Visual servoing for mobile robots navigation with collision avoidance and field-of-view constraints

Wenhao Fu

Université Evry Val d'Essonne

THÈSE

présentée pour obtenir le grade de

DOCTEUR DE UNIVERSITÉ EVRY VAL D'ESSONNE

par

Wenhao Fu

Équipe d'accueil : IRA2

École doctorale : Sciences & Ingénierie

Composante Universitaire : Informatique, Biologie Intégrative et Systèmes Complexes

Titre de la thèse :

Visual Servoing for Mobile Robots Navigation with Collision Avoidance and Field-of-View Constraints

soutenue le 18 avril 2014 devant le jury composé de :

M.	Alain	Pruski	Rapporteurs
	Cédric	Demonceaux	
	Malik	Mallem	Examinateurs
	Omar	Ait Aider	
	Hicham	Hadj-Abdelkader	Encadrant
	Etienne	Colle	Directeur de thèse

Acknowledgements

I want to thank all the people who have been with me in my PhD research. Firstly, I am grateful to my advisor Etienne Colle for his help and support throughout my PhD course. I also want to thank my supervisor Hicham HADJ-ABDELKADER for his advising of my research work. I would like also to thank all the members at Lab IBISC, especially in the IR2 team: Malik, Samir, Frédéric, Jean-Yves, Christophe, Bruno, Paul, Antonio, BOUCHER Maxime, Jean-Clément, JUBERT Maxime. I am also grateful to Professor Saïd Mammar for his help and support of my research academic activity throughout my PhD course. I would also like to acknowledge Florant and Sabine. The are responsible for their to assistant our research work in the laboratory. Finally, thanks to my parents and friends for their heed and support.

Abstract

This thesis is concerned with the problem of vision-based navigation for mobile robots in indoor environments. More recently, many works have been carried out to solve the navigation using a visual path, namely appearance-based navigation. The visual path is generated after a training step. During the navigation, the mobile robot tracks the trained visual path by visual servoing approaches from key to key images. Compared to the metric map-based navigation approaches, the appearance-based scheme can avoid environment modeling, loop closing problem and the time consuming of planning algorithms. However, using this scheme the robot motion is limited to the trained visual path. For safety navigation, the robot should have the ability to avoid obstacles during the navigation process. The potential collision can make robot deviate from the current visual path, in which the visual landmarks can be lost in the current field of view. To the best of our knowledge, seldom works consider collision avoidance and landmark loss in the framework of appearance-based navigation.

We outline a mobile robot navigation framework in order to enhance the capability of appearance-based method, especially in case of collision avoidance and field-of-view constraints. Our framework introduces several technical contributions. First of all, the motion constraints are considered into the visual landmark detection to improve the detection performance. Next then, we model the obstacle boundary using B-Spline by interpolating the convex polygonal chain of the boundary. The B-Spline representation has no accidented regions due to the convex polygonal chain and can generate a smooth motion for the collision avoidance task. Additionally, we propose an vision-based control strategy, which can deal with the complete target loss. Finally, we use spherical image to handle the case of ambiguity and infinity projections due to perspective projection. The real experiments demonstrate the feasibility and the effectiveness of our framework and methods.

Contents

List of Figures

List of Tables

Synthèse en Français

Introduction

Objectif de la thèse

Notre travail de recherche vise à développer un système de robot mobile autonome pour aider les personnes agées ou handicapées dans des situations quotidiennes. Notre domaine de recherche se concentre sur les scénarios à usage domestique. Nous voulons concevoir un robot mobile qui peut de manière autonome et en toute sécurité passer d'une pièce à l'autre en utilisant des repères visuels dans l'environnement. Notre environnement est marqué avec des repères visuels. L'environnement est présenté par un ensemble d'images clés jouant le rôle repères visuels. Ces images sont stockées dans la mémoire du robot en tant que base de données. Nous supposons que le robot a une caméra fixe, un laser 2D à balayage et une odométrie. Le robot mobile peut naviguer entre les repères visuels en utilisant des approches de contrôle basées sur la vision en présence d'obstacles.

Nous nous concentrons sur le défi de chaque étape de la tache de navigation comme décrit dans la figure 1. Notre champ d'application est de faire naviguer le robot jusqu'à un repère visuel tout en considérant l'espace libre de collisions, la contrainte de champ de vision limité ainsi que les contraintes non holonomes. A chaque étape de la tache de navigation, trois tâches principalement sont considérées: (1) la détection et le suivi des repères visuels pour naviguer entre eux. Le mouvement du robot est contrôlé par des approches basées sur la vision. Cette tâche entraîne le robot vers un but désiré. Ainsi, il est appelée tache orientée objectifs. (2) détection d'obstacle et d'évitement, dans laquelle la sécurité de mouvement du robot est considérée. (3) l'estimation de la cible quand il est perdu après l'exécution de la tache d'évitement d'obstacles, afin de retrouver la cible après évitement d'obstacle.

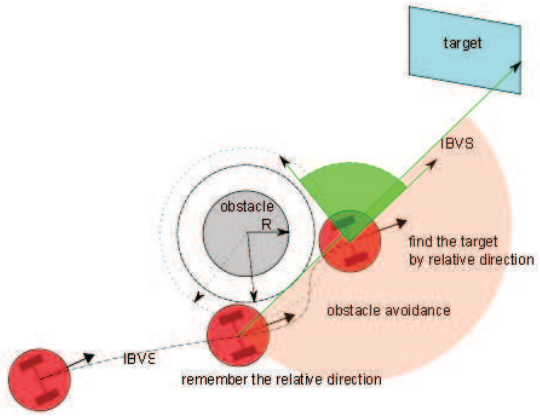


Figure 1: Illustration des défis de navigation basée apparence.

Contributions

Dans le cadre des approches de navigation par leur apparence, robots mobiles sont généralement limitées à la voie visuelle. Cependant, les obstacles inconnus dynamiques peuvent faire que les robots s'écartent du chemin visuel formé de sorte que les repères visuels peuvent être perdus dans le champ de vision de la caméra. Finalement, le chemin visuel est totalement perdu et la tache de navigation échoue. Il y a quelques travaux qui prennent en compte l'évitement d'obstacles et la perte complète du chemin visuel pendant la navigation basée sur la vision. Certains travaux utilisent un système redondant combinant le controleur basé sur la vision avec le contrôleur d'évitement d'obstacles pour maintenir la cible visuelle visible pendant l'évitement d'obstacle. Certains travaux utilisent le système de caméra catadioptrique central pour augmenter le champ de vision du système de la caméra. Cependant, ces travaux ne essentiellement résout pas le problème de la perte complète de cible visuelle. Folio et al. propose un procédé utilisant l'algorithme d'estimation visuelle de données. Il utilise les mesures visuelles antérieures et les entrées de commande pour estimer les caractéristiques visuelles courantes quand elles sont perdus. La principale contribution de cette thèse est la solution pour gérer la perte totale de repère visuel causé par l'évitement d'obstacles pendant la navigation basée sur la vision. Dans les détails, les domaines suivants sont des contributions originales:

- Présenter une approche RANSAC deux-points pour le calcul de l'homographie d'un objet plan posé verticalement. L'approche de détection d'objet est utilisée pour initialiser une approche efficace de suivi visuel. La méthode de détection etde suivi de détection et de suivi visuel peut détecter un objet à partir d'une

grande distance et la suivre de façon robuste en temps réel.

- Proposer une détection d'obstacle, sa représentation et son évitement efficaces. La représentation B-spline proposée permet un mouvement fluide lors de l'évitement d'obstacles.
- Développer une approche de l'asservissement visuel basé homographie qui peut traiter la perte de cible en raison de contraintes de champ de vision et les contraintes de sécurité de collision.
- Etendre l'utilisation de l'image sphérique à l'asservissement visuel pour traiter le problème de cible perdue.

Chapitre 1
Fonction de Sélection pour la Détection d'Objets

De nombreux détecteurs et de descripteurs de caractéristiques ont été proposées dans la littérature. Afin de décider de la méthode d'extraction de caractéristiques appropriée pour notre application, nous évaluons certaines approches prometteuses de détection, la description et la détection de caractéristiques dans ce chapitre. Nous utilisons notre propre ensemble de données en tenant compte de critères liés à notre application pour l'évaluation.

1.1 Expérience pour la sélection d'entité

Dans la littérature, SIFT est absolument la méthode la plus utilisée avec SURF. SURF, similaire à SIFT, efficace d'un point de vue calculatoire. MSER est particuliérement robuste pour les changements de point de vue et d'éclairage d'après [106]. BREF, ORB, BRISK et FREAK sont la récente vague d'extracteurs qui garantit la robustesse tout en réalisant unefficacité élevée de calcul. Par conséquent, nous avons comparé ces méthodes.

Dans les applications de robotique d'intérieur, la performance de ces caractéristiques locales pour les changements d'échelle, point de vue et le flou de mouvement est le principal facteur pris en considération. Ainsi, l'évaluation de ces caractéristiques des détecteurs et des descripteurs est réalisée avec trois ensembles de données: le changement d'échelle, le changement de point de vue, et de flou de bougé. On compare la qualité de l'extraction et le temps de calcul par les critères suivants:

- **temps par image**: temps total en millisecondes passé pour l'extraction de caractéristiques d'un cadre unique.
- **fois par keypoint**: temps d'extraction pour un seul point-clé. Il est évalué par le temps d'extraction totale divisée au nombre de points-clés détectés.
- **nombre de vrais appariement**.
- **Pourcentage de vrais appariements**: évalué comme le nombre de vrais apparienments divisé par le nombre de points-clés minimal des images de référence et de la scène.

Les expériences montrent que SIFT fournit toujours un nombre de vrais appariements suffisant même en présence de changement de point de vue et d'échelle. Le pourcentage de vrais appariement est élévé. Cependant, les SIFT, SURF et MSER ne conviennent pas pour des applications temps réels. BRISK, BREF, ORB ainsi que FREAK satisfont à l'exigence temps réel. Afin de gérer la détection à grande distance dans notre application, nous choisissons le détecteur et le descripteur SIFT. Afin d'assurer la performance en temps réel de suivi visuel, nous utilisons le résultat de détection pour initialiser une méthode de suivi basée sur un modèle.

1.2 Contrainte RANSAC pour Homographie Estimation

Dans nos expériences de détection planaire de repère visuel, des résultats inattendus peuvent se produire comme illustré dans la figure 2. Sur la gauche de la figure est l'image de référence, dans lequel une affiche est l'objet. Sur la droite est l'image de la scène. Les matchs sont représentés par des lignes droites. Les lignes vertes représentent les points d'intérêt satisfaisant à une transformation d'homographie, qui est choisie par l'algorithme RANSAC 4-points. Les lignes roses représentent les valeurs aberrantes, qui satisfont pas l'homographie estimée. Le résultat correspondant montrent de toute évidence que l'algorithme standard de RANSAC 4-points pour l'estimation de homographie peut sélectionner un modèle faux. Par conséquent, les véritables "inliears" ne sont pas sélectionnés. C'est parce que l'algorithme de RANSAC fonctionne de façon aléatoire et choisit un modèle avec une erreur minimum. Et l'homographie code le mouvement de la caméra entre deux points de vue et la structure de l'objet plan. L'algorithme RANSAC 4-points considère un mouvement de caméra à 6 DDL par rapport à un objet plan orienté de manière arbitraire.

Cependant, dans notre application de l'environnement intérieur, le robot se déplace dans un plan. Pour notre configuration caméra-robot, la caméra est fixée sur le robot

Figure 2: Le problème de la 4-Points RANSAC pour la d'etection d'objet.

mobile et son mouvement est parallèle au sol. Ainsi, le mouvement de la caméra dispose de 3 DoF, deux translations et une rotation. L'objet planaire est placé en parallèle avec la paroi et à la verticale au sol. Par conséquent, le vecteur normal de l'objet plan est également parallèle au sol. Afin d'améliorer le résultat de détection, nous considérons le mouvement contraint de la caméra et l'orientation contrainte de l'objet plan pour formuler une homographie contrainte. Afin d'améliorer le résultat de la détection, on utilise l'algorithme RANSAC pour estimer l'homographie contrainte dans l'étape d'appariement.

Considérent que le mouvement de la caméra et son Plan $x-z$ sont à la fois parallèle au sol. Le vecteur normal de l'objet plan est parallèle au sol. Par conséquent, l'homographie est:

$$\begin{aligned}\mathbf{H} &= \mathbf{R} + \frac{\mathbf{t}}{d}\mathbf{n}^\top \\ &= \begin{bmatrix} \cos\theta + \frac{t_x n_x}{d} & 0 & \sin\theta + \frac{t_x n_z}{d} \\ 0 & 1 & 0 \\ -\sin\theta + \frac{t_z n_x}{d} & 0 & \cos\theta + \frac{t_z n_z}{d} \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & 0 & h_{13} \\ 0 & h_{22} & 0 \\ h_{31} & 0 & h_{33} \end{bmatrix}. \end{aligned} \tag{1}$$

Pour chaque correspondance $(\mathbf{x}_1, \mathbf{x}_2)$, nous avons $\mathbf{x}_2 = \mathbf{H}\mathbf{x}_1$, ce qui est équivalent à $\mathbf{x}_2 \times \mathbf{H}\mathbf{x}_1 = \mathbf{0}$. Si nous notons l'$j^{\text{th}}$ row de $\mathbf{H}$ en utilisant un vecteur ligne $\mathbf{h}_j^T$, tous les éléments de $\mathbf{H}$ présenté par un vecteur $\mathbf{h} = [\mathbf{h}_1^\top \ \mathbf{h}_2^\top \ \mathbf{h}_3^\top]^\top$. Le produit croisé peut être écrit comme la forme de la matrice $\mathbf{A}\mathbf{h} = \mathbf{0}$, où $\mathbf{A}$ est la matrice des paramètres. Considérant l'équation (1), le vecteur $\mathbf{h}$ est réduite à $\mathbf{h}_r = [h_{11}\ h_{13}\ h_{22}\ h_{31}\ h_{33}]^\top$. Donc

la matrice **A** sera réduite à

$$\mathbf{A}_r = \begin{bmatrix} 0 & 0 & -y_1 z_2 & x_1 y_2 & z_1 y_2 \\ x_1 z_2 & z_1 z_2 & 0 & -x_1 x_2 & -z_1 x_2 \end{bmatrix} \tag{2}$$

Le vecteur **h** a cinq entrées. Un appariement donne deux équations. Par conséquent, seuls deux appariements sont assez pour résoudre **h** à un facteur d'échelle près. Par conséquent, nous avons adapté l'estimation de homographie avec seulement 2 points compte tenu des contraintes de mouvement de la caméra et le vecteur normal de l'objet plan. Le résultat expérimental de l'algorithme RANSAC 2-Points est illustré sur la figure 3.

Figure 3: Le résultat expérimental de l'algorithme RANSAC 2-Points.

Chapitre 2
Evitement Réactif Temps Réel d'Obstacles

2.1 Introduction

Le déplacement en toute sécurité du robot dans un environnement réaliste est une des fonctionnalités essentielles d'un robot. La détection d'obstacles, la représentation ainsi que l'évitement sont des taches cruciales pour la navigation de robot mobile en milieu intérieur et extérieur. Ce chapitre présente l'évitement d'obstacle de notre robot dans un environnement intérieur inconnu notre robot mobile dans un environnement inconnu, à partir de données fournies par un laser 2D. Les méthodes proposées peuvent obtenir un mouvement sans collision pendant la navigation du robot mobile.

2.2 Détection d'Obstacles et Représentation

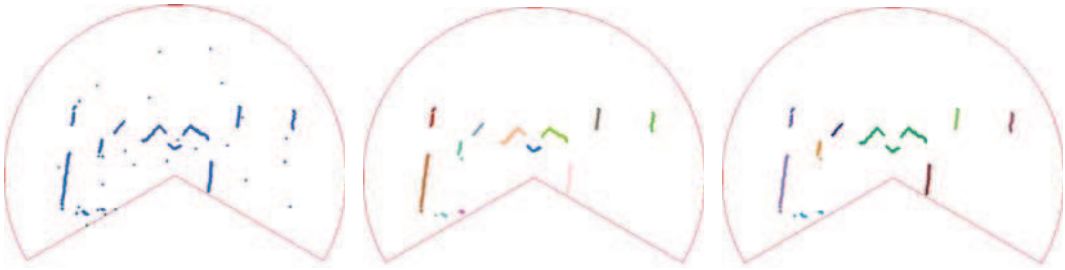
Dans cette section, nous présentons nos approches de détection d'obstacles et de représentation basée sur les mesures d'un laser plan. Contrairement à d'autres procédés de représentation

d'obstacle, on analyse la forme convexe et concave de la limite des obstacles et génère une courbe lisse par rapport à la limite convexe.

2.2.1 Détection d'Obstacles

La détection d'obstacles commence habituellement par la segmentation puis la fusion de points, comme indiqué dans la figure 4. La collision est ensuite vérifiée par calcul de prédiction de mouvement.

Afin de distinguer des objet différents, nous segmentons les balayages laser dans des grappes de points adjacents. Chaque groupe correspond à un contour visible d'un objet détecté. Le Successive Edge Following (SEF) algorithme [144] est utilisé dans le processus de segmentation. Dans le processus de segmentation, nous éliminons les points isolés considérés comme du bruit. La figure4(b) montre différents segments de couleurs. À cause du bruit ou d'occlusion, un contour de l'objet peut être divisé en plusieurs segments adjacents. Afin de faire face à cette question, nous fusionnons les segments proches dans un grand segment en utilisant l'écart angulaire entre deux points des deux segments les plus proches. La figure 4(c) montre le résultat de la fusion finale.



(a) Les données brutes laser (b) Segmentation des données laser (c) Fusion des données laser

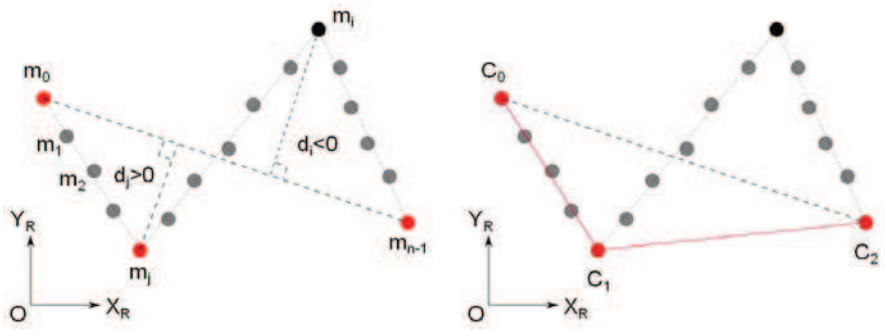
Figure 4: Segmentation et fusion des données laser.

2.2.2 La Représentation

La représentation des obstacles vise à trouver un modèle de la frontière de l'obstacle détecté. Dans cette section, nous allons modéliser la limite d'obstacle sur la base de lignes polygonales.

Représentation d'une chaine polygonale convexe Les chaines polygonales peuvent être utilisées pour se rapprocher d'autres courbes [130] et formes planes [142]. Elle peuvent également être utilisés pour rapprocher les contours d'obstacles. Cependant,

les sommets concaves sont inadéquates pour l'évitement d'obstacles lorsque le robot mobile va le long de la chaîne polygonale. Parce que le robot peut se déplacer dans certaines parties concaves qui ne sont pas nécessaires, la trajectoire n'est pas lisse. Afin d'assurer un mouvement fluide, la convexité de la chaîne polygonale extraite doit être satisfaite. Nous utilisons une zone du triangle signé dans l'algorithme split-et-fusion pour extraire uniquement la courbe convexe. Notre idée est similaire à Han [61]. La figure 5 illustre l'extraction convexe de la chaîne polygonale.



(a) (b)

Figure 5: Extraction de courbe convexe.

Approximation par des B-splines Pour obtenir une approximation de la représentation lisse de la limite de l'obstacle, on représente la limite d'obstacle par une courbe B-spline cubique paramétrique en interpolant les sommets de la chaîne polygonale convexe. L'avantage d'utiliser les courbes B-splines, est qu'elles permettent une continuité d'ordre arbitraire pour représenter les frontières de l'obstacle, ce qui est très approprié pour le contrôle de mouvement de robot pendant l'évitement d'obstacles. En outre, les B-splines peuvent s'adapter à n'importe quelle courbe définie par les points de contrôle, et sa production est intuitive. En outre, ce sont des polynômes par morceaux, de sorte qu'ils peuvent effectuer une bonne approximation avec un faible degré, tout en évitant l'instabilité due à un phénomène Runges. La figure 6 illustre un résultat interpolé (dessiné en vert) à partir d'une analyse réelle de relevé appliquée à la chaîne polygonale sélectionnée (dessinée en rouge) correspondant à l'obstacle à éviter par le robot mobile.

Figure 6: Sélection d'obstacles et raccord aux limites

2.3 Approches d'évitement d'obstacles réactif

Soit $\mathbf{u} = [\upsilon\ \omega]^\top$ comme entrées de commande pour le système de robot. υ et ω sont respectivement les vitesses linéaires et angulaires (positifs dans le sens antihoraire) du robot. Par conséquent, le modèle cinématique monocycle est donné par

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \upsilon \\ \omega \end{pmatrix} \tag{3}$$

Nous considérons le suivi de chemin pour traiter l'évitement d'obstacle suivant. Deux enveloppes ξ_0 et ξ_+ sont générées en étendant l'enveloppe ξ_{obs} comme illustré dans la figure 7. Soit ξ_{obs} représentant le rapprochement de la frontière de l'obstacle. ξ_0 et ξ_+ entourent l'obstacle à la distance $|d_0| < |d_+|$ représentant le risque de collision. ξ_+ définit une zone à l'intérieur de laquelle l'obstacle est détecté. Nous supposons que la distance entre deux obstacles est plus grande que $2|d_+|$, ce qui assure que le robot traite un obstacle à chaque fois. ξ_0 est le chemin de référence à suivre lorsque le robot contourne l'obstacle. Un cadre de référence $\mathcal{F}_0$ est fixé à ξ_0 et le vecteur d'erreur (δ, α, χ) est ensuite calculé. (Voir la figure 7 pour les composants de vecteur d'erreur.)

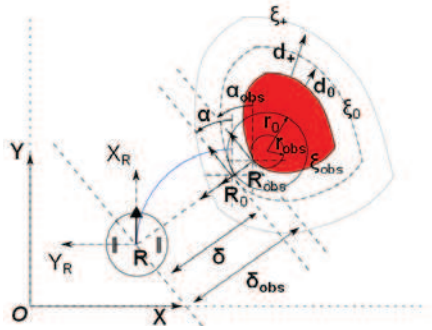


Figure 7: Évitement des collisions en utilisant le suivi de chemin

Afin de contrôler le robot suivant le chemin ξ_0, nous utilisons la loi de commande proposée dans la littérature [136]. La loi de commande peut être décrite par une mise en vitesse linéaire non nulle et une vitesse angulaire

$$\omega = -\upsilon(k\delta + \alpha + 2k\sin\alpha - \chi\cos\alpha) \tag{4}$$

où k est un gain positif à ajuster. La vitesse linéaire peut être une valeur constante ou peut être un profil souhaité.

Naturellement, la tache d'évitement est fusionnée avec une tache principale (chemin suivant, asservissement visuel, ... etc). Folio et al. dans [50] propose une méthode de commande hybride pour fusionner l'asservissement visuel avec des taches d'évitement d'obstacle au niveau de la commande. Ici, nous construisons un contrôleur similaire à celui présenté dans [50]. Soit $\dot{\mathbf{q}}_{go}$ est la loi de commande pour la tache orientée objectifs, et $\dot{\mathbf{q}}_{glb}$ est l'entrée de commande d'évitement de collision. Le contrôleur $\dot{\mathbf{q}}_{glb}$ peut être défini par

$$\dot{\mathbf{q}}_{glb} = (1-\lambda)\dot{\mathbf{q}}_{go} + \lambda\dot{\mathbf{q}}_{co}, \quad \lambda \in [0,1] \tag{5}$$

où λ est une fonction permettant de basculer entre les deux contrôleurs. Afin de faciliter le contrôle global, la limite ξ_+ est prise en compte pour définir la zone dangereuse de collision et changer progressivement λ comme indiqué dans [50]. Le λ peut être modifié facilement

$$\lambda = \begin{cases} 0, & \text{if} \quad |\delta_{obs}| \geq d_+ \\ \frac{d_+ - |\delta_{obs}|}{d_+ - d_0}, & \text{si} \quad d_0 \leq |\delta_{obs}| < d_+ \quad \text{et} \quad \text{évasion} = \text{faux} \\ \frac{d_+ - |\delta_{obs}|}{d_+ - d_e}, & \text{si} \quad d_e \leq |\delta_{obs}| < d_+ \quad \text{et} \quad \text{évasion} = \text{vrai} \\ 1, & \text{autrement} \end{cases} \tag{6}$$

La condition d'échappement " évasion = vrai " est satisfaite lorsque l'obstacle n'est plus dangereux pour le robot, à savoir quand

$$\begin{cases} \text{le robot évite l'obstacle dans le sens antihoraire et } \omega_{go} < \omega_{co} \\ \textit{ou} \text{ le robot évite l'obstacle dans le sens horaire et } \omega_{go} > \omega_{co} \end{cases} \tag{7}$$

2.4 Résultats Expérimentaux

Nous avons mis en œuvre la détection d'obstacle proposée, la représentation et la méthode d'évitement sur notre plate-forme de robot mobile nommée Lina. Le robot mobile est équipé d'un télémètre laser de Hokuyo Ltd., qui a une portée maximale

de mesure de 5 m et est capable de balayer un angle de 240°. Dans cette expérience, nous utilisons un schéma de contrôle d'asservissement visuel pour faire naviguer le robot vers une cible donnée. La tache de navigation est fusionnée avec l'approche d'évitement d'obstacles proposé. Les obstacles sont placés entre la position initiale du robot mobile et une souhaitée.

Trois positions du robot sont représentées dans la figure 8 des cercles rouges avec un nombre de 1 à 3. Les croix noires sont les données laser. La ligne bleue est la trajectoire du robot. Deux B-Spline tracées en vert modélisant la limite d'obstacles sont indiquées par rapport à la position 1 et 2. La figure 9 montre le résultat dans l'espace du laser aux trois positions. Au début, le robot est en dehors de la zone dangereuse des obstacles. Comme il va de l'avant contrôlé par asservissement visuel, il rencontre un obstacle (position 1). L'obstacle est alors segmenté, et la courbe convexe est extraite. Ensuite, l'obstacle segmenté est modélisé par la courbe B-spline (voir la figure 9(a)). Le contrôleur global commute progressivement à l'unité de commande d'évitement d'obstacle, ce qui fait passer le robot autour de l'obstacle en douceur. Lorsque le robot peut atteindre l'objectif sans collision (position 2), le contrôleur global revient au contrôleur de l'asservissement visuel, et le robot s'arrête lorsque l'objectif est atteint (position 3). Le résultat expérimental montre que le robot peut se déplacer en douceur sans être piégé dans la partie concave de l'obstacle.

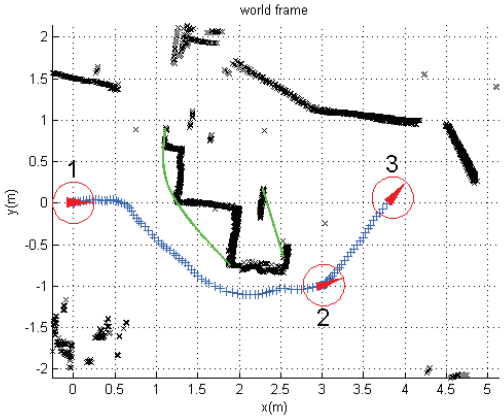
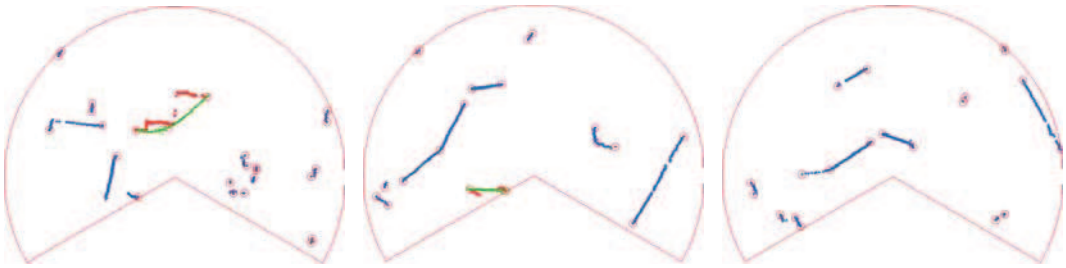


Figure 8: Résultat expérimental: navigation



(a) position 1 (b) position 2 (c) position 3

Figure 9: Résultat expérimental: données laser

Chapitre 3
Contrôle basé sur la Vision pour les Robots Mobiles

Ce chapitre examine le problème de l'asservissement visuel d'un robot mobile à entraînement différentiel. Le but est de positionner le robot vers une position souhaitée à l'aide de l'asservissement visuel tout en considérant trois contraintes: (1) Les contraintes de champ de vision (FOV) imposées par le système de caméra; (2) des contraintes imposées par la non-holonomie cinématique du robot; (3) les contraintes de sécurité imposées par les environnements encombrés. Les schémas proposés sont testés dans les taches de navigation d'un robot mobile réel.

3.1 Asservissement Visuel Pour Robot Mobile

Nous considérons un robot mobile à deux roues motrices équipé d'une caméra non orientable, un télémètre laser 2D et un odomètre. Soit $\mathbf{v}_c = [\upsilon_x\ \upsilon_y\ \upsilon_z\ \omega_x\ \omega_y\ \omega_z]^\top$ soit le torseur cinématique de la caméra, et $\mathbf{v}_r = [\upsilon\ \omega]^\top$ l'entrée de commande du robot mobile. Pour cette configuration, ils sont liés comme

$$\mathbf{v}_c = \mathbf{J}\,\mathbf{v}_r \tag{8}$$

avec

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_\upsilon & \mathbf{J}_\omega \end{bmatrix} \tag{9}$$

où$\mathbf{J}_\upsilon$ et $\mathbf{J}_\omega$ sont les première et seconde colonnes de $\mathbf{J}$ respectivement.

Les caractéristiques visuelles de contrôle basé sur la vision sont définies sur le plan de l'image et la tache de commande s'effectue directement dans le plan de l'image. Soit $\mathbf{L_s}$ la matrice d'interaction liée à un point. Tenant compte de la jacobienne robot-système

de vision donnée dans l'équation (8), nous obtenons

$$\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s},\upsilon}\,\upsilon + \mathbf{L}_{\mathbf{s},\omega}\,\omega \tag{10}$$

où $\mathbf{L}_{\mathbf{s},\upsilon} = \mathbf{L}_{\mathbf{s}}\mathbf{J}_{\upsilon}$ et $\mathbf{L}_{\mathbf{s},\omega} = \mathbf{L}_{\mathbf{s}}\mathbf{J}_{\omega}$. Pour une vitesse linéaire υ, la vitesse angulaire peut être obtenue sous la forme

$$\omega = -\mathbf{L}^{+}_{\mathbf{s},\omega}(\lambda\mathbf{e} + \mathbf{L}_{\mathbf{s},\upsilon}\,\upsilon) \tag{11}$$

Notez que la vitesse linéaire υ peut être affectée à une valeur constante ou calculée à partir d'un profil de vitesse défini.

3.2 Asservissement Visuel Based Navigation avec Perte Complète de Cible

3.2.1 Énoncé du problème

Lorsque le robot exécute la navigation basée sur la vision dans un environnement inconnu et encombré, il est nécessaire non seulement de préserver la visibilité des caractéristiques de l'image lors de l'exécution de l'asservissement visuel mais aussi de prévenir le robot mobile de la collision avec les obstacles de la manière décrite dans le deuxième chapitre. Toutefois, lorsque le robot contourne les obstacles, la cible peut être perdue en raison du fait que la caméra est fixe et que son champ de vision est limitée. Dans cette situation, l'asservissement visuel échouera. C'est un problème difficile à résoudre dans le cadre de l'asser asservissement visuel basé sur l'image et la navigation basée apparence. Dans cette section, nous nous concentrons sur le problème de la perte de cible visuelle dans le champ de vision de la caméra. Inspiré par l'intégration de chemin et la réaction à des repères visuels dans la navigation des animaux, nous proposons une stratégie pour faire face à la perte de caractéristiques visuelles en profitant des données odométriques.

3.2.2 Cadre de Notre Système de Navigation

Notre système de navigation basée sur la vision peut être décomposé en quatre grands blocs (voir la figure 10):

– Génération de comportements: ce bloc génère un comportement adapté à la génération de mouvements et de perception. Le comportement généré est une tache prédéfinie pour ces deux composants. Il fonctionne comme une machine

d'état, qui décompose une tache de navigation en plusieurs comportements de base (voir la figure 10). Ces comportements de base comprennent le suivi de cible visuelle et l'asservissement, la détection de collision, l'évitement d'obstacles, la réacquisition de cible, et l'arrêt du robot. La machine d'état va sélectionner une tache en raison des résultats de perception actuels et le comportement précédent.

– Génération de mouvements: cette composante se compose de plusieurs contrôleurs, y compris le contrôleur par objectifs basée sur l'asservissement visuel, le contrôleur d'évitement d'obstacle sur la base des données du télémètre laser et le contrôleur de réacquisition de la cible. Le contrôleur d'asservissement visuel entraîne le robot mobile en direction de la configuration souhaitée. Le dispositif de commande d'évitement d'obstacles assure la sécurité de mouvement en présence d'obstacles. Le contrôleur de réacquisition de la cible peut aligner le robot en direction de la cible perdue si nécessaire. Chaque contrôleur calcule un résultat à partir de la perception de l'information en cours. Les résultats de chaque contrôleur sont ensuite combinés pour entraîner le robot mobile à réaliser la configuration souhaitée tout en assurant l'évitement de collision. La combinaison est calculée avec le comportement donné.

– Perception & Estimation: la composante de perception et estimation traite les données provenant de différents capteurs afin de détecter et de suivre les repères visuels, et de détecter les obstacles qui approchent. Les résultats du traitement sont ensuite utilisés par d'autres composants. Les résultats de suivi visuel sont utilisés pour le contrôle de l'asservissement visuel. La détection de collision et la modélisation d'obstacles sont utilisées pour invoquer la tâche d'évitement d'obstacles.

– Capteurs & Actionneurs: Ce composant fournit un moyen d'interagir avec l'environnement. Notre système de robot mobile comprend un système stéréo de vision, un télémètre laser et deux actionneurs. Nous utilisons la vision mono pour détecter et suivre la cible visuelle. Nous utilisons le système de vision stéréo pour estimer le mouvement du robot, pour simuler les données odomètriques, les codeurs du système odométrique du robot étant déficients. Le télémètre 2D est utilisé pour détecter et modéliser les obstacles.

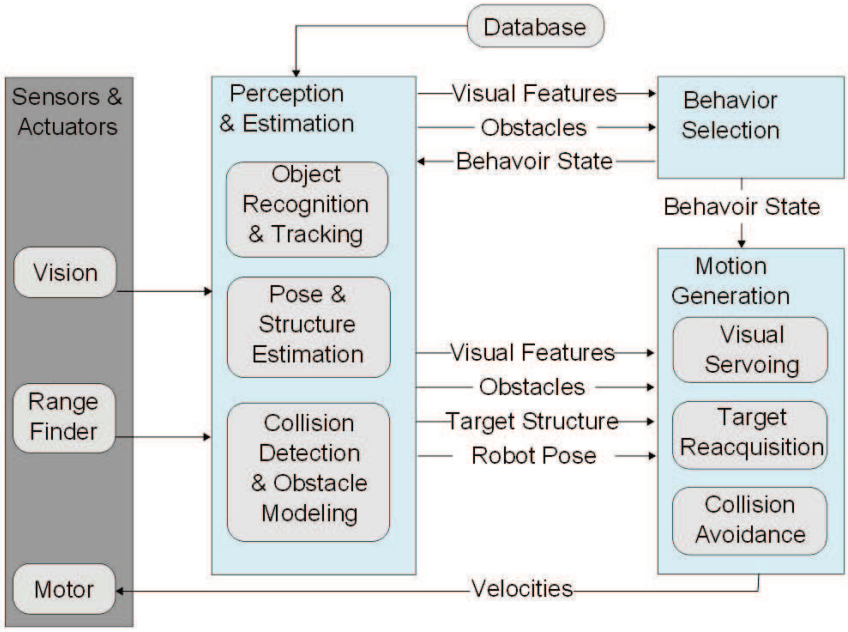


Figure 10: Notre système de navigation.

3.2.3 Algorithme de génération de comportements

La composante génération de comportement est conçue sur la base des exigences pour accomplir la tache de navigation. Notre tache de navigation, est que le robot se déplace vers une configuration désirée tout en évitant les obstacles. Pour accomplir la tache de navigation, nous concevons six comportements de base: comportement détection d'objets, le comportement d'attraction par le but, le comportement de prédiction de mouvement, le comportement d'évitement d'obstacle, le comportement d'estimation de données visuelles et le comportement de ré-acquisition de cible. La génération de comportement fonctionne comme une machine d'état. Elle sélectionne le comportement actuel approprié en raison des résultats de la perception actuelle et le comportement précédent. Le comportement de détection d'objet est toujours au début de la tache de navigation. Le résultat de détection est d'initialiser le comportement d'attraction par le but en utilisant le contrôle de l'asservissement visuel. Pendant le comportement d'attraction par le but, si une collision potentielle est détectée, éviter les obstacles et les comportements d'estimation de mouvement sont invoquées. Le robot mobile contourne l'obstacle détecté et estimé la cible dans l'espace de l'image en cas de perte. Lorsque l'obstacle est évité, le comportement de ré-acquisition de la cible est exécuté pour aligner le robot en direction de la cible, alors le comportement de détection d'objet est

redémarré.

3.2.4 Algorithme de génération de mouvements

Compte tenu du comportement sélectionné, le composant de génération de mouvement est dédié à la conception des contrôleurs et calcule la vitesse angulaire et linéaire pour le mouvement. Il y a trois contrôleurs: le contrôleur de l'asservissement visuel, le contrôleur d'évitement d'obstacles et le contrôleur objet ré-acquisition. Le contrôleur de l'asservissement visuel est décrit au début de ce chapitre. Le contrôleur d'évitement d'obstacle est introduit dans le deuxième chapitre. Ici, nous concevons le comportement de ré-acquisition de la cible.

Le comportement de ré-acquisition de la cible peut orienter le robot en direction du centre de la cible afin d'amener cette extrémité dans le champ de vision. Soit $\Delta\theta$ l'erreur angulaire, qui est la différence entre les orientations courante et souhaitée. L'orientation désirée est donnée par rapport à la position lorsque le robot fait face au centre de la cible visuelle. Le contrôleur de cap est

$$\omega_h = -\lambda_h \Delta\theta, \tag{12}$$

où λ_h est un gain positif à régler.

L'erreur $\Delta\theta$ peut être calculée en utilisant les points visuels estimés dans le plan de l'image normalisée. L'objectif principal de la ré-acquisition de la cible est alors de déplacer la projection de l'image du centre de masse (x_{mc}, y_{mc}) de la cible vers l'axe de y du plan de l'image normalisée. L'erreur angulaire est définie comme

$$\Delta\theta = \arctan(-x_{mc}). \tag{13}$$

L'estimation des données visuelles sera présenté dans la section suivante.

3.2.5 L'estimation des données visuelles en cas de Perte Cible

Si la cible est en dehors du champ de vision de la caméra, le contrôleur d'asservissement visuel échoue. Pour résoudre ce problème, nous proposons une stratégie prenant l'avantage des techniques d'intégration de chemin et des techniques de reconnaissance de la réaction de cible visuelle. Une cible plane observée dans deux images définit une relation d'homographie. Si nous connaissons la cible dans une image de référence et la transformation d'homographie entre l'image de référence et l'image courante, la cible peut être récupérée dans l'image courante. Si la cible est hors du champ de vue actuel, nous

pouvons toujours le récupérer virtuellement. La stratégie est illustrée à la figure 11. Le travail principal est d'estimer la transformation de la caméra ($\mathbf{R}$, $\mathbf{t}$) et la structure de la cible plane ($\mathbf{n}$, d). Dans la suite, nous allons introduire la méthodes de calcul de ces paramètres.

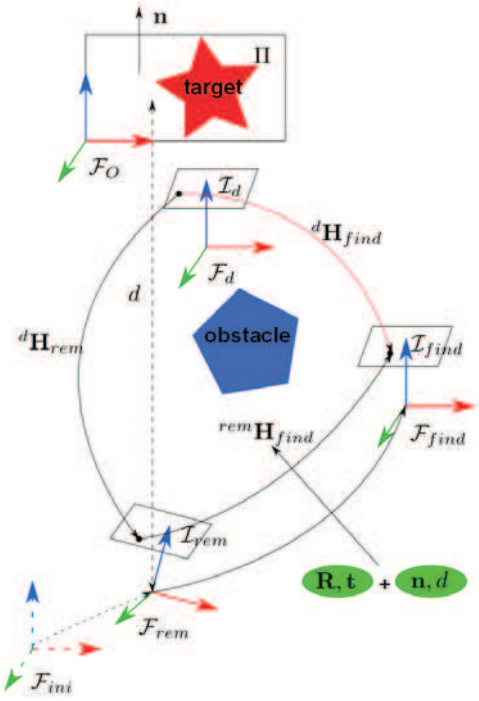


Figure 11: Suivi de l'homographie en cas d'invisibilité de la cible

Estimation du mouvement de la caméra

Le mouvement de la caméra peut être estimé en utilisant les systèmes de vision stéréo comme décrit dans l'annexe B.6 intégrant les mesures odométriques. Lorsque on considère un système monovision, la rotation de la caméra peut être récupérée et la direction de la translation obtenue. Sans perte de généralité, nous décrivons le mouvement estimé de la caméra tel que ($\hat{\mathbf{R}}$, $\lambda\hat{\mathbf{t}}$), où $\lambda = 1$ si l'odométrie est utilisée.

Estimation de la structure planaire à partir du suivi visuel

Lorsque la cible est visible, le suivi fournit la relation d'homographie $\mathbf{H}$ entre le repère actuel de la caméra et le précédent. La structure de la cible plane peut être calculée en décomposant l'homographie $\mathbf{H}$ dans ($\mathbf{R_H}, \mathbf{t_H}, \mathbf{n}$). Notez que le vecteur de translation $\mathbf{t_H}$ est normalisé par la distance au plan cible d. Le mouvement estimé de la caméra

peut être décrit comme ($\mathbf{R_H}$, $d\,\mathbf{t_H}$). La structure de la cible est ($\mathbf{n}$, d). La valeur d est obtenue en utilisant les résultats de l'estimation de mouvement. Considérant l'estimation de mouvement, nous avons $\lambda\,\hat{\mathbf{t}} = d\,\mathbf{t_H}$. puis

$$d = \lambda\,\hat{\mathbf{t}} \,/\, \mathbf{t_H}. \tag{14}$$

Si λ est inconnue, nous attribuons une valeur arbitraire. d est attribuée en conséquence. La structure ($\mathbf{n}$, d) est mémorisée avant la perte de cible.

Recherche de la cible visuelle

En utilisant la mémoire de la structure de l'objet ($\mathbf{n}_{rem}$, d_{rem}) et le mouvement estimé de la caméra ($\hat{\mathbf{R}}_{lost}$, $\lambda\hat{\mathbf{t}}_{lost}$) quand la cible perdue, l'homographie peut être retrouvée

$$\begin{aligned}\mathbf{H}_{lost} &= \hat{\mathbf{R}}_{lost} + \frac{\lambda\hat{\mathbf{t}}_{lost}}{d_{rem}}\mathbf{n}_{rem}^{\top} \\ &= \hat{\mathbf{R}}_{lost} + \frac{\mathbf{H}\mathbf{t}_{rem}\hat{\mathbf{t}}_{lost}}{\hat{\mathbf{t}}_{rem}}\mathbf{n}_{rem}^{\top}\end{aligned} \tag{15}$$

Nous pouvons noter que pour retrouver l'homographie $\mathbf{H}_{lost}$ perdue la distance d et le facteur d'échelle λ ne sont pas nécessaires. Tout ce qui est nécessaire est le mouvement estimé de la caméra ($\hat{\mathbf{R}}_{lost}$, $\lambda\hat{\mathbf{t}}_{lost}$) depuis la perte de la cible, la normale mémorisée de la cible plane, la transformation provenant de la décomposition de l'homographie [45] et l'estimation de mouvement avant la perte de cible. Le figure 12 illustre les résultats de la récupération de la cible perdue. Les quatre points de couleur sont les quatre coins de l'objet suivi. Les figures 12(b) à 12(h) sont les résultats de suivi utilisant l'estimation. Les figures 12(d) à 12(f) montrent que la stratégie proposée peut gérer la cible perdue.

3.2.6 Résultats Expérimentaux

Le figure 13 montre les étapes de navigation où les différentes positions sont données dans les cercles rouges et les chiffres (1-5). La trajectoire du robot mobile est tracée en bleu. Les obstacles détectés par le télémètre laser sont donnés dans la couleur noire. On suppose que la cible est visible par la caméra à la position de départ. Tout d'abord, la cible est détectée en comparant l'image en cours avec l'une de référence (acquis à la position désirée). Ensuite, le suivi visuel est initialisé à la position 1. Le robot mobile commence à atteindre l'objectif en utilisant le système de contrôle de l'asservissement visuel. Lorsque le robot se déplace à la position 2, un obstacle est détecté et l'évitement d'obstacle est invoqué. Et la structure du plan de référence (le vecteur normal et

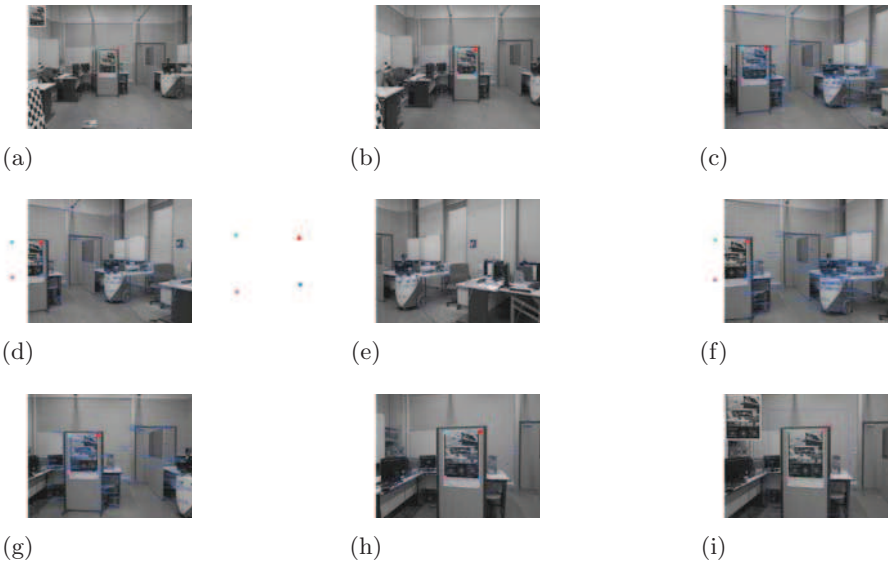


Figure 12: Récupération de la cible inconnue en estimant l'homographie.

sa distance par rapport au repère de la caméra) sont des estimations. Au cours de l'évitement de l'obstacle de la position 2 à 3, la position du robot mobile est récupérée en utilisant des données d'odométrie. Lorsque l'obstacle est totalement évité en position 3, le robot mobile est tourné vers la cible en utilisant la cible virtuelle estimée. Lorsque le robot arrive à la position 4, la cible est à peu près au centre du champ de vue. Puis l'asservissement visuel est redémarré pour atteindre l'objectif en position 5. La figure 14 montre des images et des données de laser à différentes étapes (1-5) comme décrit dans la figure 13. Une vidéo de l'expérience réelle est disponible à `http://aramis.iup.univ-evry.fr:8080/~hadj-abdelkader/Videos/Wenhao/video_MMAR2013.wmv`.

Conclusion

Cette thèse a présenté un ensemble cohérent d'algorithmes permettant à un robot mobile autonome de se déplacer dans un environnement intérieur en présence d'obstacles, en tout sécurité. Le robot peut détecter des images planes de référence, utilisées comme amers, à grande distance. La détection est basée sur l'appariement de points d'intérêt qui sont sont ensuite suivis à l'aide d'une méthode utilisant l'homographie. L'asservissement visuel est adopté pour contrôler le robot vers l'image de référence. Pendant le contrôle visuel, on considère les contraintes de sécurité imposées par l'environnement et le champ de vision imposé par la caméra, ainsi que les contraintes imposées par la

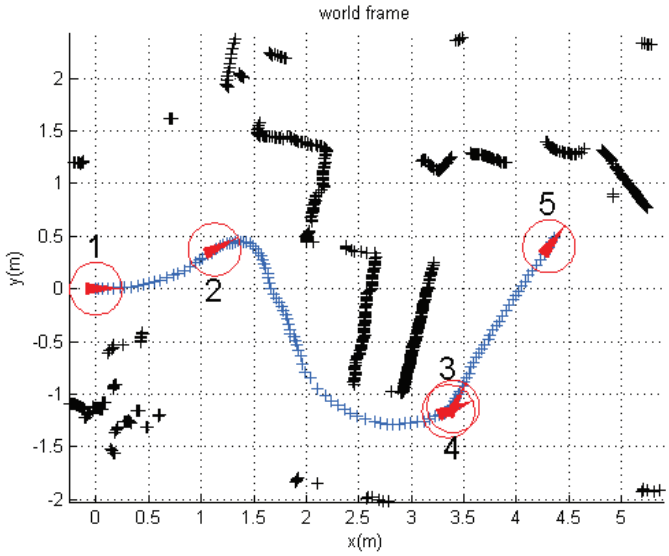


Figure 13: Navigation basée sur la vision

non-holonomie cinématique du robot.

Dans le contexte de la détection d'objet, nous avons présenté une approche homographique RANSAC deux-points pour la détection d'objets plan posés verticalement. Cet algorithme qui prend en compte les contraintes de notre application présente de très bonnes performances. Pour éviter les collisions, nous avons présenté des méthodes efficaces de détection, de représentation et d'évitement d'obstacles utilisant des données d'un télémètre laser 2D. nous avons modélisé les frontières des obstacles modélisés par des chaînes polygonales avec de B-splines de lissage. Nous avons ensuite un évitement d'obstacles réactif comme un suivi de chemin matérialisé par les B-splines. Le résultat expérimental montre que le robot peut se déplacer en douceur sans être piégé dans la partie concave d'obstacles. Enfin, nous avons discuté de la navigation basée sur la vision considérant trois contraintes, la non-holonomie, l'évitement des collisions et le champ de vision restreint de la caméra. Nous avons proposé une stratégie, qui peut traiter la perte complète de cible pendant le contrôle par vision. Nous avons prolongé la stratégie proposée à l'image sphérique pour résoudre l'inconvénient du modèle de projection perspective.

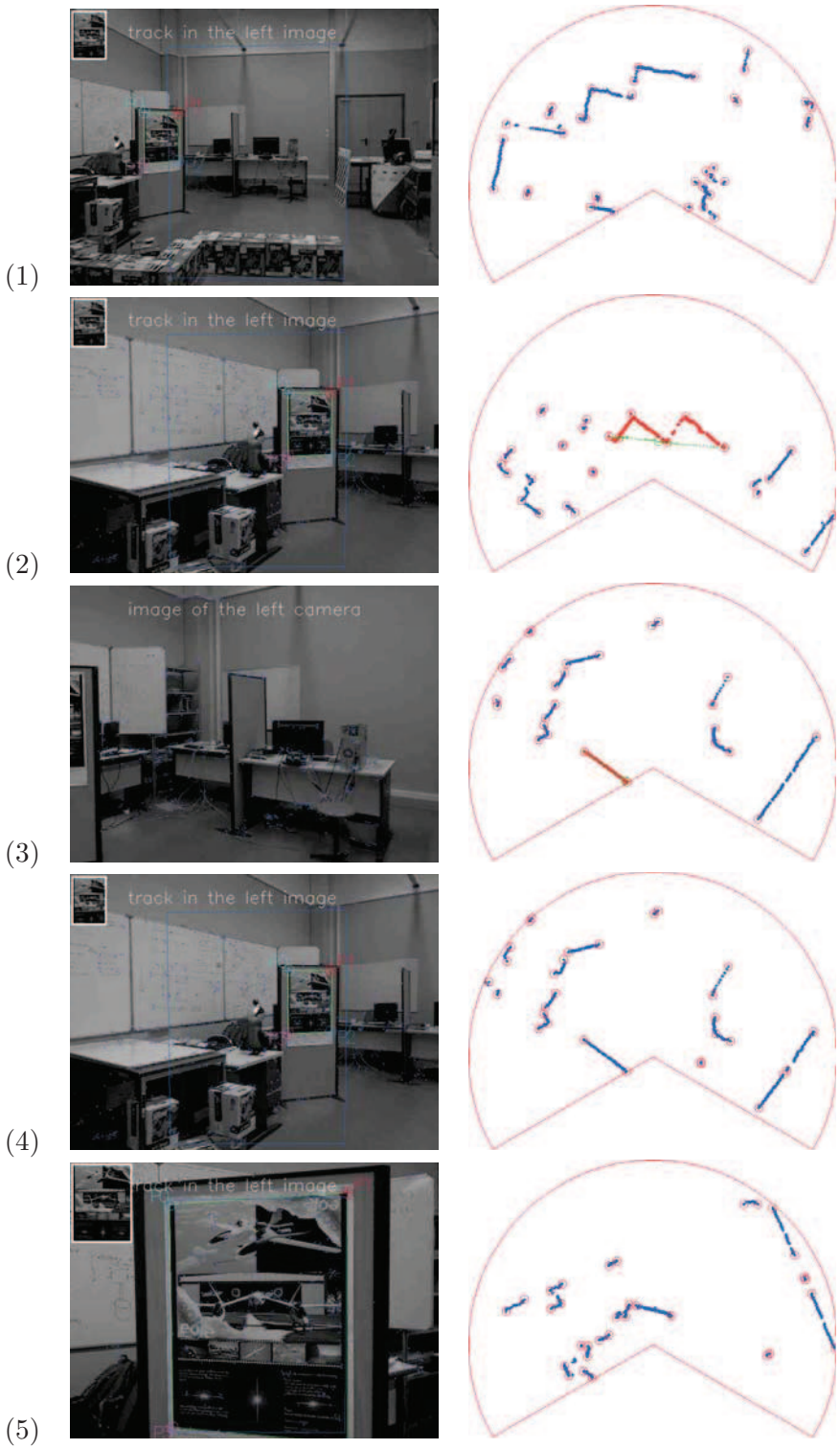
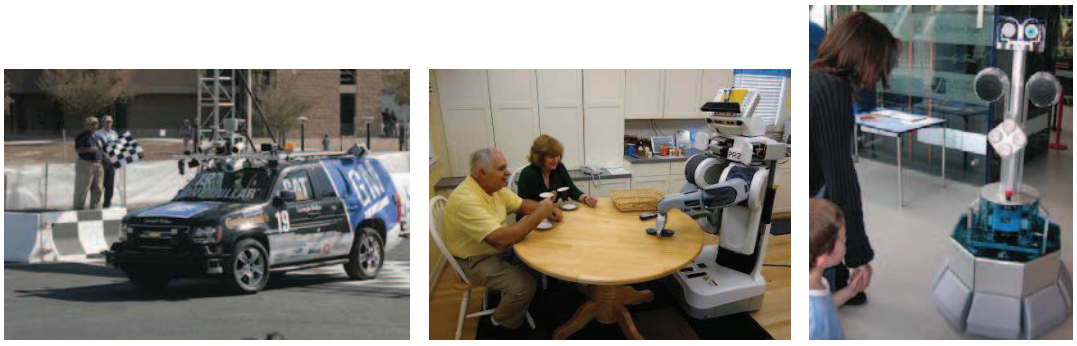


Figure 14: Données de capteur

Introduction

Autonomous Mobile Robot Navigation

Autonomous mobile robots answer to a wide range of applications. In the military field, they can go to the battlefield instead of soldier, significantly reduce the casualties of soldiers. DARPA Grand Challenge (see Figure 15(a)) is a prize competition for American autonomous vehicles, aiming to bridge the gap between fundamental discoveries and military use. In the person service, a particular application domain is assistance to elderly and disabled people, and the systems are typically named assistive robotics (see Figure 15(b)). Some kinds of mobile robots serve in the public place such as tour guides in museums or exhibitions. They can move autonomously acquiring the attention of the visitors and interact with them efficiently. For example, RoboX is a tour-guide robot at the Swiss National Exhibition Expo.02 as shown in Figure 15(c).



(a) Tartan Racing in the DARPA Urban Challenge 2007.

(b) Assistive robotics research of The Georgia Tech.

(c) RobotX a tour guide robot.

Fig. 15: Examples of autonomous mobile robots.

In order to move autonomously and safely, the mobile robot systems should have the capability of navigation. Autonomous mobile robot navigation can be roughly described as the process of perceiving environments and determining a suitable and safe path

between a starting and a goal position for a robot traveling between them. Different sensors have been used to this purpose. Vision not only provides rich information, but also has the advantages of low cost, small size, and high resolution. It is very useful for autonomous navigation capability of mobile robots [37] [16]. Even through vision encounters some challenging problems, it would seem to be the most promising one for the intelligent systems in the long term. In the context of vision based mobile robot navigation, the environment can be generally represented by geometric model, graphic, a dataset of images, or some semantic objects. According to the configuration of the camera, there are monocular systems, binocular (stereo) systems, and omnidirectional cameras.

Much work deals with navigation by using or building the map (such as SLAM) of the environment and then planning the trajectory in the map. Many SLAM and path planning methods have been proposed in the literature. However, a human would need neither the exactly whole metric model of the environment nor the exact location, but would still know how to go to the destination with visual landmarks. In the other hand, in order to accomplish a navigation task, it is not necessary to construct the whole model with precise geometrical details of the environment. Inspired by this, topological representation are proposed and widely used. And mapless navigation systems are proposed. These systems mostly include reactive techniques based on visual clues derived from the segmentation of an image, appearance-based localization, optical flow, features tracking, plane ground detection/tracking, etc. More recently, many works have been carried out to solve the navigation by tracking a visual path. These works refer to appearance-based navigation dealing with the image retrieval and the design of visual controllers. Different with map-based navigation, appearance-based navigation can control the mobile robot by matching the current image with the datasets and computing the control parameters directly from the matched images without knowning any information of the environment model. Hence, appearance-based mobile robot navigation approaches avoid environment modeling, loop closing problem and the time consuming planning algorithms, which are required in the metric map-based navigation approaches. However, a global map of the environment is necessary for a complete autonomy. Appearance-based navigation approaches are sufficient for some defined navigation applications, since this kind of navigation approach is limited to the given visual path. And currently, in the appearance-based navigation, seldom works consider collision avoidance, which can make the robot deviate from the given visual path. We aim to extend this method.

Objective of the Thesis

Our research work aims to develop autonomous mobile robot systems to assist elder or handicapped people in everyday situations. Our area of research focuses on the scenarios for domestic use. We want to design a mobile robot which can autonomously and safely move from one room to the other room with visual landmarks in the environment. Figure 16 illustrates our environment with the visual landmarks. The visual landmarks are posed on the walls near the doors. The environment is presented by the images of those visual landmarks. The images are stored in the memory of the robot as the database. The mobile robot can navigate between visual landmarks in presence of obstacles. We suppose that the robot has a fixed pinhole camera, a 2-D

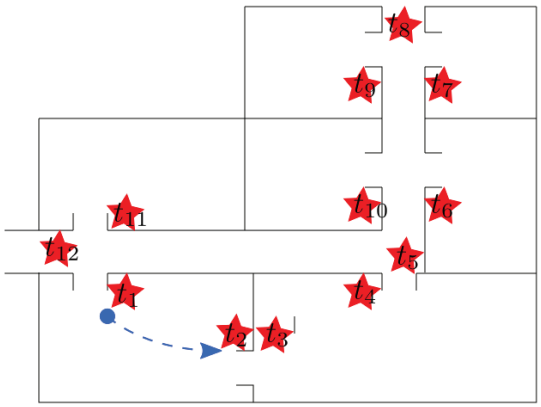


Figure 16: Illustration of the environment representation using visual landmarks (Some room of the IBISC LAB).

laser range finder and an odometry. However, in our platform the odometry does not provide sufficiently accurate motion measurements. Because our mobile robot platform is equipped with a stereo vision system, we use stereo visual odometry to circumvent this problem. With the pinhole camera, the robot can move autonomously by detecting and tracking the visual landmarks. During the motion the safety is ensured using the 2-D laser range finder. This somewhat alleviates the responsibility of the vision perception software, but also encounter the problem of 3-D obstacle detection. Smooth motion is important, as sharp movement produces motion blur in the image and the human anticipate the movement when interactive with the robot. The obstacle avoidance should run in real-time in order to leave enough processing resource for the vision system. Because of field-of-view (FOV) constraints imposed by the camera system, the visual target can be lost during the vision-based navigation, especially when the robot

encounters an obstacle and moving around it. In order to handle the target loss, the robot estimates the lost target using the visual odometry.

We focus on the challenge of each step in appearance-based navigation, which is navigating to a visual landmark while considering the collision free constraints and field of view constraints as well as the nonholonomic constraints. One step of navigation is described in Figure 17. In each step, three principally tasks are considered: (1)detecting and tracking visual landmarks in order to navigate between them. The motion of the robot is controlled by vision-based approaches. This task drives the robot toward a desired goal. Thus it is called goal-driven task. (2) collision detection and avoidance, in which the safety of robot motion is considered. This task is called collision avoidance task. (3) the target estimation when it is lost due to collision avoidance, in order to reacquire it after the obstacle is avoided.

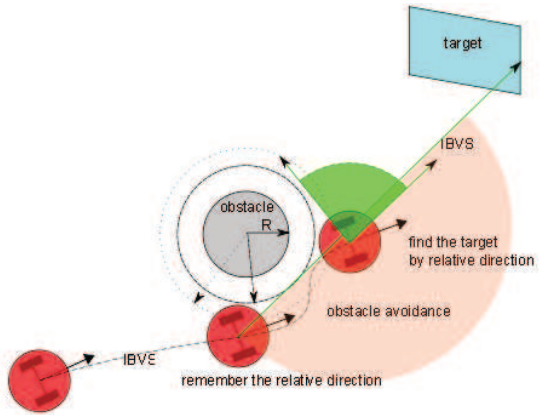


Figure 17: Illustration of appearance-based navigation challenges.

Contributions

Appearance-based approaches are limited to the given visual path. However, the unknown obstacles may make the robot deviate from the trained visual path so that the visual path can be lost in the camera field of view. As far as we known, there is not so many works to handle the visual path lost during vision-based navigation. The principal contribution of this thesis is the solution to handle the visual path lost caused by collision avoidance during the vision-based navigation. In details, the following areas are original works:

– Development of an efficient obstacle representation and avoidance approach to

enable smooth motion during collision avoidance.

- Development of a homography-based object detection and tracking system able to track a natural planar object from a large distance.
- Designing a vision-based navigation system and strategies in present of obstacles.
- Development of a homography-based visual servoing which can handle the target loss due to field of view constraints and collision safety constraints.
- Extension to the use of spherical image based visual servoing for the target lost problem.

Overview

The rest of the thesis are organized as follows:

Chapter 1 studies the method of planar object recognition and tracking. It is crucial to detect and track the landmarks during mobile robot navigation. It can be particularly challenging in the face of changes in perspective, size, or scale, and when the object is partially obstructed from view. There is an extensive body of work in computer vision about object recognition. We choose local feature-based object recognition methods after overview the object recognition approaches. Then the object recognition is changed into image matching problems. It is known that the performance of feature-based methods related to application. Therefore, we introduce the criteria of local feature selection for our navigation tasks. Local feature-based methods consist of three steps: feature detection, feature description, and feature matching. We introduce the theoretical bases in Section 1.2. The state of the art and the representative methods about local feature detection, description and matching are overviewed in Section 1.3 and Section 1.4. We select the appropriated methods with the defined criteria. In Section 1.5, we show experiments of local feature-based method used in our navigation scene. Section 1.6 present the visual target tracking method in our vision system. Section 1.7 concludes the chapter.

Chapter 2 addresses the problem of motion safety for mobile robots in unknown dynamic environments. This refers to collision avoidance. We study two methods for reactive collision avoidance. For the first one, we formulate collision avoidance by path following with respect to the obstacle boundary. For the second, we formulate it as potential field method. In order to ensure a good performance of robot motion, we aim to find a smooth representation for arbitrary obstacle boundaries which can adaptive

to the measurements when the robot reacts to unknown dynamic environments. We respectively introduce the problem and overview the state of the art in Section 2.1 and Section 2.2. We propose our obstacle detection and representation methods in Section 2.3. We formulate the obstacle avoidance approach and simulate it in Section 2.4. We present the experimental result in Section 2.5. Section 2.6 concludes the chapter.

Chapter 3 dedicates the challenging problems of visual navigation. The motion of robot during navigation is controlled by visual servoing approach, which is set up by the output of visual target tracking. The challenge is that visual target may be lost when the robot avoids obstacles. The loss of target can be due to the limited camera field for view or front objects. We design a strategy using path integration to estimate the lost target. The estimated virtual target can be used to continue the visual servoing. However, the estimation effect from the errors. The error can be minimized by reset the object recognition. The strategy has the ability of reacquiring the lost target after the collision is free. On the observation that the visual target may come to the backward of the image plane using perspective projection model. We find solution using spherical projection model, and spherical image-based visual servoing methods are discussed.

The conclusion concludes the thesis with recalling the problems and main results, general discussion, and perspectives for future work both in short-term and long-term.

Chapter 1

Feature Selection for Object Recognition and Visual tracking

1.1 Introduction

In the topological map, the nodes are defined in the image space, which are a set of images. The mobile robot navigates using these images as visual landmarks. By recognizing and tracking these landmarks, the robot can locate itself in the map and move toward them. This chapter discusses these vision abilities of our navigation system.

For increasing robustness and efficiency, artificial markers like Data Matrix can be used as landmarks. However, it is generally undesirable to place this kind of markers in the environment. Thanks to the modern computer vision algorithms, the working environments of a mobile robot are not limited to artificial markers. It is possible to use more normal objects as landmarks. We consider more complex visual targets such as posters or paintings, which are common in indoor environments, for instance, office and home. In order to use the visual landmarks for location and motion control, the robot needs to recognize and track the landmarks efficiently. Our vision system recognizes the landmarks by extracting and matching local features. Then a template-based visual tracker is initialized by the recognition result. The result of the tracker encodes the relative pose of the robot, for example, a homography matrix. It can be used for motion control over the navigation. So the vision system performs as the front-end of a vision-based controller. In this chapter, we introduce the details of our vision system for landmark recognition and tracking.

We organize the rest of this chapter as follows. Section 1.2 introduces some con-

ceptions about homogeneous transformation, image formation, and homography. The conceptions are the basics of geometric computer vision and vision-based control for the following sections and chapters. Local feature extraction and matching are described in Section 1.3 and 1.4 before landmark recognition in Section 1.5. In Section 1.6, a visual tracking framework is presented. The tracking is initialized by the recognition results.

1.2 Theoretical Bases

In this section, we introduce some basic conceptions, including homogeneous transformation, image formation, and homography. Homogeneous transformation is important to describe the relationship between different frames, for example, the sensor frames and the robot frame, or the sensor frames and object frames. Image formation gives the relationship between the object and the image with camera parameters. Homography illustrates multiple view geometry for planar objects. Image formation and homography will be used for vision tasks in the thesis such as landmark recognition and vision-based control.

1.2.1 Homogeneous Transformation

The relationship between two coordinate frames can be expressed by

$$\mathbf{T} = (\mathbf{R}, \mathbf{t}) \in SE(3) \tag{1.1}$$

or by a homogenous transformation matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{1.2}$$

It is noted that the homogenous transformation consists of a pure rotation $\mathbf{R}$ followed by a pure translation $\mathbf{t}$. The inverse homogeneous transformation is

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{1.3}$$

We use the transformation matrix with subscripts ${}^2\mathbf{T}_1$ to note the transformation from the frame $\mathcal{F}_1$ to the frame $\mathcal{F}_2$ or the transformation of the frame $\mathcal{F}_1$ with respect to the frame $\mathcal{F}_2$. Suppose a point $\mathbf{P}$ in the world frame. If we note $\mathbf{X}_1$ and $\mathbf{X}_2$ as the

normalized homogenous coordinates of the points $\mathbf{P}$ in the frame $\mathcal{F}_1$ and the frame $\mathcal{F}_2$ respectively, we have the transformation between the two coordinates

$$\mathbf{X}_2 = {}^2\mathbf{T}_1 \, \mathbf{X}_1. \tag{1.4}$$

A homogeneous transformation can be divided into a sequence of transformations

$${}^2\mathbf{T}_0 = {}^2\mathbf{T}_1 \, {}^1\mathbf{T}_0. \tag{1.5}$$

We will remove the upper and lower standards of $\mathbf{T}$ in case there is no ambiguity of frame transformations.

1.2.2 Image Formation

Image formation describes the mapping from world points to the corresponding image points. Here we describe the image formation of planar perspective projection. In the planar perspective projection, a world point $\mathbf{P}$ with coordinates $\mathbf{X}_0$ is projected into an image plane $\Pi_{\mathbf{m}}$ with coordinates $\mathbf{m}$ in pixels as illustrated in Figure 1.1. The projection can be divided into three steps:

1. Homogeneous transformation of the point from the world frame to the camera frame using equation 1.4 and equation 1.3

   $$\mathbf{X} = {}^C\mathbf{T}_W \, \mathbf{X}_0 = ({}^W\mathbf{T}_C)^{-1} \, \mathbf{X}_0, \tag{1.6}$$

   where $\mathbf{X}$ is the coordinates of the point $\mathbf{P}$ in the camera frame. We note the transformation as $\mathbf{g}$, where we have

   $$\mathbf{X} = \mathbf{g}\mathbf{X}_0. \tag{1.7}$$

2. Perspective projection of the point $\mathbf{P}$ with the coordinates $\mathbf{X} = [X \; Y \; Z]^\top$ in the camera frame to a point $\mathbf{p}$ on the normalized image plane $\Pi_{\mathbf{x}}$ with the image coordinates $\mathbf{x} = [x \; y]^\top$ in meters,

   $$\begin{cases} x &= \frac{X}{Z} \\ y &= \frac{Y}{Z} \end{cases}. \tag{1.8}$$

   This projection uses an ideal pin-hole camera model. It supposes that the image plane $\Pi_{\mathbf{x}}$ is parallel to the xy plane of the camera frame $\mathcal{F}_C$, and the projection

transforms the coordinates linearly without any distortion. The projected image point $\mathbf{p}$ with the coordinates $\mathbf{x}$ is the intersection of the line, passing through $\mathbf{C}$ and $\mathbf{P}$, with the image plane $\Pi_{\mathbf{x}}$. The distance of the center projection $\mathbf{C}$ to the image plane $\Pi_{\mathbf{x}}$ is f, referred to the focal length. Since the normalized or canonical image plane is considered, here we have $f = 1$. Note that the dimension related to the depth is lost through this process. This projection can also be written in the matrix form

$$\lambda \mathbf{x} = \Pi_0 \mathbf{X} \tag{1.9}$$

where $\mathbf{X}$ and $\mathbf{x}$ are the normalized homogeneous coordinates, $\lambda = Z$ is the depth of $\mathbf{X}$, and Π_0 is the standard projection matrix

$$\Pi_0 = [\, \mathbf{I} \mid \mathbf{0} \,] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.10}$$

An alternative expression of equation 1.9 is

$$\mathbf{x} = \lambda' \mathbf{X}, \tag{1.11}$$

where only $\mathbf{x}$ is the normalized homogeneous coordinates, and $\lambda' = 1/Z$.

3. Conversion from the normalized image coordinates $\mathbf{x} = [x \; y]^\top$ in meters to the coordinates $\mathbf{m} = [u \; v]^\top$ in pixels considering the effect of an actual camera. The effect of the camera is expressed by its intrinsic parameters. First, the focal length f is considered to move the projected point $\mathbf{p}$ from the normalized image plane to the factual image plane. Then the scaling factors or even distortions are considered to obtain the coordinates in pixels. The two processes without

distortions can be described by

$$\begin{aligned}\mathbf{m} &= \begin{bmatrix} s_u & s_\theta & u_c \\ 0 & s_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x} \\ &= \begin{bmatrix} fs_u & fs_\theta & u_c \\ 0 & fs_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x} \\ &= \begin{bmatrix} f_u & \alpha & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}, \end{aligned} \tag{1.12}$$

where $[f_u \ f_v]^\top$ is the vector of focal length in pixels. $[u_c \ v_c]$ is the vector of the principal point coordinates in pixels. α is the skew coefficient equaling the angle between the x and y pixel axes with a scale f_u. The matrix of parameters describes the camera model. That is the intrinsic parameter matrix of the camera. If this matrix is noted as $\mathbf{K}$, we have

$$\mathbf{m} = \mathbf{K}\mathbf{x}, \tag{1.13}$$

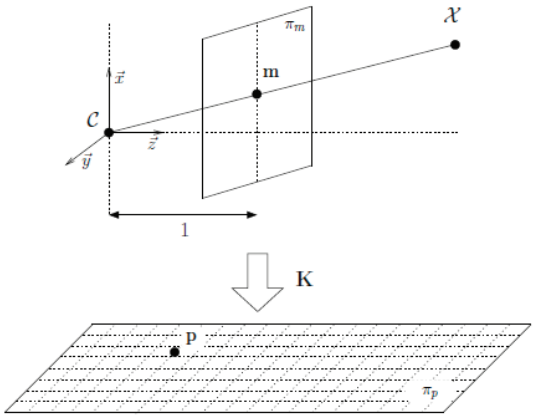


Figure 1.1: Planar perspective projection.

The first two steps are related to extrinsic parameters, which is a set of geometric parameters. While the third step is related to intrinsic parameters, which characterize the optical, geometrical distortion and digital characteristics of the camera. They

are necessary to link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame. The main goal of camera calibration is to identify these parameters.

The whole planar perspective projection can be described by the following equation:

$$\lambda \mathbf{m} = \mathbf{K}\Pi_0 \mathbf{g}\mathbf{X}_0 = \mathbb{P}\mathbf{X}_0 \tag{1.14}$$

where $\mathbb{P}$ is a 3×4 matrix. The matrix $\mathbb{P}$ is the projection matrix, which transforms a 3-D point on an image plane with coordinates in pixels. Camera calibration, both intrinsic and extrinsic, aims to calculate the matrix $\mathbb{P}$.

Although the direct measurement of a point is the image coordinates $\mathbf{m}$ in pixels, from equation 1.13 we can always know the relative coordinates $\mathbf{x} = \mathbf{K}^{-1}\mathbf{m}$. Because sensors are generally calibrated before used in robotics. Therefore, if there are no special instructions, we use the coordinates $\mathbf{x}$ as the image measurements in the thesis.

1.2.3 Homography Transformation

The relationship between two camera frames with respect to a planar object as shown in Figure 1.2 can be described by homography transformation. In Euclidean space, let a 3-D point $\mathbf{P}$ in the world frame with coordinates $\mathbf{X} = (X, Y, Z)$. The point $\mathbf{P}$ is observed from two different camera frames $\mathcal{F}_1$ and $\mathcal{F}_2$. The corresponding camera coordinates are $\mathbf{X}_1 = (X_1, Y_1, Z_1)$ and $\mathbf{X}_2 = (X_2, Y_2, Z_2)$ individually. The projections of $\mathbf{P}$ onto the two image planes are $\mathbf{p}_1$ and $\mathbf{p}_2$ with coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively. The projection coordinates in pixels are $\mathbf{m}_1$ and $\mathbf{m}_2$.

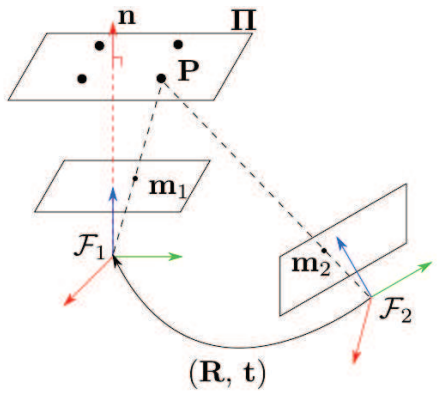


Figure 1.2: A special case of the two-view configuration: points on a plane

Suppose the homogeneous transformation between $\mathcal{F}_1$ and $\mathcal{F}_2$ is ${}^2\mathbf{T}_1$. We have the

relation in the homogeneous form

$$\mathbf{X}_2 = {}^2\mathbf{T}_1\mathbf{X}_1. \tag{1.15}$$

If we note the rotation matrix and translation vector as ${}^2\mathbf{R}_1$ and ${}^2\mathbf{t}_1$ respectively, we also have

$$\mathbf{X}_2 = {}^2\mathbf{R}_1\mathbf{X}_1 + {}^2\mathbf{t}_1. \tag{1.16}$$

In the frame $\mathcal{F}_1$, if we know the structure of the plane target, namely the normal vector $\mathbf{n}_1$ and the distance d_1 from the origin of $\mathcal{F}_1$ to the plane Π, we have the plane function at point $\mathbf{P}_1$

$$\mathbf{n}_1{}^\top\mathbf{X}_1 = d_1. \tag{1.17}$$

Then we have

$$\frac{\mathbf{n}_1{}^\top\mathbf{X}_1}{d_1} = 1. \tag{1.18}$$

Inserting the above equation in the transformation equation 1.16, we have

$$\mathbf{X}_2 = {}^2\mathbf{R}_1\mathbf{X}_1 + \frac{{}^2\mathbf{t}_1}{d_1}\mathbf{n}_1{}^\top\mathbf{X}_1 = ({}^2\mathbf{R}_1 + \frac{{}^2\mathbf{t}_1}{d_1}\mathbf{n}_1{}^\top)\mathbf{X}_1, \tag{1.19}$$

which represents a transformation of a point in two different frames. It can be noted as

$$\mathbf{X}_2 = {}^2\mathbf{H}_1\mathbf{X}_1 \tag{1.20}$$

where ${}^2\mathbf{H}_1$ is the homography matrix in Euclidean space

$${}^2\mathbf{H}_1 = \mathbf{R} + \mathbf{t}\mathbf{n}^\top. \tag{1.21}$$

where

$$\begin{cases} \mathbf{R} &= {}^2\mathbf{R}_1 \\ \mathbf{t} &= \frac{{}^2\mathbf{t}_1}{d_1} \\ \mathbf{n} &= \mathbf{n}_1 \end{cases}. \tag{1.22}$$

We will remove the upper and lower standards of $\mathbf{H}$ in case there is no ambiguity.

When considering the coordinates on the image plane, we insert equation 1.11 into equation 1.20

$$\mathbf{x}_2 = \frac{\lambda_2}{\lambda_1}\mathbf{H}\mathbf{x}_1, \tag{1.23}$$

which can also be simplified as

$$\mathbf{x}_2 = \mathbf{H}_N\mathbf{x}_1, \tag{1.24}$$

where $\mathbf{H}_N$ is the homography matrix in the normalized coordinates space

$$\mathbf{H}_N = \frac{\lambda_2}{\lambda_1}\mathbf{H} = \lambda\mathbf{H}. \tag{1.25}$$

where λ equals the middle eigenvalue of $\mathbf{H}_N$ [94]

$$\lambda = \sigma_2(\mathbf{H}_N) \tag{1.26}$$

When considering the coordinates on the image plane in pixels, we insert equation 1.13 into equation 1.25

$$\mathbf{m}_2 = \mathbf{K}_2\mathbf{H}_N{\mathbf{K}_1}^{-1}\mathbf{m}_1. \tag{1.27}$$

which can also be written as

$$\mathbf{m}_2 = \mathbf{G}\mathbf{m}_1, \tag{1.28}$$

where $\mathbf{G}$ is the homography matrix in the image space with pixel coordinates

$$\mathbf{G} = \mathbf{K}_2\mathbf{H}_N{\mathbf{K}_1}^{-1}. \tag{1.29}$$

1.3 Local Feature Detection and Description

Local feature has a wide application, including wide baseline matching for stereo pairs [126] [155] [101], image retrieval from large databases [139], object retrieval in video [150], robot localization [140] and servoing [154]. This section reviews the literature of detecting and describing features from an image, and introduces some representative methods. This section aims to provide the background of this area in order to choose a suitable method for landmark recognition, which will be discussed in Section 1.5.

1.3.1 Local Feature Detectors

A feature detector refers to an algorithm that processes an image and finds subsets of the image. The subsets encode representative information and satisfy the given type of feature. The detected features are typically points or regions. In this section, we will give an overview of feature detectors. We then introduce the representative algorithms.

1.3.1.1 Overview of the State of the Art

A wide variety of local feature detectors have been proposed in the literature. A comprehensive survey of this area has been given in the literature [153]. We review

some of the typical detectors, which had a particularly significant impact in this research field. The review focuses on corner feature and blob features detectors, since these kind detectors are distinctive, intuitive, and utilized in a wide range of applications. These features typically present where the image intensity change rapidly. The intensity changes can be detected using differential approaches, commonly first- and second-order gray-value derivatives. That is the underlying idea in most of the local feature detection methods.

A popular corner detector is Harris corner [62]. Harris corners are detected by the eigenvalues of the second-moment matrix, in which the components are the first-order derivatives of Gaussian smoothed image intensity. The accuracy of this detector can be improved to subpixel precision by approximating the cornerness function in the neighborhood of a local maximum through a quadratic function. In the literature [143], Shi and Tomasi proposed the scheme to select the good features strongly based on Harris corner detector. Lindeberg et al. have proposed a scale selection technique in feature detection [81] [83] and affine region estimation [82] [84]. Based on these techniques, Mikolajczyk et al. have extended Harris corner detector to scale and affine invariant, namely Harris-Laplace detector and Harris-affine detector [107].

Blob features are typically detected based on Hessian matrix, in which the components are second-order derivatives of Gaussian smoothed image intensity. A earlist work of Hessian-based detector was proposed by Beaudet [8]. Hessian matrix encodes the structure of image intensity. The image structure can be analyzed by the determinant and the trace of Hessian matrix. Blob features can be detected by finding the local maxima of the determinant or the trace. The trace of Hessian matrix is referred to the Laplace operator. SIFT [88] detector effectively approximates the Laplace operator by a Difference of Gaussians (DoG) filter. SURF [7] detector efficiently approximates the determinant of Hessian matrix with box-type filters and integral images. SURF performs comparably to SIFT at much lower computational cost. Mikolajczyk et al. detected local features using both the trace and the determinant, and have proposed Hessian-Laplace and Hessian-affine detectors in the literature [107].

A very efficient keypoint detector is FAST [131]. A FAST feature is detected by comparing pixels on a ring centered at the feature point. Its variants include ORB [134] detector and BRISK [79] detector. ORB detector extends FAST with an orientation assignment efficiently computed by intensity centroid moment. BRISK detector extends the FAST detection framework to scale-space in order to achieve invariance to scale.

An extensive performance evaluation of affine region detectors has been presented by Mikolajczyk et al. [106]. The evaluation is analyzed using the repeatability cri-

terion under different imaging conditions, including viewpoint change, scale, lighting change, defocus/blur, rotation and image compression. It has reported that MSER is particularly robust to viewpoint and lighting changes. In the literature [108] [64], the evaluation includes recent efficient methods.

1.3.1.2 Representative Feature Detectors

We introduce two popular detectors: Harris corner and Difference of Gaussians (DoG). Harris detector is a corner detector, while DoG is a blob detector.

Harris Corners The Harris corner detector, proposed by Harris and Stephens [62], is based on first-order derivatives to find local extrema. The detector can be expressed by the second-moment matrix, also called the auto-correlation matrix:

$$\mathbf{M} = \sigma_D^2 G(\sigma_I) * \begin{bmatrix} I_u^2(\mathbf{m}, \sigma_D) & I_u(\mathbf{m}, \sigma_D) I_v(\mathbf{m}, \sigma_D) \\ I_u(\mathbf{m}, \sigma_D) I_v(\mathbf{m}, \sigma_D) & I_v^2(\mathbf{m}, \sigma_D) \end{bmatrix}, \tag{1.30}$$

where σ_I and σ_D are the scales of the Gaussian kernels for integration and differentiation, which is an effective edge detector. $I_u(\mathbf{m}, \sigma_D)$, $I_v(\mathbf{m}, \sigma_D)$ are the convolutions of the Gaussian first order derivatives with the image at the coordinates $\mathbf{m}$

$$I_u(\mathbf{m}, \sigma_D) = \frac{\partial}{\partial u} G(\sigma_D) * I(\mathbf{m}), \tag{1.31}$$

$$G(\sigma_D) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}. \tag{1.32}$$

The matrix $\mathbf{M}$ is positive semidefinite, thus it has two nonnegative eigenvalues λ_1 and λ_2. The eigenvectors of $\mathbf{M}$ encode the dominant change directions, which are orthogonal, while the eigenvalues encode the variational strengths. Rather than computing the eigenvalues directly, which is computationally expensive, the Harris cornerness function is given by the determination $\det(\mathbf{M})$ and the trace $tr(\mathbf{M})$:

$$cornerness = \det(\mathbf{M}) - \kappa\, tr^2(\mathbf{M}). \tag{1.33}$$

where κ is a scale gain, typically set as 0.04.

Harris corner is invariant to 2D image rotations, since the filter window is a circular symmetric Gaussian function. It is invariant to affine intensity changes. However, it is not invariant to geometric affine transformations or scale changes.

Difference of Gaussians (DoG) DoG is a scale invariant detector used in SIFT. DoG provides a close and efficient approximation to LoG as studied by Lindeberg in the literature [81]. The detector finds local extrema in Gaussian scale space. That is both in space and scale, referred 3-D extrema. The relationship between DoG and LoG can be deduced from the heat diffusion equation. Replacing the parameter t by σ^2, we have

$$\frac{\partial G}{\partial \sigma} - \sigma \nabla^2 G = 0, \tag{1.34}$$

Approximating the first derivative $\frac{\partial G}{\partial \sigma}$ by difference of nearby scales at $k\sigma$ and σ, we obtain

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}. \tag{1.35}$$

where (x, y) is the variable of space, σ is the variable of scale. Therefore

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma) \approx (kt - t)\sigma^2 \nabla^2 G. \tag{1.36}$$

The Gaussian scale space is generated by smoothing the image several times with a Gaussian convolution kernel as well as sampling to different octaves. The 3-D local extrema are selected both over space and scales with non-maximal suppression. The local extrema are detected in all the octaves. The detailed scheme is illustrated in Figure 1.3. SIFT detector is based on DoG detector. Rather than detecting upright scale and invariant features using DoG detector, SIFT detector uses refinement steps to remain more stable features. The detected features are refined by eliminating low-contrast responses or responses close to edges, which are likely to be unstable. Figure 1.4 shows the results of each refine step to detect local features in DoG scale space.

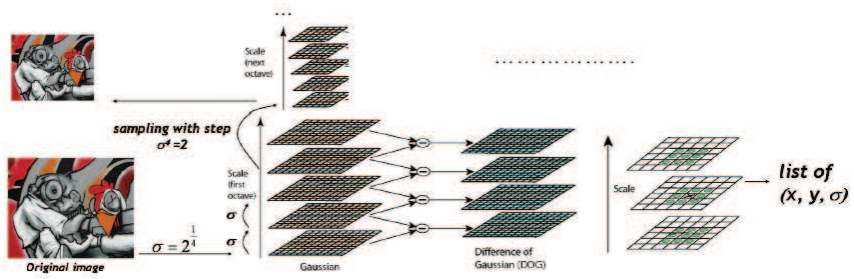
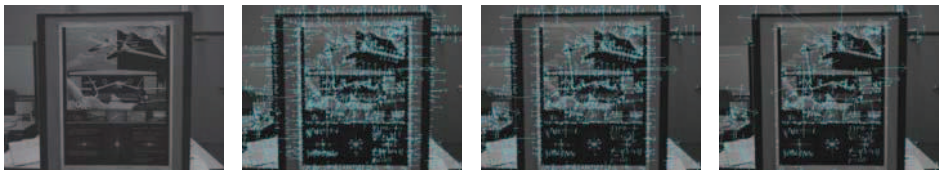


Fig. 1.3: An illustration of DoG scale invariant detector [153].



(a) (b) (c) (d)

Fig. 1.4: This figure shows the stages of SIFT feature selection in DoG scale space.

1.3.2 Local Feature Descriptors

In order to match the detected feature point, we need a descriptor to represent the feature point. The descriptor is a vector extracted from the patch defined by the feature point. The descriptor typically encodes the distinctive information about the patch such as appearance, color, or contour shape. The descriptor should be robust to changes in viewing conditions as well as errors of the detector.

1.3.2.1 Overview of the State of the Art

Many different local feature descriptors have also been proposed in the literature. A comprehensive survey can be found in [105] [80]. The most influencing one is the SIFT descriptor, proposed by Lowe [88]. The SIFT descriptor has been shown to be one of the most effective [105]. It is a histogram-type descriptor, which is based on the spatial distribution of the image gradient within the interest point neighborhood. SIFT descriptor is computational complexity, also time consuming for matching because of its high dimension. Therefore, SIFT is unsuitable for real-time application.

Over the past decade, a variety of methods have been proposed to improve the efficiency and matching quality. The SURF [7] descriptor as its detector approximates the gradient calculation by box-type filters and integral images.

Recently several papers have extended the SIFT descriptor. The SURF descriptor, proposed in [7], is based on histograms of Haar wavelet responses. SURF is as an efficient variant of SIFT. It has been shown in [158] that combining color to the SIFT descriptor can increase illumination invariance and distinctive power. Hence, Color SIFT outperforms intensity-based SIFT. Arandjelovic has presented in the literature [4] that normalizing the SIFT descriptor with square root transformation, referred to RootSIFT, yields superior performance in object retrieval without increasing processing or storage requirements.

SIFT and SURF reveal good performances, but they are usually in high dimen-

sions. For instance, SIFT is 128-dimension. These descriptors have issues of memory and computation time, especially for smart phone applications, which have lower computation power. To solve it, the high dimensional local features can be quantized or clustered into a vocabulary of visual words [150]. An alternative way is to use binary descriptors. The binary descriptors are fast and compact. The most common binary descriptors are LBP [120] [121], BRIEF [22] [21], ORB [134], BRISK [79]. ORB is an extension of BRIEF descriptor by introducing orientation invariance. Other various methods include CARD [3], FREAK [122], and LDB [163]. FREAK means Fast Retina Keypoint. It is a standalone keypoint descriptor. It is biologically inspired by the retinal pattern in the eye. Compared with vector-based descriptors, they are efficient to compute, more compact to store and faster to compare with each other.

In the literature [105], Mikolajczyk and Schmid have evaluated various feature descriptors under different image transformations, such as affine transformation, scale changes, rotation, blur, jpeg compression and illumination changes. The results have shown that SIFT descriptors as well as gradient location and orientation histogram (GLOH) outperform the others such as shape context [11], PCA-SIFT [67], spin images [77], steerable filters [53], differential invariants [71], complex filters [138], moment invariants [159], and cross-correlation of sampled pixel values.

1.3.2.2 Representative Feature Descriptors

Here we introduce the SIFT descriptor which is wildly used in computer vision and robotics. SIFT descriptor, proposed by Lowe [88], is scale invariant. Given a region with respect to the detected interest points and the scales, SIFT descriptor constructs a 3-D spacial histogram of local gradient locations and orientations. The histogram is a 128 dimension vector representing the distribution of spatial gradient orientations in the detected regions around interest point and scales. The local region around a detected interest point is divided into 4×4 spatial bins and 8 orientation bins. The ordination invariant is accomplished by assigning keypoints orientation to the dominant gradient orientation of the histogram. The illuminance invariant is accomplished by normalizing the vector of the histogram to unit length. SIFT descriptors can be used for matching by comparing the distance. The details of SIFT descriptor is illustrated in Figure 1.5 in which 2×2 spatial bins and 8 orientation bins are demonstrated. Feature-point orientation is usually computed as a direction of the dominant image gradient in a particular area.

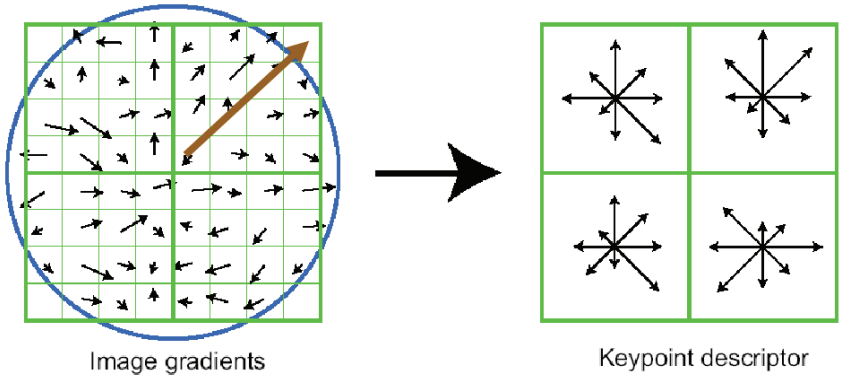


Fig. 1.5: An illustration of SIFT descriptor [88].

1.4 Matching

Matching aims to associate the extracted features between different images or date sets in order to accomplish high level tasks such as image or object recognition. The association measures the similarity between the descriptors. For two feature descriptors, the similarity is computed with a score that defines the closeness between the two vectors. Usually $L1$-norm, $L2$-norm as well as Mahalanobis distances are used, or hamming distance is used for binary feature vectors. For two images or date sets, the matching is to search the nearest neighbor from one set of descriptors for every element of another set. Hence, matching can be formulated by the nearest neighbor searching.

1.4.1 Overview of the State of the Art

Nearest neighbor searching for high-dimensional vectors and large data sets is challenging. In order to searching efficiently, various schemes have been proposed to minimize the searching space or data sets. Those schemes typically perform a hierarchical partition of the searching space or the data points to eliminate the unnecessary calculation, which can support faster processing. The k-d tree [54] is a representative example, and widely used for nearest-neighbor search. K-d tree search approach can find the exact nearest neighbor. Nevertheless, it is not efficient for finding the nearest neighbor in high dimensional spaces. It performs no speedup as dimensionality increases. Fukunaga and Narendra [55] have proposed an algorithm to partition the searching space by clustering the data points with the k-mean algorithm. Then the searching is recursively for each clustered group.

To obtain speedup, numerous approximate nearest neighbor (ANN) search algo-

rithms are proposed in the literature. The underlying ideas are to cluster the data points and/or use tree-search schemes, which are extended from the work of Riedman et al. [54] and Fukunaga et al. [55]. The approximate algorithms obtain large speedups at the cost of the optimum precision. While the approximated neighbors should be close to the exact neighbors to provide near-optimal accuracy. The most promising method of approximate nearest neighbor search is the multiple randomized kd-tree algorithm [147]. Randomized kd-tree splits the searching space randomly. Muja and Lowe [116] have proposed a system to select an approximate algorithm automatically and its parameters with respect to a given dataset and desired degree of precision. An alternative scheme to speedup is to reduce the dimension of searching space by clustering the data points. For example, a local image feature can be clustered into a visual word by k-means or similar algorithms. Then an object or an image is represented by a bag of words, which is a set of feature words. Feature matching or object recognition problems are reduced to word retrieval, which is similar and inspired from the text retrieval system. Bag of word representation is widely used for the image retrieval from large datasets.

In the case of binary-value features, the approximate nearest-neighbor algorithms described above for the vector-based descriptors are unsuitable. Since they assume the features exist within a vector space where each dimension of the features can be continuously averaged. The approximate nearest neighbor search algorithms used in the literature are mostly based on various hashing techniques such as locality sensitive hashing [134], semantic hashing [135] or min-hash [164]. Muja and Lowe have introduced an algorithm in [117]. It is based on priority search of multiple hierarchical clustering trees. It performs well for large datasets, both in terms of speed and memory efficiency.

Moreover, Lepetit and Fua [78] formulate the keypoints matching as a classification problem using Randomized Trees as classifiers.

1.4.2 Outlier Removal using RANSAC

The nearest neighbor searching compares only feature descriptors and gives the putative matches, which includes both inliers and outliers. Inliers are correct matches, which fit well the transformation model between two cameras such as Fundamental matrix, Essential matrix if the camera is calibrated, or even homography transformation if the target is planar. Outliers do not fit the transformation model. Outliers come from true-false matches or noises. In order to select the inliers from the putative

matches, it is needed to estimate the model and verify the matches. Classical estimation methods such as the least square for data fitting use all the putative matches. That will produce a model with a bad fit to the inliers. In order to estimate the transformation robustly, the outliers are needed to be removed from putative matches. Only inliers are remained. RANSAC, abbreviated for "Random Sample Consensus", is for this aim. It finds the inliers using a distance threshold and the transformation constraint between the putative matches to eliminate outliers. The RANSAC algorithm was first published by Fischler and Bolles in 1981 [47]. It is robust and widely used in computer vision.

RANSAC assumes that given a set of data even including inliers, there exist a procedure which can estimate the parameters of a model that optimally fit these inliers. RANSAC finds the inliers in the sense of a hypothesize-and-verify way. That is a minimum independent set of data is randomly selected and supposed as inliers, with which a transformation is fitted. Other data are verified with this fitted model with a distance threshold. If they fit well with this model, they are selected as inliers, otherwise outliers. This process is repeat several times so that the probability of choosing only inliers in the sampled data is sufficiently high. The minimum iteration time is decided by a given confidence.

$$k = \frac{log(1-p)}{1-w^n}, \tag{1.37}$$

where n is the minimum number needed for fitting a model (at least four matches to estimate homography.), p is the probability that the RANSAC algorithm in some iteration selects only inliers from the input data set, and w is the rate of inliers with respect to the data.

Note that RANSAC needs initializing the number of interaction k. However, k depends upon the probability of inliers p. p is unknown before the interaction. Anyway, k can be initialized by any value and updated during the iteration. In addition, RANSAC can only estimate one model for a particular data set. If more than one model are presented, RANSAC may fail. If that is the case, the Hough transform is an alternative robust estimation technique to find the solution [89].

1.5 Feature Selection for Object Recognition

As described in the above sections, many different feature detectors and descriptors have been proposed in the literature. Each of them has its own superiority and weakness. Their performance depends on the application and the type of the image data. In order to decide which feature extraction method is appropriate for our application,

we evaluate the performance of some promising approaches of feature detection and description with our own data set. In the experiment for evaluation, we consider some criteria related with our mobile robot application.

1.5.1 Experimental Framework

From the literature, SIFT is absolutely the most influencing method. SURF, similar to SIFT, results in the efficient computation. MSER is particularly robust to viewpoint and lighting changes reported in [106]. BRIEF, ORB, BRISK and FREAK are the recent wave of feature extractors providing robustness while achieving high computational efficiency. Therefore, we select these methods for comparison.

In indoor mobile robot application, the performances of these local features with changes in scale, viewpoint and motion blur are the main considered factors. Thus, the performance evaluation of these features detectors and descriptors is carried out with three datasets: scale change, viewpoint change, and image blur (see Figure 1.6 to Figure 1.8). In each test, we use the first image of the dataset as the reference image.

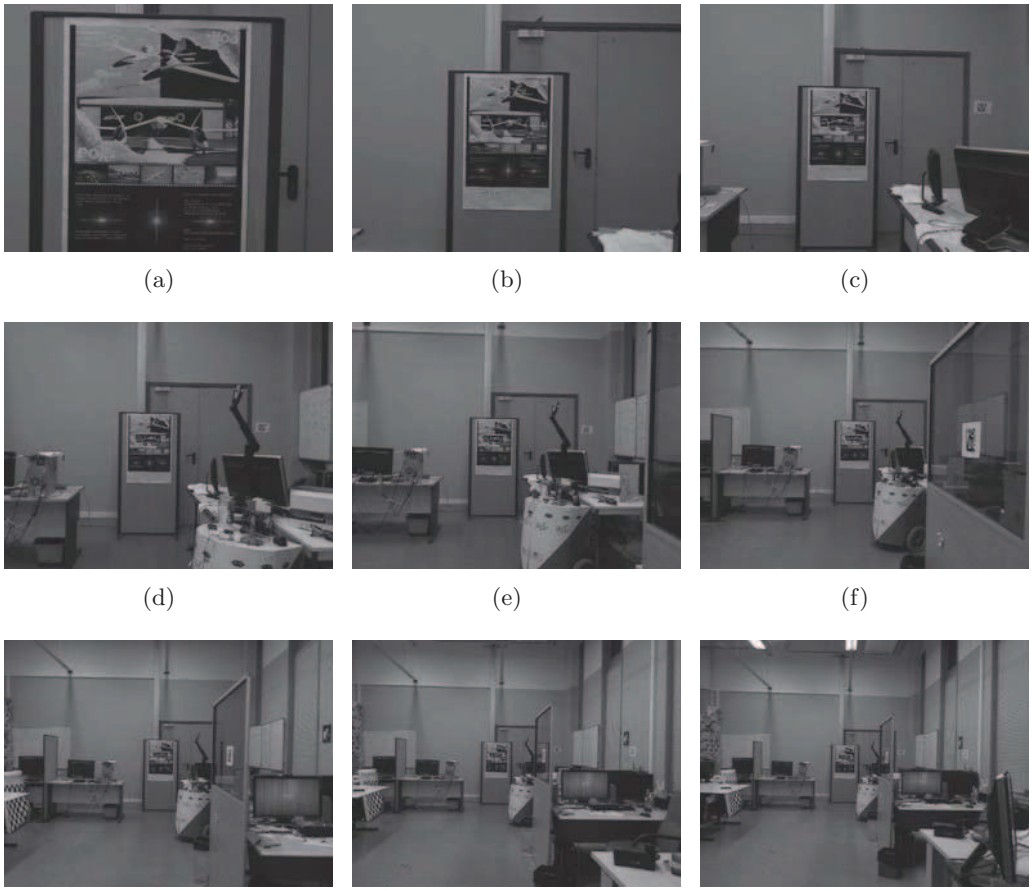
We compare the matching quality and the computation time through the following criteria:

- **time per frame**: absolute total time in milliseconds spent to the feature extraction of a single frame.
- **time per keypoint**: extraction time for a single keypoint. Evaluated as total extraction time divided to number of detected keypoints.
- **number of true matches**: the inliers of the putative matches.
- **percent of true matches**: evaluated as the number of true matches divided to the minimum keypoints of the reference and scene images.

1.5.2 Experimental Results

In the experiments, we test the methods: SIFT, SURF, MSER, BRIEF, BRISK, ORB, FREAK. The tests are on the three image sequences (Figure 1.6 to Figure 1.8). The performances are evaluated with computation times, number of true matches ans percent of true matches.

Figure 1.9 shows the average results of computation times (in millisecond) both for a frame and an individual feature. It notes that SIFT, SURF and MSER are not

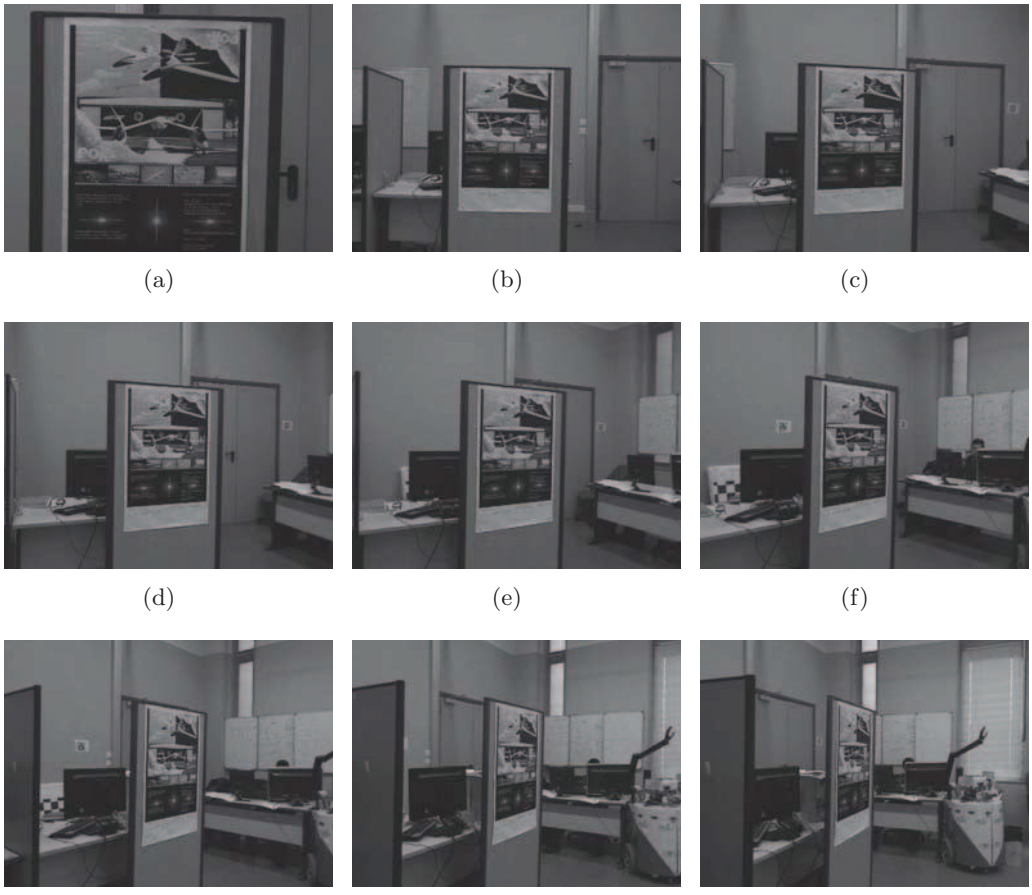


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.6: Test images change in scale.

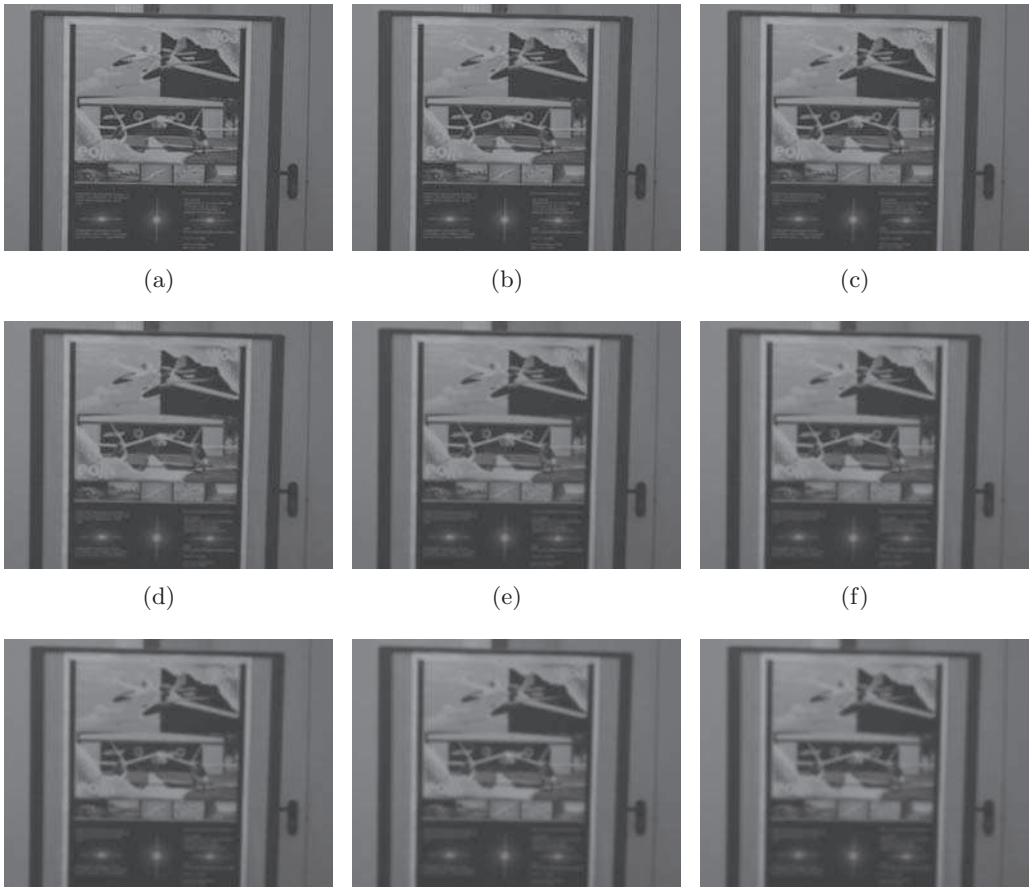


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.7: Test images change in viewpoint.

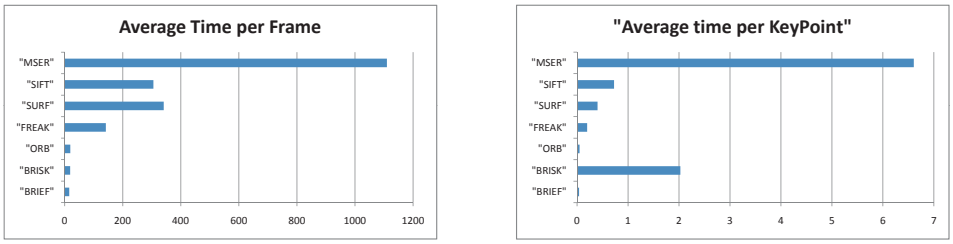


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.8: Test images change in image blur.



(a) Average time per Frame (b) Average time per Keypoint

Figure 1.9: Computation times.

efficient for time-constrained application. BRISK, BRIEF, ORB as well as FREAK satisfy the time-constrained requirement.

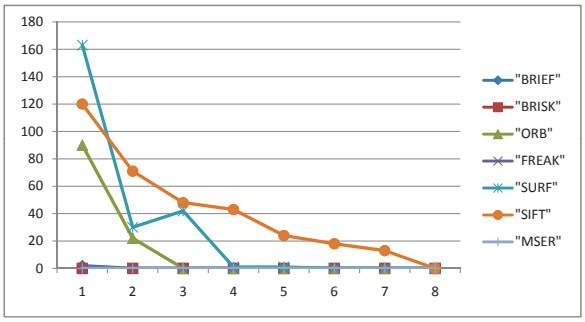
Figure 1.10 and 1.11 respectively illustrate the number of true matches and the percent of true matches. The results show that SIFT provide sufficient true matches especially in case of scaling and viewpoint changes. The percent of true matches are also high for SIFT. It is noted that ORB can compare with SIFT in case of viewpoint changes and image blur while achieving computational efficiency. But it can not provide sufficient good matches for scale changes.

The visual landmark detection method is expected performing both effectively and efficiently at the same time. The method should effectively detect a visual landmark under different viewing conditions as well as efficiently in running time. From the experimental results, no method can perform both effectiveness and efficiency at the same time. Consider that recognizing a visual landmark effectively is the first important. So we choose SIFT. SIFT is hard for real-time application. We use it to initialize a template-based visual tracking. The details of visual tracking will be introduced in section 1.6.

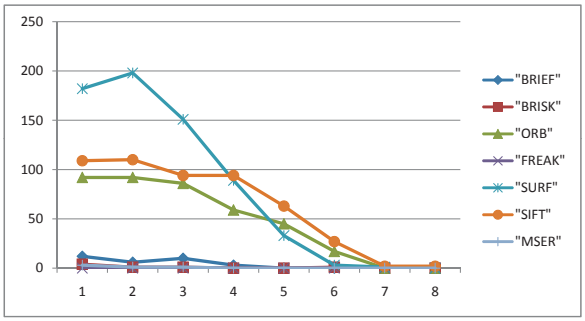
1.5.3 Constraint RANSAC for Homography Estimation

In our experiments of planar visual landmark detection, unexpected results can happen as illustrated in Figure 2. On the left of the figure is the reference image, in which a poster is the object. On the right is the scene image. The matches are represented by straight lines. The green lines represent the inliers satisfying a homography transformation, which is selected by the 4-points RANSAC algorithm. The pink lines represent the outliers, which are not satisfied the estimated homography. The matching result obviously shows that the standard 4-points RANSAC algorithm for homography estimation can select the false model. Hence, the real inliears are not selected. This is because the RANSAC algorithm works randomly and selects a model with minimum error. And the homography encode the camera motion between two viewpoints and the structure of the planar object. The 4-points RANSAC algorithm considers 6 DoF camera motion with respect to an arbitrary oriented planar object.

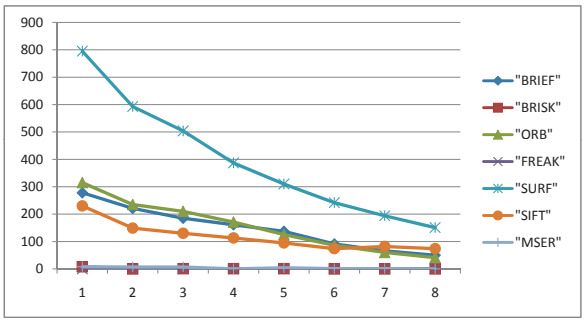
However, in our indoor environment application, the robot moves in a plane. For our camera-robot configuration, the camera is fixed on the mobile robot and its motion is parallel with the ground. Hence, the camera motion has 3 DoF, two translation and one rotation. The planar object is placed in parallel with the wall and vertically to the ground. Hence, the normal vector of the planar object is also parallel to the ground.



(a) Scaling

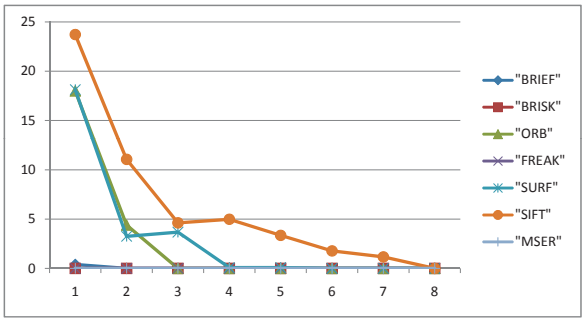


(b) Viewpoint

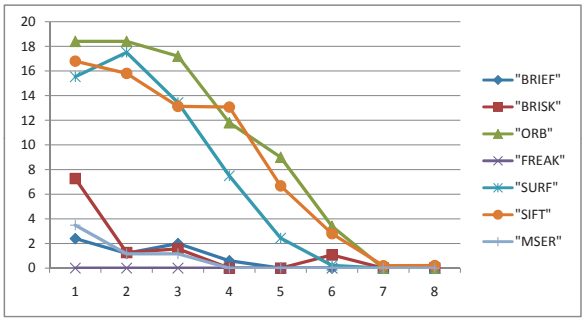


(c) Image blur

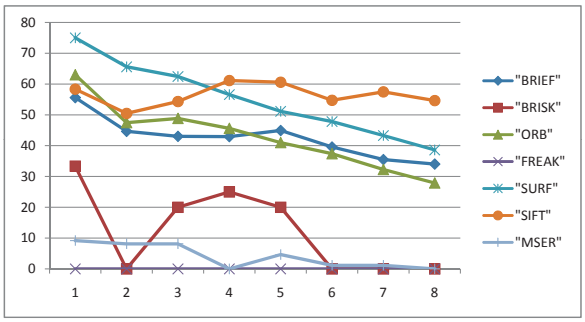
Figure 1.10: Number of correct matches.



(a) Scaling



(b) Viewpoint



(c) Image blur

Figure 1.11: Percent of correct matches.

Figure 1.12: The problem of 4-Points RANSAC for object recognition.

In order to improve the detection result, we consider the constrained motion of the camera and the constrained orientation of the planar object to formulate a constrained homography. In order to improve the detection result, we use the RANSAC algorithm to estimate the constrained homography in the matching step.

Consider the motion constraint configuration, in which the camera is fixed on a mobile robot. The motion of the camera and its xz-plane are both parallel with the ground, the transformation matrix of the camera is decreased by:

$$\begin{aligned}\mathbf{T} &= \begin{bmatrix} \mathbf{R}(\theta_y) & \mathbf{t}_{xz} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & 0 & \sin\theta & t_x \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{1.38}$$

The observed vertical planar target gives the special structure norm $\mathbf{n}_{xz} = [n_x\ 0\ n_z]^\top$ in the camera frame. Hence, the homography is:

$$\begin{aligned}\mathbf{H} &= \mathbf{R} + \frac{\mathbf{t}}{d}\mathbf{n}^\top \\ &= \begin{bmatrix} \cos\theta + \frac{t_x n_x}{d} & 0 & \sin\theta + \frac{t_x n_z}{d} \\ 0 & 1 & 0 \\ -\sin\theta + \frac{t_z n_x}{d} & 0 & \cos\theta + \frac{t_z n_z}{d} \end{bmatrix}. \end{aligned} \tag{1.39}$$

From equation (1.25), we have

$$\begin{aligned}\mathbf{H}_N &= \lambda\mathbf{H} \\ &= \begin{bmatrix} \lambda(\cos\theta + \frac{t_x n_x}{d}) & 0 & \lambda(\sin\theta + \frac{t_x n_z}{d}) \\ 0 & \lambda & 0 \\ \lambda(-\sin\theta + \frac{t_z n_x}{d}) & 0 & \lambda(\cos\theta + \frac{t_z n_z}{d}) \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & 0 & h_{13} \\ 0 & h_{22} & 0 \\ h_{31} & 0 & h_{33} \end{bmatrix}. \end{aligned} \tag{1.40}$$

For each correspondence $(\mathbf{x}_1, \mathbf{x}_2)$ with homogeneous coordinates, considering equation (1.24), we have $\mathbf{x}_2 = \mathbf{H}_N\mathbf{x}_1$, which is equivalent to

$$\mathbf{x}_2 \times \mathbf{H}_N\mathbf{x}_1 = \mathbf{0}. \tag{1.41}$$

If we note the j^{th} row of $\mathbf{H}_N$ using a row vector $\mathbf{h}_j^T$, all the items of $\mathbf{H}_N$ presented by a vector $\mathbf{h} = [\mathbf{h}_1^\top\ \mathbf{h}_2^\top\ \mathbf{h}_3^\top]^\top$. The cross product can be written as

$$\begin{aligned}\mathbf{x}_2 \times \mathbf{H}_N\mathbf{x}_1 &= \begin{bmatrix} \mathbf{0}^T & -z_2\mathbf{x}_1^T & y_2\mathbf{x}_1^T \\ z_2\mathbf{x}_1^T & \mathbf{0}^T & -x_2\mathbf{x}_1^T \\ -y_2\mathbf{x}_1^T & x_2\mathbf{x}_1^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix} \\ &= 0, \end{aligned} \tag{1.42}$$

Note that in equation (1.42), the third row can be obtained by the first and the second row linearly. Only two rows are linearly independent. Hence, each match gives two equations in the elements of $\mathbf{H}_N$. If we omit the third row of the equation (1.42), it can be written as

$$\begin{bmatrix} \mathbf{0}^T & -z_2\mathbf{x}_1^T & y_2\mathbf{x}_1^T \\ z_2\mathbf{x}_1^T & \mathbf{0}^T & -x_2\mathbf{x}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix} = 0. \tag{1.43}$$

In matrix form, equation (1.43) is

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \tag{1.44}$$

where $\mathbf{A}$ is the matrix of the parameters in equation (1.43). Considering equation (1.40), the vector $\mathbf{h}$ is reduced to $\mathbf{h}_r = [h_{11}\ h_{13}\ h_{22}\ h_{31}\ h_{33}]^\top$. So the matrix $\mathbf{A}$ will be

reduced to

$$\mathbf{A}_r = \begin{bmatrix} 0 & 0 & -y_1 z_2 & x_1 y_2 & z_1 y_2 \\ x_1 z_2 & z_1 z_2 & 0 & -x_1 x_2 & -z_1 x_2 \end{bmatrix} \tag{1.45}$$

For N matches, just stack each matrix $\mathbf{A}_i$ into a matrix $\mathbb{A}$:

$$\mathbb{A}\mathbf{h}_r = \begin{bmatrix} \mathbf{A}_0 \\ \vdots \\ \mathbf{A}_{N-1} \end{bmatrix} \mathbf{h} = \mathbf{0}, \tag{1.46}$$

where the $2N \times 5$ matrix $\mathbb{A}$ depends on all the coordinates of the matches. The vector $\mathbf{h}_r$ has 5 entries. Each correspondence gives two equations. Therefore only 2 matches are enough to solve $\mathbf{h}_r$ up to a scale factor. Equation (1.46) is a linear homogeneous system, which can be solved using SVD decomposition $\text{svd}(\mathbb{A}) = \mathbf{USV}^\top$. The least square solution of $\mathbf{h}$ is the eigenvector with respect to the least eigenvalue of $\mathbf{S}$, that is 5^{th} column of the matrix $\mathbf{V}$. More detail about SVD is described in Appendix B.3. Hence, we adapt the homography estimation with only 2 points considering the constraints of the camera motion and the normal vector of the planar object. The experimental result with the 2-Points RANSAC algorithm is illustrated in Figure 1.13. The result shows that considering the constrains can increase the probability of selecting the expected model in the random process.

Figure 1.13: The result of visual landmark detection using 2-points RANSAC.

1.5.4 Discussion

For the experiment results, we get that SIFT almost performs best than others, especially for large scale changes. However, it is time consuming. BRIEF, BRISK and ORB are efficient. But they are at the expense of the qualities of matching. In order to ensure a good detection result we select SIFT.

1.6 Planar Object Tracking

This section illustrates efficient ways of visual tracking. Visual tracking is to estimate the object motion in consecutive frames of a video sequence. Different from object recognition, the properties of tracking can be enhanced using the associations between two consecutive frames. The tracking results can be used for visual servoing and motion estimation. It will be seen in Chapter 3 how the visual tracking is used for vision-based mobile robot navigation. In this section, an overview of visual tracking and the representative tracking approaches are presented followed by the development of our tracking scheme, which can track an unknown planar objects efficiently.

1.6.1 Overview of Tracking Approaches in the Literature

Visual tracking is crucial for many applications such as visual odometry [119], visual Simultaneous Localization and Mapping (SLAM) [33], Augmented Reality (AR) [70], and visual servoing [13], etc. Numerous algorithms have been proposed in this context. The underlying schemes of those proposed methods can be typically classified into feature-based tracking and template-based tracking.

Feature-based tracking approaches use a tracking by detection scheme. They detect the visual object by extracting and matching local features in every frame independently. Many feature extraction methods can be used. Harris points [62] and FAST features [132] provide plentiful local features. But they have difficulties tracking objects that exhibit wide-baseline changes. SIFT [89] and SURF [7] features perform well for wide-baseline matching and in clutter environments. They are time consuming, and not satisfied for high frequent vision-based control tasks. Moreover, there is learning-based technique [78], which formulates the wide-baseline matching problem as a more generic classification problem. This method leads to solutions that are much less computationally demanding. However, it is not as robust as SIFT especially in case of large scale changes.

Template-based tracking approaches describe the object by a target template, for example, an image patch or a color histogram. The object motion is calculated as a transformation that minimizes the mismatch between the target template and the candidate patch. In the minimization step, a well-known method uses sum-of-squared differences (SSD). It can be traced back to the work by Lucas, Kanade [91] and later Shi and Tomasi [143]. The work of Benhimane and Malis [14] presents a homography-based tracking method. The method computes the parameterized homography using a minimization method, which efficiently approximates the second-order of the cost

function without computing Hessian matrices. This method is also extended with central catadioptric cameras [1].

The template-based tracking methods are generally faster and more precise than feature-based tracking methods. They are particularly well adapted to robotic tasks, such as visual servoing and motion estimation. But they are more sensitive to occlusion and less well adapted to large inter-frame displacement.

In addition, there are also some paradigms to strengthen the tracking results, for example, combining the detection and pure tracking. Because detection and tracking are closely related, recent tracking-by-detection paradigm leads to several of the most successful tracking methods. Tracking-by-detection is to use the output of object detectors serves as observation for tracking, to cope with difficult scenarios.

1.6.2 Template-Based Tracking

Template-based tracking matches between image templates $\mathbf{P}_r$ and $\mathbf{P}_c$ in a reference image $\mathbf{I}_r$ and a current image $\mathbf{I}_c$ respectively. Suppose that the relation of the two image template can be presented by a transform function $\mathbf{F}(\mathbf{x})$, where $\mathbf{x}$ is a vector consisting all the parameters of $\mathbf{F}$. $\mathbf{F}$ can be any transformation such as translation transformation, affine transformation or even homography matrix. $\mathbf{F}$ transforms each pixel $\mathbf{m}_r$ of the reference image template $\mathbf{P}_r$ into its corresponding pixel $\mathbf{m}_c$ in the current image template $\mathbf{P}_c$, with the equation:

$$\mathbf{m}_r = F(\mathbf{m}_c, \mathbf{x}) \tag{1.47}$$

The measurement of the mismatch between $\mathbf{P}_r$ and $\mathbf{P}_c$ is defined as the sum of squared differences on intensity (SSD) over the templates:

$$y(\mathbf{x}) = \sum_{\mathbf{m}_c \in \mathbf{P}_c} (\mathbf{I}_r(\mathbf{F}(\mathbf{m}_c, \mathbf{x})) - \mathbf{I}_c(\mathbf{m}_c))^2. \tag{1.48}$$

If the two image templates are perfectly matched, the measurement $y(\mathbf{x})$ will be zero. Template-based tracking problem is reduced to find an optimal transformation $\mathbf{F}(\mathbf{x_0})$, especially it parameter vector $\mathbf{x} = \mathbf{x}_0$, while minimizing the object function $y(\mathbf{x})$, defined in equation (1.48). The function can be minimized using different methods. Here we focus on an efficient second-order minimization method, the so-called ESM tracking, to solve the tracking problem.

In order to minimize the object function 1.48 using numerical calculation methods,

the Taylor series expansion of $y(\mathbf{x})$ about a point $\mathbf{x} = 0$ is:

$$y(\mathbf{x}) = y(\mathbf{0}) + \nabla y(\mathbf{0})\mathbf{x} + \frac{1}{2}\mathbf{x}^\top \nabla\nabla y(\mathbf{0})\mathbf{x} + \dots. \tag{1.49}$$

First order approaches approximate the function $y(\mathbf{x})$ by truncating to the linear term:

$$y(\mathbf{x}) = y(\mathbf{0}) + \nabla y(\mathbf{0})\mathbf{x}. \tag{1.50}$$

Second order approaches approximate the function $y(\mathbf{x})$ by truncating the quadratic term:

$$y(\mathbf{x}) = y(\mathbf{0}) + \nabla y(\mathbf{0})\mathbf{x} + \frac{1}{2}\mathbf{x}^\top \nabla\nabla y(\mathbf{0})\mathbf{x}. \tag{1.51}$$

During the iteration of minimization process, the pixels of $\mathbf{P}_c$ flow to the reference pixels of $\mathbf{P}_r$. First order approach performs low convergence rate. Second order approach performs high convergence rate, however it needs to calculate Hessian matrix which is computational complexity. ESM uses homography matrix, defined in equation (1.28), as the transformation $\mathbf{F}$, and uses an efficient second-order minimization method. It performs high convergence rate and computes efficiently in each iteration. It approximates the second order derivatives by the first order derivatives. The Taylor series expansion of $\nabla y(\mathbf{x})$ about a point $\mathbf{x} = 0$ is:

$$\nabla \mathbf{y}(\mathbf{x}) = \nabla \mathbf{y}(\mathbf{0}) + \mathbf{x}^\top \nabla\nabla \mathbf{y}(\mathbf{0}) + \dots. \tag{1.52}$$

Hence $\mathbf{x}^\top \nabla\nabla \mathbf{y}(\mathbf{0})$ can be approximated as:

$$\mathbf{x}^\top \nabla\nabla \mathbf{y}(\mathbf{0}) \approx \nabla \mathbf{y}(\mathbf{x}) - \nabla \mathbf{y}(\mathbf{0}). \tag{1.53}$$

Equating the approximation of $\mathbf{x}^\top \nabla\nabla \mathbf{y}(\mathbf{0})$ in equation (1.51), we can approximate $y(\mathbf{x})$ as:

$$\mathbf{y}(\mathbf{x}) \approx \mathbf{y}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\mathbf{x}_0))\mathbf{x}_0. \tag{1.54}$$

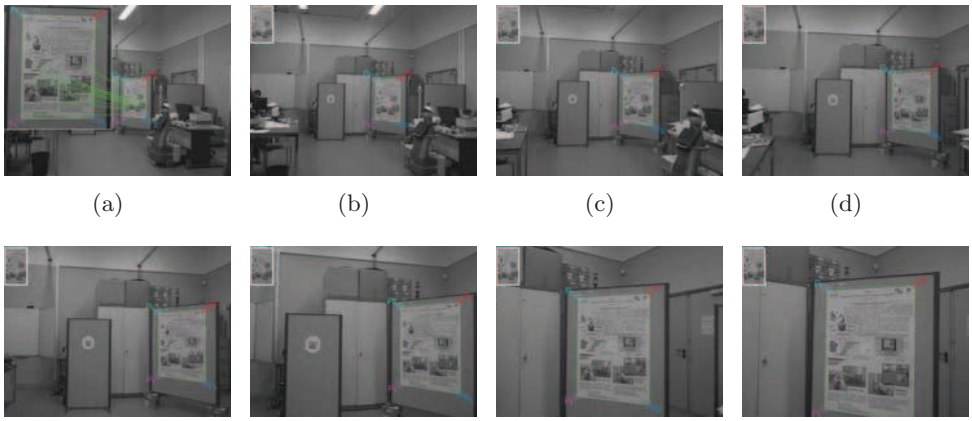
Equation (1.54) is the underlying idea of ESM tracking.

1.6.3 Our Visual Tracking System

After recognizing the target region using SIFT, we find the N best features in the initial image limited in the identified target region.

Figure 1.14 shows the results of ESM tracking. Eight frames were sampled from the test sequence with 769 images. Figure 1.14(a) is the initialization step. The target

to track (the poster) is shown on the left-up sub-region of the image. Figure 1.14(b) to 1.14(h) are the tracking process.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1.14: ESM tracking experimental results.

1.7 Conclusion

In this chapter, we have presented the theoretical bases, overview the literature of feature detectors, descriptors and matching. We have compared the promising approaches of detectors and descriptors to select the suitable one for the place recognition task. In our evaluation, we found that it is hard for a method have robustness with respect to viewing condition changing and computation efficiency at the same time. SIFT was best with changes in scale and was successful in object recognition even when the scale change is large. We select SIFT. However, SIFT is computational complexity. In order to balance computation complexity and effectiveness, we have designed a framework of tracking. In the tracking framework, the result of recognition by SIFT is used to initialize a template-based tracking method, named ESM. The tracking framework can recognize and track a landmark from a long distance. The tracking framework could be a frond-end of vision-based control, which will be discussed in Chapter 3.

Chapter 2

Real-time Reactive Obstacle Avoidance

2.1 Introduction

Safe interaction with the environment is one of the essential components of a robot system. Specifically, for service robots, a flexible, reactive and safety-oriented control of interaction between robots and humans allows a closer cooperation [15]. Therefore, obstacle detection and representation as well as avoidance are crucial tasks for indoor and outdoor mobile robot navigation. This chapter is about reactive collision avoidance for our mobile robot navigating in an unknown environment with obstacles. We aim to seek efficient reactive collision avoidance approaches using a forward-looking 2-D laser range finder. The proposed methods can ensure collision-free motion during mobile robot navigation.

Planning or Reacting? There are mainly two paradigms to deal with collision avoidance in the mobile robotics community: motion planning [74] and reactive obstacle avoidance [69]. Motion planning generates a collision-free motion with differential constraints, kinematic or dynamic constraints, from a start to a goal configuration among obstacles. Motion planning methods provide complete and global solutions to the free-collision problem. However, they assume accurate models of the environment and the robot. Furthermore, these methods have difficulties in case that the environment is unknown and unpredictable. It is also impossible to do replanning in each control loop, especially for large-scale navigation. As Khatib indicated in [69] that the time required to perform global path planning computations may limit "the robot's

real-time capabilities for precise, fast, and highly interactive operations in a cluttered and evolving environment."

Reactive obstacle avoidance is a complementary way to address the collision-free problem. Reactive obstacle avoidance methods drive a robot towards a target configuration without collision with obstacles detected by the sensors within the control loop. Hence, reactive methods are suitable for the uncertain and dynamic environment. On the other hand, since they perform locally, they are inability to generate an optimal solution, even though they may encounter a trap situation without reaching a goal.

Motion planning provides complete and optimal trajectories using a map from a global point of view, whereas obstacle avoidance generates motion reacting to evolving surroundings based on sensor perception from a local point of view. They are parallel approaches to address the collision-free problem. They have a strong complementarity. Various works combine these two paradigms by considering sensor perception into the planning level, namely sensor-based motion planning. For example, the BUG algorithm [93] initially considered as sensor-based motion planning. Some methods deform the planned path in real-time through the onboard sensor as the elastic band method [127] [73].

What is suitable for us? In our vision-based mobile robot navigation framework, which will be introduced in Chapter 3, the navigation task is defined with series of positions defined in the image space without building a metric map. Collision avoidance is used to adapt the robot motion to any contingency incompatible with the visual trajectory. Therefore, we choose reactive obstacle avoidance techniques, which are combined with the vision control in our robot system, to ensure the collision-free motion in real time.

A complete collision avoidance framework is composed basically of three parts: (1) Perception of the environment through the exteroceptive sensors; (2) Collision avoidance algorithm; (3) Robot motion control. During the perception of the environment, obstacles are detected and typically represented by models. The models are then used by a collision avoidance algorithm to calculate the robot motion. In the end, the robot motion is sent to the robot.

However, seldom obstacle avoidance approaches in the literature consider a representation of the obstacle for smooth motion. An appropriate representation can provide good performance in robot motion. This chapter aims at representation of obstacles based on a 2-D laser range finder for unknown and dynamic environments, and applies the proposed representation method to a reactive obstacle avoidance task.

The remainder of this chapter is organized as follows: it begins by presenting relevant methods for obstacle detection and representation, and obstacle avoidance in section 2.2. Section 2.3 focuses on the core of this chapter, by describing obstacle detection and representation algorithms. We begin with the polygonal approximation. Then we introduce a criterion to extract the convex polygonal chain. At last, we use a cubic B-spline to interpolate a smooth curve to represent the obstacle contour. In section 2.4, we present the obstacle avoidance methods based on our proposed obstacle representation. The experimental results are provided in section 2.5. The chapter concludes with a discussion in section 2.6.

2.2 Overview of the State of the Art

In robotics, obstacle avoidance has been a research topic for around three decades. The amount of related work is relatively important. In this section, we present an overview of literature relevant to obstacle representation and avoidance. We are not intended to give an exhaustive overview on path planning and obstacle avoidance. We will limit the scope to the reactive collision avoidance strategy, which is used in this thesis. Consult the textbooks such as [146] [111] for more introductory details.

2.2.1 Obstacle Detection and Representation

The environment representation aims to model the objects in the robot working space for collision checking and motion planning. There are primarily two approaches: configuration space and geometric representation. The configuration space of the robot refers to the set of parameterized positions reachable by the robot system. It initially aims to deal with path planning problems for collision avoidance [90], especially for a robot's end-effector. The planning is quite easy in configuration space, where the robot motion is treated as a point moving in a High-dimensional space. However, configuration space is hard to calculate, especially for a system with high degrees of freedom.

Here we mainly introduce the geometrical representation. In the context of geometric representation, obstacles can be described through an occupancy grids map or geometrical primitives.

Occupancy Grid An occupancy grid [41] or a certainty grid [114] is a matrix of cells. Each cell holds a certainty value that indicates a belief that an obstacle resides in the space represented by this cell. Occupancy grid concept was originally developed by Moravec and Elfes, called certainty grid, in 1980's [113] [40] [115] for mapping

at Carnegie Mellon University. This method uses a probability profile to calculate the certainty value of each cell. It is efficient to handle the inaccuracy of sensors and convenient for sensor fusion [114]. However, this procedure is computationally intensive. Borenstein and Koren extended the certainty grid concept to the histogram grid for obstacle representation [17], which is a two-dimensional Cartesian grid. They simplified the certainty grid concept by a probabilistic distribution to reduce computation. This approach is efficient for computation and suitable for real time obstacle avoidance and robot navigation.

Occupancy grid method can easily be updated upon sensory input, and it can efficiently deal with sensor noise. However, this approach suffers from discretization problems. In addition, it becomes problematic if the robot maps a sizable environment, where the map quickly becomes too large for processing and storage [76].

Geometric Primitives Various 2-D geometric representations of the obstacles are used in practice, for example: points, lines, bounding boxes [148] [124] [161], ellipses [123] [59], and polygonal lines [145] [160] [60].

Box and ellipse shapes are generally used in traffic scenarios with model-based assumptions. They have limitations for unknown object representation, since they are not able to approximate arbitrary data in a precise manner. Polygonal representation is described by a sequence of line segments connected at their vertices. The line segment is the simplest feature among many geometric primitives. Line segment is easy to be used for describing most indoor environments [58], whereas also used for outdoor environment [95]. In the literature [76], polylines are used to represent an Absolute Space Representation (ASR). In the literature [118], different popular line extraction algorithms are evaluated for indoor environment. However, these works have not used the information of the convex and concave of the polygonal line. In this thesis, we consider the convex polygonal line for obstacle representation. Compared with Occupancy grid, representations based on geometric primitives significantly reduce the size of the data. Hence, they are more compact.

2.2.2 Obstacle Avoidance Approaches

In this section, we present the representative approaches that are commonly used in robotics. Obstacle avoidance aims to adapt the robot motion to the current or recent sensor measurements while taking into account the target configuration. The generated motion makes the robot move to the target configuration while the generated trajectory is free of collisions with the obstacles.

Khatib proposed an idea of imaginary forces acting on a robot, called potential field method (PFM) [69]. In this method, obstacles exert repulsive forces, while the target applies an attractive force to the robot. For a given robot position, a resultant force vector is calculated by comprising the sum of a target-directed attractive force and repulsive forces from obstacles. The result force vector acts as the accelerating force on the robot. The robot's new position is updated for a given time interval. The algorithm is then repeated. In another word, a path to the goal is calculated by searching through the valleys of the overall potential field. This approach can be used for planning, and it has been proved popular in different domains. An extension of PFM is the vortex field method, proposed in the literature [35] and used in [34] for local incremental planning. The vortex field method uses a flow rotating around an obstacle instead of the antigradient of the repulsive field. PFM is popular in the field of robots and mobile robots, particularly due to its simplicity. However, PFM has significantly inherent problems as reported in [72]: trap situation (traps to local minima) and oscillations. Trap situations happen, for example, when the robot moves inside a U-shape obstacle or attempts to pass between two closely spaced obstacles. The trap problem can be resolved by heuristic or global recovery. Oscillations become apparent when the PFM is used in a robot with high-speed and real-time performance.

Borenstein and Koren developed Vector Field Histogram (VFH) [18] method for real-time obstacle avoidance at high speed. This method permits the detection of unknown obstacles and avoids collisions while simultaneously steering the mobile robot toward the target. The surrounding of the robot is represented by a histogram grid. An one-dimensional polar histogram is computed from the histogram grid. And then the most suitable open area is selected as the moving direction. VFH was improved by taking into account the robot width and the trajectory of the mobile robot in the literature [156]. It was improved to solve the problem of pure local obstacle avoidance algorithm, dead-ends situation, by using look-ahead verification in the literature [157].

Both PFMs and VFHs do not take system dynamics into account. They aim to select a suitable travel direction. The velocities are then generated based on the selected direction. This will be problematic in cluttered environments and high speed situations. To overcome this issue, some approaches formulate the obstacle avoidance problem in the velocity space of the robot rather than Cartesian or configuration space. The velocity space of a robot is the set of controllable velocities. Thus, these approaches calculate the steering commands directly in the velocity space rather through a travel direction. The representative methods are Curvature-Velocity Method (CVM) [149] and Dynamic window approach (DWA) [52]. CVM treats obstacle avoidance as a

constrained optimization in velocity space. The constrained optimization is defined in terms of speed, safety, and goal-directedness. An optimal point in the velocity space is chosen such that it satisfies all the constraints and maximizes an objective function. DWA is very similar to the CVM in the sense that it uses constrained search in velocity space to determine actuator commands. It also trades off speed, safety, and goal-directedness. However, the grid-based representation makes it more straightforward to compute velocity space obstacles, at the cost of increased memory requirements.

Nearness-Diagram Navigation (ND) is a new Real Time Collision Avoidance Approach proposed in the literature [110]. ND uses a sectored (polar) environment representation that is used to express distances to obstacles and allows selecting an optimal valley. As navigation strategy, five laws of motion are used, selected in an interpretation step.

Randomized Kinodynamic Planning(RRT) [75] is a trajectory planning approach that takes into account kinematic and dynamic constraints. It is approached using Rapidly-exploring Random Trees (RRT), which allows for continuous-domain representation and probabilistically complete planning at the cost of non-optimality. RRTs is a randomized planning technique specially designed for nonholonomic constraints and high dimensional.

The described approaches of representation and obstacle avoidance have advantages and disadvantages depending on the navigation context, such as uncertain worlds, motion at high speeds, motion in confined or troublesome spaces, etc. VFH is suitable to the uncertain worlds. Dynamic window is good at motion at high speed. ND can deal with troublesome spaces. However, none of these methods analyzes the convex and concave of the obstacle boundary when reacting with them. Therefore, in the following sections, we reformulate the obstacle avoidance approaches considering the convex of the obstacle boundary and seeing whether this information can make the robot motion smoother during the obstacle avoidance execution.

2.3 Obstacle Detection and Representation

The statical or moving objects can be hit by the robot, which causes dangerous or undesirable behavior. The objects are irrelevant if they are not in the path generated by a navigation task. Hence, an object is an obstacle if it can be hit in the near future. Our robot system senses obstacles through a 2-D laser range finder. The laser range finder is accurate compared to sonar and radar. Moreover, it is efficient in processing, and unaffected by illumination conditions compared to a vision system. In this section,

we present our obstacle detection and representation approaches based on a 2-D laser range finder. Unlike other obstacle representation methods, we detect and represent obstacles using polygonal lines while analyzing the convex and concave of the obstacle boundary.

2.3.1 Obstacle Detection

As shown in Figure 2.1, a 2-D laser scanner produces a set of 2-D points representing the contour of the visible environment around the sensor. Obstacle detection is usually through the point segmenting and clustering processes to identify each object around the robot. The collision is then checked with the identified objects with a motion prediction step. The process extracts a set of segments out of the raw laser range measurements, and each segment corresponds to an object contour. We segment the raw range measurements after data filtering and then merge the adjacent segments as one using the distance parameters. At last, a predictor is used to select the one which cuts the robot trajectory and can be considered as an obstacle.

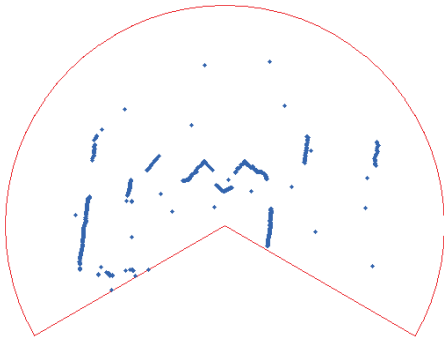


Figure 2.1: Data from a 2-D laser range finder.

Laser Data Consider $\mathcal{M}_t$ a set of points representing the laser scan of an environment at time t as $\mathcal{M}_t = \{m_i = (\rho_i, \theta_i)^\top \mid i \in 0 \dots N-1\}$, where $(\rho_i, \theta_i)^\top$ are the polar coordinates of N points for each scan. The Cartesian coordinates of a laser point are given by

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \rho_i \cos\theta_i \\ \rho_i \sin\theta_i \end{pmatrix} \tag{2.1}$$

The distance between two laser points is defined as

$$\begin{aligned} d(m_i, m_j) =& \parallel m_i - m_j \parallel \\ &= \sqrt{\rho_i^2 + \rho_j^2 - 2\rho_i\rho_j \cos\Delta\theta}, \text{ with } \Delta\theta = \theta_i - \theta_j. \end{aligned} \tag{2.2}$$

Laser range finders are a kind of time-of-flight active ranging sensor using laser light. They provide easily interpreted outputs. They can directly measure the distances from the robot to objects in the neighborhood with high accuracy depending on the quality of the sensor. Ranges are estimated by measuring the difference between transmitted and received signals, for instance, the phase shift. In the applications of occlusion avoidance or collision avoidance, they are widely used on mobile robots.

Data Filtering The first step of obstacle detection based on a laser range finder is to appropriately filter the raw data with sensing errors. For the laser scanner, the main error is "salt and pepper" noise. Sensor errors, multi-path reflection, object surface reflectance and occlusion are all possible causes of the noise.

The noises are the points occurring at stochastic intervals, so they are usually the isolated points of time and space. To be efficient, here we filter the noise considering only one scan. Hence, we primarily discard these spatial isolated points as noise. If multi scans are considered, temporal isolated points can be identified and discard. An spatial isolated point happens in a splitting position of the measurements where there is a big jump with both the two adjacent points. Consider that the set of laser measurements is ordered, and typically, the angular resolution is small $\cos\Delta\theta \approx 1$. Therefore, equation 2.2 could be simplified as

$$d(m_i, m_{i+1}) = \parallel \rho_{i+1} - \rho_i \parallel \tag{2.3}$$

Then isolated points can be found by comparing the distances with a given threshold. Figures 2.2 shows a result of noise filtering from a raw laser scan as shown in Figure 2.1. The threshold used for the result is fixed to $0.1m$.

Data Segmentation The objective of segmentation is to divide a laser scan into meaningful pieces. Each piece corresponds to a visible contour of a sensed object. We use Successive Edge Following (SEF) algorithms described in the literature [144] to segment the filtered laser scans. The SEF algorithm works directly on the distances of the filtered laser scan measurements as illustrated in Figure 2.3. A segment is

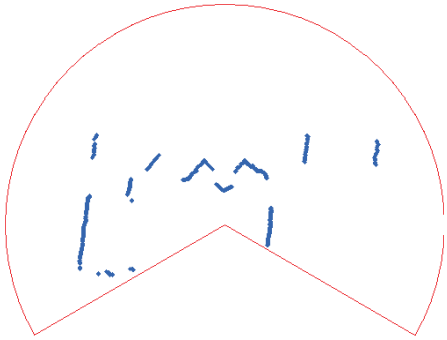


Figure 2.2: Filtered data of laser range finder.

completed when the different of the distance between two adjacent points exceeds a given threshold. After segmentation, the filtered measurements are divided into groups of close points. In order to obtain stable results against variations from the complexity of the environment, the groups are discarded if they consist of too few points, for instance, 3 close points. Figure 2.4 shows different segments in colors using a thresholds 0.2 m.

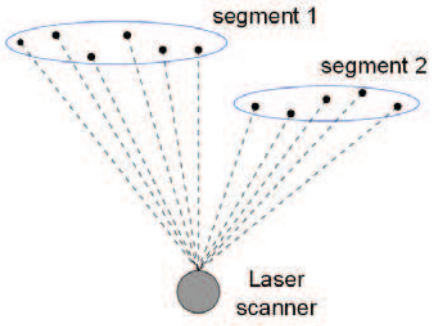


Fig. 2.3: Segmentation using Successive Edge Following algorithm.

Data Merging Due to the noise or occlusion, an object contour may be divided into several adjacent segments, which should be one. In order to cope this issue, we associate the adjacent groups using the angular deviation between two nearest points of the two groups. If the angular deviation is big, the threshold is set to a small value. Otherwise, the threshold is considered big. Figure 2.5 shows the final clustering result with two thresholds 0.2 m and 0.6 m.

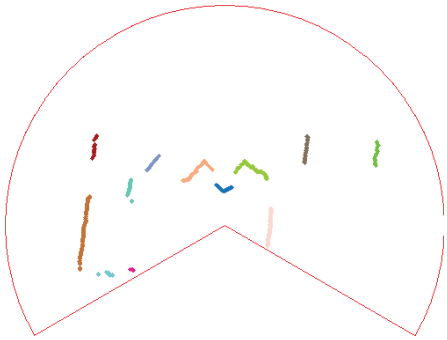


Figure 2.4: Laser data segmentation.

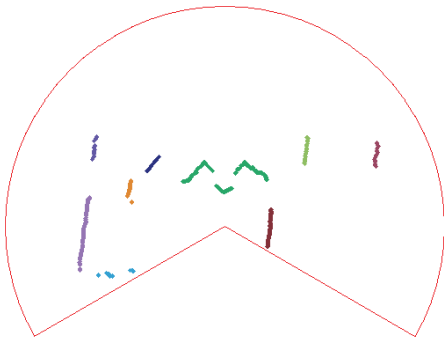


Figure 2.5: Laser data clustering and merging.

Collision Detection Collision detection consists in determining whether a predicted trajectory is clear from obstructions for safe motion by a mobile robot. The prediction simulates the robot motion taking into account the kinematics model of the mobile robot inside a predetermined sampling period $T = \{t_i \mid i \in 0 \dots n-1\}$ (n time intervals) and identifies the interesting obstacles which are in the trajectory using collision checking. In the prediction, the simulated controller is the same as the one used in the real navigation task. The motion model will be introduced in more detail in Section 2.4.1. The given interval is selected to make sure that the robot can safely stop before collision. In each time intervals, the collision is checked through the geometric relationship between the region of the robot and those of clustered objects. More formally, if $\mathcal{R}(Q_{t_i}, T) \bigcap \mathcal{O}(q_{t_i}) \neq \emptyset$, with $\mathcal{R}$ the robot, $\mathcal{O}$ the detected objects and $\mathbf{q}_t$ the configuration at time t, collision will happen. The object is directly modeled by their point sets. This makes the collision checking simple. If collision is detected, the dangerous obstacles are selected, and then the robot speed will be adjusted in reaction to the obstacles. In order to obtain good performance during the reaction, we will analyze the effects with different representations in the following section.

2.3.2 Obstacle Representation

Obstacle representation aims to find a model of the detected obstacle boundary. In this section, we will model the obstacle boundary based on polygonal lines.

2.3.2.1 Polygonal Chain Representation

A polygonal chain or polygonal curve is a piecewise linear curve, which is a connected series of line segments. More formally, let $\mathcal{P}$ be a polygonal chain. $\mathcal{P}$ is specified by a sequence of points $(\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_n)$ called its vertices so that the curve consists of the line segments connecting the consecutive vertices. If the first vertex coincides with the last one, or the first and the last vertices are also connected by a line segment, $\mathcal{P}$ is a closed polygonal chain. Polygonal chains can be used to approximate other curves [130] and boundaries of real-life objects. In the literature [142], closed planar shapes are approximated by polygons in order to decompose 2-D shapes. They can thus also be used to approximate the contours of obstacles. In the case of the 2-D laser scanning parallel to the ground, the boundaries of objects are partly sensed due to occlusion. Therefore, they are approximated by open and simple polygonal chains as shown in Figure 2.6, in which only consecutive (or the first and the last) segments intersect and only at their endpoints.

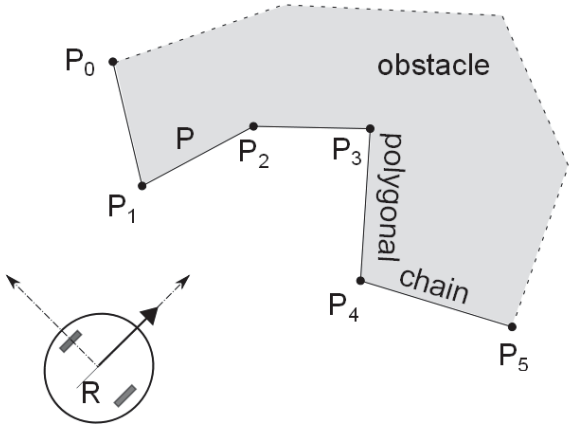


Fig. 2.6: Illustration of a boundary representation by a polygonal chain.

The main goal of extracting polygonal chain with respect to a given curve $\mathbb{C} = \{P_i\}_{i=1}^{N}$ is to identify the vertex locations. $\mathbb{C}$ is a segment based on adjacent groups of range measurements. The polygonal chain extraction can be solved by a line extraction algorithm. The main problem in line extraction algorithm in unknown environments is how many line segments should be used to approximate the given data set, which points assign to which line, and how accurate the line segments can be extracted. In the case of range data, there are several methods to detect line features. A comparison of six popular algorithms can be found in the literature [118]. We select a widely used algorithm, named split-and-merge algorithm, which goes back to the work of Ramer [128], and Douglas and Peucker [39], thanks to its speed and good correctness [118].

The main idea behind the algorithm can be illustrated in Figure 2.7. Suppose a laser scan measurements $\mathcal{M} = \{m_0 \dots m_{n-1}\}$. Initially the algorithm approximates the set of points by the line segment that connects its two endpoints m_0 and m_{n-1}. This approximation is evaluated using a distance criterion and a predefined threshold, noted as thr_p. In another word, the distances of all the points, except the two endpoints, are calculated and compared with the given threshold. If the criterion is not verified, the line segment is sub-divided into two segments at the measurement point m_i, which is the farthest point to the straight line segment. This procedure is recursively repeated until the resulting approximation satisfies the error tolerance specified for the given distance criterion. Figure 2.7(c) shows that $\mathcal{M}$ can be approximated by a polygonal chain with vertices m_0, m_j, m_i, m_{n-1}. $\mathcal{P} = \{P_0, P_1, P_2, P_3\}$ is the extracted polygonal chain as shown in Figure 2.7(d), where P_i is the vertices of the polygonal chain.

To obtain a more accurate polygonal chain, each line segment can be refined using,

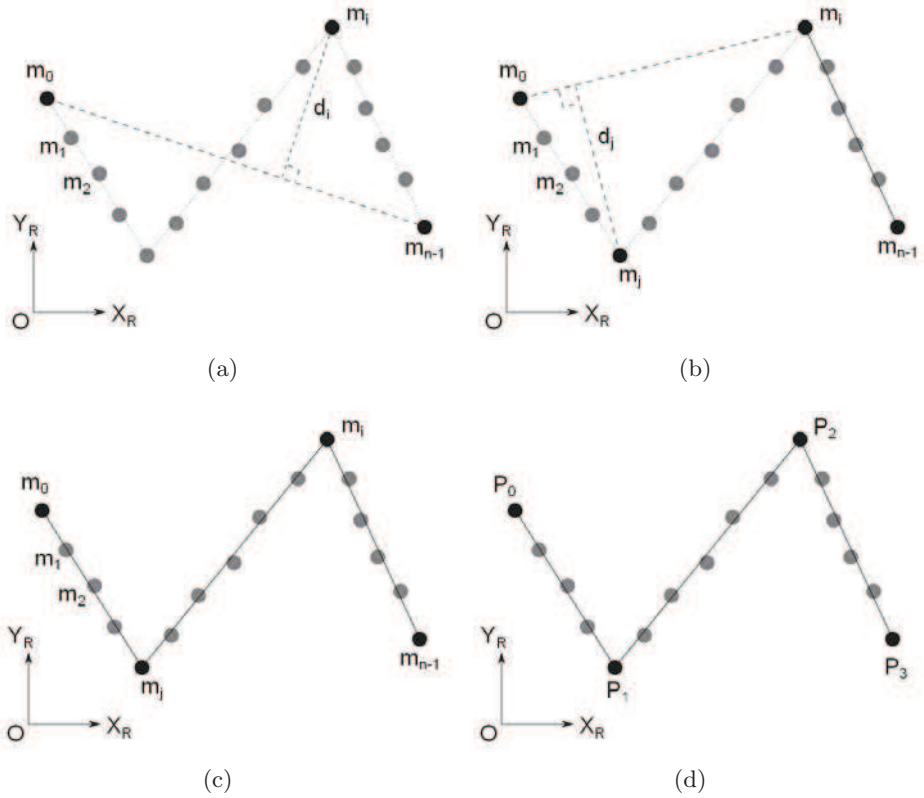
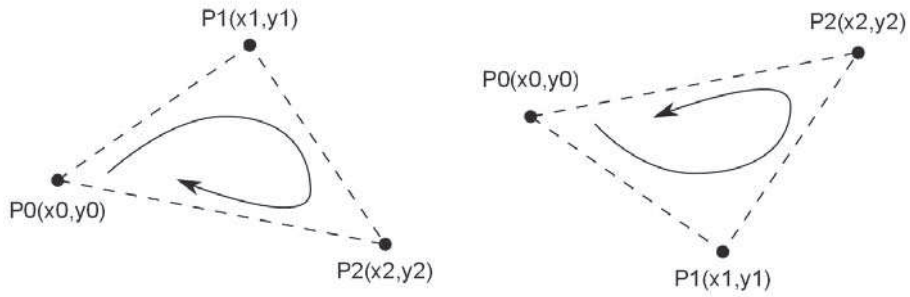


Fig. 2.7: Split-and-merge algorithm applied to a point set.

for example, least squares method to fit a linear regression line [109]. This can make the polygonal chain fits the raw laser measurement much better.

2.3.2.2 Convex Polygonal Chain Representation

Polygonal chain can effectively approximate the boundaries of most objects. However, concave vertices are inadequate for obstacle avoidance when the mobile robot goes around the polygonal chain as shown in Figure 2.6. In order to avoid this situation, the convexity of the extracted polygonal chain should be satisfied. Han proposes in [61] an algorithm to detect convex and concave curves using a signed distance in the split-and-merge algorithm. Similar as Han, in order to extract only convex curve, we use the split-and-merge algorithm [61]. The sign of the distance is given by the relationship between three ordered points, noted $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. The relationship can be established in two ways: in clockwise order or counterclockwise order as shown in Figure 2.8. If $P_1 = (x_1, y_1)$ is left side of the vector $\overline{P_0P_2}$, they are in clockwise order, otherwise, they are in counterclockwise order.



(a) in clockwise order (b) in counterclockwise order

Fig. 2.8: Three points relationship.

The relationship can be identified by the area of the triangular

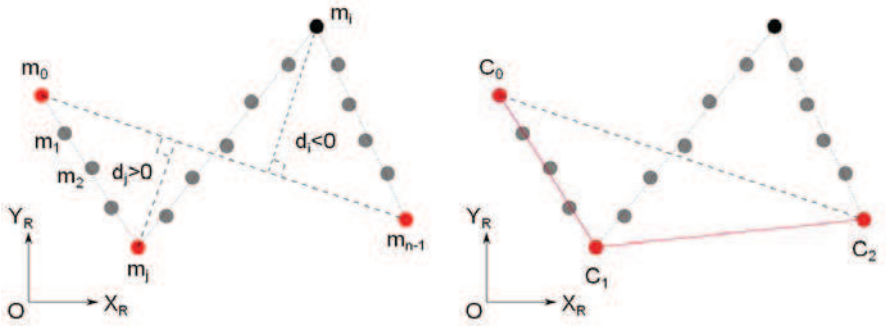
$$a = \frac{1}{2}\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix} \tag{2.4}$$

Equation 2.4 implies the position relationship of the three vertices with the conclusions

– If the area is positive, the points occur in counterclockwise order, and P_1 is right side of the vector $\overline{P_0P_2}$.

– If the area is negative, then they are in clockwise order, and P_1 is left side of the vector $\overline{P_0P_2}$.

As we aim to select the convex points, the point with positive area will be chosen. Since the threshold is also positive, only the points with the biggest distances and bigger than a given threshold thr_p in each iteration are marked as convex points (see Figure 2.9). And thus, convex polygonal chain can be obtained as illustrated in Figure 2.10.



(a) (b)

Figure 2.9: Convex curve extraction.

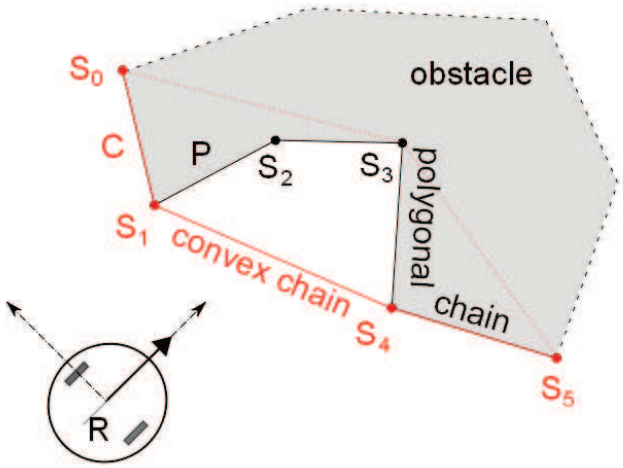


Figure 2.10: Illustration of convex chain.

2.3.2.3 B-Spline Representation

To obtain a smooth representation approximating the obstacle boundary, we use a cubic parametric B-spline curve to model the obstacle boundary by interpolating the vertices of the extracted convex polygonal chain. The advantage of using B-spline curves is that they can have arbitrary order continuity of the obstacle boundaries, which is very suitable for robot motion control during obstacle avoidance. In addition, B-spline can adapt to any curve defined by control points, and its generation is intuitive. Furthermore, they are piecewise polynomials, so they can perform a good approximation with a low degree while avoiding instability due to Runges phenomenon as illustrated in A.1.

Consider $n+1$ vertices points $\mathcal{S} = \{S_0 \dots S_n\}$, the objective is to fit them with a B-spline curve $S(t)$ of degree p with $p <= n+1$. For the sake of simplicity and convenience, we use Bezier curves as the segments of B-spline. We have

$$\begin{cases} S_0 &= B_0 \\ S_1 &= \frac{1}{6}B_0 + \frac{2}{3}B_1 + \frac{1}{6}B_2 \\ &\vdots \\ S_i &= \frac{1}{6}B_{i-1} + \frac{2}{3}B_i + \frac{1}{6}B_{i+1} \\ &\vdots \\ S_{n-1} &= \frac{1}{6}B_{n-2} + \frac{2}{3}B_{n-1} + \frac{1}{6}B_n \\ S_n &= B_n \end{cases} . \tag{2.5}$$

The details how to get equation 2.5 are given in the Appendix A.1. The system 2.5 can be written as

$$\mathbf{MB} = \mathbf{S}, \tag{2.6}$$

where $\mathbf{B} = [B_0 \dots B_n]^\top$, $\mathbf{S} = [S_0 \dots S_n]^\top$ and the $(n+1)\times(n+1)$ coefficient matrix $\mathbf{M}$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{2.7}$$

Hence, $\mathbf{B}$ can be obtained by

$$\mathbf{B} = \mathbf{M}^{-1}\mathbf{S}. \tag{2.8}$$

In practice, we compute $\mathbf{B}$ by solving the linear system 2.6 as given in Appendix B.3 rather than computing the inverse matrix which is time consuming.

Figure 2.11 illustrates an interpolated result (drawn in green color) from a real laser scan applied to the selected polygonal chain (drawn in red color) corresponding to the obstacle to be avoided by the mobile robot.

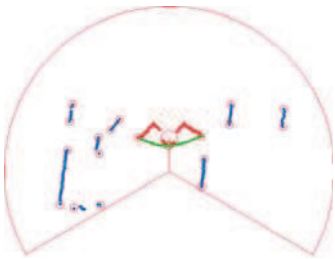


Figure 2.11: Obstacle selecting and boundary fitting

2.3.3 Discussion

This section introduces our obstacle detection and representation approach using a 2-D laser range finder. The raw data is transformed into geometric features, from a polygonal chain to a convex polygonal chain. At last, a cubic B-spline is interpolated based on the extracted convex polygonal chain, and it instantaneously approximates the convex boundary of the obstacle to be avoided. This transformation smoothes the obstacle boundary, but they do not preserve the details of the real boundary of the detected obstacle. The representation by a polygonal chain always leads to a certain loss of information as it is an approximation of the raw data. Since the representation of the convex polygonal chain looses concave features, which makes a rougher approximation. The B-spline curve provides a smooth representation based on the rough approximation. In the case of obstacle avoidance, we note that the B-spline interpolation based on the convex polygonal chain is more interesting, which avoids the concave features and smooths the boundary.

2.4 Reactive Obstacle Avoidance Approaches

This section presents our reactive obstacle avoidance methods using the described representations.

2.4.1 Motion Modeling

The kinematics of our differential drive mobile robot can be described as a unicycle model, which applies to a large class of mobile robots.

With reference to Figure 2.12, let $\mathcal{F}_R$ be the robot frame with the origin located at mid-distance of the robot's actuated wheels. x axis is chosen to assign the robot heading. y axis is along the left. z axis is assigned to up. In the world frame $\mathcal{F}_W$, the workspace of the robot is the xy plane. Under this convention, the configuration of the mobile robot is described by $\mathbf{q} = (x, y, \theta) \in \mathbb{R}^2 \times \mathcal{S}^1$, where $(x,\ y)$ represents the Cartesian position of the reference point $\mathbf{R}$ in $\mathcal{F}_W$, and the parameter $\theta \in [-\pi\ \pi]$ is the robot heading (positive counterclockwise) with respect to the world frame x axis.

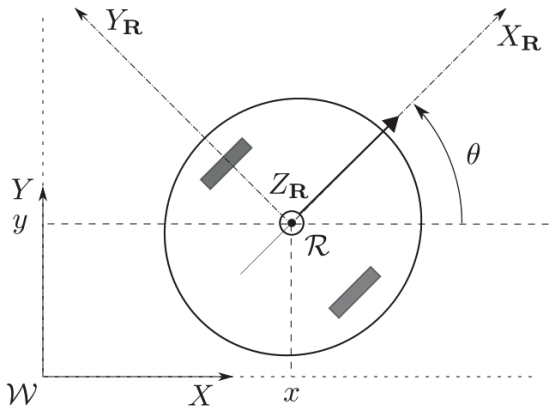


Figure 2.12: Illustration of robot configuration using unicycle model.

We choose $\mathbf{u} = [\upsilon\ \omega]^\top$ as the control inputs for the robot system. υ and ω are respectively the linear and angular velocities (positive counterclockwise) of the robot. Hence, the kinematic model is given as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \upsilon \\ \omega \end{pmatrix}. \tag{2.9}$$

2.4.2 Obstacle Avoidance based on Path Following

Bug algorithm [92] is a typical obstacle avoidance algorithm which follows the contour of each obstacle in the robot's way and thus circumnavigates it. In this section, we also formulate the obstacle avoidance as following the contour of the obstacle.

2.4.2.1 Formalism

In order to use path following method to deal with obstacle avoidance, two envelops ξ_0 and ξ_+ are generated expanding from envelope ξ_{obs} as illustrated in Figure 2.13. In the following, we will see that using the multi envelopes the control can be smoothed when switching between the obstacle avoidance controller and the navigation controller. Let ξ_{obs} represent the approximation of the obstacle's boundary. ξ_0 and ξ_+ surround the obstacle at distance $|d_0| < |d_+|$ representing the risk of collision. When generating ξ_0 and ξ_+, we suppose that a Frenet–Serret frame moves on ξ_{obs} aligning with the robot's moving direction, and d_0 and d_+ are the ordinates defined in the Frenet–Serret frame. ξ_+ defines a zone, inside which the obstacle is detected. We suppose that the distance between each two obstacles is bigger than $2|d_+|$, which ensures that the robot deals with one obstacle at each time. ξ_0 is the reference path to follow when the robot circumnavigates the obstacle. A reference frame $\mathcal{F}_0$ is attached to ξ_0 and the error vector (δ, α, χ) is then calculated.

As described in the section 2.3, we extract the boundary of the obstacle ξ_{obs}. Hence, we, firstly, calculate the error vector $(\delta_{obs}, \alpha_{obs}, \chi_{obs})$ on ξ_{obs}. Then, the error vector $\mathbf{e}$ can be computed as

$$\delta = \delta_{obs} - d_0 \tag{2.10a}$$

$$\alpha = \alpha_{obs} \tag{2.10b}$$

$$\chi = \frac{1}{(r_{obs} + d_0)} = \frac{1}{\frac{1}{\chi_{obs}} + d_0} \tag{2.10c}$$

Path Following In order to control the robot following the path ξ_0, we use the control law proposed in the literature [136]. Here we give an introduction about the path following method used in this thesis.

We assume, without loss of generality, the pre-defined path and the robot position are represented in the global reference frame $\mathcal{F}_O$ as illustrated in Figure 2.14. For a given curve $\mathcal{C}$, the error between the robot frame $\mathcal{F}_R$ and $\mathcal{C}$ has to be stabilized to zero during the path following. To define this error, a reference frame $\mathcal{F}'_R$ is attached to the

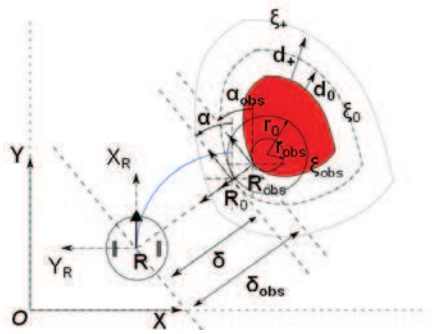


Figure 2.13: Collision avoidance using path following

path $\mathcal{C}$ at point $\mathcal{R}'$, obtained through the orthogonal projection of the origin point $\mathcal{R}$ of $\mathcal{F}_R$ onto $\mathcal{C}$. Therefore, $\mathcal{R}'$ is the closest point on the path $\mathcal{C}$ to the mobile robot. Let r be the curvature radius of $\mathcal{C}$ at the point $\mathcal{R}'$, and θ_r be the angular deviation between frames $\mathcal{F}'_R$ and $\mathcal{F}_O$. Then, the error vector between $\mathcal{F}_R$ and $\mathcal{C}$ can be chosen as $\mathbf{e} = (\delta, \alpha, \chi)$ with

- δ is the distance between $\mathcal{F}_R$ and $\mathcal{F}'_R$.
- $\alpha = \theta - \theta_r$ is the angle characterizing the orientation of the robot with respect to the frame $\mathcal{F}'_R$.
- χ is the curvature of $\mathcal{C}$ at the point $\mathcal{R}'$: $\chi = \frac{1}{r}$, where r is the turning radius. In the case of straight line following, $\chi = 0$.

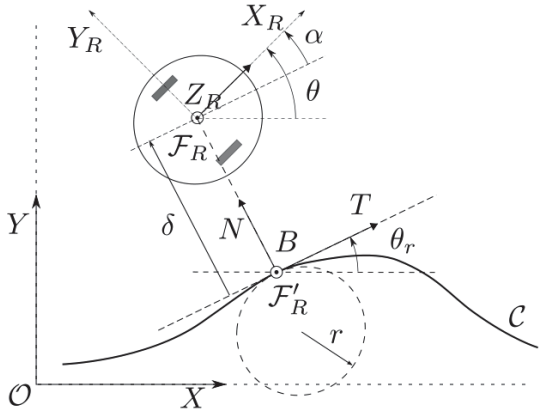


Figure 2.14: Path following.

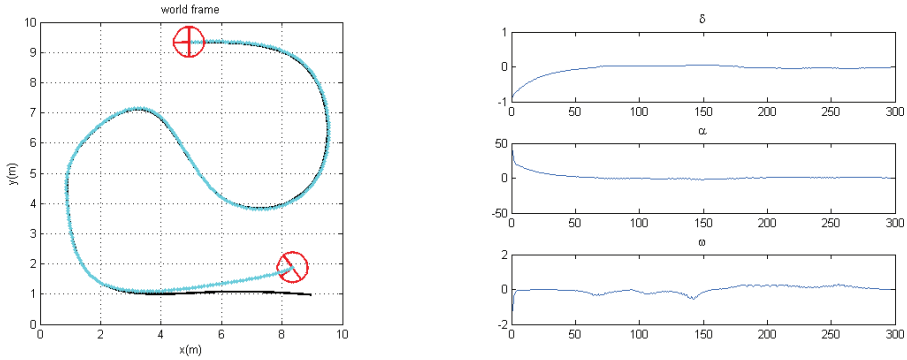
By using the task function formalism [137] and the path following approach [136], the control law can be designed by setting a non-zero linear velocity and an angular

velocity as

$$\omega = -v(k\delta + \alpha + 2k\sin\alpha - \chi\cos\alpha) \tag{2.11}$$

where k is a positive gain to be tuned. The linear velocity can be a constant value or can be a desired profile.

Figure 2.15 is a simulation result of path following with respect to a cubic B-spline curve. The black path is the reference B-spline path. The mobile robot marked red follows the path from the initial position right bottom. The motion of the robot is marked with cyan-blue. We can see that after the robot starts to move, it converges to the defined B-spline curve.



(a) trajectory of motion (b) parameters

Figure 2.15: Path following: cubic B-spline

Conjunction with a Navigation Task By using the control law (2.11), the robot has the capability to move along the obstacle boundary represented by a B-spline when the obstacle is detected during the navigation task. Naturally, the avoidance task is merged with a main task (path following, visual servoing, ...etc). Folio et al. in [50] propose a hybrid control method to merge visual servoing with obstacle avoidance tasks at the control level. Here, we build a multi-task controller as presented in [50]. Here we briefly introduce the control scheme.

Let $\dot{\mathbf{q}}_{go}$ be the control law for goal-driven task, and $\dot{\mathbf{q}}_{co}$ be the control input of collision avoidance. The global controller $\dot{\mathbf{q}}_{glb}$ may be defined by

$$\dot{\mathbf{q}}_{glb} = (1-\lambda)\dot{\mathbf{q}}_{go} + \lambda\dot{\mathbf{q}}_{co}, \quad \lambda \in [0,1] \tag{2.12}$$

where λ is a function allowing to switch between the both controllers. In order to

smooth the global control, the boundary ξ_+ is taken into account to define the collision dangerous zone and to change progressively λ as given in [50]. The λ can be built in two ways: direct switching 2.13 or smooth switching 2.14.

$$\lambda = \begin{cases} 0, & \text{if} \quad |\delta_{obs}| \geq d_0 \\ 1, & \text{if} \quad |\delta_{obs}| < d_0 \quad \text{and} \quad \text{escape} = \text{false} \\ 0, & \text{if} \quad |\delta_{obs}| < d_0 \quad \text{and} \quad \text{escape} = \text{true} \end{cases} \tag{2.13}$$

$$\lambda = \begin{cases} 0, & \text{if} \quad |\delta_{obs}| \geq d_+ \\ \frac{d_+ - |\delta_{obs}|}{d_+ - d_0}, & \text{if} \quad d_0 \leq |\delta_{obs}| < d_+ \quad \text{and} \quad escape = false \\ \frac{d_+ - |\delta_{obs}|}{d_+ - d_e}, & \text{if} \quad d_e \leq |\delta_{obs}| < d_+ \quad \text{and} \quad escape = true \\ 1, & \text{otherwise} \end{cases} \tag{2.14}$$

Escape Condition As indicated above, in order to combine the multi controllers, an escape decision is needed to decide when the obstacle is no more dangerous to the robot. Then the multi controller switches back to the goal-driven task. Consider the process of obstacle avoidance. Two controllers are calculated: $\dot{\mathbf{q}}_{go}$ and $\dot{\mathbf{q}}_{co}$. If the goal-driven control $\dot{\mathbf{q}}_{go}$ makes the motion to conflict the obstacle, the collision avoidance controller $\dot{\mathbf{q}}_{co}$ will aim to decrease the movement tendency or generates a reverse movement trend toward the obstacle. Accordingly, during the collision avoidance process, when $\dot{\mathbf{q}}_{go}$ generates a decreasing movement tendency or a reverse movement trend with respect to the obstacle, the escape condition is contended. The movement trend can be evaluated using the angular velocity with respect to the direction of the obstacle. Thus, the escape is contended "escape = true" under the conditions

$$\begin{cases} \omega_{go} < \omega_{co} & \text{if the robot avoids the obstacle in counterclockwise} \\ \omega_{go} > \omega_{co} & \text{if the robot avoids the obstacle in clockwise} \end{cases} \tag{2.15}$$

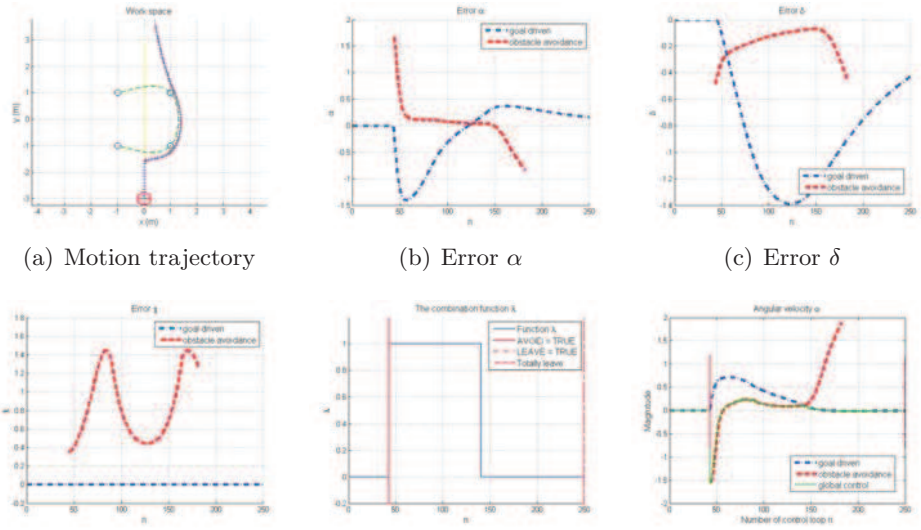
2.4.2.2 Simulation Results

We have simulated a mission whose objective is to navigate the robot along a straight line while avoiding an obstacle placed on the line. The straight line is defined by $x = 0$. The obstacle boundary is modeled by a B-spline, interpolated using four points $(-1, -1)$, $(1, -1)$, $(1, 1)$ and $(-1, 1)$. The sample time is $T_s = 100$ms.

As showed in Figure 2.16(a), the first task is straight line following. The line is defined as $x = 0$. The obstacle is placed on the line at the point $(0, 0)$. The robot starts at the point $(-3, 0)$ heading to the obstacle with constant linear velocity $v = 0.2m/s$.

At the beginning, the robot does not detect the obstacle, so it follows the straight line. As it moves forward, it encounters the obstacle and avoids it.

Figures 2.16 illustrates the simulation results using equation 2.13. Figure 2.16(a) is the motion trajectory in the work space. Figure 2.16(b), 2.16(c), and 2.16(d) give the α, δ and χ calculated from the two controls. Figure 2.16(e) shows the function of λ using switch method. And 2.16(f) is the generated angular velocities.



(a) Motion trajectory (b) Error α (c) Error δ

(d) Error χ (e) Function λ (f) Angular velocity ω

Fig. 2.16: obstacle avoidance simulation: direct switching

The path following based obstacle avoidance could deal with collision avoidance in conjunction with a main navigation task. The method assumes that the robot reacts with one obstacle at one time. In order to adapt to a more cluttered environment, the method should be extended. In addition, it only merges the motion at the control level, not at the task level. It is hard to predict the motion of the robot.

2.4.3 Potential Field Method (PFM)

Path following based obstacle avoidance method is a model based method. The model is the obstacle boundary. In this section, we introduce a model free obstacle avoidance method, namely potential field technique.

2.4.3.1 Formulation

The potential field method (PFM) models an obstacle as a particle moving in a potential field in which the motion is under the influence of a force field. The robot is considered as a particle. While the target generates a force that attracts the particle, and the obstacles generate repulsive forces. The motion of the robot is computed to follow the direction of the artificial force induced by the sum of both potentials. The potential function $U(d_{coll})$ of the repulsive force is given as

$$U(d_{coll}) = \begin{cases} \frac{1}{2}k_1\left(\frac{1}{\delta_{obs}} - \frac{1}{d_+}\right)^2 + \frac{1}{2}k_2(\delta_{obs} - d_+)^2, & \text{if } |\delta_{obs}| \leq d_+ \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

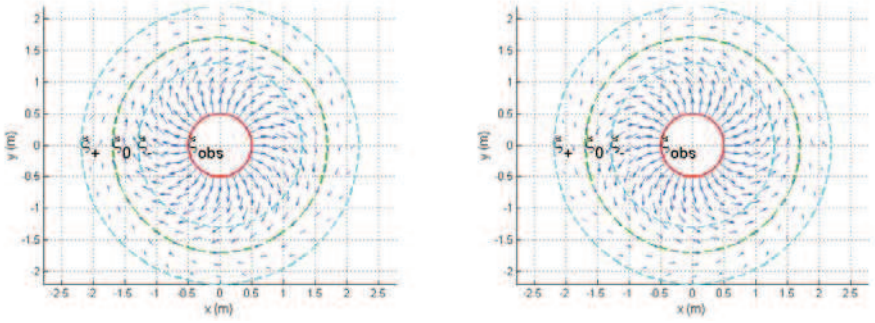
where k_1 and k_2 are positive gains to be chosen. δ_{obs} and d_+ are the same as defined in Section 2.4.2. The modulus of the virtual repulsive force

$$F(\delta_{obs}) = -\frac{\partial U}{\partial \delta_{obs}} = \begin{cases} k_1\frac{1}{\delta_{obs}^2}\left(\frac{1}{\delta_{obs}} - \frac{1}{d_+}\right) - k_2(\delta_{obs} - d_+), & \text{if } |\delta_{obs}| \leq d_+ \\ 0, & \text{otherwise} \end{cases} \tag{2.17}$$

The orientation of the potential force is defined as

$$\beta = \begin{cases} \alpha + \frac{\pi}{2}(1 - \frac{\delta_{obs}}{d_0}), & \text{if clockwise} \\ \alpha - \frac{\pi}{2}(1 - \frac{\delta_{obs}}{d_0}), & \text{otherwise} \end{cases} \tag{2.18}$$

Figure 2.17 illustrates the virtual repulsive force. Note that the virtual repulsive force is related with the configuration of the robot. In order to simplify the illustration, we suppose that the orientation of the robot is always parallel to the obstacle boundary. Figure 2.17 illustrates this potential in the case of a circle obstacle.



(a) Clockwise (b) Counterclockwise

Fig. 2.17: Illustration of virtual repulsive force

The control law can be defined as

$$\dot{q}_{coll} = (v_{coll} \quad \omega_{coll})^\top = (k_v F \cos\beta \quad \frac{k_\omega}{D_x} F \sin\beta)^\top. \tag{2.19}$$

where k_v and k_ω are positive gains to be chosen.

As said in [69], the potential uses the shortest distance to an obstacle $\mathcal{O}$. Thus, δ_{obs} is the shortest distance to the obstacle $\mathcal{O}$. In our work, we can follow the same idea. We can use the closest point of the obstacle boundary to generate the repulsive force. The value of the force is calculated using Equation 2.17. The force direction is set as the normal of the point. Our approach of obstacle representation is convex, which complies with the requirement that equation is continuous and differentiable. Hence, we can calculate the normal of every point on the path.

Potential field approach can be used for path planning. It is accomplished through finding the minimization energy by following the negated gradient of potential energy function. The total repulsive potential field can be obtained by summing up the potentials caused by all of the obstacles.

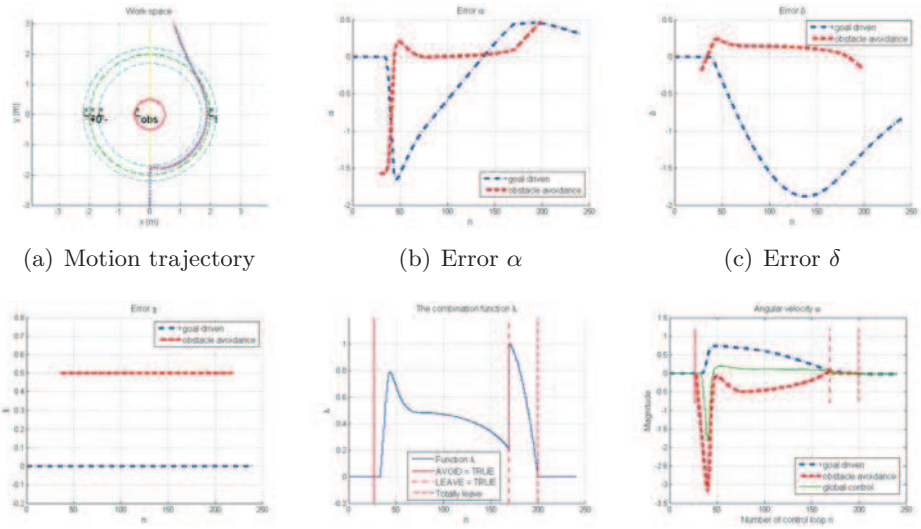
A local-minima free potential field method is proposed in the literature [5]. However, this method has discretization problems. Navigation functions are used in [129]. These functions have the only minimum at the goal position. Cherubini used a vortex potential field, derived from an occupancy grid, in [27] [28] [29].

2.4.3.2 Simulation

A simulation of obstacle avoidance using potential field method is illustrated in Figure 2.18. The robot is following a straight line, while avoiding a circle obstacle on the path. We use the path following control to replace the attracts force in the classical PFM. The different boundaries of regions are used in order to have a smooth motion. It can be noted that the motion is sharper than the direct path following method at the beginning of obstacle avoidance.

2.4.3.3 Discussion

Compared to direct path following method, potential field method may have sharp motion is some cases. However, it can consider multi obstacles. Hence, it is adapted to cluttered environments with obstacles. As a local planar, PFM could encounter a trap-situation, which may occur when the robot runs into a U-shaped obstacle. But this problem can be solved by combining the PFM algorithm with our convex representation



(a) Motion trajectory (b) Error α (c) Error δ

(d) Error χ (e) Function υ (f) Angular velocity ω

Fig. 2.18: obstacle avoidance simulation using PFM: smooth switching

for obstacles.

2.5 Experimental Results

We have implemented the proposed path following based obstacle avoidance method on our mobile robot platform named Lina. The mobile robot is equipped with a laser range finder from Hokuyo Ltd., which has a maximum measurement range of 5 m and is able to scan an angle of 240°. In this experiment, we use visual servoing control scheme to navigate the robot towards a given target. The navigation task is merged with the proposed obstacle avoidance approach to ensure collision free. The obstacles are placed between the initial position of the mobile robot and the desired one.

Three positions of the robot are represented in red circles and number 1 to 3. The black crosses are the laser data. The blue line is the trajectory of the robot. Two green B-spline modeling the obstacle boundary are shown with respect to the position 1 and 2. Figure 2.20 shows the result at each position in the laser space. At the beginning, the robot is outside the dangerous zone of the obstacles. As it moves forward controlled by visual servoing, it encounters an obstacle (position 1). The obstacle is thus segmented, and the convex curve is extracted. Then, the segmented obstacle is modeled through the B-spline curve (see Figure 2.20(a)). The global controller progressively switches to the obstacle avoidance controller, which makes the robot move around the obstacle

smoothly. When the robot can reach the goal without collision (position 2), the global controller goes back to visual servoing controller, and the robot stops when the goal is reached (position 3). The experimental result shows that the robot can move smoothly without being trapped in the concave part of obstacles.

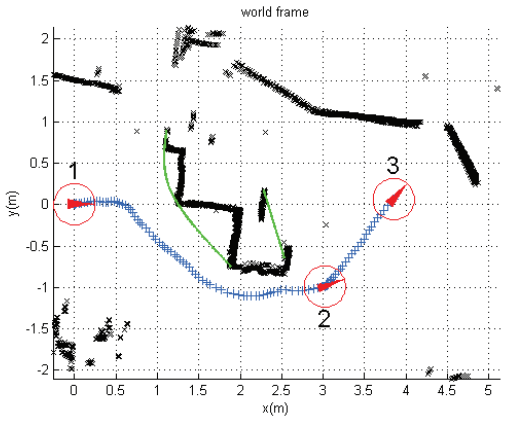


Figure 2.19: Experimental result: navigation

(a) position 1 (b) position 2 (c) position 3

Figure 2.20: Experimental result: laser data

2.6 Conclusion

This chapter presented efficient obstacle detection and representation methods for mobile robot reactive collision avoidance. In the case of unknown and unstructured environments, objects are usually partly sensed, which makes it challenging to have a good representation of the object boundary, especially for an obstacle avoidance task. To model the obstacle boundary, we have extracted the convex polygonal lines of the obstacle boundary followed by B-spline interpolation of the convex polygonal lines. Therefore, we have a smooth representation of the obstacle boundary, and the method is computational efficient. The extracted B-spline is used with path following control for the mobile robot circumnavigating the boundary of the obstacle to avoid it. Since

the model of the obstacle boundary is extracted through convex polygonal lines, the robot can move without being trapped in the concave part of obstacles. Our obstacle modeling method is efficient, since the calculation uses only one frame of laser scan and is inside the control loop. So it is not necessary to build the local map. The simulational and experimental results show that our obstacle representation method performs well enough for the smooth motion when the robot reactives with the potential collision object.

In addition, we have also implemented potential field method, which is a model free obstacle avoidance method. Compared with model-based obstacle avoidance methods, it can be used for more complex environments, since not all the objects can be represented by a model like a B-splines. In the method of extracting convex boundary with path following, we consider one obstacle at the avoidance process. This method may encounter difficulty in the case that the robot needs to enter the concave part of the obstacle boundary in order to avoid the dangerous of other obstacles. However, the convex boundary modeling can be extended to hierarchical obstacle boundary representation to solve the problem.

In the next chapter, we will introduce vision based control for our mobile robot in order to navigate with visual landmarks in the indoor environment.

Chapter 3

Visual Servo Control for Mobile Robots

This chapter considers the problem of visual servo control of a differential drive mobile robot. The aim is to positioning the robot to a desired position using visual servo control while considering three constraints: (1) Field-of-view (FOV) constraints imposed by the camera system; (2) Nonholonomic constraints imposed by the robot kinematics; (3) Safety constraints imposed by the clustered environments. The proposed schemes are tested in real mobile robot navigation tasks.

3.1 Introduction

Most of the recent mobile robot navigation approaches are based on visual servoing techniques. Visual servoing consists in using vision in the robot control loop [66] [26]. It can make the vision based navigation more efficient. Since, compared with model-based mobile robot navigation, visual servoing directly operates in the sensor space without requiring either a precise localization or a map of the environment. Moreover, visual servoing can accomplish various robot tasks due to the vast information acquired from the camera.

Visual servoing approaches were initially developed for 6 DOF manipulators. However, classical visual servoing techniques can not be directly applied to a mobile robot, in particular, a nonholonomic mobile robot with a fixed camera system. For differential drive mobile robots, they have a nonholonomic property that the DOF of the control input, typically linear and angular velocities, is less than that of the robot configuration. This property causes that a pure state feedback stabilization, namely image-based

visual servoing, around a given reference configuration is impossible. Except nonholonomic constraints, when a mobile robot executes a vision-based task in an unknown environment, the robot generally executes multi tasks. Other tasks, notably collision avoidance, are ensured as well as controlling the robot to a goal position using visual servoing. Therefore, it is necessary to combine other controllers with the visual servoing controllers. Moreover, visibility problems can occur due to the field-of-view constraints of the camera system or visual occlusions. They can also come from the motion flexibility limitation imposed by nonholonomic system or affected by the controllers from other tasks.

Therefore, when we consider visual servoing based navigation of a mobile robot, three critical issues should be taken into account: (1) Maneuvering the mobile robot to a desired pose while considering nonholonomic constraints imposed by the robot kinematics; (2) Combining visual servo controller with other tasks, especially the tasks of other sensors such as laser range finder. (3) Visibility of the visual features during servoing, namely field-of-view constraints and visual occlusions. Visibility is more critical when visual servoing is used to enable a mobile robot with a fixed pinhole camera. In this chapter, we will discuss these issues of control a differential drive mobile robot with respect to a goal image using visual seroving while executing collision avoidance task. The visual servo control is based on homography relationship as described in Chapter 1. In particular, we consider the case when the robot avoids the obstacles, the camera could lose its target due to field-of-view constraints and the motion flexibility limitation, which makes visual servoing fail. To handle this, we design strategies by estimating the homography relationship.

The rest of this chapter is organized as follows. Section 3.2 gives an overview the literatures of visual servoing. Section 3.4 presents the visual servoing methods for mobile robots with nonholonomic constraints. Section 3.5 discuss the strategies using visual servoing with complete target loss during collision avoidance. Section 3.6 shows the advantages of using the spherical projection model. Finally, Section 3.7 gives the main conclusions of this chapter.

3.2 State of the Art

Visual servoing designs motion controllers based on visual information using computer vision methods. Visual servoing was initially applied to control manipulators. Vision-based robot control can be classified, depending on the error used to compute the control law, into three groups: position-based, image-based, and hybrid. In a position-

based control system, the error is computed in the 3-D Cartesian space [2] [162] [99] [56] (for this reason, this approach can be called 3-D visual servoing). The position error can be computed using or not the model of the target. If the model of the target is used, we call it model-based 3-D visual servoing. Otherwise, we call it model-free 3-D visual servoing [6] [65]. The position-based visual servoing directly controls the camera trajectory in Cartesian space, therefore it can generate nice behaviors. However, the image features used in the pose estimation for the control law may leave the image, since there is no control in the image. This can lead to servoing failure. Moreover, it suffers from the calibration errors of the camera and the errors in the 3-D model of the target. In an image-based control system, the error is computed in the 2-D image space (for this reason, this approach can be called 2-D visual servoing) [63] [42]. It is robust to the robot calibration errors [43] and other errors from the system, and is a model-free approach. However, the system is coupled and non-linear, which brings the convergence analysis difficult. In an hybrid control system, the error is composed of information in the 2-D image space and in the 3-D space. For example, in [96] a hybrid approach which is called 2 1/2 D visual servoing is proposed. Hybrid approaches can avoid the drawbacks of both the position-based and image-based approaches. Compared to position-based visual servoing, they have the control in the image which can remain the target in the camera field of view. And they simplify the camera pose estimation by estimating partial camera displacement from the current to the desired camera poses without using the target model. Compared to image-based visual servoing, hybrid approaches design a decoupled control law, and thus ensure the convergence of the control law in the whole task space. For more details see [24] [25] for a comprehensive review of state-of-the-art methods, an extensive bibliography and evaluations of each method.

Visual servoing has been extended to mobile robots in many works, for example, by adding extra degree of freedom provided by a hand-eye system [152]. Some works use visual servoing to follow a visual path [100] [20] [133] [9] [141] [38]. Others use visual servoing to position the robot to a desired configuration [98] [10] [86]. Positioning task is more difficult than visual path following task due to the nonholonomic constraints. In the literature [98] [10], epipolar geometry is used to accomplish desired configuration alignment for a nonholonomic mobile robot.

However, all these methods have not take into account obstacle avoidance and the visibility problem of visual targets (or visual features). When a mobile robot uses vision-based control working in an unknown and cluttered environment, which is usually in the presence of obstacles, it is necessary not only to preserve the visual features visibility

during the visual servoing task but also to prevent the mobile robot from collision with the obstacles. Hence, a second task is needed to ensure collision free. In order to combine the vision-based controller with the obstacle avoidance controller, redundancy is used in the literature [51] [27]. In these works, redundancy enable reactive obstacle avoidance without requiring a model of the environment. A robot system is redundant when it has more DOFs than those required for the primary task, so that a secondary task can also be executed simultaneously. In the literature [30], visual navigation and collision avoidance are achieved simultaneously. The camera pan angle is actuated to maintain scene visibility during the collision avoidance. However, this kind of methods normally requires an actuated camera system, which is able to move independently from the robot base [125].

The visibility requires that a minimum number of visual features must remain in the camera field of view (FOV) during the servoing [57]. If these features are lost, visual servoing will fail. Feature loss can be caused by occlusion or by the control generated from other tasks such as obstacle avoidance. Earlier methods, who deal with this problem, are dedicated to avoiding occlusions and loss by preserving all visual features or the whole object visible during the entire execution of the visual servoing task [12] [31] [103] [104] [102] [151]. These methods are dedicated to manipulator arms with 6 DOFs, and they benefit from redundancy. In case of mobile robots, Folio et al. propose in [50] a method to take into account both visibility and collision avoidance.

Central catadioptric camera systems are now widely used in application of visual servoing for increasing the field of view of camera systems. They combine mirrors with a conventional imaging system. Contrary to keeping visibility of all the visual features, a first approach in [57] allows the changes in visibility of the visual features during the servoing. This method is limited to partial changes of visibility. If the appearing visual features are less than a required minimum number or totally loss, the method will also fail. Most of these methods require an actuated camera or an omnidirectional camera systems, and they are dedicated to preserve the image features in the camera field of view or select the visible features. In [85], a homography-based switching control scheme is designed. Three path classes are defined: rotations, straight-line segments, and logarithmic spirals. The control laws as well as switching conditions are defined directly in terms of the entries of homography matrices rather than decomposing the homography to compute the pose parameters. The experimental results have shown that the proposed switching scheme can handle nonholonomic and field-of-view constraints.

However, most of these methods do not take into account the case of the target

complete loss. Since, when the mobile robot circumnavigates the obstacles, the target can be lost due to the motion flexibility limitation of the robot-camera system and the field-of-view constraints of the camera system. In order to handle the total loss of visual features, a method is introduced by Folio et al. in [49]. The method benefits from a visual data estimation algorithm. It uses the previous visual measurements and the control inputs to estimate the current visual features when they are lost. Our work is similar to [49], where visual feature estimation enables the ability of target loss. However, there are various differences with that work. First, Folio et al. use point-like target while we use natural visual target. Second, Folio et al. proposed method is subjected to estimation errors which may increase quickly. Moreover, Folio et al. used a perspective camera model. However, this model is not adapted when a field of view is greater than 180°. Since in Equation 1.24, we suppose that the optical center of the camera never passes through the plane. However, that case may happen during the obstacle avoidance task. Therefore, the planar homography constraint is not suitable. In order to solve this problem, we have adopted an spherical image representation.

3.3 Robot-Vision System Configuration

We consider a two wheeled differential drive mobile robot equipped with a fixed pinhole camera, a 2D laser range finder and odometry sensor.

We consider the motion of the mobile robot in the inertial frame, namely world frame $\mathcal{F}_W$. Let $\mathcal{F}_R$ be the frame attached to the mobile robot. The origin R will be termed reference point of the mobile robot. The x-axis is aligned with the heading of the robot. The z-axis is perpendicular to the motion plane. For more details about the mobile robot configuration definition see section 2.4.1.

We consider a perspective camera fixed on the mobile robot. Let $\mathcal{F}_C$ be the camera frame. The origin C will be termed optical center, and the z-axis is the optical axis and is parallel to the x axis of the mobile robot frame. A pinhole camera model is considered as shown in Figure 3.1(b). Let $\mathcal{P}$ be a 3-D point with homogeneous coordinates $\mathbf{X} = [X\ Y\ Z\ 1]^\top$ with respect to $\mathcal{F}_C$. The point $\mathcal{P}$ is projected into the normalized image plane to a point of homogeneous coordinates $\mathbf{x} = [x\ y\ 1]^\top$. The image point $\mathbf{m} = [u, v, 1]$ is the homogeneous coordinates given in pixel as $\mathbf{m} = \mathbf{K}\mathbf{x}$, where $\mathbf{K}$ is the camera intrinsic matrix, obtained after calibration.

Let $\mathbf{v}_c = [v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^\top$ be the camera kinematic screw, and $\mathbf{v}_r = [v\ \omega]^\top$ be

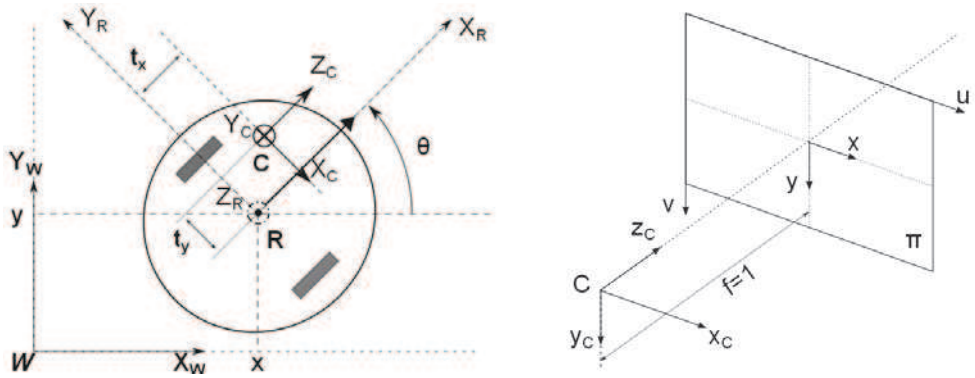
the control input of the mobile robot. For such configuration, they are related as

$$\mathbf{v}_c = \mathbf{J}\,\mathbf{v}_r \tag{3.1}$$

with

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_\upsilon & \mathbf{J}_\omega \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ -t_x & 0 & -t_y & 0 & -1 & 0 \end{bmatrix}^\top \tag{3.2}$$

where $\mathbf{J}_\upsilon$ and $\mathbf{J}_\omega$ include the first and the second columns of $\mathbf{J}$ respectively. The parameters t_x and t_y are the displacements between the robot frame and the camera frame.



(a) Model of non-holonomic wheeled mobile robot with camera

(b) Pinhole camera model

Figure 3.1: System modeling

3.4 Visual Servoing for Mobile Robot

3.4.1 General Formulation of Visual Servoing

Visual servoing refers to control the motion of a robot using vision data. The vision data may be acquired from a camera in different configurations. The camera can be carried by the robot observing the target or fixed in the world observing both the target and the robot. We consider only the former one, which is referred to eye-in-hand configuration, since it is typically used in mobile robot navigation.

Vision-based control schemes can be generally described as minimizing an error $\mathbf{e}(t)$, which is defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*, \tag{3.3}$$

with $\mathbf{s}$ is the current selected visual feature vector and $\mathbf{s}^*$ is the desired one. As we will see further, there are different ways to select $\mathbf{s}$. Whatever the choice of $\mathbf{s}$, they are computed from image measurements $\mathbf{m}(t)$. The parameter $\mathbf{a}$ can be the camera intrinsic parameters and/or 3-D model of objects. Thus, visual feature $\mathbf{s}$ can be noted as function as $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$. After given a desired visual feature vector $\mathbf{s}^*$, the error $\mathbf{e}(t)$ can be calculated as in equation (3.3).

We consider here the case of controlling the six degrees of freedom (6 DOF), through the velocity $\mathbf{v}_c(\boldsymbol{\upsilon}, \boldsymbol{\omega})$, where $\boldsymbol{\upsilon}$ is the instantaneous linear velocities of the camera frame, and $\boldsymbol{\omega}$ is the instantaneous angular velocities. Since the motion of the camera causes the motion of the observed visual features $\mathbf{s}$, a relationship between the time variation of $\mathbf{s}$ and the camera velocity $\mathbf{v}_c$ can be established as

$$\dot{\mathbf{s}} = \mathbf{L}_\mathbf{s}\mathbf{v}_c. \tag{3.4}$$

with $\mathbf{L}_s \in \mathbb{R}^{k\times 6}$ is termed the interaction matrix related to $\mathbf{s}$, k is the dimensions of $\mathbf{s}$. It encodes the relationship between the motion of visual features and the motion of camera. Considering a motionless target and a fixed desired configuration, the time variation of $\mathbf{e}$ is

$$\dot{\mathbf{e}} = \mathbf{L}_\mathbf{s}\mathbf{v}_c. \tag{3.5}$$

The velocity controller $\mathbf{v}_c$ can be thus obtained using equation (3.5)

$$\mathbf{v}_c = \mathbf{L}_\mathbf{s}{}^+\dot{\mathbf{e}}, \tag{3.6}$$

where $\mathbf{L}_\mathbf{s}{}^+ \in \mathbb{R}^{6\times k}$ is the Moore-Penrose pseudoinverse of $\mathbf{L}_\mathbf{s}$ ($\mathbf{L}_\mathbf{s}{}^+ = (\mathbf{L}_\mathbf{s}{}^\top\mathbf{L}_\mathbf{s})^{-1}\mathbf{L}_\mathbf{s}{}^\top$). To ensure a solution of the linear system, $\mathbf{L}_\mathbf{s}$ should be of full rank 6.

Equation (3.6) is the general form of control law namely used in VS. Since $\mathbf{e}$ is defined in the space of features $\mathbf{s}$, equation (3.6) indicates that the camera motion is controlled in the feature space. Some special motions of $\mathbf{s}$ can be considered. This refers to the works of path planning in the feature space or feature-based trajectory generation [46] [102] [104]. A simple and straightforward approach is using a linear controller in the feature space, given by

$$\dot{\mathbf{s}} = \lambda(\mathbf{s}^* - \mathbf{s}). \tag{3.7}$$

Using this controller, the motion of $\mathbf{s}$ is expected to move along a straight line from the initial state of $\mathbf{s}$ to the desired one $\mathbf{s}^*$. Considering equation (3.3), the control law

in equation (3.6) can be rewritten as

$$\mathbf{v}_c = -\lambda \mathbf{L_s}^{+} \mathbf{e}. \tag{3.8}$$

Note that an estimation of $\mathbf{L_s}$ or $\mathbf{L_s}^{+}$ should be used. As we will see further, some elements of $\mathbf{L_s}$ include the 3-D parameters that are not available directly from the image measurements. We denote the approximation by $\widehat{\mathbf{L_e}^{+}}$. Then, the control law in real setup (or experiment) is given by

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L_s}^{+}} \mathbf{e} = -\lambda \widehat{\mathbf{L_s}^{+}} (\mathbf{s} - \mathbf{s}^*). \tag{3.9}$$

Equation (3.9) is the basic implementation of most visual servoing control. When coming to a specific visual servoing task, we need to accomplish selecting a visual feature $\mathbf{s}$, deducing the interaction matrix $\mathbf{L_s}$ with respect to $\mathbf{s}$, and then estimating both $\mathbf{e}$ and $\mathbf{L_s}$ using computer vision data in order to calculate velocity controller $\mathbf{v}_c$. Features vector $\mathbf{s}$ can be designed in any space, typically in the image space, the 3-D Cartesian space, or both. In addition, they can consist of any information such as point, line, pose or even combination of them. These are the rule, by which visual servoing is commonly classified. Depending on the selected feature space, visual servoing schemes are commonly classified into image-based visual servoing(IBVS), position-based visual servoing(PBVS), and hybrid visual servoing(HVS).

In the following, we will introduce the three classes of visual servoing approaches: IBVS, PBVS, and hybrid approach. For each class of approaches, we first give the formulation for 6 DOF camera system. Then we explain how adapt them to control mobile robots.

3.4.2 Image-Based Visual Servoing (IBVS)

In the camera imaging formation, the image of the target is a function of the relative pose ${}^{C}\mathbf{T}_{T}$ of the target with respect to the camera. The motion of camera causes the motion in the image, which is known as optical flow. In other words, the features in the image space imply the relative pose of the target and the camera, and the optical flow involves the relative motion between them. It is possible to control the Cartesian motion of the camera through directly the image features for positioning tasks. This is referred to IBVS, where the visual features are defined on the image plane and the control task is performed directly in the image plane.

We consider an image point feature with 2-D normalized coordinates $\mathbf{p} = (x, y)$

as the visual features. The image measurements $\mathbf{m}$ of $\mathbf{p}$ are the coordinates of the image point in pixel. $\mathbf{p}$ is calculated by transforming the point coordinates from pixel coordinates to normalized image coordinates with the camera intrinsic parameters. The camera can be calibrated using for example the calibration toolbox [19].

The interaction matrix $\mathbf{L_p}$ related to the point feature $\mathbf{p}$ is given in [23] by

$$\mathbf{L_p} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{3.10}$$

Note that $\mathbf{L_p}$ includes the depth Z of the 3-D point corresponding to $\mathbf{p}$. It is possible to use an approximation of Z, since IBVS is remarkably tolerant to errors of Z in practice. A number of approaches have been proposed to approximate $\mathbf{L_p}$. The depth Z can be simply given as a constant value, or can be estimated using computer vision techniques [6].

Here we formulate IBVS for mobile robots taking into account the robot-vision jacobian given in equation (3.1), which linking the camera velocity to the one of the mobile robot, we obtain

$$\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s},\upsilon}\,\upsilon + \mathbf{L}_{\mathbf{s},\omega}\,\omega \tag{3.11}$$

where $\mathbf{L}_{\mathbf{s},\upsilon} = \mathbf{L_s}\mathbf{J}_\upsilon$ and $\mathbf{L}_{\mathbf{s},\omega} = \mathbf{L_s}\mathbf{J}_\omega$. For a desired linear velocity υ, the angular velocity can be obtained as

$$\omega = -\mathbf{L}^{+}_{\mathbf{s},\omega}(\lambda\mathbf{e} + \mathbf{L}_{\mathbf{s},\upsilon}\,\upsilon) \tag{3.12}$$

Note that the linear velocity υ can be assigned to a constant value or computed from a designed velocity profile.

In the real experiments, the interaction matrix can be different depending on the selections of features and the approximation of the depth Z. Here we give some designations.

Image jacobian points controller

In the image jacobian points controller (IJP), the visual features are the current image coordinates of the image point. The interaction matrices $\mathbf{L}_{\mathbf{s},\upsilon}$ and $\mathbf{L}_{\mathbf{s},\omega}$ are given as

$$\mathbf{L}_{\mathbf{s},\upsilon} = \begin{bmatrix} \frac{x}{\widehat{Z}} \\ \frac{y}{\widehat{Z}} \end{bmatrix} \qquad \mathbf{L}_{\mathbf{s},\omega} = \begin{bmatrix} -\frac{t_x}{\widehat{Z}} - \frac{xt_y}{\widehat{Z}} + 1 + x^2 \\ xy \end{bmatrix} \tag{3.13}$$

where $\widehat{Z}$ is the estimation of the depth Z.

Image jacobian points controller with uniform depths

The image jacobian points controller with uniform depths (IJPU) uses uniform depths instead of estimations of Z as in IJP. For example, the depths of all point features can be set to a constant $\overline{Z}$

$$Z_i = \overline{Z} \qquad \forall i = 1, \ldots, n. \tag{3.14}$$

Then we have the following interaction matrix

$$\mathbf{L}_{\mathbf{s},\upsilon} = \begin{bmatrix} \frac{x}{\overline{Z}} \\ \frac{y}{\overline{Z}} \end{bmatrix} \qquad \mathbf{L}_{\mathbf{s},\omega} = \begin{bmatrix} -\frac{t_x}{\overline{Z}} - \frac{xt_y}{\overline{Z}} + 1 + x^2 \\ xy \end{bmatrix} \tag{3.15}$$

Image jacobian centroid controller

Image jacobian centroid controller (IJC) uses the current coordinates of the centroid of the n points as the visual features, rather than directly the coordinates of points as IJP. The visual feature is described thus by

$$\mathbf{s} = \begin{bmatrix} x_g \\ y_g \end{bmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \in \mathbb{R}^2. \tag{3.16}$$

The error to be minimized is given by

$$\mathbf{e} = \begin{bmatrix} x_g - x_g^* \\ y_g - y_g^* \end{bmatrix} \in \mathbb{R}^2. \tag{3.17}$$

and the corresponding interaction matrix is given by

$$\mathbf{L}_{C,\upsilon} = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} \frac{x_i}{Z_i} \\ \frac{y_i}{Z_i} \end{bmatrix} \qquad \mathbf{L}_{C,\omega} = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} -\frac{t_x}{Z_i} - \frac{x_i t_y}{Z_i} + 1 + x_i^2 \\ x_i y_i \end{bmatrix} \tag{3.18}$$

Image jacobian centroid controller with uniform depths

In the same way as in IJPU, image jacobian centroid controller with uniform depths (IJCU) uses uniform depths instead of estimations $Z_i = \overline{Z} \quad \forall i = 1, \ldots, n$.

3.4.3 Position-Based Visual Servoing (PBVS)

PBVS uses observed visual features to estimate the pose ${}^{C}\widehat{\mathbf{T}}_{T}$ of the target with respect to the camera. There are varied pose estimation algorithms. A typically way is the use of observed visual features with the known information of camera calibration and the target model. With a given desired pose ${}^{C^*}\mathbf{T}_{T}$, an error is calculated in the workspace,

which is commonly $SE(3)$. For example, as shown in Figure 3.2 the error is $\Delta\mathbf{T} = {}^{C}\mathbf{T}_{C^*} = {}^{C}\widehat{\mathbf{T}}_{T}({}^{C^*}\mathbf{T}_{T})^{-1}$. The control is then performed in the workspace in order to bring $\Delta\mathbf{T}$ to identity. Hence, the camera is positioned toward the desired pose ${}^{C^*}\mathbf{T}_{O}$ and the robot moves to the expected pose.

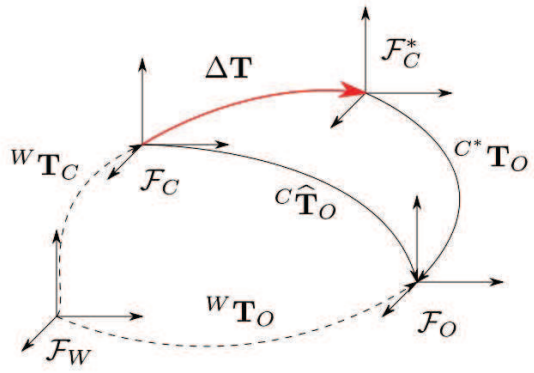


Figure 3.2: The illustration of PBVS.

Position-based visual servoing select the visual features in the 3-D space. The visual features $\mathbf{s}$ can be defined to be $[\mathbf{t}\ \theta\mathbf{u}]^{\top}$, where $\mathbf{t}$ is the translation vector and $\theta\mathbf{u}$ is the orientation vector. The corresponding interaction matrix is given by

$$\mathbf{L_e} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \tag{3.19}$$

The control law is given by

$$\begin{cases} \upsilon_c &= -\lambda\mathbf{R}^{\top}\mathbf{t} \\ \omega_c &= -\lambda\theta\mathbf{u} \end{cases}. \tag{3.20}$$

To control the angular velocity, a direct way is to use a heading controller to align the camera to the centroid of the visual object. The control error is defined by an angular error $\Delta\theta = \theta - \theta^*$. Then, the control law has the following form

$$\omega = -\lambda\Delta\theta. \tag{3.21}$$

Note that only the rotation matrix $\mathbf{R}$ is enough to derive the angular error.

3.4.4 2D 1/2 Visual Servoing (HVS)

Hybrid visual servoing method [97] calculates the control error partially in image space and partially in Cartesian space. It can avoid some of the limitations of the IBVS and the PBVS.

The visual feature can be defined as [97]

$$\mathbf{s} = \begin{bmatrix} \mathbf{p}_e^\top & \mathbf{u}^\top\theta \end{bmatrix}^\top. \tag{3.22}$$

where $\mathbf{p}_e$ is the extended image coordinates as follows

$$\mathbf{p}_e = \begin{bmatrix} x & y & \log(\frac{Z}{d^*}) \end{bmatrix}^\top. \tag{3.23}$$

Here $\mathbf{p}_e$ is assigned to a selected point, for example the center moment of the a set of points.

The interaction matrix is

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_\upsilon & \mathbf{L}_\omega \\ \mathbf{0}_3 & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \tag{3.24}$$

with

$$\mathbf{L}_\upsilon = -\frac{1}{Z^*\rho_Z}\begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix}, \tag{3.25}$$

and

$$\mathbf{L}_\omega = \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \\ -y & x & 0 \end{bmatrix}, \tag{3.26}$$

The interaction matrix given in equation (3.24) has the triangular form. The hybrid visual servo scheme decouples the rotational motions from the translational ones. Hence, the scheme exhibits decoupling properties.

3.5 Visual Servoing Based Navigation with Complete Target Loss

3.5.1 Problem Statement

We consider a mobile robot equipped with a fixed pin-hole camera and a forward-looking laser range finder. The desired pose of the robot is defined by a natural image

of the observed planar target previously taken at the goal pose. Visual servoing is used to control the robot from an initial pose to the desired one. During the visual servoing task, the current image at time t is captured at the current pose, which is related to the desired pose by a rotation matrix $\mathbf{R}(t)$ and translation vector $\mathbf{t}(t)$. The current and desired images are related by a homography matrix $\mathbf{H}_N \in \mathbb{R}^{3\times3}$ as described in Section 1.2.3, and the vision-based controller we used is based on this relationship. By supposing that the planar visual target is fixed, only rotation matrix $\mathbf{R}(t)$ and translation vector $\mathbf{t}(t)$ change during the robot motion, while both the normal vector $\mathbf{n}$ and the distance d are constant. Taking into account the planar motion and the vertical planar target, the homography has the form given in equation (1.40).

When the robot executes vision-based navigation in an unknown and cluttered environment, it is necessary not only to preserve the image features visibility during the visual servoing execution but also to prevent the mobile robot from collision with the obstacles as described in Chapter 2. However, when the robot circumnavigates the obstacles, the target can be lost due to the fixed camera-robot configuration and the field-of-view constraints of the camera system (see Figure 3.3). Visual servoing will thus fail. This is a challenging problem in image-based visual servoing and appearance-based navigation. The robot can not move without the information of the visual target. Although some of the problems can be solved using omnidirectional vision enlarging the field of view or actuated cameras preserving the visual features, it still has difficulty to the entire lost case.

In this section, we focus on the problem of visual target loss in the camera field of view, but it can adapted to general target lost situations, for example, occlusion. Inspired by path integration and visual landmark reaction in animal navigation, we propose a strategy to deal with the loss of visual features by taking advantage of the odometric data sensing.

In order to solve the problem, we consider the animal navigation ability. Many animals have strong navigation abilities of path integration and visual landmark reaction [112]. Path integration refers to the ability that an animal can keep a continuously updated record of its current direction and distance from some reference point as they move away from that place. With such ability, the animal can reach other places of which it knows the path integration coordinates. The precision of reaching the places depends on the accuracy of its position estimation through path integration. Etienne et al. [44] proved that the inevitable errors in path integration can be reduced through information supplied by landmarks. And they show that hamsters use visual landmarks to reset their path integrator.

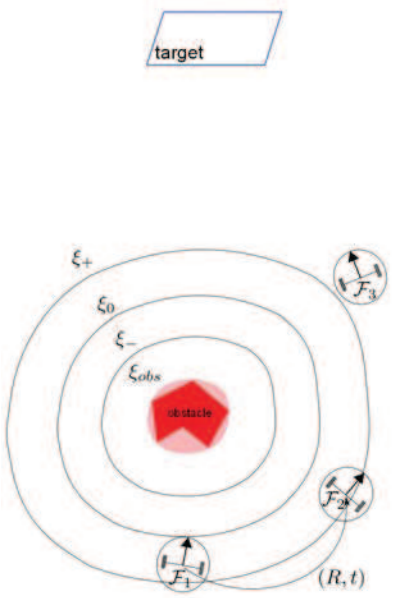


Figure 3.3: Illustration of visual servoing with complete target loss.

Inspired by animal navigation, the invisible visual targets can be recovered by path integration during navigation in order to ensure visual servoing task or reacquire the real target. Path integration can be accomplished by motion estimation methods. The error of integration can be reduced through visual landmark reaction by an active sensing process, and visual tracking is reset. This is the basic idea of our strategy to deal with the problem of target loss during navigation. A similar work of using feature estimation to handle feature loss during visual servoing control is described in [49]. Our work has some differences which will be introduced in the sequel.

This section is organized as follows. Section 3.5.2 introduces the structure of our navigation system. Section 3.5.3 and Section 3.5.4 describe the behavior designation and motion generation components of the navigation system. Section 3.5.5 gives our strategy to handle target complete loss. Section 3.5.6 shows the experimental results.

3.5.2 Navigation System

The system of our vision-based navigation can be decomposed into four major blocks (see Figure 3.4):

– Behavior Generation: This block generates a suitable behavior for the Motion Generation and Perception components in order to select a predefined task for

those two components to solve. It works as a state machine, which decomposes a navigation task into several basic behaviors (see Figure 3.5). These basic behaviors include visual target tracking and servoing, collision detection, collision avoidance, target re-acquisition, and robot stop. The state machine will select a task due to the current perception results and the previous behavior.

– Motion Generation: This component consists of several controllers, including goal-driven controller based on visual servoing, collision avoidance controller based on laser range finder and target re-acquisition controller. The visual servoing controller drives the mobile robot toward the desired configuration. The collision avoidance controller ensures the motion safety in the presence of obstacles. The target re-acquisition controller can align the robot heading to the lost target if necessary. Each controller computes a result from the current perception information. The results from each controller are then combined to drive the mobile robot achieving a desired configuration while ensuring collision free. The combination is computed with the given behavior.

– Perception & Estimation: The Perception and Estimation component processes the data from different sensors in order to detect and track the visual landmarks, and detect approaching obstacles. The processing results are then used by other components. The visual tracking results is used for visual servoing control. Collision detection and obstacle modeling is used to invoke collision avoidance task.

– Sensors & Actuators: The Sensor & Actuators component provides a way to interact with the environment. Our mobile robot system includes a stereo vision system, a laser range finder, odometric sensors, and actuators. We use mono vision to detect and track visual target. The odometry does not provide sufficiently accurate motion measurements, so we use the stereo vision system to estimate the robot motion. A forward-looking range finder is used to detect collision and modeling the obstacles.

3.5.3 Behavior Generation Algorithm

In the navigation system (see Figure 3.4), the Target Detection & Tracking is described in Chapter 1. Collision Detection & Obstacle Modeling and Collision Avoidance are described in Chapter 2. Here we give the algorithm of Behavior Generation. The rest of parts of our navigation system will be given in the further.

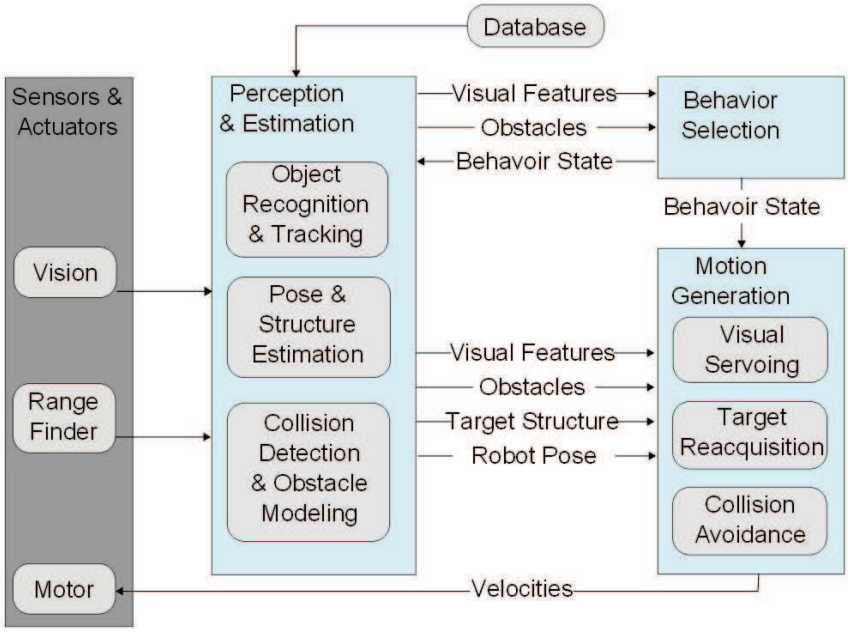


Figure 3.4: Structure of navigation system.

The Behavior Generation component is designed based on the requirements to accomplish the navigation task, which is moving towards a desired configuration while avoiding obstacles. We consider a positioning task. In our designation (see Figure 3.5), six basic behaviors are object recognition behavior, goal-driven behavior, motion prediction behavior, collision avoidance behavior, visual data estimation behavior and target re-acquisition behavior. The motion prediction and collision detection behaviors work during the entire navigation process, since safety is the most important. The object recognition behavior always comes at the beginning of the navigation task. If the object is recognized, the visual tracking is initialized and the goal driven behavior is executed using visual servoing control. If the potential collision is detected, collision avoidance and vision-based estimation behaviors are invoked. The mobile robot circumnavigates the detected obstacle and estimates the target in the image space if lost. When the obstacle is avoided, target re-acquisition behavior is executed to align the robot heading to the target, then the object recognition is restarted.

3.5.4 Motion Generation Algorithm

The motion generation component is dedicated to compute the linear and angular velocities for the motion. There are three controllers: visual servoing controller, collision

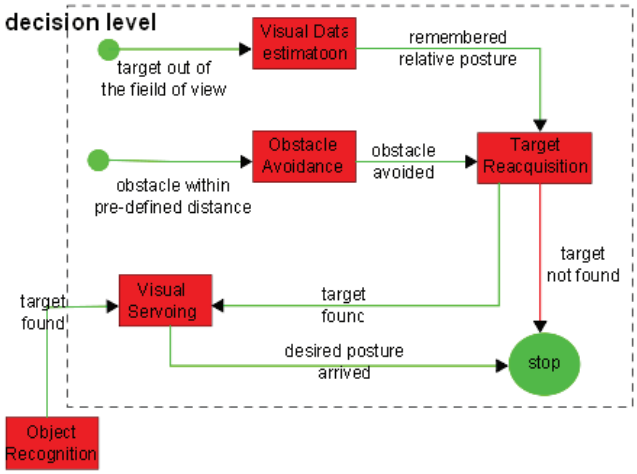


Figure 3.5: Behavior designation for positioning task.

avoidance controller and object re-acquisition controller. The visual servoing controller is given in Section 3.4. The collision avoidance controller is introduced in Chapter 2. Here we give the target re-acquisition controller.

The target re-acquisition controller is a heading controller, which can rotate the robot facing the center of the target in order to bring this end in the field of view. Let $\Delta\theta$ be the angular error. The heading controller is thus

$$\omega_h = -\lambda_h \Delta\theta, \tag{3.27}$$

where λ_h is a positive gain to be set.

The error $\Delta\theta$ can be computed using the estimated visual points in the normalized image plane. The visual data estimation will be introduced in the following section. The main objective of target re-acquisition is then to move the image projection of the mass center of the target to the y axis of the normalized image plane (see Figure 3.6). The mass center (x_{mc}, y_{mc}) of the estimated target can be computed as

$$\begin{cases} x_{mc} &= \frac{m_{10}}{m_{00}} \\ y_{mc} &= \frac{m_{01}}{m_{00}} \end{cases} \tag{3.28}$$

where m_{ji} is the spatial moments computed as $m_{ij} = \sum_{x,y} x^i y^j$. Since the focal length of the normalized image plane equals one, the angular error is defined as

$$\Delta\theta = \arctan(-x_{mc}). \tag{3.29}$$

where x_{mc} is illustrated in Figure 3.6.

The heading controller is terminated when the distance of the mass center to the y axis of the normalized image plane less than a given threshold $|-x_{mc}| < thr_{HC}$.

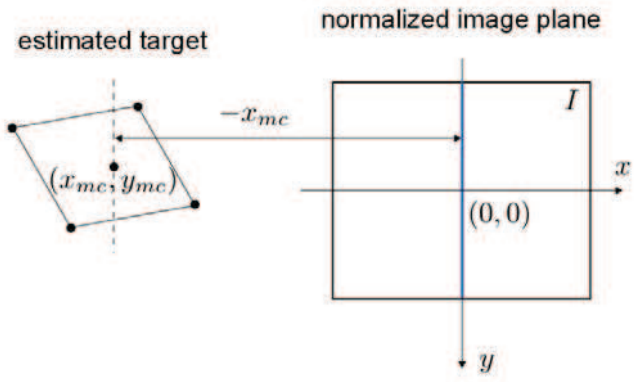


Figure 3.6: Illustration of target reacquisition error.

3.5.5 Visual Data Estimation in Case of Target Loss

3.5.5.1 Using a Known Target

We suppose that the target information is remembered before loss and the motion of the camera is estimated, the lost target can be recovered. A simple case is that the target is known and the motion of the camera can be obtain directly through the visual odometry. Using a known target, the pose of the camera with respect to the observed target can be computed from a set of measurements in one image with the known camera intrinsic parameters. This is the classical pose estimation problem in computer vision which can be solved using linear methods [36] or nonlinear methods [87]. The target pose with respect to the camera is remembered before loss. During the loss, the remembered target pose and the measured camera motion is used to recovery the current pose of the target with respect to the camera. Then the 3-D object is projected onto the current image plane.

3.5.5.2 Using an Unknown Planar Target through Homography Recovery

Our image-based visual servoing is based on homography tracking. If the target is out of the camera field of view, the homography tracking fails. To solve this problem, we

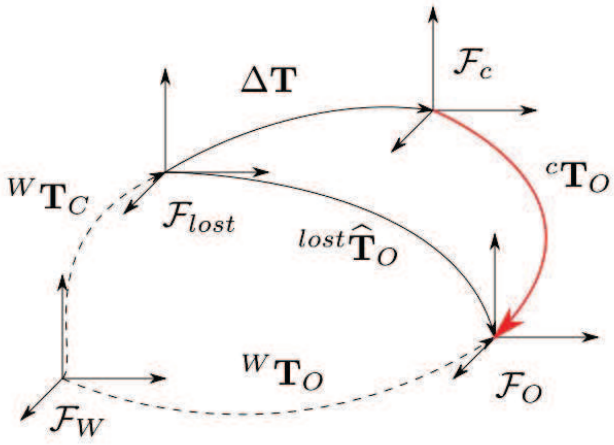


Figure 3.7: Lost target recovery using a known target.

propose a strategy taking the advantage of path integration and visual target reaction recognition.

A planar target observed in two images contents a homography relationship. If the target is visible in both images with at least four matches or two matches in case of a mobile robot with constrained motion, the homography can be calculated using only the corresponding image points. If the homography is known, the target in one image can be obtained from the other image through the homography transformation. In case of the target loss in the current image, if we know the target in the reference image and the homography transformation between these two images, the target can still be recovered by generating a virtual target in the current image. The recovered virtual target is then used to continue visual servoing control. The reference image can be remembered before the target loss. While during the invisibility of the target in the current image, we recover the homography. In equation (1.21), the transformation of the camera ($\mathbf{R}$, $\mathbf{t}$) and the structure of the planar target ($\mathbf{n}$, d) are needed. The strategy is illustrated in Figure 3.8. In the sequel, we will introduce the methods of computing these parameters.

Camera Motion Estimation

The camera motion can be estimated using stereo vision systems as described in Appendix B.6 or integrating odometric measurements. When mono vision systems is considered, the rotation of the camera can be recovered, and only the direction of the translation is obtained. Without loss of generality, we describe the estimated motion

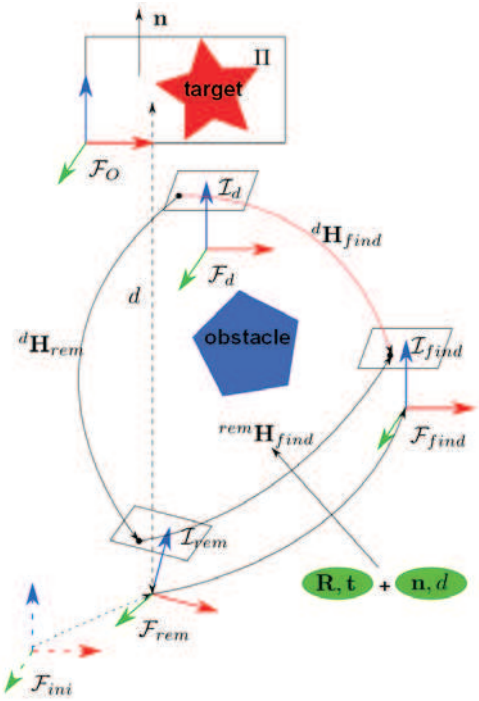


Figure 3.8: Homography tracking in case of invisibility

of camera as $(\hat{\mathbf{R}},\, \lambda\hat{\mathbf{t}})$, where $\lambda = 1$ if stereo vision systems or odometry are used.

Planar Structure Estimation from Visual Tracking

When the target is visible, the tracking provides the homography relationship $\mathbf{H}$ between the current camera frame and the previous one. The structure of the planar target can be calculated by decomposing the homography $\mathbf{H}$, using equation (1.21), into $(\mathbf{R_H}, \mathbf{t_H}, \mathbf{n})$. Note that the translation vector $\mathbf{t_H}$ is normalized by the distance to the target plane d as given in equation (1.22). The estimated motion of the camera can be described as $(\mathbf{R_H},\, d\,\mathbf{t_H})$. The structure of the target is $(\mathbf{n},\, d)$. The value d is waited to be assigned using the results from motion estimation. Considering motion estimation, we have $\lambda\,\hat{\mathbf{t}} = d\,\mathbf{t_H}$. Then

$$d = \lambda\,\hat{\mathbf{t}}\,/\,\mathbf{t_H}. \tag{3.30}$$

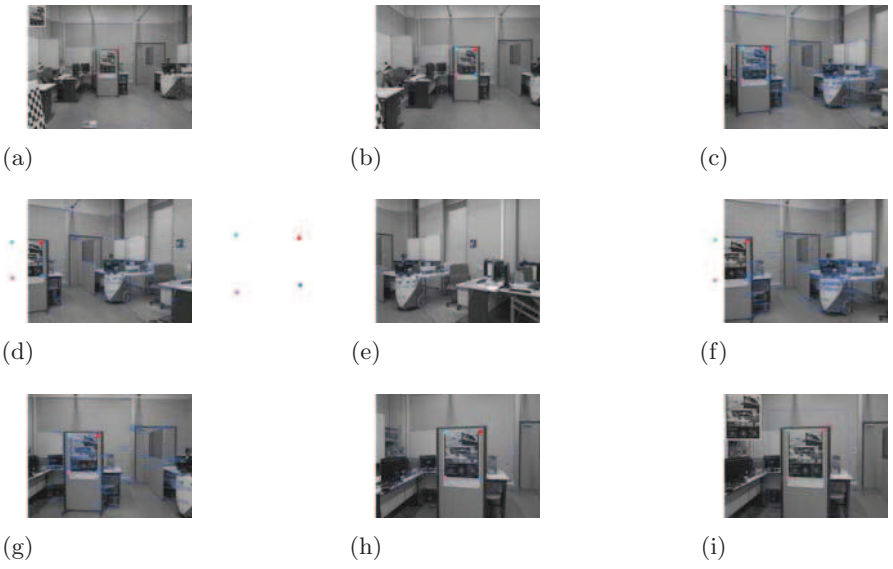
If λ is unknown, we assign an arbitrary value. d is assigned correspondingly. The structure $(\mathbf{n},\, d)$ is remembered before target loss.

Virtual Target Recovery

Using the remembered structure of the object ($\mathbf{n}_{rem}$, d_{rem}) and the estimated motion of camera ($\hat{\mathbf{R}}_{lost}$, $\lambda\hat{\mathbf{t}}_{lost}$) during the target lost, the homography can be recovered using equations (1.21), (1.22), and (3.30)

$$\begin{aligned}\mathbf{H}_{lost} &= \hat{\mathbf{R}}_{lost} + \frac{\lambda\hat{\mathbf{t}}_{lost}}{d_{rem}}\mathbf{n}_{rem}^{\top} \\ &= \hat{\mathbf{R}}_{lost} + \frac{\mathbf{H}\mathbf{t}_{rem}\hat{\mathbf{t}}_{lost}}{\hat{\mathbf{t}}_{rem}}\mathbf{n}_{rem}^{\top}\end{aligned} \tag{3.31}$$

We can note that the distance d and the scale factor λ are not necessary to recovery the homography $\mathbf{H}_{lost}$. All that is needed are the estimated motion of camera ($\hat{\mathbf{R}}_{lost}$, $\lambda\hat{\mathbf{t}}_{lost}$) during the target lost, the remembered normal of the planar target, the transformation from both homography decomposition [45] and the motion estimation in a duration before the target loss. The interesting conclusion is that even we use mono vision we can ignore the effect of scale factor and the distance indeed is offset. Figure 3.9 illustrates the results of lost target recovery. The four color points are the four corner points of the tracked object. Figures 3.9(b) to 3.9(h) are the tracking results using estimation. Figures 3.9(d) to 3.9(f) show that the proposed strategy can handle the lost target.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.9: Unknown target recovery by estimating homography.

3.5.5.3 Navigation Using Straight Line

Instead of retaining the visual servoing control using the estimated feature, another efficient strategy for navigation motion prediction is to follow the straight line which directs the target direction. As shown in Figure 3.10, a linear path can be defined using the retained direction. A basic control law $\omega_{virtual}$ (using [136]) for linear path following can thus be designed during collision avoidance task since the motion is estimated between the current and collision avoidance activation times, thanks to the odometric data. This control law aims to correct the lateral and the angular deviations (d_l, α_l) with respect to the linear path. In our case, this control law is virtual since it is never applied to control the mobile robot, and only used as a condition to decide when the collision avoidance task can be deactivated. Indeed, the condition is based on a comparison between the angular velocities of both controllers (virtual and collision avoidance). If $|\omega_{coll}| > |\omega_{virtual}|$ and $sign(\omega_{coll}) = sign(\omega_{virtual})$, then deactivate collision avoidance; otherwise, continue collision avoidance.

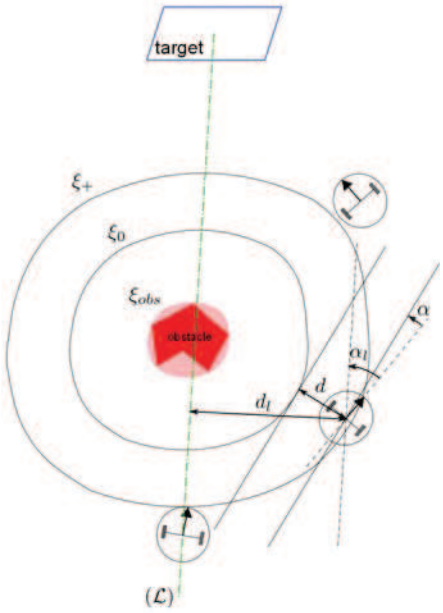


Figure 3.10: Linear path following during target loss

3.5.6 Experimental Results

We have implemented the above methods on our mobile robot platform named Lina (see Figure 3.11). Lina mobile robot is equipped with various exteroceptive and proprioceptive-

tive sensors such as odometry, laser range finder, two ACTi IP cameras of resolution of 640×480 where only one camera (left camera) is used in this experiment. All data processing is done on a deported PC. Controls and data sensing are respectively sent and received to the robot platform through a local network (wired or wireless). The mobile robot is driven towards a goal represented in the image space by a natural image (a poster of size 55.8×73.5 cm). The four corner points of the target image are selected as visual features to build the visual servoing control. We use box-like objects as obstacles. They are placed between the initial and desired positions of the mobile robot.

Figure 3.11: Mobile robot Lina

Figure 3.12 shows the navigation steps where the different positions are given in red circles and numbers (1 to 5). The trajectory of the mobile robot is drawn in blue color. Obstacles detected by the laser range finder are given in black color. We suppose that the target is visible by the camera at the starting position. First, the target is detected by matching the current image with the reference one (acquired at the desired position). Then, the template based tracking using the ESM tracking technique, is initialized (at position 1). The mobile robot starts to reach the goal through the visual servoing control scheme. During servoing, collision avoidance is considered in the mobile robot control when obstacles are detected in the path towards the goal (at position 2). During the obstacle avoidance (from position 2 to 3), the mobile robot position is recovered

using the odometric data, and the reference plane parameters (the normal vector of the reference plane and its distance with respect to the camera frame) are estimated. When the obstacle is totally avoided (at position 3), the mobile robot is oriented towards the target (position 4) and then the visual servoing is restarted to reach the goal (position 5).

Figure 3.13 shows images and laser data at different steps (1 to 5) as described in Figure 3.12. A video of the real experiment is available at http://aramis.iup.univ-evry.fr:8080/~hadj-abdelkader/Videos/Wenhao/video_MMAR2013.wmv.

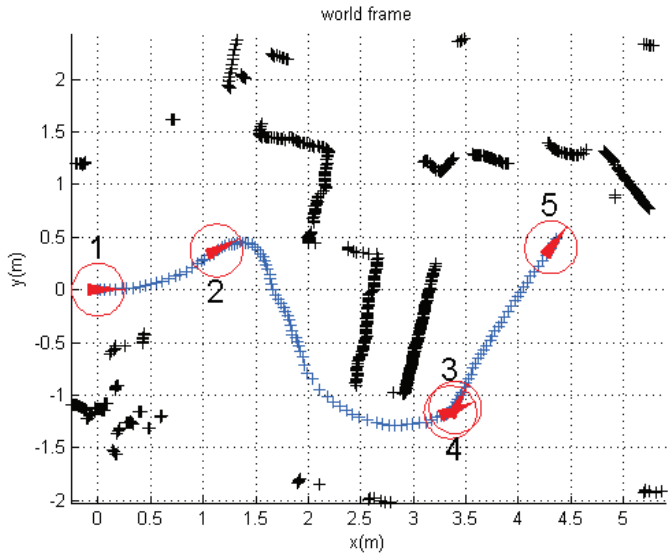


Figure 3.12: Vision-based navigation

In summary a solution for mobile robot navigation in indoor environments has been presented. Vision based navigation and laser range finder based obstacle avoidance have been used to reach a desired position of the mobile robot safely. Visual target loss while obstacle avoidance, has been solved by exploiting the visual odometry based motion estimation and the homography relationship.

3.6 Spherical Image-Based Visual Servoing (SIBVS)

3.6.1 Why Spherical Projection

The planar perspective projection we used is the standard approach to define perspective cameras, which considers planar imaging surfaces. However, the perspective

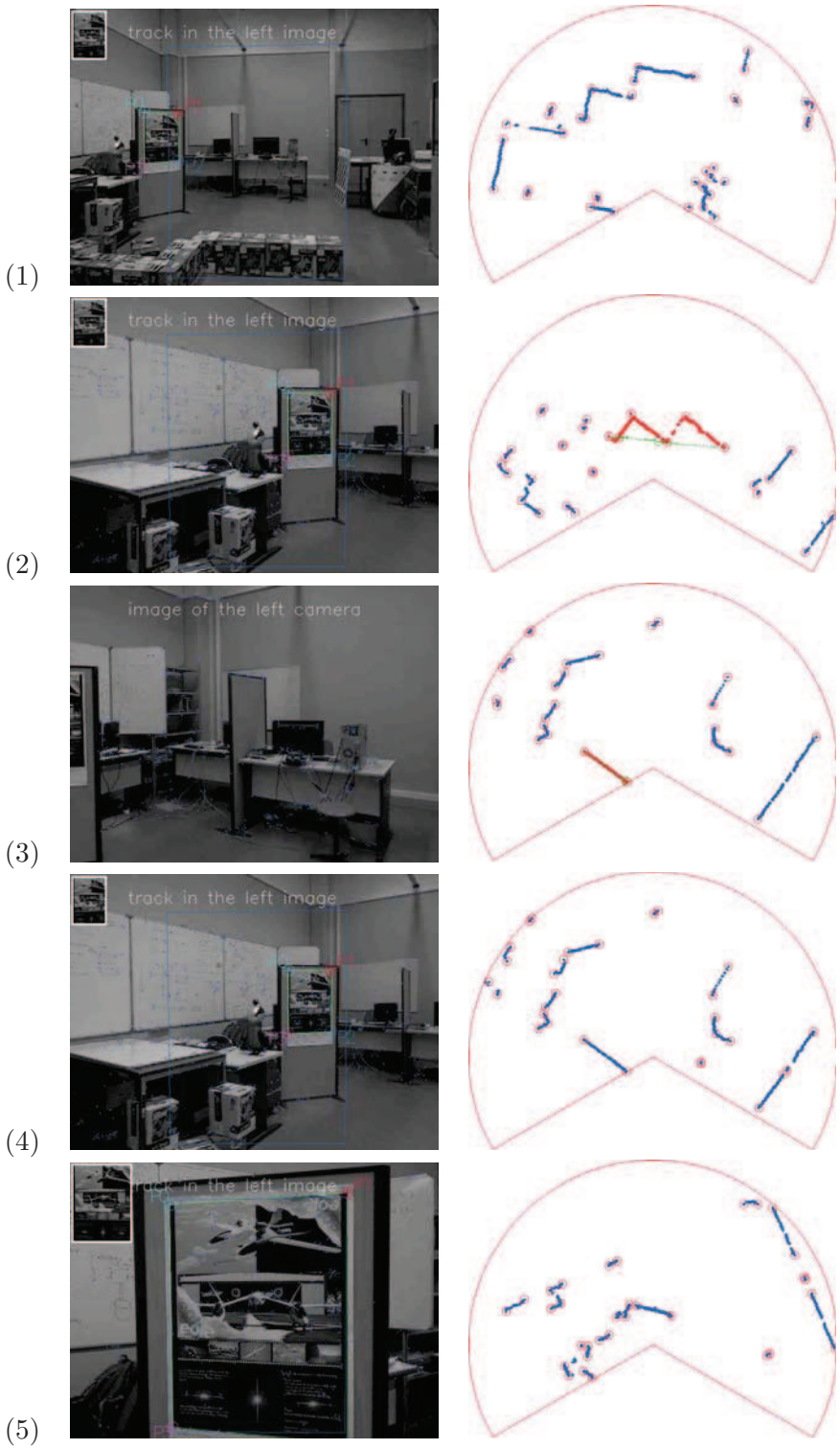
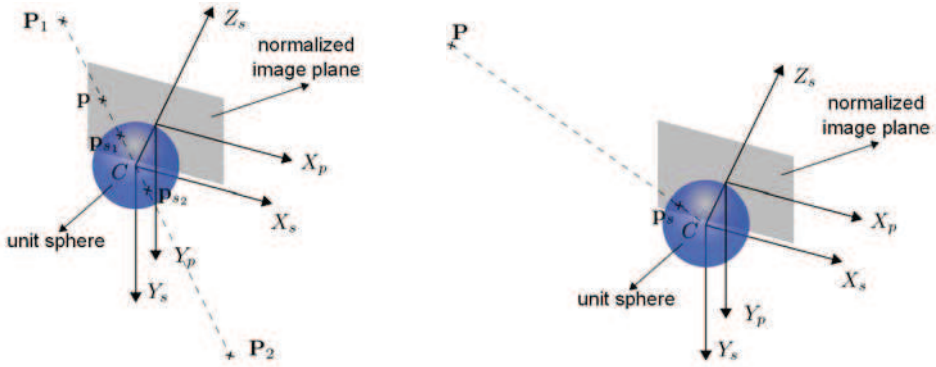


Figure 3.13: Sensor data

imaging model has mainly two disadvantages as illustrated in Figure 3.14. First, this model can not be adapted to the situations in which the field of view is greater than 180°, and then the visual object comes to the backward of the image plane. As illustrated in Figure 3.14(a), the 3-D points $\mathbf{P}_1$ and $\mathbf{P}_2$ are projected onto the same image point $\mathbf{p}$. The projection will be ambiguous for the IBVS. Second, an infinity control input will be generated for IBVS when the projection of the visual feature is to be located at infinity on the image plane. As illustrated in Figure 3.14(b), a 3-D point $\mathbf{P}$ projects at infinity when its depth is near to zero. In order to solve these problems, we adopt an alternative projection model, named unified projection model. This is a general model as motioned in [48]. The unified spherical projection model can distinguish the projections of the 3-D points $\mathbf{P}_1$ and $\mathbf{P}_2$ as $\mathbf{p}_{s_1}$ and $\mathbf{p}_{s_2}$. It can also handle the infinity projection on the unit sphere. In the following, we will discuss our solution using spherical image model.



(a) Ambiguity in perspective projection. (b) Infinity in perspective projection.

Figure 3.14: The illustration of the problems for the perspective imaging model.

3.6.2 Spherical Projection Formulation

Spherical model Spherical projection model considers sphere imaging surfaces. It is well adapted to wide angle views, even a field of view greater than 180°.

Let $\mathbf{P} = (X, Y, Z)$ the coordinates of a 3-D point, represented in the frame attached to the spherical image. $\mathbf{P}$ is projected onto the surface of the unit sphere at point

$\mathbf{p}_s = (x, y, z)$ by a ray passing through the center of the sphere

$$\begin{cases} x &= \frac{X}{R} \\ y &= \frac{Y}{R} \\ z &= \frac{Z}{R} \end{cases} . \tag{3.32}$$

where $R = \sqrt{X^2 + Y^2 + Z^2}$ is the distance from the origin of the sphere to the 3-D point $\mathbf{P}$.

An alternative coordinate system is the spherical coordinate (r, θ, φ). As shown in Figure 3.15, $r = 1$ is the radial distance of $\mathbf{p}_s$ from a the origin of the coordinates. $\theta \in [0, \pi]$ is the polar angle, and $\varphi \in [-\pi, \pi]$ is the azimuthal angle.

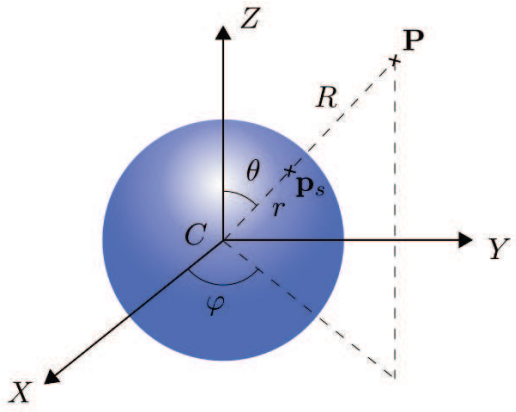


Figure 3.15: Spherical coordinate definition.

The relationships between these two coordinate systems are

$$\begin{cases} \rho &= \sqrt{x^2 + y^2}, & \\ \theta &= \arctan \frac{\rho}{z}, & \theta \in [0, \pi] \\ \varphi &= \arctan \frac{y}{x}, & \varphi \in [-\pi, \pi] \end{cases} . \tag{3.33}$$

Perspective and spherical projection transformation Spherical projection model projects a 3-D point $\mathbf{P}(X, Y, Z)$ onto the unit sphere instead of projecting onto the plane as perspective projection model. Let the point $\mathbf{p}(x, y, 1)$ be the perspective projection of the 3-D point, and $\mathbf{p}_s(x_s, y_s, z_s)$ be its spherical projection. The spherical coordinates

$\mathbf{p}_s$ can be computed as

$$\begin{cases} h &= \sqrt{x^2+y^2+1} \\ x_s &= \frac{x}{h} \\ y_s &= \frac{y}{h} \\ z_s &= \frac{1}{h} \end{cases} . \tag{3.34}$$

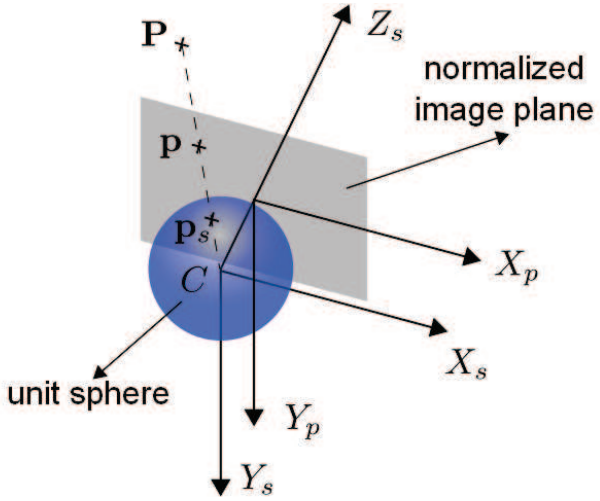


Figure 3.16: The convention spherical image coordinate.

In order to use spherical image based visual servoing, here we introduce spherical image interaction matrix of point as given in [32]

$$\mathbf{L_{sa}} = \begin{bmatrix} -\frac{\cos\varphi\cos\theta}{R} & -\frac{\sin\varphi\cos\theta}{R} & \frac{\sin\theta}{R} & \sin\varphi & -\cos\varphi & 0 \\ \frac{\sin\varphi}{R\sin\theta} & -\frac{\cos\varphi}{R\sin\theta} & 0 & \frac{\cos\varphi\cos\theta}{\sin\theta} & \frac{\sin\varphi\cos\theta}{\sin\theta} & -1 \end{bmatrix} \tag{3.35}$$

The homography for spherical imaging has the same form as the planar one. The relationship between two spherical points is given as

$$\mathbf{p}_s = \lambda \mathbf{H}_s \mathbf{p}_s^*. \tag{3.36}$$

3.6.3 Adjustment of SIBVS

In order to use SIBVS for mobile robots, we test the properties of SIBVS with simulational results in this section. In the simulations, the visual features are four points and the desired frame are kept to a fixed frame. Figure 3.17 illustrate the four points and the desired frame in the world frame. The four visual points are located in $(-1, 0, -1)$, $(-1, 0, 1)$, $(1, 0, 1)$ and $(1, 0, -1)$. The initial camera frame is at $(0, 2.5, 0)$ with orien-

tation $(\pi/2, 0, \pi)$ in roll-pitch-yaw angles.

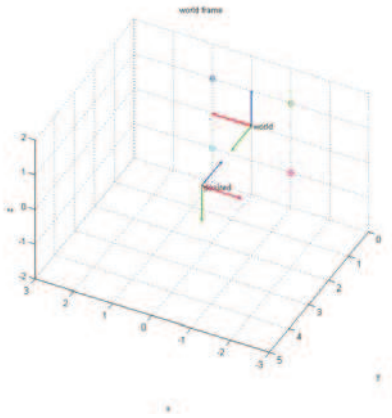


Figure 3.17: Configuration of visual point and desired frame.

3.6.3.1 Coordinate Selection

We present the simulation results of SIBVS for the cases: pure z-axis translation, pure z-axis rotation, z-axis rotation and translation, x-axis translation, pure x-axis rotation, and x-axis rotation and translation (see Appendix C). From the simulation results, we note that the camera performs unexpected motion for the cases of x-axis translation, pure x-axis rotation, and x-axis rotation and translation while expected motion for the cases pure z-axis translation, pure z-axis rotation, z-axis rotation and translation. This is due to the decoupling of the third and the sixth columns in the interaction matrix (see equation (3.35)). Considering the robot-camera configuration and the motion constraints, translation in the xy-plane and rotation in the z-axis direction, different with the conventional spherical image coordinate system (see Figure 3.16), we change the coordinate system as illustrated in Figure 3.18. We assign the x-axis of the spherical model to the camera optical axis. The positive orientation of the y-axis of the camera frame is assign to the left side and positive orientation of the z-axis of the camera frame is assign to the upside.

To illustrate the difference between the convention coordinate system and the changed coordinate, here we give simulational results. The initial frame of the camera is located at $(2, 2.5, 0)$ with orientation $(0, 0, \pi)$ in roll-pitch-yaw angles. The simulation result with the conventional coordinate system is given in Figure 3.19. It shows that the camera does not generate a nice behavior, especially a big shake at the beginning. The trajectories of the visual features are not straight lines.

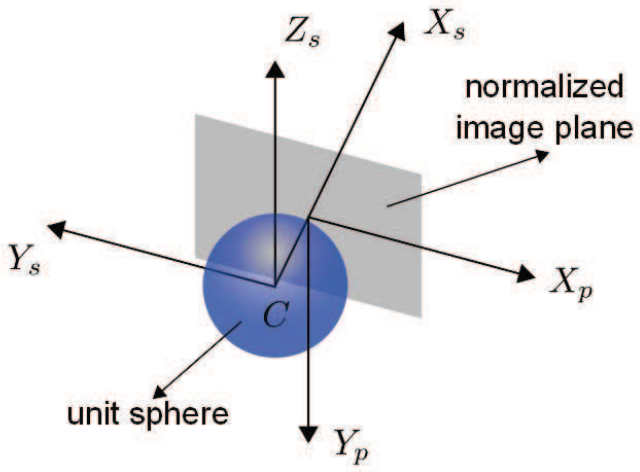
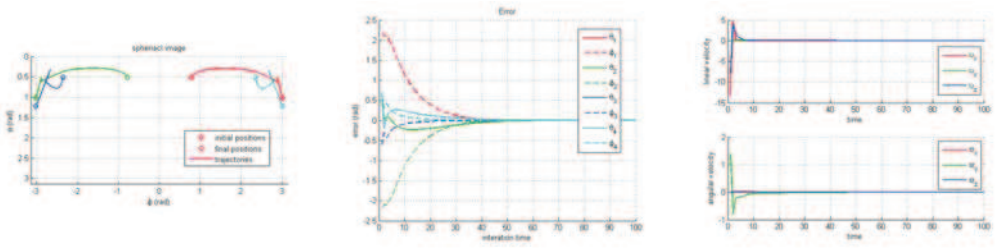


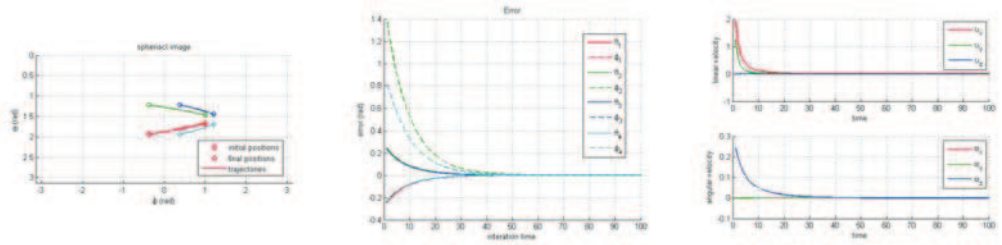
Figure 3.18: Changed coordinate system of the spherical image.



(a) Camera view. (b) Error. (c) Camera velocities.

Figure 3.19: The simulation results of the conventional spherical image coordinate system.

The simulation result with changed coordinate system is given in Figure 3.20. Different with the conventional coordinate system, the velocities are more smooth and with lower values. The camera has done a desired motion with rotation about z-axis and translation in the x- and y- axes directions. The trajectories of the visual features are approximately straight lines and the error approximately performs an exponential decoupled decrease.



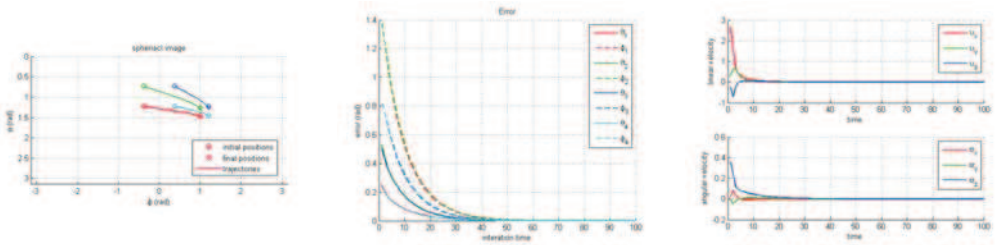
(a) Camera view. (b) Error. (c) Camera velocities.

Figure 3.20: The simulation results of the changed spherical image coordinate system.

3.6.3.2 Symmetric versus Nonsymmetric of Visual Point Position

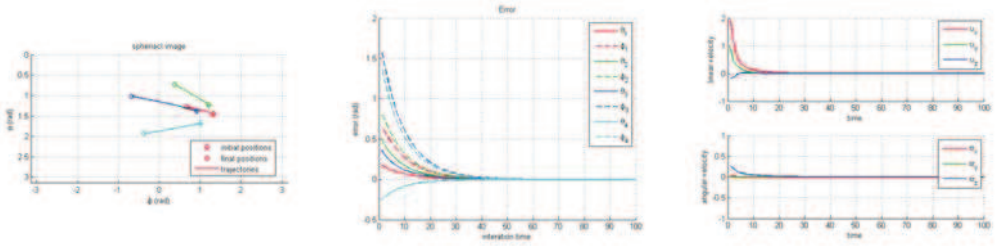
In all the simulation we use the four visual points which are symmetric position around coordinate axes. In the real experiments, it is not always sure in that case. We test the difference between using the visual points in symmetric and nonsymmetric position. The simulation of Figure 3.20 changes the position of the four visual points to be nonsymmetric. Two conditions have been tested. One simulation only moves the four visual point along z-axis to $(-1, 0, 1)$, $(-1, 0, 3)$, $(1, 0, 3)$ and $(1, 0, 1)$. The result is given in Figure 3.21. It is shown that the camera has done an unwanted translational motion in the z-axis direction and unwanted rotational motion about x- and y-axes directions. The other set the four points in arbitrary positions such as $(2, 0, 1)$, $(1, 0, 3)$, $(-2, 0, 2)$ and $(-1, 0, -1)$. The result is given in Figure 3.22. Compared Figure 3.21 and 3.22 with 3.20, the nonsymmetric of visual point position leads to unexpected camera motion.

In order to handle the unexpected motion in Figure 3.21 and Figure 3.22, instead of using the symmetric positions of the visual points, we can also constraint the motion in the xy plane, and rotate along z axis. The velocity $\mathbf{v}$ has the form $\mathbf{v} = (\upsilon_x, \upsilon_y, \omega_z)$. For the interaction matrix, we keep the items related to translation along xy axis and



(a) Camera view. (b) Error. (c) Camera velocities.

Figure 3.21: A general initial configuration using modified coordinates and nonsymmetric position of visual features with drift.



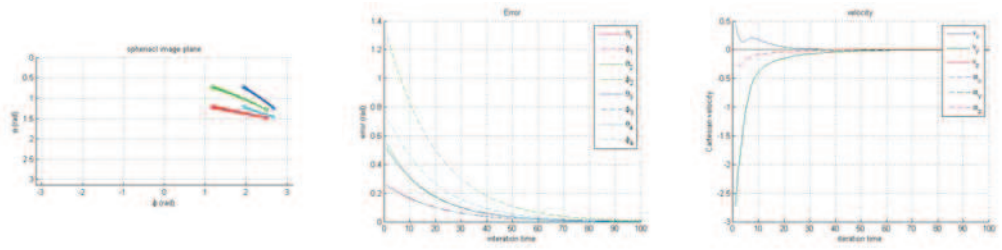
(a) Camera view. (b) Error. (c) Camera velocities.

Figure 3.22: A general initial configuration using modified coordinates and arbitrary nonsymmetric position of visual features.

rotation along z.

$$\mathbf{L_{sa}} = \begin{bmatrix} -\frac{\cos\varphi\cos\theta}{R} & -\frac{\sin\varphi\cos\theta}{R} & 0 \\ \frac{\sin\varphi}{R\sin\theta} & -\frac{\cos\varphi}{R\sin\theta} & -1 \end{bmatrix} \tag{3.37}$$

The simulation result is shown in Figure 3.23. The result is



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure 3.23: Redefine camera coordinates using spherical IBVS with respect to non-symmetric features. The motion is constrained in xy plane and rotation along z axis.

In this section, the basic performances of SIBVS are discussed. We found that the selection of camera coordinates and the position of the visual feature can effect the result. In order to have a good performance, we change the selection of the camera coordinates. We also found that using the visual feature in symmetric position performs difference with using the nonsymmetric one. We see that using the symmetric visual features improves the behavior of the camera motion. Hence, in the follows, we use the symmetric visual features for simulations. If in the real case the visual features are nonsymmetric, it is possible to transform them into the symmetric visual features using the homography transformation.

3.6.4 Spherical Visual Servoing for Mobile Robot

In the last section we have evaluated the basic performance for SIBVS, and discussed the selection of coordinate system. In this section, we will adapt the spherical visual servoing to the mobile robot with nonholonomic constraints and the target loss.

3.6.4.1 System Modeling

Using the selected camera coordinate for spherical model, the robot-camera system is presented in Figure 3.24. For such a configuration, the jacobin matrix **J** in equation

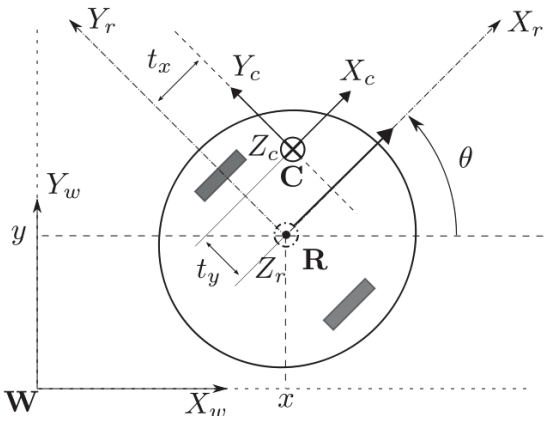


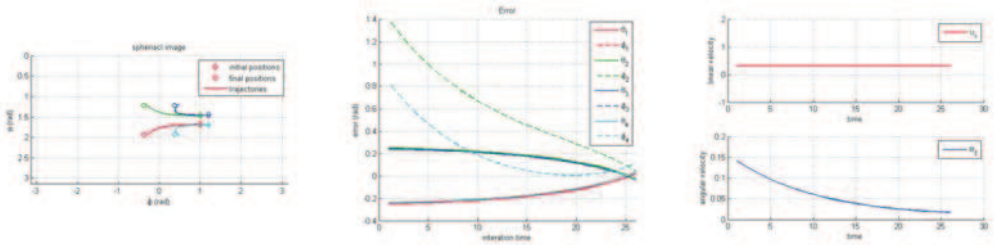
Figure 3.24: Robot-camera configuration using the selected spherical image coordinate.

(3.1) is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_v & \mathbf{J}_\omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -t_y & t_x & 0 & 0 & 0 & 1 \end{bmatrix}^\top . \tag{3.38}$$

3.6.4.2 Control Law Using Constant Linear Velocity

We use the same control law as given in equation (3.12) while $\mathbf{L_s}$ is replaced by $\mathbf{L}_{sa}$. We set $\upsilon = 0.3$ and $\lambda = 0.1$. We stop the camera when the norms of the error vector begins to increase, since we use the constant linear velocity. The simulation result is shown in Figure 3.25.



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure 3.25: SIBVS for mobile robot using constant linear velocity.

3.6.4.3 Control Law Scaling the Error with Different Values

Instead using constant linear velocity, we can also use different scale for the error function. For one visual feature, we define

$$\dot{\mathbf{e}} = \begin{bmatrix} \lambda_\theta & 0 \\ 0 & \lambda_\varphi \end{bmatrix} \begin{bmatrix} e_\theta \\ e_\varphi \end{bmatrix}, \tag{3.39}$$

If we note the parameter matrix as Λ, which is a diagonal matrix of λ_θ and λ_φ of all features, we can write

$$\dot{\mathbf{e}} = -\Lambda \mathbf{e}. \tag{3.40}$$

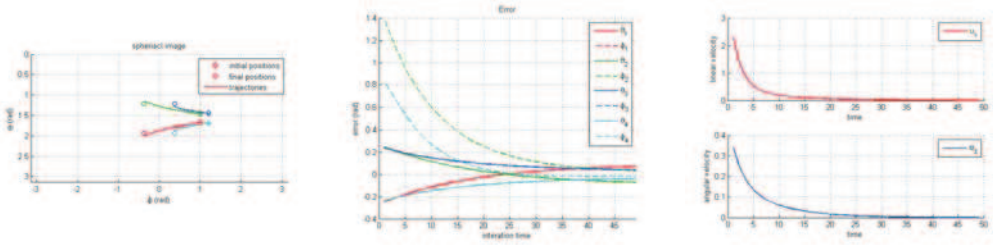
From the relationship

$$\dot{\mathbf{e}} = \mathbf{L}^C \mathbf{J}_R \mathbf{v}_R, \tag{3.41}$$

the control law is

$$\mathbf{v}_R = -(\mathbf{L}^C \mathbf{J}_R)^+ \Lambda \mathbf{e}. \tag{3.42}$$

Figure 3.26 illustrates a simulation result of spherical IBVS forming the control law with different scales. In the simulation $(\lambda_\theta, \lambda_\varphi) = (0.01, 0.1)$.



(a) Camera view. (b) Error vector. (c) Camera velocities.

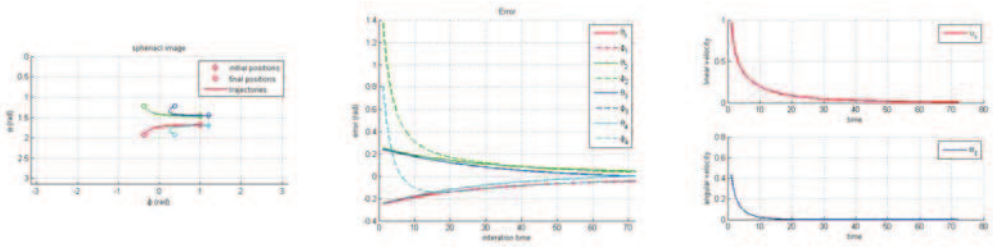
Figure 3.26: SIBVS for mobile robot using control law scaling the error with different values.

3.6.4.4 Control Law Scaling the Velocity with Different Values

Rather than scaling the control law, we can also scale the velocities. To this aim, the control law can be written

$$\mathbf{v}_R = -\begin{bmatrix} \lambda_\upsilon & 0 \\ 0 & \lambda_\omega \end{bmatrix} (\mathbf{L}^C \mathbf{J}_R)^+ \mathbf{e}. \tag{3.43}$$

Figure 3.27 has illustrated a simulation result of spherical IBVS scaling the velocity with different values. In the simulation $(\lambda_\upsilon, \lambda_\omega) = (0.03, 0.1)$.



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure 3.27: SIBVS for mobile robot using control law scaling the velocity with different values.

3.6.4.5 Switching Schemes

In order to optimize the performance, we design a switching scheme. It divides the positioning task into several steps [98]. Each step uses a controller. The scheme is to design the rule for multi steps and the switch condition of continuous steps. Our designation of the behavior is illustrated in Figure 3.28.

Control law design To simplify our design,we first generate four symmetric virtual features. Those virtual features can be warped to the current image using the same homography transformation. The proposed control strategy is performed in four steps as follows:

- Step one: A pure rotation in order to face to the x axis of the desired robot frame. It means $\mathbf{n}_c\mathbf{n}_d = 0$.
- Step two: A Straight line motion in order to go to the straight backward of the reference frame.
- Step three: A pure rotation in order to align to the direction of the reference frame.
- Step four: A straight line motion in order to go the desired frame.

In step one, we select the feature as $\mathbf{u}\theta$. We have the relationship

$$\mathbf{u}\dot{\theta} \simeq \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \mathbf{v}. \tag{3.44}$$

We note the $\mathbf{u}\theta$ along z axis as $\mathbf{u}_z\theta$. In this step, the desired heading of the robot is perpendicular face to the x axis of the desired frame. We have the desired $\mathbf{u}_z\theta$ as

$$\theta^* = \begin{cases} \frac{\pi}{2}, & \text{if} \quad \tan(\mathbf{t}_x, \mathbf{t}_y) < 0 \\ -\frac{\pi}{2}, & \text{if} \quad \tan(\mathbf{t}_x, \mathbf{t}_y) > 0 \end{cases} \tag{3.45}$$

The pure rotation control law is thus given by

$$\omega = -\lambda(\theta - \theta^*)\mathbf{u}_z. \tag{3.46}$$

It is not always needed that the robot runs from step one, for example, the robot is near to the x axis of the desired frame. So we can give some conditions. If the initial position of the robot in a defined region represented by a angle σ, the robot executes the step one. The region is defined by ${}^d\mathbf{t}_c$ in the condition $|\tan(\mathbf{t}_x, \mathbf{t}_y)| < \pi - \sigma$.

In the second step, the pure translation is accomplished by aligning the center of the gravity $\mathbf{p}_g$ of the feature points to the y axis of the current frame. It is equally to bring the feature φ_g to φ_g^* as follows:

$$\varphi_g^* = \begin{cases} -\frac{\pi}{2}, & \text{if} \quad \tan(\mathbf{t}_x, \mathbf{t}_y) < 0 \\ \frac{\pi}{2}, & \text{if} \quad \tan(\mathbf{t}_x, \mathbf{t}_y) > 0 \end{cases} \tag{3.47}$$

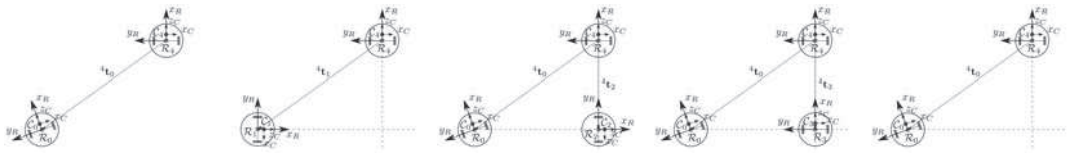
The pure translation control is

$$\upsilon = -\lambda {\mathbf{L}_\upsilon}^+ \mathbf{e}_t \tag{3.48}$$

with

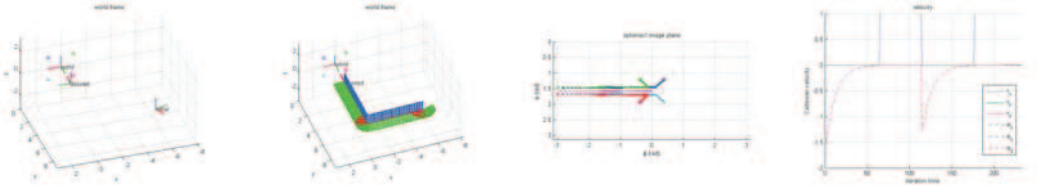
$$\mathbf{L}_\upsilon = \begin{bmatrix} -\frac{\cos\varphi\cos\theta}{R} & -\frac{\sin\varphi\cos\theta}{R} & \frac{\sin\theta}{R} \\ \frac{\sin\varphi}{R\sin\theta} & -\frac{\cos\varphi}{R\sin\theta} & 0 \end{bmatrix} \tag{3.49}$$

The third step is similar as the first step. It is also a pure rotation control. The difference is that the desired feature is $\theta^* = 0$. The forth step can use the control law as given in equation (3.12). The simulation result is shown in Figure 3.29.



(a) Initial configuration. (b) Step one. (c) Step two. (d) Step three. (e) Step four.

Figure 3.28: Illustration of the switching scheme in four steps.



(a) Camera view. (b) Camera view. (c) Error vector. (d) Camera velocities.

Figure 3.29: Switch scheme using spherical IBVS.

3.6.5 Discussion

There are mainly two reasons to use sphere projection model. First, sphere projection model has no undesired back forward projection problem. Second, there is no infinity projection problem on the image plane, which leads to an infinity control input for visual servoing.

Spherical projection model has many advantages for visual servoing. Firstly, it enlarges the field of view. Hence, it eliminates the need to explicitly keep features in the field of view, which is a problem with PBVS, some hybrid schemes, and IBVS combining with other tasks such as collision avoidance. Secondly, computing the control law in terms of spherical coordinates can avoid the infinity control input. In the case of planar projection model, when the feature points are near the image plane, the imaging point may approach to infinity. Therefore, the control input will be also closed to infinity, which is unattainable for actual systems. Furthermore, it is a unified projection model. The information for any other projection models can be projected onto spherical projection model.

In this section,we have designed different schemes of spherical IBVS. Using constant linear velocity is hard to have a satisfaction result. The final configuration of the robot is always differen with the desired one because of the nonholonomic constrain. However, the scheme using different scales can enhance the flexibility of the system. They still encounter the problem of parameters selecting. Rather than using a single control law, the switching scheme can handle the nonholonomic constrain and considering the desired condition as the same time by using the appropriate switching conditions. But it is not an uniform control form. Moreover, in each of the designed scheme, we use the property of symmetric visual features, which can be obtained by any given visual features and a homography transformation relation.

3.7 Conclusions

This chapter presents a solution to target loss and collision free during visual servoing based navigation in an indoor environment. The proposed strategy remembers the information as little as possible with a path integration to recover the lost target. It is shown that interaction between different navigation strategies, especially active sensing, can reduce navigation error, and may lead to new capacities.

In this section, we suppose that the object is in the field of view before the mobile robot start navigation task. When the assumption is not met in the real cases, some strategies can be used such as random moving and searching. The basic idea of the strategy is not limited to planar visual target. If a 3-D target is choose, essential matrix could be used instead of homography matrix. The controller to re-acquire the lost target can be designed based on the error between a point to the related epipolar line.

Spherical image is more adaptive to the navigation task, which can remove the ambiguity between the front and the back of the camera in the conventional perspective camera model.

In the navigation system, the object recognition is not real-time, and hence the robot has to stop before re-initializing the visual tracking. This makes the system loss the continuity in the motion. In order to solve this inefficiency, several alternative schemes can be considered. For example, we can increase the number of ordered key images in the database used for appearance-based navigation. Then the robust but inefficient recognition algorithm can be replaced by more efficient algorithms. In addition, we can consider to use graphics processing unit (GPU) or distributed Computing to have some performance gain.

Conclusion and future research

Summary of the Thesis

The final goal of this thesis is to design an autonomous mobile robot which can safely navigate with the visual landmarks within the indoor environment in present of obstacles. In the work, the robot can detect the planar visual landmark from a large distance based on local feature matching, and then the landmark is tracked using a homography-based method. Visual servoing is adopted to control the robot toward the landmark. During the visual control, we consider the safety constraints imposed by the environments and the field of view constraints imposed by the camera system, as well as nonholonomic constraints imposed by the robot kinematics.

In the context of object recognition, the use of local features provided for a large progress in terms of the robustness, efficiency and quality of results. We experimentally evaluated the different standard algorithms usually used in this context. Robustness and real-time performance are hard to be contented at same time. In the landmark recognition and tracking framework, the landmark can be detected from a large distance, then the tracking is initialized. It is important for the robot to detect the visual landmark robustly and accurately. So we adopted SIFT algorithm, which is robust and has good performance for object detection.

For collision avoidance, we have presented an efficient obstacle detection and representation method using 2-D laser range finder. We use line fitting techniques to extract the polygonal chain to model the obstacle boundary. In the line fitting process, we analyze the convex and the concave structure of the polygonal chain. We extract the convex polygonal chain with B-spline interpolation to generate a smooth curve for the obstacle boundary. We then formulated reactive obstacle avoidance as path following with respect to the extracted convex curve. The experimental result shows that the robot can move smoothly without being trapped in the concave part of obstacles.

We have discussed vision-based navigation considering three constraints, including

nonholonomic constraints, collision free constraints and field of view constraints. We principally described our strategies to handle target loss problem during vision-based navigation. Visual odometry is used to estimate the lost target when the robot avoids obstacles. However, the perspective projection model encounters difficulty when the target is in the back forward of the image plane. We extend the proposed homography based visual servoing to the spherical image. The simulation results show that the feasibility of using spherical image.

We mainly proposed a strategy for vision-based mobile robot navigation which can handle the target loss due to the field of view constraints and obstacle avoidance tasks. In our strategy, the homography relationship is estimated through the visual odometry measurement and the remembered planar target structure. We originally introduced spherical imaging model into our framework for the target loss problem, which can solve the drawback of the perspective projection model. In addition, we proposed an efficient obstacle detection and representation methods through convex polygonal chain and B-spline. The representation can provide a smooth modeling of the obstacle boundary. Furthermore, we proposed a vision framework combining the feature-based object detection and template-based target tracking, which can track a planar target from a long distance.

Perspective of the Research Work

For the short-term of future work, we can continue our works as follows. Object recognition is the basic in the appearance-based navigation system. In our experiments, we aim to sparse landmarks. There may be a large distance between two landmarks. Therefore, the object recognition is not real-time, which affects the performance of the whole system. In order to improve the performances, we will shorten the distance between the landmarks. Moreover, in this thesis, we focus on challenges in each step of appearance-based navigation. We will extend our work for mobile robot navigation between a sequence of landmarks in order to navigation the mobile robot moving from one room to another one. We will consider to introduce the bag-of-the word techniques for landmark detection in the appearance-based navigation to replace the detection and tracking frame work used in our system.

In the work of reactive collision avoidance, we use convex polygonal curve and B-spline to represent the boundary of the obstacle. This representation is combined with path following techniques for obstacle avoidance. Using the approach, the mobile

robot performs smooth motion efficiently in the sense of computational effort. The representation can be also extended to be combined with other obstacle avoidance approaches. However, the idea to find a model of the obstacle boundary is hard to be adapted to clustered environments, since it is impossible to find a model for any scene. We will consider more about model free based reactive collision avoidance approaches, such as VFH or dynamic window approach for more complicated environments.

In the simulation and experiment of vision-based navigation, we thought that two works can be continued. First, the trajectories of visual features have special patterns. It is promising to plan the path for the visual features depending on the motion behaviors. To solve the positioning task in the context of visual servoing control, the fuzzy Logic can be considered in the context of car backing problems.

Over the long term, we aim at designing a complete vision based mobile robot navigation system. However, our system is not a complete vision-based navigation system, since laser range finder is used for reactive collision avoidance. We will consider using vision to solve obstacle detection problem so that all the functions of autonomous navigation is based on vision. The motion estimation in our system uses stereo vision. The estimation results are affected by the accuracy of calibration. And the stereo camera system is needed to be calibrated regularly, since the two cameras are not tightly fixed. It is interesting to use mono vision instead of stereo vision, since mono vision is more compact and more common. However, for mono vision, it is a fundamental problem that the camera moves on pure rotation.

In this thesis, we have discussed the target loss when the robot executes collision avoidance task. However, the visibility problem can also occur due to occlusion. We will take account visual occlusion detection in the future work.

In addition, appearance based navigation is limited to the given visual path. Even thought we introduce visual odometry to estimate the deviation of the robot from the visual path, the robot also depends on the given visual path to a large extent. While humans have the reasoning ability based on the memorized knowledge when facing a new scene. Humans can build the relationship between the known appearance and the new appearance. Furthermore, humans perform more sematic-based and behavior-based navigation in the daily life. Humans can get the semantic idea of where they are and relate to specific behavior. These abilities can also strengthen the navigation ability for a mobile robot. Therefore, in the future, we aim to strengthen the perception module. The robot can identify the semantic meaning of the object in the environment to do some high level tasks as humans. The robot has the ability to build the link of known appearance and new appearance. In order to accomplish it, we will think

about machine learning methods to reinforce the ability based on previous navigation experience.

We are furthermore keen to give them more intelligence using vision to autonomously navigation in challenging environments. This requires new methods for perception representation and decision.

Appendix A

B-spline Curve

A.1 Cubic B-spline interpolation

Problem statement Suppose that there are n data points $\mathcal{D} = \{D_0 \dots D_{n-1}\}$ and the objective is to fit them with a B-spline curve $S(t)$ of degree p, where $p <= n$. The interpolating spline passes all the data points $\mathcal{D}$. More precisely, the values of the interpolating spline at the give n data points equal to the data points themselves, $S(D_i) = D_i$. Furthermore, the generated curve satisfies conditions such as continuous and end point conditions.

Polynomials vs. Splines There are many different methods to interpolate smooth curves. Among these method, polynomials and splines are widely used. Polynomials are infinitely differentiable. They can approximate smooth curves when a smooth function is to be approximated locally. However, if a function is to be approximated on a larger interval and the approximated curve is more complicated, the degree of the approximating polynomial may have to be chosen unacceptably large. Moreover, the polynomial interpolation may exhibit oscillatory artifacts, especially at the end points, which is known as Runge's phenomenon. The alternative is to use piecewise polynomials, namely splines, to interpolate curves. Spline subdivide the interval of approximation into sufficiently small intervals, so that, on each such interval, a polynomial of relatively low degree can provide a good approximation while avoiding instability due to Runge's phenomenon, as shown in figure A.1. Continuity conditions are taken into account so as to make a smooth approximating spline. Moreover, splines are popular

curves, especially in computer graphic, because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design. There are generally two problems relative to use B-spline: fitting and interpolation. In this section, we focus on the application of interpolation.

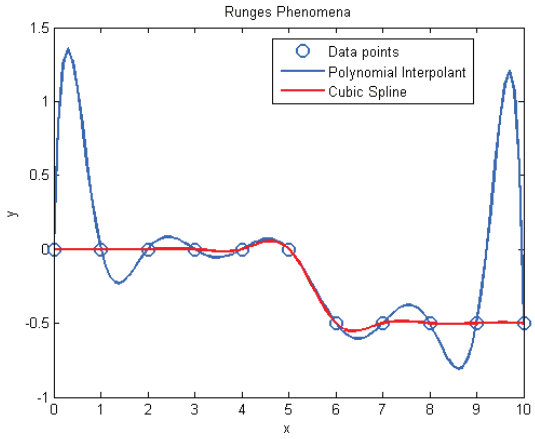


Fig. A.1: Runge's phenomenon

Bezier curve For the sake of simplicity and convenience, we use Bezier curves as the segments of B-spline. A Bezier curve is defined by a set of control points $P_0 \dots P_n$. More formally:

$$\mathbf{B}(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i, \ t \in [0,1] \tag{A.1}$$

where the polynomials

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \ i = 0, \dots, n \tag{A.2}$$

are known as Bernstein basis polynomials of degree n. Bezier curves can be accomplished using, for example De Casteljau's algorithm. If $n = 3$, a cubic Bzier curve is expressed as:

$$\mathbf{B}(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \ t \in [0,1] \tag{A.3}$$

The equation A.3 can be expanded the terms in t:

$$\mathbf{B}(t) = (-P_0+3P_1-3P_2+P_3)t^3+(3P_0-6P_1+3P_2)t^2+(-3P_0+P_1)t+P_0,\ t \in [0,1] \quad \text{(A.4)}$$

Equation A.4 is written in matrix form as:

$$\mathbf{B}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad \text{(A.5)}$$

The first derivative is

$$\mathbf{B}'(t) = 3(-P_0+3P_1-3P_2+P_3)t^2+2(3P_0-6P_1+3P_2)t+(-3P_0+P_1),\ t \in [0,1] \quad \text{(A.6)}$$

The second derivative is

$$\mathbf{B}''(t) = 6(-P_0+3P_1-3P_2+P_3)t+2(3P_0-6P_1+3P_2),\ t \in [0,1] \quad \text{(A.7)}$$

Then we obtain the first derivatives at the end points $t = 0$ and $t = 1$ are

$$\begin{aligned} \mathbf{B}'(0) &= 3(\mathbf{P}_1 - \mathbf{P}_0) \\ \mathbf{B}'(1) &= 3(\mathbf{P}_3 - \mathbf{P}_2) \end{aligned} \quad \text{(A.8)}$$

And the second derivatives at the end points are

$$\begin{aligned} \mathbf{B}''(0) &= 6(\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2) \\ \mathbf{B}''(1) &= 6(\mathbf{P}_1 - 2\mathbf{P}_2 + \mathbf{P}_3) \end{aligned} \quad \text{(A.9)}$$

A cubic spline curve is relaxed if its second derivative is zero at each end point. If $\mathbf{B}''(0) = 0$, we have $2P_1 = P_0 + P_2$, which means P_1 is the midpoints of $\overline{P_0P_2}$. A similar relation holds in case $B''(1) = 0$.

B-spline continuity We want to develop $\mathcal{C}^2$ splines (the 0th through 2th derivatives are continuous). For each segment, the cubic Bezier curve is already $\mathcal{C}^2$ continues.

Therefore, we only consider the continuity at the joints. First, we consider the case of two cubic Bezier curves $\mathbf{B}_1(t)$ with control points P_0^1, P_1^1, P_2^1 and P_3^1 and $\mathbf{B}_2(t)$ with control points P_0^2, P_1^2, P_2^2 and P_3^2. The $\mathcal{C}^2$ continuity requires that

$$\mathcal{C}^0 : \mathbf{B}_1(1) = \mathbf{B}_2(0) \tag{A.10a}$$
$$\mathcal{C}^1 : \mathbf{B}_1'(1) = \mathbf{B}_2'(0) \tag{A.10b}$$
$$\mathcal{C}^2 : \mathbf{B}_1''(1) = \mathbf{B}_2''(0) \tag{A.10c}$$

With equations A.3, A.6 and A.7, we obtain:

$$\mathcal{C}^0 : P_3^1 = P_0^2 \tag{A.11a}$$
$$\mathcal{C}^1 : P_3^1 - P_2^1 = P_1^2 - P_0^2 \tag{A.11b}$$
$$\mathcal{C}^2 : P_1^1 - 2P_2^1 + P_3^1 = P_0^2 - 2P_1^2 + P_2^2 \tag{A.11c}$$

Equation A.11a means the joint, which can be noted as S, then $S = P_3^1 = P_0^2$. Equation A.11b means the S is the middle point of P_1^2 and P_2^1, more formally $2S = (P_2^1 + P_1^2)$. Equation A.11c can be equivalently written as $2P_2^1 - P_1^1 = 2P_1^2 - P_2^2$. If a point B is assigned to $B = 2P_2^1 - P_1^1 = 2P_1^2 - P_2^2$, the points P_1^1, P_2^1, B, P_1^2 and P_2^2 form a structure, named A-frame as shown in figure A.2. In the A-frame, the joint S is the midpoint of $\overline{P_2^1 P_1^2}$, P_2^1 is the midpoint of $\overline{P_1^1 B}$, and P_1^2 is the midpoint of $\overline{B P_2^2}$. B is called the control point or de Boor point of the interpolated B-spline. Each joint corresponds to a control point.

$$S_i = \frac{1}{2}(P_2^{i-1} + P_1^i) = \frac{1}{2}\big((\frac{1}{3}B_{i-1} + \frac{2}{3}B_i) + (\frac{2}{3}B_i + \frac{1}{3}B_{i+1})\big) = \frac{1}{6}B_{i-1} + \frac{4}{6}B_i + \frac{1}{6}B_{i+1} \tag{A.12}$$

B-spline interpolation Suppose that there are $n+1$ data points $S_0 \dots S_n$. A relaxed cubic spline curve $S(t)$ with $0 \le t \le n$ are interpolated passing all these data points. More precisely, a piecewise-polynomial curve is found that satisfies $S(i) = S_i$ for all $0 \le i \le n$. $S(t)$ is determined by a control polygon with control points $\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_n$ as shown in figure A.3. If these control points are known, taking into account the $\mathfrak{C}^2$ continuity conditions (A-frame condition: equation A.12) and the relaxed end conditions, for ith piece, the four Bezier control points can be obtained as:

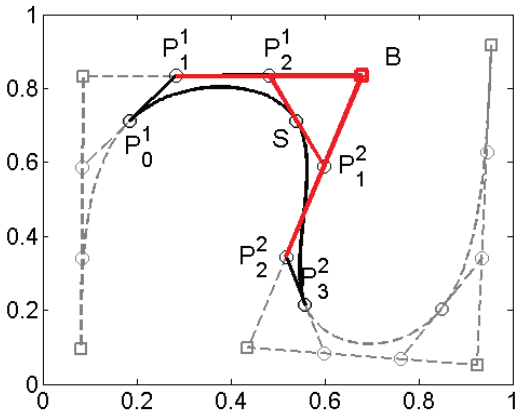


Figure A.2: joint two Bezier curve

$$P_0^i = S_i = \frac{1}{6}B_{i-1} + \frac{4}{6}B_i + \frac{1}{6}B_{i+1} \tag{A.13a}$$

$$P_1^i = \frac{2}{3}B_i + \frac{1}{3}B_{i+1} \tag{A.13b}$$

$$P_2^i = \frac{1}{3}B_i + \frac{2}{3}B_{i+1} \tag{A.13c}$$

$$P_3^i = S_{i+1} = \frac{1}{6}B_i + \frac{4}{6}B_{i+1} + \frac{1}{6}B_{i+2} \tag{A.13d}$$

Which can be written in matrix form as

$$\begin{bmatrix} P_0^i \\ P_1^i \\ P_2^i \\ P_3^i \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}\begin{bmatrix} B_{i-1} \\ B_i \\ B_{i+1} \\ B_{i+2} \end{bmatrix} \tag{A.14}$$

Substituting A.13 into A.5, we obtain the B-spline representation of ith piece

$$\mathbf{S}_i(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}\frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} B_{i-1} \\ B_i \\ B_{i+1} \\ B_{i+2} \end{bmatrix} \tag{A.15}$$

The first method is to calculate $B_0, B_1, \ldots, B_n$ from the known $S_0 \ldots S_n$. Recall that the two end points $S_0 = B_0$ and $S_n = B_n$, and the equation A.12:

$$\begin{cases} S_0 &= B_0 \\ S_1 &= \frac{1}{6}B_0 + \frac{2}{3}B_1 + \frac{1}{6}B_2 \\ &\vdots \\ S_i &= \frac{1}{6}B_{i-1} + \frac{2}{3}B_i + \frac{1}{6}B_{i+1} \\ &\vdots \\ S_{n-1} &= \frac{1}{6}B_{n-2} + \frac{2}{3}B_{n-1} + \frac{1}{6}B_n \\ S_n &= B_n \end{cases} \tag{A.16}$$

Here the B_i and S_i are points, so that in $\mathbf{R}^2$ they are pairs of numbers. The equations can be written as matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ \vdots \\ B_{n-2} \\ B_{n-1} \\ B_n \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ \vdots \\ S_{n-2} \\ S_{n-1} \\ S_n \end{bmatrix} \tag{A.17}$$

Let $\mathbf{M}$ be the matrix of coefficients, let $\mathbf{B}$ be the matrix $\mathbf{B} = [B_0 \ldots B_n]^\top$, and let $\mathbf{S}$ be the matrix $\mathbf{S} = [S_0 \ldots S_n]^\top$. Then equation can be simplified as

$$\mathbf{MB} = \mathbf{S} \tag{A.18}$$

Note that the $(n+1) \times (n+1)$ matrix $\mathbf{M}$ is a strictly diagonally dominant matrix that for every row of $\mathbf{M}$, the magnitude of the diagonal entry in a row is larger than the sum of the magnitudes of all the other. Recall that a strictly diagonally dominant matrix is non-singular so that it is invertible. Hence, $\mathbf{B}$ can be obtained by

$$\mathbf{B} = \mathbf{M}^{-1}\mathbf{S} \tag{A.19}$$

Having the B-spline control points, the Bezier control points for each segment can be

calculated as indicated in equation A.13 or matrix form A.14. In some materials, the coefficient matrix $\mathbf{M}$ can be derived as "1 4 1" matrix from equation A.16.

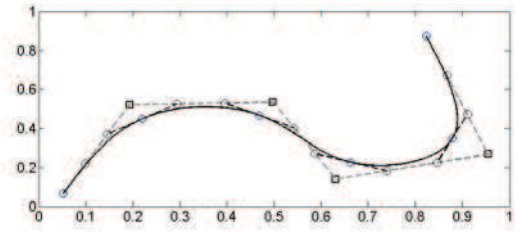


Figure A.3: cubic B-spline interpolation: the first method

An alternative method can be accomplished without recover the control points of B-spline. The idea behind is to set up $\mathcal{C}^2$ continuity constrains and solve for the first derivatives (tangent vector) D_i at each S_i. Once $D_0 \dots D_n$ are known, Bezier control points can be computed from them using equation A.8. We have:

$$P_0^i = S_i \tag{A.20a}$$

$$P_1^i = S_i + \frac{1}{3} D_i \tag{A.20b}$$

$$P_2^i = S_{i+1} - \frac{1}{3} D_{i+1} \tag{A.20c}$$

$$P_3^i = S_{i+1} \tag{A.20d}$$

Because of $S_i''(1) = S_{i+1}''(0)$, we have $6(P_1^i - P_2^i + P_3^i) = 6(P_0^{i+1} - P_1^{i+1} + P_2^{i+1})$,and taking into account A.20, we obtain $D_i + 4D_{i+1} + D_{i+2} = 3(S_{i+2} - S_i)$. Considering the relax end conditions, we obtain $D_0 + 2D_1 = 3(S_1 - S_0)$ and $D_{n-1} + 2D_n = 3(S_n - S_{n-1})$. Therefore, we have the matrix form as

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{3} & \frac{4}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_{n-2} \\ D_{n-1} \\ D_n \end{bmatrix} = \begin{bmatrix} C_1 - C_0 \\ C_2 - C_0 \\ C_3 - C_1 \\ \vdots \\ C_{n-1} - C_{n-3} \\ C_n - C_{n-2} \\ C_n - C_{n-1} \end{bmatrix} \tag{A.21}$$

We can calculate $D_0 \dots D_n$ by inverse the coefficient matrix. Then we can obtain the

control points of each Bezier piece by equation A.20. A simulation is illustrated in figure A.4.

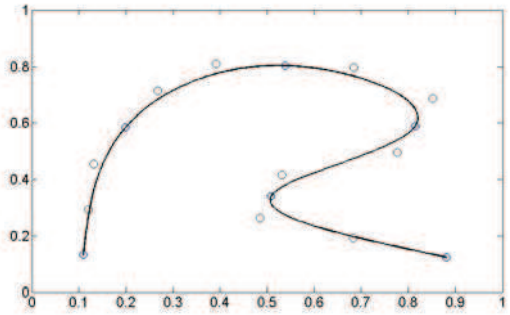


Figure A.4: Cubic B-spline interpolation: the second method

Appendix B

Appendix B

B.1 Orientation and Rotation

The rotation of a rigid body in 3-D can be described in several ways. Here, we give the descriptions of the methods used in this thesis.

Rotation matrix

A rotation is referred to a rotation matrix, which is an orthogonal matrix. The determinant of rotation matrix is always 1.

Euler rotation

Euler rotation splits the complete rotation into three simpler constitutive rotations. Depending on the rotation axex and the order of a series of rotations, there are 24 possible constitutive rotations to represent arbitrary 3-D rotations. The rotation axes can be original fixed frame or rotated frame. For example, considering rotated frame, using the xyz convention, a rotation matrix is

$$\mathbf{R} = \mathbf{R}_x(roll)\mathbf{R}_y(pitch)\mathbf{R}_z(yaw). \tag{B.1}$$

If using the zyx convention, the rotation matrix is

$$\mathbf{R} = \mathbf{R}_z(roll)\mathbf{R}_y(pitch)\mathbf{R}_x(yaw). \tag{B.2}$$

If considering original fixed frame, using xyz convention, the rotation matrix is

$$\mathbf{R} = \mathbf{R}_z(yaw)\mathbf{R}_y(pitch)\mathbf{R}_x(roll), \tag{B.3}$$

which is similar to the zyx convention, in rotated frame, and has the opposite order with the xyz convention. For example, in Figure B.1, the current camera frame is red. In order to get the rotation of current camera frame, the parameters of different methods are given in Table B.1.

Table B.1: Parameters: an example of Euler rotation

Frame	Convention	$\mathbf{R}$
In the rotated frame	xyz	$\mathbf{R}_x(-\pi/2)\mathbf{R}_y(\pi/6)\mathbf{R}_z(0)$
	zyx	$\mathbf{R}_z(-\pi/6)\mathbf{R}_y(0)\mathbf{R}_x(-\pi/2)$
In the original fixed frame	xyz	$\mathbf{R}_z(-\pi/6)\mathbf{R}_y(0)\mathbf{R}_x(-\pi/2)$

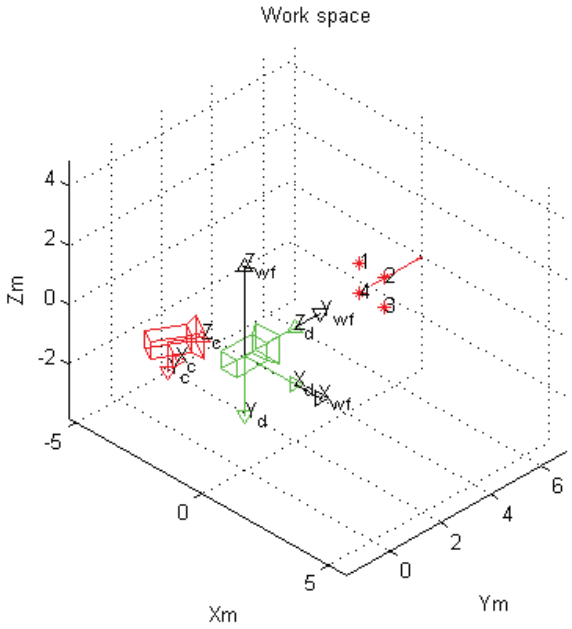


Figure B.1: Work space: an example of Euler rotations.

Commonly rotations are represented by Euler angles

Rotations are also commonly represented by roll-pitch-yaw angles. For example, considering rotated frame, the rotation matrix is $\mathbf{R} = \mathbf{R}_x(roll)\mathbf{R}_y(pitch)\mathbf{R}_z(yaw)$ using xyz convention. If considering original fixed frame, the rotation matrix is $\mathbf{R} = \mathbf{R}_z(yaw)\mathbf{R}_y(pitch)\mathbf{R}_x(roll)$ using xyz convention.

Angular axis

Any rotation can be expressed in terms of a single rotation about some axis. This is referred to angular axis representation $(\theta, \mathbf{u})$, where θ is the angle and $\mathbf{u}$ is unit vector representing the direction of the rotation axis. The representation $(\theta, \mathbf{u})$ is typically considered as a rotation vector:

$$\mathbf{v} = \theta\mathbf{u}, \tag{B.4}$$

where

$$\begin{aligned} \theta &= |\mathbf{v}| \\ \mathbf{u} &= \tfrac{\mathbf{v}}{\theta} \end{aligned} \tag{B.5}$$

The relationship between rotation matrix $\mathbf{R}$ and angular-axis vector $\mathbf{v}$ is represented by rodriguar equation.

$$\mathbf{R} = \cos\theta\, \mathbf{I}_3 + (1 - \cos\theta)\, \mathbf{u}\mathbf{u}^\top + \sin\theta\, [\mathbf{u}]_\times \tag{B.6}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix and $[\mathbf{u}]_\times$ is the skew-symmetric matrix of $\mathbf{u}$.

B.2 Robot Jacobian

The kinematic screw of the camera is the instantaneous forward kinematics $\mathbf{V}_C = [\boldsymbol{\upsilon}_c^\top\ \boldsymbol{\omega}_c^\top]^\top$. If the camera is fixed on the mobile robot, $\mathbf{V}_C$ can be obtained from the transformation of screws, which is given by [68]:

$$\mathbf{V}_C = {}^C\mathbb{T}_R \mathbf{V}_R \tag{B.7}$$

where ${}^{C}\mathbb{T}_{R}$ is the (6×6) transformation matrix between screws:

$$ {}^{C}\mathbb{T}_{R} = \begin{bmatrix} {}^{C}\mathbf{A}_{R} & -{}^{C}\mathbf{A}_{R}\,{}^{C}\hat{\mathbf{P}}_{R} \\ \mathbf{0} & {}^{C}\mathbf{A}_{R} \end{bmatrix} \tag{B.8} $$

where ${}^{C}\hat{\mathbf{P}}_{R}$ is the 3×3 skew-symmetric matrix of the translation vector ${}^{C}\mathbf{t}_{R}$ matrix. ${}^{C}\hat{\mathbf{P}}_{R}$ is given by

$$ {}^{C}\mathbf{t}_{R} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{B.9} $$

Consider a simple configuration illustrated in Figure **??**. We obtain:

$$ {}^{C}\mathbf{T}_{R} = \begin{bmatrix} 0 & -1 & 0 & -t_x \\ 0 & 0 & -1 & -t_y \\ 1 & 0 & 0 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.10} $$

then we obtain ${}^{C}\mathbb{T}_{R}$:

$$ {}^{C}\mathbb{T}_{R} = \begin{bmatrix} 0 & -1 & 0 & t_z & 0 & -t_x \\ 0 & 0 & -1 & -t_y & t_x & 0 \\ 1 & 0 & 0 & 0 & t_z & -t_y \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{B.11} $$

B.3 Linear Least Square

Inhomogeneous Linear Least Squares Problem

Inhomogenous linear least square problem can be described as:

$$ \mathbf{A}X = \mathbf{b}. \tag{B.12} $$

in which the $\mathbf{x}$ can be solve using the pseudoinverse or inverse of $\mathbf{A}$.

Homogeneous Linear Least Squares Problem

Homogenous linear least square problem can be described as:

$$\mathbf{A}X = \mathbf{0}. \tag{B.13}$$

which is different with inhomogeneous case. The pseudoinverse or inverse does not work. Instead, homogenous case can be solved using Singular Value Decomposition (SVD).

B.4 Non-linear Least Squares Problem and Gaussian-Newton Optimization

Non-linear least squares uses least squares analysis to fit a set of M observations with a model, which is non-linear in N unknown parameters ($N < M$).

Suppose a model $h(\mathbf{q}) : x \mapsto y$, where $\mathbf{q}_{N\times 1}$ is the vector of parameters. Given a set of M data points $(X, Y) = \{(x_i, y_i)| \ \forall \ i = 0, \ldots, M-1\}$. The basis of the method is to estimate the optimal $\mathbf{q}$ with which the model can fit best with the data points (X, Y) in the least square sense. That is, the approximation is evaluated using the sum of squares

$$f(\mathbf{q}) = \sum_{i=0}^{M-1} r_i^2 = \|\mathbf{r}\|^2, \tag{B.14}$$

and $f(\mathbf{q})$ should be minimized. In equation B.14, $\mathbf{r} = [r_0 \ \ldots \ r_{M-1}]^\top$ is the vector of residuals (or errors), which is given by:

$$r_i = y_i - h(x_i, \mathbf{q}), \ \forall \ i = 0, \ldots, M-1. \tag{B.15}$$

Since $h(x_i, \mathbf{q})$ depends non-linearly on the vector of parameters $\mathbf{q}$, each residual may also depend non-linearly on $\mathbf{q}$. That is the reason that the minimization of $f(\mathbf{q})$ is a non-linear least squares problem.

The function $f(\mathbf{q})$ gets the minimum value when the gradient vector $\mathbf{g}(\mathbf{q})$ is zero:

$$\mathbf{g}(\mathbf{q}) \doteq \nabla f(\mathbf{q}) = \left[\begin{array}{ccc} \frac{\partial f}{\partial q_0} & \cdots & \frac{\partial f}{\partial q_{N-1}} \end{array} \right]^\top = \mathbf{0}. \tag{B.16}$$

Using optimal methods, this equation (or equation B.14) can be approximated and the parameters are refined iteratively. In each iteration step, expand the function $f(\mathbf{q})$ by its Taylor series expansion about the point $\mathbf{q}_k$

$$f(\mathbf{q}_k + \delta\mathbf{q}) = f(\mathbf{q}_k) + \mathbf{g}(\mathbf{q}_k)^\top \delta\mathbf{q} + \frac{1}{2}\delta\mathbf{q}^\top \mathbf{H}(\mathbf{q}_k)\delta\mathbf{q} + \ldots. \tag{B.17}$$

where $\mathbf{H}(\mathbf{q})$ is the Hessian matrix of $f(\mathbf{q})$:

$$\mathbf{H}(\mathbf{q}) \doteq \nabla\nabla f(\mathbf{q}) = \left[\begin{array}{ccc} \frac{\partial^2 f}{\partial q_0^2} & \cdots & \frac{\partial^2 f}{\partial q_0 \partial q_{N-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial q_0 \partial q_{N-1}} & \cdots & \frac{\partial^2 f}{\partial q_{N-1}^2} \end{array} \right]. \tag{B.18}$$

Gaussian Newton algorithm approximate $f(\mathbf{q})$ by truncating the quadratic term:

$$f(\mathbf{q}_k + \delta\mathbf{q}) \approx f(\mathbf{q}_k) + \mathbf{g}(\mathbf{q}_k)^\top \delta\mathbf{q} + \frac{1}{2}\delta\mathbf{q}^\top \mathbf{H}(\mathbf{q}_k)\delta\mathbf{q}. \tag{B.19}$$

Find the $\delta\mathbf{q}$ which minimizes this quadratic approximation:

$$\nabla f(\mathbf{q}_k + \delta\mathbf{q}) = \mathbf{g}(\mathbf{q}_k) + \mathbf{H}(\mathbf{q}_k)\delta\mathbf{q} = 0. \tag{B.20}$$

So the shift vector of parameters is:

$$\delta\mathbf{q} = \mathbf{q}_{k+1} - \mathbf{q}_k = -\mathbf{H}(\mathbf{q}_k)^{-1}\mathbf{g}(\mathbf{q}_k). \tag{B.21}$$

In each iteration step, the predicted vector of parameters is:

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \mathbf{H}(\mathbf{q}_k)^{-1}\mathbf{g}(\mathbf{q}_k). \tag{B.22}$$

Then $mathbfq_k$ is updated with $\mathbf{q}_{k+1}$ in the next iteration. The iteration is stopped

when

$$\|\frac{\delta \mathbf{q}}{\mathbf{q}_k}\| < \tau, \tag{B.23}$$

where τ is a given value.

Since $\forall j = 0, \ldots, N-1$

$$\frac{\partial f}{\partial q_j} = \frac{\partial}{\partial q_j} \sum_{i=0}^{M-1} r_i^2 = \sum_{i=0}^{M-1} 2 r_i \frac{\partial r_i}{\partial q_j}, \tag{B.24}$$

hence

$$\mathbf{g}(\mathbf{q}) = 2\mathbf{J}_\mathbf{r}(\mathbf{q})^\top \mathbf{r} \tag{B.25}$$

where $\mathbf{J}_\mathbf{r}(\mathbf{q})$ is the $M \times N$ Jacobian matrix of the vector of residuals $\mathbf{r}$:

$$\mathbf{J}_\mathbf{r}(\mathbf{q}) = \begin{bmatrix} \frac{\partial r_0}{q_0} & \cdots & \frac{\partial r_0}{q_{N-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_{M-1}}{q_0} & \cdots & \frac{\partial r_{M-1}}{q_{N-1}} \end{bmatrix}. \tag{B.26}$$

Since

$$\frac{\partial^2 f}{\partial q_l \partial q_k} = \frac{\partial^2}{\partial q_l \partial q_k} \sum_{i=0}^{M-1} r_i^2 = 2\frac{\partial}{\partial q_l} \sum_{i=0}^{M-1} r_i \frac{\partial r_i}{\partial q_k} \tag{B.27}$$

$$= 2 \sum_{i=0}^{M-1} \frac{\partial r_i}{\partial q_k \partial q_l} + 2 \sum_{i=0}^{M-1} r_i \frac{\partial^2 r_i}{\partial q_k \partial q_l}, \tag{B.28}$$

Suppose r_i is very small, this equation can be approximated by

$$\frac{\partial^2 f}{\partial q_l \partial q_k} \approx 2 \sum_{i=0}^{M-1} \frac{\partial r_i}{\partial q_k \partial q_l}. \tag{B.29}$$

Hence

$$\mathbf{H}_\mathbf{r} \approx 2\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r}. \tag{B.30}$$

Then the shift vector of parameters can be approximated by:

$$\delta \mathbf{q} \approx -(\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r})^{-1} \mathbf{J}_\mathbf{r}^\top \mathbf{r}. \tag{B.31}$$

Usually instead of computing the inverse matrix $(\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r})^{-1}$ directly, $\delta \mathbf{q}$ is obtained by solving the function

$$\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r} \delta \mathbf{q} \approx -\mathbf{J}_\mathbf{r}^\top \mathbf{r}. \tag{B.32}$$

which is called the normal equations.

B.5 Visual target moment

2-D basic moments are defined by

$$m_{ij} = \int \int x^i y^j dx dy \tag{B.33}$$

If in the discrete space, the basic moments are defined by

$$m_{ij} = \sum_{k=1}^{n} x_k^i y_k^j \tag{B.34}$$

A useful moment is the gravity center. Let (x_g, y_g) be the coordinates of the gravity center of object, whose maximum order is 1. These coordinates are defined as follows:

$$\begin{cases} x_g = \frac{m_{01}}{m_{00}} \\ y_g = \frac{m_{10}}{m_{00}} \end{cases} \tag{B.35}$$

B.6 Motion Estimation using Visual Tracking

Efficient outlier removing Outliers removing by RANSAC is computational expensive especial for high dimensional data. An alternative way to remove outliers is to use circle matching between consecutive frames in a closed way, which is common in stereo vision systems. This scheme can not be guaranteed in theory. But it is practical and effective in the sense that it provides credible matching results. Figure B.2 illustrate an experimental results of circle matching for stereo vision systems.

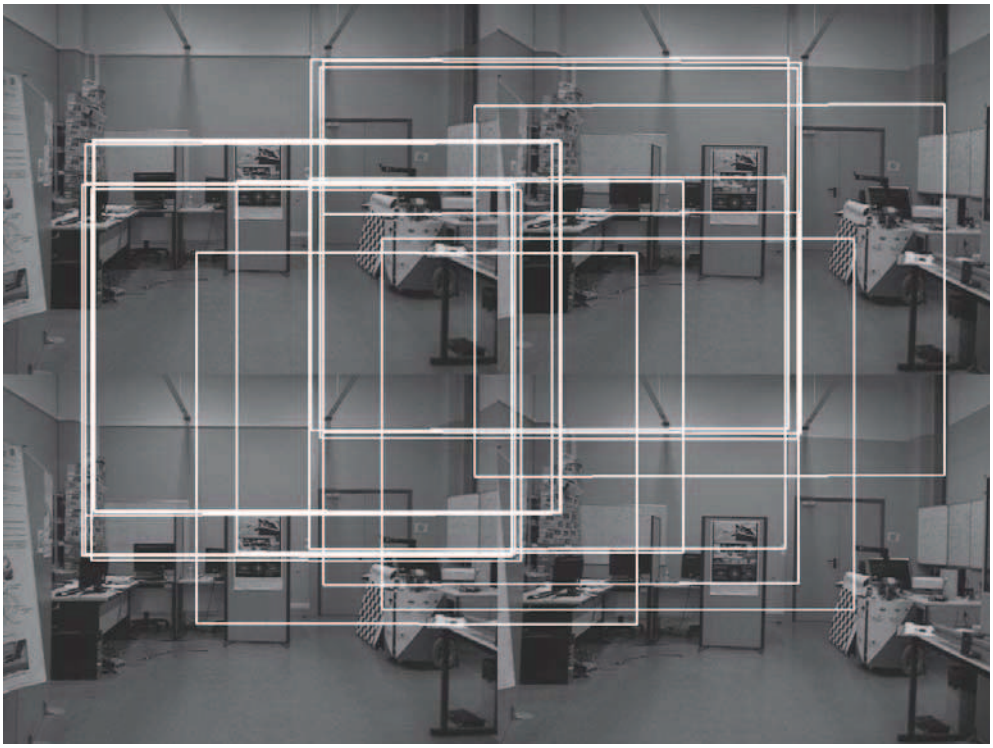


Fig. B.2: An efficient outliers removing using circle matching.

Suppose the camera moves in 6-DoF. The motion is parameterized by a vector $[\mathbf{r}^\top, \mathbf{t}^\top]$. $\mathbf{r}^\top = [r_x, r_y, r_z]^\top$ includes rotation angles with respect to x, y, and z axes respectively. $\mathbf{t}^\top$ is the translation vector. The vector of motion parameters is estimated by minimizing the sum of re-projection errors.

The coordinate System is defined as illustrated in Figure B.3. The left previous camera frame is selected as reference frame. The motion estimation is to compute the transformation ${}^{p}\mathbf{T}_c$ between the previous and the current frames of the left camera. The motion can be parameterized by a vector $[\mathbf{r}^\top, \mathbf{t}^\top]$. $\mathbf{r}^\top = [r_x, r_y, r_z]^\top$ includes rotation angles with respect to x, y, and z axes respectively. $\mathbf{t}^\top$ is the translation vector. The vector of motion parameters is estimated by minimizing the sum of re-projection errors.

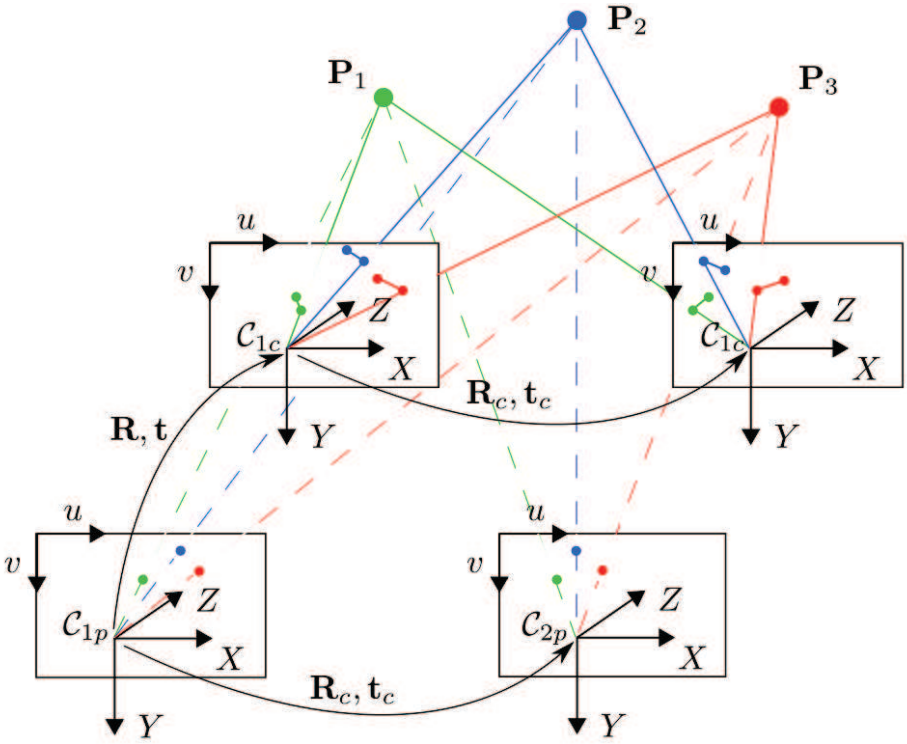


Fig. B.3: The illustration of stereo visual odometry.

For each two consecutive stereo pairs, the correspondences are found with image matching. The 3-D points $\{\mathbf{P}_i\}$ are triangulated from the previous stereo images. The

triangulation of stereo pairs in the left camera frame can be expressed:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{b(u_{1p}-u_{1c})}{d} \\ \frac{b(v_{1p}-v_{1c})}{d} \\ \frac{fb}{d} \end{bmatrix} \tag{B.36}$$

where b is the baseline of calibrated stereo cameras. $(u_{1p} - u_{1c})$ is the Parallax.

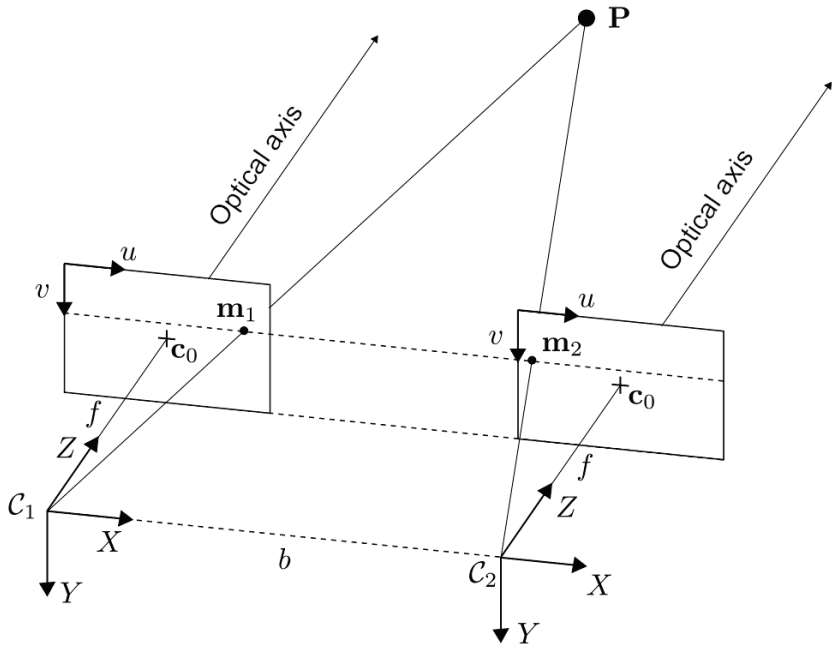


Fig. B.4: Stereo vision.

$\{\mathbf{P}_i\}$ are then transformed to the current left camera frame and re-projected to the current left and right image planes with unknown vector of parameters $\mathbf{q}$. The re-projection is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \left[[\mathbf{R}(\mathbf{r})\,\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} \right], \tag{B.37}$$

where

$$d = \begin{cases} 0, & \text{for the left camera} \\ b, & \text{for the right camera} \end{cases} . \tag{B.38}$$

In the current left and right image planes, the re-projected image points are compared with the matched image points to give the re-projection errors. The re-projection errors are measurement as:

$$y(\mathbf{q}) = \sum_{i=0}^{M-1} (\|m_i^l - \Pi^l(\mathbf{X}_i, \mathbf{q})\|^2 + \|m_i^r - \Pi^r(\mathbf{X}_i, \mathbf{q})\|^2). \tag{B.39}$$

The motion vector of parameters $\mathbf{q}$ can be estimated by minimizing this re-projection errors. Gaussian-Newton algorithm B.4 is used to computer the optimal $\mathbf{q}$ when the equation B.39 gets the minimum value. The scheme randomly selects a minimum sample points. 3 independent correspondences are enough to estimate 6-dimensional vector of parameters. These 3 correspondences are used to estimated the vector of parameters $\mathbf{q}$, and the rest of correspondences are evaluated inliers or outliers. This step repeat a given times to select the best estimated inliers, which includes most correspondences.

Appendix C

Basic Performances of Spherical Image-based Visual Servoing

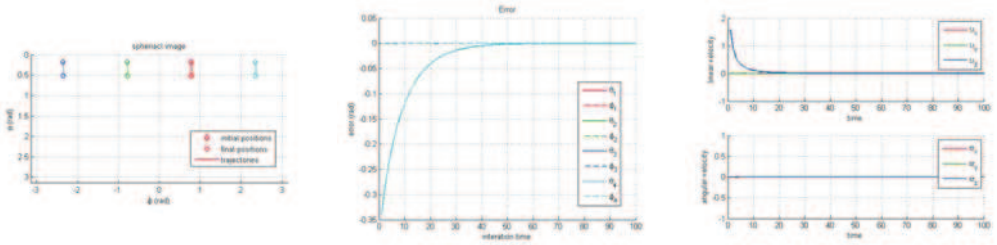
Here, we present the simulation results of SIBVS for the cases: pure z-axis translation, pure z-axis rotation, z-axis rotation and translation, x-axis translation, pure x-axis rotation, and x-axis rotation and translation. The cases of pure y-axis translation, pure y-axis rotation, and y-axis rotation and translation are not presented, as they are similar to the cases of x-axis.

Pure z-axis Translation

We simulate a pure z-axis translation by positioning the initial camera frame at $(0, 8, 0)$ with orientation $(\pi/2, 0, \pi)$ in roll-pitch-yaw angles. The simulation result is given in Figure C.1. Figure C.1(a) shows that the visual features have moved along lines of constant longitude. Figure C.1(b) shows that the error approximately performs an exponential decoupled decrease. Figure C.1(c) shows that the camera has done a desired pure translational motion in the z-axis.

Pure z-axis Rotation

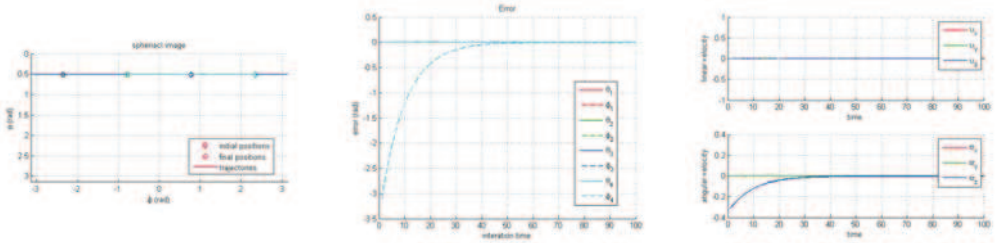
We simulate a pure z-axis rotation by positioning the initial camera frame at $(0, 2.5, 0)$ with orientation $(\pi/2, 0, 0)$ in roll-pitch-yaw angles. The simulation result is given by Figure C.2, which shows that the camera has done a desired pure rotation without camera retreat phenomenon, which happened using perspective projection model. The



(a) Camera view. (b) Error. (c) Camera velocities.

Figure C.1: Pure translation in z-axis.

visual features only have motion along φ axis in the spherical image, and the error approximately performs an exponential decoupled decrease.



(a) Camera view. (b) Error. (c) Camera velocities.

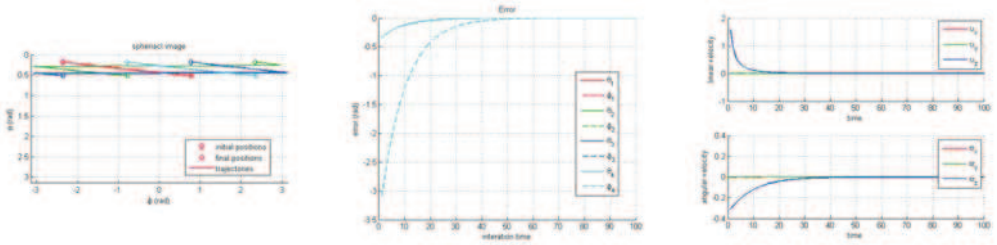
Figure C.2: Pure rotation along z-axis.

z-axis Translation and Rotation

We simulate a z-axis translation and rotation by positioning the initial camera frame at $(0, 8, 0)$ with orientation $(\pi/2, 0, 0)$ in roll-pitch-yaw angles. The simulation result is given by Figure C.3, which shows that the camera has done a desired translation and rotation only with respect to z-axis. The trajectories of visual features are broken lines in the spherical image. The trajectories should be straight lines. They are broken because of the periodicity of angles. Broken points of the lines are located at boundary values, where $\varphi = -\pi$ or $\varphi = \pi$.

Pure x-axis Translation

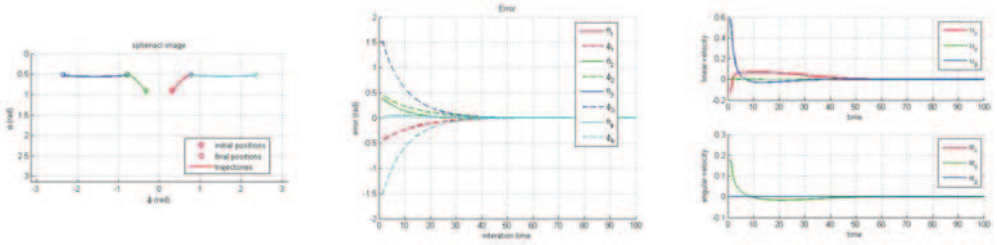
We simulate a pure x-axis translation by positioning the initial camera frame at $(2, 2.5, 0)$ with orientation $(\pi/2, 0, \pi)$ in roll-pitch-yaw angles. The simulation result is given in Figure C.4. Figure C.4(a) shows that the trajectories of the visual features are approx-



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure C.3: A combined translation and rotation about z-axis

imately straight lines. Figure C.4(b) shows that the error approximately performs an exponential decoupled decrease. Figure C.4(c) shows that the camera has done an unexpected translational motion in the z-axis direction and unexpected rotational motion along the y-axis.



(a) Camera view. (b) Error. (c) Camera velocities.

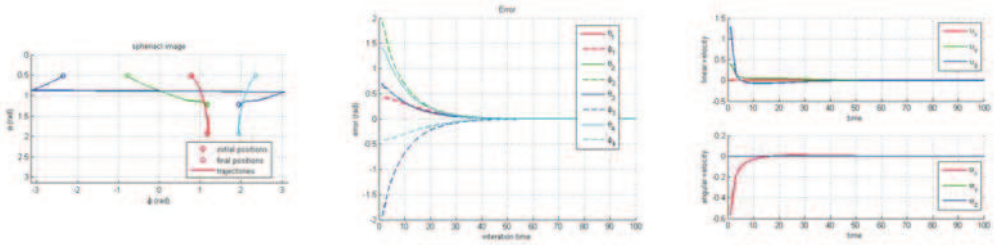
Figure C.4: Pure translation about x-axis.

Pure x-axis Rotation

We simulate a pure x-axis rotation by positioning the initial camera frame at $(0, 2.5, 0)$ with orientation $(0, 0, \pi)$ in roll-pitch-yaw angles. The simulation result is given in Figure C.5. It shows that the camera has done an unexpected translational motion in the y- and z-axes direction. This unexpected translation motion is also existed in the IBVS using perspective projecton model as shown in Figure C.6. It is a result of crosscoupling terms in the interaction matrix, equation 3.35 for spherical IBVS or equation 3.10 for perspective IBVS.

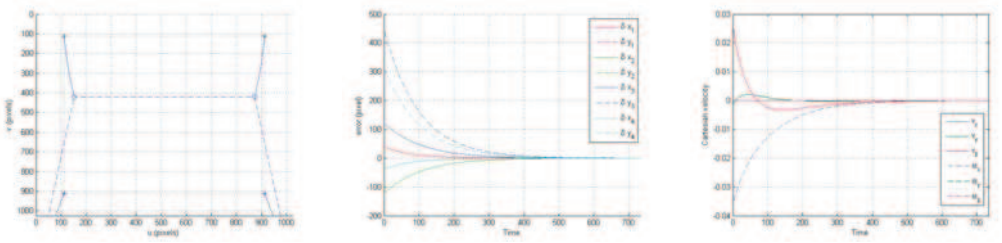
x-axis Translation and Rotation

We simulate a x-axis translation and rotation by positioning the initial camera frame



(a) Camera view. (b) Error. (c) Camera velocities.

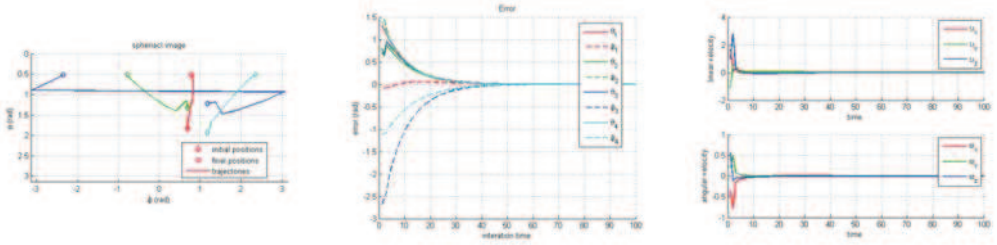
Figure C.5: Pure rotation along x-axis.



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure C.6: Pure rotation about x-axis $\mathbf{R}_x(\frac{\pi}{9})$ using IBVS

at $(2, 2.5, 0)$ with orientation $(0, 0, \pi)$ in roll-pitch-yaw angles. The simulation result is given by Figure C.7, which shows that the camera has unwanted translational and rotational motion about the y- and z-axes.



(a) Camera view. (b) Error vector. (c) Camera velocities.

Figure C.7: A combined translation and rotation about x-axis

References

[1] H.H. Abdelkader, Y. Mezouar, N. Andreff, and P. Martinet. 2 1/2 d visual servoing with central catadioptric cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 3572–3577, 2005. 34

[2] P.K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *Robotics and Automation, IEEE Transactions on*, 9(2):152–165, 1993. 67

[3] M. Ambai and Y. Yoshida. CARD: Compact And Real-time Descriptors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 97 – 104, nov. 2011. 19

[4] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, pages 2911–2918, 2012. 18

[5] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, 1992. 61

[6] Ronen Basri, Ehud Rivlin, and Ilan Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 33(2):117–137, 1999. 67, 73

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. 15, 18, 33

[8] P. R. Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, pages 579–583, November 1978. 15

[9] Hector M. Becerra, Jonathan Courbon, Youcef Mezouar, and Carlos Sagüés. Wheeled mobile robots navigation from a visual memory using wide field of view cameras. In *IROS*, pages 5693–5699, 2010. 67

[10] H.M. Becerra, G. López-Nicolás, and C. Sagüés. A sliding-mode-control law for mobile robots based on epipolar visual servoing from three views. *Robotics, IEEE Transactions on*, 27(1):175–183, 2011. 67

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509 –522, apr 2002. 19

[12] S. Benhimane and E. Malis. Vision-based control with respect to planar and non-planar objects using a zooming camera. In *IEEE International Conference on Advanced Robotics*, pages 991–996, 2003. 68

[13] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *I. J. Robotic Res.*, 26(7):661–676, 2007. 33

[14] Selim Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 1, pages 943–948 vol.1, 2004. 33

[15] A. Bicchi, M. Peshkin, and J. Colgate. Safety for physical humanrobot interaction. In B. Siciliano and O. Khatib, editors, *The Handbook of Robotics*, pages 1335–1348. Springer, Berlin, Heidelberg, Germany, 2008. 37

[16] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008. 2

[17] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *Robotics and Automation, IEEE Transactions on*, 7(4):535–539, 1991. 40

[18] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991. 41

[19] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. 73

[20] D. Burschka and G. Hager. Vision-based control of mobile robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1707–1713 vol.2, 2001. 67

[21] Michael Calonder, Vincent Lepetit, Mustafa Özuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1281–1298, 2012. 19

[22] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision (ECCV): Part IV*. 19

[23] F. Chaumette. *La relation vision-commande: théorie et application à des tâches robotiques*. PhD thesis, Université de Rennes 1, July 1990. 73

[24] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13(4):82–90, 2006. 67

[25] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *Robotics Automation Magazine, IEEE*, 14(1):109–118, 2007. 67

[26] F. Chaumette and S. Hutchinson. Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, March 2007. 65

[27] A. Cherubini and F. Chaumette. A redundancy-based approach for obstacle avoidance in mobile robot navigation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5700–5705, 2010. 61, 68

[28] A. Cherubini and F. Chaumette. Visual navigation with obstacle avoidance. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1593–1598, 2011. 61

[29] A. Cherubini, F. Spindler, and F. Chaumette. A redundancy-based approach for visual navigation with collision avoidance. In *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2011 IEEE Symposium on*, pages 8–15, 2011. 61

[30] Andrea Cherubini and François Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *I. J. Robotic Res.*, 32(2):189–205, 2013. 68

[31] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. Keeping features in the camera's field of view: a visual servoing strategy. In *15th Int. Symp. on Mathematical Theory of Networks and Systems*, August 2002. 68

[32] Peter I. Corke. Spherical image-based visual servo and structure estimation. In *ICRA*, pages 5550–5555, 2010. 92

[33] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052 –1067, june 2007. 33

[34] A. De Luca and G. Oriolo. Local incremental planning for nonholonomic mobile robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 104–110 vol.1, 1994. 41

[35] Claudio De Medio and Giuseppe Oriolo. Robot obstacle avoidance using vortex fields, 1991. 41

[36] Daniel DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1-2):123–141, 1995. 82

[37] G.N. DeSouza and A.C. Kak. Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267, 2002. 2

[38] Albert Diosi, Sinisa Segvic, Anthony Remazeilles, and François Chaumette. Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):870–883, 2011. 67

[39] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. 48

[40] A. Elfes. Sonar-based real-world mapping and navigation. *Robotics and Automation, IEEE Journal of*, 3(3):249–265, 1987. 39

[41] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 39

[42] B. Espiau, F. Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *Robotics and Automation, IEEE Transactions on*, 8(3):313–326, 1992. 67

[43] Bernard Espiau. Effect of camera calibration errors on visual servoing in robotics. In *ISER*, pages 182–192, 1993. 67

[44] Ariane S Etienne, Roland Maurer, Valérie Boulens, Arik Levy, and Tiffany Rowe. Resetting the path integrator: a basic condition for route-based navigation. *Journal of Experimental Biology*, 207(9):1491–1508, 2004. 77

[45] OD FAUGERAS and F LUSTMAN. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988. 18, 85

[46] J.T. Feddema and Owen R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *Robotics and Automation, IEEE Transactions on*, 5(5):691–700, 1989. 71

[47] M.A. Fischler and R.C. Bolles. Perceptual organization and curve partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):100–105, Jan. 1986. 22

[48] Margaret Fleck. Perspective projection: The wrong imaging model. Technical Report technical report 95-01, Computer Science, University of Iowa, 1995. 90

[49] D. Folio and V. Cadenat. A sensor-based controller able to treat total image loss and to guarantee non-collision during a vision-based navigation task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2008.*, pages 3052–3057, Sept. 69, 78

[50] D. Folio and V. Cadenat. A controller to avoid both occlusions and obstacles during a vision-based navigation task in a cluttered environment. In *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05*, pages 3898–3903, 2005. 10, 57, 58, 68

[51] D. Folio and V. Cadenat. A redundancy-based scheme to perform safe vision-based tasks amidst obstacles. In *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pages 13–18, 2006. 68

[52] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, 1997. 41

[53] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(9):891 –906, sep 1991. 19

[54] Jerome H. Friedman, Jon Louis Bentley, and Raphael A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977. 20, 21

[55] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithms for computing k-nearest neighbors. *IEEE Trans. Computers*, 24(7):750–753, 1975. 20, 21

[56] Jacques Gangloff, Michel de Mathelin, and Gabriel Abba. 6 dof high speed dynamic visual servoing using gpc controllers. In *ICRA*, pages 2008–2013, 1998. 67

[57] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, and C. Perez-Vidal. Continuous visual servoing despite the changes of visibility in image features. *Robotics, IEEE Transactions on*, 21(6):1214–1220, 2005. 68

[58] Héctor H. González-Baños and Jean-Claude Latombe. Navigation Strategies for Exploring Indoor Environments. 40

[59] Karl Granström, Christian Lundquist, and Umut Orguner. Tracking rectangular and elliptical extended targets using laser measurements. In *FUSION*, pages 1–8, 2011. 40

[60] Jens-Steffen Gutmann, Thilo Weigel, and Bernhard Nebel. A fast, accurate and robust method for self-localization in polygonal environments using laser range finders. *Advanced Robotics*, 14(8):651–667, 2001. 40

[61] J.H. Han. Detection of convex and concave discontinuous points in a plane curve. In *Proceedings, Third International Conference on Computer Vision*, pages 71–74, 1990. 8, 50

[62] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, Aug. 1988. 15, 16, 33

[63] K. Hashimoto, T. Kimoto, T. Ebine, and Hidenori Kimura. Manipulator control with image-based visual servo. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2267–2271 vol.3, 1991. 67

[64] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *ECCV (2)*, pages 759–773, 2012. 16

[65] Thomas S. Huang and Olivier D. Faugeras. Some properties of the e matrix in two-view motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(12):1310–1312, 1989. 67

[66] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, 1996. 65

[67] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506 – II–513 Vol.2, june-2 july 2004. 19

[68] Wisama Khalil and Etienne Dombre. *Modeling, Identification And Control of Robots*. Penton, 2002. 119

[69] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 2, pages 500 – 505, mar 1985. 37, 41, 61

[70] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. 33

[71] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987. 19

[72] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404 vol.2, 1991. 41

[73] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *Robotics, IEEE Transactions on*, 20(6):967–977, 2004. 38

[74] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991. 37

[75] S. M. Lavalle and J. J. Kuffner. Rapidly-Exploring Random Trees: Progress and Prospects. pages 293–308. 2001. 42

[76] Kennard R Laviers and Gilbert L Peterson. Cognitive robot mapping with polylines and an absolute space representation. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3771–3776, 2004. 40

[77] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–319 – II–324 vol.2, june 2003. 19

[78] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 21, 33

[79] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011. 15, 19

[80] Jing Li and Nigel M. Allinson. A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10-12):1771–1787, 2008. 18

[81] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993. 15, 17

[82] Tony Lindeberg. Direct estimation of affine image deformations using visual front-end operations with automatic scale selection. In *ICCV*, pages 134–141, 1995. 15

[83] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998. 15

[84] Tony Lindeberg and Jonas Gårding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image Vision Comput.*, 15(6):415–434, 1997. 15

[85] Gonzalo López-Nicolás, Nicholas R. Gans, Sourabh Bhattacharya, Carlos Sagüés, José Jesús Guerrero, and Seth Hutchinson. Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 40(4):1115–1127, 2010. 68

[86] Gonzalo López-Nicolás and Carlos Sagüés. Vision-based exponential stabilization of mobile robots. *Auton. Robots*, 30(3):293–306, 2011. 67

[87] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artif. Intell.*, 31(3):355–395, 1987. 82

[88] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. viii, 15, 18, 19, 20

[89] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 –1157 vol.2, 1999. 22, 33

[90] T. Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, C-32(2):108–120, 1983. 39

[91] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981. 33

[92] V. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man and Cybernetics*, 20(5):1058–1069, 1990. 55

[93] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430, 1987. 38

[94] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. 14

[95] Raj Madhavan. Terrain aided localization of autonomous vehicles. *NIST SPECIAL PUBLICATION SP*, pages 511–518, 2003. 40

[96] E. Malis, F. Chaumette, and S. Boudet. Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1352–1359 vol.2, 1998. 67

[97] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 d visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2):238–250, April 1999. 76

[98] G.L. Mariottini, G. Oriolo, and D. Prattichizzo. Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. *Robotics, IEEE Transactions on*, 23(1):87–100, 2007. 67, 100

[99] Philippe Martinet, Nadine Daucher, Jean Gallice, and Michel Dhome. Robot control using monocular pose estimation. *Worshop on New Trends in Image-based Robot Servoing, IROS*, 97:1–12, 1997. 67

[100] Y. Masutani, M. Mikawa, N. Maru, and F. Miyazaki. Visual servoing for non-holonomic mobile robots. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 2, pages 1133–1140 vol.2, 1994. 67

[101] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Comput.*, 22(10):761–767, 2004. 14

[102] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *Robotics and Automation, IEEE Transactions on*, 18(4):534–549, 2002. 68, 71

[103] Youcef Mezouar and François Chaumette. Avoiding self-occlusions and preserving visibility by path planning in the image. *Robotics and Autonomous Systems*, 41(2-3):77–87, 2002. 68

[104] Youcef Mezouar and François Chaumette. Optimal camera trajectory with image-based control. *I. J. Robotic Res.*, 22(10-11):781–804, 2003. 68, 71

[105] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615 –1630, oct. 2005. 18, 19

[106] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65:43–72, 2005. 3, 15, 23

[107] Krystian Mikolajczyk and Cordelia Schmid. Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60:63–86, 2004. 15

[108] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *ICPR*, pages 2681–2684, 2012. 16

[109] J.S. Milton and J.C. Arnold. *Introduction to probability and statistics: principles and applications for engineering and the computing sciences*. McGraw-Hill, 2003. 50

[110] J. Minguez and L. Montano. Nearness diagram navigation (nd): a new real time collision avoidance approach. In *Proceedings. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000. (IROS 2000)*, volume 3, pages 2094–2100 vol.3, 2000. 42

[111] Javier Minguez, Florent Lamiraux, and Jean-Paul Laumond. Motion planning and obstacle avoidance. In *Springer Handbook of Robotics*, pages 827–852. 2008. 39

[112] H. Mittelstaedt and M.-L. Mittelstaedt. Homing by path integration. In Floriano Papi and HansG. Wallraff, editors, *Avian Navigation*, Proceedings in Life Sciences, pages 290–297. Springer Berlin Heidelberg, 1982. 77

[113] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116 – 121, mar 1985. 39

[114] Hans P Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988. 39, 40

[115] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988. 39

[116] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340. INSTICC Press, 2009. 21

[117] Marius Muja and David G. Lowe. Fast matching of binary features. In *Conference on Computer and Robot Vision (CRV)*, pages 404–410, 2012. 21

[118] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Auton. Robots*, 23(2):97–111, 2007. 40, 48

[119] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, pages I–652 – I–659 Vol.1, june-2 july 2004. 33

[120] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th ICPR International Conference on*, pages 582 –585, 1994. 19

[121] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002. 19

[122] Raphael Ortiz. FREAK: Fast Retina Keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 19

[123] Y. Petillot, I. Tena Ruiz, and D.M. Lane. Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar. *Oceanic Engineering, IEEE Journal of*, 26(2):240–251, Apr 2001. 40

[124] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Auton. Robots*, 26(2-3):123–139, 2009. 40

[125] R. Pissard-Gibollet and P. Rives. Applying visual servoing techniques to control a mobile hand-eye system. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 166 –171 vol.1, 1995. 68

[126] Philip Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *ICCV*, pages 754–760, 1998. 14

[127] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807 vol.2, 1993. 38

[128] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. 48

[129] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *Robotics and Automation, IEEE Transactions on*, 8(5):501–518, 1992. 61

[130] P.L. Rosin. Techniques for assessing polygonal approximations of curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):659–666, 1997. 7, 47

[131] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. 15

[132] Edward Rosten, Reid Porter, and Tom Drummond. FASTER and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. 33

[133] Eric Royer, Maxime Lhuillier, Michel Dhome, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007. 67

[134] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, pages 2564–2571, 2011. 15, 19, 21

[135] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. 21

[136] C. Samson. Path following and time-varying feedback stabilization of a wheeled mobile robot. In *Int. Conf. ICARCV'92*, 1992. 10, 55, 56, 86

[137] C. Samson, M. Leborgne, and B. Espiau. Robot control. the task-function approach. In *Oxford Engineering Science Series, vol. 22*. Oxford University Press, 1991. 56

[138] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do i organize my holiday snaps?. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2350, pages 414–431. 2002. 19

[139] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):530–535, 1997. 14

[140] Stephen Se, David G. Lowe, and James J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *I. J. Robotic Res.*, 21(8):735–760, 2002. 14

[141] Sinisa Segvic, Anthony Remazeilles, Albert Diosi, and François Chaumette. A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172–187, 2009. 67

[142] L.G. Shapiro and R.M. Haralick. Decomposition of two-dimensional shapes by graph-theoretic clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):10–20, Jan. 1979. 7, 47

[143] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR '94*, pages 593–600, 1994. 15, 33

[144] Ali Siadat, Axel Kaske, Siegfried Klausmann, Michel Dufaut, and René Husson. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In *3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, pages 153–158, 1997. 7, 44

[145] Ali Siadat, Axel Kaske, Siegfried Klausmann, Michel Dufaut, and René Husson. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In *3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, pages 153–158, 1997. 40

[146] R. Siegwart, I.R. Nourbakhsh, and D. Scaramuzza, editors. *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents. The MIT Press Cambridge, Massachusetts, London, England, 2004. 39

[147] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008. 21

[148] Thierry Siméon, Stéphane Leroy, and Jean-Paul Laumond. Path coordination for multiple mobile robots: a resolution-complete algorithm. *IEEE T. Robotics and Automation*, 18(1):42–49, 2002. 40

[149] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings., 1996 IEEE International Conference on Robotics and Automation, 1996*, volume 4, pages 3375–3382, 1996. 41

[150] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003. 14, 19

[151] Benoit Thuilot, Philippe Martinet, Lionel Cordesses, and Jean Gallice. Position based visual servoing: Keeping the object in the field of vision. In *ICRA*, pages 1624–1629, 2002. 68

[152] DimitrisP. Tsakiris, Patrick Rives, and Claude Samson. Extending visual servoing techniques to nonholonomic mobile robots. In DavidJ. Kriegman, GregoryD. Hager, and A.Stephen Morse, editors, *The confluence of vision and control*, volume 237 of *Lecture Notes in Control and Information Sciences*, pages 106–117. Springer London, 1998. 67

[153] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and trends in computer graphics and vision*, 3(3):177–280, 2008. viii, 14, 17

[154] Tinne Tuytelaars, Luc J. Van Gool, L. D'haene, and Reinhard Koch. Matching of affinely invariant regions for visual servoing. In *ICRA*, pages 1601–1606, 1999. 14

[155] Tinne Tuytelaars and Luc Van Gool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *Proceedings of the British Machine Vision Conference*, pages 412–425, 2000. 14

[156] I. Ulrich and J. Borenstein. Vfh+: reliable obstacle avoidance for fast mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1572 –1577 vol.2, may 1998. 41

[157] I. Ulrich and J. Borenstein. Vfh*: local obstacle avoidance with look-ahead verification. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2505 –2511 vol.3, 2000. 41

[158] Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596, 2010. 18

[159] Luc Van Gool, Theo Moons, and Dorin Ungureanu. Affine / photometric invariants for planar intensity patterns. In *Computer Vision ECCV '96*, volume 1064, pages 642–651. 1996. 19

[160] A. Vatavu, S. Nedevschi, and F. Oniga. Real-time environment representation based on occupancy grid temporal analysis using a dense stereo-vision system. In *Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on*, pages 203–209, 2010. 40

[161] Trung-Dung Vu and Olivier Aycard. Laser-based detection and tracking moving objects using data-driven markov chain monte carlo. In *ICRA*, pages 3800–3806, 2009. 40

[162] William J. Wilson, C.C. Williams Hulls, and G.S. Bell. Relative end-effector control using cartesian position based visual servoing. *Robotics and Automation, IEEE Transactions on*, 12(5):684–696, 1996. 67

[163] Xin Yang and Kwang-Ting Cheng. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In *ISMAR*, pages 49–57, 2012. 19

[164] C. Lawrence Zitnick. Binary Coherent Edge Descriptors. In *European Conference on Computer Vision*, pages 1–14, 2010. 21