Temporal Properties Analysis in Mixed Reality Applications ». In : *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, p. 55–65. DOI : 10.1109/ISSRE.2014.33. URL : http://dx.doi.org/10.1109/ISSRE.2014.33

DIDIER, J.-Y. et MALLEM, M. [2014]. « A new approach to detect potential race conditions in component-based systems ». In : *Proceedings of the 17th international ACM Sigsoft symposium on*

Component-based software engineering. ACM, p. 97–106

DEVILLERS, R. et al. [2013]. « Implementing timed automata specifications : the "sandwich" approach ». In : *13th International Conference on Application of Concurrency to System Design (ACSD2013)*

DIDIER, J.-Y. et al. [2013]. « An Improved Approach to Build Safer Mixed Reality Systems by Analysing Time Constraints ». In : *Joint Virtual Reality Conference (JVRC) 2013 Poster, Demo and Industrial Track*, p. 87–90

DIDIER, J.-Y. et MALLEM, M. [2012]. « Automated Concurrent Access Management for Components of Augmented Reality Applications ». In : *7èmes journées de l'AFRV*. Strasbourg, France, p. 13–19

DIDIER, J.-Y. et al. [2009a]. « The MIRELA framework : modeling and analyzing mixed reality applications using timed automata ». In : *Journal of Virtual Reality and Broadcasting*. VRIC 2008 (Laval Virtual) Special Issue 6.1. Sous la dir. de J. HERDER et al. t urn :nbn :de :0009-6-17423,, ISSN 1860-2037. URL : http://www.jvrb.org/archiv/1742/

DIDIER, J.-Y. et al. [2008d]. « The MIRELA Framework : modeling and analyzing mixed reality applications using timed automata ». In : *10th Virtual Reality International Conference*. Laval, France, p. 189–199

DIDIER, J.-Y. et al. [2008c]. « Modeling and analyzing mixed reality applications using timed automata ». In : *1st Mediterranean Conference on Intelligent Systems and Automation (CISA ?08)*. Sous la dir. de H. ARIOUI et al. AIP, p. 173–178. URL : http://evra.ibisc.univ-evry.fr/Proceedings/CISA08/cdr_pdfs/indexed/stage4_copyr/173_1.pdf

Chapitre 4

Localisation, interprétation et interaction en réalité augmentée

Ce chapitre a pour objectif de réaliser la synthèse sur les travaux menés en relation avec les exigences fonctionnelles de la réalité augmentée. Nous aborderons plus particulièrement les points touchant à la fonction de localisation d'un système au sein de son espace de travail, puis nous verrons que nous pouvons aller plus loin en termes d'interprétation de ce qui se passe dans la scène tant au niveau des processus métiers visualisés qu'au niveau des interactions avec l'utilisateur. Enfin, nous aborderons quelques pistes concernant le traitement de la multi-modalité, en particulier dans le cas de combinaisons visuelles et haptiques.

Ces applications peuvent être vues comme des études de cas, qui permettent d'enrichir la conception et améliorer les architectures logicielles par le biais de deux voies :

— Aborder les problématiques des systèmes de réalité augmentée permet de rester proche de ces dernières en vue de proposer des solutions adaptées en termes d'architecture logicielle ;

— Les études de cas pour lesquelles nous allons jusqu'à l'implémentation permettent de valider les concepts architecturaux présentés.

Une partie des travaux présentés seront donc parfois implémentés sous le *framework* ARCS que nous avons proposé, ce qui sera alors signalé. L'ordre de présentation de ces travaux ne suit pas un ordre chronologique mais plutôt thématique.

4.1 Localisation et recalage en réalité augmentée

De part les caractéristiques intrinsèques d'un système de réalité augmentée (voir définition 2.4 page 50), la localisation est une de ses fonctionnalités essentielles puisqu'elle permet le recalage des objets virtuels sur les objets réels.

De ce fait, la littérature traitant de cette problématique est très abondante. Pour l'expliquer rapidement, l'arsenal de méthodes de localisation dépend très fortement du système et de l'application visée. Nous avons brièvement présenté en section 2.1.2.1 page 51 les différentes techniques utilisées pour la localisation.

Nous allons à présent introduire quelques uns de nos travaux dans le contexte du projet RAXENV, dans lequel nous avons développé un capteur hybride.

4.1.1 Contexte du projet RAXENV

Le projet RAXENV [FRAUCIEL et al. 2008], pour Réalité augmentée en eXtérieur appliquée aux métiers de l'ENVironnement, a fait partie du programme Technologies Logicielles de l'Agence Nationale pour la Recherche de 2007 à 2010. Ce projet visait à démontrer la faisabilité et l'utilisabilité

d'un système de réalité augmentée dans le domaine des géotechniques. Le consortium mené par le Bureau des Ressources Géologiques et Minières (BRGM) regroupait l'Université de Bordeaux I et son Laboratoire Bordelais de Recherche en Informatique (LaBRI), l'Université d'Evry-val d'Essonne et son laboratoire IBISC et une entreprise spécialisée dans les reconstructions 3D de milieux urbains, ArchiVideo.

Les fonctionnalités principales développées dans ce projet sont :

— se localiser spatialement à l'aide d'un système de positionnement primaire de type GPS et en s'aidant de « marqueurs » naturels ou artificiels dont la position est consignée dans des données géoréférencées locales ou distantes (modèle numérique de terrain, système d'information géographique, modèle urbain, ...) ;

— offrir à son utilisateur des fonctionnalités de visualisation et d'interrogation de données de surface et de sous-sol.

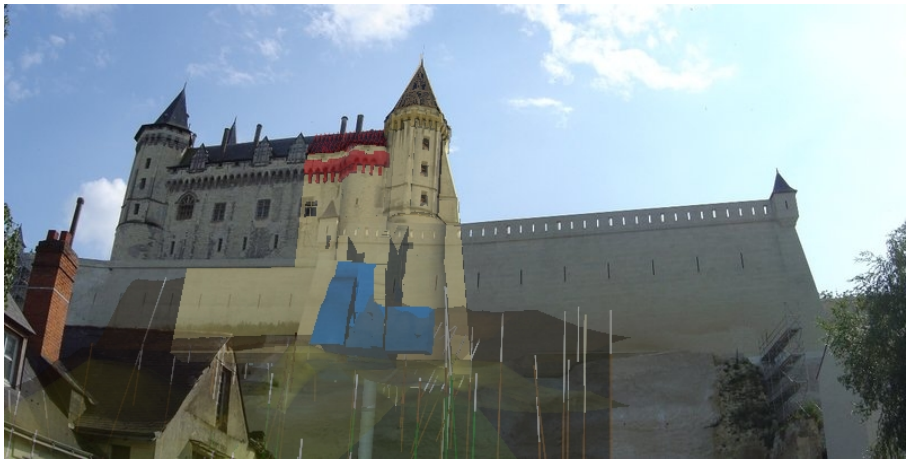


FIGURE 4.1 – Concept du projet RAXENV (montage photographique)

Dans sa configuration initiale, le projet devait traiter trois scénarios de cas d'utilisation. Toutefois, seul l'un d'entre eux a été vraiment mené à terme. Il s'agit du suivi de chantier géotechnique. Dans ce cadre, nous avons travaillé sur le site du Château de Saumur. En 2001, le rempart d'enceinte de ce site s'était effondré, les éboulis ayant atteint les habitations en contrebas. Le BRGM, alors dépêché sur place, avait effectué quelques 300 forages pour déterminer la géologie du sous-sol. La tour nord de l'édifice qui était aussi menacée a été placée sous surveillance et a été intégralement scannée pour être reconstruite en 3D. L'idée était de pouvoir, à l'aide du système de réalité augmentée, accéder à ces données dans leur contexte géolocalisé.

Le laboratoire IBISC, dans ce contexte, avait la charge de développer la fonctionnalité de localisation de ce système. Étant donné la large échelle de l'espace de travail de l'application, nous nous sommes tournés vers un capteur hybride composé d'un récepteur GPS, d'une centrale inertielle et d'une caméra.

4.1.2 Capteurs et stratégie d'hybridation

Dans cette partie, nous verrons un état de l'art succinct sur les capteurs hybride. Puis, nous détaillerons la stratégie que nous avons retenue pour l'hybridation qui emprunte le chemin de la suppléance et non pas de le schéma habituel de fusion. Enfin, nous traiterons de la problématique de calibration d'un capteur hybride, ses sous-systèmes ne mesurant pas les informations concernant la trajectoire dans le même repère.

4.1.2.1 Bref état de l'art sur les capteurs hybrides

L'idée de combiner différents capteurs pour localiser un système n'est pas une chose nouvelle. Par exemple, dans le domaine de la robotique mobile, [VIÉVILLE et al. 1993] propose d'associer une caméra et une centrale inertielle en vue de corriger la trajectoire d'un robot. Cette idée a ensuite été reprise par la communauté de la réalité augmentée. De nombreux travaux ont proposé de fusionner les informations provenant d'une caméra et d'une centrale inertielle à l'aide d'un filtre de Kalman [YOU et al. 1999 ; RIBO et al. 2002 ; HOL et al. 2006 ; BLESER et STRICKER 2009] ou d'un filtre particulaire [ABABSA et MALLEM 2007]. La stratégie consiste alors à fusionner l'ensemble des données pour localiser le système en suivant un modèle de type prédiction/correction.

D'autres schémas sont aussi développés avec les capteurs hybride. En effet, [ARON et al. 2007] utilise les informations d'une centrale inertielle pour prendre le relais de la caméra en cas de défaillance du processus de vision afin d'estimer l'orientation du système complet. [MAIDI et al. 2009] ont utilisé la même combinaison pour estimer également la position.

4.1.2.2 Stratégie d'hybridation par suppléance

Dans notre cas, nous subdivisons les capteurs en deux classes d'équivalence, chacune étant capable de fournir la position et l'orientation du système, à savoir :

1. La caméra qui, associée à un processus de vision par ordinateur, fournit la position et l'orientation du système ;
2. Le récepteur GPS et la centrale inertielle qui mesurent respectivement une position et une orientation.

Chaque classe de capteur dispose de ses avantages et de ses inconvénients :

— l'estimation de la pose obtenue à partir des images de la caméra permet de recaler au plus près les objets virtuels sur le réel. Celle-ci sera plus approximative avec le couple GPS - centrale inertielle. En effet, la précision d'un récepteur GPS grand public est de l'ordre d'une dizaine de mètres [WING et al. 2005] et notre centrale inertielle, intégrant un magnétomètre, peut être perturbée par les champs magnétiques à proximité ;

— la pose fournie par le couple GPS et centrale inertielle est utilisable en tout temps, contrairement à celle estimée par traitement d'image pour des raisons diverses (changements de luminosité, occultations partielles de la scène, ...)

En partant de ce constat, nous nous sommes positionnés de la manière suivante :

Positionnement 4.1 : Schéma d'assistance

Nous proposons un schéma d'assistance entre deux classes équivalentes de capteurs.

1. En temps normal, nous nous appuyons sur la pose obtenue à partir des images de la caméra pour localiser le système ;
2. En cas de défaillance du capteur précédent, le sous-système auxiliaire composé du récepteur GPS et de la centrale inertielle prend le relais pour la localisation.

La vue globale du système correspond à celle de la figure 4.2 page suivante. Nous allons présenter la plate-forme matérielle ainsi que la stratégie d'hybridation retenue, par ailleurs publiée dans [ZENDJEBIL et al. 2011].

Plate-forme matérielle

Du point de vue du matériel, notre plate-forme se compose d'une tablette-PC Latitude XT (Dell) pourvu d'un processeur Intel Core2 Duo cadencé à 1,33GHz, d'une centrale inertielle MTi (XSens),

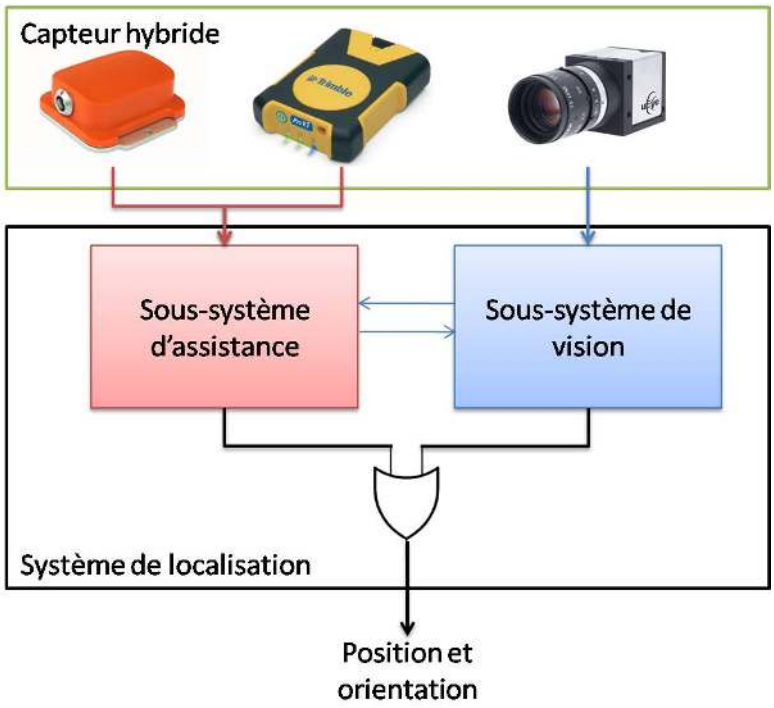


FIGURE 4.2 – Vue globale du système de localisation

(a) Tablette PC, caméra et centrale inertielle (b) Récepteur GPS

FIGURE 4.3 – Plateforme matérielle du projet RAXENV

d'une camera UI-2220-RC-C (uEye) et d'un récepteur GPS Pathfinder ProXT de Trimble (voir figure 4.3). La centrale inertielle est elle-même une agrégation de plusieurs capteurs : un accéléromètre mesurant le champ d'accélération auquel elle est soumise (y compris la gravité), un gyromètre et un magnétomètre (indiquant le nord magnétique). La caméra et la centrale inertielle sont rigidement liées l'une à l'autre et communiquent avec la tablette via USB. Le GPS transmet ses données en utilisant la norme Bluetooth. Au niveau des cadences de réception des informations,

— les trames du GPS sont transmises à 1Hz,

— la caméra acquiert 25 images de taille 768×576 par seconde,

— la centrale inertielle opère à 100Hz.

Stratégie d'hybridation

La figure 4.4 page suivante indique la manière nous avons décomposé le fonctionnement de notre capteur hybride de localisation en états nominaux à savoir :

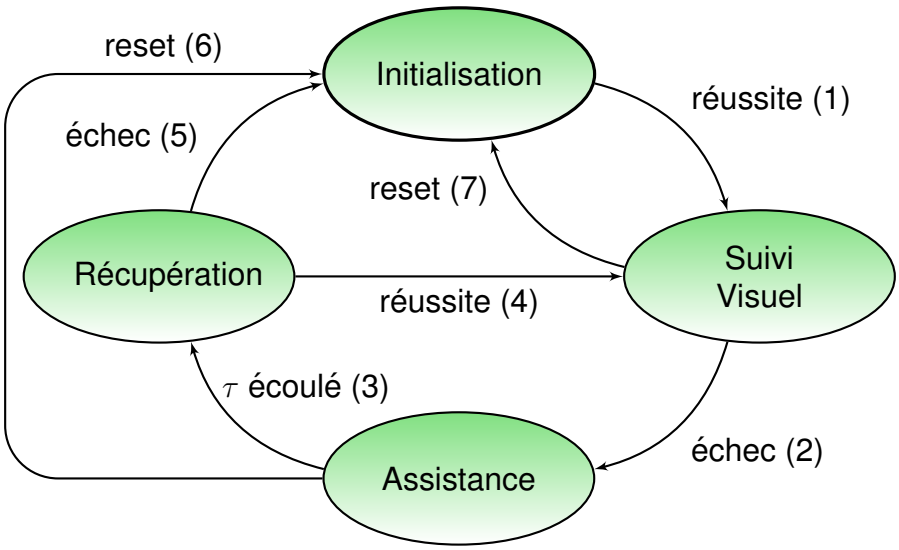


FIGURE 4.4 – Décomposition par état du fonctionnement du capteur de localisation hybride

— Un état d'*initialisation* où le système recherche des appariements 2D/3D suivant une procédure semi-automatique ;

— Un état où est estimée la localisation à l'aide d'un processus visuel (*suivi visuel*) ;

— Un état où le recepteur GPS et la centrale inertielle alimentent le système en données de localisation (*assistance*) ;

— Un état intermédiaire de (*récupération*) du suivi par la vision.

Les transitions traduisent les conditions suivantes :

1. L'initialisation est validée et réussie ;
2. La pose estimée est invalidée car le résultat ne concorde pas avec les hypothèses nécessaires au suivi visuel ;
3. Après une période de temps τ écoulée, le système tente une réinitialisation du processus de suivi visuel ;
4. En cas de réussite de la récupération, le système bascule vers l'état de suivi visuel ;
5. En cas d'échec de la récupération, le système bascule vers l'état d'initialisation ;
6. Sur intervention de l'utilisateur, une réinitialisation est possible. Cela est aussi vrai pour la transition (7).

L'une des difficultés de l'hybridation des capteurs est d'exprimer les relations mathématiques qui existent entre leurs mesures. La phase de calibration consiste à identifier les paramètres de ces relations. Nous allons voir comment cela est résolu dans la section suivante et détaillerons le fonctionnement des différents états de notre système.

4.1.3 Calibration du capteur hybride

Les capteurs mesurent des valeurs par rapport à des référentiels différents. Ainsi, la pose de la caméra est estimée par rapport à un repère monde arbitraire. La centrale inertielle mesure son orientation par rapport à un repère orienté suivant la direction de la gravité et le nord magnétique. Le GPS situe le dispositif sur un géoïde de référence qui est une approximation du globe terrestre.

Ainsi, les capteurs fournissent des données par rapport à des repères différents et leurs mesures ne sont pas toutes exprimées dans le même espace métrique.

Nous allons voir comment unifier ces données dans un même espace, tout d'abord en exprimant les relations de passage entre le repère local à la caméra et à la centrale inertielle, puis entre le récepteur GPS et la caméra. Nous avons publié cette méthode dans [ZENDJEBIL et al. 2008b ; ZENDJEBIL et al. 2010].

Détermination de la relation entre la caméra et la centrale inertielle

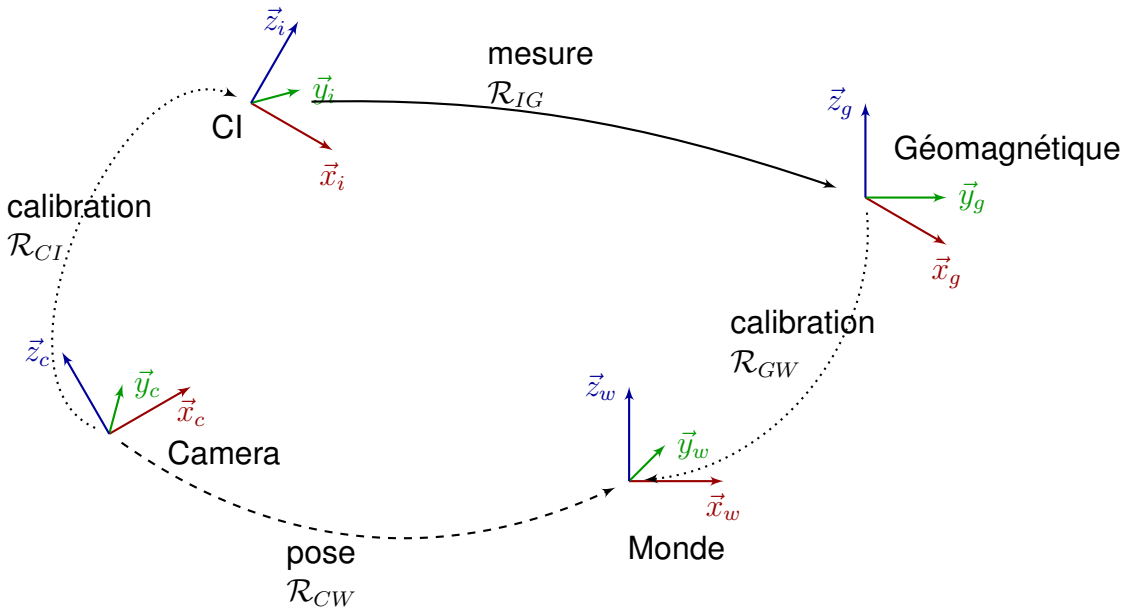


FIGURE 4.5 – Orientations associées aux capteurs du système RAXENV

La centrale inertielle ne fournissant qu'une information en orientation, nous devons déterminer la matrice de rotation, composante de la transformation la liant à la caméra. La figure 4.5 indique les repères qui interviennent dans les processus de mesure et de calibration. Ainsi, nous notons :

- $\mathcal{R}_{CW}$ la rotation du repère caméra (C) par rapport au repère monde (W) obtenue par estimation de la pose ;
- $\mathcal{R}_{CI}$, la rotation du repère caméra par rapport au repère de la centrale inertielle (I) ;
- $\mathcal{R}_{IG}$, rotation de la centrale inertielle (mesurée par cette dernière) par rapport au repère géomagnétique (G) ;
- $\mathcal{R}_{GW}$ représente la rotation entre le repère géomagnétique et le repère monde.

La relation entre les rotations s'écrit donc :

$$\mathcal{R}_{CW} = \mathcal{R}_{CI}\mathcal{R}_{IG}\mathcal{R}_{GW} \tag{4.1}$$

Nous posons une hypothèse supplémentaire : la direction verticale des repères monde et géomagnétique sont colinéraires. Ainsi, la transformation entre les deux repères est une rotation d'angle θ autour de l'axe $\vec{z}_g = \vec{z}_w$, ce qui se traduit, pour la rotation $\mathcal{R}_{GW}$ par :

$$\mathcal{R}_{GW} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

En notant $\mathcal{R}_{IG} = (r_i^{IG})$ pour $i = \{1, 2, 3\}$, les colonnes de la matrice $\mathcal{R}_{IG}$, l'équation 4.1 se réécrit :

$$\mathcal{R}_{CW} = \begin{pmatrix} \mathcal{R}_{CI}r_1^{IG} & \mathcal{R}_{CI}r_2^{IG} & \mathcal{R}_{CI}r_3^{IG} \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

Nous pouvons en déduire

$$r_3^{CW} = R_{CI} r_3^{IG} \tag{4.4}$$

Si nous traduisons cette équation sous une forme linéaire ($Ax = b$) pour au moins trois jeux de mesures, nous sommes en capacité d'estimer $\mathcal{R}_{CI}$ à l'aide de la méthode des moindres carrés. Une fois cette matrice estimée, $\mathcal{R}_G W$ s'obtient à partir de l'équation 4.1 :

$$R_{GW} = R_{IG}^T R_{CI}^T R_{CW} \tag{4.5}$$

Détermination de la relation entre les positions du GPS et celles de la caméra

Le récepteur GPS fournit ses coordonnées selon le repère défini par le système géodésique WGS84, ce qui en fait un système de coordonnées non euclidien. Nous avons donc décidé de convertir ces coordonnées en nous basant sur la projection conique conforme de Lambert, utilisée en France pour les coordonnées cartographiques, ce qui nous fournit un repère métrique $\mathcal{R}_L$. Pour compléter, la donnée d'altitude peut-être mesurée par le récepteur GPS, toutefois, au vu de sa fiabilité, il est plus efficace de l'obtenir à partir d'un modèle d'élévation de terrain. Nous lui ajoutons une hauteur moyenne à laquelle l'utilisateur porte le dispositif.

Nous devons estimer la transformation $\mathcal{T}_{WL} = (\mathcal{R}_{WL}, t_{WL})$ entre le repère monde et le repère cartographique. Pour chaque position du GPS transformée (notée p_{gps}), nous la mettons en relation avec la position de la caméra (notée p_{cam}) exprimée dans le repère monde :

$$p_{cam} = \mathcal{R}_{WL} p_{gps} + t_{WL} \tag{4.6}$$

Cette transformation est obtenue en mesurant n positions successives de la caméra et du récepteur GPS, puis en minimisant le critère suivant :

$$\sum_i^n \| p_{cam}^i - \mathcal{R}_{WL} p_{gps}^i + t_{WL} \|^2 \tag{4.7}$$

Pour déterminer la transformation optimale, nous définissons les vecteurs normalisés $\vec{N}_{gps}^i$ et $\vec{N}_{cam}^i$ associés à p_{gps}^1 et p_{cam}^i par les relations :

$$\overrightarrow{N}^i_{gps} = \frac{p_{gps}^j - p_{gps}^i}{\|p_{gps}^j - p_{gps}^i\|} \qquad \overrightarrow{N}^i_{cam} = \frac{p_{cam}^j - p_{cam}^i}{\|p_{cam}^j - p_{cam}^i\|} \tag{4.8}$$

Avec $i = [\![1, \frac{n}{2}]\!]$ et $j = [\![\frac{n}{2} + 1..n]\!]$. La relation entre ces vecteurs est donc :

$$\sum_i^n \| \overrightarrow{N}^i_{cam} - \mathcal{R}_{WL} \overrightarrow{N}^i_{gps} \|^2 \tag{4.9}$$

De l'équation 4.9, nous pouvons estimer la rotation $\mathcal{R}_{WL}$ selon la méthode décrite dans le chapitre 7 de [HORAUD et MONGA 1995]. Celle-ci utilise la représentation axe-angle $(\vec{n}, \theta)$ pour une estimation de la rotation optimale. Une fois qu'elle est déterminée, nous pouvons déduire la translation de la manière suivante :

$$t_{WL} = \overline{p}_{cam} - \mathcal{R}_{WL} \overline{p}_{gps} \tag{4.10}$$

$\overline{p}_{cam}$ et $\overline{p}_{gps}$ étant les vecteurs moyens des n positions successive de la caméra et du GPS.

Validation de la procédure de calibration

Afin de valider notre procédure de calibration, nous avons déplacé notre système en 20 points de test différents. Pour chacun, nous avons collecté les mesures des capteurs et avons déterminé l'écart entre l'orientation de la caméra obtenue par estimation de la pose et celle calculée à partir de la centrale inertielle. Le même mode opératoire a été appliqué par rapport au récepteur GPS.

Nous avons obtenu les résultats présentés à la table 4.1. Un protocole d'évaluation de la qualité de la calibration plus conséquent a été également mis en place dont les détails sont consignés dans [ZENDJEBIL et al. 2010]. Les valeurs obtenues mettent en avant une erreur raisonnable au vu de la distance d'observation du bâtiment. Nous verrons toutefois comment nous pouvons réduire cet écart à l'aide d'un processus de correction en ligne.

(a) Erreurs angulaires

	ϕ	θ	ψ
Erreur moyenne (en $^\circ$)	0.3460	0.7211	2.293

(b) Erreurs en position

	μ_x	σ_x^2	μ_y	σ_y^2
Valeurs	$0.7572m$	$1.7087m^2$	$2.344m$	$1.8473m^2$

TABLE 4.1 – Écart entre la pose de la caméra estimée par vision et celle obtenue par la centrale inertielle et le récepteur GPS

4.1.4 Détails des états du capteur hybride

Le mode opératoire de notre capteur hybride implique un fonctionnement fractionné en quatre états (voir figure 4.4 page 139). Leurs détails seront à présent examiné.

4.1.4.1 Initialisation semi-automatique

Cette approche de localisation semi-automatique implique l'intervention de l'utilisateur. Elle se déroule en trois phases :

1. Les coordonnées obtenues par le récepteur GPS permettent de sélectionner des aspects du modèle 3D à afficher en fil de fer à destination de l'interface ;
2. L'utilisateur sélectionne l'aspect le plus proche de son angle de vue sur l'édifice observé, puis se positionne à un endroit où l'aspect en fil de fer est approximativement aligné avec sa contre-partie réelle et signale quand il estime être bien placé ;
3. Le système procède à une mise en correspondance précise du modèle 3D et du bâtiment (voir figure 4.6 page suivante).

Pour la mise en correspondance, nous projetons les points 3D de l'aspect sur l'image et définissons une zone de recherche rectangulaire autour de ces derniers. Des points d'intérêt sont extraits de ces régions à l'aide du détecteur de Harris [HARRIS 1992] pour rechercher des coins, éléments saillants dans le modèle 3D. Un descripteur SURF [BAY et al. 2006] [1] est associé à leur voisinage. Ils sont ensuite comparés avec des descripteurs de référence associés aux points 3D du modèle. RANSAC [FISCHLER et BOLLES 1981] est utilisé pour filtrer les points aberrants pour améliorer la précision de la localisation. La pose est ensuite estimée à l'aide de l'algorithme de l'itération orthogonale [LU et al. 2000].

1. Ce descripteur, à l'époque de ces travaux était celui considéré comme ayant le meilleur compromis entre temps de calcul et robustesse de la mise en correspondance.

(a) Initalisation manuelle par l'utilisateur (b) Mise en correspondance par le système

FIGURE 4.6 – Étapes de l'initialisation semi-automatique

Cette étape est validée par comparaison de la pose estimée (P_a) avec celle utilisée pour générer l'aspect (P). Cela est réalisé en calculant la trace du produit matriciel ($P_a P^{-1}$). Si elle est supérieure à un seuil empiriquement déterminé, l'étape d'initialisation est réussie.

4.1.4.2 Localisation par la vision

Cet état du système accorde la pré-éminence au sous-système de vision. Dans le même temps, nous enregistrons les écarts de mesure relevés avec le sous-système d'assistance. Cela nous permet, lorsque la vision est désactivée, de prédire la localisation du point de vue. Cet état comporte deux processus distincts :

— La localisation par la vision ;

— L'estimation des erreurs de localisation du sous-système d'assistance.

Nous décrirons les hypothèses que doit vérifier la pose estimée lors du suivi visuel, une violation de ces dernières poussant le système à estimer que le suivi est un échec.

Suivi visuel

Une fois l'initialisation réalisée, nous passons dans un mode de suivi. Pour estimer la pose de la caméra, nous devons conserver les correspondances 2D/3D sur chaque image. Nous le réalisons à l'aide d'un suivi 2D-2D. Il se base sur une hypothèse de cohérence spatiale et temporelle : les primitives se déplaçant peu entre deux images successives. Pour les suivre, il suffit de faire une recherche dans le voisinage de leur coordonnées. La méthode retenue, en raison de sa rapidité, est le *tracker* KLT [TOMASI et KANADE 1991]. Cet algorithme de flot optique permet de suivre un ensemble de points d'une image I_{t-1} à l'image courante I_t. Afin de minimiser le temps de recherche, KLT se base sur une pyramide d'images. Le suivi est donc réalisé à un niveau grossier d'abord, puis affiné, ce qui permet un suivi sur une longue distance et de manière précise. La pose est estimée à l'aide de l'algorithme de l'itération orthogonale et s'appuie sur les correspondances 2D-3D maintenues. En revanche, cette approche rapide et précise ne gère pas les occultations.

Apprentissage des données du sous-système d'assistance

Le processus d'apprentissage des données nous permet de prédire la pose du point de vue à partir des données de la centrale inertielle et du récepteur GPS. L'erreur relevée entre la pose estimée par

le suivi visuel et celle obtenue par le sous-système d'assistance est modélisée à l'aide d'un processus Gaussien [WILLIAMS 1998]. Il apprend en ligne l'erreur et utilise la covariance calculée pour prédire l'erreur à l'instant courant.

Nous décrirons d'abord son fonctionnement. Soit $(x_1, x_2, ..., x_n)$ un ensemble de données considéré comme ensemble d'apprentissage associé à $(y_1, y_2, ..., y_n)$ telles que $y_i = f(x_i)$. L'objectif est de prédire la valeur y_{n+1} associée à la nouvelle donnée x_{n+1}.

Nous considérons $(Y_1, ..., Y_{n+1})$ un ensemble de $n+1$ variables aléatoires qui suivent une distribution gaussienne de moyenne nulle et de matrice $(n+1) \times (n+1)$ de covariance Σ_{n+1} telle que :

$$\Sigma_{n+1} = \begin{pmatrix} \Sigma_n & \mathcal{K} \\ \mathcal{K}^T & \kappa_{n+1} \end{pmatrix} \tag{4.11}$$

Où Σ_n est une matrice $n \times n$, $\mathcal{K}$ un vecteur ligne de taille n et κ_{n+1} un scalaire. Si $y_1, ..., y_n$ sont les observations associées à $x_1, ..., x_n$, alors la distribution conditionnelle $P(Y_{n+1}|Y_1, ..., Y_n)$ suit une distribution gaussienne telle que :

$$\mu_{Y_{n+1}} = \mathcal{K}^T \Sigma_n^{-1} y^n \tag{4.12}$$

$$\sigma^2_{Y_{n+1}} = \kappa_{n+1} - \mathcal{K}^T \Sigma_n^{-1} \mathcal{K} \tag{4.13}$$

Où $y^n = (y_1, ..., y_n)^T$, $\kappa_{n+1} = cov(y_{n+1}, y_{n+1})$, $\mathcal{K}_i = cov(y_{n+1}, y_i)$ et $\Sigma_{ij} = cov(y_i, y_j)$. La covariance entre y_i et y_j est une fonction de x_i et x_j telle que :

$$cov(x_i, x_j) = cov(x_j, x_i) = \frac{1}{N - |i-j|} \sum_{n=1}^{N-|i-j|} x_n x_{n+|i-j|} \tag{4.14}$$

Dans notre cas, x_i représente les données du récepteur GPS ou les orientations de la centrale inertielle et y_i est l'erreur de localisation que nous souhaitons prédire. Durant la phase de suivi visuel, cette erreur est enregistrée comme donnée d'apprentissage pour le processus Gaussien. Quand le suivi visuel échoue, il prédit les valeurs de l'erreur que nous utilisons pour ajuster l'estimation de la pose du point de vue. De plus nous pouvons calculer la vraisemblance de l'erreur de localisation et y associer un intervalle de confiance.

Conditions de rejet de la pose estimée

Trois critères sont utilisés pour déterminer si nous pouvons avoir confiance dans la pose estimée par le suivi visuel. La violation de l'un de ces critères déclenche l'utilisation du sous-système d'assistance :

1. Le nombre de points suivis. S'il diminue et passe sous un certain seuil, l'estimation de la pose est considérée comme étant médiocre. Ce seuil est fixé entre 10 et 20 points en fonction de l'application ;
2. L'erreur de reprojection. Plus elle est importante, plus l'estimation de la pose est médiocre. Nous l'avons fixée empiriquement à 10 pixels ;
3. L'intervalle de confiance. Les données fournies par le sous-système d'assistance permettent de vérifier la cohérence de la pose estimée par la vision. Si un écart trop important est constaté, elle est considérée comme invalide. En orientation, le seuil est fixé empiriquement à 18 °. En position, il est le triple de l'intervalle de confiance donné sur l'erreur par le processus gaussien.

Systématiquement, nous enregistrons la dernière image sur laquelle la pose est considérée valide. Elle sera réutilisée lors du processus de récupération.

4.1.4.3 Localisation par le sous-système supplétif

Le sous-système supplétif, ou d'assistance, réalise la localisation du point de vue à l'aide des données fournies par le récepteur GPS et la centrale inertielle. Le processus Gaussien utilisé en mode apprentissage lors du suivi visuel pour estimer l'erreur de localisation entre les deux systèmes est à présent utilisé pour prédire cette erreur afin de corriger la pose du point de vue.

Par diverses expérimentations, nous avons déterminé qu'un historique de 5 mesures suffisent pour le processus de prédiction et de correction. Nous avons ensuite relevé les écarts entre les positions prédites à partir du GPS et la pose estimée par le suivi visuel lorsque les deux sont exprimés dans le même repère cartographique. Les résultats, pour deux jeux de données différents, sont restitués à la table 4.2. Les trajectoires de la caméra, du GPS et du GPS corrigé sont tracées pour chaque coordonnées de l'un des jeux de données à la figure 4.7. Elles mettent en évidence une meilleure estimation de la trajectoire GPS lors de la correction par rapport à la trajectoire relevée par le suivi visuel.

		Moyenne (m)	Ecart type
Jeu 1	Axe X	0.9090	0.4572
	Axe Y	1.2710	0.7016
Jeu 2	Axe X	1.0893	0.5524
	Axe Y	1.0025	0.7348

TABLE 4.2 – Erreurs de prédiction

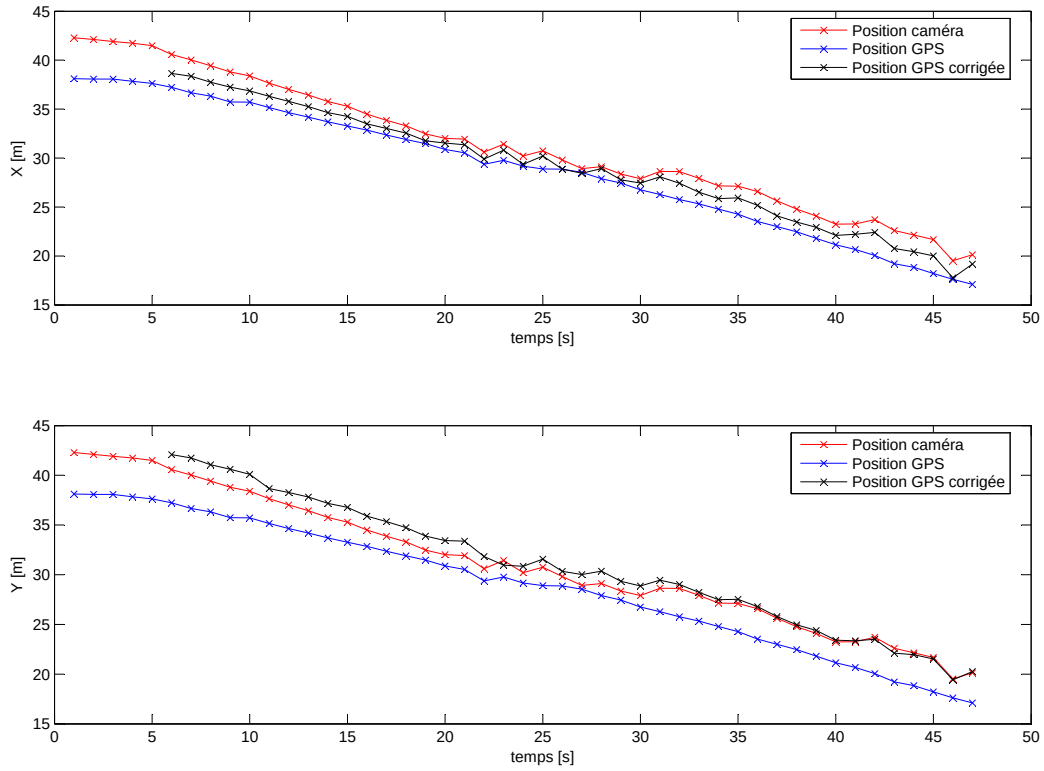


FIGURE 4.7 – Positions caméra, GPS et GPS corrigée obtenues sur un jeu de données.

Au bout d'un certain temps (nous l'avons fixé à 10 images acquises), un processus de récupération et de réinitialisation automatique du suivi visuel est activé.

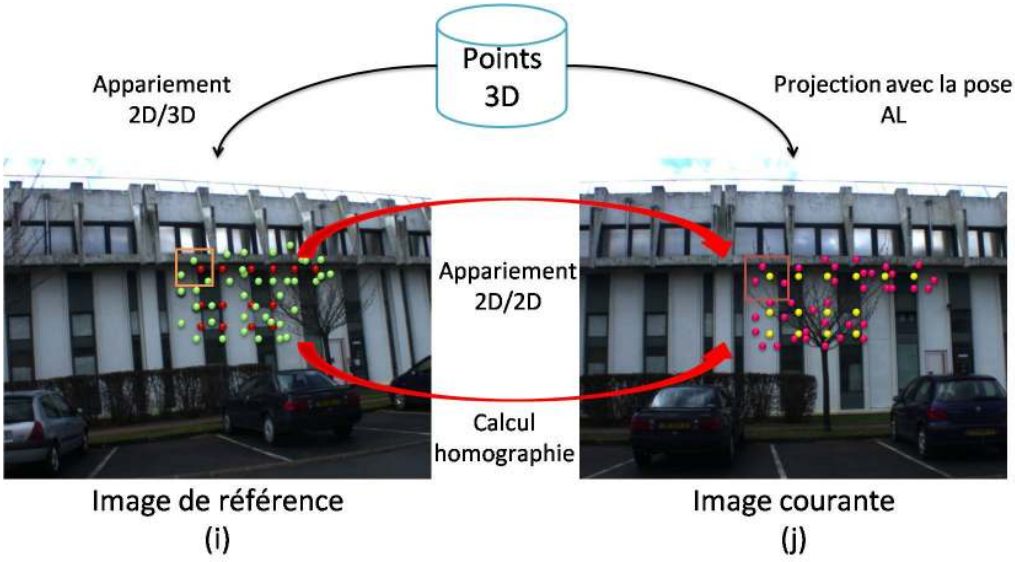


FIGURE 4.8 – Principe de fonctionnement de la réinitialisation de la vision

4.1.4.4 Récupération pour la vision

A l'inverse de l'état d'initialisation semi-automatique, cette procédure de récupération pour la vision réinitialise automatiquement le suivi. Son principe de fonctionnement est illustré à la figure 4.8.

Nous considérons F_i comme étant une image de référence. Il s'agit de la dernière image enregistrée avant l'échec du suivi visuel. A cette dernière sont associées les correspondances 2D-3D. L'idée est de générer de nouvelles correspondances 2D-3D dans l'image courante.

Pour cela, nous commençons par projeter les points 3D sur l'image courante en utilisant la pose estimée par le sous-système d'assistance pour initialiser des zones de recherche rectangulaires centrées sur les points projetés. Nous en extrayons des points d'intérêt SURF que nous mettons en correspondance avec ceux trouvés dans l'image de référence. Pour y parvenir, nous devons identifier la transformation permettant de passer d'un ensemble de points d'intérêt d'une image à une autre. Nous supposons que cette transformation est une homographie car le mouvement entre les deux images est considéré comme faible.

Si nous notons H_{ij} cette homographie, m_i et m_j les points d'intérêt extraits respectivement de l'image de référence F_i et de l'image courante F_j, la relation liant m_i, m_j et H_{ij} s'écrit comme suit :

$$m_j = H_{ij} m_i$$

Une fois l'homographie estimée, nous l'appliquons sur les points 2D associés aux points 3D de l'image de référence afin de les transposer à l'image courante. Cela permet de mettre à jour les correspondances 2D-3D pour l'image courante et donc réinitialiser le processus de suivi visuel.

L'utilisation de l'homographie, qui suppose en principe que les points sont coplanaires, aurait pu être un désavantage. Nous avons donc testé l'algorithme de réinitialisation de la vision sur des points non coplanaires. Le constat est que la méthode est également applicable comme nous pouvons le voir sur les deux séries d'images de test de la figure 4.9 page suivante. Toutefois, la mise en correspondance fournira des résultats approchés.

Pour terminer sur cette partie, nous avons mesuré le temps pris par cette phase de réinitialisation. Nous avons également analysé l'impact du processus de prédiction lorsqu'il est employé pour estimer la pose à partir des informations issues de la centrale inertielle et du récepteur GPS. Les durées mesurées sont consignées à la table 4.3 page ci-contre. La prédiction permet de diminuer la durée de cette réinitialisation.

(a) Points coplanaires (à gauche, image de référence)

(b) Points non-coplanaires (à gauche, image de référence)

FIGURE 4.9 – Mise en correspondance lors de la phase de réinitialisation

Étape (durées en ms)	Sans Prédiction	Avec Prédiction
Extraction vue de référence	15	15
Extraction vue courante	370	240
Appariement	20	10
RANSAC	10	10
Total	415	275

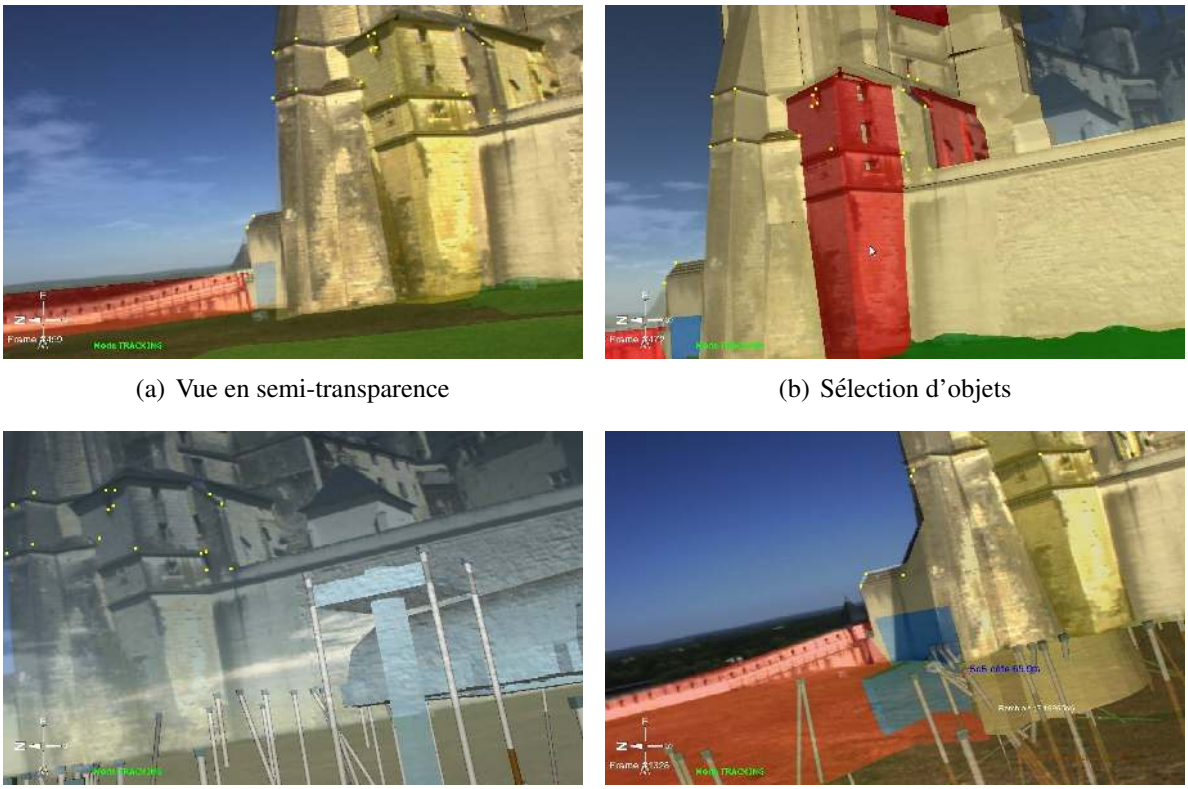
TABLE 4.3 – Comparaison des temps d'exécution de la réinitialisation : avec et sans prédiction

Remarque 4.1 : Temps de réinitialisation

Les temps affichés peuvent sembler long. Toutefois, comparés à d'autres propositions, notamment celle de [REITMAYR et DRUMMOND 2007] dont la durée est de 3 secondes, nous sommes un ordre de grandeur en dessous, ce qui indique un progrès manifeste à l'époque dans ce type d'application.

4.1.5 Bilan

Le sous-système de localisation a été intégré dans l'architecture globale du projet. Sa partie logicielle ayant été par ailleurs présentée dans le chapitre précédent (voir figure 3.13 de la section 3.1.3.4 page 80), nous ne la détaillerons pas davantage. La figure 4.10 page suivante donne quelques captures d'écran du rendu de l'application au cours de son utilisation. Au cours de ce bilan, nous ferons la synthèse des contributions apportées dans ce projet et verrons les liens entretenus avec l'étude des architectures logicielles.



(a) Vue en semi-transparence

(b) Sélection d'objets

(c) Visualisation de structures cachées

(d) Visualisation des sondages

FIGURE 4.10 – Captures de quelques vues en réalité augmentée du projet RAXENV

Synthèse des contributions

Contrairement à l'approche classique utilisée dans le cadre du développement d'un capteur hybride, nous avons décidé, au lieu d'employer un schéma de fusion entre les données, d'utiliser une stratégie de suppléance. Elle est fondée sur le fait que l'estimation de la pose par la vision permet un recalage optimal des entités virtuelles sur les images réelles.

Nous avons proposé dans ce cadre un protocole de calibration complet détermine les relations entre les positions et orientations mesurées par un récepteur GPS et une centrale inertielle d'une part avec les poses estimées par un processus de suivi visuel d'autre part.

Dans le fonctionnement de notre système, une particularité est que nous impliquons l'utilisateur dans le processus d'initialisation pour que le système puisse ensuite se localiser de manière automatique, soit prioritairement par la vision, soit par un sous-système d'assistance. Le passage peut s'effectuer de l'un à l'autre en passant par une procédure de réinitialisation de la vision dont l'exécution est plus rapide que celles proposées jusqu'au moment de ces travaux.

Liens avec les architectures logicielles

RAXENV a servi d'étude de cas pour la première version du *framework* ARCS. En particulier, le modèle de composant et le modèle d'application, composée de feuilles représentant des chemins de données différents ont été exploités. En effet, le fonctionnement du capteur hybride, avec ses 4 états nominaux débouche sur 4 chemins de données différents entre composants que nous avons pu piloter grâce à l'automate à états finis représenté à la figure 4.4 page 139. Ce dernier a naturellement trouvé sa contrepartie dans ARCS.

Les contraintes du projet, en particulier sur les capacités d'intégration de notre *framework* ont pu être prises en compte. Elles ont suscité une question au cours du projet et nous avons pu y répondre

par les mécanismes décrits en section 3.1.3.3 page 77 et leur implémentation.

De plus, l'application pilotant l'acquisition de données provenant de plusieurs capteurs en même temps, nous avons fait appel à des techniques de gestion de la concurrence pour lesquelles nous avons rencontré des problèmes de mise au point. Ceci est à la base des travaux ultérieurs que nous avons présenté en section 3.3.1 page 110 dans lesquels nous avons pu caractériser l'apport des techniques automatiques de gestion de la concurrence par rapport à ce que nous avions alors réalisé à la main.

4.2 Interprétation de la scène

L'interprétation de la scène, ou plus exactement des processus se déroulant dans l'espace de travail revêtent un caractère important dans la mesure où le système peut ainsi fournir une assistance à l'opérateur contextualisée.

Dans cette partie, nous aborderons deux séries de travaux focalisées sur cet objectif. Elles adoptent des points de vue complémentaires. En effet, la première se concentre sur le suivi d'objets manipulés par l'utilisateur en vue d'en faire un système de supervision de tâches de maintenance. La deuxième concerne directement la reconnaissance, et donc l'interprétation, des gestes de l'utilisateur. Nous lui adjoignons une problématique corollaire qui est l'identification des émotions sous-jacentes à un geste expressif. Il s'agit d'un premier pas vers un système de téléopération robotique conscient de l'état émotionnel de l'opérateur afin de pouvoir réagir en conséquence.

4.2.1 Suivi des étapes d'une procédure de maintenance industrielle

L'industrie est un des champs applicatifs historiques de la réalité augmentée. En effet, l'un des systèmes mis au point par David Mizell et Tom Caudell, au moment auquel leur est prêtée la paternité du terme réalité augmentée, visait à assister des opérateurs sur des chaînes d'assemblage pour les harnais de câblage à monter dans les avions de lignes produits par Boeing [CAUDELL et MIZELL 1992]. Plus particulièrement, les gammes de montage/démontage ou les opérations de maintenance faisant appel à ces dernières ont fourni une base au développement de nombreux prototypes de systèmes de réalité augmentée tels que [FEINER et al. 1997 ; RAGHAVAN et al. 1999 ; HALLER et al. 2003 ; AGRAWALA et al. 2003 ; SALONEN et al. 2007] pour n'en citer que quelques uns. La réalité augmentée offre deux fonctionnalités intéressantes pour ce domaine [DIDIER et al. 2005b] :

— elle favorise une mise en contexte des informations plus rapide, afin d'aider les mainteneurs expérimentés aux prises avec un équipement dont la maintenance est peu fréquente ;

— elle permet aux mainteneurs inexpérimentés de voir sur site la séquence d'actions à effectuer sur l'objet de la procédure.

De fait, l'utilisation de la réalité augmentée dans des industries comme l'automobile ou l'aérospatial a déjà démontré son efficacité [TANG et al. 2003 ; REGENBRECHT et al. 2005 ; FITE-GEORGEL 2011] dans des activités telles que la formation, la réparation, la maintenance et l'inspection. De plus les performances des travailleurs, en termes de productivité et d'attention dans les tâches d'assemblage sont accrues avec l'usage de tels systèmes si on les compare à l'emploi de manuels instructions, souvent volumineux [SÄÄSKI et al. 2008].

Toutefois, la plupart des systèmes de réalité augmentée se contentent d'afficher chaque étape de la procédure en boucle [HAKKARAINEN et al. 2008 ; HENDERSON et FEINER 2009] jusqu'à ce que l'utilisateur décide de passer à la suite. Cela peut engendrer une fatigue de l'opérateur, ce dernier ayant à interrompre son activité proprement dite pour gérer le système, le paradoxe étant que l'opérateur se met à aider le système censé l'assister. Quelques dispositifs ont tenté de remédier à ce problème en suivant les actions effectuées par l'opérateur. Ainsi, [WARD et al. 2006] instrumente l'espace de travail en utilisant des microphones et des centrales inertielles afin de détecter des activités telles que l'ouverture d'un tiroir, le resserrement d'un écrou ou l'utilisation d'un tourne-vis. Ces dernières

sont catégorisées en combinant l'application de l'analyse discriminante linéaire (ou LDA pour *Linear Discriminant Analysis*) sur les données sonores et du modèle de Markov caché sur les accélérations. Dans la même veine, [HUIKARI et al. 2010] exploite les données de centrales inertielles attachées aux poignets de l'opérateur pour classifier ses activités. [RADKOWSKI et STRITZKE 2012] exploite les informations retournées par le capteur Kinect de Microsoft pour suivre les mouvements des mains de l'opérateur. Enfin, [BLESER et al. 2015] combine des données visuelles (pour suivre les objets de l'espace de travail) et inertielles (pour suivre l'opérateur) et utilise des méthodes issues de l'apprentissage automatique pour effectuer le suivi des étapes de la procédure de maintenance.

Positionnement 4.2

Nous nous inscrivons dans le cadre de la maintenance industrielle et visons à proposer un système s'appuyant uniquement sur le suivi visuel des pièces faisant partie de l'assemblage à maintenir. L'objectif, *in fine* est d'aller vers un système automatique de certification du respect des étapes de la procédure de maintenance.

Les travaux que nous allons présenter s'inscrivent dans le cadre d'une thèse CIFRE [RUKUBAYIHUNGA 2016] montée en partenariat avec l'entreprise Wassa, spécialisée dans le développement d'applications qui incorporent des solutions du domaine de la vision par ordinateur.

Nous nous baserons, dans le cadre de notre étude sur l'hypothèse 4.1.

Hypothèse 4.1 : Procédure de maintenance

Une procédure de maintenance est constituée en grande partie de gammes de montage et de démontage.

Autrement dit, la détection d'une étape d'une procédure de maintenance passe par la détection de l'instant d'assemblage ou de désassemblage des pièces considérées. Nous allons voir, à partir de deux définitions différentes de la notion d'assemblage, comment nous pouvons détecter ces instants.

4.2.1.1 L'assemblage rigide

Nous démarrons avec une première définition de l'assemblage :

Définition 4.1 : Assemblage restreint

Un assemblage entre deux pièces est caractérisé par le fait qu'il existe une transformation rigide, constante au cours du temps, permettant de passer du repère local d'une pièce à l'autre.

De cette dernière, nous allons voir comment nous pouvons déterminer si deux pièces sont assemblées (c'est à dire liées rigidement ou non). Nous sommes dans le cas où les pièces de l'assemblage sont répertoriées et leurs géométries connues (décrites dans un fichier CAO), ce qui nous permet de nous appuyer sur le suivi basé modèle. Nous allons donc nous intéresser au *pipeline* de traitements utilisés en réalité augmentée dans le cadre du recalage basé modèle tel que l'on peut le représenter à la figure 4.11 page suivante. Ce processus débute avec l'acquisition de l'image qui est ensuite traitée pour extraire des primitives 2D. Ces dernières sont mises en correspondance avec des primitives 3D issues de la connaissance *a priori* du monde réel. Les appariements résultants permettent d'estimer la position et l'orientation de la caméra par rapport à un repère donné (ou de la scène par rapport à la caméra selon la nature de l'algorithme employé). Enfin, pour vérifier la validité de la pose fournie, l'erreur de reprojection, qui consiste à mesurer l'écart entre les primitives 2D et la projection des primitives 3D réalisée à l'aide de la pose de la caméra et de ses paramètres intrinsèques, est calculée.

Il est à noter que l'étape d'estimation de la pose fournit déjà les informations de localisation nécessaire à un recalage du monde virtuel sur le réel ; l'erreur de reprojection consiste en une vérification de ce processus (cette dernière, dans le cas d'algorithmes itératifs de calcul de la pose est calculée

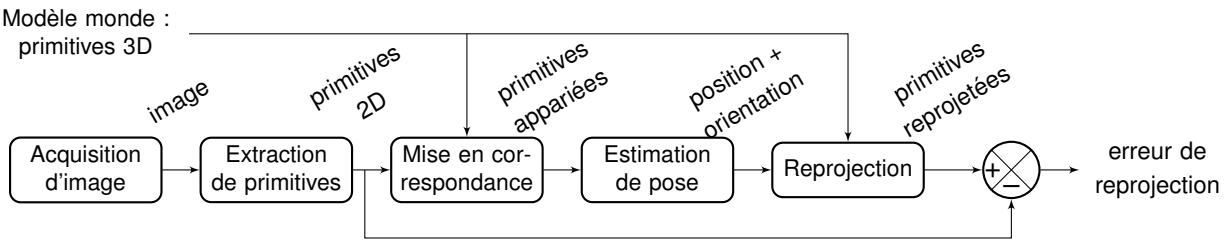


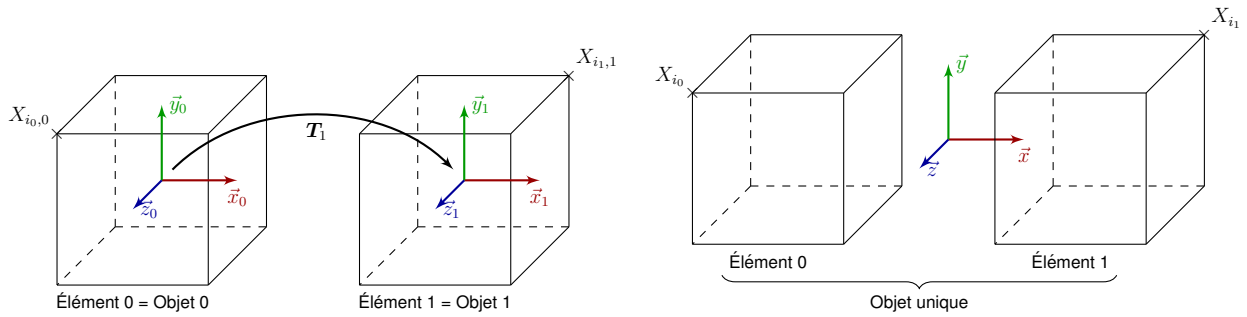
FIGURE 4.11 – *Pipeline* générique d'estimation de la pose et de calcul de l'erreur de reprojection dans les approches basées modèles

à chaque itération et sert de critère de convergence à l'instar de l'algorithme de l'itération orthogonale [LU et al. 2000]). Partant du constat que l'erreur de reprojection est en quelque sorte un sous-produit du *pipeline* de traitement, nous nous sommes demandés dans quelle mesure nous pourrions l'utiliser en tant que métrique pour détecter les instants d'assemblage/désassemblage [RUKUBAYIHUNGA et al. 2015].

De l'erreur de reprojection en tant que métrique

Nous avons commencé par simuler une étape de désassemblage entre deux objets simples pour vérifier notre hypothèse. Dans cette partie, nous calculons l'erreur de reprojection d'un objet selon deux hypothèses différentes illustrées à la figure 4.12 :

1. l'assemblage est composé d'objets indépendants ;
2. l'assemblage est composé de deux objets formant un objet unique.



(a) Configuration : objets indépendants

(b) configuration : objet global unique

FIGURE 4.12 – Hypothèse d'assemblage des modèles 3D.

Dans le cas de pièces indépendantes, les coordonnées de chaque primitive 3D sont exprimées dans le repère local de son objet d'appartenance. Nous supposons qu'il existe une transformation rigide permettant de passer d'un repère à l'autre et nous adoptons les notations suivantes :

— $\boldsymbol{x}_{i,j,t}$ représente les coordonnées 2D de la primitive i extraite de l'image t ;
— $\boldsymbol{X}_{i,j}$ est sa contrepartie 3D exprimée dans le repère local à l'objet j ;
— $\boldsymbol{P}_{j,t}$ est la matrice de projection reliant les primitives 2D et 3D.

La relation entre les deux peut s'exprimer de la manière suivante :

$$\boldsymbol{x}_{i,j,t} \cong \boldsymbol{P}_{j,t}\boldsymbol{X}_{i,j} \tag{4.15}$$

Mais également :

$$\boldsymbol{P}_{j,t} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]_{j,t} \tag{4.16}$$

Où :

— $\boldsymbol{K}$ est la matrice des paramètres intrinsèques de la caméra qui identifie la fonction de projection de cette dernière, permettant de passer du repère local à la caméra au repère image. Cette identification des paramètres peut-être effectuée hors ligne en utilisant une procédure de calibration telle que celle de [ZHANG 1999] ;

— $[\boldsymbol{R}|\boldsymbol{t}]_{j,t}$ est la transformation passant du repère objet au repère local de la caméra.

Nos objets étant assemblés de manière supposée rigide, une matrice de transformation dont les paramètres sont constants au cours du temps peut-être définie. Nous la nommerons $\boldsymbol{T}_j$ et elle désigne la transformation entre l'objet de référence de l'assemblage (objet 0) et l'objet j. Cette dernière peut-être déterminée de deux manières :

— Par extraction d'information des modèles de CAO, si elle y est contenue ;

— Par observation de la scène pendant une certaine durée, en nous basant sur la relation suivante ci-dessous.

$$[\boldsymbol{R}|\boldsymbol{t}]_{j,t} = \boldsymbol{T}_j[\boldsymbol{R}|\boldsymbol{t}]_{0,t}, \forall j \in \{1..n\} \tag{4.17}$$

L'erreur de reprojection e_t de notre modèle sera alors exprimée par la relation :

$$\boldsymbol{e}_t = \sqrt{\frac{1}{m}\sum_l \sum_k \|\boldsymbol{y}_{k,l,t} - \boldsymbol{x}_{k,l,t}\|^2} \tag{4.18}$$

où :

— m désigne le nombre de primitives mises en correspondance,

— l est l'index de l'objet 3D,

— k est l'index de la primitive 3D mise en correspondance,

— $\boldsymbol{x}_{k,l,t}$ désigne les coordonnées de la primitive 2D détectée dans l'image,

— $\boldsymbol{y}_{k,l,t}$ désigne les coordonnées 2D de la primitive 3D reprojetée en combinant les équations 4.15, 4.16 et 4.17, ce qui nous amène à l'équation ci-dessous :

$$\boldsymbol{y}_{k,l,t} \cong \boldsymbol{K}\boldsymbol{T}_l[\boldsymbol{R}|\boldsymbol{t}]_{0,t}\boldsymbol{X}_{k,l} \tag{4.19}$$

Dans le cas d'un assemblage décrit comme un objet unique, les équations décrivant l'erreur de reprojection sont plus simples à écrire. Ainsi, nous notons :

— $\boldsymbol{x}_{i,t}$ les coordonnées dans l'image t de la $i$ème primitive 2D,

— $\boldsymbol{X}_i$ les coordonnées de sa contrepartie 3D,

— m le nombre total de primitives mises en correspondance,

— k l'index de la primitive 3D mise en correspondance,

— $\boldsymbol{y}_{i,t}$ les coordonnées de la projection de $\boldsymbol{X}_i$ calculées comme suit :

$$\boldsymbol{y}_{i,t} \cong \boldsymbol{P}_t\boldsymbol{X}_i \tag{4.20}$$

L'erreur de reprojection est alors calculée de la manière suivante :

$$\boldsymbol{e}_t = \sqrt{\frac{1}{m}\sum_i \|\boldsymbol{y}_{i,t} - \boldsymbol{x}_{i,t}\|^2} \tag{4.21}$$

Dans les deux cas, l'observation que nous devrions réaliser devrait mettre en évidence que la stabilité de l'erreur de reprojection est lié à celle de l'assemblage. En cas de désassemblage, cette erreur devrait croître.

Validation de l'hypothèse

Pour valider l'utilisabilité de l'erreur de reprojection en tant que métrique de l'état d'assemblage des pièces, nous avons calculé cette dernière sur des scènes virtuelles (ce qui nous permet de disposer d'une vérité de terrain), puis sur des scènes réelles comme nous pouvons le voir aux figures 4.13 et 4.14 page suivante. Au niveau du *pipeline* de traitement, pour cette validation préliminaire, nous avons utilisé :

— Des descripteurs SIFT (*Scale-Invariant Feature Transform*) [LOWE 1999] en tant que primitives 2D en raison de leur plus grande robustesse pour la mise en correspondance. Les primitives 3D sont associées à des descripteurs de même nature ;

— La bibliothèque FLANN (*Fast Library from Approximate Nearest Neighbors*) [MUJA et LOWE 2009] pour la mise en correspondance ;

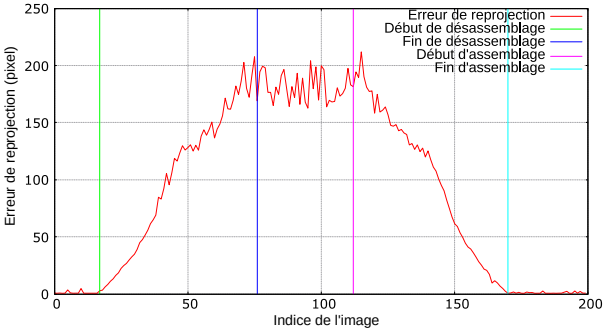
— L'algorithme E-PNP pour l'estimation de la pose [LEPETIT et al. 2009].

Nous avons ensuite effectué divers mouvements canoniques tels que des rotations et des translations autour d'un axe particulier de l'espace pour tester nos hypothèses.

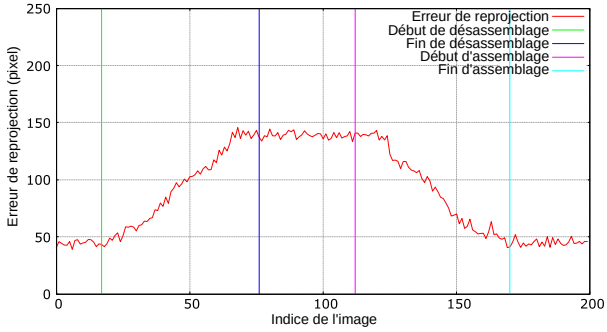
(a) Image 1

(b) Image 50

(c) Image 80



(d) Evolution de l'erreur de reprojection (suivi indépendant)



(e) Evolution de l'erreur de reprojection(suivi global)

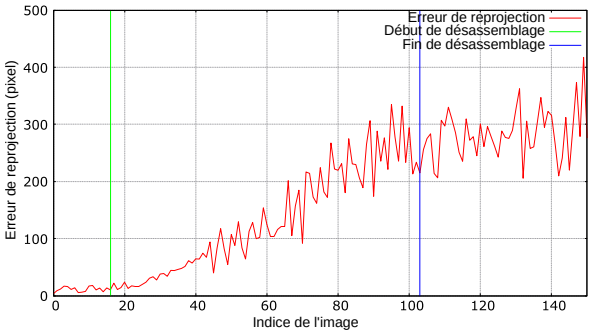
FIGURE 4.13 – Scène virtuelle d'assemblage et de désassemblage de deux objets.

D'après les résultats obtenus (graphiques 4.13(d), 4.13(e), 4.14(d) et 4.14(e) page suivante), l'erreur de reprojection peut-être considérée comme une métrique de détection des passages d'un état assemblé à un état désassemblé. Sans surprise, les scènes virtuelles mettent nettement en avant ce comportement. Ce dernier, moins marqué, est aussi observable sur les scènes réelles. En revanche, l'erreur de reprojection calculée sur la globalité des deux pièces, considérées comme ne constituant qu'un objet, exhibe un comportement moins marqué que l'autre variante considérant les pièces comme indépendantes. En effet, ce mode de calcul a tendance, de part son comportement, à « écraser » le débattement des valeurs possibles pour l'erreur de reprojection.

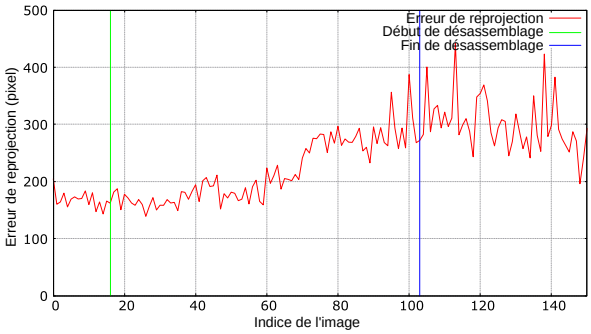
(a) Image 0

(b) Image 60

(c) Image 120



(d) Evolution de l'erreur de reprojection (suivi indépendant)



(e) Evolution de l'erreur de reprojection(suivi global)

FIGURE 4.14 – Désassemblage dans des conditions réelles.

Détection de l'instant de changement d'état

Si les graphiques permettent de vérifier *a posteriori* que l'erreur de reprojection est plus élevée lorsque les objets sont désassemblés, il est plus délicat de déterminer l'instant où cela survient puisque l'analyse ne peut se faire qu'en fonction des données déjà disponibles.

Pour cela, nous avons exploré un certain nombres d'algorithmes permettant de détecter une rupture dans une série temporelle de données. Ces algorithmes, généralement hors ligne, peuvent être ensuite adaptés pour fonctionner en ligne. Les méthodes retenues s'appuient sur des approches utilisant des données statistiques, l'idée étant d'utiliser des tests basés sur deux hypothèses de comportement différentes du signal (chaque hypothèse décrit le signal comme se comportant suivant une loi normale d'écart type et de moyenne définie). En effet, il s'agit d'identifier, dans le cas général, quand le système passe d'un état normal à un état anormal [FILLATRE 2011]. Nous le transposerons à la détection du passage d'un état assemblé à un état désassemblé.

Parmi les algorithmes que nous avons appliqué, nous recensons :

— La carte de contrôle de Shewhart [SHEWHART 1930] fondée sur un calcul de vraisemblance ;

— La carte de contrôle avec moyenne géométrique glissante (ou GMA – *Geometric Mobile Average*), conférant une importance moindre aux données plus anciennes [ROBERTS 1959 ; BASSEVILLE et NIKIFOROV 1993] ;

— La carte de contrôle avec moyenne finie glissante (ou FMA – *Finite Mobile Average*) [BASSEVILLE et NIKIFOROV 1993], différant de la précédente de part l'utilisation d'une fenêtre glissante de taille fixe ;

— L'algorithme CUSUM (*Cumulative Sum*) [WALD 1947 ; PAGE 1954] fondé sur le calcul de la somme cumulées des logarithmes du rapport de vraisemblance et comparé à un seuil adaptatif déterminé en fonction des observations précédentes ;

— GLR (*Generalized Likelihood Ratio*) [LORDEN 1971], variante de CUSUM où la deuxième hypothèse (loi normale) est inconnue et qui procède en recherchant la valeur maximale du rapport de vraisemblance ;

— MLR (*Mixture Likelihood Ratio*) [POLLAK et SIEGMUND 1975], variation de GLR qui se fonde sur une connaissance *a priori* de la distribution statistique correspondant à la première hypothèse.

Ces méthodes remonteront une alarme dès que les valeurs calculées se retrouvent au delà d'un seuil déterminé empiriquement.

Nous y avons également ajouté nos propres méthodes, KSUM et KSUMratio. En effet, les algorithmes précédents sont adaptés à la détection d'une rupture franche et sont moins efficaces lorsque la variation du signal est graduelle. De plus, la plupart s'appuient sur une connaissance *a priori* du signal, or le nôtre dépend du mouvement effectué tant au niveau du type de déplacement que de la vitesse d'exécution de l'opérateur. Nous nous proposons d'appliquer un noyau gaussien sur les données de manière à les filtrer afin de les rendre exploitables. Dans le cas de KSUM, le filtrage de la valeur est effectué comme suit :

$$g_k = K(\sum_{i=1}^{k} y_i) \tag{4.22}$$

avec le noyau gaussien $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$.

La deuxième formulation consiste à évaluer le rapport entre le KSUM de deux mesures successives. Nous l'appellons KSUMratio et le formulons de la manière suivante :

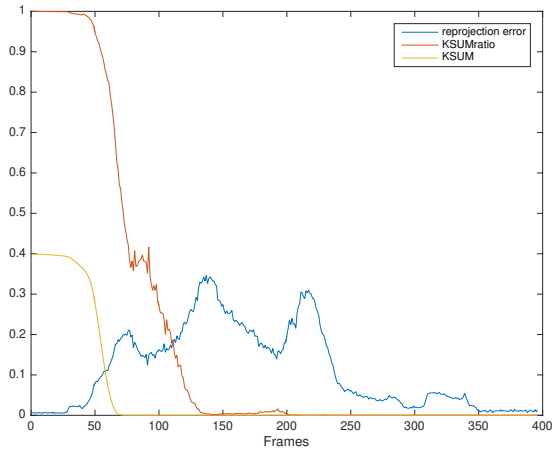
$$g_k = \frac{K(\sum_{i=1}^{k} y_i)}{K(\sum_{i=1}^{k-1} y_i)} \tag{4.23}$$

Dans les deux cas, l'alarme est déclenchée lorsque g_k franchit un seuil h fixé empiriquement. Les premiers tests effectués avec KSUM et KSUMratio ont été concluants. En effet, nous observons un comportement similaire quelque soit le mouvement effectué comme nous pouvons le constater à la figure 4.15 page suivante. KSUM, retourne une valeur proche de 0.4 dans le cas où les pièces sont assemblées et proche de 0 le désassemblage réalisé. Pour KSUMratio, ces valeurs seront respectivement proches de 1 et de 0.

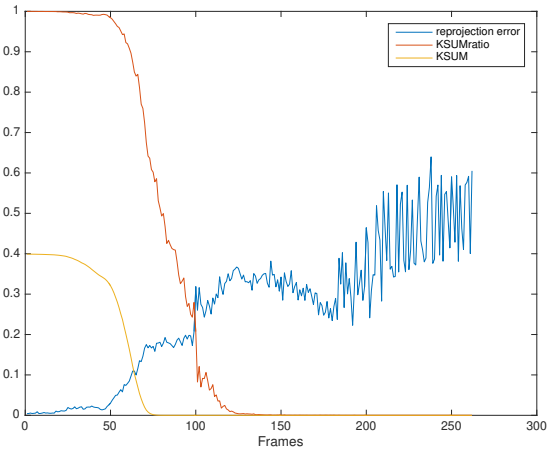
Expérimentation et résultats

Nous avons appliqué les différents algorithmes cités précédemment en faisant varier quelques paramètres tels que la taille de la fenêtre glissante ou la manière dont l'effet mémoire de ces algorithmes s'applique. Les résultats obtenus sont représentés à la table 4.4 page 158. Nous avons testé les algorithmes sur 17 types de mouvements différents tels que des rotations, des translations et vissages pour lesquels nous avons fait varier les directions principales dans l'espace. Nous avons indiqué, dans les critères de comparaison, le délai de détection moyen en nombre d'images et en nombre de secondes pour chacun des algorithmes (ce nombre peut-être négatif si l'algorithme détecte le désassemblage avant que celui-ci ait réellement lieu). Nous y avons aussi recensé le temps d'exécution moyen des algorithmes testés. Il en ressort que CUSUM, l'une des méthodes les plus populaires de détection de rupture est le plus pertinent en termes de délais de détection. KSUMratio présente, de son côté, un excellent compromis entre temps d'exécution et délai de détection.

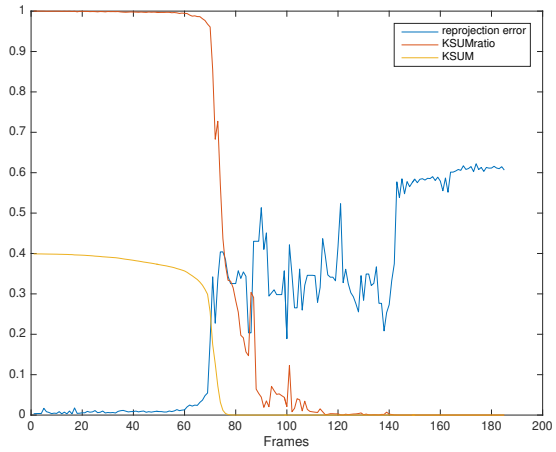
Nous avons donc ainsi démontré que l'erreur de reprojection pouvait être employée pour détecter l'instant où le système observé passe d'un état assemblé à un état désassemblé et inversement. Pour détecter cet instant, nous avons comparé les performances respectives d'algorithmes d'analyse statistique inspirés des cartes de contrôle. Nous avons également proposé nos propres algorithmes, KSUM et KSUMratio. Ce dernier offre un bon compromis entre délai de détection de l'instant de changement d'état et temps de calcul. Cela en fait donc un candidat souhaitable pour une utilisation dans un système de réalité augmentée.



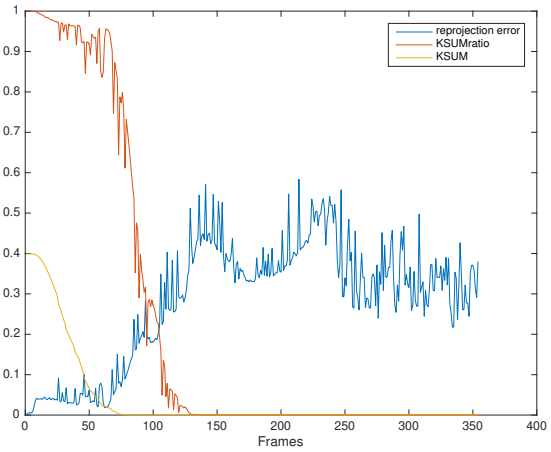
(a) rotation espace x



(b) rotation espace y01



(c) translation espace x



(d) vissage espace x

FIGURE 4.15 – KSUM et KSUMratio appliqués sur l'erreur de reprojection enregistrée suivant différents mouvements de pièces.

Cette première série de contribution s'appuie cependant sur une définition restreinte de l'assemblage. Nous allons, dans la partie suivante, indiquer comment nous pouvons généraliser cette définition afin de détecter des assemblages plus complexes.

4.2.1.2 L'assemblage en tant que liaison mécanique entre deux pièces

Nous nous proposons de définir à présent l'assemblage de la manière suivante :

Définition 4.2 : Assemblage généralisé

Un assemblage entre deux pièces est caractérisé par le fait qu'il existe une liaison mécanique entre elles dont le nombre de degrés de liberté est constant dans le temps.

Il est à noter que cette définition englobe la précédente. En effet, la liaison encastrement, qui fait partie des liaisons mécaniques répertoriées entre deux pièces, est conforme à notre définition restreinte de l'assemblage.

Dans ce cadre, nous supposons que le modèle CAO du mécanisme nous fournit les informations concernant les liaisons mécaniques entre ses différentes parties. L'objectif est alors de déterminer

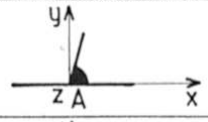
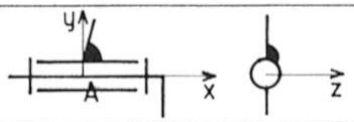
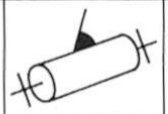
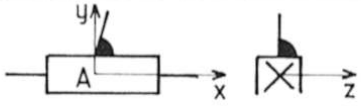
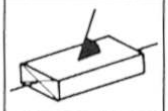
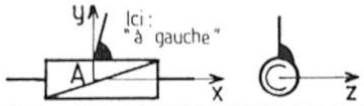
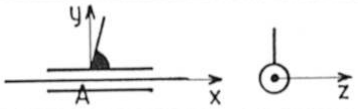
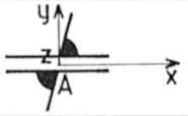
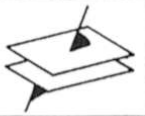
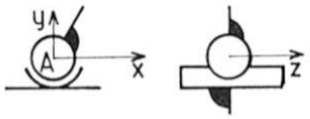
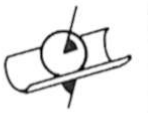
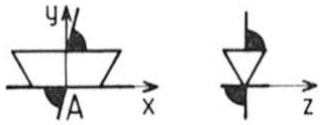
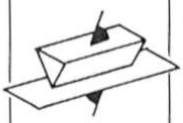
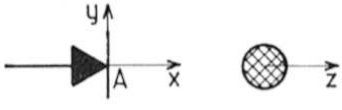
NOM de la liaison	SCHÉMATISATION longitudinalement · en bout	Perspective	Mouv^ts relatifs	TORSEUR (simplifié)	n_s
Encastrement (liaison fixe)			0	$\begin{Bmatrix} X & L_A \\ Y & M_A \\ Z & N_A \end{Bmatrix}$	6
Pivot			R_x	$\begin{Bmatrix} X & 0 \\ Y & M_A \\ Z & N_A \end{Bmatrix}$	5
Glissière			T_x	$\begin{Bmatrix} 0 & L_A \\ Y & M_A \\ Z & N_A \end{Bmatrix}$	5
Hélicoïdale			T_x R_x (liés)	$\begin{Bmatrix} X\text{ (liés)} & L_A \\ Y & M_A \\ Z & N_A \end{Bmatrix}$	5
Pivot glissant			T_x R_x	$\begin{Bmatrix} 0 & 0 \\ Y & M_A \\ Z & N_A \end{Bmatrix}$	4
Sphérique à doigt			R_y R_z	$\begin{Bmatrix} X & L_A \\ Y & 0 \\ Z & 0 \end{Bmatrix}$	4
Rotule (sphérique)			R_x R_y R_z	$\begin{Bmatrix} X & 0 \\ Y & 0 \\ Z & 0 \end{Bmatrix}$	3
Appui plan			T_x T_z R_y	$\begin{Bmatrix} 0 & L_A \\ Y & 0 \\ 0 & N_A \end{Bmatrix}$	3
Linéaire annulaire (Sph-Cyl)			T_z R_x R_y R_z	$\begin{Bmatrix} X & 0 \\ Y & 0 \\ 0 & 0 \end{Bmatrix}$	2
Linéaire rectiligne (Arête-Plan)			T_x T_z R_x R_y	$\begin{Bmatrix} 0 & 0 \\ Y & 0 \\ 0 & N_A \end{Bmatrix}$	2
Ponctuelle (Sphère-Plan)			T_y T_z R_x R_y R_z	$\begin{Bmatrix} X & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}$	1

FIGURE 4.16 – Liaisons mécaniques répertoriées [TINEL et DARDY 1995]

Algorithme	Délai (en nb frames)		Délai (en s)		Temps d'exécution (en 10^{-4}s)
	moyenne	écart-type	moyenne	écart type	
Shewart1	3.00	29.59	0.125	1.23	8.5
Shewart2	1.35	31.30	0.056	1.30	13.3
Shewart3	16.59	57.32	0.691	2.38	12.9
GMA1	6.06	24.99	0.252	1.04	26.8
GMA2	5.12	30.37	0.213	1.27	32.6
GMA3	77.88	105.55	3.245	4.40	30.8
FMA1	-34.47	12.07	-1.436	0.50	16.0
FMA2	-36.59	12.00	-1.524	0.50	22.6
CUSUM1	17.24	21.66	0.718	0.90	16.6
CUSUM2	19.88	16.72	0.828	0.70	19.9
CUSUM3	1.59	21.34	**0.066**	**0.89**	18.9
GLR1	113.94	16.01	4.748	0.67	7977.9
GLR2	18.35	25.86	0.765	1.08	8033.2
GLR3	30.53	20.97	1.270	0.87	7719.5
MLR1	23.82	52.63	0.992	2.19	8312.5
MLR2	28.82	63.60	1.201	2.65	8359.0
MLR3	32.53	58.63	1.355	2.44	7721.2
KSUM	-9.41	26.40	-0.392	1.10	**6.2**
KSUMratio	0.29	26.49	0.012	1.10	6.4

TABLE 4.4 – Délai de détection par rapport à l'instant de changement effectif.

si les mouvements relatifs observés entre deux pièces sont conformes aux degrés de liberté de la liaison. Si des degrés de liberté supplémentaires apparaissent lors de l'analyse, alors nous pouvons considérer que le mécanisme passe à un état désassemblé. Inversement, si des degrés de liberté surnuméraires disparaissent pendant un mouvement relatif entre deux pièces, le mécanisme passe d'un état désassemblé à un état assemblé. Les différents types de liaisons mécaniques étant répertoriées et documentées (voir figure 4.16 page précédente), nous connaissons pour chacune ses degrés de liberté.

Nous nous sommes donc attachés à examiner la variation des paramètres des mouvements. Nous avons exprimé la matrice de transformation permettant de passer du repère local à l'objet de référence au repère local de l'objet en mouvement, l'avons transcrite sous la forme d'une rotation au format axe-angle et d'un vecteur de translation. Nous pouvons voir le résultat du tracé de ces paramètres pour un exemple de mouvement (selon une liaison glissière n'autorisant qu'une translation suivant l'axe des X) sur des jeux de données synthétiques et réelles (figure 4.17 page ci-contre).

Nous remarquons que ces données sont très bruitées. Il en résulte qu'il est délicat de déterminer quelle liaison mécanique est formée par deux pièces assemblées sans filtrer ces signaux. Cela s'explique par le bruit intrinsèque associé à l'acquisition des images mais également par le fait que nous opérons par différenciation des transformations, ce qui a pour effet d'amplifier le bruit.

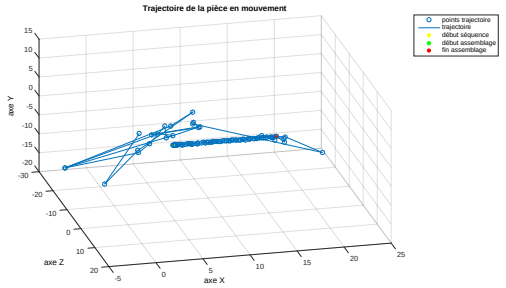
Filtrer les données de transformation

De nombreuses méthodes de réduction du bruit existent dans la littérature telles que la transformation en *wavelet* [XU et al. 1994 ; PIZURICA et al. 2003] ou de nombreux autres filtres (filtre passe-bas, Savitzky-Golay, Tchebychev, Butterworth, Wiener, ...) [SAVITZKY et GOLAY 1964 ; CHEN et al. 2006]. A cela, nous pouvons ajouter les filtres medians ou gaussiens. Toutefois, la plupart introduisent du délai dans le traitement des données ou, en cas forte perturbation, la répercute sur les données suivantes à l'instar du filtre de Kalman qui nécessite, par ailleurs, une connaissance *a priori* de l'évolution du mouvement, ce qui est précisément ce que nous cherchons à déterminer.

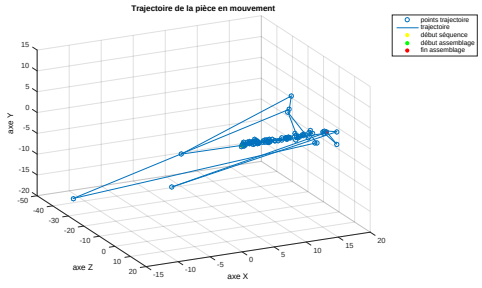
Nous avons donc proposé dans [RUKUBAYIHUNGA et al. 2016a] une méthode pour réduire le bruit en nous appuyant sur un score de confiance calculé en temps réel par une machine à vecteurs de support (*Support vector machine – SVM*) sur des caractéristiques particulières du signal. Les paramètres d'entrée pour effectuer notre apprentissage avec le SVM sont :

— le nombre d'*inliers* de chaque objet détecté ;

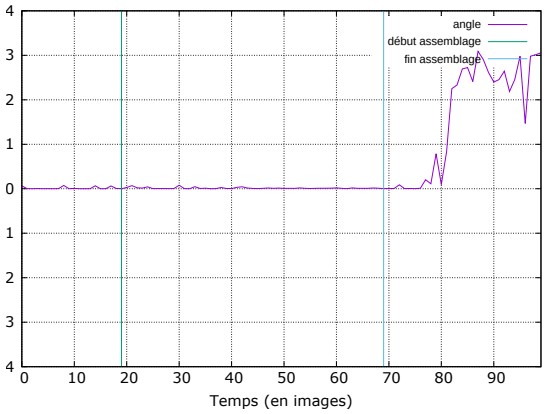
— la qualité moyenne des correspondances de chaque objet.



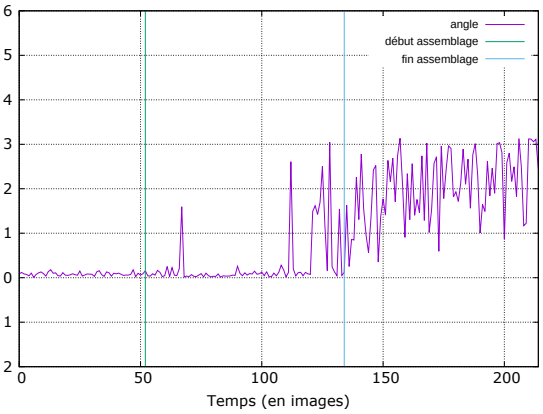
(a) Trajectoire obtenue (synthétique)



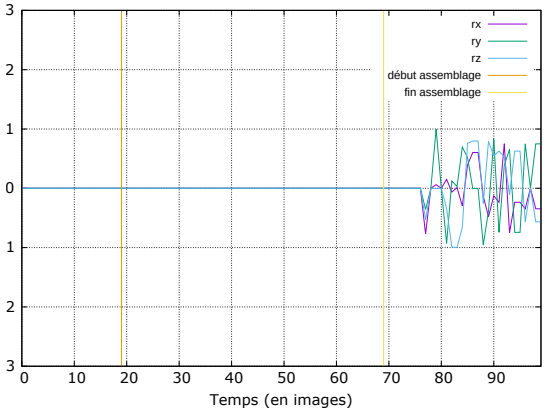
(b) Trajectoire obtenue (réel)



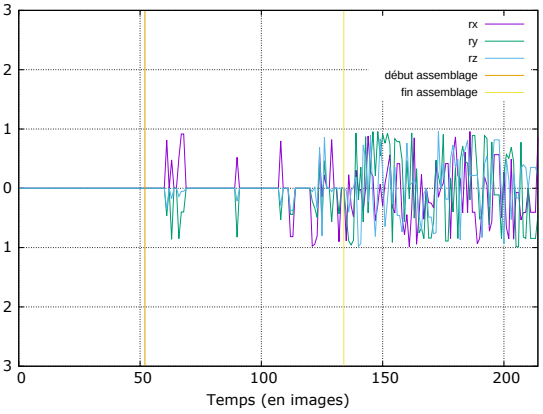
(c) Angle de rotation (synthétique)



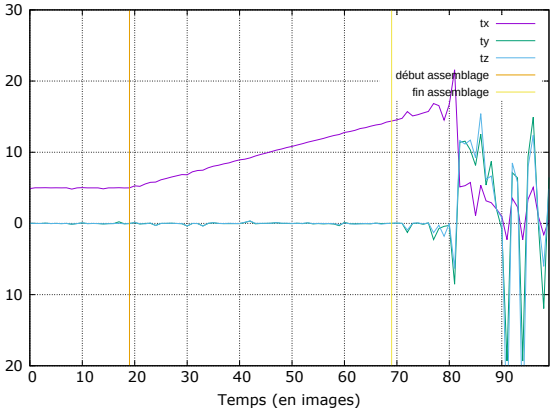
(d) Angle de rotation (réel)



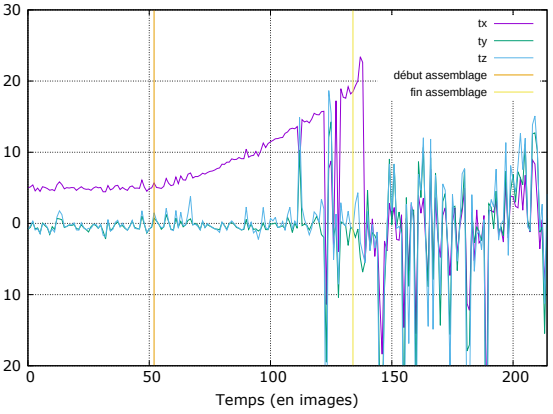
(e) Evolution du vecteur de rotation (synthétique)



(f) Evolution du vecteur de rotation (réel)



(g) Evolution de la translation (synthétique)



(h) Evolution de la translation (synthétique)

FIGURE 4.17 – Evolution des paramètres de mouvement lors du désassemblage de deux pièces suivant une liaison glissière le long de l'axe X.

L'étiquetage des données est réalisé en fonction de la valeur à chaque instant de l'erreur entre la matrice de transformation théorique et celle expérimentale (ce que nous avons pu faire sur les vidéos synthétiques pour lesquelles nous connaissons la vérité de terrain). Deux étiquettes sont générées :

— 0 indique que les matrices de transformation sont similaires ;

— 1 signifie une forte dissimilarité.

La similarité entre deux transformations est mesurée par la trace (ou norme de Frobenius) du produit de la matrice obtenue expérimentalement avec l'inverse de la matrice théorique. Il en résulte la fonction d'étiquetage ci-dessous :

$$\boldsymbol{Tag}_t = \begin{cases} 0 \text{ si } e_t = \|T_{exp} T_{théorique}^{-1}\|_{frobenius} \leq \varepsilon \\ 1 \text{ sinon} \end{cases} \tag{4.24}$$

Empiriquement, nous avons fixé le seuil ε à 2.5.

Le modèle SVM avec un noyau gaussien [VAPNIK 1998] a été entraîné sur les données sélectionnées (après normalisation). Cela nous a permis d'obtenir un hyperplan de séparation avec une précision de $86,4\%$. Le modèle nous permet ensuite d'obtenir un score de confiance de nos données réelles, c'est à dire la probabilité que nos données appartiennent à l'un ou l'autre des ensemble.

Notre correction des données s'appuie alors sur le score de confiance obtenu. Nous supposerons que, dans une séquence vidéo, les objets observés ont un faible déplacement d'une image sur l'autre, ce qui se traduit par une faible variation de l'angle de rotation et du vecteur de translation. Notre mouvement est caractérisé par les paramètres suivants :

— α, angle de rotation ;

— $\vec{\omega}$, vecteur support de l'axe de rotation ;

— $\vec{t}$, vecteur de translation.

Nous appliquons la formule suivante pour réduire le bruit sur le paramètre angulaire et donc estimer sa valeur :

$$\hat{\alpha}_{t|g=1} = \begin{cases} \alpha_{t-1|g=0} + (1-c)(\alpha_{t|g=1} - \alpha_{t-1|g=0}) \\ \hat{\alpha}_{t-1|g=1} + (1-c)(\alpha_{t|g=1} - \hat{\alpha}_{t-1|g=1}) \end{cases} \tag{4.25}$$

g est ici l'étiquette prédite par le modèle SVM, c est le score de confiance qui y est associé à l'instant t, α_t la valeur du paramètre angulaire et $\hat{\alpha}_t$ son estimation. En cas de faible score de confiance ou de variation importante de la mesure, nous appliquons la formule suivante :

$$\hat{\alpha}_{t|g=0} = \alpha_{t|g=0} - k(1-c)(\alpha_{t|g=0} - \hat{\alpha}_{t-1}) \tag{4.26}$$

k est un coéfficient déterminé empiriquement. Chaque composante du vecteur translation subit le même traitement.

Le cas du vecteur rotation est différent, ce dernier n'étant pas pertinent si le mouvement relatif est très réduit. Cela se traduira par les équations suivantes si nous avons une faible confiance dans la matrice de transformation relative :

$$\hat{\vec{\boldsymbol{\omega}}}_{t|g=0} = \begin{cases} 0 \text{ si } \hat{\alpha}_{t|g=0} \leq \varepsilon_\alpha \\ \vec{\boldsymbol{\omega}}_{t|g=0} \text{ si } \hat{\alpha}_{t|g=0} \geq \varepsilon_\alpha \text{ et } c \geq \varepsilon_{confiance} \\ \hat{\vec{\boldsymbol{\omega}}}_{t-1} \text{ sinon} \end{cases} \tag{4.27}$$

Lorsque la matrice sera correcte, l'estimation sera la suivante :

$$\hat{\vec{\boldsymbol{\omega}}}_{t|g=1} = \begin{cases} 0 \text{ si } \hat{\alpha}_{t|g=1} \leq \varepsilon_\alpha \\ \hat{\vec{\boldsymbol{\omega}}}_{t-1} \text{ sinon} \end{cases} \tag{4.28}$$

Où ε_α et $\varepsilon_{confiance}$ sont respectivement des seuils angulaire et de score de confiance.

Nous avons ensuite appliqué ce système de filtrage sur des données réelles. La figure 4.18 page suivante indique les résultats obtenus lorsque nous avons procédé au mouvement de pièces assemblées selon une liaison pivot. Ils indiquent que la méthode utilisée est efficace pour stabiliser l'estimation des valeurs des paramètres du mouvement.

Reconnaître les liaisons mécaniques

Chaque liaison mécanique est caractérisée par un torseur cinématique comme nous pouvons le constater à la figure 4.16 page 157. L'écriture des matrices de transformations pour les identifier ultérieurement à ces torseurs doivent se faire à partir du ou des points d'application de la liaison mécanique. Ainsi, chaque paramètre du mouvement relatif doit être exprimé dans le repère local lié à la liaison. Si nous considérons que le repère local d'un des objets (nommé objet 0) est aussi celui du repère monde, la transformation relative permettant de passer de l'objet 0 à tout autre objet i à un instant t (matrice $\boldsymbol{T}_{0\to i_t}$) peut s'écrire en incluant un passage par le repère local à la liaison :

$$\boldsymbol{T}_{0\to i_t} = \boldsymbol{T}_{L\to i_t}\boldsymbol{T}_{0\to L} \tag{4.29}$$

L'expression du torseur cinématique, écrit dans le repère local à la liaison (R_L) est de la forme :

$$\mathcal{V}_{O_L\in R_L} = \begin{cases} \vec{\boldsymbol{\Omega}}_{R_L} \\ \vec{\boldsymbol{V}}_{O_L\in R_L} \end{cases} \tag{4.30}$$

Où $\vec{\boldsymbol{\Omega}}_{R_L}$ et $\vec{\boldsymbol{V}}_{O_L\in R_L}$ représentent respectivement les vitesses angulaire et linéaire de notre objet dans le repère de la liaison. Le torseur cinématique transposé au point d'application à l'origine du repère local de l'objet i ($O_{i,t}$) est exprimé par :

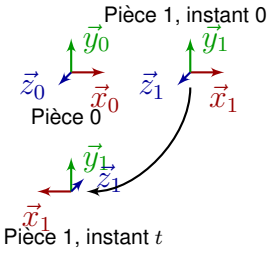
$$\mathcal{V}_{O_{i,t}\in R_L} = \begin{cases} \vec{\boldsymbol{\Omega}}_{R_L} \\ \vec{\boldsymbol{V}}_{O_L\in R_L} + \overrightarrow{\boldsymbol{O}_{i,t}\boldsymbol{O}}_L \wedge \vec{\boldsymbol{\Omega}}_{R_L} \end{cases} \tag{4.31}$$

Les valeurs du torseur cinématique peuvent être obtenues par différenciation sur la matrice $\boldsymbol{T}_{L\to i_t}$, lorsqu'elle est décomposée en vecteur de rotation $\vec{\omega}_{i,t}$ et de translation $\vec{\boldsymbol{t}}_{i,t}$. Cela donne alors, exprimé dans le repère de la liaison :

$$\begin{cases} \vec{\boldsymbol{\Omega}}_{R_L} = \dfrac{\vec{\omega}_{i,t} - \vec{\omega}_{i,t_0}}{t - t_0} \\ \\ \vec{\boldsymbol{V}}_{O_{i,t}\in R_L} = \dfrac{\vec{\boldsymbol{t}}_{i,t} - \vec{\boldsymbol{t}}_{i,t_0}}{t - t_0} \end{cases} \tag{4.32}$$

Où t_0 représente l'instant de détection du début du mouvement déterminé à l'aide de l'algorithme KSUM présenté précédemment. Nous déterminons alors le nombre de degré de liberté du mouvement observé en analysant le torseur cinématique associé au point d'application $O_{i,t}$, centre du repère local à l'objet i.

Pour cela, chaque composante des vecteurs du torseur est comparée à un seuil déterminé empiriquement. Afin de ne pas être à la merci d'une anomalie statistique (*outlier*), nous contrôlons également que le nombre de degrés de liberté détecté persiste au cours du temps. Nous avons donc rejoué les valeurs filtrées présentées à la figure 4.18 page suivante pour identifier les degrés de liberté associés au mouvement et, par ce biais, la liaison mécanique entre les deux objets comme nous pouvons le voir à la figure 4.20 page 164. Les seuils retenus sont 0.01 rad/s pour la vitesse angulaire et 0.2 m/s pour la vitesse linéaire. La persistence se compte en images successives, ici 5. Les figures 4.20(c) et 4.20(d) page 164 donnent respectivement le décompte des degrés de libertés trouvés par analyse au cours du temps et les périodes au cours du temps pendant lesquelles la liaison pivot entre les deux objets est identifiée en tant que tel (en vert).

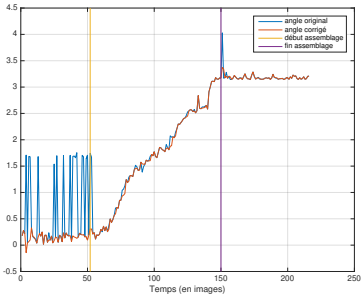


(a) Schéma du mouvement

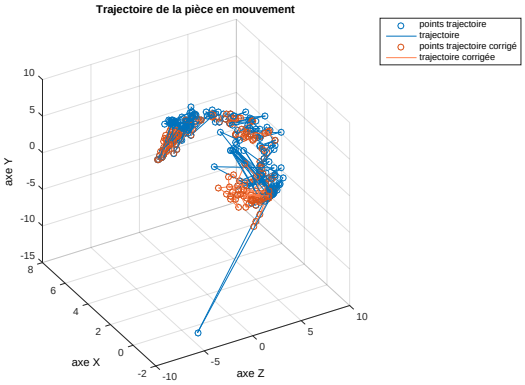
(b) Image 0

(c) Image 82

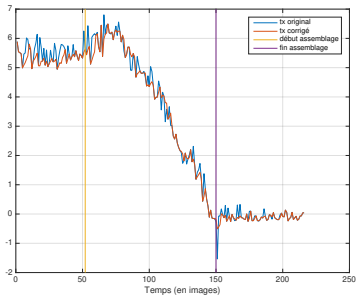
(d) Image 190



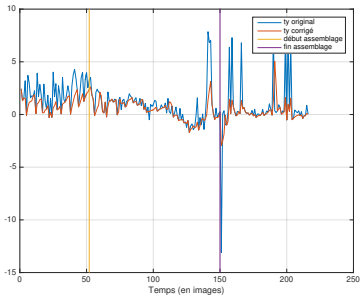
(e) Correction de l'angle de rotation



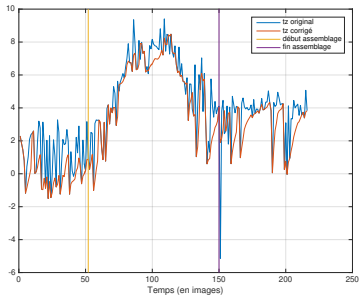
(f) Correction de la trajectoire



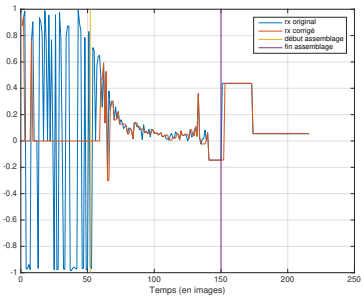
(g) translation axe X



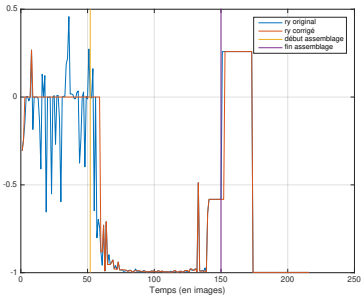
(h) translation axe Y



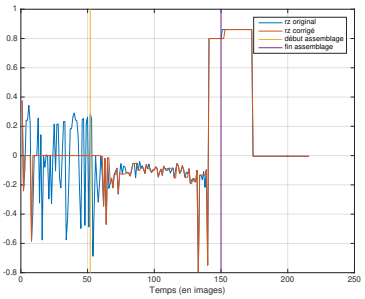
(i) translation axe Z



(j) rotation axe X



(k) rotation axe Y



(l) rotation axe Z

FIGURE 4.18 – Assemblage de deux pièces par une liaison pivot dans des conditions réelles (pièces non posées sur une surface plane).

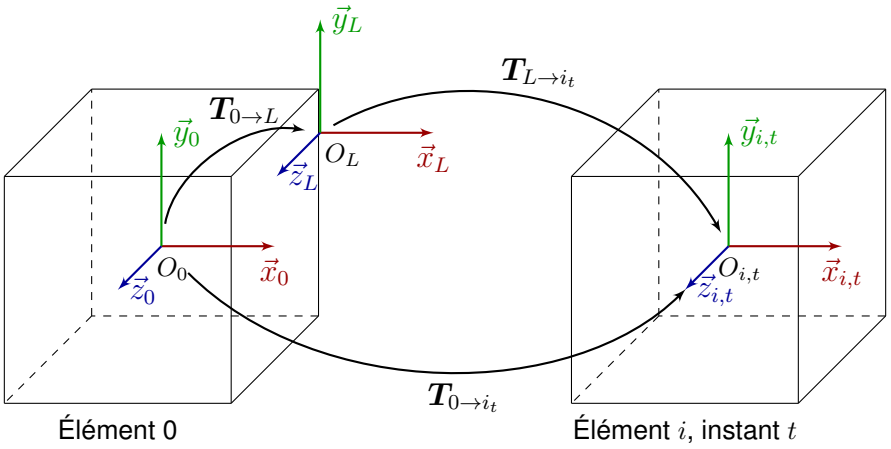


FIGURE 4.19 – Changement de repères pour la reconnaissance des liaisons entre les pièces (cas d'une liaison pivot d'axe une des arêtes du cube).

Discussion et perspectives

La méthode présentée identifie des liaisons simples (pivot et glissière) au cours du mouvement relatif entre les deux pièces. Nous avons montré qu'en nous basant sur l'analyse du mouvement, nous pouvons déduire les degrés de liberté de ce dernier puis reconnaître la liaison mécanique. L'une des étapes ultérieures est de rendre cette méthode plus générique afin de reconnaître n'importe quel type de liaison, que les pièces soient en mouvement relatif les unes par rapport aux autres ou non.

De plus, cette preuve de concept est pour l'instant associée à un *pipeline* de réalité augmenté dans lequel les méthodes choisies pour suivre le mouvement des pièces est à adapter à un contexte industriel dans lequel les pièces ne sont pas aussi texturées que dans notre banc de test.

Une fois la chaîne de détection complètement mise au point, nous pourrons l'utiliser pour effectuer un suivi de scénario de maintenance industrielle.

Quelques mots sur l'implémentation

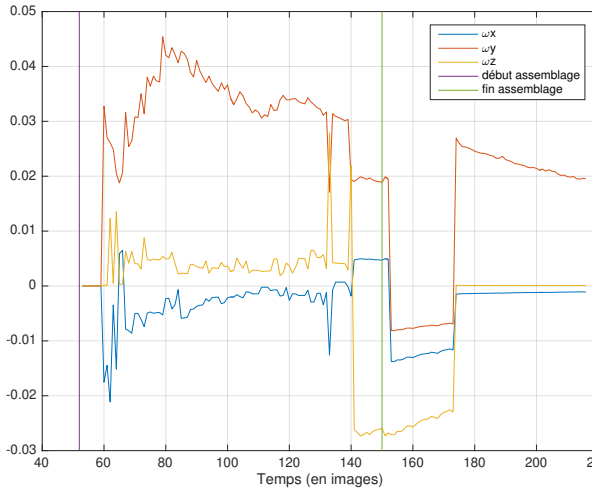
Les concepts présentés ici ont été ensuite implémentés en utilisant la variante javascript du *framework* ARCS. En particulier, la première partie utilisant l'erreur de reprojection (section 4.2.1.1 page 150) a été traduite sous la forme de composants suivant le *pipeline* de calcul de la pose en réalité augmentée (figure 4.21 page 165). Le tracé des valeurs des différents paramètres des mouvements obtenus directement dans le navigateur comme en témoigne la figure 4.22 page 165.

4.2.1.3 Bilan

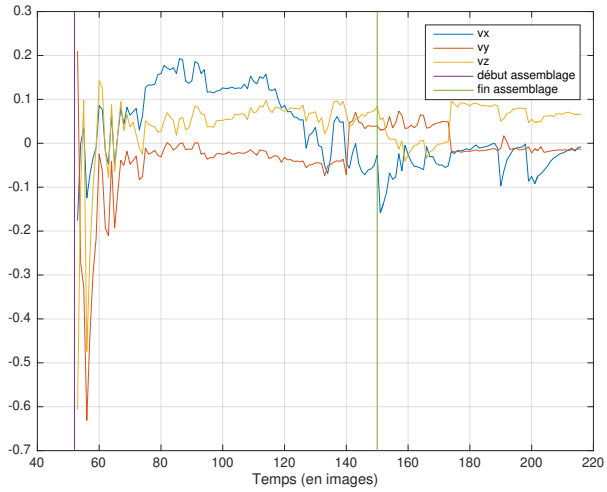
Dans cette partie, nous avons travaillé sur la détection des instants d'assemblage et de désassemblage afin de suivre les étapes des gammes de montage/démontage qui composent une procédure de maintenance.

Nous avons considéré deux définitions de l'assemblage. La première implique que les objets sont liés par une transformation rigide. Nous avons montré dans ce contexte que nous pouvions exploiter l'erreur de reprojection, calculée en temps normal par le *pipeline* de calcul de la pose, pour détecter ces instants de rupture. Pour y parvenir, nous avons introduit nos algorithmes KSUM et KSUMratio qui présentent un compromis intéressant entre retard à la détection et temps de calcul par rapport aux algorithmes de la littérature.

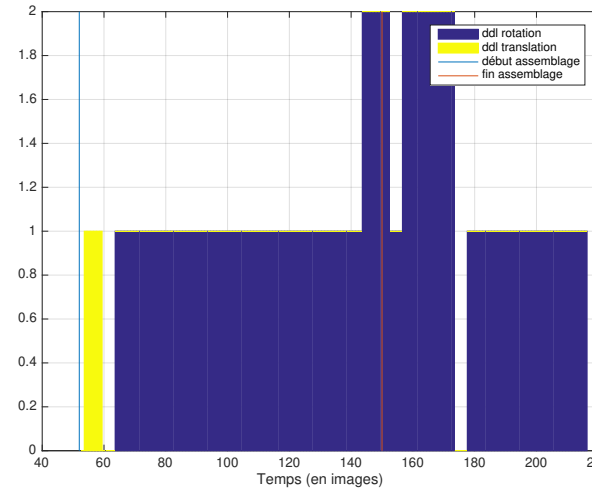
La deuxième considère que les objets sont assemblés par une liaison mécanique. Nous nous



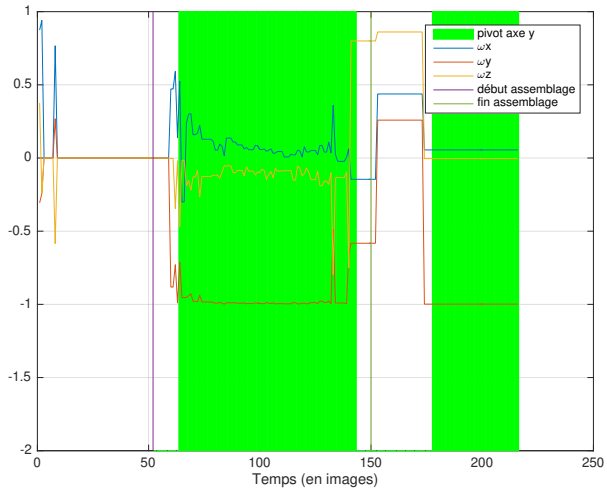
(a) vitesse angulaire



(b) vitesse linéaire



(c) nombre de degrés de liberté



(d) identification liaison

FIGURE 4.20 – Identification d'une liaison pivot d'axe Y par le nombre de degrés de liberté.

sommes donc attelés à reconnaître ces dernières par analyse des mouvements relatifs des pièces dans les images. Une étape nécessaire à cette reconnaissance a été de filtrer les données, ce que nous avons réalisé à l'aide d'un modèle SVM qui nous indique quand les données extraites sont dignes de confiance. Ensuite, par analyse du mouvement, nous sommes en mesure de déterminer ses degrés de liberté et donc d'identifier la laison mécanique entre les deux pièces.

Les perspectives consistent à affiner cette première preuve de concept (implémentée dans la variante javascript du *framework* ARCS) afin de pouvoir l'exploiter dans un véritable scénario industriel.

Dans ces travaux, nous nous sommes intéressé aux mouvements relatifs des pièces pour interpréter et reconnaître le processus qui se déroule dans l'espace de travail du système de réalité augmentée. Nous allons maintenant voir comment nous pouvons travailler sur les gestes de l'opérateur afin de les interpréter, étape nécessaire pour pouvoir interagir, par geste avec le système.

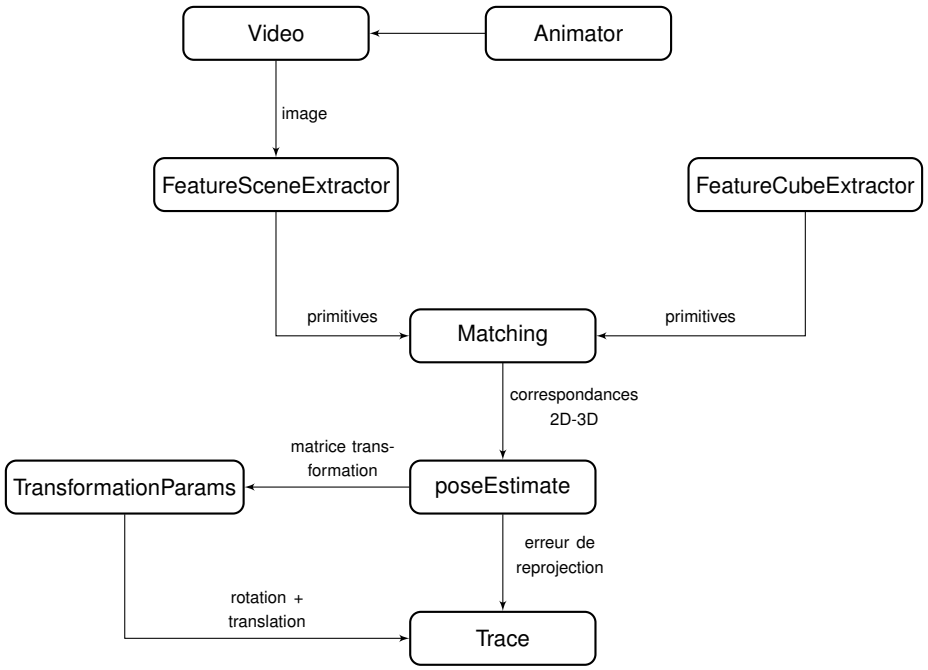


FIGURE 4.21 – Connexion entre les composants ARCS pour le tracé de l'erreur de reprojection et des paramètres de mouvement au cours du temps.

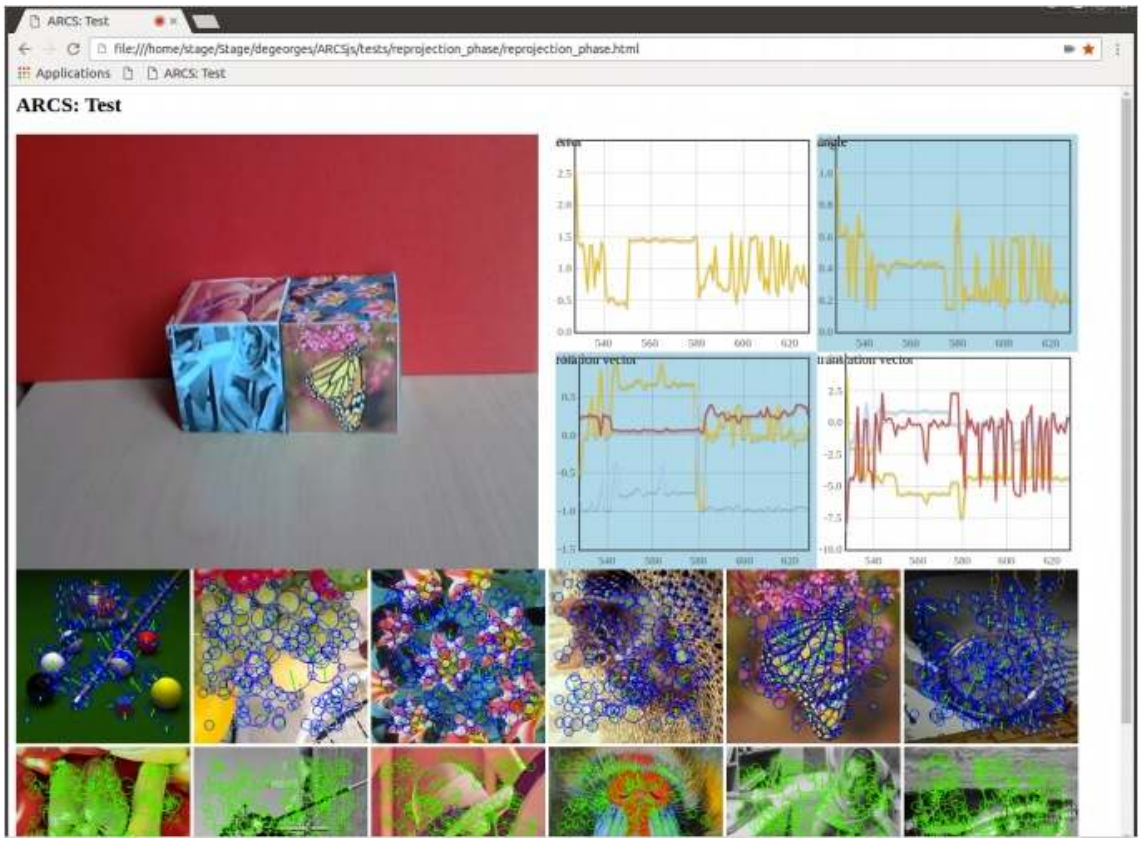


FIGURE 4.22 – Capture d'écran du navigateur.

4.2.2 Reconnaissance des gestes et des émotions pour la téléopération

Dans cette partie, nous aborderons des travaux préparatoire à la téléopération de robot par le geste. Il s'agit de reconnaître ces derniers, afin de transmettre des ordres de haut niveau au robot. Nous verrons comment nous pouvons identifier des émotions au travers des gestes dans l'idée d'adapter ultérieurement le comportement du robot par rapport à l'état émotionnel de l'utilisateur.

Les travaux présentés seront subdivisés en deux parties :

— La première indique comment nous avons utilisé et adapté des techniques de reconnaissance du geste telle que les modèles de Markov cachés. L'originalité réside dans le choix du vecteur descripteur des caractéristiques du geste, basé sur la méthode d'analyse de mouvement de Laban ;

— La deuxième concerne la reconnaissance des émotions lorsque le geste est effectué. Nous verrons comment nous avons mis au point un classifieur à même de reconnaître ces dernières.

Avant de les évoquer, nous préciserons le contexte de ces travaux, l'analyse du mouvement de Laban dont nous nous inspirons et les jeux de données que nous avons utilisé.

4.2.2.1 Contexte : téléopération de robot par le geste

Notre objectif est de développer une interaction gestuelle naturelle entre l'homme et le robot via un système de reconnaissance de geste. Le dispositif envisagé à terme, et décrit dans [AJILI et al. 2016], est représenté à la figure 4.23. L'idée est d'opérer un robot à distance et d'interagir avec lui en utilisant un casque de réalité virtuelle et une camera RGB-D (de type Kinect et donc capable de restituer la profondeur) pour capturer les gestes de l'utilisateur. Nous pensons nous appuyer sur un intergiciel pour communiquer avec le robot distant et récupérer les données de ses capteurs dont sa vision de la scène afin de l'augmenter et la restituer à l'opérateur.

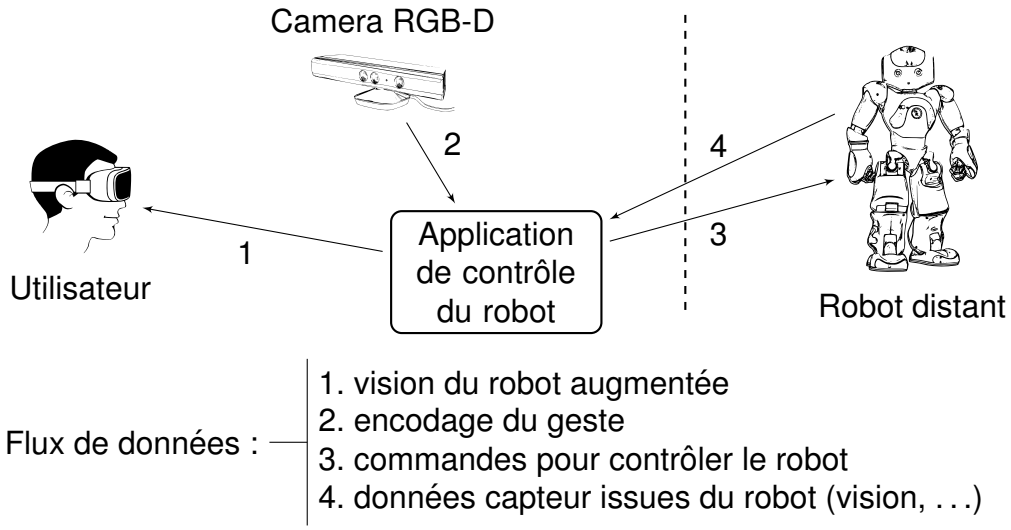


FIGURE 4.23 – Dispositif de téléopération de robot

Dans les travaux de cette partie, nous nous intéressons à la problématique de la reconnaissance du geste en nous appuyant sur les données retournées par le capteur RGB-D, c'est à dire le positionnement des articulations d'un squelette lié à l'utilisateur.

4.2.2.2 Description du mouvement selon Laban

L'une des particularités de nos travaux est qu'ils s'appuient sur l'analyse du mouvement de Laban [VON LABAN et ULLMANN 2011], une méthode et un langage qui permettent de décrire, visualiser, interpréter et documenter le mouvement humain. Mise au point par Rudolf Laban, elle a été à l'origine utilisée dans le domaine de la danse. Toutefois, des travaux récents l'utilisent dans le domaine de la reconnaissance du geste et l'émotion véhiculée par ce dernier [TRUONG et al. 2016].

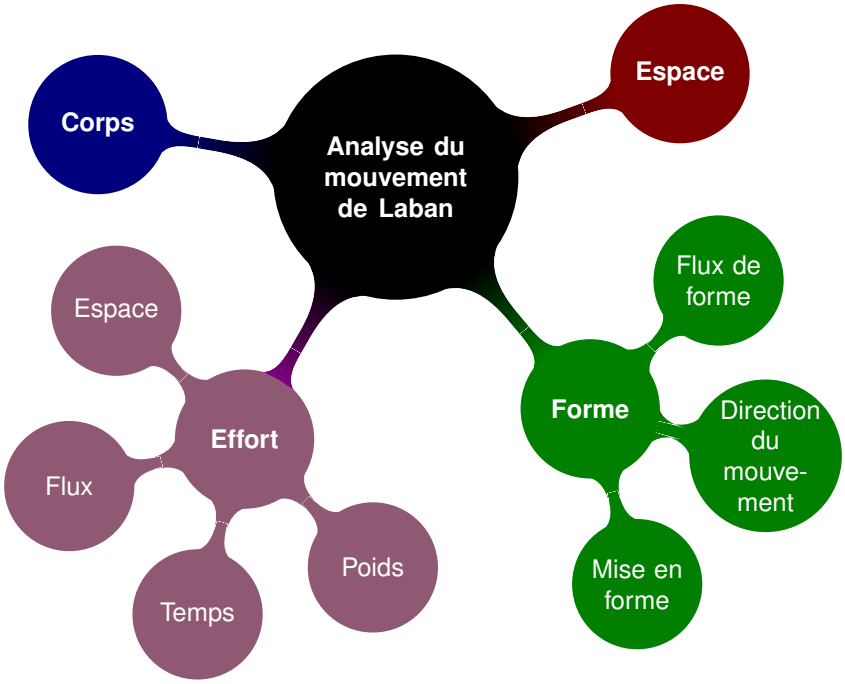


FIGURE 4.24 – Concepts utilisés dans la description du mouvement de Laban.

Dans l'analyse, le mouvement est décomposé suivant plusieurs composantes connaissant elles-même des subdivisions comme nous pouvons le voir à la figure 4.24.

- La composante **corps** définit quelles parties du corps sont en mouvement, de quelle manière et suivant quel séquençage ;
- La composante **espace** décrit dans quel espace s'inscrit le mouvement ;
- La composante **forme** s'intéresse à la manière dont le corps change de forme durant le mouvement :
  - Le **flux de forme** caractérise le changement de la forme du corps ;
  - La **direction du mouvement** définit le chemin du mouvement dans l'espace ;
  - La **mise en forme** représente la relation entre le corps en mouvement et l'espace 3D, en particulier le plan dans lequel celui-ci s'étend.
- La composante **effort** décrit l'expressivité du mouvement suivant quatre facteurs :
  - le **temps** répartit la rapidité du mouvement entre deux qualités (soudain et soutenu) et décrit le rythme du mouvement relativement à son urgence ;
  - le **poids** répartit la force du mouvement entre deux qualités (fort et léger). Il qualifie la force ou la puissance avec laquelle un mouvement est effectué ;
  - l'**espace** répartit la directivité du mouvement entre deux qualités (direct et indirect) et exprime la qualité de l'attention qu'une personne porte sur son environnement ;
  - le **flux** répartit la fluidité du mouvement entre deux qualités (lié et libre) et correspond au degré perçu de contrôle du mouvement.

Nous verrons ultérieurement comment employer ces caractéristiques pour construire des vecteurs descripteurs des mouvements de l'utilisateur et les classifier.

4.2.2.3 Jeux de données

Avant de rentrer dans le détail des méthodes que nous avons proposé, nous allons présenter les jeux de données sur lesquels nous avons travaillé. Nous nous sommes appuyés sur des bases publiques d'une part (MSRC-12 [FOTHERGILL et al. 2012], MSR Action 3D [LI et al. 2010] et UT Kinect [XIA et al. 2012]), afin d'avoir un point de comparaison pour notre méthode et sur nos propres acquisitions d'autre part (CMKinect-10 et ECMXsens-5). Toutes, à l'exception de ECMXsens-5, opèrent sur des données issues du capteur Kinect dont sont extraites les coordonnées de 20 articulations au cours du temps. En ce qui concerne ECMXsens-5, nous avons utilisé le système MVN Awinda [ROETENBERG et al. 2009] qui opère le suivi de 17 articulations. La table 4.5 indique les principales caractéristiques de chacune des bases.

Base	MSRC-12	MSR Action 3D	UT Kinect	CMKinect-10	ECMXsens-5
Capteur	Kinect	Kinect	Kinect	Kinect	MVN
Classes de geste	12	20	10	10	5
Nombre de personnes	30	10	10	20	11
Articulations suivies	20	20	20	20	17
Nombre de gestes	5653	567	200	2000	1100

TABLE 4.5 – Jeux de données pour la reconnaissance du geste.

Lors de la construction de nos propres jeux de données, à l'instar des autres bases, nous avons du normaliser ces dernières, afin de supprimer les variations dues au positionnement de l'utilisateur par rapport au capteur. Plus de détails sur cette procédure sont consignés dans [AJILI et al. 2017a].

4.2.2.4 Reconnaissance des gestes

La reconnaissance du geste fait partie d'un domaine étudié intensivement dans la littérature. L'utilisation du capteur Kinect ou d'un squelette 3D dans ce domaine font l'objet d'état de l'art comportant plus d'une centaine de référence [LUN et ZHAO 2015 ; PRESTI et LA CASCIA 2016]. L'intégration de cette approche pour une interface de téléopération n'est pas non plus nouvelle en soi. Pour ne citer que quelques exemples, [CANAL et al. 2015] applique des méthodes de reconnaissance du geste pour une interaction entre un utilisateur et plusieurs robots. Leur approche est basée sur l'utilisation d'une variante de la déformation temporelle dynamique (en anglais *Dynamic Time Warping* ou DTW) qui permet de mesurer la similarité entre deux suites qui varient au cours du temps. Dans [CICIRELLI et al. 2015], 10 gestes sont reconnus à l'aide d'un classifieur à réseau de neurones pour obtenir un haut score de reconnaissance. [LEHRMANN et al. 2014] utilise les modèles de Markov cachés pour reconnaître les gestes. Par ailleurs, leurs expérimentations utilisent en partie les mêmes bases publiques que nous ce qui nous permettra de comparer nos résultats.

En revanche, peu de recherches font appel à l'analyse du mouvement de Laban (LMA). Parmi les travaux les plus proches des notres, nous pouvons citer ceux de [TRUONG et ZAHARIA 2016] qui ont utilisé la méthode LMA pour classifier les gestes des chefs d'orchestre ainsi que l'expression véhiculée.

Nous allons détailler la méthode que nous proposons et les résultats obtenus.

Méthode proposée

La méthode que nous proposons pour la reconnaissance des gestes utilise les modèles de Markov cachés (MMC) et s'inspire de l'analyse du mouvement de Laban. Le MMC est caractérisé par des

probabilités de transition entre des états cachés et de génération de symboles observés à chaque état. L'idée est qu'un mouvement est décrit par une séquence d'observations et l'objectif est de trouver les relations de dépendance entre les états cachés et les observations.

L'application des MMC à la reconnaissance de mouvement passe par 4 étapes :

1. L'extraction des caractéristiques : il s'agit de l'étape de préparation des données ;
2. L'échantillonnage, afin que les gestes soient décrits avec un nombre fixe d'échantillons ;
3. La quantification, qui regroupe les données d'apprentissage en des classes discrètes ;
4. L'entraînement et la classification du geste proprement dit.

Caractéristiques retenues

En nous inspirant de l'analyse du mouvement de Laban, nous caractérisons le mouvement de la manière suivante :

— La description du **corps** comprend 10 mesures sur le squelette. Il s'agit d'angles et de distance permettant de discriminer parmi les différents gestes effectués ;

— L'**espace** se manifeste par le calcul de la direction normale du torse, ce qui ajoute 3 composantes à notre vecteur descripteur ;

— La **forme** est décomposée suivant ses sous-caractéristiques. Nous retenons :

  — le volume de l'enveloppe convexe du squelette pour caractériser le flux de forme ;

  — la courbure des trajectoires des mains et de la tête pour le mouvement directionnel ;

  — les distances entre l'articulation du torse à l'instant initial et les articulations de la tête, des bras (mains et coudes) et des jambes (pieds et genoux) projetées sur chaque plan (X, Y), (Y, Z) et (Z, X), soit 27 caractéristiques supplémentaires.

Au total, notre vecteur descripteur comprend 47 composantes.

Échantillonnage

L'échantillonnage consiste à réduire la taille des données et standardiser le nombre de vecteurs à associer à un geste. Ainsi, l'idée est de passer d'une séquence gestuelle s_{in} comprenant N vecteurs descripteurs f_i, $i \in [\![1, N]\!]$ à une séquence s_{out} n'en comprenant que T nommés h_j, $j \in [\![1, T]\!]$. Nous effectuons cet échantillonnage en deux étapes. Tout d'abord, nous éliminons les positions proches l'une de l'autre que nous considérons comme des doublons puis nous échantillonnons les séquences. L'algorithme 4 page suivante décrit la procédure d'échantillonnage retenue.

Quantification

L'étape de quantification est réalisée à l'aide de la méthode de K-moyennes. Elle consiste à répartir les vecteurs descripteurs obtenus dans un ensemble de K classes, chacune correspondant à un symbole particulier.

Entraînement et classification

Nous choisissons, pour notre modèle de Markov caché, la structure de Bakis qui est un modèle linéaire où seules les auto-transitions et les transitions aux états suivants sont autorisées. L'entraînement du MMC s'effectue avec l'algorithme de Baum-Welch [BAUM et al. 1970]. La phase de classification de nos gestes s'effectue à l'aide de l'algorithme *forward* qui calcule la probabilité d'appartenance du geste à une classe ou à une autre.

Algorithme 4 : Échantillonnage des vecteurs descripteurs.

Données : $s_{in} = \{f_1, f_2, \ldots, f_N\}, T$
Résultat : $s_{out} = \{h_1, h_2, \ldots, h_T\}$
// initialisation
$g_1 \leftarrow f_1$
$N' \leftarrow 1$
$k \leftarrow 1$
// filtrage des doublons
pour i *de* 2 *à* N **faire**
  $D \leftarrow \sqrt{\sum_l (f_{i,l} - f_{k,l})^2}$
  // D est la distance entre les deux vecteurs descripteurs
  si $D \geq \epsilon$ **alors**
    $g_j \leftarrow f_i$
    $N' \leftarrow N' + 1$
    $k \leftarrow i$
  fin
fin
// échantillonnage des données
$s \leftarrow \frac{N'}{T}$
pour i *de* 1 *à* T **faire**
  $j \leftarrow i \times s$
  $h_i \leftarrow g_j$
fin

Remarque 4.2 : Amélioration de la reconnaissance du geste

Les modèles de Markov cachés distinguent difficilement deux gestes dont le démarrage de la séquence est semblable. Pour traiter ce cas, nous utilisons deux MMC :

— le premier classifie le geste joué dans le sens normal ;

— le deuxième classifie le geste joué dans le sens inverse.

La probabilité finale pour un geste d'appartenir à une classe sera le maximum des deux probabilités retournées par chaque modèle.

Expérimentations et résultats

Nous avons testé notre méthode sur les bases MSRC-12, MSR Action 3D, UT Kinect et CMKinect-10. Pour chacun des tests, nous avons fait varier deux paramètres :

— K, le nombre de classes utilisées lors de l'étape de quantification ;

— S, le nombre d'états de notre modèle de Markov caché.

Dans le cas de MSRC-12 et MSR Action 3D, les meilleurs taux de reconnaissance sont obtenus pour $K = 40$ et S valant 5 ou 10. Dans le cas de UTKinect, ils sont obtenus pour $K = 30$. Dans notre base, les meilleurs taux sont obtenus pour $K = 40$ et $S = 20$. Les résultats que nous obtenons sont comparables à ceux de la littérature voire un peu meilleurs.

Ainsi, les table 4.6 et 4.7 page suivante indiquent les taux que nous avons obtenus respectivement sur les bases MSRC-12 et MSR Action 3D. MSRC-12 sépare ses gestes en deux catégories, iconiques

et métaphoriques. Pour MSR Action 3D, ils sont subdivisés en trois catégories (AS1, AS2 et AS3). Les deux premières regroupent des gestes par similarité tandis que la troisième contient des gestes considérés comme plus complexes. Nous avons conservé ces distinctions à des fins de comparaison. Nous indiquons également les résultats de l'application de notre méthode lorsqu'il s'agit d'un MMC simple ou d'un MMC double (voir remarque 4.2).

Méthodes	Gestes iconiques	Gestes métaphoriques
LEHRMANN et al. 2014	90.90	-
SONG et al. 2013	79.77	81.00
TRUONG et al. 2016	88.60	75.20
Notre méthode (MMC simple)	**96.81**	**83.13**
Notre méthode (MMC double)	**98.07**	**93.33**

TABLE 4.6 – Taux de reconnaissance comparés sur la base MSRC-12.

Méthodes	AS1	AS2	AS3	Moyenne
LI et al. 2010	72.90	71.90	79.20	74.66
ALWANI et al. 2014	86.30	65.40	77.70	76.46
XIA et al. 2012	87.48	85.48	63.46	78.97
SOH et DEMIRIS 2012	80.60	74.90	87.10	80.87
YANG et TIAN 2014	74.50	76.10	96.40	82.33
Notre méthode (MMC simple)	**83.65**	**80.76**	**84.61**	**83.00**
CHAARAOUI et al. 2012	87.90	74.12	89.21	83.74
NEGIN et al. 2015	82.66	83.33	87.17	84.38
GHORBEL et al. 2015	83.08	79.46	93.69	85.41
Notre méthode (MMC double)	**84.61**	**84.61**	**90.30**	**86.50**
CHAARAOUI et al. 2014	91.59	90.83	97.28	93.23

TABLE 4.7 – Taux de reconnaissance comparés sur la base MSR Action 3D.

4.2.2.5 Reconnaissance des émotions au travers du geste

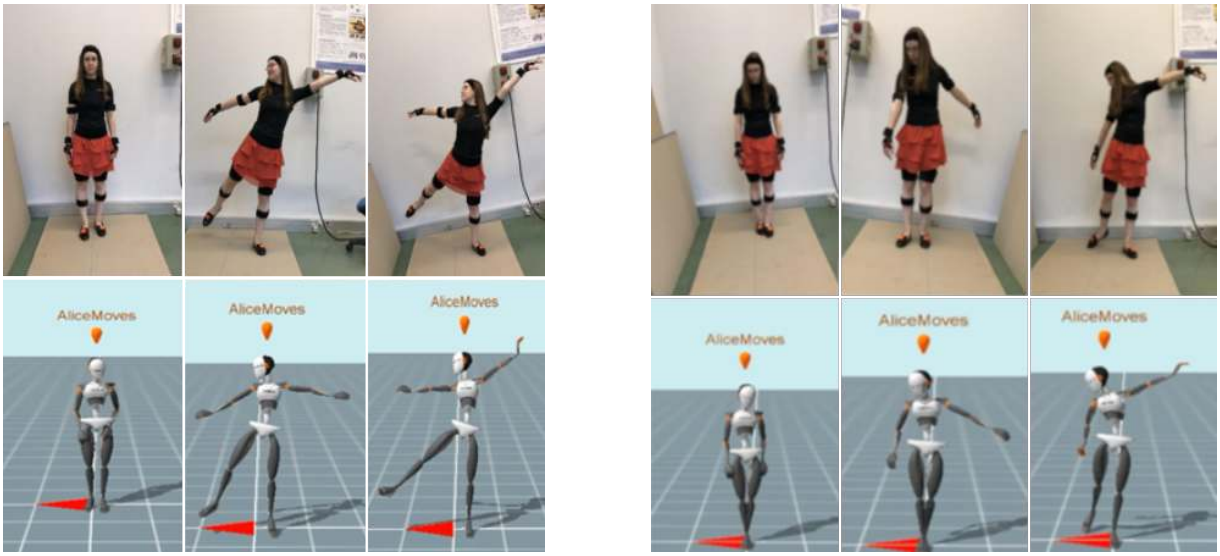
Nous avons souhaité aller plus loin et voir s'il était possible de reconnaître les émotions de l'utilisateur au travers de son geste. Ce point semble fondamental pour [KLEINSMITH et BIANCHI-BERTHOUZE 2013] qui examinent l'importance de l'expression du corps lors d'une communication non-verbale pour créer des technologies sensibles aux sentiments.

De part sa nature, l'analyse du mouvement de Laban dont certaines composantes s'attachent à l'expressivité du geste tient une place privilégiée dans la reconnaissance des émotions au travers du langage corporel. Nous pouvons citer les travaux de [ZACHARATOS et al. 2013] qui se basent sur la composante *effort* dans le cadre de la reconnaissance d'émotions pour le jeu vidéo. Ils emploient un perceptron multi-couche pour reconnaître quatre états émotionnels (méditation, concentration, allégresse et frustration).

D'autres études, moins complètes, puisque se focalisant sur un seul geste, exploitent l'analyse du mouvement de Laban. Ainsi, [LOURENS et al. 2010 ; BARAKOVA et LOURENS 2010] se focalisent sur le geste de salutation, exprimé avec quatre états émotionnels différents (colère, tristesse, joie et politesse). Ils ont étudié les liens entre ces dernières et les caractéristiques *temps*, *poids* et *flux* de la composante *effort* du modèle de Laban. [CIMEN et al. 2013] se concentrent sur le geste de la marche en exprimant quatre émotions différentes (colère, joie, tristesse, détente). Ils l'ont classifié en utilisant une machine à vecteur de support.

Pour notre part, nous allons montrer comment nous pouvons reconnaître les émotions au travers de plusieurs gestes. Tout d'abord, nous expliquerons comment nous avons construit notre base de gestes puis comment nous modifions notre descripteur. Dans le cadre de notre étude, nous avons comparé les performances de plusieurs méthodes de classification, puis examiné quelles caractéristiques sont réellement discriminantes. Enfin, nous établirons un parallèle avec les caractéristiques pertinentes pour l'être humain.

Construction d'une base de gestes expressifs



(a) état de joie (b) état de tristesse

FIGURE 4.25 – Geste de danse effectué avec différentes émotions

Il s'agit de la base ECMXens-5 que nous avions déjà brièvement introduite. Elle est composée de 5 mouvements (danser, avancer, faire un signe, pointer et s'arrêter) durant lesquels sont exprimés l'état neutre, de joie, de colère et de tristesse. A titre d'exemple, la figure 4.25 montre le mouvement de danse exprimé avec deux états émotionnels : la joie et la tristesse. Onze personnes ont participé à l'élaboration de la base. Elles ont répété chaque geste cinq fois pour chaque état émotionnel à exprimer. Pour les aider à s'imprégner de ces derniers, nous leur avons proposé la lecture de scénarios véhiculant l'émotion recherchée. A l'aide du système MVN Avinda de XSens, nous avons mesuré les coordonnées de 17 articulations au cours du temps. Nous disposons ainsi d'un *corpus* de 1100 gestes anonymisés.

Descripteur d'un geste expressif

Nous reprenons le descripteur utilisé précédemment pour en extraire des caractéristiques globales sur l'intégralité du geste. Ainsi,

- pour la composante de *corps*, nous calculons la moyenne, l'écart type et la plage (minimum et maximum) des caractéristiques extraites pour la phase de reconnaissance ;
- pour la composante de *forme*, nous calculons :
  - la moyenne, l'écart type et la plage du facteur *flux de forme* ;
  - les mouvements moyens relatifs à la position initiale du torse sur les plans principaux pour le facteur de *mise en forme* ;
  - l'indice de courbure global pour la *direction du mouvement*.

Nous lui ajoutons des caractéristiques associées à la composante *effort* pour lesquelles nous ne nous intéresserons qu'aux articulations du haut du corps, à savoir la tête, les mains et le torse :

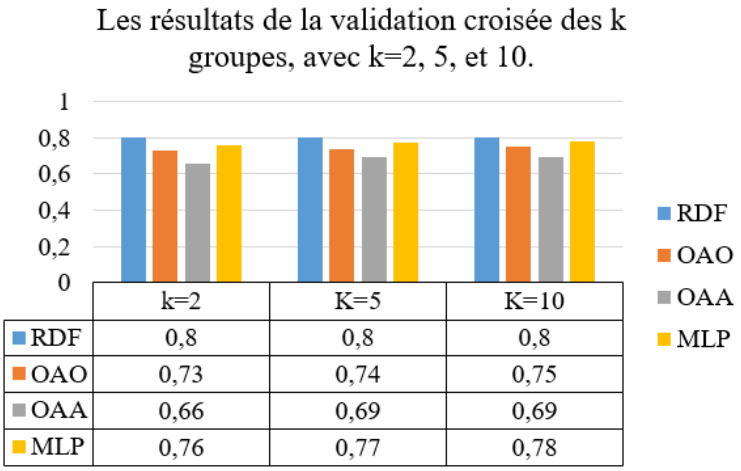
— Le facteur d'*espace* est caractérisé par le rapport entre la distance linéaire séparant la position d'une articulation en début et fin de mouvement et la somme des distances entre les positions successives de cette articulation au cours du geste ;

— Le facteur de *temps* est determiné en calculant la vitesse moyenne des articulations du haut du corps ainsi que l'écart-type et la plage de chaque vitesse ;

— Le facteur de *poids* utilise un mode de calcul similaire pour les accélérations ;

— Le facteur de *flux* est identifié par les débattements des angles de lacet et de tangage pour les articulations.

Choix de la méthode de classification



	k=2	K=5	K=10
RDF	0,8	0,8	0,8
OAO	0,73	0,74	0,75
OAA	0,66	0,69	0,69
MLP	0,76	0,77	0,78

FIGURE 4.26 – F-scores moyens des classifieurs appliqués sur la base ECMXsens-5.

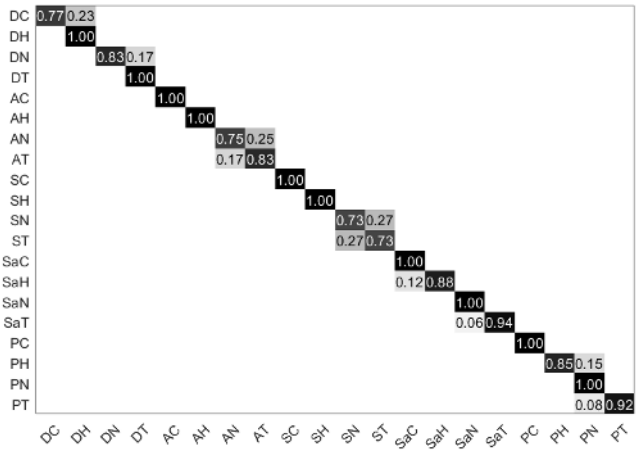


FIGURE 4.27 – Matrice de confusion de la méthode RDF appliquée sur la base ECMXsens-5.

Nous avons mené une étude préliminaire avec quatre méthodes de classification en raison de leur disponibilité dans la bibliothèque Scikit-Learn [PEDREGOSA et al. 2011] :

— les forêts d'arbres décisionnels (*Random Tree Forest* – RDF) ;

— le perceptron multi-couche (*Multi-layer perceptron* – MLP) ;
— la machine à vecteur de support multi-classes (*Support Vector Machine* – SVM) dans ses variantes un-contre-un (*one-against-one* – OAO) et un-contre-tous (*one-against-all* – OAA).

Nous utilisons la technique de validation croisée de k groupes qui consiste à partitionner les données en k groupes, dont l'un permet de tester les capacités de généralisation du classifieur et les autres sont utilisées pour l'apprentissage. Nous mesurons le score de performance de la méthode (k-score) puis nous la répétons k fois en changeant la partitition sur laquelle le test de généralisation est effectué. Le score final de la méthode est la moyenne des k-scores obtenus.

La mesure de performance utilisée est le F-score qui combine les mesures de précision et de rappel (permettant de prendre en compte les cas où le classifieur se trompe) donnée par la formule suivante :

$$Fs = 2\frac{p \times r}{p + r} \quad \text{où} \quad p = \frac{V_p}{V_p + F_p} \quad \text{et} \quad r = \frac{V_p}{V_p + F_n}$$

p désigne la précision, r le rappel, V_p le nombre de vrais positifs, F_p le nombre de faux positifs et F_n le nombre de faux négatifs.

Les valeurs de k choisies sont 2, 5 et 10 et nous comparons les différentes méthodes sur les bases MSRC-12, MSR Action 3D, UTKinect et ECMXsens-5. Des différentes expérimentations, il en ressort que la méthode RDF donne les meilleurs scores de performance. A titre d'exemple, nous indiquons les résultats obtenus par les classifieurs sur la base ECMXsens-5 (figure 4.26 page précédente) et la matrice de confusion résultante pour la méthode RDF (figure 4.27 page précédente).

Analyse des caractéristiques discriminantes

Dans une seconde phase, afin d'optimiser le fonctionnement de l'algorithme d'apprentissage, nous cherchons à déterminer, parmi les caractéristiques de notre vecteur descripteur lesquelles sont réellement discriminantes, en vue de supprimer les informations non pertinentes ou redondantes.

L'algorithme RDF est intéressant en ce qu'il est également capable de mesurer la pertinence des caractéristiques par le bais de taux d'erreur calculés sur des échantillons de test lors de la phase d'apprentissage. Afin de déterminer le sous-ensemble des caractéristiques pertinentes, nous appliquons l'algorithme 5 page suivante.

La table 4.8 page 176 indique les résultats obtenus pour chaque combinaison de geste et d'émotion. Afin de déterminer si une caractéristique est discriminante ou non, nous la comparons, pour chaque combinaison, au même geste effectué dans l'état neutre. Si elle est estimée pertinente dans les deux situations, alors elle n'est pas discriminante. Il en ressort que les facteurs de la composante *effort*, en particulier le *temps* et le *poids*, caractérisent la joie et la colère tandis que la tristesse est essentiellement décrite par les composantes *effort* et *forme* (dans laquelle le facteur de *direction de mouvement* est négligeable).

Comparaison avec les performances humaines

Nous nous sommes demandés si notre classifieur utilisait les mêmes caractéristiques discriminantes que l'être humain. A cette fin, nous avons mené une évaluation avec 10 personnes qui ont été invitées à regarder les gestes exprimant des émotions rejoués par un avatar virtuel sans expression faciale (figure 4.28 page 177). Les participants ont évalué l'émotion exprimée par chacun des gestes à l'aide d'une échelle de Likert allant de 1 à 5 (5 étant le score maximal d'adéquation entre l'émotion et le geste observé). Nous nous sommes assurés, à la fin de notre expérimentation que les appréciations des utilisateurs étaient cohérentes en calculant le coefficient de Cronbach, qui est resté supérieur à $0,8$ (la plus basse valeur est $0,814$ pour l'état neutre) quelque soit l'état émotionnel observé, ce qui

Algorithme 5 : Méthode de sélection des caractéristiques pertinentes.

Données : $v_0 = \{f_i\}, i = 1, \ldots, p$, ensemble des caractéristiques.
Résultat : $v^* = \{f_j\}, j = 1, \ldots, p^*$, sous-ensemble des caractéristiques les plus pertinentes.

$k \leftarrow 0$
tant que $Card(v_k) \geq 1$ **faire**
  Calculer et enregistrer le taux d'erreur d'échantillonnage $E_k(v_k)$
  pour i *de* 1 *à* p **faire**
    Calculer $I(f_i)$ `// importance de chaque caractéristique dans` v_k
  fin
  Trier $\{f_i\}$ dans l'ordre décroissant suivant les valeurs de $I(f_i)$
  $f_{min} \leftarrow \underset{i}{\text{argmin}}\{I(f_i)\}$
  Sélectionner l'ensemble des caractéristiques $\{f_t\}$ qui ne contribuent pas à un changement significatif de E_k (Test de Tukey)
  $R \leftarrow f_{min} \bigcup \{f_t\}$
  $v_{k+1} \leftarrow v_k \backslash R$
  $k \leftarrow k + 1$
fin
$m \leftarrow \underset{l}{\text{argmin}}\{E_l(v_l))\}$
$v^* \leftarrow v_m$

indique une grande cohérence entre les observations des utilisateurs. La figure 4.29 page 177 indique les scores moyens évalués pour chaque émotion exprimée.

Dans un deuxième temps, nous avons demandé à un deuxième groupe de 10 personnes d'évaluer les composantes de LMA discriminantes au niveau du geste. Chaque participant reporte l'adéquation entre le geste exprimée et la composante LMA concernée sur une échelle de 1 à 7. Les coefficients de Cronbach calculés sont supérieur à $0,7$ (le plus bas score étant $0,703$), ce qui indique une forte cohérence entre les estimations des participants.

Nous mesurons ensuite la corrélation entre les évaluations faites dans la perception des émotions et dans la caractérisation des facteurs LMA afin de déterminer les relations entre elles. Nous les exprimons sous la forme des coefficients de corrélation de Pearson qui sont compris entre -1 (indiquant une corrélation parfaitement négative) et 1 (corrélation parfaitement positive). Un coefficient proche de 0 indique que la composante n'est pas discriminante.

La table 4.9 page 178 regroupe les coefficients de Pearson calculés.

Nous observons que la joie est associée à tous les facteurs de *forme* et est caractérisée par le *poids* et le *temps* au niveau de la composante d'*effort*. L'état de joie est donc caractérisé par une augmentation de la *forme* et une extension des membres du corps, tout en étant associée à la rapidité et à la force du mouvement. Ceci est corroboré par l'étude de [MASUDA et KATO 2010] qui confirme la relation avec les qualités d'*effort*. Notre classifieur automatique s'appuie également sur ces caractéristiques mais fait abstraction des facteurs de *forme*.

La colère est fortement corrélée avec tous les facteurs de la composante d'*effort*. Elle est aussi associée au facteur de *direction du mouvement*. L'émotion de la colère est donc caractérisée par un mouvement rapide, fort, direct, rectiligne et libre. Nous relevons des disparités avec les résultats de [MASUDA et KATO 2010] qui sont plus proches de ceux de notre classifieur automatique, à savoir que les paramètres discriminants sont le *poids* et le *temps*.

L'émotion de la tristesse est négativement corrélée avec tous les facteurs d'*effort* et de *forme* à l'exception du facteur de flux. Elle est donc caractérisée par une forme rétrécie, des extrémités du corps contractées, un mouvement courbé tout en étant léger, lié, soutenu et indirect. [MASUDA et KATO 2010] confirment ces résultats et notre classifieur automatique exploite les mêmes informations.

Gestes expressifs		**Descripteur**					
		Forme		**Effort**			
		Flux de forme	Mise en forme	Temps	Poids	Espace	Flux
Joie	Avancer			E_{v^l} P_{v^l} E_{v^r} P_{v^r}	M_{a^l} E_{a^l} M_{a^r} E_{a^r} P_{a^r} E_{a^h} P_{a^h}		
	Danser				M_{a^l} E_{a^l} E_{a^r} M_{a^h} E_{a^h}		
	Faire un signe			E_{v^l} P_{v^r}	E_{a^l} E_{a^r}		
	S'arrêter			M_{v^l} E_{v^l} P_{v^l} M_{v^r} E_{v^r} P_{v^r} E_{v^h} P_{v^h}	M_{a^l} E_{a^l} P_{a^l} M_{a^r} E_{a^r} P_{a^r} M_{a^h} E_{a^h} P_{a^h} M_{a^s} E_{a^s} P_{a^s}		
	Pointer			P_{v^l} E_{v^r} P_{v^r} E_{v^s} P_{v^s}	M_{a^l} M_{a^r} E_{a^r} P_{a^r} M_{a^h} E_{a^h} P_{a^h} M_{a^s} E_{a^s} P_{a^s}		
Colère	Avancer			P_{v^l},	E_{a^l} E_{a^r} P_{a^r}		
	Danser				E_{a^l} P_{a^l} E_{a^h}		
	Faire un signe			P_{v^r}	M_{a^l} E_{a^l} P_{a^l} M_{a^r} E_{a^r} P_{a^r}		
	S'arrêter				M_{a^h} E_{a^h} M_{a^s}		
	Pointer				E_{a^r},P_{a^r}		
Tristesse	Avancer	M_{datav} E_{datav} P_{datav}	D_H D_F	M_{v^l} E_{v^l} M_{v^r} E_{v^r}	M_{a^l} M_{a^r} M_{a^h} M_{a^s}	S^l S^r	$P^r_{tangage}$ P^r_{lacet} $P^h_{tangage}$ P^h_{lacet}
	Danser	E_{datav}		E_{v^r},P_{v^r}	M_{a^l} E_{a^l} P_{a^l} E_{a^r} M_{a^s} E_{a^s}	S^r	$P^l_{tangage}$ P^r_{lacet} $P^h_{tangage}$
	Faire un signe		D_H D_S	M_{v^l} E_{v^l} P_{v^l} M_{v^r} E_{v^r} P_{v^r},	M_{a^l} E_{a^l} M_{a^r} E_{a^r}		
	S'arrêter	E_{datav} P_{datav}	D_F	M_{v^l} E_{v^l} P_{v^l} M_{v^r} E_{v^r} P_{v^r} M_{v^h} E_{v^h} E_{v^s} P_{v^s}	M_{a^l} E_{a^l} $M_{a^r}E_{a^r}$ M_{a^s} E_{a^s}	S^s	$P^r_{tangage}$
	Pointer	M_{datav} P_{datav}		M_{v^r} E_{v^r} P_{v^r}	M_{a^r} E_{a^r} Pa^r	S^r	$P^h_{tangage}$ P^h_{lacet}

TABLE 4.8 – Caractéristiques discriminantes relevées pour chaque combinaison d'émotion et de geste.

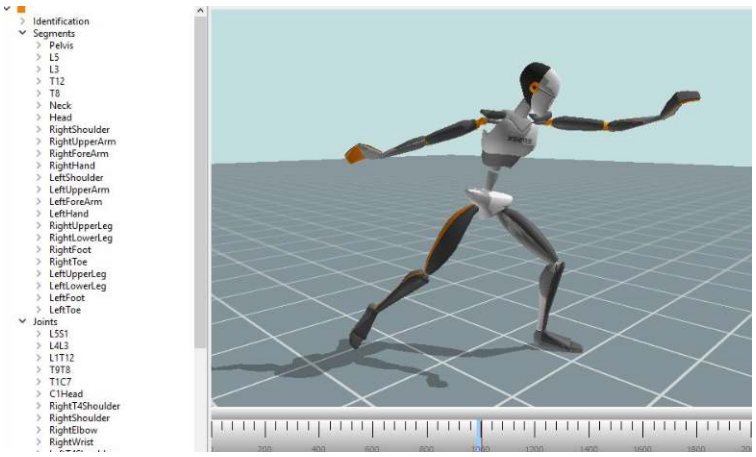


FIGURE 4.28 – Reproduction des gestes par un avatar.

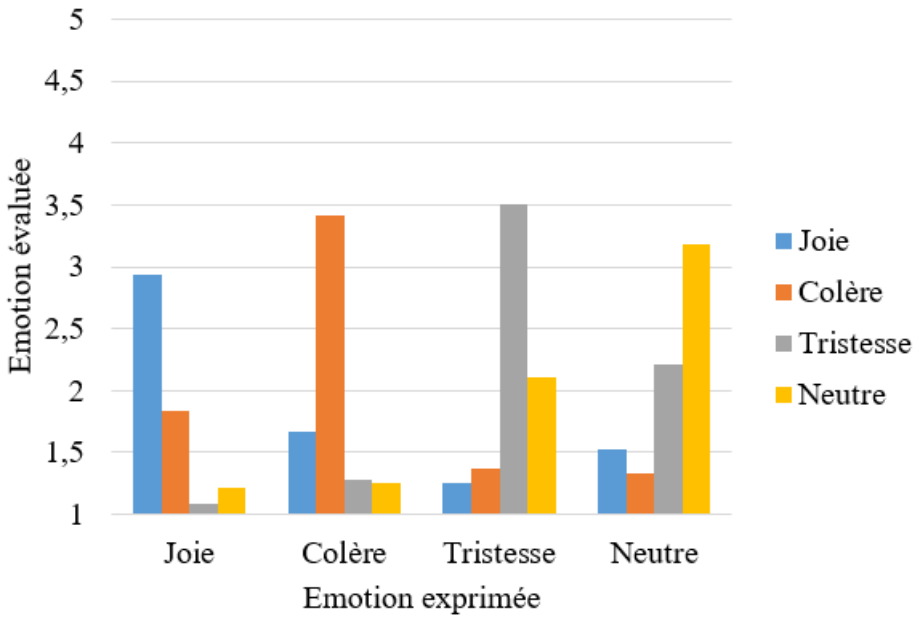


FIGURE 4.29 – Scores moyens de perception des émotions.

En résumé, notre classifieur automatique des émotions exploite, à quelques différences près, les mêmes caractéristiques que celles d'une personne pour différencier les émotions au travers des gestes. Ces résultats sont, de plus, cohérents avec ce que l'on peut lire dans la littérature.

4.2.3 Bilan et perspectives

Dans cette partie, nous avons présenté une méthode permettant de reconnaître les gestes d'un opérateur, encodés sous la forme de squelette liant les positions des articulations au cours du temps, à des fins de téléopération. Cette méthode, basée sur les modèles de Markov cachés et s'inspirant de l'analyse du mouvement de Laban, donne des performances comparables à celles de la littérature, voire meilleures en fonction des bases utilisées. Il est à noter que l'ajout d'un deuxième modèle de Markov parcourant la séquence du geste dans le sens inverse a contribué à améliorer les performances de notre système.

Nous avons ensuite mis au point un système de reconnaissance des émotions exprimées au travers les gestes dans le but de mettre au point une interaction homme-robot consciente de l'état émotionnel de l'opérateur. Nous avons comparé quatre algorithmes de classification pour conclure que la méthode

		Joie	Colère	Tristesse	Neutre
Forme	Flux de forme	**0.541**	0.132	**-0.524**	-0.316
	Mise en forme	**0.542**	0.194	**-0.613**	-0.328
	Mvt Directionnel	0.269	**0.568**	**-0.505**	-0.397
Effort	Temps	**0.555**	**0.622**	**-0.795**	**-0.554**
	Poids	**0.543**	**0.640**	**-0.780**	**-0.594**
	Espace	0.326	**0.682**	**-0.566**	-0.487
	Flux	-0.316	**-0.665**	**0.559**	0.497

TABLE 4.9 – Coefficients de Pearson pour la corrélation entre les scores donnés aux facteurs de Effort-Forme et ceux donnés aux émotions exprimées.

la plus pertinente est celle des forêts d'arbres décisionnels (RDF). Ensuite, nous nous sommes attachés à déterminer quelles sont les caractéristiques les plus pertinentes associées aux gestes afin d'optimiser le processus de reconnaissance des émotions. Nous en avons profité pour comparer les caractéristiques jugées pertinentes par notre classifieur par rapport à celles relevés par un participant humain et à ce qui est déjà connu dans la littérature. Le résultat indique quelques divergences mineures et un accord sur le fond des trois méthodes, ce qui semble indiquer que notre classifieur peut-être utilisé dans le cadre que nous lui avons fixé.

Au niveau des perspectives, ce travail reste de la reconnaissance de geste hors ligne (c'est à dire effectué *a posteriori*). Il nous reste à relever le défi de le faire fonctionner en ligne, ce qui nécessitera des adaptations non triviales à réaliser. De plus, il nous reste à mettre au point le système global de téléopération. C'est dans ce contexte que les travaux amorcés en matière de distribution des architectures logicielles vont trouver une nouvelle voie d'application et, peut-être, nous permettre d'envisager des améliorations en termes d'architecture pour répondre au mieux aux besoins.

Remarque 4.3 : Valorisation des résultats

Les travaux présentés dans la partie de reconnaissance des émotions font l'objet, à l'heure de l'écriture de ces lignes, d'une valorisation en cours. L'une des étapes clés est la soutenance de la thèse prévue en décembre 2018.

4.3 Interaction multi-modale

Comme nous l'avions relevé à la section **??** page ??, la réalité augmentée peut-être multimodale, c'est à dire augmenter la perception de plusieurs sens à la fois. Le présent chapitre présente une de nos incursions dans le domaine de l'haptique [2] et de l'exploration de ses liens avec la réalité augmentée.

4.3.1 Réalité augmentée haptique

Remarque 4.4 : Contexte des travaux sur la réalité augmenté haptique

Ce contexte est un peu particulier puisqu'il s'agit d'une thèse dont j'ai rejoint l'encadrement tardivement, mon arrivée en poste coincidant avec la dernière année du doctorant. Les concepts présentés ici sont donc antérieurs à mon arrivée. J'ai par contre été pleinement associé au travaux qui en ont découlé.

Nous avons travaillé sur deux applications explorant le concept de la réalité augmentée haptique. Dans un travail préliminaire [BAYART et KHEDDAR 2006b], cette dernière avait été subdi-

2. L'haptique englobe le sens du toucher et la perception du corps dans l'environnement (kinesthésie).

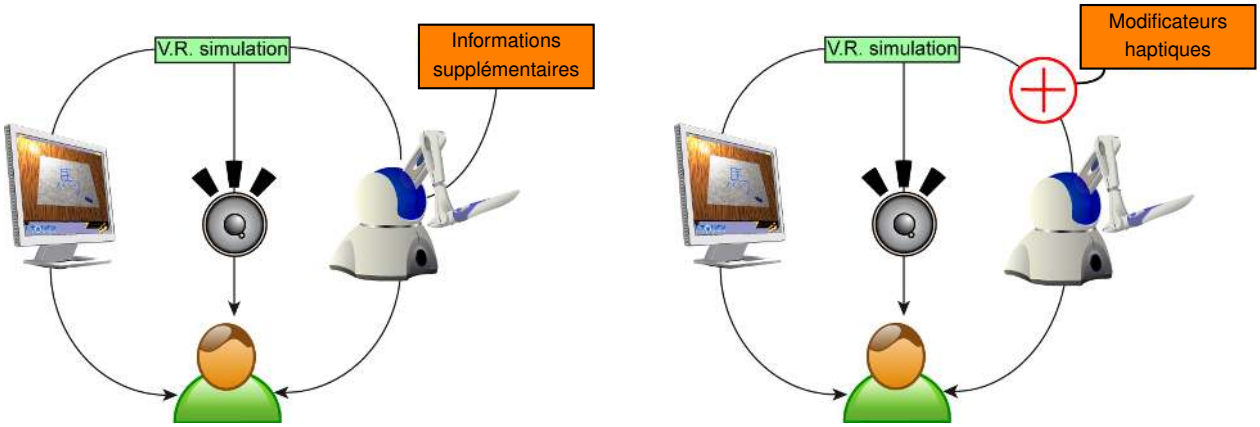
visée en deux catégories, l'haptique augmentée et l'augmentation haptique. La figure 4.30 illustre ces concepts.

Définition 4.3 : Augmentation haptique

L'augmentation haptique ou *haptic enhancing* est utilisé pour mélanger deux types d'informations haptique : la première étant un retour direct de l'application et la deuxième représentant une information ajoutée sur le caractère haptique en accord avec un *mapping* prédéfini ou une règle mathématique.

Définition 4.4 : Haptique augmentée

L'haptique augmentée ou *enhanced haptic* survient quand la modalité haptique amplifie ou module une donnée haptique renvoyée à l'utilisateur.



(a) Augmentation haptique ou *haptic enhancing* ; une composante haptique est ajoutée pour fournir une information supplémentaire

(b) Haptique augmentée ou *enhanced haptic* ; la composante haptique existante est modifiée (amplification, extrapolation, ...)

FIGURE 4.30 – Formes de réalité augmentée haptique [BAYART et KHEDDAR 2006b]

Nous avons mis en oeuvre ces concepts au travers de deux applications :

— L'exploration haptique d'un objet mixte [BAYART et al. 2007] ;

— L'application de peinture virtuelle sur un objet distant [BAYART et al. 2008].

Avant de détailler ces travaux, nous verrons quels étaient, à l'époque les travaux explorant les liens entre haptique et réalité augmentée. Nous décrirons ensuite rapidement le dispositif qui nous a permis de toucher à distance des objets et sur lequel nous avons pu développer ces concepts.

De l'inclusion de l'haptique en tant que modalité en réalité augmentée

Plusieurs système ont, à l'époque intégré l'haptique en tant que modalité dans un système de réalité augmentée.

Les premiers systèmes se sont attachés à calibrer les différents dispositifs pour faire cohabiter l'espace haptique et l'espace visuel. Toutefois, ces premières applications ne permettaient que de toucher des objets virtuels. Ainsi [VALLINO et BROWN 1999] introduit un système de réalité augmentée produisant à la fois des augmentations visuelles et haptiques en temps réel. [BIANCHI et al. 2006] ont revu et amélioré le procédé de calibration afin d'intégrer de manière plus harmonieuse le retour haptique et les augmentations visuelles. Dans [YE et al. 2003], les auteurs ont conçu et réalisé un système appelé VisHap qui utilise le suivi visuel pour fournir un retour haptique à la demande. L'utilisateur

peut déplacer son doigt dans l'environnement et un bras à retour d'effort vient à la rencontre de ce dernier lorsqu'il s'approche d'une zone où il devrait rencontrer un objet virtuel.

Les premiers travaux où sont apparus une combinaison de retour haptique réel et virtuel sont ceux de [BORST et VOLZ 2005] qui tire partie de l'utilisation d'un gant à retour d'effort, ce qui permet à l'utilisateur de toucher les objets réels (le gant étant passif) et de ressentir également les objets virtuels (le gant étant actif). Ce sont les travaux de [TURRO et KHATIB 2001] et de [HWANG et al. 2006] qui ont fait apparaître une combinaison de retours haptique actifs permettant de toucher objets réels et virtuels. Le concept utilisé est alors celui de la téléopération, le bras haptique commandant un dispositif mécanique distant permettant de toucher l'environnement et de mesurer les forces de contact exercées.

Le système que nous avons exploité entre dans cette catégorie, la distinction étant dans les applications réalisées et dans une exploration systématique des concepts d'augmentation haptique et d'haptique augmentée.

4.3.2 Système de réalité augmentée haptique

La figure 4.31 donne le synoptique de fonctionnement du système. Du côté de l'utilisateur, la restitution visuelle est opérée par un écran et le retour haptique par un bras robotisé (*Omni* de *Sensable*). Du côté du site téléopéré, une caméra (*Fire-I* de *Unibrain*) filme la scène et une table cartésienne (ou table XYZ qui peut-être asservie en position dans l'espace) couplée à un capteur (sonde haptique) mesurant les forces de contact suivant les 6 degrés de liberté est interfacée à un ordinateur distant pour la contrôler. Un descriptif plus complet de cette table est donnée dans KHEDDAR et al. 2004. Le site maître et le site esclave communiquent via le protocole TCP/IP, ce qui occasionne un délai d'une milli-seconde entre les deux.

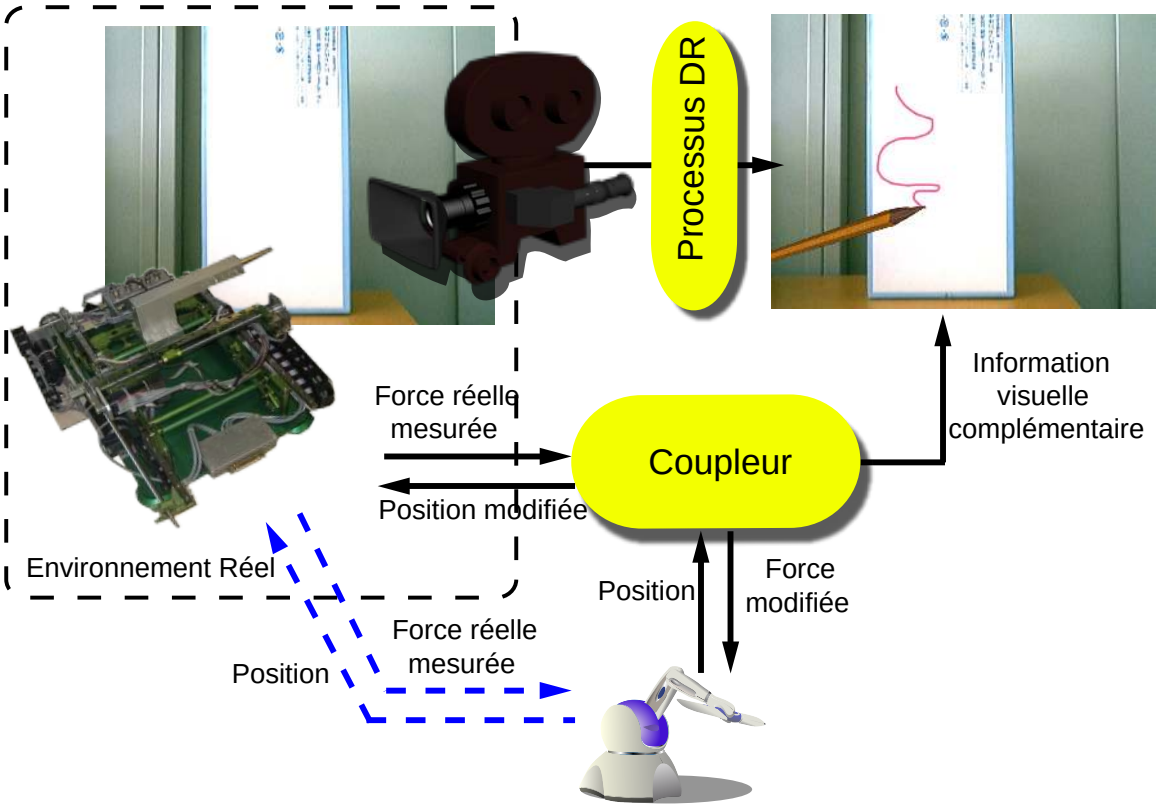


FIGURE 4.31 – Synoptique du système de réalité augmentée haptique

Au niveau logiciel et contrôle, plusieurs modalités d'augmentation sont fournies. Tout d'abord,

un processus de réalité diminuée (*diminished reality – DR*) intervient entre l'image capturée sur le site et sa restitution à l'écran. Son objectif est de faire disparaître la table XYZ du champ de vision afin de renforcer le sentiment de présence de l'utilisateur. Un sous-système de couplage récupère les commandes en position du bras haptique pour les traiter et les envoyer à la table XYZ. Cette dernière envoie en retour ses mesures des forces de contact qui seront restituées, au travers du coupleur, au bras haptique. De plus, le coupleur ajoutera des informations visuelles à la scène filmée.

Recalage des environnements virtuels et réel

Afin d'interagir à la fois dans les environnements réel et virtuel, une étape de calibration est nécessaire. En fonction du type d'application, il sera nécessaire de connaître la pose (position et orientation) de l'extrémité de la sonde par rapport à la caméra ou par rapport à l'objet exploré.

Dans ce qui suit, nous appelerons T_{xy} la transformation géométrique permettant de passer du repère local d'un objet X (noté $\{x\}$) à celui d'un objet Y (noté $\{y\}$).

La première étape de notre procédure de calibration consiste à déterminer la pose de la sonde (p) à l'étape initiale. Nous effectuons donc une calibration des paramètres de la caméra à l'aide de la méthode de ZHANG 1999, puis utilisons plusieurs marqueurs de la bibliothèque ARToolkit [KATO et BILLINGHURST 1999]. L'un de ces marqueurs est placé à la base de la sonde (b), avec un repère local aligné à celui de la table cartésienne, et un autre sur l'objet (o). La transformation rigide (T_{pb}) entre la base et l'extremité de la sonde est supposée connue et peut se mesurer physiquement. Ces repères sont représentés à la figure 4.32. Les transformations T_{cb} et T_{co} sont obtenues par estimation de la pose des marqueurs dans l'image.

Dans le cas où l'objectif est de déterminer la position de l'extremité de la sonde par rapport à l'objet, la transformation est calculée de la manière suivante :

$$T_{op_0} = T_{co} \times T_{cb}^{-1} \times T_{pb}$$

La seconde étape consiste à donner l'expression de la pose de l'extrémité de la sonde à tout moment, la table cartésienne étant équipée de capteurs de position. La transformation relative entre la position de départ de la base de la sonde et sa position actuelle est connue (T_{pp_0}). La position de l'extremité de la sonde en tout temps est donc exprimée par la relation suivante :

$$T_{op}(t) = T_{op_0} \times T_{p_0p}(t)$$

S'il s'agit d'exprimer la position et l'orientation de l'extrémité de la sonde par rapport à la caméra, nous appliquons des formules similaires dans lesquelles nous retirons le terme T_{co}, ce qui peut directement s'écrire :

$$T_{cp}(t) = T_{cb}^{-1} \times T_{pb} \times T_{p_0p}(t)$$

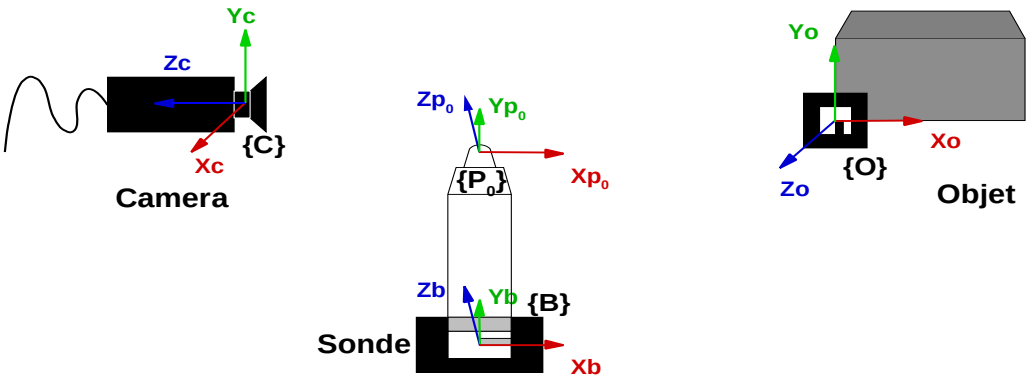


FIGURE 4.32 – Transformations entre les repères associés à la caméra, la sonde et l'objet.

Processus de réalité diminuée

La réalité diminuée désigne une application dans laquelle des objets ou des parties de la scène réelle sont retirées ou remplacées lors de leur restitution. Nous nous sommes inspirés du travail de [YOKOKOHJI et al. 1996] pour mettre au point notre processus de réalité diminué. L'idée est de peindre la table XYZ dans une couleur facile à segmenter par rapport à l'environnement. Il s'agit d'une variante de la technique de *chroma-keying* utilisée au cinéma qui consiste à filmer des acteurs sur un fond uni en vue de les intégrer dans un décor.

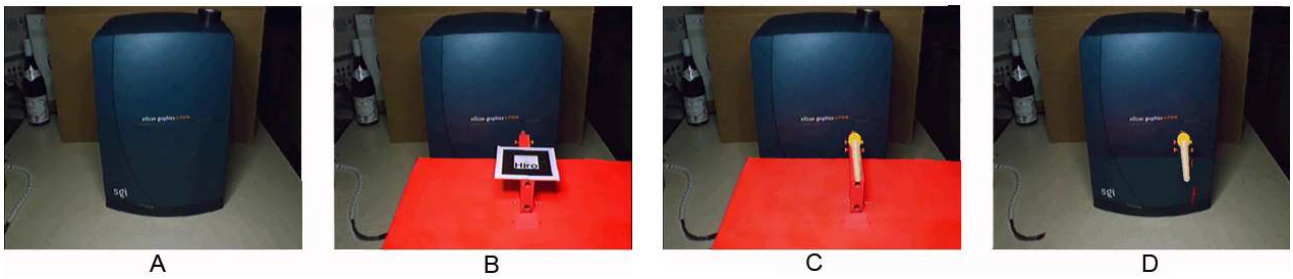


FIGURE 4.33 – Étapes du processus de réalité diminuée

La figure 4.33 représente les étapes de ce processus de réalité diminuée tel que nous l'avons décrit dans [BAYART et al. 2007] :

1. une image de l'arrière-plan est enregistrée en temps que référence (A) ;
2. la sonde haptique est placée dans la scène (B) ;
3. le marqueur permet calibrer le système pour recaler l'objet virtuel sur la sonde réelle avant de disparaître (C) ;
4. le processus de réalité diminuée détecte la table cartésienne et la dissimule en utilisant des parties de l'image de référence à la place (D).

Ce processus n'est opérationnel que tant que la caméra est immobile et que la couleur choisie pour la table XYZ et la sonde soit suffisemment contrastée par rapport à la scène. Il n'est pas exempt de défauts. Ainsi, l'inclusion de la table et de sa sonde interfèrent avec la colorimétrie générale de la scène. Aussi, nous avons eu recours à une technique de mixage à de multiples résolution [ILIE et al. 2005] pour lisser la transition entre l'image réelle et l'image de référence. En revanche, nous n'avons pas traité le problème de l'ombre projetée par la table sur la scène.

4.3.3 Application des concepts

4.3.3.1 Exploration haptique d'un objet mixte

Dans cette première application, nous nous sommes intéressé à l'exploration haptique d'un objet mixte. C'est à dire que nous souhaitons pouvoir interagir à la fois sur un objet réel et sa contrepartie virtuelle. Pour cela, nous avons proposé des fonctions de mélange d'information haptique inspirées des équations utilisées pour gérer la transparence en informatique graphique (connues sous le terme d'α-*blending*) dont ou pourra retrouver le détail dans [BAYART et al. 2007].

Nous devons traiter essentiellement deux cas dans le cadre d'une exploration haptique mixte :

— Les zones d'exploration de l'objet physique ont une géométrie connue ;

— Les zones d'exploration de l'objet physique ont une géométrie inconnue.

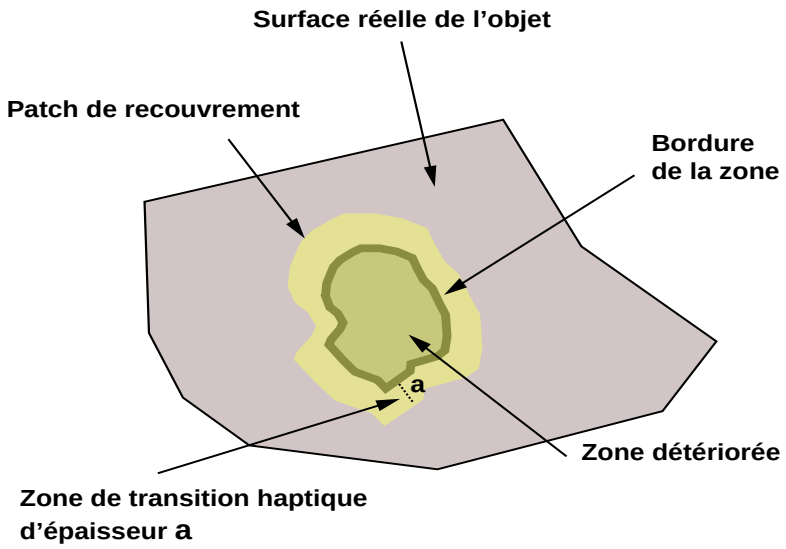


FIGURE 4.34 – Placement d'un *patch* recouvrant un défaut d'un objet

Zones d'exploration à géométrie connue

Dans le premier cas, il s'agit de défauts connus de l'objet physique. Nous proposons ici un objet virtuel qui compense les défauts de l'objet physique. Nous souhaitons alors assurer la transition en douceur entre les forces réelles (F_r) mesurées sur l'objet et les forces virtuelles (F_v) retournées par la simulation haptique. C'est à ce niveau que le coupleur de notre système intervient. Dans le cas où nous plaçons un *patch* sur la zone pour recouvrir le défaut, nous aurons non seulement une zone dans laquelle la géométrie du patch est décrite, mais aussi une zone de transition haptique autour de ce dernier (voir figure 4.34). Les forces retournées à l'utilisateur (F_u) sont calculées de la manière suivante :

$$F_u = (1 - \alpha)F_r + \alpha F_v$$

La valeur de α dépend de la position de la sonde par rapport à la zone d'exploration :

— A l'extérieur, sur la surface réelle de l'objet, α vaut 0 ;

— A l'intérieur, dans la zone détériorée, α vaut 1 ;

— Dans la zone de transition, α varie progressivement de 0 à 1 en fonction de sa proximité avec la zone détériorée.

Le coupleur de notre système se charge alors de calculer F_u et de la retourner à l'utilisateur.

Nous avons mené une expérience dans laquelle notre table cartésienne explore un objet haptique mixte (figure 4.37 page 186). La séquence de vues nous montre un objet pourvu d'un défaut sur lequel nous avons appliqué un patch (en jaune). La sonde explore en premier lieu l'objet réel et passe graduellement sur le patch où une combinaison de forces réelles et virtuelles est délivrée à l'opérateur.

La figure 4.36(a) page suivante montre l'évolution de la position de la sonde qui vient en contact avec l'objet (en $z = -16$mm), puis qui suit l'objet jusqu'à passer sur la zone défectueuse. Le patch comblant le trou, la sonde passe sur l'objet comme si le défaut n'existait pas. La figure 4.36(b) présente l'évolution des forces retournées à l'utilisateur ainsi que la valeur du paramètre α. Ce qui est intéressant à relever est que les forces retournées conservent une certaine continuité alors que, sans que l'utilisateur s'en rende compte, le système bascule progressivement d'une restitution de forces réelles à des forces purement virtuelles.

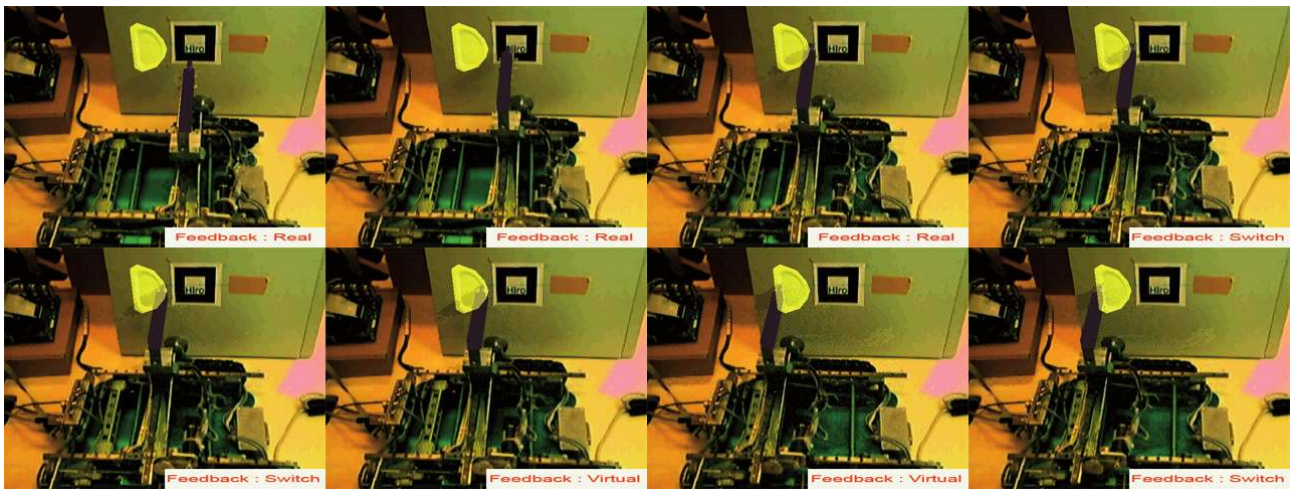
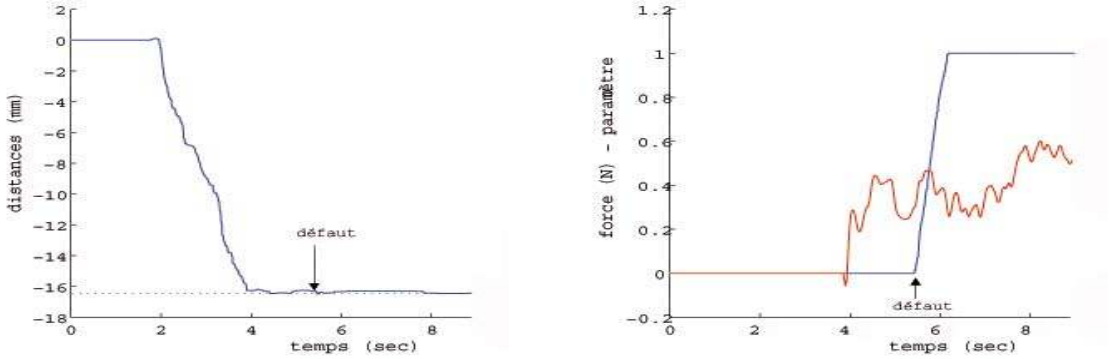


FIGURE 4.35 – Exploration d'un objet mixte à l'aide de la sonde



(a) Position mesurée par la sonde

(b) Force retournée (en rouge) et valeur de α (en bleu)

FIGURE 4.36 – Paramètres enregistrés lors de l'exploration d'un objet haptique mixte

Zones d'exploration à géométrie inconnue

Dans le cas où les zones d'exploration sont à géométrie inconnue, notre méthode précédente ne peut s'appliquer. Cela se produit quand l'objet est, par exemple, pourvu d'un défaut qui n'avait pas été modélisé. Dans ce cas, appliquer la stratégie précédente n'est plus possible car les positions réelles (P_r) et virtuelles (P_v) de la sonde ne coïncident plus. Lorsqu'une différence est détectée entre les représentations virtuelles et réelles, l'utilisateur a le choix de continuer le parcours en réel ou en virtuel. Dans ce cas, les deux sondes sont découplées, l'une restant en position et l'autre évoluant indépendemment. Le chemin de la sonde libre est enregistré de manière à pouvoir le rejouer. A tout moment, l'utilisateur peut choisir de basculer entre les deux modes. A chaque fois, l'exploration reprend de l'endroit où la sonde s'était arrêtée. Il est ensuite possible de montrer à l'utilisateur, en post-traitement, quelles sont les différences entre objets réel et virtuel, ce qui fait de cette application un système de télédiagnostique par une exploration haptique. Ceci est traduit par l'algorithme 6 page ci-contre.

La figure 4.37 page 186 illustre les étapes d'un scénario d'exploration. Ici, l'objet réel (en gris) présente un défaut (trou) positionné à gauche tandis que la pièce virtuelle (en bleu) présente un défaut (bosse) positionné à droite. La position réelle P_r de la sonde est représentée en gris, tandis que la position virtuelle P_v est représentée en orange. Les étapes de l'exploration sont les suivantes :

– **A.** Le mode d'exploration est réel, la sonde active est P_r. La position des deux sondes coïncide et l'opérateur perçoit les forces réelles ;

Algorithme 6 : Bascule entre interaction réelle et virtuelle

```
pour chaque pas de la simulation haptique faire
    transmettre P_m à la table cartésienne et récupérer P_r et F_r
    intégrer P_m dans la simulation virtuelle et calculer P_v
    si exploration en mode REEL alors
        si P_r ≠ P_v alors
            empiler P_r sur R_{P_r}, figer P_v, avertir l'utilisateur
        fin
        appliquer F_r
    sinon si exploration en mode VIRTUEL alors
        si P_r ≠ P_v alors
            empilerP_v sur R_{P_v}, figer P_r, avertir l'utilisateur
        fin
        appliquer F_v
    sinon
        // exploration en mode BASCULE
        si passage du mode REEL au VIRTUEL alors
            dépiler totalement R_{P_r} pour restaurer P_r
            appliquer F(P_m, P_r)
        sinon
            dépiler totalement R_{P_v} pour restaurer P_v
            appliquer F(P_m, P_v)
        fin
    fin
fin
```

– **B.** une erreur de position entre les deux sondes est détectée en raison de la présence d'un défaut dans la pièce. En effet, P_v ne peut suivre P_r puisque le défaut n'est pas présent sur la pièce virtuelle. En ce cas, la sonde P_v est figée et P_r poursuit l'exploration. Les forces retournées à l'opérateur sont les valeurs réelles ;

– **C.** l'opérateur continue l'exploration du trou dans la pièce réelle ;

– **D.** l'opérateur décide de basculer du mode réel au mode virtuel. La sonde réelle revient alors en arrière jusqu'à la position de la sonde virtuelle en rejouant dans l'ordre inverse les déplacements de la sonde réelle ;

– **E.** la sonde virtuelle P_v est à présent active et la sonde P_r suit son mouvement tant que les pièces réelles et virtuelles concordent. L'opérateur perçoit à présent les forces virtuelles ;

– **F.** c'est à présent au tour de la sonde P_r d'être figée pendant que les positions de la sonde P_v sont enregistrées et que l'utilisateur explore le défaut de l'objet virtuel.

La figure 4.38 page 187 illustre l'évolution de la position de l'opérateur P_m et des sondes réelle P_r et virtuelle P_v suivant les directions normales et tangentielle à la trajectoire au cours de ce scénario. Nous observons ainsi que lors de la rencontre du premier défaut, la sonde virtuelle reste figée (en pointillés rouge) alors que la tendance s'inverse lors de la rencontre suivante.

Cette application illustre les stratégies de couplage entre le réel et le virtuel lors de l'exploration d'un objet haptique mixte. Nous sommes dans le cas de l'haptique augmentée, c'est à dire qu'une information haptique virtuelle est superposée à une information haptique réelle. Nous allons voir dans la prochaine application une démonstration du concept d'augmentation haptique.

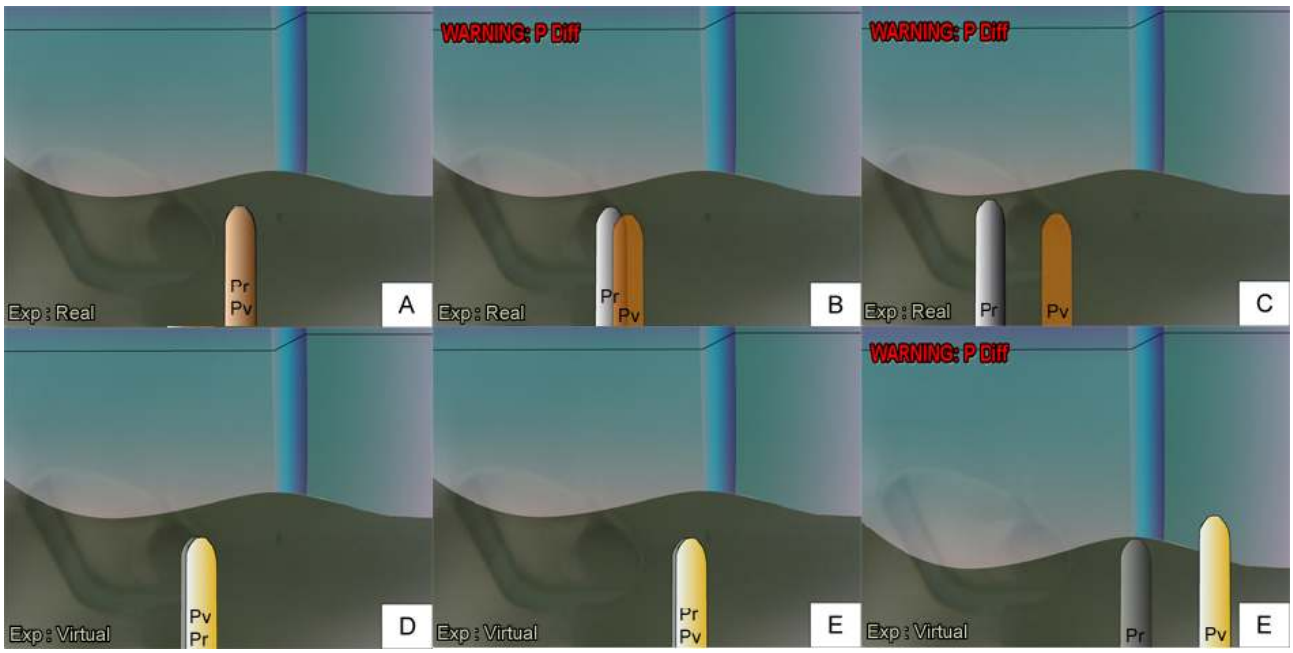


FIGURE 4.37 – Exploration de zones à la géométrie inconnue

4.3.3.2 Application de peinture virtuelle

Dans cette application, le coupleur haptique modélise le comportement d'un outil virtuel qui est différent de celui de la sonde de la table cartésienne lorsque cette dernière vient en contact avec un objet réel. En conséquence, les informations haptique et visuelle retournées à l'opérateur doivent être ajustées. Le coupleur $\mathcal{C}$ considère donc un modèle d'outil virtuel et permet :

1. de modifier le modèle visuel virtuel M_V de l'outil afin de prendre en compte les forces réelles mesurées :
$$M'_V = \mathcal{C}_1(F_R)$$
2. de modifier les forces réelles en fonction des limitations du modèle :
$$F'_R = \mathcal{C}_2(M'_V)$$
3. de modifier le résultat visuel de l'interaction I_v en fonction du modèle de l'outil virtuel et des forces réelles :
$$I_v = \mathcal{C}_3(F_R, M_V)$$

Nous mettons en oeuvre cette méthode sur une application qui applique de la peinture virtuelle sur un objet réel à l'aide de trois outils virtuels : un crayon, une brosse éponge et un pinceau de calligraphie (figure 4.39 page suivante).

Dans le cadre de la première étape de couplage $\mathcal{C}_1$, nous calculons les déformations des modèles en fonction des forces réelles appliquées :

— le crayon étant rigide, aucune déformation ne sera appliquée ;

— la brosse éponge se déforme en largeur w et en hauteur h en fonction de la force réelle appliquée. Ceci sera lié à des coefficients de raideur k_1 et k_2 dans les deux directions. L'équation de déformation qui traduit un écrasement en hauteur et un élargissement s'écrit alors :

$$\begin{cases} \Delta h' &= h(1 - k_1 F_R) \\ \Delta w' &= w(1 + k_2 F_R) \end{cases}$$

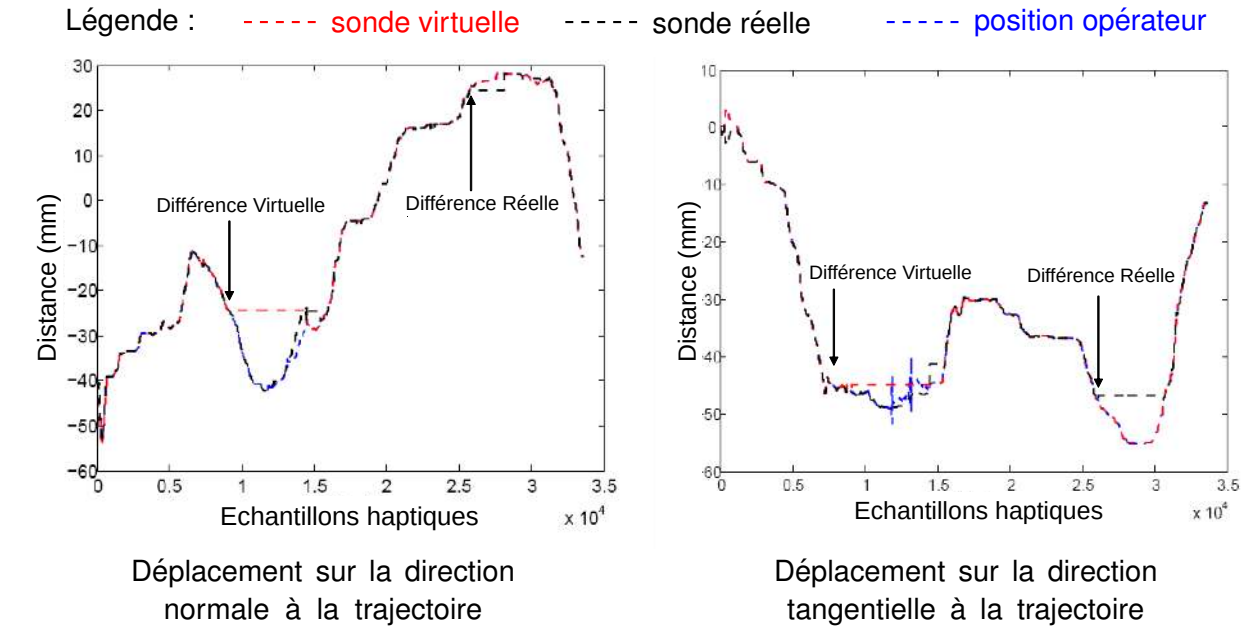


FIGURE 4.38 – Evolution des positions de l'opérateur et des deux sondes réelles et virtuelles

FIGURE 4.39 – Modèles virtuels des trois outils

— le pinceau de calligraphie utilise un modèle de poutre pour lequel nous considérons une extrémité fixe et une autre sur laquelle la force réelle est appliquée. La déformation étant supposée élastique, nous prenons en compte la longueur L du pinceau, son module de Young E, son inertie I et la surface normale de contact S. Les déformations longitudinales et latérales ΔL et f sous les forces normale F_n et latérale F_l peuvent s'écrire :

$$\left\{ \begin{array}{rcl} f & = & \frac{F_l L^3}{3EI} \\ \Delta L & = & \frac{F_n L}{ES} \end{array} \right.$$

A la deuxième étape de couplage $\mathcal{C}_2$, les contraintes des modèles virtuels sont prises en compte pour modifier les forces restituées :

— dans le cas du crayon, les forces ne subissent aucune modification ;

— dans le cas de la brosse éponge et du pinceau de calligraphie, la déformation de leur géométrie est limitée. La force maximale est envoyée lorsqu'une limite géométrique est atteinte. Dans les autres cas, les forces renvoyées à l'utilisateur correspondent au forces réelles modifiées par le modèle virtuel.

L'application de peinture dépend du résultat de l'interaction entre l'outil virtuel et l'objet réel. La taille des coups de pinceau est proportionnelle à l'intensité des forces réelles tandis que leur orientation et leur forme dépend de la géométrie de l'outil. Cela est calculé lors de la troisième étape de couplage $\mathcal{C}_3$. LA figure 4.40 page suivante présente les résultats de l'application de peinture sur des surfaces virtuelles, puis réelles. Pour le pinceau, l'intensité de la couleur dépend de la force exercée

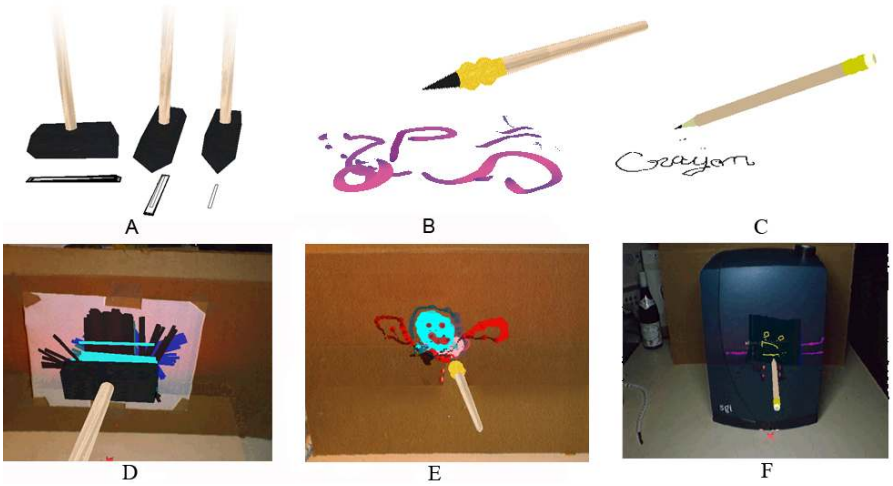


FIGURE 4.40 – Application de peinture à l'aide des trois outils sur surfaces virtuelles (A, B, C) et réelles (D, E, F).

sur ce dernier. Pour la brosse éponge, la géométrie du tracé sera affecté par les déformations de l'outil.

Cet exemple montre comment modifier les retours des modalités haptiques et visuelles, les deux étant inter-dépendantes. L'avantage de cette méthode est qu'aucun modèle virtuel de l'objet réel n'est nécessaire. Il est donc possible, avec le même dispositif, d'interagir avec différents types d'objet. L'application présentée démontre le concept d'augmentation haptique où la force retournée à l'utilisateur est modulée en fonction de modificateurs haptiques. La combinaison des modalités est également intéressante, puisqu'ici le processus de réalité diminuée visuel fonctionne en parallèle avec le processus d'augmentation haptique.

4.3.4 Liens avec les architectures logicielles

Ces applications multi-modales offrent une complexité intéressante du point de vue des architectures logicielles. En particulier, elles ont influencé la subdivision en composants élémentaires du langage MIRELA, suite à l'utilisation en parallèle de plusieurs boucles de rendu, une visuelle, une haptique et une dernière mécanique pour commander la table cartésienne. Les échelles de temps employées sont aussi disparates. En effet, pour simuler efficacement les contacts rigides, la boucle de rendu haptique doit fonctionner à une fréquence de 1kHz. En revanche, la caméra était cadencée pour une acquisition à 12 images par seconde.

La dernière application, celle utilisant de la peinture virtuelle, a également été prise en exemple pour montrer les limites de l'analysabilité de modèles MIRELA [DIDIER et al. 2009a]. En effet, les échelles de temps très différentes et l'ensemble des capteurs et boucles de rendu mobilisées rendent ce système difficile à analyser d'un seul tenant. Cela a motivé la recherche de techniques d'abstraction de parties du système afin de parvenir à l'analyser et détecter des comportements potentiellement néfastes.

4.4 Résumé des contributions du chapitre

Ce chapitre a traité des exigences fonctionnelles des systèmes de réalité augmentée. Nous les avons passées en revue, au travers de quelques applications développées.

Dans le cadre de la problématique de la localisation du système par rapport à son espace de travail, nous avons proposé un capteur hybride fonctionnant suivant un schéma de suppléance subdivisant les

appareils de mesure en deux classes équivalentes, la caméra d'une part et le GPS et la centrale inertielle d'autre part. Cette application a aussi été l'occasion de tester l'architecture logicielle proposée dans le *framework* ARCS, en raison de sa modélisation des applications à l'aide d'une machine à état adaptée ici aux modes de fontionnement de notre capteur hybride.

Pour être en mesure d'assister l'utilisateur dans sa tâche, le système doit pouvoir interpréter les phénomènes qui se déroulent dans son espace de travail. Dans ce cadre, nous avons traité deux axes complémentaires. Le premier suit les conséquences des actions des opérateurs sur les objets de la scène. Il s'est traduit par la mise au point d'une chaîne de traitement dont l'objectif est de détecter les opérations d'assemblage ou de désassemblage qui composent une procédure de maintenance. Le deuxième axe s'intéresse directement aux gestes de l'opérateur afin de les distinguer, puis de reconnaître les émotions exprimées au travers de ces derniers dans le but de créer un système de téléopération robotique sensible à l'état d'esprit de l'opérateur. Une partie de ces travaux s'est retrouvée implémentée dans le *framework* ARCS.

Enfin, nous nous sommes intéressés aux problématiques de l'interaction, conséquence directe de l'interprétation de la scène. En particulier, nous avons mené des incursions dans le domaine de la réalité augmentée haptique et exploré les concepts et contraintes associés aux interactions multimodales. Ces recherches ont alimenté en retour les réflexions menées autour des contraintes temporelles dans les applications de réalité augmentée.

Publications associées à ce chapitre

Localisation et recalage en réalité augmentée

ABABSA, F. et al. [2012]. « Outdoor augmented reality system for geological applications ». In : *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, p. 416–421. DOI : 10.1109/AIM.2012.6265927

ZENDJEBIL, I. et al. [2011]. « Large Scale Localization - For Mobile Outdoor Augmented Reality Applications ». In : *International Conference on Computer Vision Theory and Applications (VISAPP 2011)*. Vilamoura, Algarve, Portugal, p. 492–501

ZENDJEBIL, I. et al. [2010]. « A GPS-IMU-camera modelization and calibration for 3D localization dedicated to outdoor mobile applications ». In : *Control Automation and Systems (ICCAS), 2010 International Conference on*. IEEE, p. 1580–1585

ABABSA, F. et al. [2008a]. « Markerless Vision-Based Tracking of Partially Known 3D Scenes for Outdoor Augmented Reality Applications ». In : *ISVC (1)*, p. 498–507

ZENDJEBIL, I. et al. [2008b]. « On the Hybrid Aid-Localization for Outdoor Augmented Reality Applications ». In : *ACM Symposium on Virtual Reality Software and Technology*. Bordeaux, France

ZENDJEBIL, I. et al. [2008d]. « Outdoor Augmented Reality : State of the Art and Issues ». In : *Virtual Reality International Conference*, p. 177–187

Interprétation de la scène

Suivi des étapes d'une procédure de maintenance industrielle

RUKUBAYIHUNGA, A. et al. [2016b]. « Towards assembly steps recognition in augmented reality ». In : *Proceedings of the 2016 Virtual Reality International Conference*. ACM, p. 17

RUKUBAYIHUNGA, A. et al. [2016a]. « Real Time Noise Reduction to Identify Motion Parameters in AR Maintenance Scenario ». In : *Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on*. IEEE, p. 27–30

RUKUBAYIHUNGA, A. et al. [2015]. « Reprojection error as a new metric to detect assembly/disassembly maintenance tasks ». In : *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*. IEEE, p. 513–518

Reconnaissance des gestes et des émotions pour la téléopération

AJILI, I. et al. [2018b]. « Relevant LMA Features for Human Motion Recognition ». In : *ICIAP 2018 : International Conference on Image Analysis and Processing, Paris, France, (Oct 29-30, 2018)*. T. 12. 10, p. 2899

AJILI, I. et al. [2018a]. « An Efficient Motion Recognition System Based on LMA Technique and a Discrete Hidden Markov Model ». In : *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 12.9, p. 707–713. ISSN : eISSN :1307-6892

AJILI, I. et al. [2017a]. « Gesture recognition for humanoid robot teleoperation ». In : *26th IEEE International Symposium on Robot and Human Interactive Communication (RO'MAN 2017)*. Lisbon, Portugal

AJILI, I. et al. [2017b]. « Robust human action recognition system using Laban Movement Analysis ». In : *Knowledge-Based and Intelligent Information & Engineering Systems : Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*. P. 554–563. DOI : 10.1016/j.procs.2017.08.168. URL : https://doi.org/10.1016/j.procs.2017.08.168

AJILI, I. et al. [2016]. « Gesture recognition for robot teleoperation ». In : *11ème journées de l'AFRV*

Interaction multi-modale

BAYART, B. et al. [2008]. « Force Feedback Virtual Painting on Real Objects : A Paradigm of Augmented Reality Haptics ». In : *Haptics : Perception, Devices and Scenarios (Proceedings of EuroHaptics 2008)*. T. 5024/2008. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 776–785

BAYART, B. et al. [2007]. « Visuo-Haptic Blending Applied to a Tele-Touch-Diagnosis Application ». In : *HCI (14)*. Beijing, China, p. 617–626

Conclusion et perspectives

Bilan

La première communication que j'ai publiée en tant que premier auteur en 2003 s'intitulait « Architecture Logicielle Modulaire adaptée au Recalage Dynamique dans un Système de Réalité Augmentée en Vision Directe ». Il est intéressant de constater qu'elle portait modestement les germes des travaux entrepris depuis. Ceux-ci se sont naturellement prolongés au cours de ma thèse qui s'est terminée avec les bases de la proposition d'architecture donnée dans la première partie du chapitre 3.

Lors de ma première année en poste, j'ai eu l'occasion d'encadrer la dernière année de thèse de Benjamin Bayart (2006-2007). J'ai pu travailler sur les problématiques liées aux applications de réalité augmentée multi-modales. Cela a par la suite alimenté les travaux sur les contraintes temporelles, la synchronisation des tâches étant importante dans ce type de système.

En parallèle, mes premiers travaux ont vu leur application dans le projet RAXENV qui m'a donné l'occasion d'encadrer la thèse d'Imane Zendjebil (2006-2010). C'est au cours de cette dernière que nous avons mis au point un capteur hybride. Mes propositions architecturales ont non seulement été appliquées dans RAXENV, mais en plus ce projet nous a fourni de la matière pour travailler sur les problèmes de concurrence dans les applications de réalité augmentée.

Face aux problèmes de puissance embarqué des terminaux de l'époque, l'idée nous est venue d'explorer la problématique de la distribution des traitements. La thèse de Mehdi Chouiten (2009-2013) nous a permis de travailler dessus et d'apporter divers éléments de réponse.

Le contexte de la thèse CIFRE entre le laboratoire IBISC et Wassa, m'a permis de renouer avec le domaine applicatif de la maintenance industrielle. C'est dans cette lignée que nous avons proposé, avec Alia Rukubayihunga (2013-2016), un système de suivi des procédures de maintenance basé sur la détection des instants d'assemblage/désassemblage de pièces mécaniques. Durant cette thèse, nous avons aussi exploré quelques concepts architecturaux nouveaux, utilisant le navigateur web comme plate-forme de réalité augmentée.

Enfin, la thèse d'Insaf Ajili, dont la clôture est prévue en décembre de cette année, nous a permis de travailler sur une problématique voisine qui est celle de l'interprétation des actions de l'utilisateur, alliant ici reconnaissance des gestes et des émotions exprimées au travers de ces derniers.

Ce cheminement permet de retrouver les deux axes présentés au cours de cette habilitation avec un dosage équilibré entre architectures (exigences non fonctionnelles) d'une part et applications de réalité augmentée (exigences fonctionnelles) d'autre part.

Contributions liées aux architectures logicielles

Nous avons posé les bases, au chapitre 3, d'une architecture modulaire et reconfigurable. Cette dernière s'appuie sur le concept de la programmation orientée composant. Nous avons défini un modèle de composant et un modèle d'application pour développer des logiciels de réalité augmentée. Nous y avons ajouté les concepts de capacité étendue d'intégration et de famille de composants, exploités dans les travaux ultérieurs sur la distribution. Ces concepts ont été implémentés dans notre *framework* ARCS (pour *Augmented Reality Component System*) qui a connu deux évolutions dans sa

version C++ et une version javascript pouvant s'exécuter dans un navigateur web.

Au niveau de la distribution des traitements, nous avons proposé un protocole *ad-hoc* s'appuyant sur les familles de composants précédemment évoquées. Nous avons aussi exploré les liens existants entre applications de réalité augmentée et *webservices*. Sur la base de la méthodologie *Software Performance Engineering* que nous avons adapté à notre contexte, nous avons pu édicter des recommandations sur la manière dont les traitements doivent être distribués. Nous avons ensuite montré comment nous pouvions intégrer ARCS dans une application distribuée de pilotage de robot sous-marin. Ces travaux restent encore ouverts, en particuler en ce qui concerne les liens avec les *webservices* qui n'ont pu être totalement explorés.

Enfin, nous avons abordé le volet de la concurrence et des contraintes temporelles. Nous avons proposé un algorithme de détection automatique des composants à monitorer, c'est à dire exposés à des situations de compétition, afin de pouvoir contrôler leur accès à l'aide d'exclusions mutuelles. Au niveau des contraintes temporelles, nous avons proposé un langage de haut niveau, MIRELA (*Mixed Reality Language*) pour décrire une application de réalité mixte avant de la traduire dans une spécification formelle à base d'automates temporisés afin d'analyser les possibles comportements néfastes pour les corriger avant implémentation Nous avons aussi exploré des pistes pour garantir la conservation des propriétés analysées lors de l'implémentation directe du modèle à base d'automates. Les outils de *model checking* offrent une perspective intéressante en ce sens qu'ils permettent une vérification exhaustive du comportement des applications de réalité mixte. Toutefois, en raison de l'espace d'état à parcourir, il faut recourir à des techniques d'abstraction et d'analyse par morceaux des modèles qui sont encore à affiner.

A propos de la localisation, de l'interprétation et de l'interaction

Le chapitre 4 nous a permis d'exposer nos travaux en matière d'exigences fonctionnelles pour la réalité augmentée. Nous avons commencé par travailler sur la localisation où nous avons proposé un capteur hybride opérant suivant un schéma de suppléance, les appareils de mesure élémentaires étant divisé en deux classes d'équivalence : la caméra d'une part, le récepteur GPS et la centrale inertielle d'autre part. Nous avons apporté des contributions dans la manière de calibrer un tel dispositif et de l'opérer en vue d'estimer en permanence la pose de la caméra par rapport à son environnement. Ces travaux ont été appliqués dans le cadre du projet RAXENV.

Nous nous sommes ensuite intéressés à la problématique de l'interprétation de la scène afin que le système de réalité augmentée puisse fournir une assistance contextualisée à l'opérateur. Une première série de travaux s'est focalisée sur le suivi des procédures de maintenance industrielle considérées comme majoritairement composées de gammes de montage et de démontage. Nous avons montré comment nous pouvions détecter les instants d'assemblage ou de désassemblage. Nous avons étudié deux cas. Dans le premier, nous considérons les pièces rigidement liées et nous avons montré comment nous pouvions exploiter l'erreur de reprojection pour détecter les instants d'assemblage/désassemblage. Dans ce cadre, nous avons proposé deux méthodes de détection de rupture, KSUM et KSUMratio qui offrent un compromis intéressant entre temps de calcul et délai de détection. Dans le deuxième cas, nous considérons des pièces assemblées par une liaison mécanique et avons montré comment nous pouvions suivre l'évolution des degrés de liberté entre les pièces.

Une deuxième série de travaux porte sur l'interprétation du mouvement et, au travers de ce dernier, de l'état émotionnel de l'opérateur. Nous avons proposé une méthode de reconnaissance de geste s'appuyant sur le modèle de Markov caché et l'analyse du mouvement de Laban. Ses performances ont été comparées avec celles d'autres méthodes de la littérature en utilisant des bases de gestes publiques. Elles leur sont similaires voire légèrement supérieures. Nous avons créé notre propre base de

gestes expressifs afin de reconnaître les émotions. Nous avons testé quatre méthodes d'apprentissage automatique pour arriver à la conclusion que les forêts d'arbres décisionnels sont les plus adaptés à notre problématiques. Enfin, nous avons cherché à optimiser notre classifieur en déterminant quelles sont les caractéristiques à conserver dans le vecteur descripteur que nous avons associé au mouvement. Nous avons aussi étudié quelles caractéristiques du mouvement nous associons aux émotions en tant qu'êtres humains et avons constaté que notre classifieur tend à se reposer sur les mêmes.

Enfin, nous avons relaté notre brève incursion dans le domaine de l'interaction multi-modale. Nous avons proposé des applications permettant d'explorer les concepts d'haptique augmentée et d'augmentation haptique. Dans ce cadre, nous avons conçu, en nous appuyant sur un système téléopéré, une application de télédiagnostic fonctionnant par exploration haptique tout en ayant un retour visuel et une application permettant d'appliquer de la peinture virtuelle sur n'importe quel objet distant.

Perspectives

Avant d'aborder les perspectives proprement dites, il convient de regarder l'évolution des méthodes et des techniques dans le domaine du grand public. Par exemple, la plate-forme matérielle du projet RAXENV en 2008 (voir section 4.1.1 page 135) contenant une caméra, un récepteur GPS, une centrale inertielle et une unité de calcul pourvue d'un écran existe maintenant sous la forme des *smartphones* actuels.

Au niveau logiciel, les applications grand public commencent à arriver. Plus précisément, les GAFAM (Google, Apple, Facebook, Amazon et Microsoft) semblent s'y intéresser de très près, ce qui indique que le domaine de la réalité augmentée s'approche d'un certain niveau de maturité technologique. Pour s'en convaincre, voici quelques anonces récentes sur une période datant de 18 mois :

— **juin 2017,** Apple lance officiellement ARkit, sa plate-forme pour produire des applications de réalité augmentée sur iOS. Depuis juin 2018, c'est la version 2 qui a été annoncée ;

— **octobre 2017,** Microsoft annonce le Windows Mixed Reality OS, destiné à accompagner son produit HoloLens ;

— **novembre 2017,** Amazon lance Sumerian, son logiciel de création et d'édition de contenu pour les réalités virtuelle et augmentée. Ce dernier a la particularité de fonctionner avec AWS, la solution *cloud* d'Amazon ;

— **décembre 2017,** Facebook annonce AR Studio, un logiciel dédié à la création de contenu en réalité augmentée. Ce dernier peut-être partagé au travers du réseau social du même nom ;

— **mars 2018,** Google publie ARCore, un kit de développement fonctionnant sur Android pour créer des applications de réalité augmentée ;

— **avril 2018,** Mozilla annonce Firefox Reality, une version particulière de son navigateur phare pour être exploité au travers des dispositifs de réalité virtuelle et augmentée.

Cette liste, loin d'être exhaustive, annonce les bouleversements prochains que va connaître le domaine, à commencer sans doute par une future bataille des standards. Un plan de recherches doit donc tenir compte de ces évènements et de la force de frappe que pourront investir ces entreprises dans les domaines où elles commencent à faire des annonces. En particulier, il convient de bien choisir les pistes de recherche à explorer sous peine d'être très vite dépassé par les évènements.

Nous conservons néanmoins la même structure pour le plan de travail, à savoir deux axes d'exploration, un étant porté sur les architectures et l'autre sur les applications.

Perspectives sur les architectures logicielles

Au niveau des architectures logicielles, nous avons des perspectives à court terme que nous souhaiterions clôturer. Ainsi, les liens entre applications de réalité augmentée et *webservices* nécessitent d'être explorés davantage. La nature ouverte et interopérable des technologies web en font un atout intéressant face aux systèmes proposés par les GAFAM.

Sur le plus long terme, nous souhaiterions explorer deux pistes différentes :

— la première concerne la mobilité du code ;

— la deuxième touche aux architectures de calcul hétérogènes.

Mobilité du code

Cette perspective est la conséquence directe des travaux entrepris sur la distribution. La mobilité du code implique que les codes de calcul peuvent en quelque sorte « voyager » sur le réseau afin de se réorganiser en fonction du contexte de l'application. Cette propriété est souvent associée aux systèmes multi-agents mais ne nécessite pas forcément ces derniers pour être mise en oeuvre. Nous pouvons tirer partie des règles de distribution que nous avons déjà déterminé pour faire voyager le code aux endroits appropriés.

Les questions à résoudre dans ce contexte sont :

— Comment assurer une organisation optimale des codes mobiles ?

— Comment offrir des garanties de sécurité liées à l'exécution de ces codes ?

— Comment garantir leur bon fonctionnement ?

Architectures de calcul hétérogènes

Cet axe touche à la distribution mais aussi à la concurrence. En effet, dans un réseau maillé de périphériques avec différentes capacités de calcul, comment assurer la répartition de ces derniers ? De plus, la plupart des systèmes actuels embarquent à présent deux processeurs : le processeur central (ou CPU) au fonctionnement généraliste et le processeur graphique (ou GPU) au fonctionnement spécialisé. Toutefois, depuis un peu plus de 10 ans, la barrière entre les deux n'a cessé de s'estomper. La puissance de calcul offerte par les GPU est également très intéressante en matière de parallélisme. En effet, si nous prenons l'exemple de l'architecture Pascal (2016) que NVidia utilise pour ses cartes graphique GeForce GTX 1080, le processeur graphique ne renferme pas moins de 2560 coeurs pouvant faire des calculs en parallèle.

Dans ce contexte, il convient de se demander ce que nous pouvons faire en matière d'architecture de calcul hétérogènes fonctionnant sur une grande variété de CPU et de GPU.

Perspectives sur les applications

Ici aussi, nous retrouvons des perspectives à court et moyen terme. Les perspectives à court terme concernent les travaux entrepris sur le suivi des procédures de maintenance et l'interprétation du geste. Dans les deux cas, seules des briques ont été construites. Les solutions complètes restent à réaliser.

En ce qui concerne les perspectives sur le long terme, elles concernent essentiellement les problématiques d'interprétation de la scène et d'interaction. En effet, la problématique de la localisation qui a été pendant des années l'un des verrous majeurs en réalité augmentée est en train de perdre de son importance, en partie parce que ces technologies ont évolué et sont plus matures.

Nous nous proposons d'explorer deux sous-axes :

— l'interprétation sémantique de la scène ;

— les liens entre systèmes cyber-physique et réalité augmentée.

Interprétation sémantique de la scène

La problématique est ici de se demander comment donner du sens à une scène observée par un processus de vision par ordinateur ? Dans le cadre d'une coopération homme-machine, nous avons vu que l'interprétation de la scène permet au système de fournir une assistance contextualisée à l'opérateur. Nous nous proposons d'étendre ces travaux en exploitant les techniques d'apprentissage automatique. Cette tendance a d'ailleurs déjà été amorcée dans nos précédents travaux.

Systèmes cyber-physique et réalité augmentée

Derrière ce mot à la mode se cache un concept ancien : les systèmes cyber-physiques sont des systèmes informatiques capables d'interagir avec les processus physiques. Ainsi, la table cartésienne que nous avons présenté au chapitre 4, le robot sous-marin présenté au chapitre 3 ainsi que le robot NAO que nous souhaitons téléopérer par le geste font, par définition, partie de cette catégorie de systèmes.

Ils offrent de nouvelles dimensions à l'interaction. En effet, dans un système de réalité augmentée, plusieurs espaces cohabitent. Il y a d'un côté l'espace réel, avec lequel nous pouvons interagir physiquement et de l'autre côté l'espace virtuel. Dans un système de réalité augmentée classique, il est possible d'interagir directement avec l'espace réel et l'espace virtuel. En modifiant le premier, il est possible d'interagir avec le second. Toutefois, les actions dans l'espace virtuel n'influent que rarement sur l'espace réel. Les systèmes cyber-physiques, de part leur nature hybride, viennent combler cet aspect manquant de ces interactions multi-latérales entre espaces réels et virtuels. Il convient donc d'explorer plus profondément les couplages entre systèmes de réalité augmentée et systèmes cyber-physiques.

Bibliographie

Atol les Opticiens lance l'application pour essayer ses lunettes grâce à la réalité augmentée (2011). http://www.servicesmobiles.fr/atol-les-opticiens-lance-lapplication-pour-essayer-ses-lunettes-grace-a-la-realite-augmentee-2647/ (cf. p. 59).

SIM – Coin3D 3D Graphics Development Kit (2014). http://www.coin3d.org (cf. p. 79).

(2017). https://www.qualcomm.com/products/snapdragon-845-mobile-platform (cf. p. 109).

Wikitude (2017). https://www.wikitude.com (cf. p. 59).

ABABSA, F., **DIDIER, J.-Y.**, MALLEM, M. et ROUSSEL, D. (2003). « Head motion prediction in augmented reality systems using monte carlo particle filters ». In : *Proceedings of the 13th International Conference on Artificial Reality and Telexixtance (ICAT 2003)*. The Virtual Reality Society of Japan. Tokyo (Japan), p. 83–88 (cf. p. 65).

ABAWI, D., DORNER, R., HALLER, M. et ZAUNER, J. (2004). « Efficient Mixed Reality Application Development ». In : *1st European Conference on Visual Media Production (CVMP)*. London : IEEE, p. 289–294 (cf. p. 70, 73).

ABADI, M., FLANAGAN, C. et FREUND, S. N. (2006). « Types for safe locking : Static race detection for Java ». In : *ACM Transactions on Programming Languages and Systems (TOPLAS)* 28.2, p. 207–255 (cf. p. 111).

ABABSA, F.-E. et MALLEM, M. (2007). « Hybrid three-dimensional camera pose estimation using particle filter sensor fusion ». In : *Advanced Robotics* 21.1-2, p. 165–181 (cf. p. 137).

ABABSA, F., **DIDIER, J.-Y.**, ZENDJEBIL, I. M. et MALLEM, M. (2008a). « Markerless Vision-Based Tracking of Partially Known 3D Scenes for Outdoor Augmented Reality Applications ». In : *ISVC (1)*, p. 498–507 (cf. p. 65, 189).

ABABSA, F., MAIDI, M., **DIDIER, J.-Y.** et MALLEM, M. (2008b). « Vision-Based Tracking for Mobile Augmented Reality ». In : *Multimedia Services in Intelligent Environments, Springer*. P. 297–326 (cf. p. 65).

ABABSA, F., ZENDJEBIL, I., DIDIER, J. Y., POUDEROUX, J. et VAIRON, J. (2012). « Outdoor augmented reality system for geological applications ». In : *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, p. 416–421. DOI : 10.1109/AIM.2012.6265927 (cf. p. 60, 119, 189).

AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P. et TVERSKY, B. (2003). « Designing Effective Step-by-step Assembly Instructions ». In : *ACM Trans. Graph.* 22.3, p. 828–837 (cf. p. 149).

AJILI, I., MALLEM, M. et **DIDIER, J.-Y.** (2016). « Gesture recognition for robot teleoperation ». In : *11ème journées de l'AFRV* (cf. p. 65, 166, 190).

AJILI, I., **DIDIER, J.-Y.** et MALLEM, M. (2017a). « Gesture recognition for humanoid robot teleoperation ». In : *26th IEEE International Symposium on Robot and Human Interactive Communication (RO'MAN 2017)*. Lisbon, Portugal (cf. p. 65, 168, 190).

AJILI, I., MALLEM, M. et **DIDIER, J.** (2017b). « Robust human action recognition system using Laban Movement Analysis ». In : *Knowledge-Based and Intelligent Information & Engineering Systems : Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*. P. 554–563. DOI : 10.1016/j.procs.2017.08.168. URL : https://doi.org/10.1016/j.procs.2017.08.168 (cf. p. 65, 190).

AJILI, I., MALLEM, M. et **DIDIER, J.-Y.** (2018a). « An Efficient Motion Recognition System Based on LMA Technique and a Discrete Hidden Markov Model ». In : *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 12.9, p. 707–713. ISSN : eISSN :1307-6892 (cf. p. 190).

— (2018b). « Relevant LMA Features for Human Motion Recognition ». In : *ICIAP 2018 : International Conference on Image Analysis and Processing, Paris, France, (Oct 29-30, 2018)*. T. 12. 10, p. 2899 (cf. p. 190).

ALUR, R. et DILL, D. L. (1994). « A theory of timed automata ». In : *Theoretical Computer Science* 126.2. ISSN 0304-3975, p. 183–235. URL : citeseer.ist.psu.edu/alur94theory.html (cf. p. 120).

ALWANI, A. A., CHAHIR, Y., GOUMIDI, D. E., MOLINA, M. et JOUEN, F. (2014). « 3D-Posture Recognition Using Joint Angle Representation ». In : *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Sous la dir. d'A. LAURENT, O. STRAUSS, B. BOUCHON-MEUNIER et R. R. YAGER. Cham : Springer International Publishing, p. 106–115. ISBN : 978-3-319-08855-6 (cf. p. 171).

ARCILE, J., CZACHÓRSKI, T., DEVILLERS, R. R., **DIDIER, J.**, KLAUDEL, H. et RATAJ, A. (2015a). « Modelling and Analysing Mixed Reality Applications ». In : *Man-Machine Interactions 4 - 4th International Conference on Man-Machine Interactions, ICMMI 2015, Kocierz Pass, Poland, October 6-9, 2015*, p. 3–17. DOI : 10.1007/978-3-319-23437-3_1. URL : http://dx.doi.org/10.1007/978-3-319-23437-3_1 (cf. p. 65, 133).

ARCILE, J., **DIDIER, J.-Y.**, KLAUDEL, H., DEVILLERS, R. R. et RATAJ, A. (2015b). « Indefinite waitings in MIRELA systems ». In : *Proceedings 4th International Workshop on Engineering Safety and Security Systems, ESSS 2015, Oslo, Norway, June 22, 2015*. P. 5–18. DOI : 10.4204/EPTCS.184.1. URL : http://dx.doi.org/10.4204/EPTCS.184.1 (cf. p. 65, 118, 123–125, 133).

ARON, M., SIMON, G. et BERGER, M.-O. (2007). « Use of inertial sensors to support video tracking ». In : *Computer Animation and Virtual Worlds* 18.1, p. 57–68 (cf. p. 137).

AZUMA, R. (1995). « Predictive Tracking for augmented reality ». Thèse de doct. Computer Science Department, University of North Carolina (cf. p. 54).

— (1997). « A survey of augmented reality ». In : *Presence : Teleoperators and Virtual Environments* 6.4, p. 355–385 (cf. p. 15, 50).

BACCA, J., BALDIRIS, S., FABREGAT, R., GRAF, S. et al. (2014). « Augmented reality trends in education : a systematic review of research and applications ». In : *Journal of Educational Technology & Society* 17.4, p. 133 (cf. p. 60).

BAKKEN, D. (2001). « Middleware ». In : *Encyclopedia of Distributed Computing* 11 (cf. p. 69).

BANERJEE, U., BLISS, B., MA, Z. et PETERSEN, P. (2006). « A theory of data race detection ». In : *Proceedings of the 2006 workshop on Parallel and distributed systems : testing and debugging*. ACM, p. 69–78 (cf. p. 111).

BARAKOVA, E. I. et LOURENS, T. (2010). « Expressing and interpreting emotional movements in social games with robots ». In : *Personal and ubiquitous computing* 14.5, p. 457–467 (cf. p. 171).

BARSOM, E. Z., GRAAFLAND, M. et SCHIJVEN, M. P. (2016). « Systematic review on the effectiveness of augmented reality applications in medical training ». In : *Surgical Endoscopy* 30.10, p. 4174–4183. ISSN : 1432-2218. DOI : 10.1007/s00464-016-4800-6. URL : https://doi.org/10.1007/s00464-016-4800-6 (cf. p. 58).

BASS, L., CLEMENTS, P. et KAZMAN, R. (2003). *Software Architecture in Practice*. 2nd edition. Addison-Wesley Professional. ISBN : 9780321154958 (cf. p. 67).

BASSEVILLE, M. et NIKIFOROV, I. V. (1993). *Detection of Abrupt Changes : Theory and Application*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc. (cf. p. 154).

BAUER, M., BRUEGGE, B., KLINKER, G., MACWILLIAMS, A., REICHER, T., RISS, S., SANDOR, C. et WAGNER, M. (2001). « Design of a Component-Based Augmented Reality Framework ». In : *Proceedings of the International Symposium on Augmented Reality (ISAR)*, p. 45–54. ISBN : 0-7695-1375-1. URL : http://citeseer.nj.nec.com/bauer01design.html (cf. p. 70).

BAUM, L. E., PETRIE, T., SOULES, G. et WEISS, N. (1970). « A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains ». In : *The annals of mathematical statistics* 41.1, p. 164–171 (cf. p. 169).

BAY, H., TUYTELAARS, T. et VAN GOOL, L. (2006). « Surf : Speeded up robust features ». In : *Computer vision–ECCV 2006*, p. 404–417 (cf. p. 52, 142).

BAYART, B. et KHEDDAR, A. (2006a). « Haptic augmented reality taxonomy : haptic enhancing and enhanced haptics ». In : *EuroHaptics 2006*, p. 641–644 (cf. p. 56).

— (2006b). « Haptic augmented reality taxonomy : haptic enhancing and enhanced haptics ». In : *Proceedings of EuroHaptics*. Citeseer, p. 641–644 (cf. p. 178, 179).

BAYART, B., DRIF, A., KHEDDAR, A. et **DIDIER, J.-Y.** (2007). « Visuo-Haptic Blending Applied to a Tele-Touch-Diagnosis Application ». In : *HCI (14)*. Beijing, China, p. 617–626 (cf. p. 65, 179, 182, 190).

BAYART, B., **DIDIER, J.-Y.** et KHEDDAR, A. (2008). « Force Feedback Virtual Painting on Real Objects : A Paradigm of Augmented Reality Haptics ». In : *Haptics : Perception, Devices and Scenarios (Proceedings of EuroHaptics 2008)*. T. 5024/2008. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 776–785 (cf. p. 65, 179, 190).

BIANCHI, G., KNOERLEIN, B., SZEKELY, G. et HARDERS, M. (2006). « High Precision Augmented Reality Haptics ». In : *EuroHaptics 2006*, p. 169–178 (cf. p. 56, 179).

BILLINGHURST, M., CLARK, A., LEE, G. et al. (2015). « A survey of augmented reality ». In : *Foundations and Trends® in Human–Computer Interaction* 8.2-3, p. 73–272 (cf. p. 52, 54).

BIRRELL, A. D. et NELSON, B. J. (1984). « Implementing remote procedure calls ». In : *ACM Transactions on Computer Systems (TOCS)* 2.1, p. 39–59 (cf. p. 90).

BLESER, G. et STRICKER, D. (2009). « Advanced tracking through efficient image processing and visual–inertial sensor fusion ». In : *Computers & Graphics* 33.1, p. 59–72 (cf. p. 137).

BLESER, G., DAMEN, D., BEHERA, A., HENDEBY, G., MURA, K., MIEZAL, M., GEE, A., PETERSEN, N., MAÇÃES, G., DOMINGUES, H. et al. (2015). « Cognitive learning, monitoring and assistance of industrial workflows using egocentric sensor networks ». In : *PloS one* 10.6 (cf. p. 150).

BORST, C. W. et VOLZ, R. A. (2005). « Evaluation of a haptic mixed reality system for interactions with a virtual control panel ». In : *Presence : Teleoperators & Virtual Environments* 14.6, p. 677–696 (cf. p. 180).

BOWMAN, D., KRUIJFF, E., LAVIOLA JR, J. J. et POUPYREV, I. P. (2004). *3D User Interfaces : Theory and Practice, CourseSmart eTextbook*. Addison-Wesley (cf. p. 53, 56).

CANAL, G., ANGULO, C. et ESCALERA, S. (2015). « Gesture based human multi-robot interaction ». In : *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, p. 1–8 (cf. p. 168).

CAUDELL, T. P. et MIZELL, D. W. (1992). « Augmented reality : An application of heads-up display technology to manual manufacturing processes ». In : *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*. T. 2. IEEE, p. 659–669 (cf. p. 49, 58, 149).

CHAARAOUI, A. A., CLIMENT-PÉREZ, P. et FLÓREZ-REVUELTA, F. (2012). « An Efficient Approach for Multi-view Human Action Recognition Based on Bag-of-Key-Poses ». In : *Human Behavior Understanding*. Sous la dir. d'A. A. SALAH, J. RUIZ-DEL-SOLAR, Ç. MERIÇLI et P.-Y. OUDEYER. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 29–40. ISBN : 978-3-642-34014-7 (cf. p. 171).

CHAARAOUI, A. A., PADILLA-LÓPEZ, J. R., CLIMENT-PÉREZ, P. et FLÓREZ-REVUELTA, F. (2014). « Evolutionary joint selection to improve human action recognition with RGB-D devices ». In : *Expert Systems with Applications* 41.3. Methods and Applications of Artificial and Computational Intelligence, p. 786–794. ISSN : 0957-4174. DOI : https://doi.org/10.1016/j.eswa.2013.08.009. URL : http://www.sciencedirect.com/science/article/pii/S0957417413006210 (cf. p. 171).

CHAPPELL, D. (2009). *Introducing Windows Communication Foundation in .NET Framework 4*. https://msdn.microsoft.com/library/ee958158.aspx (cf. p. 91).

CHEN, J., BENESTY, J., HUANG, Y. et DOCLO, S. (2006). « New insights into the noise reduction Wiener filter ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 14.4, p. 1218–1234 (cf. p. 158).

CHI, H.-L., KANG, S.-C. et WANG, X. (2013). « Research trends and opportunities of augmented reality applications in architecture, engineering, and construction ». In : *Automation in construction* 33, p. 116–122 (cf. p. 60).

CHOUITEN, M., **DIDIER, J.-Y.** et MALLEM, M. (2011). « Component-based middleware for distributed augmented reality applications ». In : *Proceedings of the 5th International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE 2011)*. Verona, Italy : ACM, p. 3 (cf. p. 65, 95, 133).

— (2012a). « Distributed Augmented Reality Systems : How Much Performance is Enough ? » In : *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, p. 337–342. DOI : 10.1109/ICMEW.2012.64 (cf. p. 65, 99, 104, 133).

CHOUITEN, M., DOMINGUES, C., **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2012b). « Distributed mixed reality for remote underwater telerobotics exploration ». In : *Proceedings of the 2012 Virtual Reality International Conference*. VRIC '12. Laval, France : ACM, 1 :1–1 :6. ISBN : 978-1-4503-1243-1. DOI : 10.1145/2331714.2331716. URL : http://doi.acm.org/10.1145/2331714.2331716 (cf. p. 65, 133).

CHOUITEN, M., **DIDIER, J.-Y.** et MALLEM, M. (2013). « A Framework for Service Based Composite Augmented Reality Applications ». In : *Ubiquitous Virtual Reality (ISUVR), 2013 International Symposium on*. IEEE, p. 19–22 (cf. p. 65).

CHOUITEN, M., DOMINGUES, C., **DIDIER, J.-Y.**, OTMANE, S. et MALLEM, M. (2014). « Distributed Mixed reality for diving and underwater tasks using Remotely Operated Vehicles ». In : *International Journal on Computational Sciences & Applications* 5.4, elec–proc (cf. p. 65, 104, 133).

CHOUITEN, M. (2013). « Architecture distribuée dédiée aux applications de Réalité Augmentée mobile ». Thèse de doct. Université d'Evry val d'Essonne (cf. p. 65, 104, 133).

CICIRELLI, G., ATTOLICO, C., GUARAGNELLA, C. et D'ORAZIO, T. (2015). « A kinect-based gesture recognition approach for a natural human robot interface ». In : *International Journal of Advanced Robotic Systems* 12.3, p. 22 (cf. p. 168).

CIMEN, G., ILHAN, H., CAPIN, T. et GURCAY, H. (2013). « Classification of human motion based on affective state descriptors ». In : *Computer Animation and Virtual Worlds* 24.3-4, p. 355–363 (cf. p. 171).

CUNNINGHAM, W. (2013). *People Projects and Patterns*. http ://wiki.c2.com/ ?PeopleProjectsAndPatterns (cf. p. 68, 73).

DAVISON, A. J., REID, I. D., MOLTON, N. D. et STASSE, O. (2007). « MonoSLAM : Real-time single camera SLAM ». In : *IEEE transactions on pattern analysis and machine intelligence* 29.6, p. 1052–1067 (cf. p. 52).

DEVILLERS, R., **DIDIER, J.-Y.** et KLAUDEL, H. (2013). « Implementing timed automata specifications : the "sandwich" approach ». In : *13th International Conference on Application of Concurrency to System Design (ACSD2013)* (cf. p. 65, 121, 126, 130, 134).

DEVILLERS, R. R., **DIDIER, J.**, KLAUDEL, H. et ARCILE, J. (2014). « Deadlock and Temporal Properties Analysis in Mixed Reality Applications ». In : *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, p. 55–65. DOI : 10.1109/ISSRE.2014.33. URL : http://dx.doi.org/10.1109/ISSRE.2014.33 (cf. p. 65, 124, 125, 133).

DEVILLERS, R. et KLAUDEL, H. (2016). « Abstraction strategies for computing travelling or looping durations in networks of timed automata ». In : *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, p. 140–156 (cf. p. 126).

DIDIER, J.-Y., OTMANE, S., ROUSSEL, D. et MALLEM, M. (2003). « Architecture Logicielle Modulaire adaptée au Recalage Dynamique dans un Système de Réalité Augmentée en Vision Directe ». In : *17ème journée des Jeunes Chercheurs en Robotique*, p. 102–106 (cf. p. 65, 191).

DIDIER, J.-Y., ROUSSEL, D. et MALLEM, M. (2004). « A Texture Based Time Delay Compensation Method for Augmented Reality. » In : *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004)*. Arlington (USA), p. 262–263 (cf. p. 65).

— (2005a). « A Time Delay Compensation Method Improving Registration for Augmented Reality ». In : *2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*. Barcelona (Spain), p. 3396–3400 (cf. p. 65).

DIDIER, J.-Y., ROUSSEL, D., MALLEM, M., OTMANE, S., NAUDET, S., PHAM, Q.-C., BOURGEOIS, S., MÉGARD, C., LEROUX, C. et HOCQUARD, A. (2005b). « AMRA : Augmented Reality assistance in train maintenance tasks ». In : *Workshop on Industrial Augmented Reality (ISMAR'05)*. Vienna (Austria) (cf. p. 58, 65, 149).

DIDIER, J.-Y., OTMANE, S. et MALLEM, M. (2006). « A Component Model for Augmented/Mixed Reality Applications with Reconfigurable Data-flow ». In : *8th International Conference on Virtual Reality (VRIC 2006)*. Laval (France), p. 243–252 (cf. p. 65, 74, 133).

DIDIER, J.-Y., ABABSA, F.-e. et MALLEM, M. (2008a). « Hybrid Camera Pose Estimation Combining Square Fiducials Localization Technique and Orthogonal Iteration Algorithm ». In : *International Journal of Image and Graphics (IJIG)* 8.1, p. 169–188 (cf. p. 65).

DIDIER, J.-Y., DJAFRI, B. et KLAUDEL, H. (2008b). « MIRELA : A Language for Modeling and Analyzing Mixed Reality Applications Using Timed Automata ». In : *IEEE Virtual Reality 08*. Sous la dir. de M. LIN, A. STEED et C. CRUZ-NEIRA. IEEE. Reno, Nevada, p. 249–250 (cf. p. 65, 118).

— (2008c). « Modeling and analyzing mixed reality applications using timed automata ». In : *1st Mediterranean Conference on Intelligent Systems and Automation (CISA ?08)*. Sous la dir. de H. ARIOUI, R. MERZOUKI et H. A. ABBASSI. AIP, p. 173–178. URL : http://evra.ibisc.univ-evry.fr/Proceedings/CISA08/cdr_pdfs/indexed/stage4_copyr/173_1.pdf (cf. p. 65, 118, 134).

— (2008d). « The MIRELA Framework : modeling and analyzing mixed reality applications using timed automata ». In : *10th Virtual Reality International Conference*. Laval, France, p. 189–199 (cf. p. 65, 118, 134).

— (2009a). « The MIRELA framework : modeling and analyzing mixed reality applications using timed automata ». In : *Journal of Virtual Reality and Broadcasting*. VRIC 2008 (Laval Virtual) Special Issue 6.1. Sous la dir. de J. HERDER, S. RICHIR et I. THOUVENIN. t urn :nbn :de :0009-

6-17423,, ISSN 1860-2037. URL : http://www.jvrb.org/archiv/1742/ (cf. p. 65, 118, 124, 134, 188).

DIDIER, J.-Y., OTMANE, S. et MALLEM, M. (2009b). « ARCS : Une Architecture Logicielle Reconfigurable pour la conception des Applications de Réalité Augmentée ». In : *Technique et Science Informatiques (TSI), Réalité Virtuelle - Réalité Augmentée* 28.6-7/2009. Numéro spécial, p. 891–919 (cf. p. 65, 74, 133).

DIDIER, J.-Y., CHOUITEN, M., MALLEM, M. et OTMANE, S. (2012). « ARCS : A framework with extended software integration capabilities to build Augmented Reality applications ». In : *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on*, p. 60–67. DOI : 10.1109/SEARIS.2012.6231170 (cf. p. 65, 74, 76, 78, 95, 133).

DIDIER, J.-Y. et MALLEM, M. (2012). « Automated Concurrent Access Management for Components of Augmented Reality Applications ». In : *7èmes journées de l'AFRV*. Strasbourg, France, p. 13–19 (cf. p. 65, 110, 134).

DIDIER, J.-Y., KLAUDEL, H. et MOINE, M. (2013). « An Improved Approach to Build Safer Mixed Reality Systems by Analysing Time Constraints ». In : *Joint Virtual Reality Conference (JVRC) 2013 Poster, Demo and Industrial Track*, p. 87–90 (cf. p. 65, 126, 134).

DIDIER, J.-Y. et MALLEM, M. (2014). « A new approach to detect potential race conditions in component-based systems ». In : *Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*. ACM, p. 97–106 (cf. p. 65, 110, 115, 133).

DIDIER, J.-Y. (2002). « Recalage dynamique dans un système de réalité augmentée en vision indirecte ». Mém.de mast. Université d'Evry-Val d'Essone (cf. p. 65).

— (2005). « Contributions à la dextérité d'un système de réalité augmentée mobile appliqué à la maintenance industrielle ». Thèse de doct. Evry : Université d'Evry-Val d'Essonne (cf. p. 65, 74).

DÖRNER, R., GEIGER, C., HALLER, M. et PAELKE, V. (2002). « Authoring Mixed Reality. A Component and Framework-Based Approach ». In : *First International Workshop on Entertainment Computing (IWEC 2002)*. Makuhari, Chiba, Japon, p. 405–413 (cf. p. 70).

DOUGLASS, B. P. (1997). *Real-time UML : developing efficient objects for embedded systems*. Addison-Wesley Longman Publishing Co., Inc. (cf. p. 120).

DUBOIS, E., GRAY, P. D. et NIGAY, L. (2003). « ASUR++ : a design notation for mobile mixed systems ». In : *Interacting with computers* 15.4, p. 497–520 (cf. p. 49).

ELLIS, S. R., BREANT, F., MANGES, B., JACOBY, R. et ADELSTEIN, B. D. (1997). « Factors influencing operator interaction with virtual objects viewed via head-mounted see-through displays : viewing conditions and rendering latency ». In : *Virtual Reality Annual International Symposium, 1997., IEEE 1997*. IEEE, p. 138–145 (cf. p. 103).

ELMENREICH, W. (2002). *An introduction to sensor fusion*. Rapp. tech. Vienna University of Technology, Austria (cf. p. 53).

EMMI, M., FISCHER, J. S., JHALA, R. et MAJUMDAR, R. (2007). « Lock allocation ». In : *ACM SIGPLAN Notices*. T. 42. 1. ACM, p. 291–296 (cf. p. 112).

ENDRES, C., BUTZ, A. et MACWILLIAMS, A. (2005). « A Survey of Software Infrastructures and Frameworks for Ubiquitous Computing ». In : *Mobile Information Systems Journal* 1.1, p. 41–80 (cf. p. 69).

ENGLER, D. et ASHCRAFT, K. (2003). « RacerX : effective, static detection of race conditions and deadlocks ». In : *ACM SIGOPS Operating Systems Review*. T. 37. 5. ACM, p. 237–252 (cf. p. 111).

FEINER, S., MACINTYRE, B., HOLLERER, T. et WEBSTER, A. (1997). « A Touring Machine : Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment ». In : *ISWC*

'97 : Proceedings of the 1st IEEE International Symposium on Wearable Computers. Washington, DC, USA : IEEE Computer Society, p. 74. ISBN : 0-8186-8192-6 (cf. p. 59, 149).

FEINER, S. K. (2002). « Augmented reality : A new way of seeing ». In : *Scientific American* 286.4, p. 48–55 (cf. p. 49).

FIELDING, R. T. (2000). « Architectural styles and the design of network-based software architectures ». Thèse de doct. University of California, Irvine (cf. p. 93).

FIGUEROA, P., HOOVER, J. et BOULANGER, P. (2004). « Intml concepts ». In : *University of Alberta. Computing Science Department, Tech. Rep* (cf. p. 117).

FIGUEROA, P., BISCHOF, W. F., BOULANGER, P., HOOVER, H. J. et TAYLOR, R. (2008). « Intml : A dataflow oriented development system for virtual reality applications ». In : *Presence : Teleoperators and Virtual Environments* 17.5, p. 492–511 (cf. p. 117).

FILLATRE, L. (2011). « Contributions en Détection et Classification Statistique Paramétrique ». Habilitation à Diriger des Recherches. Institut Charles DELAUNAY – Université de technologie de Troyes (cf. p. 154).

FISCHLER, M. et BOLLES, R. (1981). « Random Sample Consensus : A paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography ». In : *Communications of the ACM* 24.6, p. 381–395 (cf. p. 142).

FITE-GEORGEL, P. (2011). « Is there a reality in Industrial Augmented Reality ? » In : *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, p. 201–210 (cf. p. 149).

FLANAGAN, C. et FREUND, S. N. (2009). « FastTrack : efficient and precise dynamic race detection ». In : *ACM Sigplan Notices*. T. 44. 6. ACM, p. 121–133 (cf. p. 111).

FOTHERGILL, S., MENTIS, H., KOHLI, P. et NOWOZIN, S. (2012). « Instructing people for training gestural interactive systems ». In : *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, p. 1737–1746 (cf. p. 168).

FRAUCIEL, L., VAIRON, J., NEHLIG, P., THIERRY, P., ZENDJEBIL, I. et ABABSA, F. (2008). « Outdoor Augmented Reality as a tool for bringing 3D geology to the field : the RAXENV project ». In : *International Geological Congress*. Oslo : (cf. p. 135).

FRIEDRICH, W., JAHN, D. et SCHMIDT, L. (2002). « ARVIKA-Augmented Reality for Development, Production and Service. » In : *ISMAR*. T. 2002, p. 3–4 (cf. p. 58).

« Le Traité de la réalité virtuelle » (2006). In : sous la dir. de P. FUCHS et G. MOREAU. Troisième. T. 3. École des Mines de Paris. Chap. Virtools et la réalité virtuelle, p. 399–410 (cf. p. 71).

FUCHS, H., LIVINGSTON, M. A., RASKAR, R., KELLER, K., CRAWFORD, J. R., RADEMACHER, P., DRAKE, S. H., MEYER, A. A. et al. (1998). « Augmented reality visualization for laparoscopic surgery ». In : *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, p. 934–943 (cf. p. 58).

GALLAGHER, S. (2017). *Heads up : Augmented reality prepares for the battlefield*. https://arstechnica.com/information-technology/2017/05/heads-up-augmented-reality-prepares-for-the-battlefield/ (cf. p. 59).

GAMMA, E., HELM, R., JOHNSON, R. et VLISSIDES, J. (1993). « Design patterns : Abstraction and reuse of object-oriented design ». In : *European Conference on Object-Oriented Programming*. Springer, p. 406–431 (cf. p. 68, 89).

— (1999). *Design Patterns - Catalogue de modèles de conceptions réutilisables*. Vuibert (cf. p. 68).

GARLAN, D. et SHAW, M. (1993). « An introduction to Software Architecture ». In : *Advances in Software Engineering and Knowledge Engineering*. Sous la dir. de V. AMBRIOLA et G. TORTORA. T. 1. New Jersey : World Scientific Publishing Company, p. 1–40 (cf. p. 68, 89).

GHORBEL, E., BOUTTEAU, R., BOONAERT, J., SAVATIER, X. et LECOEUCHE, S. (2015). « 3D real-time human action recognition using a spline interpolation approach ». In : *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, p. 61–66. DOI : 10.1109/IPTA.2015.7367097 (cf. p. 171).

GREENBERG, S. et FITCHETT, C. (2001). « Phidgets : easy development of physical interfaces through physical widgets ». In : *Proceedings of the 14th annual ACM symposium on User interface software and technology*. ACM, p. 209–218 (cf. p. 54).

GROSSO, W. (2001). *Java RMI. Java Series*. O'Reilly Media (cf. p. 90).

GUDGIN, M., HADLEY, M., MENDELSOHN, N., MOREAU, J.-J., NIELSEN, H. F., KARMARKAR, A. et LAFON, Y. (2007). *SOAP Version 1.2 Part 1 : Messaging Framework*. Rapp. tech. W3C (cf. p. 92).

HAAS, H. et BROWN, A. (2004). *Web Services Glossary*. Rapp. tech. W3C (cf. p. 92).

HAKKARAINEN, M., WOODWARD, C. et BILLINGHURST, M. (2008). « Augmented Assembly Using a Mobile Phone ». In : *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '08. Washington, DC, USA : IEEE Computer Society, p. 167–168 (cf. p. 149).

HALLER, M., ZAUNER, J., HARTMANN, W. et LUCKENEDER, T. (2003). *A generic framework for a training application based on Mixed Reality.* Rapp. tech. Hagenberg, Austria : Upper Austria University of Applied Sciences (cf. p. 149).

HALPERT, R. L., PICKETT, C. J. et VERBRUGGE, C. (2007). « Component-based lock allocation ». In : *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*. IEEE Computer Society, p. 353–364 (cf. p. 112).

HAOUCHINE, N., DEQUIDT, J., PETERLIK, I., KERRIEN, E., BERGER, M.-O. et COTIN, S. (2013). « Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery ». In : *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. IEEE, p. 199–208 (cf. p. 58).

HAREL, D. (1987a). « On the formal semantics of statecharts ». In : *Proc of 2nd IEEE Symposium on Logic in Computer Science, 1987* (cf. p. 120).

— (1987b). « Statecharts : A visual formalism for complex systems ». In : *Science of computer programming* 8.3, p. 231–274 (cf. p. 120).

HARRIS, C. (1992). « Tracking with rigid models ». In : *Active vision*, p. 59–73 (cf. p. 142).

HENDERSON, S. J. et FEINER, S. (2009). « Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret ». In : *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, p. 135–144 (cf. p. 149).

HETTINGER, L. J. et RICCIO, G. E. (1992). « Visually Induced Motion Sickness in Virtual Environments ». In : *Presence : Teleoperators and Virtual Environments* 1.3, p. 306–310. DOI : 10.1162/pres.1992.1.3.306. eprint : http://dx.doi.org/10.1162/pres.1992.1.3.306. URL : http://dx.doi.org/10.1162/pres.1992.1.3.306 (cf. p. 55).

HOLWEG, D. et SCHNEIDER, O. (2004). « GEIST : Mobile Outdoor AR-Information System for Historical Education with Digital Storytelling ». In : *Federal Ministry of Education and Research : Virtual and Augmented Reality Status Conference* (cf. p. 59).

HOL, J. D., SCHON, T., GUSTAFSSON, F. et SLYCKE, P. J. (2006). « Sensor fusion for augmented reality ». In : *Information Fusion, 2006 9th International Conference on*. IEEE, p. 1–6 (cf. p. 137).

HORAUD, R. et MONGA, O. (1995). *Vision par ordinateur : outils fondamentaux*. Editions Hermès (cf. p. 141).

HUGHES, C. E., STAPLETON, C. B., HUGHES, D. E. et SMITH, E. M. (2005). « Mixed reality in education, entertainment, and training ». In : *IEEE computer graphics and applications* 25.6, p. 24–30 (cf. p. 72).

HUIKARI, V., KOSKIMÄKI, H., SIIRTOLA, P. et RONING, J. (2010). « User-independent activity recognition for industrial assembly lines-feature vs. instance selection ». In : *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, p. 307–312 (cf. p. 150).

HWANG, G., AARNO, D. et HASHIMOTO, H. (2006). « Haptic guidant bilateral teleoperation for single-master multi-slave system ». In : *Proceedings of Eurohaptics*. T. 6. 2006.7. Citeseer (cf. p. 180).

ILIE, A., RASKAR, R. et YU, J. (2005). « Gradient domain context enhancement for fixed cameras ». In : *International Journal of Pattern Recognition and Artificial Intelligence* 19.04, p. 533–549 (cf. p. 182).

IRAWATI, S., AHN, S., KIM, J. et KO, H. (2008). « Varu framework : Enabling rapid prototyping of VR, AR and ubiquitous applications ». In : *Virtual Reality Conference, 2008. VR'08. IEEE*. IEEE, p. 201–208 (cf. p. 72).

JOHNSON, R. E. (1997). « Components, frameworks, patterns ». In : *ACM SIGSOFT Software Engineering Notes*. T. 22. 3. ACM, p. 10–17 (cf. p. 68, 72).

JULIER, S., BAILLOT, Y., LANZAGORTA, M., BROWN, D. et ROSENBLUM, L. (2000). « BARS : Battlefield Augmented Reality system ». In : *NATO Symposium on Information Processing Techniques for Military Systems*. Istanbul, Turkey (cf. p. 59).

KATO, H. et BILLINGHURST, M. (1999). « Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System ». In : *IWAR '99 : Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*. Washington, DC, USA : IEEE Computer Society, p. 85–92. ISBN : 0-7695-0359-4 (cf. p. 52, 181).

KAUFMANN, H. et SCHMALSTIEG, D. (2003). « Mathematics and geometry education with collaborative augmented reality ». In : *Computers & graphics* 27.3, p. 339–345 (cf. p. 60).

KAUFMANN, H., MOSSEL, A., SCHÖNAUER, C. et GERSTWEILER, G. (2012). « ARTiFICe-Augmented Reality Framework for Distributed Collaboration ». In : *International Journal of Virtual Reality (IJVR)* 11.3, p. 1–7 (cf. p. 71).

KERALIA, D., VYAS, K. et DEULKAR, K. (2014). « Google project tango–a convenient 3D modeling device ». In : *International Journal of Current Engineering and Technology* 4.5, p. 3139–3142 (cf. p. 53).

KHEDDAR, A., DRIF, A., CITÉRIN, J. et LE MERCIER, B. (2004). « A multi-level haptic rendering concept ». In : *EuroHaptics, Munich, Herbert Hieronymus*, p. 147–154 (cf. p. 180).

KLEINSMITH, A. et BIANCHI-BERTHOUZE, N. (2013). « Affective body expression perception and recognition : A survey ». In : *IEEE Transactions on Affective Computing* 4.1, p. 15–33 (cf. p. 171).

KRESS, B. C. et CUMMINGS, W. J. (2017). « 11-1 : Invited Paper : Towards the Ultimate Mixed Reality Experience : HoloLens Display Architecture Choices ». In : *SID Symposium Digest of Technical Papers* 48.1, p. 127–131. ISSN : 2168-0159. DOI : 10.1002/sdtp.11586. URL : http://dx.doi.org/10.1002/sdtp.11586 (cf. p. 54).

KUCK, R., WIND, J., RIEGE, K., BOGEN, M. et BIRLINGHOVEN, S. (2008). « Improving the avango vr/ar framework : Lessons learned ». In : *Workshop Virtuelle und Erweiterte Realität*, p. 209–220 (cf. p. 71, 73).

KWIATKOWSKA, M., NORMAN, G. et PARKER, D. (2011). « PRISM 4.0 : Verification of probabilistic real-time systems ». In : *International conference on computer aided verification*. Springer, p. 585–591 (cf. p. 120, 123).

LAMPORT, L. (1978a). « Time, clocks, and the ordering of events in a distributed system ». In : *Communications of the ACM* 21.7, p. 558–565 (cf. p. 88).

— (1978b). « Time, clocks, and the ordering of events in a distributed system ». In : *Communications of the ACM* 21.7, p. 558–565 (cf. p. 111).

LARSEN, K. G., PETTERSSON, P. et YI, W. (1997). « UPPAAL in a Nutshell ». In : *International Journal on Software Tools for Technology Transfer (STTT)* 1.1-2. ISSN 1433-2779, p. 134–152 (cf. p. 120, 123).

LATOSCHIK, M. E. (2002). « Designing transition networks for multimodal VR-interactions using a markup language ». In : *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*. IEEE Computer Society, p. 411 (cf. p. 117).

LEHRMANN, A. M., GEHLER, P. V. et NOWOZIN, S. (2014). « Efficient nonlinear markov models for human motion ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1314–1321 (cf. p. 168, 171).

LEPETIT, V., MORENO-NOGUER, F. et FUA, P. (2009). « EPnP : An Accurate O(n) Solution to the PnP Problem ». In : *International Journal of Computer Vision* 81.2, p. 155–166 (cf. p. 153).

LI, Z., DURAISWAMI, R. et DAVIS, L. S. (2004). « Recording and Reproducing High Order Surround Auditory Scenes for Mixed and Augmented Reality ». In : *ISMAR '04 : Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA : IEEE Computer Society, p. 240–249. ISBN : 0-7695-2191-6. DOI : http://dx.doi.org/10.1109/ISMAR.2004.51 (cf. p. 56).

LI, W., ZHANG, Z. et LIU, Z. (2010). « Action recognition based on a bag of 3d points ». In : *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, p. 9–14 (cf. p. 168, 171).

LINDEMAN, R. W. et NOMA, H. (2007). « A classification scheme for multi-sensory augmented reality ». In : *VRST '07 : Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. New York, NY, USA : ACM, p. 175–178 (cf. p. 50).

LINDEMAN, R. W., NOMA, H. et BARROS, P. G. de (2007). « Hear-through and mic-through augmented reality : Using bone conduction to display spatialized audio ». In : *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, p. 173–176 (cf. p. 56).

LORDEN, G. (1971). « Procedures for Reacting to a Change in Distribution ». In : *Ann. Math. Statist.* 42.6, p. 1897–1908 (cf. p. 154).

LOURENS, T., VAN BERKEL, R. et BARAKOVA, E. (2010). « Communicating emotions and mental states to robots in a real time parallel framework using Laban movement analysis ». In : *Robotics and Autonomous Systems* 58.12, p. 1256–1265 (cf. p. 171).

LOWE, D. G. (1999). « Object recognition from local scale-invariant features ». In : *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. T. 2. Ieee, p. 1150–1157 (cf. p. 52, 153).

LU, C.-P., HAGER, G. et MJOLSNESS, E. (2000). « Fast and Globally Convergent Pose Estimation from Video Images ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.6, p. 610–622. ISSN : 0162-8828. DOI : http://dx.doi.org/10.1109/34.862199 (cf. p. 142, 151).

LUCAS, T. (2013). *Diotasoft industrialise la réalité augmentée*. http://www.usinenouvelle.com/article/diotasoft-industrialise-la-realite-augmentee.N197327 (cf. p. 59).

LUN, R. et ZHAO, W. (2015). « A survey of applications and human motion recognition with microsoft kinect ». In : *International Journal of Pattern Recognition and Artificial Intelligence* 29.05, p. 1555008 (cf. p. 168).

MACWILLIAMS, A., REICHER, T., KLINKER, G. et BRÜEGGE, B. (2004). « Design Patterns for Augmented Reality Systems ». In : *Proceedings of the International Workshop exploring the Design and Engineering of Mixed Reality Systems (MIXER), Funchal, Madeira, CEUR Workshop Proceedings*. URL : http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-91/paperE4.pdf (cf. p. 98, 99, 106).

MAIDI, M., ABABSA, F. et MALLEM, M. (2009). « Vision-inertial tracking system for robust fiducials registration in augmented reality ». In : *Computational Intelligence for Multimedia Signal and Vision Processing, 2009. CIMSVP'09. IEEE Symposium on*. IEEE, p. 83–90 (cf. p. 137).

MAIDI, M., **DIDIER, J.-Y.**, ABABSA, F. et MALLEM, M. (2010). « A Performance Study for Camera Pose Estimation using Visual Marker based Tracking ». In : *Machine Vision and Applications, IAPR International Journal, Springer* 21.3, p. 365–376 (cf. p. 65).

MARLET, R. (2011). *Ingénierie de la spécialisation de programmes : Tome 1, Principes et applications*. T. 1. Lavoisier (cf. p. 73, 86).

MASUDA, M. et KATO, S. (2010). « Motion rendering system for emotion expression of human form robots based on laban movement analysis ». In : *RO-MAN, 2010 IEEE*. IEEE, p. 324–329 (cf. p. 175).

MERAD, D., **DIDIER, J.-Y.** et SCUTURICI, M. (2006). « Tracking 3D free form object in video sequence ». In : *Third Canadian Conference on Computer and Robot Vision*. Quebec city (Canada), p. 50 (cf. p. 65).

MILGRAM, P., TAKEMURA, H., UTSUMI, A. et KISHINO, F. (1994). « Augmented Reality : A class of displays on the reality-virtuality continuum ». In : *SPIE : Telemanipulator and Telepresence Technologies* 2351, p. 282–292 (cf. p. 50, 51, 54).

MOORE, E. F. (1956). « Gedanken Experiments on Sequential Machines ». In : *Automata Studies*. Princeton, p. 129–153 (cf. p. 76).

MUJA, M. et LOWE, D. G. (2009). « Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration ». In : *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, p. 331–340 (cf. p. 153).

NARUMI, T., NISHIZAKA, S., KAJINAMI, T., TANIKAWA, T. et HIROSE, M. (2011). « Meta Cookie+ : An Illusion-Based Gustatory Display ». In : *Virtual and Mixed Reality - New Trends : International Conference, Virtual and Mixed Reality 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I*. Sous la dir. de R. SHUMAKER. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 260–269. ISBN : 978-3-642-22021-0. DOI : 10.1007/978-3-642-22021-0_29. URL : https://doi.org/10.1007/978-3-642-22021-0_29 (cf. p. 56).

NAVARRE, D., PALANQUE, P., BASTIDE, R., SCHYN, A., WINCKLER, M., NEDEL, L. P. et FREITAS, C. M. (2005). « A formal description of multimodal interaction techniques for immersive virtual reality applications ». In : *IFIP Conference on Human-Computer Interaction*. Springer, p. 170–183 (cf. p. 117).

NAVAB, N., BLUM, T., WANG, L., OKUR, A. et WENDLER, T. (2012). « First Deployments of Augmented Reality in Operating Rooms ». In : *Computer* 45.7, p. 48–55. ISSN : 0018-9162. DOI : 10.1109/MC.2012.75 (cf. p. 58).

NEGIN, F., AKGÜL, C. B., YÜKSEL, K. A. et ERÇIL, A. (2015). « An RDF-based action recognition framework with feature selection capability, considering therapy exercises utilizing depth cameras ». In : (cf. p. 171).

NEWCOMER, E. et LOMOW, G. (2004). *Understanding SOA with Web services*. Addison-Wesley (cf. p. 92, 93).

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S. et FITZGIBBON, A. (2011). « KinectFusion : Real-time dense surface mapping and tracking ». In : *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, p. 127–136 (cf. p. 52).

NIANTIC (2012). *Ingress Website*. http://www.ingress.com (cf. p. 101).

— (2015). *Pokemon Go*. https://pokemongolive.com/ (cf. p. 101).

OHLENBURG, J., HERBST, I., LINDT, I., FRÖHLICH, T. et BROLL, W. (2004). « The MORGAN framework : enabling dynamic multi-user AR and VR projects. » In : *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2004*. Honk Kong, China, p. 166–169 (cf. p. 71).

OLIVEIRA, R., FARINHA, T., RAPOSO, H. et PIRES, N. (2014). « Augmented reality and the future of maintenance ». In : *Proceedings of Maintenance Performance Measurement and Management (MPMM)*. Imprensa da Universidade de Coimbra. Coimbra : Imprensa da Universidade de Coimbra, p. 81–88. ISBN : 978-972-8954-42-0 (PDF). DOI : http://dx.doi.org/10.14195/978-972-8954-42-0_12. URL : https://digitalis.uc.pt/handle/10316.2/33320 (cf. p. 58).

OMG (1991). *Common Object Request Broker Architecture*. Rapp. tech. OMG (cf. p. 90).

OWEN, C., TANG, A. et XIAO, F. (2003). « ImageTclAR : A Blended Script and Compiled Code Development System for Augmented Reality ». In : *Proceedings of the International Workshop on Software Technology for Augmented Reality Systems*. Tokyo, Japon, p. 23–28 (cf. p. 70).

PAGE, E. S. (1954). « Continuous Inspection Schemes ». In : *Biometrika* 41.1-2, p. 100–115 (cf. p. 154).

PALMARINI, R., ERKOYUNCU, J. A., ROY, R. et TORABMOSTAEDI, H. (2018). « A systematic review of augmented reality applications in maintenance ». In : *Robotics and Computer-Integrated Manufacturing* 49, p. 215–228 (cf. p. 58).

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V. et al. (2011). « Scikit-learn : Machine learning in Python ». In : *Journal of machine learning research* 12.Oct, p. 2825–2830 (cf. p. 173).

PEDERSEN, I., GALE, N., MIRZA-BABAEI, P. et REID, S. (2017). « More Than Meets the Eye : The Benefits of Augmented Reality and Holographic Displays for Digital Cultural Heritage ». In : *J. Comput. Cult. Herit.* 10.2, 11 :1–11 :15. ISSN : 1556-4673. DOI : 10.1145/3051480. URL : http://doi.acm.org/10.1145/3051480 (cf. p. 59).

PIEKARSKI, W. et THOMAS, B. (2003). « An Object-Oriented Software Architecture for 3D Mixed Reality Applications ». In : *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'03)*. Tokyo, Japan : IEEE Computer Society, p. 247–256. ISBN : 0-7695-2006-5. URL : http://www.tinmith.net/papers/piekarski-ismar-arch-2003.pdf (cf. p. 70, 73).

PIEKARSKI, W. (2004). *Hacking Your Own Virtual and Augmented Reality Apps for Fun and Profit!* ftp://ftp3.au.freebsd.org/pub/linux.conf.au/2004/papers/34-wayne-piekarski-paper.pdf (cf. p. 62).

PIZURICA, A., PHILIPS, W., LEMAHIEU, I. et ACHEROY, M. (2003). « A versatile wavelet domain noise filtration technique for medical imaging ». In : *IEEE Transactions on Medical Imaging* 22.3, p. 323–331 (cf. p. 158).

PNUELI, A. et MANNA, Z. (1992). « The temporal logic of reactive and concurrent systems ». In : *Springer* 16, p. 12 (cf. p. 120).

POLLAK, M. et SIEGMUND, D. (1975). « Approximations to the Expected Sample Size of Certain Sequential Tests ». In : *Ann. Statist.* 3.6, p. 1267–1282 (cf. p. 155).

PONDER, M., PAPAGIANNAKIS, G., MOLET, T., MAGNENAT-THALMANN, N. et THALMANN, D. (2003). « VHD++ development framework : Towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies ». In : *Computer Graphics International, 2003. Proceedings*. IEEE, p. 96–104 (cf. p. 69, 70).

POZNIANSKY, E. et SCHUSTER, A. (2007). « MultiRace : efficient on-the-fly data race detection in multithreaded C++ programs ». In : *Concurrency and Computation : Practice and Experience* 19.3, p. 327–340 (cf. p. 111).

PRATIKAKIS, P., FOSTER, J. S. et HICKS, M. (2006). « LOCKSMITH : context-sensitive correlation analysis for race detection ». In : *ACM SIGPLAN Notices* 41.6, p. 320–331 (cf. p. 111).

PRESTI, L. L. et LA CASCIA, M. (2016). « 3D skeleton-based human action classification : A survey ». In : *Pattern Recognition* 53, p. 130–147 (cf. p. 168).

RADKOWSKI, R. et STRITZKE, C. (2012). « Interactive hand gesture-based assembly for augmented reality applications ». In : *ACHI 2012, The Fifth International Conference on Advances in Computer-Human Interactions*, p. 303–308 (cf. p. 150).

RAGHAVAN, V., MOLINEROS, J. et SHARMA, R. (1999). « Interactive evaluation of assembly sequences using augmented reality ». In : *Robotics and Automation, IEEE Transactions on* 15.3, p. 435–449 (cf. p. 149).

REGENBRECHT, H., BARATOFF, G. et WILKE, W. (2005). « Augmented reality projects in the automotive and aerospace industries ». In : *Computer Graphics and Applications, IEEE* 25.6, p. 48–56 (cf. p. 149).

REITMAYR, G. et SCHMALSTIEG, D. (2001). « An open software architecture for virtual reality interaction ». In : *Proceedings of the ACM symposium on Virtual reality software and technology*. Baniff, Alberta, Canada : ACM Press, p. 47–54. ISBN : 1-58113-427-4. DOI : http://doi.acm.org/10.1145/505008.505018 (cf. p. 71).

REITMAYR, G., EADE, E. et DRUMMOND, T. (2005). « Localisation and interaction for augmented maps ». In : *Proceedings of the 4th IEEE/ACM international Symposium on Mixed and Augmented Reality*. IEEE Computer Society, p. 120–129 (cf. p. 60).

REITMAYR, G. et DRUMMOND, T. (2006). « Going out : robust model-based tracking for outdoor augmented reality ». In : *Fifth IEEE/ACM International Symposium on Mixed and Augmented Reality*. Santa Barbara, California, p. 109–118 (cf. p. 52).

REITMAYR, G. et DRUMMOND, T. W. (2007). « Initialisation for visual tracking in urban environments ». In : *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, p. 161–172 (cf. p. 147).

REISIG, W. (2012). *Petri nets : an introduction*. T. 4. Springer Science & Business Media (cf. p. 120).

REKIMOTO, J. et NAGAO, K. (1995). « The world through the computer : Computer augmented interaction with real world environments ». In : *Proceedings of the 8th annual ACM symposium on User interface and software technology*. ACM, p. 29–36 (cf. p. 49).

RIBO, M., LANG, P., GANSTER, H., BRANDNER, M., STOCK, C. et PINZ, A. (2002). « Hybrid tracking for outdoor augmented reality applications ». In : *IEEE Computer Graphics and Applications* 22.6, p. 54–63 (cf. p. 137).

RICHARDS, M. (2015). *Software architecture patterns*. O'Reilly Media, Incorporated (cf. p. 68).

ROBERTS, S. W. (1959). « Control Chart Tests Based on Geometric Moving Averages ». In : *Technometrics* 1.3, p. 239–250 (cf. p. 154).

ROETENBERG, D., LUINGE, H. et SLYCKE, P. (2009). « Xsens MVN : full 6DOF human motion tracking using miniature inertial sensors ». In : *Xsens Motion Technologies BV, Tech. Rep* 1 (cf. p. 168).

RUKUBAYIHUNGA, A., **DIDIER, J.-Y.**, CHOUITEN, M. et OTMANE, S. (2014). « An overview of occlusion issues handling in Augmented Reality ». In : *8èmes journées de l'Association française de réalité virtuelle, augmentée, mixte et d'interaction 3D*, p. 83–95 (cf. p. 65).

RUKUBAYIHUNGA, A., **DIDIER, J.-Y.** et OTMANE, S. (2015). « Reprojection error as a new metric to detect assembly/disassembly maintenance tasks ». In : *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*. IEEE, p. 513–518 (cf. p. 65, 151, 190).

— (2016a). « Real Time Noise Reduction to Identify Motion Parameters in AR Maintenance Scenario ». In : *Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on*. IEEE, p. 27–30 (cf. p. 65, 158, 189).

— (2016b). « Towards assembly steps recognition in augmented reality ». In : *Proceedings of the 2016 Virtual Reality International Conference*. ACM, p. 17 (cf. p. 65, 189).

RUKUBAYIHUNGA, A. (2016). « Vers un système interactif de réalité augmentée mobile pour la supervision de scénarios de maintenance industrielle ». Thèse de doct. Université d'Evry-val d'Essonne / Université de Paris-Saclay (cf. p. 65, 150).

SÄÄSKI, J., SALONEN, T., HAKKARAINEN, M., SILTANEN, S., WOODWARD, C. et LEMPIÄINEN, J. (2008). « Integration of design and assembly using augmented reality ». In : *Micro-Assembly Technologies and Applications*. Springer, p. 395–404 (cf. p. 149).

SALONEN, T., SÄÄSKI, J., HAKKARAINEN, M., KANNETIS, T., PERAKAKIS, M., SILTANEN, S., POTAMIANOS, A., KORKALO, O. et WOODWARD, C. (2007). « Demonstration of assembly work using augmented reality ». In : *Proceedings of the 6th ACM international conference on Image and video retrieval*. ACM, p. 120–123 (cf. p. 149).

SANDOR, C. et REICHER, T. (2001). « CUIML : A language for the generation of multimodal human-computer interfaces ». In : *Proceedings of the European UIML conference*. T. 124 (cf. p. 117).

SAVITZKY, A. et GOLAY, M. J. E. (1964). « Smoothing and Differentiation of Data by Simplified Least Squares Procedures. » In : *Analytical Chemistry* 36.8, p. 1627–1639 (cf. p. 158).

SAVAGE, S., BURROWS, M., NELSON, G., SOBALVARRO, P. et ANDERSON, T. (1997). « Eraser : A dynamic data race detector for multithreaded programs ». In : *ACM Transactions on Computer Systems (TOCS)* 15.4, p. 391–411 (cf. p. 111).

SCHMIDT, D. C., STAL, M., ROHNERT, H. et BUSCHMANN, F. (2000). *Pattern-Oriented Software Architecture*. T. 2 : Patterns for Concurrent and Networked Objects. John Wiley & Sons. ISBN : ISBN 0-471-60695-2 (cf. p. 110).

SCHWALD, B., FIGUE, J., CHAUVINEAU, E., VU-HONG, F., ROBERT, A., ARBOLINO, M., SCHNAIDER, M., LAVAL, B. D., RAULY, F. D. D., ANEZ, F., BALDO, O. et SANTOS, J. (2001). « Starmate : Using Augmented Reality for computer guided maintenance of complex mechanical elements. » In : *Conference e2001*. Venise (cf. p. 58).

SCHMALSTIEG, D., FUHRMANN, A., HESINA, G., SZALAVARI, Z., ENCARNAÇÄO, L. M., GERVAUTZ, M. et PURGATHOFER, W. (2002). « The studierstube augmented reality project ». In : *Presence : Teleoperators and Virtual Environments* 11.1, p. 33–54. ISSN : 1054-7460. DOI : http://dx.doi.org/10.1162/105474602317343640 (cf. p. 71).

SCHNÄDELBACH, H., KOLEVA, B., FLINTHAM, M., FRASER, M., IZADI, S., CHANDLER, P., FOSTER, M., BENFORD, S., GREENHALGH, C. et RODDEN, T. (2002). « The augurscope : a mixed reality interface for outdoors ». In : *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, p. 9–16 (cf. p. 59).

SCHALL, G., ZOLLMANN, S. et REITMAYR, G. (2013). « Smart Vidente : advances in mobile augmented reality for interactive visualization of underground infrastructure ». In : *Personal and ubiquitous computing* 17.7, p. 1533–1549 (cf. p. 60).

SCHNOEBELEN, P. (2002). « The Complexity of Temporal Logic Model Checking. » In : *Advances in modal logic* 4.393-436, p. 35 (cf. p. 126).

SHEWHART, W. A. (1930). « Economic Quality Control of Manufactured Product1 ». In : *Bell System Technical Journal* 9.2, p. 364–389 (cf. p. 154).

SIELHORST, T., OBST, T., BURGKART, R., RIENER, R. et NAVAB, N. (2004). « An augmented reality delivery simulator for medical training ». In : *International Workshop on Augmented Environments for Medical Imaging-MICCAI Satellite Workshop*. T. 141, p. 11–20 (cf. p. 58).

SLEE, M., AGARWAL, A. et KWIATKOWSKI, M. (2007). *Thrift : Scalable cross-language services implementation*. White paper. Facebook (cf. p. 90).

SMITH, C. U. et WILLIAMS, L. G. (2002). « Performance and scalability of distributed software architectures : An SPE approach ». In : *Parallel and Distributed Computing Practices* 3.4, p. 74106–0700 (cf. p. 101).

SOH, H. et DEMIRIS, Y. (2012). « Iterative temporal learning and prediction with the sparse online echo state gaussian process ». In : *The 2012 International Joint Conference on Neural Networks (IJCNN)*, p. 1–8. DOI : 10.1109/IJCNN.2012.6252504 (cf. p. 171).

SONG, Y., MORENCY, L.-P. et DAVIS, R. (2013). « Distribution-sensitive learning for imbalanced datasets ». In : *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, p. 1–6 (cf. p. 171).

STERLING, N. (1993). « WARLOCK-A Static Data Race Analysis Tool. » In : *USENIx Winter*, p. 97–106 (cf. p. 111).

SUTHERLAND, I. (1968). « A Head-Mounted Three Dimensional Display ». In : *Proceedings of the Fall Joint Computer Conference*. Washington DC : Thompson Books, p. 757–764 (cf. p. 50).

SZYPERSKI, C. (2002). *Component Software - Beyond Object-Oriented Programming*. second. Harlow, England : Addison-Wesley (cf. p. 68, 73, 74).

TANG, A., OWEN, C., BIOCCA, F. et MOU, W. (2003). « Comparative Effectiveness of Augmented Reality in Object Assembly ». In : *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. Ft. Lauderdale, Florida, USA : ACM, p. 73–80 (cf. p. 149).

THE QT COMPANY (2017). *Qt for developpers by developpers — Cross-platform development*. https://www.qt.io/developers/ (cf. p. 80).

THOMAS, B., CLOSE, B., DONOGHUE, J., SQUIRES, J., DE BONDI, P., MORRIS, M. et PIEKARSKI, W. (2000). « ARQuake : An outdoor/indoor augmented reality first person application ». In : *Wearable computers, the fourth international symposium on*. IEEE, p. 139–146 (cf. p. 56).

TINEL, J. et DARDY, F. (1995). *Les points clés de la construction mécanique*. Foucher. ISBN : 2-216-03162-3 (cf. p. 157).

TITA, B. (2015). « Smart Glasses Get New Look on Factory Floor ». In : *The Wall Street Journal*. URL : https://www.wsj.com/articles/smart-glasses-get-new-look-on-factory-floor-1433301177 (cf. p. 58).

TOKUNAGA, E., ZEE, A. van der, KURAHASHI, M., NEMOTO, M. et NAKAJIMA, T. (2003). « Object-Oriented Middleware Infrastructure for Distributed Augmented Reality ». In : *ISORC'03*, p. 156–163 (cf. p. 99).

TOMASI, C. et KANADE, T. (1991). *Detection and tracking of point features*. Rapp. tech. (cf. p. 143).

TOTALIMMERSION (2008). *D'Fusion Studio*. http://www.t-immersion.com/products/dfusion-suite/dfusion-studio. Total Immersion (cf. p. 70).

TRUONG, A., BOUJUT, H. et ZAHARIA, T. (2016). « Laban descriptors for gesture recognition and emotional analysis ». In : *The visual computer* 32.1, p. 83–98 (cf. p. 167, 171).

TRUONG, A. et ZAHARIA, T. (2016). « Dynamic Gesture Recognition with Laban Movement Analysis and Hidden Markov Models ». In : *Proceedings of the 33rd Computer Graphics International*. ACM, p. 21–24 (cf. p. 168).

TURRO, N. et KHATIB, O. (2001). « Haptically augmented teleoperation ». In : *Experimental Robotics VII*. Springer, p. 1–10 (cf. p. 180).

TWEEDIE, S. (2013). *Ikea's Augmented Reality Catalog Lets You Virtually Demo Its Furniture In Your Living Room*. http ://www.businessinsider.fr/us/ikeas-2014-augmented-reality-catalog-2013-8/ (cf. p. 59).

VALLINO, J. et BROWN, C. (1999). « Haptics in augmented reality ». In : *Multimedia Computing and Systems, 1999. IEEE International Conference on*. T. 1. IEEE, p. 195–200 (cf. p. 179).

VAPNIK, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience (cf. p. 160).

VIÉVILLE, T., ROMANN, F., HOTZ, B., MATHIEU, H., BUFFA, M., ROBERT, L., FACAO, P. D. S., FAUGERAS, O. D. et AUDREN, J.-T. (1993). « Autonomous navigation of a mobile robot using inertial and visual cues ». In : *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*. T. 1. IEEE, p. 360–367 (cf. p. 137).

VLAHAKIS, V., IOANNIDIS, N., KARIGIANNIS, J., TSOTROS, M., GOUNARIS, M., STRICKER, D., GLEUE, T., DAEHNE, P. et ALMEIDA, L. (2002). « Archeoguide : An Augmented Reality Guide for Archeological Sites ». In : *VRIC 2002 PRoceedings(Laval Virtual 4th Virtual Reality International Conference)*. Laval, p. 95–100 (cf. p. 59).

VON LABAN, R. et ULLMANN, L. (2011). *The mastery of movement*. 4th. Dance books. ISBN : 978-1852731458 (cf. p. 167).

VOUNG, J. W., JHALA, R. et LERNER, S. (2007). « RELAY : static race detection on millions of lines of code ». In : *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, p. 205–214 (cf. p. 111).

WAEZ, M. T. B., DINGEL, J. et RUDIE, K. (2013). « A survey of timed automata for the development of real-time systems ». In : *Computer Science Review* 9, p. 1–26 (cf. p. 121).

WALD, A. (1947). *Sequential Analysis*. 1st. John Wiley et Sons (cf. p. 154).

WARD, J., LUKOWICZ, P., TROSTER, G. et STARNER, T. (2006). « Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.10, p. 1553–1567 (cf. p. 149).

WELSH, G. et FOXLIN, E. (2002). « Motion Tracking : No Silver Bullet, but a Respectable Arsenal ». In : *IEEE Computer Graphics and Applications* 22.6, p. 24–38 (cf. p. 51).

WILLIAMS, C. K. (1998). « Prediction with Gaussian processes : From linear regression to linear prediction and beyond ». In : *Learning in graphical models*. Springer, p. 599–621 (cf. p. 144).

WING, M. G., EKLUND, A. et KELLOGG, L. D. (2005). « Consumer-grade global positioning system (GPS) accuracy and reliability ». In : *Journal of forestry* 103.4, p. 169 (cf. p. 137).

WINER, D. (1999). *XML-RPC specification*. Rapp. tech. UserLand Software (cf. p. 92).

XIA, L., CHEN, C.-C. et AGGARWAL, J. K. (2012). « View invariant human action recognition using histograms of 3d joints ». In : *Computer vision and pattern recognition workshops (CVPRW), 2012 IEEE computer society conference on*. IEEE, p. 20–27 (cf. p. 168, 171).

XU, Y., WEAVER, J. B., HEALY, D. M. et LU, J. (1994). « Wavelet transform domain filters : a spatially selective noise filtration technique ». In : *IEEE Transactions on Image Processing* 3.6, p. 747–758 (cf. p. 158).

YANG, X. et TIAN, Y. (2014). « Effective 3D action recognition using EigenJoints ». In : *Journal of Visual Communication and Image Representation* 25.1. Visual Understanding and Applications

with RGB-D Cameras, p. 2–11. ISSN : 1047-3203. DOI : http://dx.doi.org/10.1016/j.jvcir.2013.03.001. URL : http://www.sciencedirect.com/science/article/pii/S1047320313000333 (cf. p. 171).

YE, G., CORSO, J. J., HAGER, G. D. et OKAMURA, A. M. (2003). « Vishap : Augmented reality combining haptics and vision ». In : *IEEE INTERNATIONAL CONFERENCE ON SYSTEMS MAN AND CYBERNETICS*. T. 4, p. 3425–3431 (cf. p. 179).

YOKOKOHJI, Y., HOLLIS, R. L. et KANADE, T. (1996). « What you can see is what you can feel-development of a visual/haptic interface to virtual environment ». In : *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*. IEEE, p. 46–53 (cf. p. 182).

YOU, S., NEUMANN, U. et AZUMA, R. (1999). « Orientation tracking for outdoor augmented reality registration ». In : *IEEE Computer Graphics and Applications* 19.6, p. 36–42 (cf. p. 137).

ZACHARATOS, H., GATZOULIS, C., CHRYSANTHOU, Y. et ARISTIDOU, A. (2013). « Emotion recognition for exergames using laban movement analysis ». In : *Proceedings of Motion on Games*. ACM, p. 61–66 (cf. p. 171).

ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.**, VAIRON, J., FRAUCIEL, L., HACHET, M., GUITTON, P. et DELMONT, R. (2007). « Réalité augmentée en exterieur : enjeux et état de l'art ». In : *2èmes journées de l'AFRV*. Marseille, France (cf. p. 65).

ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.** et MALLEM, M. (2008a). « Hybrid Localization System for Mobile Outdoor Augmented Reality Applications ». In : *The First International Workshops on Image Processing Theory, Tools and Applications*. Sousse, Tunisia : IEEE (cf. p. 65).

— (2008b). « On the Hybrid Aid-Localization for Outdoor Augmented Reality Applications ». In : *ACM Symposium on Virtual Reality Software and Technology*. Bordeaux, France (cf. p. 65, 140, 189).

— (2008c). « Toward an Inertial/Vision Sensor Calibration for Outdoor Augmented Reality Applications ». In : *the 2nd International Workshop on Mobile Geospatial Augmented Reality (REGARD)*. Sous la dir. de SPRINGER. Springer. Québec, Canada : Springer (cf. p. 65).

ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.**, VAIRON, J., FRAUCIEL, L., HACHET, M., GUITTON, P. et DELMONT, R. (2008d). « Outdoor Augmented Reality : State of the Art and Issues ». In : *Virtual Reality International Conference*, p. 177–187 (cf. p. 65, 189).

ZENDJEBIL, I. M., ABABSA, F., **DIDIER, J.-Y.**, LALAGÜE, E., DECLE, F., DELMONT, R., FRAUCIEL, L. et VAIRON, J. (2009). « Réalité Augmentée en Extérieur : Etat de l'Art ». In : *Technique et Science Informatiques (TSI), Réalité Virtuelle - Réalité Augmentée* 28.6-7/2009. Numéro spécial, p. 857–890 (cf. p. 65).

ZENDJEBIL, I., ABABSA, F., **DIDIER, J.-Y.** et MALLEM, M. (2010). « A GPS-IMU-camera modelization and calibration for 3D localization dedicated to outdoor mobile applications ». In : *Control Automation and Systems (ICCAS), 2010 International Conference on*. IEEE, p. 1580–1585 (cf. p. 65, 140, 142, 189).

ZENDJEBIL, I., ABABSA, F.-E., **DIDIER, J.-Y.** et MALLEM, M. (2011). « Large Scale Localization - For Mobile Outdoor Augmented Reality Applications ». In : *International Conference on Computer Vision Theory and Applications (VISAPP 2011)*. Vilamoura, Algarve, Portugal, p. 492–501 (cf. p. 137, 189).

ZHANG, Y., SREEDHAR, V. C., ZHU, W., SARKAR, V. et GAO, G. R. (2008). « Minimum lock assignment : A method for exploiting concurrency among critical sections ». In : *Languages and Compilers for Parallel Computing*. Springer, p. 141–155 (cf. p. 112).

ZHANG, Z. (1999). « Flexible Camera Calibration by Viewing a Plane from Unknown Orientations ». In : *International Conference on Computer Vision*. T. 1. Corfu, Greece, p. 666–673 (cf. p. 152, 181).

Architectures pour les logiciels de Réalité Augmentée : des concepts aux applications

Jean-Yves Didier

Résumé

Après la présentation de notre parcours, ce mémoire d'habilitation dresse l'état de nos propositions d'architectures logicielles pour la réalité augmentée. Ces recherches s'inscrivent dans un cercle vertueux où nous confrontons nos architectures à des applications concrètes qui, en retour, lèvent de nouvelles problématiques à traiter.

Comme toute application, un logiciel de réalité augmentée répond à des exigences non fonctionnelles d'une part et fonctionnelles d'autre part. Les premières sont tout d'abord traités sous les angles de la modularité et de la reconfigurabilité afin de répondre aux besoins d'une technologie considérée comme étant jeune, de la distribution des traitements au sein d'un réseau informatique, et enfin de la gestion des aspects concurrents et des contraintes temporelles. Ces propositions ont abouties à un *framework* à base programmation orientée composants nommé ARCS – *Augmented Reality Component System* – et une chaîne de spécification, vérification des contraintes temporelles nommée MIRELA – *MIxed REality LAnguage*.

La dernière partie traite des applications réelles pour lesquelles nous avons également traité les exigences fonctionnelles telles que le recalage et la localisation au travers d'un système de capteur hybride combinant GPS, centrale inertielle et caméra, la problématique de l'interprétation de la sémantique de la scène et les interactions multi-modales de type visuo-haptiques.

Mots clés : Architecture logicielle, réalité augmentée, génie logiciel, applications réparties, vérification de contraintes temporelles, localisation et recalage, interprétation, interaction homme-machine.

Abstract

This habilitation draws the state of our software architecture proposals for augmented reality. This research is part of a virtuous circle where we confront our architectures to concrete applications that, in turn, raise new problematic to deal with.

Like any application, an augmented reality software responds to both non-functional and functional requirements. The first ones are tackled through the aspects of modularity and reconfigurability to meet the needs of a recent technology, the distribution of computing resources within a network, and finally the management of concurrent aspects and temporal constraints. These proposals have resulted in a framework based on component oriented programming named ARCS – Augmented Reality Component System – and a stack of tools for specifying, checking constraints and prototyping applications called MIRELA – MIxed REality LAnguage.

The last part deals with the real applications for which we have also addressed the functional requirements such as registration and localisation through a hybrid sensor system combining GPS, inertial motion unit and camera, interpretation of the scene semantics and multi-modal interactions of visuo-haptic type.

Keywords: Software architecture, augmented reality, software engineering, distributed applications, time constraints checking, registration and tracking, interpretation, man-machine interaction.