



**HAL**  
open science

# Mine First to See Better: Constraint-based Data Mining for Computer Vision Applications

Elisa Fromont

► **To cite this version:**

Elisa Fromont. Mine First to See Better: Constraint-based Data Mining for Computer Vision Applications: Habilitation à Diriger des Recherches . Machine Learning [cs.LG]. Université Jean Monnet, Saint-Etienne, 2015. tel-01408021

**HAL Id: tel-01408021**

**<https://hal.science/tel-01408021>**

Submitted on 2 Dec 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Habilitation à Diriger des Recherches

Section CNU : 27 - Informatique

## Mine First to See Better: Constraint-based Data Mining for Computer Vision Applications

présentée par  
**Elisa Fromont**

Maître de Conférences  
Université Jean Monnet de Saint-Etienne  
Laboratoire Hubert Curien UMR CNRS 5516

Soutenue le 11 Décembre 2015 devant la commission d'examen composée de :

Luc De Raedt	Prof. Katholieke Universiteit Leuven (Belgique)	Rapporteur
Pascale Kuntz-Cosperec	Prof. École Polytechnique, Université de Nantes	Examinatrice
Katharina Morik	Prof. TU Dortmund (Allemagne)	Rapporteuse
Marc Sebban	Prof. Université Jean Monnet, Saint-Etienne	Examineur
Arno Siebes	Prof. Universiteit Utrecht (Pays Bas)	Rapporteur
Alexandre Termier	Prof. Université de Rennes 1	Examineur



# Acknowledgments

I would like to thank the members of my jury, Professor Luc De Raedt, Professor Pascale Kuntz-Cosperec, Professor Katharina Morik, Professor Marc Sebban (who is my referee for this "habilitation"), Professor Arno Siebes and Professor Alexandre Termier for taking time to thoroughly engage with my work and provide valuable feedback.

In particular, I am grateful to Pascale Kuntz-Cosperec and Alexandre Termier for coming all the way from Nantes and Rennes to evaluate my work. Katharina, Arno, Luc: all of you have been special in my life. Katharina is for me a very lively and enthusiastic person who introduced me to wonderful persons in many conferences, whose constant fight to make sure that women are never forgotten in our male-dominated computer science community is, in my opinion, essential and has impacted my life at many levels over the years. Arno has crossed my path at every steps of my journey in the last 10 years: he reviewed the European project I worked in when I was in Leuven, I was contaminated by his enthusiasm during his MDL crusade, he introduced me to the IDA community (that ended up with me organizing the conference in Saint-Etienne) and even visited us in Saint-Etienne to talk about his new research passion. As always, it will take me some years to understand how inspiring and deep your current work is but I am already convinced that it will have an impact on my research life as strong as everything else you did in the past years. Luc has been in my thoughts during all my PhD on Inductive Logic Programming (one of the many subjects he fathered). He made sure that I was well integrated into his team in Leuven and allowed me to expand my culture in Machine Learning and Data mining immensely compared to the young padawan I was when I washed up on Leuven's shore in 2006. Thanks to him and to Hendrik (who will get some special thanks later) I was able to work in a wonderful environment with incredibly brilliant people. Without my stay in Leuven, none of the career opportunities that I had afterwards would have existed.

I'll thus start by thanking my colleagues in Leuven. The first one should be Maurice Bruynooghe, head of the DTAI group in Leuven (and former supervisor of dozens of well known researchers in Machine Learning including Luc De Raedt and Hendrik Blockeel) where I did my post-doc from February 2006 until August 2008. Maurice has trusted me from the beginning and even hosted me in his own house for weeks when I arrived in Belgium. He is one of the wisest person I know and I guess I would not have felt so comfortable in this new world without him and his family. Hendrik Blockeel is another very special person in my research life. After meeting me in few conferences and accepting to be in my PhD jury, Hendrik also took me as a post doc in his research group. He is not only a very brilliant colleague but, after sharing many adventures which involve bikes, walks, dancing, eating and playing, he has also become a dear friend over the years. I won't mention all the other protagonists in those adventures but they can rest assured that they will not be forgotten (and I actually make sure that some of them e.g. Kurt and Fabian do not forget me either by invading their home regularly!). In

Leuven, I was involved in a Belgium national project and a European one which brought me to work with many interesting people such as Siegfried Nijssen, Bart Goethals, Toon Calders, Adriana Prado who all become important persons in my life. I thank all of you for everything you have brought me over the years, your kindness, your brightness and your time.

In September 2008, I joined the Machine Learning team at the Hubert Curien Laboratory in Saint-Etienne. Again, as always in my life, somebody had to see through my insecurity, my doubts and my strange sense of humour and trust me enough to give me a position. This time, the key person was Colin de la Higuera. Of course I will never thank him enough for everything he had made possible by giving me his trust. He allowed me to meet incredible people with whom I have been working over the last seven years. I am not going to thank everybody although many persons, students and professors, in the team, in the lab and in our partner teams (in particular at LIRIS), would deserve a particular chapter. I'll just give a special thank to Pierre, Baptiste, Rémi and Leonor who had to share an office with me and put up with my constant babbling, my very loud (and prohibited) Skype sessions, my recurring technical problems, and my never satisfied gluttony. I cannot forget Marc Sebban, my mentor and my friend, who can understand my most incomprehensible thoughts or my very sparse sentences, who can spend hours answering my very naive questions without looking annoyed and who miraculously knows how to call out the best in me. Marc, I am not sure that I would have done anything good without you the last past years so thank you a million times for everything.

At the end of my PhD, I moved from my home town, Rennes, to discover other horizons. By moving, I have weakened the bounds that tied me to my family and many of my friends. However, all my family (and in particular my parents and my brother) and many of my friends have kept visiting me (from Rennes, Nantes, Bordeaux, Paris, Lyon, Grenoble, Toulouse, Lille and Nancy) over the last years and made it possible for me to enjoy my research environment while being so far away from them. Thank you all for your presence, your love and your joy.

To conclude, I would like to thank Olivier, my first and probably only fan, who has never stopped believing in me and my skills even when I was really doubtful about them myself. You are the co-author of my best work so far: my two wonderful children. They are currently too young to understand any of what I am presenting in this document but they take a huge part of my life and strongly contribute to my personal equilibrium. Thank you.

Elisa

# Contents

<b>Acknowledgments</b>	<b>iii</b>
<b>(In French) Résumé étendu</b>	<b>v</b>
I Début . . . . .	v
II Post-doctorat . . . . .	v
III MdC à l'UJM . . . . .	vii
<b>My Publications</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Constraint-based Data Mining for Inductive Databases</b>	<b>6</b>
1 Mining Views . . . . .	6
1.1 Representing Models as Sets of Concepts . . . . .	8
1.1.1 Itemsets and Association Rules . . . . .	8
1.1.2 Decision Trees . . . . .	8
1.1.3 Clustering . . . . .	10
1.2 Discussion . . . . .	11
2 Induction of Optimal Decision Trees . . . . .	12
2.1 Relationships between Itemsets and Decision Trees . . . . .	14
2.1.1 Itemsets . . . . .	14
2.1.2 Decision trees . . . . .	15
2.1.3 Link between Decision Trees and Itemsets . . . . .	15
2.2 Constraints on Decision Trees . . . . .	17
2.2.1 Properties of Constraints and Criteria . . . . .	17
2.3 Building Optimal Decision Trees from Lattices . . . . .	20
2.3.1 Decision Trees from Lattices . . . . .	20
2.3.2 Computing Lattices Beforehand . . . . .	22
2.3.3 Computing Lattices on the Fly . . . . .	24
2.4 Discussion . . . . .	25
3 (Dynamic) Plane Graph Mining . . . . .	26
3.1 Definitions . . . . .	26
3.1.1 Plane Graphs . . . . .	26
3.1.2 Isomorphism and Subgraph Isomorphism . . . . .	28
3.1.3 Support and Frequency of a Subgraph Pattern . . . . .	29
3.1.4 Dynamic Plane Graph . . . . .	29
3.1.5 Occurrence Graph and Spatio-Temporal Patterns . . . . .	30
3.2 Mining Spatio-Temporal Patterns . . . . .	33
3.2.1 Extensions . . . . .	34

3.2.2	Graph Codes . . . . .	35
3.2.3	Code Search Space and Canonical Codes . . . . .	36
3.2.4	Algorithms . . . . .	37
3.3	Experiments . . . . .	41
4	Conclusion . . . . .	41
<b>3</b>	<b>Data mining for BOW-based Image classification</b>	<b>44</b>
1	Supervised Learning of Gaussian Mixture Models for Better BOW . . . . .	44
1.1	Notations and Definitions . . . . .	45
1.2	Supervised GM-based Dictionary Learning . . . . .	46
1.2.1	Intuitive Idea . . . . .	46
1.2.2	Joint Optimization of the Likelihood and the Purity . . . . .	47
1.2.3	EM-based Learning Algorithm . . . . .	50
2	Pattern Mining in BOW . . . . .	50
2.1	Frequent Local Histogram (FLH) Mining . . . . .	51
2.2	Finding the Best FLHs for Image Classification . . . . .	53
2.3	Kernel Function for Effective Pattern Classification . . . . .	55
2.4	GRID-FLH: Incorporating Global Spatial Information to FLH . . . . .	55
3	Experiments . . . . .	55
3.1	Image Datasets and Data Preparation . . . . .	56
3.2	Experimental Results for the GMM Approach . . . . .	57
3.2.1	Other Approaches . . . . .	57
3.2.2	Pre-processing and Setting . . . . .	58
3.2.3	Results . . . . .	58
3.3	Experimental Results for the FLH-based Approach . . . . .	61
3.3.1	Comparison with Non-mining Methods . . . . .	62
3.3.2	Comparison with State-of-the-art Methods . . . . .	62
4	Conclusion . . . . .	64
<b>4</b>	<b>Data mining for Object Tracking in Videos</b>	<b>66</b>
1	Images and Videos as Graphs . . . . .	66
1.1	RAG and Triangulation . . . . .	67
1.2	Video Datasets . . . . .	67
2	Object Tracking Using Graph Mining . . . . .	70
2.1	Tracking with Patterns . . . . .	70
2.1.1	Spatio Temporal Path . . . . .	70
2.1.2	Clusters of Spatio-Temporal Patterns . . . . .	71
2.2	Meaningfulness of the (Spatio-Temporal) Patterns . . . . .	74
2.2.1	Output of PLAGRAM (plane graph patterns) . . . . .	75
2.2.2	Output of DYPLAGRAM and DYPLAGRAM_ST . . . . .	76
2.3	Spatio-Temporal Paths for Object Tracking . . . . .	79
2.4	Clusters of Spatio-Temporal Patterns for Tracking . . . . .	81
2.4.1	Experimental Design . . . . .	81
2.4.2	Results . . . . .	82
3	Conclusion . . . . .	87
<b>5</b>	<b>Conclusion</b>	<b>88</b>
	<b>Bibliography</b>	<b>97</b>

# (In French) Résumé étendu

## I Début

Après un DEA en Informatique option *Génie Logiciel et Méthodes Formelles* en 2002 à l'Université de Rennes 1, mes travaux de recherche se sont (ré)orientés en doctorat vers un domaine de l'informatique tout autre et lié à l'intelligence artificielle : celui de l'apprentissage automatique. Dans le cadre du projet RNTS CEPICA (*Conception et Évaluation d'une Prothèse Implantable Cardiaque*), en collaboration avec le LTSI (*Laboratoire de Traitement du Signal et de l'Image de L'Université de Rennes1*) et le CHU de Rennes, l'objectif principal de ma thèse (nov. 2002 - déc. 2005) était d'évaluer l'impact de l'apport de nouvelles sources de données sur la caractérisation et la reconnaissance des arythmies cardiaques. Ces travaux ont donné lieu aux publications [14, 13, 4, 6, 12], [30, 31, 32, 33, 38] qui ne feront pas l'objet d'un développement dans ce manuscrit.

## II Post-doctorat

De Février 2006 à Août 2008, j'ai effectué un post-doc à l'Université Catholique de Leuven en Belgique sur un projet lié aux *bases de données inductives* (BDI). Ce post-doc m'a permis d'étendre ma culture en apprentissage automatique et de me spécialiser dans le domaine de la fouille de données.

Le cadre des BDI, introduit par Imielinski et Mannila en 1996, propose de considérer l'extraction de connaissances comme un processus d'interrogation interactif au sens des bases de données. Les requêtes portent alors sur les données (par exemple pour sélectionner un contexte de fouille), sur des abstractions ou généralisations des données, sur des modèles (par exemple pour sélectionner des règles descriptives intéressantes) ou sur les deux composantes (par exemple pour identifier des objets qui satisfont certaines règles). Ces requêtes dites "inductives" sont ainsi utilisées pour générer (fouiller) ou appliquer les modèles induits sur les données.

Ce travail a été effectué en collaboration avec l'Université d'Anvers et en particulier, Bart Goethals, Toon Calders (qui travaille maintenant à l'Université libre de Bruxelles) et Adriana Prado (qui travaille maintenant à l'Université Fédérale Fluminense au Brésil). Ils ont proposé en 2006 une approche relativement intuitive permettant de stocker et de manipuler des objets particuliers issus d'une fouille de motifs que sont les itemsets fréquents et les règles d'associations. Cette approche se base sur les bases de données relationnelles standards et le langage SQL. Du point de vue de l'utilisateur, des tables contenant tous les itemsets et toutes les règles d'association pouvant être extraites d'un ensemble de données sont à disposition pour effectuer toutes sortes de requêtes. La manière dont ces tables sont remplies et les algorithmes de fouille de données utilisés pour les remplir sont transparents pour l'utilisateur. Nous avons étudié



cette approche et ses avantages pour apprendre et manipuler des modèles “globaux” tels que les arbres de décision ou les règles de prédiction, les algorithmes de clustering ou les réseaux Bayésiens. Par exemple, pour découvrir des règles d’association, l’utilisateur doit imposer des contraintes sur les règles qu’il recherche (typiquement dans ce cadre, leur support et leur confiance) et l’algorithme fournit un ensemble de règles qui remplissent ces contraintes. Les systèmes d’apprentissage d’arbres de décision standards (ou de clustering) ne permettent d’obtenir qu’un seul arbre de décision à la fois et les contraintes imposables sur l’arbre lui-même sont, en général, limitées au nombre minimum d’exemples “couverts” par chaque feuille de l’arbre. L’arbre appris n’est pas le plus précis construit à partir de l’ensemble des données d’apprentissage ni le plus petit ni celui qui permet d’obtenir la meilleure généralisation en suivant certains critères ; les algorithmes d’apprentissage utilisant des heuristiques permettent simplement d’obtenir des arbres relativement petits avec une précision relativement bonne. En intégrant l’apprentissage d’arbres de décision (de “clustering” ou de réseaux Bayésiens) sous contraintes dans les bases de données inductives, nous cherchions à fournir une approche aussi précise que peut être une approche liée à la recherche de règles d’association : l’utilisateur spécifie quel(s) type(s) d’arbre(s) il recherche, et le système recherche ces arbres efficacement.

Pour répondre à ces objectifs, j’ai suivi deux axes principaux : la création avec l’Université d’Anvers d’un premier prototype de bases de données inductives permettant à la fois de faire des requêtes sur des motifs locaux mais aussi sur des motifs globaux tels que les arbres de décision [2, 3] [23, 24, 26, 25]. Ce prototype permet d’extraire depuis les requêtes, des contraintes sur la taille de l’arbre et sur sa précision minimale en apprentissage. Pour pouvoir répondre à des requêtes recherchant tous les arbres répondants à des critères donnés, nous avons développé un algorithme d’apprentissage permettant une recherche exhaustive d’arbres sous contraintes. Les problèmes de complexité d’une telle recherche nous ont conduit à développer un algorithme efficace permettant de générer des arbres optimaux (selon les critères spécifiés par l’utilisateur) sous contraintes [18, 19]. Les contraintes utilisées concernent le nombre minimum d’exemples dans chaque feuille de l’arbre, la taille, la précision et la profondeur de l’arbre, mais aussi le coût de l’arbre (en terme d’erreur en apprentissage et de coût des tests utilisés dans l’arbre). Notre algorithme repose essentiellement sur la relation existant entre les contraintes applicables aux arbres de décision et celles applicables aux itemsets. Nous proposons d’exploiter des treillis d’itemsets pour extraire des arbres de décisions optimaux en temps linéaire et nous avons développé différentes stratégies permettant de construire ces treillis efficacement en tirant partie des années de recherche sur l’extraction d’itemsets fréquents. Lorsque les contraintes jouent un rôle crucial dans l’application, nous avons montré que notre algorithme obtient de meilleurs résultats que des algorithmes dédiés, par exemple pour l’apprentissage d’arbres de décision sensibles aux coût (*cost-sensitive decision trees*). En outre, c’est un outil idéal dans le cadre des bases de données inductives puisqu’il garantit une réponse exacte à un problème donné. Nous avons également proposé dans [15] un algorithme qui répond au premier objectif dans le cas du “clustering”. Ces travaux sont développés dans le Chapitre 2 de ce manuscrit.

Lors de ce post-doc en Belgique, j’ai travaillé dans le contexte de projets nationaux (FWO) et dans le cadre du projet Européen FP6-IST IQ (*Inductive Queries for Mining Patterns and Models*). Ce contexte international m’a permis de tisser des liens forts avec certaines universités européennes qui ont perduré jusqu’à aujourd’hui : ils continuent de donner lieu à des visites (bilatérales) entre partenaires et des collaborations fructueuses.

### III MdC à l'UJM

J'entame en Septembre 2015 ma 7ème année en tant que Maître de Conférences à l'Université Jean Monnet. Ces sept années ont été ponctuées par la naissance de mes deux enfants. Ma première année (2008-2009) dans l'équipe "Machine Learning" du Laboratoire Hubert Curien (LaHC) a été marquée par une forte volonté d'effectuer une reconversion thématique vers des sujets importants pour le département "Informatique, Image, Telecom" du LaHC liés à l'analyse d'images et de vidéos. J'ai pu entamer des collaborations très fructueuses non seulement avec des membres de ma propre thématique mais aussi avec des membres de la thématique "image" du même département. Ces collaborations ont été initiées dans le cadre du projet ANR SATTIC au cours duquel les membres du département image nous ont présenté des problématiques de structuration et de sélection des descripteurs bas niveaux décrivant les images et les vidéos qui pouvaient nécessiter l'utilisation et le développement de nouvelles méthodes d'apprentissage automatique ou de fouille de données (i.e. des méthodes non supervisées).

**Sac de mots visuels** La représentation par sac de mots est une description de l'image initialement développée dans le contexte d'analyse de documents textuels. Cette représentation se base sur le fait que l'ensemble des descripteurs "bas niveau" (par exemple un vecteur encodant les 3 signaux RGB d'un pixel dans une image) peut être décrit de manière abstraite au moyen d'un dictionnaire de "mots visuels". Une image peut être ainsi représentée par un histogramme (i.e un vecteur) des occurrences des mots qui la composent : pour une image donnée, chaque mot se voit affecter le nombre de fois qu'il apparaît dans l'image. Ce vecteur de représentation, potentiellement parcimonieux (ou clairsemé), a donc la même taille que le dictionnaire. La  $i$ -ème composante du vecteur indique le nombre d'occurrences du  $i$ -ème mot du dictionnaire dans l'image. La constitution du dictionnaire est une étape critique pour les performances des systèmes utilisant une telle représentation. Elle consiste en un "clustering" de l'ensemble de tous les descripteurs décrivant toutes les images du corpus en un nombre fixe de clusters égal à la taille du vocabulaire désiré. Le centroïde de chaque cluster ainsi obtenu devient un mot visuel. Chacun des attributs initiaux est alors comparé à tous les mots visuels obtenus et on leur associe le mot visuel dont il est le plus proche (selon une distance préétablie). Une image devient donc décrite par son histogramme de mots visuels ou, son sac de mots. L'utilisation des mots visuels a été popularisée en 2004 [64] et leur construction se fait principalement à partir de descripteurs SIFT à 128 dimensions. En analyse d'images et dans les meilleures conférences du domaine, des centaines d'articles ont vu le jour cette dernière décennie pour améliorer l'utilisation de ces mots visuels. J'ai pris part à ce domaine très actif en me consacrant plus particulièrement à la découverte de représentations de "moyens niveaux" (c'est à dire des représentations basées soit directement sur les pixels, soit sur une description bas niveau issue du traitement du signal) plus discriminantes qui permettent d'améliorer significativement l'état de l'art en analyse d'images et de vidéos [11, 16, 21, 10, 8, 9, 1][36, 40, 29].

**Contributions** Nous avons dans un premier temps proposé une méthode qui permet de représenter les images sous forme de séquences pondérées de mots visuels et un nouvel algorithme de distance d'édition sur ces séquences pondérées [1]. Le processus de construction des séquences choisi nous semblant trop arbitraire pour réellement bénéficier à une chaîne de traitement systématique de l'image, nous nous sommes ensuite focalisé sur la création des sacs de mots et en particulier sur un des aspects clé de leur création:

la méthode de clustering utilisée. Dans le contexte de l'image, le nombre de bases de données (étiquetées) disponibles est très important. Nous avons donc mis au point des nouveaux algorithmes de clustering qui puissent prendre en compte une information supervisée partielle [9][29, 41]. L'article paru dans le journal "Pattern recognition" [9] décrit ainsi une méthode permettant d'intégrer un terme supervisé à un algorithme d'optimisation qui apprend les paramètres d'un mélange de Gaussiennes. Ce mélange permet de construire un clustering flou ("soft-clustering") où chaque descripteur de l'image est assigné (de manière pondérée) à un ensemble de clusters de telle sorte que des descripteurs appartenant à des images d'une classe donnée soient plus susceptibles de se retrouver dans un même cluster.

Nous avons ensuite exploré la possibilité d'étendre des algorithmes de régression logistique au cadre multi-classe pour effectuer une sélection automatique des mots visuels les plus discriminants pour un problème de classification donné [8]. Ces travaux nous ont également permis d'établir des liens entre les noyaux marginalisés et les probabilités en sortie d'un algorithme de régression logistique. Les résultats obtenus en classification se sont révélés meilleurs que ceux des méthodes de l'état de l'art permettant de fusionner plusieurs types de dictionnaires de mots visuels, et ce pour plusieurs benchmarks connus en classification d'images.

En parallèle de ces recherches sur l'apprentissage supervisé, nous avons exploré l'utilisation de méthodes totalement non supervisées comme la fouille de motifs fréquents pour l'amélioration du traitement de l'image (et en particulier la classification) [10, 11] et de la vidéo (et en particulier le "tracking" d'objets dans les vidéos) [20, 7, 21][28, 37, 40].

Nous avons montré que la recherche de motifs récurrents dans les sacs de mots et la sélection intelligente de ces motifs dans le contexte de la classification d'images pouvaient améliorer significativement l'état de l'art en classification d'images sur la plupart des benchmarks connus [10, 11]. Notons que ces benchmarks n'incluent pas ceux qui contiennent plusieurs millions d'images tel qu'IMAGENET<sup>1</sup>, qui sont traités, de nos jours, principalement à partir de méthodes basées sur l'apprentissage de réseaux de neurones [10, 11] pour des raisons qui seront développées dans la conclusion de ce document. Ces contributions en sélection a posteriori des motifs calculés exhaustivement sont non seulement utiles pour le domaine de la vision par ordinateur mais également pertinentes pour le domaine de la fouille de motifs et en particulier en découverte de sous groupes discriminants.

Ces sept dernières années à Saint-Etienne m'ont également permis d'obtenir une bonne expérience de l'encadrement doctoral. J'ai co-encadré avec mes collègues Baptiste Jeudy et François Jacquenet deux doctorats sur l'analyse de vidéos à partir de méthodes basées sur la fouille de motifs. La première, celle de Fabien Diot, concernait la fouille de graphes pour le suivi d'objets dans les vidéos. Elle a été soutenue le 3 Juin 2014. Cette thèse CIFRE avec Alcatel-Lucent Bell-Labs a donné lieu au dépôt de 2 brevets. Nous avons pu montrer tout au long de cette thèse l'intérêt des motifs fréquents (et en particulier, dans ce cas, des sous-graphes fréquents) trouvés de manière non supervisée pour suivre des objets dans les vidéos dans des contextes généralistes où l'objet d'intérêt n'est pas connu a priori et peu importe quelle forme [20, 7][28, 37]. La deuxième thèse, celle de Hoang-Tung Tran s'intéressait à la correction automatique de "tags" dans les vidéos déposées sur des sites de partage tel que Youtube. Elle s'est intéressée à l'utilisation de la seule information visuelle pour la comparaison automatique des vidéos (en utilisant des motifs fréquents) et à la propagation automatique de

---

<sup>1</sup><http://www.image-net.org/>

tags sur des vidéos au contenu visuel similaire [21][40]. Cette thèse a été soutenue le 17 Juillet 2014.

J'ai également collaboré avec des collègues Belges rencontrés lors de mon post-doctorat, sur l'utilisation de la fouille de motifs (tels que des ensembles faiblement structurés) dans des flots de données pour l'analyse de vidéos [16][27, 36]. Ces résultats, par la proposition de nouveaux algorithmes efficaces, ont contribué à l'état de l'art en fouille sous contraintes dans les flux de données et offrent à terme la possibilité d'analyser les vidéos en temps réel (ce qui n'était pas possible avec les algorithmes proposés précédemment).

Enfin, depuis l'acceptation du projet ANR Solstice que je coordonne depuis Février 2014, je co-encadre deux thèses sur la découverte (supervisée ou non) de représentations discriminantes pour l'analyse d'images et de vidéos. La thèse de Romain Deville (en co-encadrement avec Baptiste Jeudy et Christine Solnon du Liris) fait suite à la thèse de Fabien Diot sur l'utilisation de la fouille de sous-graphes fréquents pour l'analyse de vidéos. Elle a fait tout d'abord l'objet d'un stage de Master 2 dans un contexte d'images et a fourni des résultats prometteurs pour l'utilisation de sacs de sous-graphes géométriques pour la description pertinente des images. La thèse de Damien Fourure est orientée vers une technique d'apprentissage automatique particulièrement populaire avec l'avènement du centre Facebook Research : le "deep learning". Le sujet porte sur l'apprentissage de représentations 3D et temporelles pour la segmentation de scènes extérieures [34, 35, 39]. Le deep learning est une des méthodes actuelles qui obtient les meilleurs résultats pour résoudre des problèmes de classification dans le domaine de la vision par ordinateur mais également dans le domaine de la reconnaissance de la parole ou du traitement de la langue naturelle. L'ambition de cette thèse est de contribuer à ce domaine mais également de comparer les méthodes se basant sur cette approche avec les méthodes de fouille de données développées précédemment.

Ces nouveaux projets de recherche sont complétés actuellement par des travaux dans le domaine bio-médical [17, 5], des travaux centrés sur l'acquisition automatique de modèles de langue [22] et sur le traitement automatique des langues (TAL) en général, des travaux sur la détection de fraudes bancaires (initiés par l'intermédiaire d'une nouvelle thèse CIFRE) et de nouvelles pistes portant sur l'utilisation des architectures telles que les auto-encodeurs pour découvrir des motifs temporels de manière non supervisée ainsi que sur l'analyse d'images satellites. Dans le domaine bio-médical, nous travaillons par exemple à prédire des réactions allergiques lors d'une transfusion sanguine à partir de marqueurs moléculaires présents dans les poches de sang au moment de la transfusion. Si des modèles relativement simples tels que les arbres de décision peuvent être utilisés dans un premier temps, l'ajout de données concernant les patients, le caractère non indépendant et identiquement distribué (I.I.D) de ces données ainsi que le grand déséquilibre des classes apprises (il y a beaucoup moins de cas positifs) nous amène à réfléchir sur de nouveaux algorithmes d'apprentissage plus efficaces dans un tel contexte. C'est aussi le cas dans le domaine de la fraude bancaire où notre but est non seulement de caractériser les fraudes (en terme d'achats et de profils utilisateur) mais également d'identifier les profils de consommateurs (par exemple en utilisant les informations des cartes de fidélité) pour réduire au maximum le nombre de faux positifs déclenchés par les systèmes de détection automatique. Dans ce contexte, un même client peut faire plusieurs transactions (éventuellement corrélées) au cours du temps (les données ne sont donc pas I.I.D) et le pourcentage de fraudes est infime comparé au nombre de transactions effectuées par exemple sur une année dans une chaîne de magasins (les classes sont donc également très déséquilibrées).

Dans le cadre du TAL, nous travaillons actuellement dans deux contextes différents. Dans le premier, nous cherchons à comprendre comment des modèles de langues peuvent être appris en prenant en compte le contexte précis dans lequel les phrases sont prononcées. Dans le second, nous travaillons sur l'assistance intelligente au pilotage de réunions. En particulier, nous cherchons à répondre à des questions du type: i) Comment construire, manipuler et mettre à jour dynamiquement une représentation sémantique des informations échangées au cours d'une réunion ? ii) Comment concevoir de nouveaux algorithmes d'apprentissage automatique et de fouille de données capables d'apprendre des modèles des interactions entre les participants à des réunions ? iii) Comment prendre en compte l'information contextuelle d'une réunion pour rendre les tâches précédentes plus précises ? En particulier, nous travaillons à apprendre automatiquement les contraintes d'un CSP ("Constraint Satisfaction Problem") en utilisant des méthodes à la frontière entre la programmation logique inductive et la fouille de motifs.

Ce document dresse le bilan des travaux que j'ai pu mener ces dix dernières années. Il montre la diversité des champs d'application et de recherche (en fouille de motifs et en apprentissage automatique) que j'ai pu couvrir et qui ont donné lieu à des publications de très bonne qualité. Il illustre également mon implication dans l'encadrement doctoral (2 étudiants ont soutenu leur doctorat et j'encadre actuellement 3 nouveaux étudiants). Il n'y sera pas fait état du reste des activités valorisées dans le cadre d'une HDR : l'encadrement de Masters (14 à ce jour), mon investissement dans l'administration de la recherche (notamment à travers l'écriture de projets, la recherche de financements, la participation à des jurys, des comités, des conseils, l'organisation de conférences, etc.) mais aussi pour l'enseignement (par le biais des responsabilités que j'ai pu prendre, le nombre de cours que j'ai pu enseigner et créer).

# My Publications

## Authored Journals/Top-conferences Peer-reviewed Articles

- [1] Cécile Barat, Christophe Ducottet, Élisabeth Fromont, Anne-Claire Legrand, and Marc Sebban. Weighted symbols-based edit distance for string-structured image classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, volume 6321 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2010.
- [2] Hendrik Blockeel, Toon Calders, Élisabeth Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. An inductive database prototype based on virtual mining views. In *KDD*, pages 1061–1064, 2008.
- [3] Hendrik Blockeel, Toon Calders, Élisabeth Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. An inductive database system based on virtual mining views. *Data Min. Knowl. Discov.*, 24(1):247–287, 2012.
- [4] Lucie Callens, Guy Carrault, Marie-Odile Cordier, Élisabeth Fromont, François Portet, and René Quiniou. Intelligent adaptive monitoring for cardiac surveillance. In *ECAI*, pages 653–657, 2008.
- [5] Fabrice Cognasse, Aloui Chaker, Kim Anh Nguyen, Hind Hamzeh-Cognasse, Fagan Jocelyne, Arthaud Charles-Antoine, Eyraud Marie-Ange, Marc Sebban, Élisabeth Fromont, Bruno Pozzetto, Sandrine Laradi, and Olivier Garraud. Platelet components associated with adverse reactions: predictive value of mitochondrial DNA relative to biological response modifiers. *Transfusion*, page To appear, 2015.
- [6] Marie-Odile Cordier, Élisabeth Fromont, and René Quiniou. Learning rules from multi-source data for cardiac monitoring. *International Journal of Biomedical Engineering and Technology (IJBET)*, special issue on "Warehousing and Mining Complex Data: Applications to Biology, Medicine, Behavior Health and Environment", Volume 3, Issue 1/2:133–155, 2010.
- [7] Fabien Diot, Élisabeth Fromont, Baptiste Jeudy, Emmanuel Marilly, and Olivier Martinot. Graph mining for object tracking in videos. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I*, volume 7523 of *Lecture Notes in Computer Science*, pages 394–409, 2012.
- [8] Basura Fernando, Élisabeth Fromont, Damien Muselet, and Marc Sebban. Discriminative feature fusion for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, June 16-21,*, pages 3434–3441, 2012.

- [9] Basura Fernando, Élisabeth Fromont, Damien Muselet, and Marc Sebban. Supervised learning of gaussian mixture models for visual vocabulary generation. *Pattern Recognition*, 45(2):897–907, 2012. 1.2.2, 1.2.2
- [10] Basura Fernando, Élisabeth Fromont, and Tinne Tuytelaars. Effective use of frequent itemset mining for image classification. In *ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I*, volume 7572 of *Lecture Notes in Computer Science*, pages 214–227. Springer, 2012.
- [11] Basura Fernando, Élisabeth Fromont, and Tinne Tuytelaars. Mining mid-level features for image classification. *International Journal of Computer Vision*, 108(3):186–203, 2014.
- [12] E. Fromont, M.-O. Cordier, and R. Quiniou. Extraction de connaissances provenant de données multisources pour la caractérisation d’arythmies cardiaques. *RNTI-E-4, Cepaduw Editions*, Fouille de données complexes, 2005.
- [13] E. Fromont, R. Quiniou, and M.-O. Cordier. Learning rules from multisource data for cardiac monitoring. In S. Miksch, J. Hunter, and E. Keravnou, editors, *AIME’05 (Artificial Intelligence in Medicine)*, volume 3581 of *LNAI*, pages 484–493, Aberdeen, Scotland, July 2005. Springer Verlag.
- [14] Elisa Fromont, Marie-Odile Cordier, and René Quiniou. Learning from multi source data. In *PKDD’04 (Knowledge Discovery in Databases)*, volume 3202 of *Lecture Notes in Artificial Intelligence*, pages 503–505, Pise, Italie, 2004. Springer.
- [15] Élisabeth Fromont, Adriana Prado, and Céline Robardet. Constraint-based subspace clustering. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 26–37. SIAM, 2009.
- [16] Hoang Thanh Lam, Wenjie Pei, Adriana Prado, Baptiste Jeudy, and Élisabeth Fromont. Mining top-k largest tiles in a data stream. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 82–97, 2014.
- [17] Kim Anh Nguyen, Hind Hamzeh-Cognasse, Marc Sebban, Elisa Fromont, Patricia Chavarin, Lena Absi, Bruno Pozzetto, Fabrice Cognasse, and Olivier Garraud. A computerized prediction model of hazardous inflammatory platelet transfusion outcomes. *PLoS One*, 9(5):e97082, 2014.
- [18] S. Nijssen and E. Fromont. Mining optimal decision trees from itemset lattices. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 530–539, San Jose, CA, USA, 2007.
- [19] Siegfried Nijssen and Elisa Fromont. Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery*, 21(1):9–51, 2010.
- [20] Adriana Prado, Baptiste Jeudy, Élisabeth Fromont, and Fabien Diot. Mining spatiotemporal patterns in dynamic plane graphs. *Intell. Data Analysis*, 17(1):71–92, 2013.

- [21] Hoang-Tung Tran, Élisabeth Fromont, François Jacquenet, and Baptiste Jeudy. Accurate visual features for automatic tag correction in videos. In *Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*, pages 404–415, 2013.

## Other Authored Articles (including books, peer-reviewed workshops and national conferences)

- [22] Leonor Becerra-Bonache, Elisa Fromont, Amaury Habrard, Michael Perrot, and Marc Sebban. Speeding up syntactic learning using contextual information. In *JMLR: Workshop and Conference Proceedings*, pages 1–5, 2012.
- [23] H. Blockeel, T. Calders, E. Fromont, B. Goethals, and A. Prado. Mining views: Database views for data mining. In *ECML/PKDD-2007 International Workshop on Constraint-Based Mining and Learning (CMILE)*, pages 21–33, Warsaw, Poland, 2007.
- [24] Hendrik Blockeel, Toon Calders, Élisabeth Fromont, Bart Goethals, and Adriana Prado. Mining views: Database views for data mining. In *Proc. IEEE ICDE*, pages 1608–1611, 2008.
- [25] Hendrik Blockeel, Toon Calders, Élisabeth Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. Inductive querying with virtual mining views. In Sašo Džeroski, Bart Goethals, and Panče Panov, editors, *Inductive Databases and Constraint-Based Data Mining book*. Springer, Germany, 2010.
- [26] Hendrik Blockeel, Toon Calders, Élisabeth Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. A practical comparative study of data mining query languages. In Sašo Džeroski, Bart Goethals, and Panče Panov, editors, *Inductive Databases and Constraint-Based Data Mining book*. Springer, Germany, 2010.
- [27] Toon Calders, Élisabeth Fromont, Hoang Thanh Lam, and Baptiste Jeudy. Analysis of videos using tile mining. In *ECML/PKDD Workshop on Real-World Challenges for Data Stream Mining*, 2013.
- [28] Fabien Diot, Elisa Fromont, Baptiste Jeudy, Emmanuel Marilly, and Olivier Martinot. Unsupervised tracking from clustered graph patterns. In *International Conference on Pattern Recognition*, 2014.
- [29] B. Fernando, E. Fromont, D. Muselet, and M. Sebban. Accurate visual word construction using a supervised approach. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–7, Nov 2010.
- [30] E. Fromont, M.-O. Cordier, R. Quiniou, and A.I. Hernandez. Kardio and calicot: a comparison of two cardiac arrhythmia classifiers. In *AIME'03 Workshop: Qualitative and Model-based Reasoning in Biomedicine*, pages 29–33, Protaras, Cyprus, 2003.



- [31] E. Fromont and F. Portet. Pilotage d'un système de monitoring cardiaque multisource. In *Actes de conférence de MajecStic'2005 (MANifestation des JEunes Chercheurs STIC)*, pages 346–354, Rennes, France, 2005.
- [32] E. Fromont, R. Quiniou, and M-O. Cordier. Apprentissage multisource par programmation logique inductive. In *RFIA'2006 : 15ème Congrès Reconnaissance des Formes et Intelligence Artificielle*, page pp.115, Tours, France, 2006.
- [33] Élisabeth Fromont, Hendrik Blockeel, and Jan Struyf. Integrating decision tree learning into inductive databases. In Sašo Džeroski and Jan Struyf, editors, *Knowledge Discovery in Inductive Databases "Revised selected papers of the workshop KDID'06"*, volume 4747 of *Lecture Notes in Computer Science*, pages 81–96. Springer Berlin Heidelberg, 2007.
- [34] Taygun Kekec, Rémi Emonet, Elisa Fromont, Alain Trémeau, and Christian Wolf. Contextually constrained deep networks for scene labeling. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.
- [35] Taygun Kekec, Rémi Emonet, Elisa Fromont, Alain Trémeau, and Christian Wolf. Prise en compte du contexte pour contraindre les réseaux profonds: Application à l'Étiquetage de scènes. In *CAP'2014 (Conférence d'Apprentissage)*, Saint-Etienne, 2014.
- [36] Hoang Thanh Lam, Wenjie Pei, Adriana Prado, Baptiste Jeudy, and Élisabeth Fromont. Extraction des k plus grandes tuiles dans un flux de données. In *CAP'2013 (Conférence d'Apprentissage)*, Lille, 2013.
- [37] Adriana Prado, Baptiste Jeudy, Elisa Fromont, and Fabien Diot. PLAGRAM : un algorithme de fouille de graphes plans efficace. In *Conférence d'Apprentissage (CAp)*, pages 342–360, 2011.
- [38] René Quiniou, Lucie Callens, Guy Carrault, Marie-Odile Cordier, Elisa Fromont, Philippe Mabo, and François Portet. Intelligent adaptive monitoring for cardiac surveillance. In Isabelle Bichindaritz, Sachin Vaidya, and Ashlesha Jain, editors, *Computational Intelligence in Healthcare 4, Advanced Methodologies Series: Studies in Computational Intelligence*. Springer, Germany, 2010.
- [39] Jose Carlos Rangel, Miguel Cazorla, Ismael Garcia-Varea, Jesus Martinez-Gomez, Elisa Fromont, and Marc Sebban. Computing image descriptors from annotations acquired from external tools. In *ROBOT 2015: Second Iberian Robotics Conference*, Lisbon, Portugal, 2015.
- [40] Hoang-Tung Tran, Élisabeth Fromont, François Jacquenet, Baptiste Jeudy, and Adrien Martins. Unsupervised video tag correction system. In *Extraction et gestion des connaissances (EGC'2013), Actes, 29 janvier - 01 février 2013, Toulouse, France*, pages 461–466, 2013.
- [41] Imtiaz Masud Ziko, Elisa Fromont, Damien Muselet, and Marc Sebban. Supervised spectral subspace clustering for visual dictionary creation in the context of image classification. In *ACPR 2015: 3rd IAPR Asian Conference on Pattern Recognition*, Kuala Lumpur, Malaysia, 2015.

# Chapter 1

## Introduction

This is the era of “Big Data”. This catchword is present everywhere in the media and in most of our research calls for projects. For good reasons: the amount of data (on every possible subjects) and its complexity is growing incredibly fast (thanks to smarter computing devices, more accessible acquisition tools, cheaper storage facilities, worldwide connections to name a few). Just as a particular example, according to a market survey performed by IDC<sup>1</sup>, between 2009 and 2020 the amount of digital information will grow by a factor of 44. In the case of digital multimedia content and if we consider for example the particular case of YouTube, the statistics provided by that company<sup>2</sup> say that, in 2014, “over 6 billion hours of video are watched each month on YouTube — that’s almost an hour for every person on Earth, and 100 hours of video are uploaded to YouTube every minute”. If “with great power comes great responsibility” then, for sure, with great data comes the necessity to design great tools to make sense of them. That is where *data mining* comes into play. The data mining step is a subpart of the knowledge discovery process which consists in discovering useful (unexpected, targeted, descriptive, meaningful) information in data. Data miners have not waited for the “Big data” era to come out to start designing algorithms that are as efficient as possible and can tackle large and complex data. In particular, the researches presented in this document focus on the use of constraints to either make data mining algorithms more efficient and/or more meaningful and/or more accurate (e.g. in the case of classification tasks).

Chapter 2 of this document will be dedicated to constraint-based data mining in the context of *inductive databases* (IDB). In particular, it will present a number of contributions that I have made in this domain during my Post-doc at the KULeuven (between 2006 and 2008) with Prof. Hendrik Blockeel. An inductive database is a database that contains not only data, but also generalizations (patterns and models) valid in the data. For such databases, ordinary (SQL) queries can be used to access and manipulate data, while inductive queries can be used to generate (mine), manipulate, and apply patterns.

We first designed, in collaboration with researchers in Antwerp a system which stores frequent itemsets, association rules [2, 3], [24, 26, 25] and decision trees [23] in a straightforward way, using usual relational database tables and the standard SQL language to represent, store and query the new generalizations made on the data. From the user’s

---

<sup>1</sup>J. Gantz and D. Reinsel, “The Digital Universe Decade, Are You Ready?,” 2011. [http://www.emc.com/digital\\_universe](http://www.emc.com/digital_universe)

<sup>2</sup><http://www.youtube.com/yt/press/statistics.html>

point of view, tables with itemsets, rules, trees, etc. exist and can be queried like any other table. How these tables are filled (which data mining algorithm is run, with which parameters, etc.) is transparent. For example, when mining itemsets to discover association rules, the user imposes some constraints on the rules to be found (typically confidence and support) and the algorithm yields the set of all rules fulfilling these conditions. On the contrary, typical machine learning algorithms, e.g. decision tree learners, only compute one particular model. In the case of decision tree, this computed tree is generally not the most accurate tree on the training set, nor the smallest one, nor the most likely to generalize well according to some criteria; the learning algorithm only tends to give relatively small trees with relatively high accuracy. The algorithm has usually a number of parameters, the meaning of which can be described quite precisely in terms of how the algorithm works, but not in terms of the results it yields. By integrating constraint-based mining of decision tree learning into inductive databases, we wanted to have an approach for decision tree learning that was just as precise as it was for association rules discovery: the user specifies what kind of trees he wants, and the system looks for such trees.

The complexity issues inherent to exhaustively searching for the best model, particularly in this Big Data context, naturally brought me to a second related research direction also described in Chapter 2: the search for efficient constraint-based decision tree learning algorithms that could be integrated in an IDB. In collaboration with Siegfried Nijssen (also post-doc at KULeuven at that time), we developed DL8 [18, 19], an algorithm to learn decision trees under constraints which optimizes criteria such as the size, the accuracy or the depth of the tree. The constraints used are: the minimal number of examples in each leaf of the tree, the size, depth and accuracy of the tree. The key idea behind our algorithm is that there is a relationship between constraints on decision trees and constraints on itemsets. We showed that optimal decision trees can be extracted from lattices of itemsets in linear time and gave several strategies to efficiently build these lattices. Apart from its obvious use in the context of inductive databases, such an algorithm can be used to evaluate the performance and better understand the behavior of current heuristic decision trees learners. For example, we showed that under the same constraints, DL8 obtains better test accuracy results than the well-known decision tree learner C4.5, which confirms that exhaustive search does not always imply over-fitting.

We also investigated how clustering algorithms could fit in this setting in [15]. We showed that to find efficiently an "optimal" clustering (i.e. which would exactly fulfill some user requirements), an approach could be to look at different subspaces under constraints such as the size of the clusters but also instance-level constraints which could give some information about whether two objects should be part of the same cluster or not. Chapter 2 also includes the description of an exhaustive constraint-based graph mining algorithm called PLAGRAM and its extension to dynamic graphs called DYPLAGRAM. These algorithms were developed more recently but could well fit in the context of inductive databases.

Note that during this stay in Belgium, I worked in the context of national Belgium projects (funded by FWO) and in the context of the European project FP6-IST IQ (*Inductive Queries for Mining Patterns and Models*). This international working context helped me to start collaborations with renowned European research groups which are still active nowadays and have resulted in several bilateral visits and publications.

Chapters 3 and 4 present either new methods that were driven by computer vision applications or applications of methods presented in Chapter 2 to computer vision prob-

lems. These works coincide with my arrival at Jean Monnet University in Saint-Etienne as an associate professor in September 2008. I joined the “Machine Learning” team (now called “Data Intelligence”) of Marc Sebban in the Hubert Curien laboratory. This team is part of a “Computer Science, Image processing and Telecom” department and I naturally started to work on computer vision related problems with the members of the image processing team. In particular, some members of both the computer and the image processing teams were involved in a joint ANR national project called SATTIC and were working on designing algorithms for structured data to solve some simple image classification tasks. In particular, the choices of the low level features and of the structure to use were an open problem.

A common approach to represent an image content was to use histograms of color, texture and edge direction features [59, 119]. Although they are computationally efficient, such histograms only use global information and so provide a crude representation of the image content. That’s why, in the 2000’s, most of the image classification techniques were using other features called *bag-of-visual-words* (BOW). These features come from the *bag-of-words* representation of text documents [112]. They are computed through four basic steps: (i) a keypoint detection step, (ii) a keypoint description step, (iii) a codebook creation step and (iv) an image representation step. Keypoints refer to small regions of interest in the image. They can be sampled densely [86], randomly [122] or extracted with various detectors [100]. Once extracted, keypoints are characterized using a local descriptor, the most widely used (and the one mainly used in this document) being SIFT [96]. A visual codebook is then learned over the collection of descriptors of a training set by using a clustering algorithm [64]. Each cluster representative (typically the centroid) is considered as a visual word of a visual dictionary and each image can then be mapped into this new space of visual words leading to a BOW (or histogram of visual words) [64].

Most of my research works of the last years have been focused on improving this BOW representation mainly by adding some structural information to this initial image representation. We first proposed a method for representing images in the form of strings whose symbols were weighted according to a TF-IDF-based weighting scheme, inspired from information retrieval. To be able to handle such real-valued weights, we introduced a new weighted string edit distance that kept the properties of a distance [1]. The chosen string representation seemed too arbitrary to be broadly used for image classification. Therefore, we decided to focus on a critical step in the BOW creation process: the clustering algorithm (step iii mentioned before). In image processing, the number of labeled image datasets available is important. We thus decided to improve the k-means algorithm that was traditionally used for the BOW creation step by taking into account partial supervised information directly while building the visual vocabulary [9],[29, 41]. In particular, our *Pattern Recognition* article [9] describes how to integrate a semi-supervised optimization term while learning the parameters of a Gaussian mixture model (GMM) over the space of descriptors. This GMM is used to build a soft clustering where each image is partially assigned to some clusters such that images of the same class are more likely to belong to the same clusters. We then explored how to extend logistic regression-based algorithms to multi-class problems to be able to select the most discriminant visual worlds in a large dictionary (or a concatenation of multiple dictionaries) [8]. The parameters of the regression were used to weigh the visual words and keep the most discriminant ones for the given classification problem. We also designed a new marginalized kernel for SVM which could use directly the probabilities output by the regression algorithm. This method is particularly relevant when, in a Big

Data context, one cannot decide what the best low level features to learn a dictionary are, or how big these dictionaries should be. This method allowed us to obtain state-of-the-art results for many image classification benchmark datasets.

In parallel to these works, we decided to assess the use of unsupervised methods such as the pattern mining methods described in Chapter 2 to improve the current image and video low level representations. We showed that frequent patterns over BOW followed by a smart post-processing step to select the best patterns, could help us to build a very good mid-level representation that could be further used by classification algorithms such as SVMs [10, 11]. In particular our proposed method drastically improved the state-of-the-art results in image classification over a very large set of datasets (excluding benchmarks such as IMAGENET which can contain several millions of images and for which other kinds of methods are more successful as discussed in the conclusion of this document). The post-processing steps proposed also helped extending the state-of-the-art in pattern mining and in particular in discriminative subgroup discovery.

These past seven years in Saint-Etienne have allowed me to gain some experience in supervising PhD students. I have co-supervised (with Baptiste Jeudy and François Jacquenet) two PhD theses on subjects at the crossroad between pattern mining and video analysis. The PhD thesis of Fabien Diot, defended in June 2014, dealt with the use of graph mining for object tracking in videos. This thesis was done in collaboration with an industrial partner, Alcatel-Lucent Bell-Labs, and allowed us to file two patents. We have shown that frequent patterns, and in this case, subgraph patterns, could be relevant features to track objects in videos without any supervised information and in very general contexts [20, 7], [28, 37]. In particular, our method is especially promising when the objects to track are not known in advance, can take any shape and, when the camera as well as the objects, are moving (which prevents us to use a standard background subtraction technique). The second thesis from Hoang-Tung Tran, defended in July 2014, dealt with automatically correcting tags in videos uploaded in general platform such as Youtube. Our main assumption was that tags that are manually added to videos are, in general, either incomplete or incorrect. To automatically correct the tags, we relied on visual information to compare similar videos (based on shared frequent visual patterns) and on a propagation method to remove or add new tags to a video according to its neighbors [21],[40]. On a similar application domain, I also collaborated with Belgian colleagues met during my Post-doc on the problem of mining frequent patterns in data streams [16],[27, 36]. Besides helping us to analyze videos in real time (which was not possible with the previous PhD works), the new proposed algorithms also contributed to improve the state-of-the-art in constraint-based stream mining.

Finally, in 2014, the *Solstice (Similarity of locally structured data in computer vision)* ANR project that I led was accepted. This project aims to explore the use of locally structured data, which combine visual features (such as interest points, segmented regions or visual words in images) with discrete structures (such as strings, trees, combinatorial maps or, more generally, graphs) in order to model local (spatio-temporal) relationships holding between these features. Since then, I am co-supervising two new PhD students who are working on the (supervised or not) discovery of discriminative representations for image and video analysis. The first PhD student, Romain Deville (co-supervised with Baptiste Jeudy and Christine Solnon from the LIRIS) works on a topic which is a direct continuation of Fabien Diot's PhD thesis. He explores the use of subgraphs patterns to analyze 2D and 3D objects (in particular images and videos). In particular, Romain is developing new geometric graph mining algorithms to extract

different kinds of substructures from a database of graphs. The second PhD student, Damien Fourure, works on a very popular machine learning technique since the rise of the *Facebook research center*: deep-learning. His subject is related to learning with deep neural networks how to fully label outdoor scenes using both temporal and 3D information [34, 35, 39]. Deep learning is one of the most successful learning method in computer vision nowadays (it is also very successful in other domains such as speech recognition and natural language processing) thanks to the availability of huge datasets and of the still increasing processing power of computers. With this PhD subject, we hope to be able not only to contribute to the field of machine learning but also to compare deep learning approaches to the pattern mining methods previously mentioned.

This *Habilitation à Diriger des Recherches* aims at assessing the work I did for the last ten years after defending my PhD thesis. It shows not only the coherence but the evolution and the multidisciplinary aspects of my work. It is organized as follows. In Chapter 2, I will present the constraint-based exhaustive data mining tools I have developed in the context of inductive databases. In particular, I will focus on the definition of a general framework of inductive database and then on the constraints and their properties to restrict the search space and make this exhaustive search possible in the case of decision trees, clustering, closed sets and graph mining. Chapters 3 and 4 will develop my computer vision applications and will give some results obtained by applying the previously introduced algorithms as well as new algorithms dedicated to this particular context. I will, in particular, cover applications in image classification and object tracking in videos. The last chapter concludes this report and draws some possible prospects for this work.

## Chapter 2

# Constraint-based Data Mining for Inductive Databases

Data mining is not a one-shot activity, but rather an iterative and interactive process. During the whole discovery process, many different data mining tasks are usually performed, their results are combined, and possibly used as input for other data mining tasks. To support this knowledge discovery process, there is a need for integrating data mining with data storage and management. The concept of inductive databases (IDB) has been proposed as a means of achieving such integration [83]. This chapter will first present the framework, called *Mining views* we proposed to implement such inductive databases and will then present two constraint-based exhaustive algorithms DL8 and PLAGRAM that have been developed and can be integrated in this context. To understand this chapter, we expect the reader to have some knowledge about data mining in general and pattern mining in particular. For further information, we refer the interested reader to [80].

### 1 Mining Views

To tackle the task of building an IDB, one has to i) choose a query language that can be general enough to cover most of the data mining and machine learning toolkit while providing enough flexibility to the users in terms of constraints, ii) ensure a closure property to be able to reuse intermediate results, and iii) provide an intuitive way to interpret the results.

Here, we describe how such an inductive database can be implemented in practice, as presented in [56], [2] and [33, 24]. Contrary to numerous proposals of data mining query languages [79, 99, 84, 124, 125, 115, 126, 104, 50], we propose a relational database model based on the so-called *virtual mining views*. The mining views are relational tables that virtually contain the complete output of data mining tasks. E.g., for itemset mining, we would consider that any IDB contains a table called *Sets* virtually storing all frequent patterns. In other words, as far as the user is concerned, all possible patterns are stored. On the physical layer, however, these tables are actually empty; whenever a query is formulated selecting, for instance, itemsets from these tables, the database system triggers a data mining algorithm (e.g., Apriori [43]), which computes the result of the query, in exactly the same way that normal views in databases are only computed at query time, and only to the extent necessary for answering the query. The complete model is illustrated in Figure 2.1.

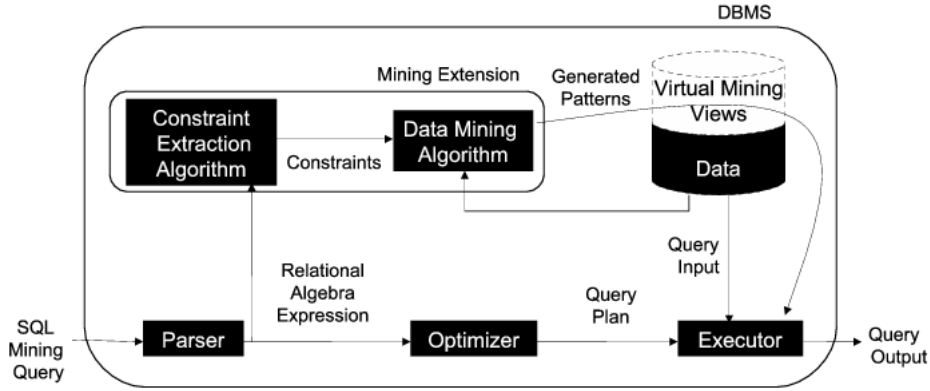


Figure 2.1: An Inductive Database.

```

create table T_Concepts
select  $A_1, A_2, \dots, A_n$ 
from T
group by cube
 $A_1, A_2, \dots, A_n$ 

```

Figure 2.2: Data cube.

A user can use the mining views in his or her query as if they were regular database tables. Given a query, the parser is then invoked by the database system, creating an equivalent relational algebra expression. At this point, the expression is processed by the *Mining Extension* which extracts from the query the constraints to be pushed into the data mining algorithms. The output of these algorithms is then materialized in the mining views. After the materialization, the work flow of the system continues as usual and, as a result, the query is executed as if all patterns and models were always stored in the database.

The mining views framework consists of relational tables that virtually contain the complete output of data mining algorithms executed over a given dataset. Every time a dataset  $T$  is created in the system, all virtual mining views associated with  $T$  are automatically created.

In order to represent the patterns and models as general as possible, we propose to add to the system a mining view called *Concepts*. Given a table  $T(A_1, \dots, A_n)$ , let  $Dom(T) = Dom(A_1) \times \dots \times Dom(A_n)$  denote the domain of  $T$ . The mining view  $T\_Concepts(cid, A_1, \dots, A_n)$  is such that for every tuple  $t$  in  $T$ , there exist up to  $2^n$  unique tuples  $\{t'_1, \dots, t'_{2^n}\}$  in  $T\_Concepts$ . More specifically, for any  $t$  in  $T$ ,  $T\_Concepts$  contains all  $t'$  for which it holds that for each  $A_j$ ,  $t'_i.A_j = t.A_j$  or  $t'_i.A_j = '?'$ .

In fact,  $T\_Concepts$  represents exactly a *data cube* [77] built from table  $T$  with the difference that the dummy value “ALL” introduced in [77] is replaced by the value ‘?’ (see Figure 2.3). By following the syntax introduced in [77], the mining view  $T\_Concepts$  would be created with the SQL-query shown in Figure 2.2 (consider adding  $cid$  after its creation).

As each of the concepts can actually cover more than one tuple in  $T$ , a unique identifier  $cid$  is associated to each of them. A tuple, or *concept*,  $(cid, a_1, \dots, a_n) \in T\_Concepts$  represents all tuples from  $Dom(T)$  satisfying the condition



PLAYTENNIS						
Day	Outlook	Temp	Humidity	Wind	Play	
D1	Sunny	Hot	High	Weak	No	
D2	Sunny	Hot	High	Strong	No	
D3	Overcast	Hot	High	Weak	Yes	
D4	Rain	Mild	High	Weak	Yes	
D5	Rain	Cool	Normal	Weak	Yes	
D6	Rain	Cool	Normal	Strong	No	
...	...	...	...	...	...	

<i>Playtennis.Concepts</i>						
cid	Day	Outlook	Temp	Humidity	Wind	Play
1	?	Sunny	?	High	?	No
2	?	Sunny	?	Normal	?	Yes
3	?	Overcast	?	?	?	Yes
4	?	Rain	?	?	Strong	No
5	?	Rain	?	?	Weak	Yes
6	?	?	?	?	?	?
...	...	...	...	...	...	...

Figure 2.3: The PLAYTENNIS data table and its corresponding mining view *Concepts*.

$$\bigwedge_{i|a_i \neq '?'} A_i = a_i.$$

Figure 2.3 shows a data table for the classic PLAYTENNIS example [102], together with a sample of its corresponding mining view *Concepts*.

## 1.1 Representing Models as Sets of Concepts

Given a data table  $T$ , and its corresponding mining view  $T\_Concepts$ , we now explain how a variety of models can be represented using yet other mining views. Although all mining views are defined over  $T$ , we omit the prefix  $T$  when it is clear from the context.

### 1.1.1 Itemsets and Association Rules

As itemsets in a relational database are conjunctions of attribute-value pairs, they can be represented as concepts. The result of frequent itemset mining can therefore be represented by a view  $Sets(cid, supp)$ . For each itemset, there is a tuple with  $cid$  the identifier of the itemset (concept) and  $supp$  its support. Also other attributes, such as  $\chi^2$  [103] or any correlation measure, could be added to the view to describe the itemsets. Similarly, association rules can be represented by a view  $Rules(rid, cida, cidc, cid, conf)$ , where  $rid$  is the rule identifier,  $cida$  and  $cidc$  are the identifiers for the concepts representing the antecedent and the consequent of the rule respectively,  $cid$  is the union (disjunction) of these, and  $conf$  is the confidence of the rule. Again, many other attributes, such as lift, conviction, or gini index, could be added to describe the rules.

In Figure 2.4, queries (A) and (B) are example mining queries over itemsets and association rules, respectively. Query (A) asks for itemsets having support of at least 30 and size of at most 5, while query (B) asks for association rules having support of at least 30 and confidence of at least 80%. Note that these two common data mining tasks and the well known constraints “minimum support” and “minimum confidence” can be expressed quite naturally with SQL queries over the mining views.

### 1.1.2 Decision Trees

A decision tree learner typically induces a single decision tree from a data set. This setting contrasts strongly with discovery of itemsets and association rules, which is set-

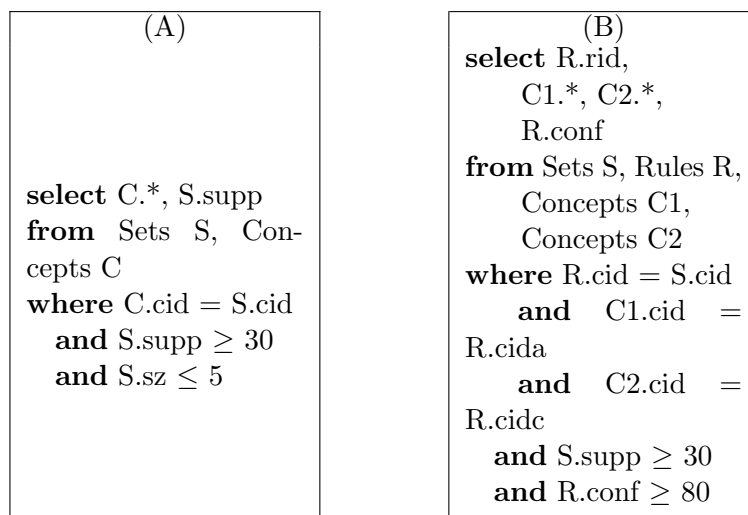


Figure 2.4: Example mining queries over itemsets and association rules.

oriented: given certain constraints, the system finds all itemsets or association rules that fit the constraints.

In decision tree induction, given a set of (sometimes implicit) constraints, one tries to find one tree that fulfills the constraints and, besides that, optimizes some other criterion.

In the inductive databases context, decision tree induction looks somewhat different.

Here, a user would typically write a query asking for all trees that fulfill a certain set of

constraints, or optimizes a particular condition. The user might ask, for instance, for the tree

with the highest training set accuracy among all trees of size at most 5; or the tree that

maximizes some function of size and training set accuracy. This leads to a much more declarative

way of mining for decision trees.

Such an approach blends in nicely in the virtual mining views framework. The set of all trees

predicting a particular target attribute  $A_i$  from other attributes is represented by a view

$Trees_{A_i}(treeid, cid)$ . A unique identifier  $treeid$  is associated with each tree and each of the trees is described as a set of concepts  $cid$ , each concept describing one leaf

of the tree. Figure 2.5 shows a decision tree built to predict

the attribute *Play* using all other attributes in the data table, and its representation in the mining view  $Trees_{Play}$ , using the mining view *Concepts* from

Figure 2.3.

Additionally, a view representing several characteristics of a tree learned for one specific target attribute  $A_i$  is added:  $Treescharac_{A_i}(treeid, acc, sz)$ . For every tree, there is a tuple with a tree identifier  $treeid$  and its corresponding accuracy ( $acc$ ) and size ( $sz$ , in number of nodes). Again, other attributes could be added to describe the decision trees.

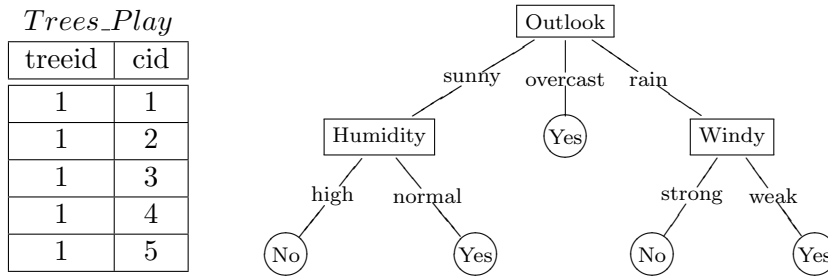


Figure 2.5: Decision Tree built to predict the attribute *Play*.

In Figure 2.6, we present some example mining queries over decision trees. Query (C) selects decision trees having the attribute *Play* as the target attribute, having maximal accuracy among all possible decision trees of size  $\leq 5$ . Query (D) asks for decision trees having a test on “Outlook=Sunny” and on “Wind=Weak”, with a size of at most 5 and an accuracy of at least 80%.

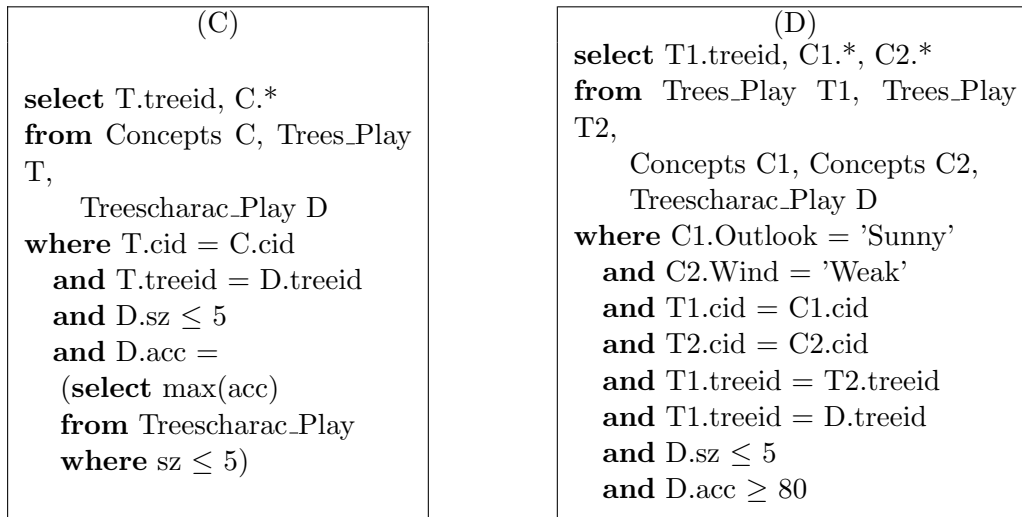


Figure 2.6: Example mining queries over decision trees.

**Prediction** In order to classify a new example using one or more of the learned decision trees, one simply looks up the concept that covers the new example. More generally, if we have a test set  $S$ , all predictions of the examples in  $S$  are obtained by equi-joining  $S$  with the semantic representation of the decision tree given in the virtual mining view *Concepts*. We join  $S$  to *Concepts* using a variant of the equi-join that requires that either the values are equal, or there is a wildcard value.

Consider the PLAYTENNIS example of Figure 2.3. Figure 2.7

shows a query that predicts the attribute *Play* for all unclassified examples in table

TEST\_SET(Day, Outlook,Temp,Humidity,Wind), using the tree of Figure 2.5 (which has identification number 1).

### 1.1.3 Clustering

We focus here on non-probabilistic extensional clustering procedures, i.e., procedures that

(E)
<pre> <b>select</b> T.treeid, S.*, C.Play <b>from</b> Test_set S, Concepts C, Trees_Play T <b>where</b> T.cid = C.cid       <b>and</b> (S.Outlook = C.Outlook <b>or</b> C.Outlook = '??')       <b>and</b> (S.Temp = C.Temp <b>or</b> C.Temp = '??')       <b>and</b> (S.Humidity = C.Humidity <b>or</b> C.Humidity = '??')       <b>and</b> (S.Wind = C.Wind <b>or</b> C.Wind = '??')       <b>and</b> T.treeid = 1 </pre>

*Figure 2.7: Prediction query.*

defines clusters simply as sets of elements. Clusterings are then easily represented as sets of clusters which are themselves sets of instances or concepts.

All possible clusterings that can be learned from  $T$  are represented in the view  $Clusterings(clusid, clid)$  and all clusters belonging to all clusterings are represented in the view  $Clusters(clid, cid)$ . A unique identifier  $clustid$  is associated to each clustering and each of the clusterings is described by a set of clusters. A unique identifier  $clid$  is associated to each cluster and each of the clusters is described by a set of concepts.

Again, a view representing the characteristics of all clusterings is added:  $Clusteringscharac(clusid, sz)$ , with  $sz$  the number of clusters. Of course other attributes could be added to this view.

Note that since the mining views  $Concepts$  virtually contain a finite number of concepts (depending on the data table), the number of partitions (in the case of clustering) as well as the number of trees that can be described using these concepts is also finite.

In Figure 2.8, queries (F) and (G), respectively, are examples of how the user can formulate the popular must-link (two instances must be in the same cluster) and cannot-link (two instances must not be in the same cluster) constraints [123] in our approach. In both queries,  $I\_Concepts(Day, cid)$  is a view associating every instance in the data table with its covering concepts, which can be easily created by the user. Hence, query (F) asks for clusterings in which the instances “D1” and “D2” are in the same cluster, that is, in which both instances are covered by concepts describing the same cluster. Query (G) is formulated by using the opposite reasoning.

For more examples of possible scenarii, we refer the reader to [2] and [24, 25]. Note that a subspace clustering algorithm which could interact with the  $Concepts$  table has been published in [15].

## 1.2 Discussion

Note that the system can potentially support as many virtual mining views as types of patterns of interest. To make the framework more general, we have represented it by an intuitive common set of mining views. However, this framework should be extended to fit more structured data such as, e.g. graphs. Graphs are typically well represented in a relational databases (and not in a single “flat” table as shown in Figure 2.3) using separate tables to describe its edges and nodes. In the case of such structure data, the  $Concepts$  table would not be sufficient to describe the components of the graph as it lacks structural information. One possibility to describe a graph would be to use a table

<p>(F)</p> <pre> <b>select</b> C.* <b>from</b> Clustering C       Clusters C11, Clusters C12,       I_Concepts I1, I_Concepts       I2 <b>where</b> I1.Day = 'D1'       <b>and</b> I2.Day = 'D2'       <b>and</b> C11.cid = I1.cid       <b>and</b> C12.cid = I2.cid       <b>and</b> C11.clid = C12.clid       <b>and</b> C.clid = C11.clid </pre>	<p>(G)</p> <pre> <b>select</b> C1.* <b>from</b> Clustering C1, Clustering C2,       Clusters C11, Clusters C12,       I_Concepts I1, I_Concepts       I2 <b>where</b> I1.Day = 'D1'       <b>and</b> I2.Day = 'D2'       <b>and</b> C11.cid = I1.cid       <b>and</b> C12.cid = I2.cid       <b>and</b> C1.clusid = C2.clusid       <b>and</b> C1.clid = C11.clid       <b>and</b> C2.clid = C12.clid       <b>and</b> C11.clid <math>\neq</math> C12.clid </pre>
---	---

Figure 2.8: Example mining queries over clusterings.

*Subgraphs* instead of the table *Concepts* to describe the possible subgraphs of a single graph or of a graph database.

Besides the design of a general and intuitive framework, another problem that arises when creating such an inductive database concerns the types of constraints that can actually be extracted from the mining queries. A first step towards solving this problem is described in [56]. For implementation details and use case, the reader can refer to [3] and [25, 26].

The last problem concerns the availability of algorithms that can find an exact solution in a reasonable time with respect to the constraints that can be extracted from the query. The next two sections present two attempts to design such algorithms first for decision tree learning and then for graph mining.

## 2 Induction of Optimal Decision Trees

Decision trees are among the most popular predictive models and have been studied from many perspectives. However, no general framework exists to constrain the induction of decision trees and guarantee an exact result with respect to the given constraints. On the other hand, the topic of exhaustively (i.e. exactly) determining all patterns satisfying certain constraints has been studied extensively in the area of *local pattern mining* (see [42, 133, 81] to cite a few of the initial works). A natural question addressed in this section, is hence if we can exploit the experience in local pattern mining for the discovery of decision trees under constraints. As such, our work takes the LeGo approach [88, 54], in that it studies how local pattern mining techniques can be used to build global models.

Our main starting point is that many decision tree learning problems can be formulated as *queries* of the following canonical form:

$$\operatorname{argmin}_T f(T) \text{ subject to } \varphi(T), \quad (\text{Canonical Decision Tree Learning Query})$$

i.e., we are interested in finding the best tree(s) according to a function  $f(T)$ , among all trees which fulfill the constraints specified in the formula  $\varphi(T)$ .

For instance, the following questions could be of interest for a decision tree user:

- Which tree has the smallest error? In this case  $f(T)$  is an error function that we wish to minimize.
- Which is the smallest tree with sufficiently high accuracy? In this case the (ranking) function  $f(T)$  should prefer smaller trees among sets of sufficiently accurate trees. Alternatively, we can reformulate the problem in a Bayesian setting [55, 62].
- Which tree is least sensitive to noise in the class labels? This could require that every leaf of a decision tree has at least a significant majority class. The latter can be seen as a constraint  $\varphi(T)$  on the trees of interest.
- Which tree preserves privacy best by being well-balanced? This would impose a constraint  $\varphi(T)$  on the trees of interest [75, 97].
- Which tree incurs the smallest amount of classification costs? For example, it can be desirable that the expected costs for classifying examples do not exceed a certain predefined threshold value [116].
- Which tree is most *justifiable* from an expert's perspective, by satisfying predefined constraints on the predictions that can be made by the tree? For instance, one could wish to enforce that certain examples are never misclassified, or certain tests are always executed in a given order.

Observe that some of these problem settings are conventional, in the sense that they are formalizations of the problem of finding models of good predictive accuracy. Other problems are less conventional, the main focus being on the syntax of the predictive model.

Many algorithms have been proposed to address these learning problems. Most common are the algorithms that rely on the principle of top-down induction through heuristics (for example C4.5 [110] and CART [53]). These algorithms do not explicitly minimize a global optimization criterion, but rely on the development of a good heuristic to obtain reasonable solutions. In practice, for each new problem setting that was studied, a new heuristic was proposed in the literature.

The benefit of an exact algorithm is that we do not need to develop new heuristics to deal with many types of learning problems and constraints. We are sure that its result is the best that one can hope to achieve according to the predefined optimization criterion and constraints; no fine-tuning of heuristics is necessary. Hence, the results of an exact algorithm can also be used to determine how well an existing heuristic decision tree learner approximates a global optimization criterion.

The development of an *exact* algorithm for learning decision trees has seldom been considered because many decision tree learning problems are known to be NP-complete [82]. Therefore an efficient algorithm for the general case most likely does not exist. This theoretical result however does not imply that the problem is intractable in all cases. Many frequent itemset mining algorithms have been applied successfully despite the exponential nature of the itemset mining problem. This is an indication that, on some datasets, exact decision tree induction may still be feasible if we can do this by using itemset mining results.

In the work published in [19], we have provided evidence that for a reasonable number of datasets, exact decision tree induction is indeed practically feasible by taking this

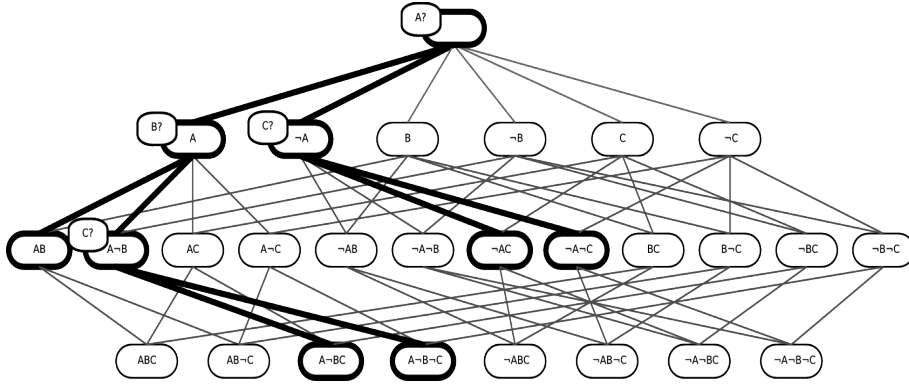


Figure 2.9: The Hasse diagram of a part of an itemset lattice for items  $\{A, \neg A, B, \neg B, C, \neg C\}$ ; binary decision tree  $A(B(C(l,l),l),C(l,l))$  is marked in this diagram

approach. We showed that decision trees can also be learned from the condensed itemset representation of *closed itemsets* [108]. This observation allows us to obtain better practical performance. For the proofs of the theorems and lemmas stated in this section, we also refer the interested user to [19].

## 2.1 Relationships between Itemsets and Decision Trees

Let us first introduce some terminology concerning *frequent itemsets* and *decision trees* before studying the relationships between these domains.

### 2.1.1 Itemsets

Let  $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$  be a set of items and let  $D = \{t_1, t_2, \dots, t_n\}$  be a bag of transactions, where each transaction  $t_k$  is an itemset such that  $t_k \subseteq \mathcal{I}$ . A transaction  $t_k$  contains a set of items  $I \subseteq \mathcal{I}$  iff  $I \subseteq t_k$ . The transaction identifier set (TID-set)  $tid(I) \subseteq \{1, 2, \dots, n\}$  of an itemset  $I \subseteq \mathcal{I}$  is the set of identifiers of all transactions that contain itemset  $I$ . The frequency of an itemset  $I \subseteq \mathcal{I}$  is defined to be the number of transactions that contain the itemset, i.e.  $\text{freq}(I) = |tid(I)|$ ; the support of an itemset is  $\text{support}(I) = \text{freq}(I)/|D|$ . An itemset  $I$  is said to be frequent if its support is higher than a given threshold  $\text{minsup}$ ; this is written as  $\text{support}(I) \geq \text{minsup}$  (or, equivalently,  $\text{freq}(I) \geq \text{minfreq}$ ).

A useful property of itemsets is that they constitute a lattice.

**Definition 1** A complete lattice is a partially ordered set in which any two elements have a unique least upper bound and a unique greatest lower bound.

In this case the partial order is defined by the subset relationship  $\subseteq$  on the elements in the set  $2^{\mathcal{I}}$ . The least upper bound of two sets is computed by the intersection ( $\cap$ ) operator, the greatest lower bound by the union ( $\cup$ ) operator. The lower bound  $\perp$  of this lattice is  $\emptyset$ ; the higher bound  $\top$  is the set  $\mathcal{I}$ .

Part of a lattice is depicted in Figure 2.9 in a *Hasse diagram*, where we assume  $\mathcal{I} = \{A, \neg A, B, \neg B, C, \neg C\}$ ; we only depict itemsets in which an item  $i$  and its negation  $\neg i$  do not occur together. Edges denote a subset relation between sets; sets are depicted as nodes. On top of the lattice is the lower bound which corresponds to the empty set

$\emptyset$  (level 0); the higher bound  $\{A, \neg A, B, \neg B, C, \neg C\}$  is not depicted as it includes items as well as their negations. There is an edge between a node in a given level and a node in the next level if the set of the former is strictly included in the set of the latter and if the size of the two sets only differs by one item.

### 2.1.2 Decision trees

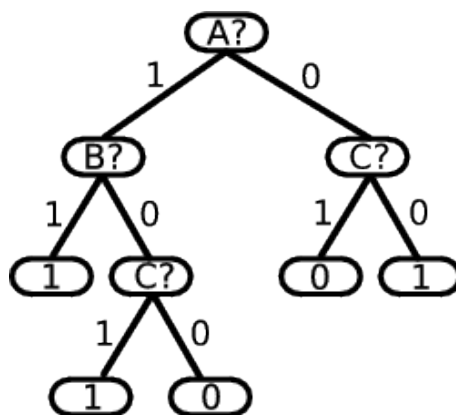


Figure 2.10: An example tree

Our running decision tree example is given in Figure 2.10. We assume that all tests are Boolean; non-binary attributes are transformed into Boolean attributes by mapping each possible value to a separate attribute. Numerical attributes are discretized and binarized beforehand (they will then be called *features*). The input of a decision tree learner is hence a binary matrix  $B$ , where  $B_{ij}$  contains the value of feature  $i$  of example  $j$ .

A common way to represent a decision tree is as a set of rules [110]. Each leaf of the tree corresponds to a rule. Our example tree can be represented in the following way:

- if  $A = 1$  and  $B = 1$  then predict 1
- if  $A = 1$  and  $B = 0$  and  $C = 1$  then predict 1
- if  $A = 1$  and  $B = 0$  and  $C = 0$  then predict 0
- if  $A = 0$  and  $C = 1$  then predict 0
- if  $A = 0$  and  $C = 0$  then predict 1

Hence we can see decision tree learning as finding a set of rules with certain properties that allow the set to be represented as a tree.

### 2.1.3 Link between Decision Trees and Itemsets

A main observation in the LeGo framework [88] is that there is a link between rules in predictive models and patterns in pattern mining. Assume that we are given an attribute-value table  $B$  in which all features are binary. We can transform table  $B$  into a transactional form  $D$  such that  $t_j = \{i \mid B_{ij} = 1\} \cup \{\neg i \mid B_{ij} = 0\}$ . Thus, every feature value is mapped to a positive  $i$  or a negative item  $\neg i$ . The head of a rule, for instance,

$$A = 1 \text{ and } B = 0 \text{ and } C = 1$$



can now be transformed into an itemset  $\{A, \neg B, C\}$ . Transactions in which the head of the rule is true correspond to transactions in which the itemset is contained. Hence the decision tree of Figure 2.10 can equivalently be represented by a set of *class association rules*:

$$\begin{aligned} \{A, B\} &\rightarrow 1 \\ \{A, \neg B, C\} &\rightarrow 1 \\ \{A, \neg B, \neg C\} &\rightarrow 0 \\ \{\neg A, C\} &\rightarrow 0 \\ \{\neg A, \neg C\} &\rightarrow 1 \end{aligned}$$

A class association rule  $I \rightarrow c$  [94] consists of an itemset  $I$  and a class value  $c$ .

The problem of learning a decision tree is now a problem of finding a set of class association rules. As we are usually interested in finding accurate trees, we can reduce this further to a problem of finding itemsets, that is, class association rules without heads: assume we compute the frequency  $\text{freq}_c(I)$  of an itemset  $I$  for each class  $c$  separately, we can associate to each itemset the class label for which its frequency is highest,

$$c(I) = \operatorname{argmax}_{c' \in \mathcal{C}} \text{freq}_{c'}(I),$$

as this will minimize the prediction error for the examples in the leaf. Given a decision tree  $T$ , we denote the set of itemsets corresponding to leaves by  $\text{leaves}(T)$ ; in our example,

$$\text{leaves}(T) = \{\{A, B\}, \{A, \neg B, C\}, \{A, \neg B, \neg C\}, \{\neg A, C\}, \{\neg A, \neg C\}\};$$

itemsets corresponding to internal nodes are denoted by  $\text{internal}(T)$ , in our example,

$$\text{internal}(T) = \{\emptyset, \{A\}, \{A, \neg B\}\};$$

Finally, all itemsets that correspond to paths in the tree are denoted with  $\text{paths}(T) = \text{internal}(T) \cup \text{leaves}(T)$ .

The problem of finding a decision tree can now alternatively also be formulated as follows. We are interested in finding a set of itemsets  $\mathcal{P} \subseteq 2^{\mathcal{I}}$  such that

$$\exists T : \text{paths}(T) = \mathcal{P} \text{ and } T = \operatorname{argmin}_T f(T) \text{ subject to } \varphi(T).$$

Note that we can easily characterize which sets of itemsets represent decision trees.

**Lemma 1** *Given a set of itemsets  $\mathcal{P} \subseteq 2^{\mathcal{I}}$ , then  $\exists T : \text{paths}(T) = \mathcal{P}$  if and only if for every itemset  $I \in \mathcal{P}$  either:*

- (1) *there is no  $I' \in \mathcal{P}$  such that  $I \subset I'$  (in this case  $I \in \text{leaves}(T)$ );*
- (2) *there is exactly one item  $i \in \mathcal{I}$  such that  $I \cup \{i\}, I \cup \{\neg i\} \in \mathcal{P}$  (in this case  $I \in \text{internal}(T)$ ).*

Hence, the problems of finding decision trees  $T$  and sets of itemsets  $\mathcal{P}$  fulfilling the conditions of Lemma 1 are equivalent. Indeed, the reader can check in our example that a set  $\mathcal{P}$  fulfilling these conditions corresponds to a decision tree with  $\text{paths}(T) = \mathcal{P}$ .

An important observation that we will exploit is that a lattice of itemsets can be thought of as a compact representation of a set of decision trees. This is illustrated

in Figure 2.9, where we have highlighted the decision tree of Figure 2.10; in principle any decision tree over binary features  $\{A, B, C\}$  consists of a similar set of paths in this lattice. Note that we assume that trees never have an item and its negation in one path and that we hence do not need to consider the part of the lattice containing such itemsets.

The most basic problem one could be interested in is that of finding an accurate decision tree. The *accuracy* of a decision tree is derived from the number of misclassified examples in the leaves:

$$accuracy(T) = \frac{|D| - error(T)}{|D|} \quad \text{where} \quad error(T) = \sum_{I \in leaves(T)} error(I)$$

and  $error(I)$  is the number of examples ending up in leaf  $I$  not labeled with the majority class of the examples in  $I$ :

$$error(I) = freq(I) - freq_{c(I)}(I)$$

For the *size* of a tree we take the size of the set  $paths(T)$ .

An example of a decision tree learning problem is to find the tree

$$\operatorname{argmin}_T (error(T), |T|)$$

that minimizes error in the first place and cuts ties between trees of equal error using the size function.

## 2.2 Constraints on Decision Trees

We are interested in expressing decision tree learning problems as queries of the form

$$\operatorname{argmin}_T f(T) \text{ subject to } \varphi(T)$$

which corresponds to finding the best tree(s) according to the function  $f(T)$  among all trees which fulfill the constraints specified in the formula  $\varphi(T)$ . In most applications the constraints in  $\varphi(T)$  and the criteria in  $f(T)$  have properties that can be exploited.

In the following, the functions  $I_1 = child_1(I, T)$  and  $I_2 = child_2(I, T)$  return the itemsets representing respectively the left-hand and right-hand child node of the internal node  $I$  in binary tree  $T$ .

### 2.2.1 Properties of Constraints and Criteria

When an **optimization criterion**  $f(T)$  is specified, this criterion may have properties that we will refer to as *additivity* and *structure independence*.

**Additivity** An *additive* optimization criterion is a function  $f(T)$  over a tree  $T$  which can be rewritten as follows:

$$f(T) = \sum_{I \in leaves(T)} f_{leaf}(I) + \sum_{I \in internal(T)} f_{internal}(I, child_1(I, T), child_2(I, T)),$$

where function  $f_{leaf}(I) \geq 0$  is a *leaf criterion* and function  $f_{internal}(I, I_1, I_2) \geq 0$  is an *internal criterion*. An example of an additive optimization criterion is *size*, in which  $f_{leaf}(I) = 1$  and  $f_{internal}(I, I_1, I_2) = 1$ .

**Structure Independence** An additive optimization criterion  $f(T)$  is *structure independent* if we can rewrite the leaf criteria and internal criteria as follows:  $f_{internal}(I, I_1, I_2) = f'_{internal}(tid(I), tid(I_1), tid(I_2))$  and  $f_{leaf}(I) = f'_{leaf}(tid(I))$ , for functions  $f'_{internal}$  and  $f'_{leaf}$  over sets of transactions. Hence, the evaluation depends only on the transactions covered by the nodes, not on the structure of the tree. Please note that *size* is also a structure independent criterion according to our definition; the reasoning is that the size of a tree is only determined by the number of partitions induced by the tree in the set of transactions; otherwise the structure of the tree is unimportant.

We show that many common optimization criteria are additive and that a restriction to such criteria is not very restrictive. For constraints we can formulate similar properties as for criteria. In most cases, the constraint  $\varphi(T)$  is a conjunction of a number of independent constraints, which can have the following properties.

**Conjunctivity over Paths** A *conjunctive path* constraint is a formula over a tree which can be written as:

$$\varphi_{conjunctive}(T) = \bigwedge_{I \in leaves(T)} \varphi_{leaf}(I) \wedge \bigwedge_{I \in internal(T)} \varphi_{internal}(I, child_1(I, T), child_2(I, T)),$$

where formula  $\varphi_{leaf}(I)$  is a *leaf constraint* and formula  $\varphi_{internal}(I, I_1, I_2)$  is an *internal constraint*.

An example of an internal constraint is that internal nodes should not have *pure* class distributions:

$$\varphi_{internal}(I, I_1, I_2) = (|tid(I)| \neq \max_{c \in C} |tid_c(I)|). \quad (2.1)$$

This internal constraint is special in the sense that it only takes  $I$ , not its children, into account. An example of a leaf constraint is that the number of examples not belonging to a majority class is small:

$$\varphi_{leaf}(I) = (|tid(I)| - \max_{c \in C} |tid_c(I)| \leq maxfreq). \quad (2.2)$$

An example of an internal constraint in which the left-hand and right-hand child are used, is:

$$\varphi_{internal}(I, I_1, I_2) = (||tid(I_1)| - |tid(I_2)|| \geq mindif),$$

which states that an internal node splits examples in balanced proportions.

**Structure Independence** A *structure independent* constraint  $\varphi_{structure\_ind}(T)$  is a conjunctive path constraint in which  $\varphi_{internal}(I, I_1, I_2) = \varphi'_{internal}(tid(I), tid(I_1), tid(I_2))$  and  $\varphi_{leaf}(I) = \varphi'_{leaf}(tid(I))$ , for formulas  $\varphi'_{internal}$  and  $\varphi'_{leaf}$  over sets of transactions.

An example of a structure independent path constraint is *minimum support*, in which

$$\varphi_{leaf}(I) = \varphi_{internal}(I, I_1, I_2) = (|tid(I)| \geq minfreq);$$

it is easy to see that this constraint is computed from  $tid(I)$  only.

**Anti-Monotonicity** An *anti-monotonic* constraint is a formula  $\varphi_{\text{antim}}(I)$  over paths which ignores the left-hand and right-hand children of internal nodes and satisfies:

$$\forall I \subseteq I' : \varphi_{\text{antim}}(I) \rightarrow \varphi_{\text{antim}}(I').$$

Minimum support is an anti-monotonic constraint. The constraint in equation (2.2) is an example of a constraint which is not anti-monotonic. If an anti-monotonic constraint is used as leaf constraint, the internal nodes will also satisfy the constraint. Internal node constraints can also be anti-monotonic if they only have one itemset as parameter; for instance, the impurity constraint (see equation (2.1)) is anti-monotonic; however, note that this constraint will usually not be used as a leaf constraint. Hence, we can distinguish internal and leaf anti-monotonic constraints; the one type will be denoted with  $\varphi_{\text{internal,antim}}$ , the other with  $\varphi_{\text{leaf,antim}}$ .

Constraints of these types can freely be combined. For instance, if we are searching for trees in which leaves are frequent, internal nodes are not pure and leaves have strong majority classes, we have a problem in which:

$$\begin{aligned} \varphi(T) = & \bigwedge_{I \in \text{internal}(T)} (|tid(I)| \neq \max_{c \in C} |tid_c(I)|) \\ & \bigwedge_{I \in \text{leaves}(T)} (((|tid(I)| - \max_{c \in C} |tid_c(I)|) \leq \text{maxfreq}) \wedge (|tid(I)| \geq \text{minfreq})). \end{aligned}$$

We can categorize these constraints as follows according to their properties:

$$\begin{aligned} \varphi_{\text{internal}}(I, I_1, I_2) &= (|tid(I)| \neq \max_{c \in C} |tid_c(I)|), \\ \varphi_{\text{leaf}}(I) &= ((|tid(I)| - \max_{c \in C} |tid_c(I)|) \leq \text{maxfreq}) \wedge (|tid(I)| \geq \text{minfreq}), \\ \varphi_{\text{leaf,antim}}(I) &= (|tid(I)| \geq \text{minfreq}). \end{aligned}$$

Note that some constraints (for example  $|tid(I)| \geq \text{minfreq}$ ) may belong to multiple categories.

**Optimization Constraints** If a constraint can be written as

$$\varphi(T) = (g(T) \leq \theta),$$

where  $g(T)$  is an integer optimization criterion and  $\theta$  is a threshold value, the constraint is called an optimization constraint. Properties of optimization criteria, such as additivity and structure independence, extend to optimization constraints. In particular, if  $g(T)$  returns a vector of values,  $\theta$  can also be a vector of thresholds, each of which should be satisfied.

For example, in [19], we have shown how the error-based pruning (used for example as pruning measure in the C4.5 algorithm [110]) can be cast as an additive, structure independent optimization criterion; the problem of learning optimal dyadic decision trees [48] can be cast as a conjunction of constraint, one is anti-monotonic, structure dependent, conjunctive path constraint, the other one internal node constraint; the problem of learning bayesian probability estimation trees [55, 62, 46] can be cast as a anti-monotonic, structure independent minimum support constraint; the problem of learning cost-sensitive decision trees [116, 70] can be cast as an additive and structure dependent criterion which uses a conjunctive, anti-monotonic constraint, etc.

### 2.3 Building Optimal Decision Trees from Lattices

Given our categorization of the previous section, these are the requirements for this algorithm:

1. The optimization criterion must be additive.
2. The constraints must be either conjunctive over paths or based on an additive optimization criterion.
3. There should be at least one anti-monotonic path constraint.

As seen in the previous section, we decompose a query in the following components, some of which may be empty:

- the anti-monotonic leaf constraint  $\varphi_{leaf, antim}(I)$ ;
- the leaf constraint  $\varphi_{leaf}(I)$ , which includes the anti-monotonic leaf constraint;
- the internal constraint  $\varphi_{internal}(I, I_1, I_2)$ ;
- the leaf optimization criterion  $f_{leaf}(I)$ ;
- the internal optimization criterion  $f_{internal}(I, I_1, I_2)$ ;
- the leaf optimization constraint  $g_{leaf}(I)$ ;
- the internal optimization constraint  $g_{internal}(I, I_1, I_2)$ ;
- the optimization constraint threshold(s)  $\theta$ .

The algorithm, which we called DL8 (Decision Trees from Lattices), is based on the link between itemset mining and decision tree learning. In this section, we first discuss how to compute trees from itemset lattices. Next, we discuss how to compute these lattices, where we consider two options:

1. Building the trees from pre-computed itemsets (the lattice is computed *before* building the decision trees).
2. Integrating itemset mining into the decision tree construction (the lattice is computed *while* building the decision trees).

#### 2.3.1 Decision Trees from Lattices

The algorithm for constructing decision trees from lattices is given in Algorithm 1. Its main component is the DL8-RECURSIVE procedure, which is called for an itemset and computes decision trees for that itemset. The main reasons why DL8 is more efficient than naïve enumeration algorithms are:

- We optimize the left-hand and right-hand branch of a node in a tree independently from each other, hence avoiding that we enumerate all possible combinations of sub-trees for the left-hand and right-hand branch of a test.
- When we compute a tree for an itemset, we store the result, and reuse it later on, hence avoiding that we compute the same result for other possible orders in which the same tests can occur in a path.

---

**Algorithm 1** DL8( $\varphi_{leaf, antim}, \varphi_{leaf}, \varphi_{internal}, f_{leaf}, f_{internal}, g_{leaf}, g_{internal}, \theta$ )

---

```

1:  $\mathcal{T} \leftarrow \text{DL8-RECURSIVE}(\emptyset)$ 
2: Compute  $\text{argmin}_{T \in \mathcal{T}} T.f$ 
3:
4: procedure DL8-RECURSIVE( $I$ )
5:   if DL8-RECURSIVE( $I$ ) was computed before then
6:     return stored result
7:   end if
8:   initialize  $\mathcal{T}$  to be an empty associative array with domain  $\{0, \dots, \theta\}$ 
9:   if  $\varphi_{leaf}(I)$  then
10:     $T.tree \leftarrow \text{leaf}(c(I))$ 
11:     $T.f \leftarrow f_{leaf}(I)$ 
12:     $T.g \leftarrow g_{leaf}(I)$ 
13:    if  $T.g \leq \theta$  then
14:       $\mathcal{T}[T.g] = T$ 
15:    end if
16:   end if
17:   for all  $i \in \mathcal{I}$  do
18:     if  $\varphi_{internal}(I, I \cup \{i\}, I \cup \{\neg i\})$  and
19:        $\varphi_{leaf, antim}(I \cup \{i\})$  and  $\varphi_{leaf, antim}(I \cup \{\neg i\})$  then
20:        $\mathcal{T}_1 \leftarrow \text{DL8-RECURSIVE}(I \cup \{i\})$ 
21:        $\mathcal{T}_2 \leftarrow \text{DL8-RECURSIVE}(I \cup \{\neg i\})$ 
22:       for all  $T_1 \in \mathcal{T}_1, T_2 \in \mathcal{T}_2$  do
23:          $T.tree \leftarrow \text{node}(i, T_1.tree, T_2.tree)$ 
24:          $T.f \leftarrow f_{internal}(I, I \cup \{i\}, I \cup \{\neg i\}) + T_1.f + T_2.f$ 
25:          $T.g \leftarrow g_{internal}(I, I \cup \{i\}, I \cup \{\neg i\}) + T_1.g + T_2.g$ 
26:         if  $T.g \leq \theta$  and ( $\mathcal{T}[T.g]$  is empty or  $\mathcal{T}[T.g].f > T.f$ ) then
27:            $\mathcal{T}[T.g] = T$ 
28:         end if
29:       end for
30:     end if
31:   end for
32:   store  $\mathcal{T}$  as the result for  $I$  and return  $\mathcal{T}$ 
33: end procedure

```

---

- We do not recurse the search when the anti-monotone constraints are not satisfied.

The correctness of this approach follows from the following facts.

- We consider queries which are additive and conjunctive, and hence, we can evaluate optimization criteria and constraints for the left-hand and right-hand branch of a node in a tree independently from each other.
- All constraints and optimization criteria are computed for itemsets, independent of the order of the items in these sets.
- If an anti-monotonic constraint is not satisfied for a path, any tree which contains this path cannot be a solution to the query either.

If  $\kappa$  is the number of edges in a lattice, the complexity of the algorithm is  $\Theta(\kappa)$ , as we consider every edge in this lattice exactly once.

In our algorithm, we use several data structures. The main data structure is the one in which the lattice is stored. For every itemset  $I$  we have an associative data structure  $\mathcal{T}$  which allows us to associate a tree and its attributes to a vector of integers. In case no optimization constraints are specified, this structure  $\mathcal{T}$  contains at most one tree. Note that at the implementation level we do not need to store associated trees in their entirety: it is sufficient to only store the roots of these trees, as the subtrees can be recovered from the lattice recursively, by searching for the trees associated to the left-hand and right-hand branch of an internal node.

In more detail, our algorithm works as follows:

**Line 8:** For each possible value that the optimization constraint can take we will store one associated tree. Initially, this data structure is empty. Note that we require an optimization criterion that is used as optimization constraint to have an integer codomain.

**Line 9–14:** In case the itemset corresponds to a possible leaf, we initialize this leaf and its statistics  $T.f$  and  $T.g$ .

**Line 17–31:** We iterate over all possible tests to split the examples further.

**Line 19:** For a possible test, we determine whether or not we create a tree in which this test is a valid internal node; furthermore we determine if we create two paths that can be part of a tree in which the anti-monotonic constraints are satisfied.

**Line 20–21:** If we can satisfy the constraints, we determine the best trees for the left-hand and right-hand branches, independently from each other; both calls return sets of trees, each tree associated to a vector of integers, each integer representing a possible value of one of the optimization constraints for the tree (if we do not have an optimization constraint, each set contains at most one tree).

**Line 22–29:** We consider all combinations of left-hand and right-hand trees.

**Line 27:** The optimization constraint of the generated tree is evaluated; if the best known tree for this constraint value is improved, we store the new tree. We only need to store intermediate trees for which the optimization constraint is not higher than the threshold value, as the additivity means that other sub-trees cannot be part of the final tree.

### 2.3.2 Computing Lattices Beforehand

While DL8 is executing, it needs to evaluate constraints based on the data. In this section we study the following question: assuming that we would like to use an itemset mining algorithm beforehand to find the itemsets and their properties in the data, which constraints should we use in this itemset miner? In other words, how do we push the decision tree mining constraints in the itemset mining process?

First, let us consider why we may be interested in separating the execution of DL8 from the itemset mining process. We believe there could be two reasons for this.

1. There are many optimized itemset mining algorithms; by using these, we exploit these optimizations, and reduce implementation efforts.

2. We might consider decision tree construction as one part of an interactive data analysis process, in which it could be of interest to know in which cases we can reuse a set of itemsets to build multiple decision trees.

The main class of constraints used by itemset miners is the class of *anti-monotonic* constraints. We can see that if we find all itemsets satisfying  $\varphi_{leaf,antim}(I)$ , we find sufficiently many itemsets to build decision trees for the case that  $\varphi_{leaf,antim}(I)$  is the leaf constraint. A more interesting question is the reverse question: are all these itemsets needed? The following example illustrates that this is not the case. Assume that  $\{A\}$  is a frequent itemset, but  $\{\neg A\}$  is not; then no tree will contain a test on feature  $A$ , as one of the branches resulting from this test will lead to an infrequent leaf. Consequently, itemset  $\{A\}$ , even though frequent, is redundant. The following explains how we can characterize the itemsets *relevant* to decision trees induction.

If we consider the DL8 algorithm, an itemset  $I = \{i_1, \dots, i_n\}$  is needed only if there is an order  $[i_{k_1}, i_{k_2}, \dots, i_{k_n}]$  of the items in  $I$  (which corresponds to an order of recursive calls of DL8-RECURSIVE) such that for none of the proper prefixes  $I' = [i_{k_1}, i_{k_2}, \dots, i_{k_m}]$  ( $m < n$ ) of this order:

- the  $\varphi_{internal}(I', I' \cup \{i_{k_{m+1}}\}, I' \cup \{\neg i_{k_{m+1}}\})$  predicate is false;
- the conjunction  $\varphi_{leaf,antim}(I' \cup \{i_{k_{m+1}}\}) \wedge \varphi_{leaf,antim}(I' \cup \{\neg i_{k_{m+1}}\})$  is false.

**Definition 2** Let  $\varphi_{leaf,antim}$  be an anti-monotonic constraint and  $\varphi_{internal}$  be an internal constraint. Then the relevance of an itemset  $I$ , denoted by  $rel(I)$ , is defined by

$$rel(I) = \begin{cases} \varphi_{internal}(I), & \text{if } I = \emptyset & \text{(Case 1)} \\ \text{true,} & \text{if } \exists i \in I \text{ such that} \\ & rel(I - i) \wedge \varphi_{internal}(I - i, I, I - i \cup \{\neg i\}) \wedge \\ & \varphi_{leaf,antim}(I) \wedge \varphi_{leaf,antim}(I - i \cup \neg i) & \text{(Case 2)} \\ \text{false,} & \text{otherwise} & \text{(Case 3)} \end{cases}$$

**Theorem 2.1** Let  $\mathcal{L}_1$  be the set of itemsets stored by DL8, and let  $\mathcal{L}_2$  be the set of itemsets  $\{I \subseteq \mathcal{I} \mid rel(I) = \text{true}\}$ . Then  $\mathcal{L}_1 = \mathcal{L}_2$ .

If we assume that the internal constraint is also anti-monotonic, *relevance* can also be used in itemset miners that exploit anti-monotonic constraints.

**Theorem 2.2** If both the internal constraint and the leaf constraint are anti-monotonic, itemset relevance is an anti-monotonic property.

It is relatively easy to integrate the computation of relevance in both breadth-first and depth-first frequent itemset mining algorithms, as long as the order of itemset generation is such that all subsets of an itemset  $I$  are enumerated before  $I$  is enumerated itself.

We implemented two versions of DL8 in which the relevance constraints are pushed in the frequent itemset mining process: DL8-APRIORI, which is based on APRIORI [42], and DL8-ECLAT, which is based on ECLAT [133].



### 2.3.3 Computing Lattices on the Fly

The second option is to access the data while building decision trees. One reason for doing this could be to avoid possible overhead caused by traversing the lattice multiple times. Another, more important, reason involves the possibility to use closed itemsets effectively.

The main observation that we exploit to this purpose is that if we are dealing with a *structure independent* query we can restrict our attention to an even smaller set of itemsets than the relevant itemsets.

The main reason for this is that if two itemsets  $I$  and  $I'$  cover the same set of examples (i.e.,  $tid(I) = tid(I')$ ), and the query is structure independent, the tree(s) we find for both itemsets must be the same. To reduce the number of itemsets that we have to store, we should avoid storing such duplicate sets of results.

To ensure that results are re-used between itemsets covering exactly the same examples, we propose to compute for every itemset its *closure*. The closure of an itemset  $I$  is the largest itemset that all transactions in  $tid(I)$  have in common. More formally, let  $items$  be the function which computes

$$items(tids) = \bigcap_{k \in tids} t_k$$

for a TID-set  $tids$ , then the *closure* of itemset  $I$  is the itemset  $items(tid(I))$ . An itemset  $I$  is called *closed* iff  $I = items(tid(I))$  [108]. If  $tid(I_1) = tid(I_2)$  it is easy to see that also  $items(tid(I_1)) = items(tid(I_2))$ .

We can use this observation by modifying DL8: instead of associating decision trees to itemsets, we associate decision trees to closed itemsets. We change line 5 such that it checks if a decision tree has already been computed for the closure of  $I$ ; in line 32, we associate computed decision tree(s) to the closure of  $I$  instead of to  $I$  itself. We refer to this algorithm as DL8-CLOSED.

In practice this means that we build a data structure of closed itemsets instead of ordinary itemsets. Lattices of closed itemsets are also known as *concept lattices*; closed itemsets are also known as *formal concepts*, and have been studied extensively in the literature [76]. In principle, one could also develop a step-wise approach in which one first computes closed itemsets and subsequently mines decision trees. However, in our algorithm we do not only need the closed itemsets; we also need the relationships between them, i.e., if we add an item to an itemset we need to know what the closure of the resulting itemset is. In other words, we do not only need to know the formal concepts, we also need to know the edges in the Hasse diagram of these itemsets. Storing these edges would not only increase the memory requirements of our algorithm, determining them in a post processing step is also not straightforward: a naïve algorithm for computing this diagram would take quadratic time, while also less naïve recent algorithms (such as [47]) require significant computation times. An approach in which itemsets are mined and decision trees are built at the same time hence seems more promising. The remainder of this section is devoted to an outline of the choices that we made in the integrated approach that we used in our experiments. This approach builds on choices that are commonly made in closed itemset mining algorithms.

The main idea is that during the search, we keep track of those items and transactions that are ‘active’. As parameters to DL8-RECURSIVE we add the following:

- the item  $i$  that was last added to  $I$ ;

- a set of *active items*, which includes item  $i$ , and represents all tests that can still be added to the itemset  $I - i$ ;
- a set of *active transaction identifiers* representing  $tid(I - i)$ ;
- the set of all items  $\mathcal{C}$  that are in the closure of  $I - i$ , but are not part of the set of active items.

In the first call to DL8-RECURSIVE, all items and transactions are active. At the start of each recursive call (before line 5 of DL8-RECURSIVE is executed) we scan each active transaction, and test if it contains the last added item  $i$ ; for each active transaction that contains item  $i$ , we determine which other active items it contains. We use this scan to compute the frequency of the active items, and build the new set of active transaction identifiers  $tid(I)$ . Those active items of which the frequency equals that of  $I$ , are added to the closure  $\mathcal{C}$ . If it turns out we have encountered this closure before, we return the corresponding previously computed result. Otherwise, we now build a new set of active items. For every item we determine if  $\varphi_{leaf,antim}(I \cup \{i\})$ ,  $\varphi_{leaf,antim}(I \cup \{\neg i\})$  and the internal constraint  $\varphi_{internal}(I, I \cup \{i\}, I \cup \{\neg i\})$  are true; if so, we add the item to the new set of active items. In line 17 we traverse the set of active items. In line 20 and 21 the updated sets of active transactions and active items are passed to the recursive calls. By computing the closure of every itemset, we traverse the Hasse diagram of closed itemsets.

Our approach for maintaining sets of active transactions is akin to the idea of maintaining projected databases that is implemented in ECLAT [133] and FP-GROWTH [81]. In contrast to these algorithms, we know in our case that we have to maintain projections that contain both an item  $i$  and its negation  $\neg i$ . As we know that  $|tid(I)| = |tid(I \cup i)| + |tid(I \cup \neg i)|$ , it is less beneficial to maintain TID-sets as in ECLAT, and we prefer a solution in which we call DL8-RECURSIVE with the set of active transactions  $tid(I - i)$  instead of  $tid(I)$ . We project a transaction set by reordering the transactions in an array. Consequently, the memory use of our algorithm is determined by the amount of memory that is needed to store the database and the closed itemsets with associated information. Per closed itemset we only store the associative array  $\mathcal{T}$  for later retrieval; to reduce memory demands, we do not store support values, edges of the Hasse diagram, or TID sets. A tree in the associative array is only represented by its root node, as any subtrees can be recovered recursively from information associated to other itemsets. The information that we store for every itemset is hence only determined by the optimization criteria that are used; if we assume the query given, the information stored per itemset is constant. Consequently, the memory use is  $\theta(|D| + |\mathcal{S}|)$ , where  $|\mathcal{S}|$  is the size of a data structure storing all closed itemsets.

Even though we hence attempt to limit the memory required by our algorithm, it should be repeated that the number of closed itemsets can be exponential in the size of the database; in practice the complexity remains high.

## 2.4 Discussion

The original paper that describe this work [19] also shows how to improve the efficiency of the DL8 algorithm by adding redundant constraints. Furthermore, the experiments show that: i) these constraints can improve the resulting accuracy of a tree; ii) an exact algorithm can indeed give significantly better results than a heuristic learner if the optimisation criterion is well-defined; iii) exact results allow to study the behavior of the trees with respect to constraints.

It is still an open question how efficient decision tree miners could become if they were thoroughly integrated with algorithms such as LCM, FP-Growth, or algorithms developed within the formal concept analysis community for processing (concept) lattices. Our investigations showed that high runtimes are however not as much a problem as the amount of memory required for storing huge amounts of itemsets. A challenging question for future research is what kind of condensed representations could be developed to represent the information that is used by DL8 more compactly; an alternative could be to trade space and time complexity more carefully.

DL8 can be seen as a relatively cheap type of post-processing on a set of itemsets. In particular, it does not require access to the training data when the model is constructed, in contrast to other approaches that use patterns for classification. Hence DL8 suits itself perfectly for interactive data mining on stored sets of patterns. This means that DL8 might be a key component of inductive databases [83] that contain both patterns and data.

### 3 (Dynamic) Plane Graph Mining

In this section we investigate how (dynamic) plane graphs can be efficiently and exhaustively mined under a set of constraints in order to extract meaningful subgraph patterns. To do this, we take advantage of a polynomial plane subgraph isomorphism algorithm ([67]) to efficiently find the occurrences of each pattern. This work has originally been done for in application context where patterns represented objects to track in videos [20, 7][37]. Each frame of the video was considered as a (plane) graph and a video was either considered as a set of graphs (one per frame) or as *sequence* of graphs forming a dynamic graph where both nodes and edges could evolve over time. In this particular context, some constraints that could guarantee the temporal and spatial coherence of the patterns were particularly relevant to find a restricted yet meaningful set of graph patterns. We thus developed two distances, a temporal one and a spatial one, and use them to output only patterns that meet some spatial and temporal constraints. The representation used to transform a video into a (dynamic) graph is given in Chapter 4. In this section, we only consider algorithmic contributions and, in particular, we describe a plane graph mining algorithm called PLAGRAM, a dynamic plane graph mining algorithm called DYPLAGRAM, and one that directly implements the spatio-temporal constraints called DYPLAGRAM.ST.

#### 3.1 Definitions

##### 3.1.1 Plane Graphs

Graphs are powerful mathematical tools that are used to model relationships among a set of elements. Simple graphs are defined as a pair of elements  $G = \langle V, E \rangle$  where  $V = \{v_1, \dots, v_n\}$  is a set of nodes and  $E \subseteq V \times V$  a set of edges connecting them. If there is an edge between each pair of nodes, the graph is said to be *complete*. Often, labels are given to nodes and edges to model more precisely the features of the elements and their relationships. Those graphs are called *labeled* or *attributed graphs*.

**Definition 3 (Labeled Graph)** *A labeled graph is a graph  $G = \langle V, E, L \rangle$  where  $V = \{v_1, \dots, v_n\}$  is a set of nodes,  $E \subseteq V \times V$  a set of edges connecting pairs of nodes  $(v_i, v_j)$ . The labeling function  $L : V \cup E \rightarrow \mathbb{N}$  maps each edge and each node of the graph to a label.*

Note that, in the case of directed graphs, each edge is an ordered pair of nodes  $(v_i, v_j)$ , while in the case of undirected graphs the pairs of nodes are unordered. Graph mining algorithms look for *subgraphs* that appear frequently in the database.

**Definition 4 (Subgraph)** *Given two graphs  $G = \langle V, E \rangle$  and  $G' = \langle V', E' \rangle$ ,  $G'$  is a subgraph of  $G$  if and only if  $V' \subseteq V$  and  $E' \subseteq E$ .*

**Definition 5 (Induced Subgraph)** *Given two graphs  $G = \langle V, E \rangle$  and  $G' = \langle V', E' \rangle$ ,  $G'$  is an induced subgraph of  $G$  if and only if  $V' \subseteq V$  and  $E' = E \cap V' \times V'$ .*

If a subgraph  $G'$  of another graph  $G$  is *complete*,  $G'$  is called a *clique* of  $G$ .

A *path* in a graph  $G = \langle V, E \rangle$  is a sequence such that consecutive nodes are connected by edges. A *finite path* is a path with a finite number of nodes. Its first node is called the *starting node* and its last one the *ending node*. If the *starting node* and the *ending node* are the same the path is called a *cycle*. A *path* is said to be *simple* if it never passes twice through the same node.

Most of the time graph mining algorithms deal with *connected graphs*, i.e., graphs in which there exist at least one *path* connecting each pair of nodes.

*Trees* are also particular graphs that are the focus of many graph mining algorithms. A tree is a *connected* graph with no *cycle*. A *rooted tree*, is a *tree* for which one node is singled out as the *root*. If no *root* is designated, the *tree* is called a *free tree*.

Planar graph are graphs that can be drawn in the plane without any of their edges crossing. A *plane* graph is a planar embedding of a planar graph. Each *plane graph* is composed of a set of faces.

**Definition 6 (Face)** *Given a plane graph, a face is a connected region of the plane which is bounded by a cycle of edges. It is represented by the list of nodes encountered when following the circuit such that the face is always on the left-hand side.*

**Definition 7 (Plane graph)** *A plane graph is a tuple  $G = (V, E, F, f_e, L)$  where  $V$  is a set of nodes,  $E$  is a set of edges,  $F$  is a set of faces and  $L$  is a labeling function on  $V \cup E$ . The unbounded region  $f_e$  in the embedding of the graph is called the outer face of the graph. The other faces are called internal faces.*

For example, Figure 2.11 presents three plane graphs and the graph  $g_1$  has two internal faces  $\langle 1, 2, 3 \rangle$  and  $\langle 2, 4, 5, 3 \rangle$ , and its outer face is  $\langle 1, 3, 5, 4, 2 \rangle$ .

**Definition 8 (k-connectedness)** *A plane graph is k-connected if k is the size of the smallest subset of vertices such that the graph becomes disconnected if you delete them.*

The k-connectedness can also be defined using Menger's theorem [98].

**Theorem 2.3 (Menger's theorem)** *A graph  $G$  is k-connected if and only if every pair of vertices is connected by k internally disjoint paths.*

Note that, using Menger's theorem, we can see that in a 2-connected graph each pair of distinct nodes is connected by at least 2 internally disjoint paths forming a simple cycle (no node or edge is used more than once). Therefore a plane graph is 2-connected if each face (and in particular the outer face) is a simple cycle.

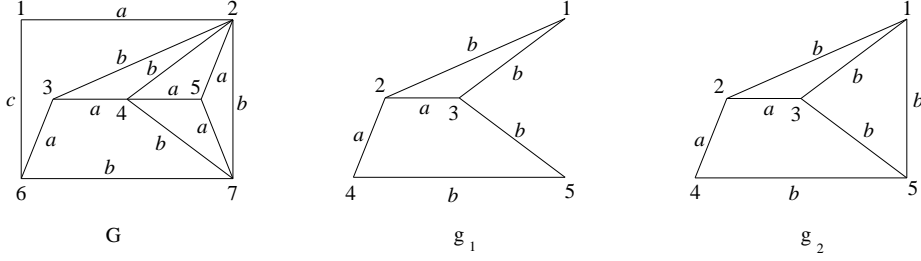


Figure 2.11: Plane graphs. The edge labels are in  $\{a, b, c\}$  and we assume that all node labels are equal to  $a$  (not represented). Graph  $g_1$  is a plane subgraph of  $G$  while  $g_2$  is not.

### 3.1.2 Isomorphism and Subgraph Isomorphism

Graph isomorphisms are used to assess if two graphs are equivalent, i.e., if we can find a mapping between the nodes that preserves the edges and the labels.

**Definition 9 (Graph Isomorphism)** Two graphs  $G = (V, E, L)$  and  $G' = (V', E', L')$  are said isomorphic if and only if there exists a bijective function  $f : V \rightarrow V'$  such that

- $\forall v \in V, L(v) = L'(f(v))$ ,
- $\forall (v_1, v_2) \in V \times V, (v_1, v_2) \in E \Leftrightarrow (f(v_1), f(v_2)) \in E'$
- $\forall (v_1, v_2) \in E, L(v_1, v_2) = L'(f(v_1), f(v_2))$ .

**Definition 10 (Subgraph Isomorphism)** A graph  $G_1$  is said to be subgraph isomorphic to a graph  $G_2$ , noted  $G_1 \subseteq G_2$ , if there exists a subgraph  $G'_2$  of  $G_2$  such that  $G_1$  is isomorphic to  $G'_2$ .

A subgraph isomorphism of a pattern  $P$  in a graph  $G$  is called an *occurrence* of  $P$  in  $G$ .

Those definitions can be extended to the specific case of plane graphs. As for general graphs, a plane graph is a plane subgraph of another plane graph if there exists a correspondence between their nodes which preserves the labels and the edges, except that the correspondence between nodes should also preserve the internal faces (if the outer face is also preserved, then the graphs are plane isomorphic).

**Definition 11 (Plane subgraph isomorphism)** Let  $G = (V, E, F, f_e, L)$  and  $G' = (V', E', F', f'_e, L)$  be two plane graphs. Graph  $G'$  is plane subgraph isomorphic to  $G$  (or  $G'$  is a plane subgraph of  $G$ ), denoted  $G' \subseteq G$ , if there is an injective function  $f : V \rightarrow V'$  such that:

- $\forall v \in V, L(v) = L'(f(v))$ ,
- $\forall (v_1, v_2) \in E, L(v_1, v_2) = L'(f(v_1), f(v_2))$ ,
- $\forall$  internal faces  $F = \langle v_1, \dots, v_k \rangle$  of  $G$ ,  $f(F) = \langle f(v_1), \dots, f(v_k) \rangle \in F'$ .

As we can see in Figure 2.11, the internal faces  $\langle 2, 4, 5 \rangle$ ,  $\langle 2, 5, 7 \rangle$  and  $\langle 4, 5, 7 \rangle$  of  $G$  are not present in  $g_2$ , therefore it is not plane subgraph isomorphic to  $G$ .

From this an occurrence of a plane graph in a larger graph is defined as follows:

**Definition 12 (Occurrence of a plane graph in a larger graph)** *Let two plane graphs  $G$  and  $G'$ . If  $G'$  is plane subgraph isomorphic to  $G$ , the corresponding injective function  $f$  is called an occurrence of  $G'$  in  $G$ .*

**Example 2.4** *In Figure 2.11, graph  $g_1$  is a plane subgraph of  $G$ . The internal faces  $\langle 1, 2, 3 \rangle$  and  $\langle 2, 4, 5, 3 \rangle$  of  $g_1$  correspond, respectively, to faces  $\langle 2, 3, 4 \rangle$  and  $\langle 3, 6, 7, 4 \rangle$  of  $G$ , with  $f(1) = 2$ ,  $f(2) = 3$ ,  $f(3) = 4$ ,  $f(4) = 6$  and  $f(5) = 7$ . Graph  $g_2$  has three internal mutually adjacent faces, one with four edges and two with three edges. Since such configuration of faces does not exist in  $G$ ,  $g_2$  is not a plane subgraph of  $G$ .*

### 3.1.3 Support and Frequency of a Subgraph Pattern

The *support* of a *subgraph pattern*  $P$  in a database  $\mathcal{D}$  corresponds to the number of graphs  $G_i \in \mathcal{D}$  to which it is *subgraph isomorphic*:

$$\text{support}_{\mathcal{D}}(P) = |\{G_i | P \subseteq G_i \text{ and } G_i \in \mathcal{D}\}|.$$

The *frequency* of a pattern  $P$  is the ratio of its *support* divided by the size of the database:

$$\text{frequency}_{\mathcal{D}}(P) = \frac{\text{support}_{\mathcal{D}}(P)}{|\mathcal{D}|}.$$

### 3.1.4 Dynamic Plane Graph

We define a dynamic plane graph as an ordered set of graphs.

**Definition 13 (Dynamic plane graph)** *A dynamic plane graph  $\mathcal{D}$  is an ordered set of plane graphs  $\{G_1, G_2, \dots, G_n\}$ . Each node of these graphs is associated to spatial coordinates  $(x, y)$ .*

Building on the definition of an occurrence of a plane graph in a larger graph (see definition 12) we define an occurrence of a plane graph in a dynamic plane graph and its frequency.

**Definition 14 (Occurrences of a plane graph in a dynamic graph)** *Given a plane graph  $P$  and a dynamic graph  $\mathcal{D} = \{G_1, \dots, G_n\}$ , the set of occurrences of  $P$  in  $\mathcal{D}$  is defined as  $\text{Occ}(P) = \{(i, f) \mid f \text{ is an occurrence of } P \text{ in } G_i\}$ .*

**Definition 15 (Barycenter of an occurrence)** *The barycenter of an occurrence is the average of the coordinates of its nodes.*

**Definition 16 (Frequency of a plane graph in a dynamic graph)** *The frequency  $\text{freq}(P)$  of a plane graph  $P$  in a dynamic graph  $\mathcal{D}$  is the number of graphs  $G_i \in \mathcal{D}$  in which there is an occurrence of  $P$ , i.e.,  $|\{i \mid \exists f, (i, f) \in \text{Occ}(P)\}|$ .*

### 3.1.5 Occurrence Graph and Spatio-Temporal Patterns

To define a frequency that takes into account the spatio-temporal distance between the occurrences, we define the notion of an occurrence graph in which occurrences of the same pattern that are close to one another are linked. Then, we define spatio-temporal patterns in this occurrence graph and the associated frequency.

Note that we use two different definitions of an occurrence graph, one for PLAGRAM and DYPLAGRAM, and one for DYPLAGRAM\_ST.

**Definition 17 (Occurrence graph and Spatio-temporal pattern)** *Two occurrences of a plane graph  $P$  in a dynamic graph  $\mathcal{D}$ ,  $o = (i, f)$  and  $o' = (i', f')$ , are close if the distance between their barycenters is lower than a spatial threshold  $\epsilon$  and their temporal distance  $|i' - i|$  is lower than a time threshold  $\tau$ . Then, given a plane graph  $P$  and a dynamic graph  $\mathcal{D}$ , we define the occurrence graph of  $P$  as a graph where the set of nodes is  $Occ(P)$  and the set of edges is  $\{(o, o') \mid o \text{ is close to } o'\}$ . In the rest of this document, each connected component of the occurrence graph of  $P$  is called a spatio-temporal pattern  $S$  based on  $P$ . The frequency of a spatio-temporal pattern corresponds to the number of frames in which it has at least one occurrence.*

**Definition 18 (Frequency of a spatio-temporal pattern)** *The frequency of a spatio-temporal pattern  $S$  in a dynamic graph  $\mathcal{D}$ , denoted  $freq_{st}(S)$ , is  $|\{i \mid \exists f, (i, f) \in S\}|$ .*

Given two plane graphs such that  $P' \subseteq P$ , if there is an occurrence of  $P$  in  $G_i$ , then there is also an occurrence of  $P'$  in  $G_i$ . Thus  $freq(P) \leq freq(P')$  and, therefore, if  $P'$  is not frequent, then neither is  $P$ . Given this behavior, we say that  $freq$  has the anti-monotonicity property. Such property can certainly be exploited to prune non-promising candidate subgraphs, as in classical graph mining algorithms.

However, when defining the occurrence graph as in definition 17,  $freq_{st}$  is not anti-monotone. Suppose that two occurrences  $a$  and  $b$  of  $P$  are close to each other, leading to a single frequent spatio-temporal pattern  $S$ . Conversely, two occurrences  $a' \subseteq a$  and  $b' \subseteq b$  of  $P'$  may be far from each other, possibly resulting in two non-frequent spatio-temporal patterns  $S'$  and  $S''$ . In other words, two spatio-temporal patterns  $S'$  and  $S''$  based on  $P'$  may be infrequent, while the spatio-temporal pattern  $S$  based on  $P$  is frequent. This is illustrated in Figure 2.12.

Nevertheless, the frequency of a spatio-temporal pattern  $S$  based on a plane graph  $P$  (i.e.,  $freq_{st}(S)$ ) can be upper bounded with two anti-monotone measures as follows:

$$freq_{st}(S) \leq freq_{seq}(P) \leq freq(P),$$

where  $freq_{seq}(P)$  is the *subsequence frequency* of  $P$  defined below.

**Definition 19 (Subsequence frequency)** *The subsequence frequency of a plane graph  $P$  in  $\mathcal{D}$ , denoted  $freq_{seq}(P)$ , is defined as the size of the longest subsequence  $G_{i_1}, G_{i_2}, \dots, i_1 < i_2 < \dots$  of  $\mathcal{D}$  such that*

- (a) for all  $j$ ,  $G_{i_j}$  contains an occurrence of  $P$  and
- (b) for all  $j$ ,  $i_{j+1} - i_j$  is lower than the time threshold  $\tau$ .

Observe that  $freq_{seq}(P)$  is an upper-bound on  $freq_{st}(S)$ , since the sequence of the  $G_i$ s that contains an occurrence of  $S$  satisfies (a) and (b) in Definition 19. Moreover, if

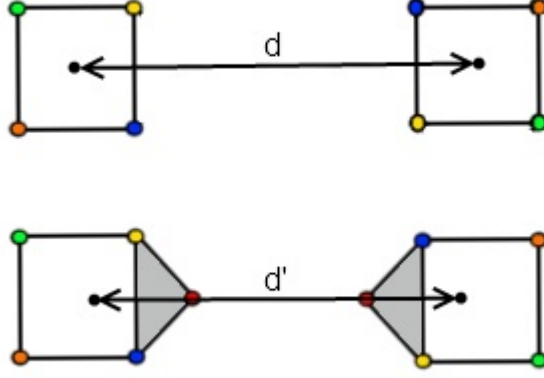


Figure 2.12: The euclidean distance between the barycenters of the two occurrences of the sub-pattern (upper part of the image) is higher than the one of the super-pattern in the bottom. This means the occurrences of the super-pattern can be close while the corresponding ones of the sub-pattern are not, resulting in a super-pattern that can be frequent, with respect to  $\text{freq}_{st}$ , while its sub-pattern is not

$P' \subseteq P$  then any sequence of  $G_i$ s satisfying (a) and (b) for pattern  $P$  also satisfies them for pattern  $P'$ .  $\text{freq}_{seq}(P) \leq \text{freq}_{seq}(P')$  and therefore  $\text{freq}_{seq}$  has the anti-monotonicity property.

While  $\text{freq}_{seq}$  has the anti-monotonicity property, it only accounts for the temporal constraint. To "give" the anti-monotonicity property to  $\text{freq}_{st}$  we need to redefine the occurrence graph so that it is not possible anymore for two distinct spatio-temporal patterns, based on the same pattern, to merge in a single spatio-temporal pattern composed of more occurrences than the two initial ones. To do so we used a different spatial distance than the euclidean distance between the barycenters of the occurrences. Instead we measure the distance between each node of one occurrence and its corresponding node in the other occurrence and keep the maximum one.

**Definition 20 (Distance between occurrences)** *The distance between two occurrences  $o = (i, f)$  and  $o' = (i', f')$  of a plane graph  $P = (V, E, F, f_e, L)$  in a dynamic graph  $\mathcal{D}$  is defined as:  $\text{dist}(o, o') = \max_{s \in V} d(f(s), f'(s))$ , where  $d$  denote the Euclidean distance between the nodes.*

This distance has an anti-monotonic property:

**Proposition 2.5** *For any pairs of patterns  $P = (V, E, F, f_e, L)$  and  $P' = (V', E', F', f'_e, L')$  such that  $P$  is a plane subgraph of  $P'$  and two occurrences  $o_1 = (f_1, i)$ ,  $o_2 = (f_2, i)$  of  $P$  and two occurrences  $o'_1 = (f'_1, i)$ ,  $o'_2 = (f'_2, i)$  of  $P'$  such that  $f_1$  is a restriction of  $f'_1$  (i.e.,  $f_1 = f'_1$  on  $V$ ) and  $f'_2$  is a restriction of  $f_2$ , then we have  $\text{dist}(o_1, o_2) \leq \text{dist}(o'_1, o'_2)$ .*

Figure 2.13 gives a graphical example of this spatial distance.

To redefine the occurrence graph so that  $\text{freq}_{st}$  has the anti-monotonicity property, we used the parent relationship on patterns, defined by the depth-first traversal of the search space performed by our mining algorithm.



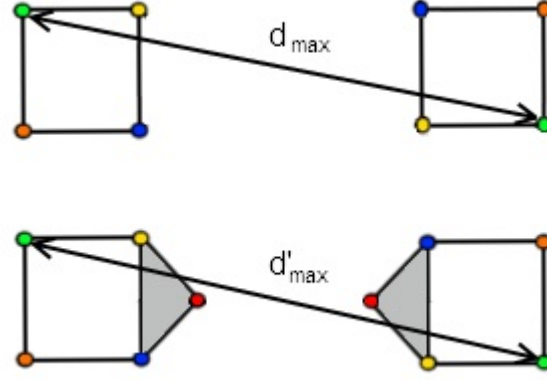


Figure 2.13: Extending a pattern cannot result in a lower spatial distance, therefore this distance has an anti-monotonic property.

**Definition 21 (Parent of a pattern and of an occurrence)** Given a pattern  $P$  with  $n \geq 2$  internal faces, the pattern  $p(P)$  with  $n - 1$  faces that can be extended into  $P$  by the addition of one face is called the parent of  $P$ . And given an occurrence  $o = (i, f)$  of  $P$ , we call the parent of  $o$  the occurrence  $p(o) = (i, f')$  such that  $f'$  is the restriction of  $f$  to the nodes of  $p(P)$ .

The definition of the parent of an occurrence is then used to define the occurrence graph. The nodes of the occurrence graph are the occurrences of a pattern and the edges connect “close” occurrences. This graph is constructed for each pattern in the mining algorithm.

**Definition 22 (Occurrence graph and Spatio-temporal pattern)** Given a spatial distance  $dist$ , a spatial threshold  $\epsilon$ , a temporal threshold  $\tau$ , a plane graph  $P = (V, E, F, f_e, L)$  and a dynamic graph  $\mathcal{D}$ , we define the occurrence graph of  $P$  as an oriented graph whose set of nodes is  $Occ(P)$ .

- If  $P$  has only one face, then there is an edge between the occurrences  $o_1 = (i, f_1)$  and  $o_2 = (j, f_2)$  such that  $j > i$  if  $0 < j - i \leq \tau$  and  $dist(o_1, o_2) \leq \epsilon \cdot (j - i)$  and there is no occurrence  $o_3 = (k, f_3)$  with  $i < k < j$  and  $dist(o_1, o_3) \leq \epsilon \cdot (k - i)$ .
- If  $P$  has more than one face, then there is an edge from  $o_1 = (i, f_1)$  to  $o_2 = (j, f_2)$  if there is an edge  $(p(o_1), p(o_2))$  in the occurrences graph of  $p(P)$  and  $dist(o_1, o_2) \leq \epsilon \cdot (j - i)$ .

A spatio-temporal pattern  $S$  based on  $P$  is a connected component of the occurrence graph of  $P$ .

This definition is such that the occurrence graph of a pattern  $P$  is always a subgraph of the occurrence graph of its parent pattern  $p(P)$  (if we identify the node  $o$  of the occurrence graph of  $P$  with the node  $p(o)$  of the occurrence graph of  $p(P)$ ). In practice,  $P$  is obtained by extending occurrences of  $p(P)$  and removing the ones that do not respect the spatio-temporal constraints. This ensures that the spatio-temporal patterns based on  $P$  get “smaller” as the pattern  $P$  grows, and this ensures that the frequency of a spatio-temporal pattern  $freq_{st}$  has the anti-monotonicity property. Beside, contrary

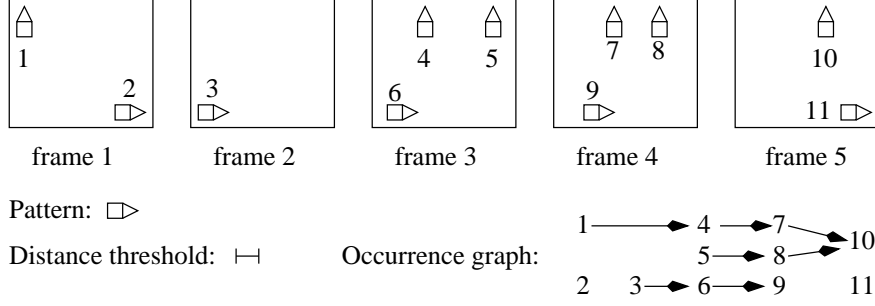


Figure 2.14: Occurrences of a pattern and occurrence graph with  $\tau = 3$ .

to definition 17, with this definition, the spatial constraint now takes into account the number of frames separating two occurrences. The idea is that if we expect an object to move 10 pixels between frames  $t$  and  $t + 1$ , we should expect it to move  $10 \times 2$  pixels between  $t$  and  $t + 2$ , therefore the spatial threshold is multiplied by the number of frames separating the two occurrences. Another improvement is the fact that we now only connect occurrences with the closest occurrences in term of time-stamp. In other words, if two occurrences  $o_1(i, f)$  and  $o_2(g, i + 1)$  are connected, no occurrence  $o_3(h, j)$  with  $j > i + 1$  can be connected to  $o_1$ , even if the spatio-temporal constraints are met. This reduces the number of edges in the occurrence graph by removing redundant transitivity edges without breaking any connected component.

Fig. 2.14 shows 11 occurrences of a pattern  $P$  in a video with five frames for  $\tau = 3$  and with  $\text{freq}(P) = 5$  (since it occurs in all 5 frames). Since occurrences 1 and 4 are close to each other, i.e., their spatial distance is lower than  $2 \times \epsilon$  and their temporal distance is  $2 \leq \tau$ , there is an edge  $(1, 4)$  in the occurrence graph of  $P$ . Conversely, the edges  $(3, 5)$  or  $(2, 11)$  do not exist in the occurrence graph, as the spatial distance between 3 and 5 or the temporal distance between 2 and 11 are too large. There are 4 spatio-temporal patterns based on  $P$ :  $S_1 = \{1, 4, 5, 7, 8, 10\}$ ,  $S_2 = \{3, 6, 9\}$ ,  $S_3 = \{2\}$  and  $S_4 = \{11\}$ . The frequencies of these patterns are:  $\text{freq}_{st}(S_1) = 4$ ,  $\text{freq}_{st}(S_2) = 3$ , and  $\text{freq}_{st}(S_3) = \text{freq}_{st}(S_4) = 1$ .

**Proposition 2.6** *Given a pattern  $P$  with more than one face, and given a spatio-temporal pattern  $S$  based on  $P$ , there is a spatio-temporal pattern  $S'$  based on the parent  $p(P)$  of  $P$  with a larger  $\text{freq}_{st}$ , i.e.,  $\text{freq}_{st}(S) \leq \text{freq}_{st}(S')$ .*

This proposition shows that, given a minimum threshold  $\sigma_{st}$  on  $\text{freq}_{st}$ , if a pattern does not have a frequent spatio-temporal pattern then any super-pattern does not either. This allows us to prune the search space of candidate patterns.

## 3.2 Mining Spatio-Temporal Patterns

**Problem Definition** Given dynamic graph  $\mathcal{D}$ , a frequency threshold  $\sigma_{st}$ , a spatial threshold  $\epsilon$  and a time threshold  $\tau$ , the problem is to compute all spatio-temporal patterns of  $\mathcal{D}$  with  $\text{freq}_{st}$  greater than  $\sigma_{st}$ .

DYPLAGRAM\_ST takes advantage of the new definition of an occurrence graph (definition 22) and can use  $\text{freq}_{st}$  to mine spatio-temporal patterns directly. PLAGRAM and DYPLAGRAM do not use  $\text{freq}_{st}$ , instead they respectively use  $\text{freq}$  and  $\text{freq}_{seq}$ . Therefore, to solve the problem defined above with those two algorithms, the idea is to first

Algorithm variant	Occurrence graph definition used	Frequency measure used	Constraints enforced during mining
PLAGRAM	def 17	freq	none
DYPLAGRAM		freq <sub>seq</sub>	temporal
DYPLAGRAM_ST	def 22	freq <sub>st</sub>	spatio-temporal

Table 2.1: Major differences between PLAGRAM, DYPLAGRAM and DYPLAGRAM\_ST

mine for all frequent graph patterns (using either freq or freq<sub>seq</sub>) and then, in a post-processing step, construct the occurrence graph of each frequent pattern to compute the spatio-temporal patterns, as described in Definition 17. The key differences between the three variants of our algorithm are summarized in table 2.1.

The rest of this subsection will give the details concerning the extension strategy used by our algorithms, the canonical codes used to avoid processing several times the same subgraph and the strategy to explore the search space. Then we give the pseudo-codes for the three variants of our approach.

### 3.2.1 Extensions

Our algorithms use a depth-first exploration strategy: each time a frequent pattern is found, it is extended into a bigger candidate pattern for further evaluation. As GSPAN, our algorithms only generate promising candidate graphs, that is, subgraphs that actually occur in  $\mathcal{D}$ . However, our extension strategy limits the number of different extensions that can be generated from a given frequent pattern, as described below.

**Definition 23 (Valid extension)** *Given a plane graph  $g$  and two nodes  $u \neq v$  on the outer face of  $g$ ,  $g$  can only be extended by the addition of a new path  $P = (u = x_1, x_2, \dots, x_k = v)$  to  $g$  between  $u$  and  $v$ . This path must lie in the outer face of  $g$ . Nodes  $x_2, \dots, x_{k-1}$  are  $(k - 2) \geq 0$  new nodes. This new graph is denoted  $g \cup P$ . Given a plane graph  $G$  such that  $g \subset G$ ,  $P$  is a valid extension of  $g$  in  $G$  if  $g \cup P \subseteq G$ .*

In other words, this definition states that any pattern graph  $g$  composed of aggregated faces can only be extended by the addition of another face lying in the outer face of  $g$ . This new face must share at least one edge with  $g$  (since  $u \neq v$ ). This restriction is related to that of GSPAN, where a graph is extended by the addition of a single edge, and only to nodes of the rightmost path of the depth-first search tree. A consequence of this extension strategy is that the generated patterns are always 2-connected (this means that for any two nodes of the pattern, there is always a cycle that contains both).

In Figure 2.15, there is only one occurrence of  $g_1$  in  $G$  and, for this occurrence, there are three valid extensions of  $g_1$  in  $G$ . Since these extensions have two edges, a new node 6 is added in the outer face of  $g_1$ . The extensions are:  $P_1 = (1, 6, 3)$  (which corresponds to 2, 5, 4 in  $G$ ),  $P_2 = (3, 6, 5)$  (corresponding to 4, 5, 7 in  $G$ ) and  $P_3 = (4, 6, 1)$  (corresponding to 6, 1, 2 in  $G$ ). Observe that the path  $P_4 = (1, 5)$  is not a valid extension since  $g_1 \cup P_4$  is the graph  $g_2$ , which is not a plane subgraph of  $G$  (see subsection 3.1.2).

Given a pattern graph  $g$  and a graph  $G_i$  in  $\mathcal{D}$ , our algorithms compute all occurrences of  $g$  in  $G_i$ . Then, for each occurrence, they generate all possible extensions. For each occurrence of  $g$  in  $G_i$  and from each node of the external face of  $g$ , there is only one possible extension.

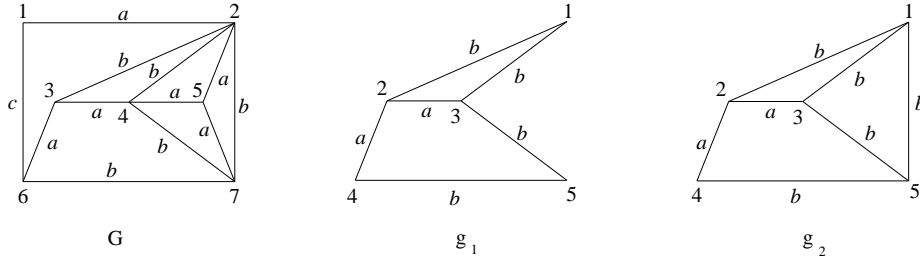


Figure 2.15: Plane graphs. The edge labels are in  $\{a, b, c\}$  and we assume that all node labels are equal to  $a$  (not represented). Graph  $g_1$  is a plane subgraph of  $G$  while  $g_2$  is not.

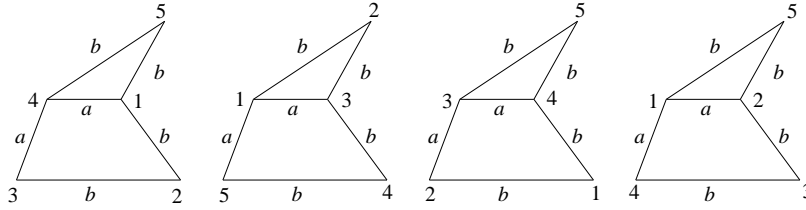


Figure 2.16: Four copies of  $g_1$  of Figure 2.15 with node indices corresponding, respectively, to the codes  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\delta$  in Table 2.2.

### 3.2.2 Graph Codes

To avoid multiple generations of the same pattern, the graphs are represented by canonical codes. Therefore, to find the frequent patterns, our algorithms explore a code search space. Here, we define these new codes. Next, we present important properties of the code search space.

A code for a plane graph  $g$  is a sequence of the edges of  $g$ . Each edge is represented by a 5-tuple  $(i, j, L(i), L(i, j), L(j))$ , where  $i$  and  $j$  are the indices of the nodes (from 1 to  $n$ , where  $n$  is the number of nodes in  $g$ ). The nodes are numbered as they first appear in the code.

#### Definition 24 (Valid code for a plane graph)

- If  $g = (V, N, F, f_e, L)$  is a plane graph with only one internal face  $\langle v_0, \dots, v_{n-1} \rangle$  (i.e.,  $g$  is a cycle), then a valid code for  $g$  is  $(1, 2, L(1), L(1, 2), L(2)).(2, 3, \dots), (3, 4, \dots) \dots, (n-1, n, \dots).(n, 1, \dots)$ . We use a “dot” to denote the concatenation of each 5-tuple representing an edge of  $g$ .
- If  $g = g' \cup P$  and  $P$  is a valid extension of  $g'$  in  $g$ , then a valid code for  $g$  is the concatenation of a valid code for  $g'$  and the code of  $P$ .

It is not obvious from Definition 24 that every 2-connected plane graph  $g$  has at least one valid code. Indeed, since  $g$  is 2-connected, it is always possible to construct a valid code by first choosing an internal face of  $g$  and then iteratively adding valid extensions to it.

Table 2.2 shows four valid codes of graph  $g_1$  in Figure 2.15 (among seven valid codes). Figure 2.16 shows the corresponding node numbering on graph  $g_1$  (recall that there is a different numbering of nodes for each code). Codes  $\alpha$ ,  $\gamma$ ,  $\delta$  start with the 4-edge face and then a 2-edge extension is added to build the second face. Code  $\beta$  starts with the

Edge	$\alpha$	$\beta$	$\gamma$	$\delta$
1	(1,2,a,b,a)	(1,2,a,b,a)	(1,2,a,b,a)	(1,2,a,a,a)
2	(2,3,a,b,a)	(2,3,a,b,a)	(2,3,a,a,a)	(2,3,a,b,a)
3	(3,4,a,a,a)	(3,1,a,a,a)	(3,4,a,a,a)	(3,4,a,b,a)
4	(4,1,a,a,a)	(3,4,a,b,a)	(4,1,a,b,a)	(4,1,a,a,a)
5	(4,5,a,b,a)	(4,5,a,b,a)	(3,5,a,b,a)	(1,5,a,b,a)
6	(5,1,a,b,a)	(5,1,a,a,a)	(5,4,a,b,a)	(5,2,a,b,a)

Table 2.2: Four valid codes for graph  $g_1$ .

3-edge face and then a 3-edge extension is added. In each column, the line separates the edges of the first face from the edges of the valid extension. A valid code for this graph can start with any of the six edges. For the edge that belongs to the two internal faces, the code can start with any of the two faces, hence the seven possible codes.

### 3.2.3 Code Search Space and Canonical Codes

The set of valid codes is organized in a *code tree*. A code  $C'$  is a child of  $C$  in the code tree if there is a valid extension  $P$  of  $C$  such that  $C'$  is the concatenation of  $C$  with the codes of the edges of  $P$ . The root of the code tree is the empty code.

An example tree rooted at code  $\alpha$  (of Table 2.2) is represented in Figure 2.17. Notice that the codes at a given level of the tree represent graphs that have one more face than the codes of the level just above. In this code tree, each graph is represented by several codes (for instance, we have already seen that graph  $g_1$  has seven valid codes). In Figure 2.17 we also see that codes  $\alpha.A.D$  and  $\alpha.C.F$  represent the same graph.

Naturally, exploring several codes that represent the same graph is not efficient. We therefore define *canonical codes* such that each graph has exactly one such code: we start by defining an order on the valid codes. We assume that there exists an order on the labels. Then, we define an order on the edges by taking the lexicographic order derived from the natural order on node indices and the order on labels. It means that  $(i, j, L(i), L(i, j), L(j)) < (x, y, L(x), L(x, y), L(y))$  if  $i < x$  or ( $i = x$  and  $j < y$ ) or ( $i = x$  and  $j = y$  and  $L(i) < L(x)$ ), and so on. Afterwards, we extend this order on edges to a lexicographic order on the codes. We thus define the *canonical code* of a graph as the biggest code that can be constructed for this graph.

**Definition 25 (Canonical code for a plane graph)** *The canonical code of a plane graph is defined as the biggest valid code that can be constructed for this graph.*

In Figure 2.16, we assume that  $a < b < c$ . Therefore,  $\alpha > \beta$  since they have the same first two edges and the third edge of  $\beta$  is smaller than the third edge of  $\alpha$ . Because of the second edge,  $\beta > \gamma$  and, finally,  $\gamma > \delta$  since the first edge of  $\gamma$  is bigger than the first edge of  $\delta$ . Code  $\alpha$  is then the biggest code for graph  $g_1$ .

PLAGRAM and DYPLAGRAM do a depth-first exploration of a code tree. The next theorem states that, if they find a non-canonical code  $C$ , then it is not necessary to explore the descendants of  $C$ ; the whole subtree rooted at  $C$  can be safely pruned.

**Theorem 2.7** *In the code search tree, if a code is not canonical, then neither are its descendants.*

**Proof 1** *Let  $C$  be a non-canonical code of a graph  $G$  and  $C.E$  a code of a descendant  $G'$  of  $G$ . Let  $C_c$  be the canonical code of  $G$ . As such, code  $C_c$  can be extended to a new*

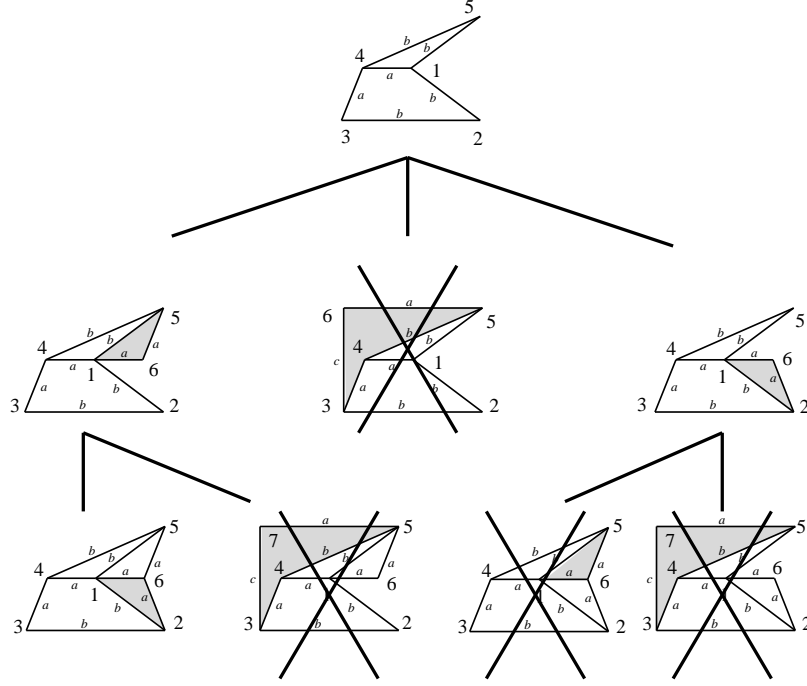


Figure 2.17: Part of the code tree starting from code  $\alpha$  of Table 2.2. For each pattern, the gray face corresponds to the last added extension. The extension codes are  $A, \dots, G$  (the complete code of the last line leftmost pattern is thus  $\alpha.A.D$ ). The crossed codes are pruned since they are not canonical.

code  $C_c.F$  for  $G'$ . Since  $C_c$  is the canonical code of  $G$ ,  $C_c > C$  and thus  $C_c.F > C.E$ . Therefore,  $C.E$  is not the biggest one and thus not canonical.

In Figure 2.17,  $\alpha.A.D$  and  $\alpha.C.F$  are two codes for the same graph. Since  $\alpha.A.D > \alpha.C.F$ , any extension of  $\alpha.A.D$  will be bigger than any extension of  $\alpha.C.F$ . Therefore, the latter code can be safely pruned.

### 3.2.4 Algorithms

**Pseudo-codes of PLAGRAM and DYPLAGRAM** The pseudo-code of PLAGRAM is given in Figures 2.18. The pseudo-code of DYPLAGRAM is very similar to the one of PLAGRAM except for the use of the subsequence frequency mentioned below.

The overall outline is very similar to that of GSPAN. The main differences are the graph code used to represent a plane graph and the way extensions are generated. As for GSPAN, our algorithms perform a depth-first recursive exploration of the code tree. Although the first level of the code tree contains codes representing graphs with one face, for efficiency reasons, PLAGRAM and DYPLAGRAM start their exploration with frequent edges. In both algorithms, the function *mine* explores the part of the code tree rooted at a code given by its parameter. It computes their extensions on every target graph in  $\mathcal{D}$  (lines 1-4) and makes a recursive call on the frequent and canonical ones (line 9).

The difference between PLAGRAM and DYPLAGRAM is on the exploited frequency measure in function *mine* (line 6). The subsequence frequency used by DYPLAGRAM needs a time threshold  $\tau$ , which defines the maximum gap allowed between two occurrences of a pattern (see Definition 19). Since  $\text{freq} \geq \text{freq}_{seq}$ , the number of extensions

<p><b>Algorithm: Plagram</b>(<math>\mathcal{D}, \sigma</math>)</p> <p>Input: graph database <math>\mathcal{D}</math> and frequency threshold <math>\sigma</math>.</p> <p>Output: plane subgraphs <math>P</math> in <math>\mathcal{D}</math> such that <math>\text{freq}(P) &gt; \sigma</math>.</p> <pre> 1 Find all frequent edge codes in <math>\mathcal{D}</math> 2 <b>for all</b> frequent edge code <math>E</math> <b>do</b> 3   mine(<math>E, \mathcal{D}, \sigma</math>) </pre>
<p><b>mine</b>(<math>P, \mathcal{D}, \sigma</math>)</p> <p>Input: the code of a pattern <math>P</math>, <math>\mathcal{D}</math>, and <math>\sigma</math>.</p> <pre> 1 <math>LE = \emptyset</math> //list of extensions of <math>P</math> 2 <b>for all</b> graph <math>G_i \in \mathcal{D}</math> <b>do</b> 3   <b>for all</b> occurrences <math>f</math> of <math>P</math> in <math>G_i</math> <b>do</b> 4     <math>LE = LE \cup \text{build\_extensions}(P, G_i, f)</math> 5   <b>for all</b> extensions <math>E</math> in <math>LE</math> <b>do</b> 6     <b>if</b> <math>\text{freq}(E) &gt; \sigma</math> <b>then</b> 7       <b>if</b> <math>P.E</math> is canonical <b>then</b> 8         output(<math>P.E</math>) 9         mine(<math>P.E, \mathcal{D}, \sigma</math>) </pre>

Figure 2.18: Algorithm PLAGRAM.

that are pruned (in line 6) is higher in DYPLAGRAM than in PLAGRAM.

Next, we present a complexity study of the main steps of function *mine*. We denote  $m$  the number of edges of a given pattern  $P$ , and  $m_i$  the number of edges of every target graph  $G_i$ .

**Pattern matching (line 3):** For each pattern  $P$ , function *mine* must find all occurrences of  $P$  in every target graph  $G_i$ . Each occurrence is found with a subgraph isomorphism test ([67]), which works as follows: first, it looks for an edge  $e$  of  $G_i$  that corresponds to the first edge of  $P$ . Once this match is performed, the complexity of matching the remaining edges of  $P$  is  $O(m)$ . So, the complexity of finding one occurrence is, in the worst case,  $O(m.m_i)$ .

The function *mine* uses an optimization that makes this subgraph isomorphism test linear: it stores, along with pattern  $P$ , the list of edges  $e$  that match the first edge of  $P$ , in every target graph  $G_i$ . This list is updated in line 4 when generating the extensions. Therefore, to find the occurrences of  $P$  in  $G_i$ , it is not necessary to consider every edge of  $G_i$ , but only those in this list. In this way, for each occurrence, the cost of a matching becomes  $O(m)$ . Since the number of occurrences of  $P$  in a target graph  $G_i$  cannot be higher than  $2m_i$  (the first edge of  $P$  may match each edge of  $G_i$  in two “directions”), the complexity of computing all occurrences of  $P$  in all target graphs  $G_i$  is  $O(m \sum m_i)$  (which is bounded later by  $O(\sum m_i^2)$  in Theorem 2.8). We show in the experimental subsection that this complexity improvement over GSPAN is visible in the measured matching times.

**Extension building (line 4):** For every occurrence  $f$  of  $P$  in a target graph  $G_i$ , function *mine* builds all possible extensions. This is done by finding a valid extension starting from every node of the outer face of  $f(P)$ . The complexity of this operation

is linear in the total size of  $P$  plus the size of the extensions. This is lower than  $2m_i$  since one edge of  $G_i$  is either in  $f(P)$  or in at most two of its extensions. Since there are at most  $2m_i$  occurrences of  $P$  in  $G_i$ , the complexity of building all extensions of all occurrences of  $P$  in all target graphs  $G_i$  is  $O(\sum m_i^2)$ .

Every time a new extension is added to the list  $LE$ , its frequency is updated. This enables the test in line 6. In the case of DYPLAGRAM, the last value of  $i$  such that the extension appears in  $G_i$  is also stored for the computation of  $\text{freq}_{\text{seq}}$ . The  $LE$  list is implemented in a way such that the addition of a new extension (together with its frequency counting) is done with a logarithmic complexity (as a function of the number of edges of the extension). Therefore, for a fixed pattern  $P$ , we bound this complexity by the total size of all its extensions in all  $G_i$ s, i.e., by  $O(\sum m_i^2)$ .

According to the conducted experiments, the extension building step of function *mine* was found to be the most expensive step.

**Canonical test (line 7):** This test is done by comparing code  $P.E$  with the canonical code of the graph represented by  $P.E$ . Since two plane graphs are isomorphic if their canonical codes are the same, the complexity of this test is at least as high as an isomorphism test. The complexity of graph isomorphism, in the general case, is unknown, but for plane graphs, polynomial algorithms exist (see, for instance, [67] for a quadratic algorithm). The simplest algorithm is to enumerate every possible code for a graph to test if one particular code is canonical (with an exponential complexity). Here is a sketch of our canonical test: the canonical code of a graph is constructed by first choosing a starting face and a starting edge in this face. Since a pattern  $P$  has  $m$  edges and considering that each edge belongs to at most two faces, there are at most  $2m$  such choices. Then, the code is extended with the biggest valid extension code. Each of these steps has a complexity of  $O(m)$  and must be repeated as many times as the number of faces in  $P$ , which is lower than  $m$ . Therefore, the complexity of finding the canonical code of a graph is, in the worst case,  $O(m^3)$ . Although not quadratic, experimental evaluations show that the canonical tests are not the bottleneck of our algorithms.

**Theorem 2.8 (Complexity)** *The total complexity of the function mine (excluding the complexity of recursive calls in line 9) is  $O(m^3 + \sum m_i^2)$ , where  $m$  is the size of the pattern  $P$  (in number of edges) and  $m_i$  is the size of the target graph  $G_i$  (in number of edges).*

A consequence of this theorem is that, contrary to general graph mining algorithms as GSPAN, PLAGRAM and DYPLAGRAM have a polynomial output delay, i.e., the time between the output of two frequent patterns is polynomial in the size of the input  $\sum m_i$  (since, of course,  $m < \sum m_i$ ).

**Theorem 2.9 (Correctness)** *PLAGRAM and DYPLAGRAM find and output exactly once all frequent 2-connected plane subgraphs in  $\mathcal{D}$  (using, respectively,  $\text{freq}$  and  $\text{freq}_{\text{seq}}$  as the frequency measure).*

**Proof 2** *Since there is a one-to-one correspondence between canonical codes and 2-connected plane graphs, we must show that the algorithms do not miss any frequent canonical code. The algorithms prune a branch of the tree either because the code is not frequent (line 6) or because it is not canonical (line 7). The frequency of the descendants of a code  $C$  cannot be higher than the frequency of  $C$ . Therefore, if a code is not frequent, its descendants are not either, and thus the pruning step in line 6 is safe. If the code is*



<p>Input: List of occurrences of <math>P</math>, frequency (<math>\text{freq}_{st}</math>) threshold <math>\sigma</math>, time threshold <math>\tau</math>, and spatial threshold <math>\epsilon</math>.</p> <p>Output: frequent spatio-temporal patterns based on <math>P</math>.</p> <pre> 1  The occurrence graph of <math>P</math> is empty. 2  <b>for all occurrences</b> <math>(x, y, k)</math> <b>do</b> 3    <b>for all</b> <math>0 &lt; j \leq k</math> and <math>k - j \leq \tau</math> <b>do</b> 4      <b>for all occurrences</b> <math>(x', y', j)</math> <b>do</b> 5        <b>if</b> <math>(x' - x)^2 + (y' - y)^2 &lt; \epsilon^2</math> <b>then</b> 6          add edge <math>((x, y, k), (x', y', j))</math> to the occurrence graph 7  Build the connected components of the occurrence graph    // each connected component is a spatio-temporal pattern 8  Output the frequent connected components.</pre>
---

Figure 2.19: Generation of spatio-temporal patterns.

not canonical, we know from theorem 2.7 that its descendants cannot be either. So, the pruning in line 7 is safe as well. In this way, the algorithms can never miss a frequent canonical code. Finally, every output code (line 8) is frequent and canonical and, since there is only one canonical code for each graph, a graph is output only once.

Actually, the algorithms output codes and not graphs. However, since a code is a list of edges, it is easy to reconstruct a graph from its code.

Once patterns have been extracted using either PLAGRAM or DYPLAGRAM, the occurrence graph can be constructed in a post-processing step to generate the spatio-temporal patterns. This is done by connecting occurrences of the same pattern with an edge if they respect the spatio-temporal constraints, and then computing the connected components of the occurrence graph.

**Post-Processing Generation of Spatio-Temporal Patterns** PLAGRAM and DYPLAGRAM respectively use  $\text{freq}$  and  $\text{freq}_{seq}$  instead of  $\text{freq}_{st}$  and do not build the occurrence graph during the mining phase. Nonetheless, to generate the spatio-temporal patterns from the frequent patterns returned by both algorithms, the occurrence graph needs to be built in a post processing phase. When those two variants output a frequent pattern  $P$  (line 8, in function *mine*), they also output a list of the occurrences of  $P$ . This list consists of triplets  $(x, y, k)$  where  $(x, y)$  are the coordinates of an occurrence, and  $k$  is the index of  $G_k \in \mathcal{D}$  where this occurrence appear. From this list, the algorithm of Figure 2.19 computes the spatio-temporal patterns based on  $P$  as follows:

first, it builds the occurrence graph of pattern  $P$  with respect to  $\epsilon$  and  $\tau$ , as defined in Definition 22 (lines 1-6). Given an occurrence  $(x, y, k)$ , the algorithm computes its distance with every other occurrence in the  $\tau$  previous graphs  $G_j$ . The number of these occurrences is at most  $O(\tau \cdot \max_i(m_i))$ , where  $\max_i(m_i)$  is the maximal size of the graphs in  $\mathcal{D}$ . Therefore, the complexity of building the occurrence graph of a pattern is  $O(\tau \cdot \max_i(m_i) \cdot \sum_i m_i)$  (since the number of occurrences of a pattern is at most  $2 \sum_i m_i$ ). The computation of the connected components and their frequency (line 7) is done by a traversal of the occurrence graph (linear complexity). Finally, the complexity of computing all frequent spatio-temporal patterns based on a pattern  $P$  is  $O(\tau \cdot \max_i(m_i) \cdot \sum_i m_i)$ .

**Pseudo-code of DyPlagram<sub>st</sub>** Given a frequency threshold  $\sigma$  (also called minimum support), a minimum threshold  $\sigma_{st}$  for  $\text{freq}_{st}$ , a spatial threshold  $\epsilon$  and a temporal threshold  $\tau$ , the proposed algorithm DYPLAGRAM<sub>ST</sub> computes all spatio-temporal patterns with  $\text{freq}_{st} \geq \sigma_{st}$  based on patterns with  $\text{freq} \geq \sigma$  (the thresholds  $\epsilon$  and  $\tau$  are used in the construction of the occurrence graph, see Def. 22).

With the new distance used for  $\text{freq}_{st}$ , the frequency constraint is now anti-monotonic (see definition 20), therefore we can use it in the DYPLAGRAM<sub>ST</sub> algorithm directly during the mining phase. However, this frequency is not defined on patterns but on spatio-temporal patterns. We must therefore also build the occurrence graph and the spatio-temporal patterns in the algorithm.

As its predecessors, DYPLAGRAM<sub>ST</sub> uses canonical codes to represent patterns and extensions. This allows us to efficiently enumerate only the so called valid extensions of a pattern. Informally, a valid extension of a pattern is an extension that leads to a pattern not already considered by the algorithm. This is a very efficient way to avoid considering several times the same pattern.

As can be seen in the pseudo code of the DYPLAGRAM<sub>ST</sub> algorithm in Figure 2.20, first all frequent one face patterns are built and then the recursive function `mine` is called for all of them.

Lines 1, 6, 7, 8, 9, 10, and 11 of the algorithm in Figure 2.20 were not in DYPLAGRAM. Thanks to Prop. 2.6, this algorithm is correct and outputs exactly the spatio-temporal patterns whose  $\text{freq}_{st}$  is above the user defined threshold  $\sigma$ .

### 3.3 Experiments

Our PLAGRAM algorithm is very similar to the well known GSPAN ([129]) algorithm. GSPAN also uses a Depth-First Search (DFS) traversal of its nodes and a corresponding *DFS code* to encode the graph. However the expansion strategy is different in GSPAN as it extends graph patterns edge by edge, and several extensions may be generated from one node. Besides, it also uses a general graph isomorphism test while mining which is more expensive than the plane graph isomorphism used in PLAGRAM. In a series of experiments shown in [20][37] we have compared i) how PLAGRAM and GSPAN scaled on our video data (see Chapter 4), ii) how efficient PLAGRAM is in finding the patterns we are interested in and iii) the impact of the spatio-temporal constraints on the efficiency of the mining phase. The experiments proved that PLAGRAM and its variants are more efficient at mining 2-connected plane graph databases than the general purpose algorithm which cannot be directly used to mine our video datasets with reasonable support thresholds. Our expansion strategy turned out to be the most critical reason for the efficiency of PLAGRAM compared to GSPAN. Besides, our algorithm benefits a lot from enforcing spatial and temporal constraints during the mining process. Indeed, by permitting to directly mine spatio-temporal constraints the algorithm generates less occurrences resulting in a lower processing time.

## 4 Conclusion

This chapter has presented an inductive database framework called *Mining views* that can be used to query patterns as well as data in a single database. To allow this to work in practice, exact algorithms that can take into account a wide range of constraints should be developed. We have presented two attempts in that direction: DL8 for mining optimal decision trees and PLAGRAM for mining plane subgraph patterns. Note

<p><b>Algorithm: DyPlagram<sub>st</sub></b>(<math>\mathcal{D}, \sigma, \sigma_{st}, \tau, \epsilon</math>)</p> <p>Input: graph database <math>\mathcal{D}</math>, frequency threshold <math>\sigma</math>, spatio-temporal frequency threshold <math>\sigma_{st}</math>, time threshold <math>\tau</math> and spatial threshold <math>\epsilon</math>.</p> <p>Output: spatio-temporal patterns <math>S</math> in <math>\mathcal{D}</math> such that <math>\text{freq}_{st}(S) &gt; \sigma_{st}</math> and <math>\text{freq}_{seq}(P) &gt; \sigma</math> with <math>P</math> the pattern on which <math>S</math> is based.</p> <pre> 1 Find all frequent face codes in <math>\mathcal{D}</math> 2 <b>for all</b> frequent face code <math>E</math> <b>do</b> 3   mine(<math>E, \mathcal{D}, \sigma, \sigma_{st}, \tau, \epsilon</math>) </pre>
<p><b>mine</b>(<math>P, \mathcal{D}, \sigma, \sigma_{st}, \tau, \epsilon</math>)</p> <pre> 1 occurrences_graph(<math>P</math>) = empty_graph 2 <math>LE = \emptyset</math> //list of extensions of <math>P</math> 3 <b>for all</b> graph <math>G_i \in \mathcal{D}</math> <b>do</b> 4   <b>for all</b> occurrences <math>f</math> of <math>P</math> in <math>G_i</math> <b>do</b> 5     <math>LE = LE \cup \text{build\_extensions}(P, G_i, f)</math> 6     Add this occurrence to occurrences_graph(<math>P</math>) 7   Computes the edges of occurrences_graph(<math>P</math>) (using <math>\epsilon</math> and <math>\tau</math>) 8   Computes all spatio-temporal patterns based on <math>P</math> 9   <b>for each</b> spatio-temporal pattern <math>S</math> based on <math>P</math> <b>do</b> 10    <b>if</b> <math>\text{freq}_{st}(S) \geq \sigma_{st}</math> <b>then</b> output(<math>S</math>) 11  <b>if</b> there is no frequent spatio-temporal pattern <b>then return</b> 12  <b>else</b> 13  <b>for all</b> extensions <math>E</math> in <math>LE</math> <b>do</b> 14    <b>if</b> <math>\text{freq}_{seq}(E) &gt; \sigma</math> <b>then</b> 15      <b>if</b> <math>P.E</math> is canonical <b>then</b> 16        mine(<math>P.E, \sigma, \sigma_{st}, \tau, \epsilon, \mathcal{D}</math>) 17  <b>return</b> </pre>

Figure 2.20: DYPLAGRAM<sub>ST</sub> algorithm

that the subspace clustering algorithm presented in [15] could also be integrated in such framework. While both algorithms presented in this chapter meet the specifications to be integrated into an IDB, in practice, they may lack efficiency to be used in an interactive data mining session.

In the case of DL8, one possible way to improve efficiency is to reuse previous results as much as possible. The system should be able to memorize the previously computed trees to check if the answer of the current query has not already been computed before triggering the data mining algorithm. For e.g, if the user is asking for all trees of size lower than 8 and later, for all trees of size lower than 6, the results computed from the first query should be reusable for the second query. Another type of problem occurs if the database has been modified between two queries. How to reuse some previously computed predicted models to compute more efficiently new predictive models from a modified database is still an open problem.

In the case of graph mining, the problem is simply too complex to be solved in real time in the general case (and thus integrated in an IDB). To gain efficiency, we already restricted ourself to the case of plane graphs which are rich enough representations for particular application domains such as computer vision. In the same line of work, we

are also exploring *geometric graphs* which particular structure could further help us answering queries in real time.

To tackle real-time problems we have also investigated the problem of mining stream data. In particular, we have focused on the problem of mining the top-k largest tiles [16][36, 27] in a data stream for moderate window sizes. Large tiles in a database are itemsets with the largest area which is defined as the itemset frequency in the database multiplied by its size. Mining these large tiles is an important pattern mining problem and we showed that it could be used both in computer vision and for emerging topic monitoring (e.g. for social media).

In the next chapter, we present a concrete computer vision application that can greatly benefit from the use of advanced data mining techniques.

## Chapter 3

# Data mining for BOW-based Image classification

Classification of images is of considerable interest in many image processing and computer vision applications. A common approach to represent the image content is to use histograms of color, texture and edge direction features [59, 119]. Although they are computationally efficient, such histograms only use global information and so, provide a crude representation of the image content. One trend in image classification is towards the use of *bag-of-visual-words* (BOW) features [65] that come from the *bag-of-words* representation of text documents [112]. The creation of these features requires four basic steps: (i) the keypoints detection (ii) the keypoints description, (iii) a codebook creation and (iv) the image representation steps. Keypoints refer to small regions of interest in the image. They can be sampled densely [86], randomly [122] or extracted with various detectors [100] commonly used in computer vision. Once extracted, the keypoints are characterized using a local descriptor which encodes a small region of the image in a  $D$ -dimensional vector. The most widely used keypoint descriptor is the 128-D SIFT descriptor [96]. Once the keypoints are described, the collection of descriptors of all images of a training set are clustered, often using the K-MEANS algorithm, to obtain a visual codebook. Each cluster representative (typically the centroid) is considered as a visual word in a visual dictionary and each image can be mapped into this new space of visual words leading to a *bag-of-visual-words* (or an histogram of visual words) representation. Some of my works have been dedicated to improving either this BOW creation [9, 8][29] or to build better representations from them [1, 10, 11], to improve image classification. This chapter presents the ideas developed in [9] and [11] which give both aspects. Experiments on the two presented methods are given at the end of this chapter.

## 1 Supervised Learning of Gaussian Mixture Models for Better BOW

Generally, unsupervised clustering algorithms, such as K-means, are employed to create the clusters from which one can deduced the visual dictionaries of the BOW. One of the common features of the unsupervised clustering methods is that they only optimize an objective function fitting to the data but ignoring their class information. Therefore, this reduces the discriminative power of the resulting visual dictionaries. For example, the K-means algorithm minimizes the within-cluster sum of squares of distances without

considering the class of the data (*i.e.* the label of the image the descriptor has been extracted from). Without any supervision, only one dictionary can thus be created for all the categories in the dataset, usually called *universal dictionary/vocabulary*. In [29], we presented an incremental gradient descent-based clustering algorithm which optimizes the visual word detection using some supervised information on the true class labels, leading to much better BOW-based classification results. However, this method assumes that each descriptor is generated from a single class and ignores the correlation in the  $D$ -dimensional space representing the descriptors. Moreover, this method, as well as other semi-supervised ones, do not try to optimize at the same time the *likelihood* of the training data and the *purity* of the clusters.

By integrating both criteria in the objective function to optimize, we claim that it is possible to jointly manage the two kinds of uncertainty the descriptors are usually subject to: the *cluster uncertainty* and the *class uncertainty*. The *cluster uncertainty* expresses the fact that it is something of an over-simplification to achieve a hard assignment (like K-means) during the construction of the clusters. For instance, a wheel can contribute to the construction of a visual word representing either a wheel of a bicycle or a wheel of a stroller, with different probabilities of membership. Taking into account this uncertainty during the creation of the visual dictionary can be realized using soft clustering such as Gaussian Mixture (GM) models, which have already been shown to outperform hard assignment-based approaches [121]. The *class uncertainty* can be illustrated by the following example: a brown patch descriptor may have been generated from both dog and cow classes. So given a brown patch descriptor, it would be short-sighted to label it by only one of these two classes. This type of uncertainty is usually ignored at the image descriptor level in most of the supervised dictionary creation algorithms. To overcome this limitation, we propose to exploit the probability for each descriptor to belong to each class. The estimation of these probabilities can be achieved by resorting to learned classifiers and approximating the Bayesian rule.

## 1.1 Notations and Definitions

Let  $X = \{x_k | k = 1 \dots n, x_k \in \mathbb{R}^D\}$  be the set of training examples, *i.e.* descriptors extracted from images and living in a  $D$ -dimensional space (*e.g.* SIFT descriptors usually live in a 128-dimensional space). Let  $C = \{c_j | j = 1 \dots R\}$  be the set of classes (*i.e.* the labels of the original images). Since labeling data can be very expensive, we assume that  $X$  may contain both labeled and non-labeled data. Let  $S = \{s_i | i = 1 \dots I\}$  be the set of clusters, where  $I > 1$ .

A Gaussian Mixture (GM) model is a generative model where it is assumed that data are *i.i.d* from an unknown probability density function [107]. In our approach, the distribution over the set of clusters is modeled using a GM model  $\Theta = \{\theta_i, i = 1 \dots I\}$  where  $\theta_i = \{\mu_i, \Sigma_i, w_i\}$  are the model parameters of the  $i^{th}$  Gaussian (corresponding to the cluster  $s_i$ ). Here,  $\mu_i$  is the mean,  $\Sigma_i$  is the covariance matrix and  $w_i$  is the weight of the  $i^{th}$  Gaussian. Given the GM model defined by its parameters  $\Theta$ , the probability of the descriptor  $x_k \in X$  is computed as follows:

$$p(x_k | \Theta) = \sum_{i=1}^I w_i \times N_{\mu_i, \Sigma_i}(x_k), \quad (3.1)$$

where  $N_{\mu,\Sigma}(x)$  is the multivariate Gaussian distribution, such that

$$N_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu)' \Sigma^{-1}(x - \mu)\right). \quad (3.2)$$

In a GM model, the posterior probability  $p(s_i|x_k, \Theta)$  is calculated as follows:

$$p(s_i|x_k, \Theta) = \frac{w_i \times p(x_k|s_i, \theta_i)}{\sum_t w_t \times p(x_k|s_t, \theta_t)} \quad (3.3)$$

subject to

$$\sum_i w_i = 1, \quad (3.4)$$

where  $p(x_k|s_i, \theta_i)$  is the probability for  $x_k$  to belong to the  $i^{th}$  Gaussian and that is exactly equal to  $N_{\mu_i, \Sigma_i}(x_k)$  given by Eq.3.2.

Usually, GM models are trained using the Expectation Maximization (EM) algorithm to find maximum likelihood parameters [68]. This is achieved by maximizing the log likelihood  $\mathcal{L}(X)$  of the training set  $X$  defined as follows:

$$\mathcal{L}(X) = \log\left(\prod_{i=1}^n p(x_i|\Theta)\right) = \sum_{i=1}^n \log(p(x_i|\Theta)). \quad (3.5)$$

Since  $p(s_i|x_k, \Theta)$  and  $p(x_k|\Theta)$  are unknown, they will be estimated by the GM and will be denoted by  $\hat{p}(s_i|x_k, \Theta)$  and  $\hat{p}(x_k|\Theta)$  respectively in the rest of the paper. Note that we will use the same notation for all the other unknown probabilities.

Unlike the K-means algorithm which builds clusters such that each descriptor belongs to the cluster with the nearest mean, a GM model has the ability to allow soft assignments by providing a probability for a given instance to belong to each cluster (thanks to Eq.3.3). It can be very useful to exploit these probabilities not only during the visual dictionary construction (as we will see in the next section) but also in the recognition step as shown in Fig. 3.1.

## 1.2 Supervised GM-based Dictionary Learning

### 1.2.1 Intuitive Idea

We claim that a relevant visual dictionary must be composed of visual words which are not only *specific* enough to be sufficiently discriminative, but also *general* enough to avoid overfitting phenomena. Note that to fulfill these two required conditions, one has to find a good compromise between the *cluster purity* and the *likelihood* of the data. If only the purity of the clusters is optimized using supervised information and without considering the likelihood of the data, the resulting clusters will be very discriminative but the generalization behavior of the BoW model will be likely subject to an overfitting phenomenon. On the other hand, when no class information is considered (e.g in a standard K-means algorithm), the resulting clusters will tend to be too general and each cluster (visual word) might represent (too) many classes, reducing the discriminative ability of the visual dictionary built from the clusters. Our objective is to optimize not only the likelihood of the data but also the cluster purity by resorting to a convex combination of both criteria optimized by a standard EM-based approach. The aim is to find a good trade-off allowing us to generate visual words that are discriminative

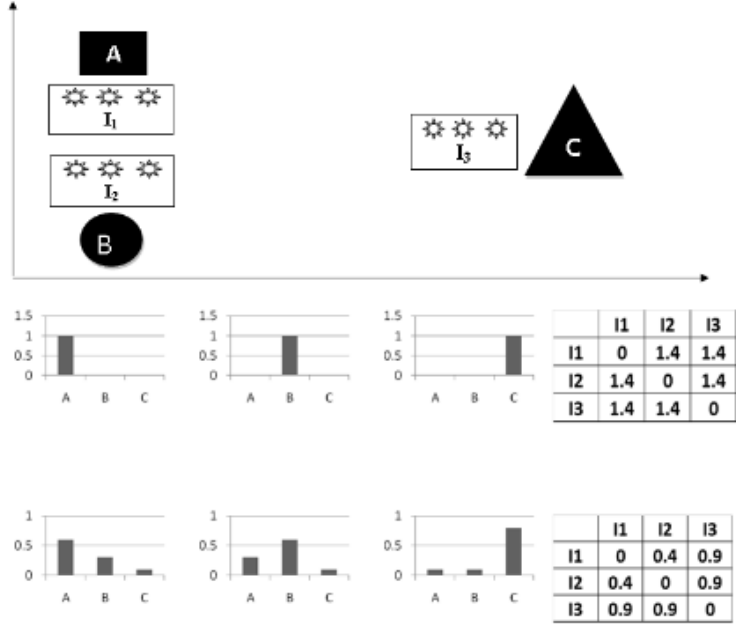


Figure 3.1: Soft assignment vs. hard assignment. Let us assume that the visual dictionary is made of three visual words  $A$ ,  $B$  and  $C$  and that three descriptors have been extracted from three images  $I_1$ ,  $I_2$  and  $I_3$ . With a hard assignment, the label  $A$  is assigned to the descriptors of  $I_1$ ,  $B$  to those of  $I_2$  and  $C$  to those of  $I_3$ . By representing each image by a (normalized) histogram, the (Euclidean) distances between  $I_1$ ,  $I_2$  and  $I_3$  are all the same while  $I_1$  and  $I_2$  are closer in the  $D$ -dimensional space. Applying a soft assignment (at the bottom) allows us to better reflect the realities of the situation.

enough to classify instances of various concepts and general enough to represent an object model. The motivation behind our method is graphically presented in Fig. 3.2.

### 1.2.2 Joint Optimization of the Likelihood and the Purity

In our method, we use a GM model where each Gaussian models a visual word. Contrary to standard GM-based approaches, our supervised GM algorithm not only takes advantage of the soft assignment allowed by a GM model, but also integrates in the objective function a term estimating the purity of the clusters.

Let  $F(s_i|\theta_i)$  be the purity of the cluster  $s_i$ , defined from the entropy of that cluster as follows:

$$F(s_i|\theta_i) = -\log\left(-\sum_j \hat{p}(c_j|s_i, \theta_i) \times \log(\hat{p}(c_j|s_i, \theta_i))\right) + \phi. \quad (3.6)$$

where  $\hat{p}(c_j|s_i, \theta_i)$  is the estimated probability of class  $c_j$  given cluster  $s_i$  which depends on the GM parameters  $\theta_i$  of cluster  $s_i$ <sup>1</sup>.  $\phi$  is a constant equal to  $\log(\log(R))$  if  $R > 2$ , otherwise  $\phi = 0$ . This constant makes sure that  $F(s_i)$  is a positive function. The higher the value of  $F(s_i)$ , the purer the cluster.

$\hat{p}(c_j|s_i)$  is estimated using its marginal distribution expansion w.r.t. all possible samples  $x \in X$ :

<sup>1</sup>To simplify the notations,  $\theta_i$  will be omitted when it is explicitly related in a formula to cluster  $s_i$ . This is the case e.g. for  $F(s_i|\theta_i)$  or  $\hat{p}(c_j|s_i, \theta_i)$ .



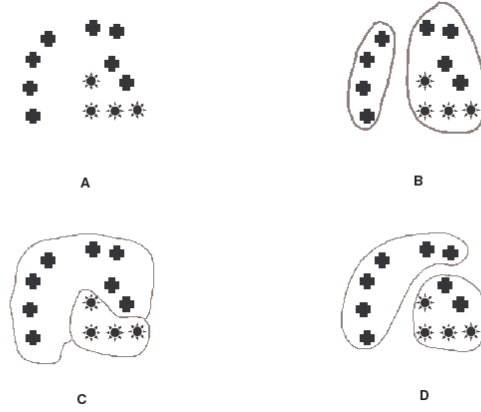


Figure 3.2: Rough illustration of the idea behind our approach. (A) Original data consisting of two classes. (B) Totally unsupervised clustering (e.g. K-Means). The purity of the cluster on the right is very low but the global likelihood (w.r.t. the centroids of the clusters) is optimized. (C) Totally supervised approach. The purity is optimal but the resulting clusters are too specific to allow some generalization ability. (D) Combination of both likelihood and purity criteria leading to a “good” trade-off.

$$\hat{p}(c_j|s_i) = \frac{\sum_k \hat{p}(x_k|c_j) \times \hat{p}(x_k|s_i)}{\sum_t \sum_k \hat{p}(x_k|c_t) \times \hat{p}(x_k|s_i)}. \quad (3.7)$$

The above equation is nothing more than a generalization of the proportion of examples which belong to a class  $c_j$  given a cluster  $s_i$ . But since we are using a GM, probabilities are used to estimate  $\hat{p}(c_j|s_i)$  rather than counting examples. To do that, we first need to estimate  $\hat{p}(x_k|s_i)$ . This can be done using the posterior probability  $\hat{p}(s_i|x_k, \Theta)$  (i.e. the so-called *cluster uncertainty*) given by Eq.3.3. Second, we have to estimate  $\hat{p}(x_k|c_j)$ . To achieve this task, we suggest to learn a classifier, use it to compute the posterior probability  $\hat{p}(c_j|x_k)$  (i.e. the so-called *class uncertainty*), and apply the Bayesian rule to get the estimate  $\hat{p}(x_k|c_j)$ . Note that this way to proceed allows the computation of the purity  $F(s_i)$  even in situations where the training set is composed of both *labeled* and *unlabeled* examples. Indeed, by taking advantage of the learned classifier to estimate  $\hat{p}(c_j|x_k)$ , it is possible to deal with a set  $X$  containing unlabeled instances  $x_k$  and therefore reduce the risk of errors and the expensive cost of manually and hardly labeling a large amount of data.

As mentioned before, the GM parameters are generally estimated by only optimizing the log likelihood of the data using the expectation maximization (EM) algorithm. Here, our objective is to find the parameters  $\Theta$  by optimizing not only the likelihood of Eq.3.5 but also the cluster purity of Eq.3.6. By this way, the two-fold uncertainty  $\hat{p}(c_j|x_k)$  and  $\hat{p}(s_i|x_k)$  are taken into consideration in the estimation. The objective function we aim at maximizing is defined as follows:

$$J(\Theta) = (1 - \alpha) \times \sum_{x_k \in X} \log(\hat{p}(x_k|\Theta)) + \alpha \times \sum_i^I \log(F(s_i)), \quad (3.8)$$

where  $\alpha$  ( $0 \leq \alpha \leq 1$ ) is a control parameter of the algorithm that determines the level

of supervision authorized in the GM. If  $\alpha = 0$ , the algorithm is totally unsupervised and so only optimizes the likelihood. In this case, it boils down to learning a standard GM model with the limitations already mentioned in the context of image classification. If  $\alpha = 1$ , the optimization process only aims at building pure clusters with the obvious risk to lead to overfitting phenomena. Therefore,  $\alpha$  plays an important role in our method and deserves a special attention in order to find a good compromise between these two extreme situations.

Our objective is to find the optimal model parameters  $\Theta^*$  that maximize the above objective function such that:

$$\Theta^* = \operatorname{argmax}_{\Theta} J(\Theta). \quad (3.9)$$

We can analytically find the solution of the objective function  $J(\Theta)$  by computing the derivatives with respect to each model parameter  $\mu_i, \Sigma_i, w_i$  for  $i = 1 \dots I$ . Since we are constrained by Eq.3.4, we use a Lagrange multiplier  $\lambda$  as follows:

$$\tilde{J}(\Theta) = J(\Theta) + \lambda(1 - \sum_i w_i). \quad (3.10)$$

Computing the partial derivatives of Eq.3.10 w.r.t.  $\mu_i, \Sigma_i, w_i$  respectively and equating them to zero allows us to find the optimal parameters. The general formula of the derivative of Eq.3.10 w.r.t.  $\mu_i$  or  $\Sigma_i$  (noted  $(\mu_i, \Sigma_i)$ ) is given by (see [9] for more details):

$$\begin{aligned} \frac{\partial \tilde{J}}{\partial (\mu_i, \Sigma_i)} &= \sum_k \{ (1 - \alpha) \hat{p}(s_i | x_k) + \\ &\quad \alpha B_i \sum_j a_i^j \times \hat{p}(x_k | c_j) \times \hat{p}(x_k | s_i) \} \times \\ &\quad \frac{\partial}{\partial (\mu_i, \Sigma_i)} \log(\hat{p}(x_k | s_i)) \end{aligned} \quad (3.11)$$

where  $B_i$ , the normalization parameter of the  $i^{\text{th}}$  Gaussian, is given by

$$\begin{aligned} B_i &= \frac{-1}{F(s_i) [\sum_j \hat{p}(c_j | s_i) \times \log(\hat{p}(c_j | s_i))]} \\ &\quad \times \frac{1}{\sum_t \sum_k \hat{p}(x_k | c_t) \times \hat{p}(x_k | s_i)}, \end{aligned} \quad (3.12)$$

and where

$$a_i^j = 1 + \log(\hat{p}(c_j | s_i)). \quad (3.13)$$

Using either  $\mu_i$  or  $\Sigma_i$  in Eq.11, and equating to zero we find the optimal parameters of each Gaussian such that:

$$\mu_i = \frac{\sum_k \{ (1 - \alpha) \hat{p}(s_i | x_k) + \alpha B_i \sum_j a_i^j \hat{p}(x_k | s_i) \hat{p}(x_k | c_j) \} x_k}{\sum_k \{ (1 - \alpha) \hat{p}(s_i | x_k) + \alpha B_i \sum_j a_i^j \hat{p}(x_k | s_i) \hat{p}(x_k | c_j) \}}, \quad (3.14)$$

$$\Sigma_i = \frac{\sum_k \{(1 - \alpha)\hat{p}(s_i|x_k) + \alpha B_i \sum_j \alpha_i^j \hat{p}(x_k|s_i)\hat{p}(x_k|c_j)\} A_i^k}{\sum_k \{(1 - \alpha)\hat{p}(s_i|x_k) + \alpha B_i \sum_j \alpha_i^j \hat{p}(x_k|s_i)\hat{p}(x_k|c_j)\}}, \quad (3.15)$$

where  $A_i^k = (\mu_i - x_k)(\mu_i - x_k)^t$ .

Computing the derivatives of Eq.3.10 with respect to parameter  $w_i$  (see [9] for more details), we get

$$\frac{\partial \tilde{J}(\Theta)}{\partial w_i} = \Sigma_k \frac{\hat{p}(x_k|s_i)}{\sum_t w_t \times \hat{p}(x_k|s_t)} - \lambda, \quad (3.16)$$

and equating to zero, we get

$$w_i = \frac{\sum_k \hat{p}(s_i|x_k)}{n}. \quad (3.17)$$

In the next section, we will use Equations 3.14,3.15 and 3.17 to update the parameters of the GM model using an EM-based iterative learning algorithm.

### 1.2.3 EM-based Learning Algorithm

After an initialization step, our EM-based algorithm iteratively performs (as usually) an expectation (E) step and a maximization (M) step. The E-step consists of estimating from the training set the expected value of the parameters, which are then used in the M-step (i) to maximize the expected value of our objective function  $J$  and (ii) to estimate the new model parameters.

To initialize our GM model, we have to predetermine the number of clusters  $I$  (*i.e.* the number of visual words). We also have to provide to the EM-algorithm a first series of estimates for the parameters  $\mu_i, \Sigma_i$  and  $w_i, i = 1 \dots I$ . To do this, we simply run K-means algorithm on the training set  $X$ . The mean  $\mu_i$  corresponds to the centroid of the  $i^{th}$  cluster  $s_i$  and  $\Sigma_i$  to the corresponding covariance matrix. Finally, the initial weight  $w_i$  is calculated by counting the number of training examples located in the cluster  $s_i$  and by normalizing it to satisfy Condition 3.4.

The pseudo-code of our algorithm (called *GEMP*) is presented in Algorithm 2. The convergence of *GEMP* is reached if the objective function  $J$  does not increase sufficiently between two iterations. This condition can be verified w.r.t. a given threshold. But note that since  $J$  is a weighted average computed from more than 30 examples, we can apply the central limit theorem stating that  $J$  asymptotically converges towards a normal distribution and check the convergence by resorting to a statistical test of average equivalence.

## 2 Pattern Mining in BOW

Mid-level or semi-local features learnt using class-level information are potentially more distinctive than the traditional low-level local features constructed in a purely bottom-up fashion such as the traditional SIFT-BOW [65]. In [11, 10] we proposed a new and effective scheme for extracting mid-level features for image classification, based on relevant pattern mining. In particular, we mine relevant patterns of *local* BOW. We refer to the new set of obtained patterns as *Frequent Local Histograms* or FLHs. During

---

**Algorithm 2** *GEMP* Algorithm.

---

- 1:  $X = \{x_1 \dots x_n\}$ , a number of clusters  $I$ , and a learned classifier providing  $\hat{p}(c_j|x_k)$   
 $\forall j, k$
  - 2: **Result** Final GM parameters
  - 3:  $t \leftarrow 0$
  - 4: **Initialization-Step**: use of K-means to initialize parameters  $\Theta_0$
  - 5: **while** No convergence **do**
  - 6:     **E-Step**
  - 7:     Estimate expected value for  $\hat{p}(x_k|s_i)$ ,  $\hat{p}(s_i|x_k)$ ,  $\hat{p}(x_k|c_j)$ ,  $B_i$  and  $a_i^j$ ,  $\forall i, j, k$
  - 8:     **M-Step**
  - 9:     Update parameters  $\Theta_{t+1}$  using Equations 3.14, 3.15 and 3.17
  - 10:     Evaluate objective function  $J(\Theta_{t+1})$
  - 11:      $t \leftarrow t + 1$
  - 12: **end while**
  - 13: **return** parameters  $\Theta_t$ ;
- 

this process, we pay special attention to keeping all the local histogram information and to selecting the most relevant reduced set of FLH patterns for classification. The careful choice of the visual primitives and an extension to exploit both local and global spatial information allow us to build powerful *bag-of-FLH*-based image representations. We show that these *bag-of-FLHs* are more discriminative than traditional bag-of-words and yield state-of-the-art results on various image classification benchmarks.

After introducing some notations, we explain how we mine frequent local histograms (FLHs) (section 2.1). We then show how we select the most relevant set of FLHs for image classification (section 2.2) and present a suitable kernel for relevant pattern-based image classification (section 2.3).

Each image  $I$  is described by a set of features  $\{f_i|i = 1 \dots n_I\}$  and a class label  $c$ ,  $c \in \{1 \dots C\}$ . We assume that all the descriptors have been clustered to obtain a set of so-called visual words. Then, each key point  $f_i$  is given a label  $w_i \in W$  known as the visual word index.  $|W|$  is the visual word dictionary size. In our approach, for each feature  $f_i$  we compute a *local histogram* (also called a *local bag-of-words* LBOW),  $\mathbf{x}_i \in \mathbb{N}^{|W|}$  using the  $K$  spatial nearest neighbours of  $f_i$  (based on the distance between image coordinates and also including  $f_i$  itself as a neighbour). In practice, we use all features within a local square neighbourhood of size  $n \times n$  around the feature. The set of all the local histograms  $\mathbf{x}_i$  created from all images is denoted by  $\Omega$ .

## 2.1 Frequent Local Histogram (FLH) Mining

**Items, Transactions and Frequencies:** In order to avoid loss of information during the transaction creation process without generating ghost patterns, we propose the following new definition of an *item*. An item is defined as a pair  $(w, s)$ ,  $w \in W$  and  $s \in \mathbb{N}$ , with  $s$  being the frequency of the visual word  $w$  in the local histogram. Note that  $0 < s \leq K$  and for a given image there is at most one item per histogram bin.

Next, we create the set of *transactions*  $X$  from the set of local histograms  $\Omega$ . For each  $\mathbf{x} \in \Omega$  there is one transaction  $x$  (i.e. a set of items). This transaction  $x$  contains all the items  $(w_j, s_j)$  such that the bin corresponding to  $w_j$  in  $\mathbf{x}$  has the nonzero value  $s_j$ . A *local histogram pattern* is an itemset  $t \subseteq \Gamma$ , where  $\Gamma$  represents the set of all possible items. For any local histogram pattern  $t$ , we define the set of transactions that

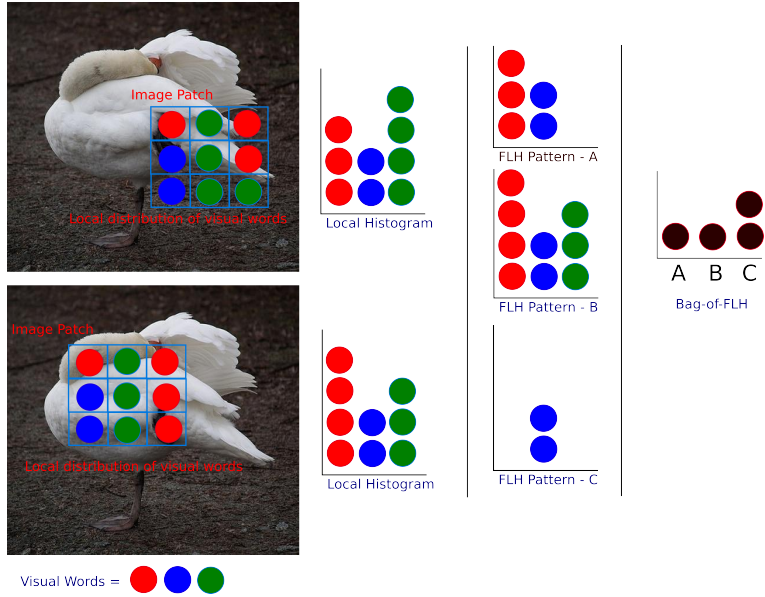


Figure 3.3: FLH mining and image representation process. First dense SIFT descriptors are extracted. Each descriptor is assigned to a visual word (hard assignment). For each dense descriptor, its  $K$  spatial nearest neighbors are selected (in practice, we use all local descriptors within a square  $n \times n$  neighbourhood). From these descriptors a local bag-of-words (LBOW) representation is created for each dense point. Then we mine for the most frequent local histograms from the entire dataset. These frequent local histograms are known as FLH patterns or just FLHs. Afterwards, using a post processing step, we select the most suitable set of FLHs for image classification. We encode the LBOWs in an image using these relevant FLH patterns (but note that some of the LBOWs won't be captured by any of the selected FLH patterns). Finally by counting how many times each FLH pattern is used to encode an image, we create a bag-of-FLHs representation.

include the pattern  $t$ ,  $X(t) = \{x \in X | t \subseteq x\}$ . The frequency of  $t$  is  $|X(t)|$ , also known as the support of the pattern  $t$  or  $supp(t)$ .

**Frequent Local Histogram:** For a given constant  $T$ , also known as the minimum support threshold, a local histogram pattern  $t$  is frequent if  $supp(t) \geq T$ . A pattern  $t$  is said to be closed if there exists no pattern  $t'$  such that  $t \subset t'$  and  $supp(t) = supp(t')$ .

The set of frequent closed patterns is a compact representation of the frequent patterns (i.e we can derive all the frequent patterns from the closed frequent ones). In this work we refer to a frequent and closed local histogram pattern as a *Frequent Local Histogram* or **FLH**.  $\Upsilon$  is the set of all FLHs.

**FLH Mining:** Given the set of transactions  $X$ , we can use any existing frequent mining algorithm to find the set of FLHs  $\Upsilon$ . What is specific to our method is that i) the input of our algorithm is a set of local histograms  $\Omega$ , and ii) a preprocessing step is performed building the set of transactions  $X$  from the local histograms  $\mathbf{x}_i$  as described above. Items  $(w_k, s_k)$  in a transaction  $x \in X$  can then be regarded as standard items in itemset mining.

The problems of finding these frequent itemsets are fundamental in data mining, and depending on the applications, fast implementations for solving the problems are needed.

In our work, we use the optimised *LCM* algorithm [118]. *LCM* uses a *prefix preserving closure extension* to completely enumerate closed itemsets. This allows counting the support of an itemset efficiently during the mining process. The *LCM* algorithm [118] supports database reduction, so that it can handle dense traditional datasets in short time and computes frequencies in linear time. It includes a strong pruning method to further reduce the computation time when the number of large frequent itemsets is small. It also generates closed itemsets with no duplication. For all these reasons, *LCM* is preferred over the well-known *APRIORI* algorithm [43]. Note though that the outcome does not depend on the choice of mining algorithm.

**Encoding a new image with FLHs:** Given a new image, we extract features by dense sampling and assign them to visual words. For each feature, we compute a LBOW around it, considering its  $K$  spatial nearest neighbours. Given this LBOW  $\mathbf{x}$ , we convert it into a transaction  $x$  and check for each FLH pattern  $t \in \Upsilon$  whether  $t \subseteq x$ . If  $t \subseteq x$  is true, then  $\mathbf{x}$  is an *instance* of the FLH pattern  $t$ . The frequency of a pattern  $t$  in a given image  $I_j$  (i.e., the number of instances of  $t$  in  $I_j$ ) is denoted as  $F(t|I_j)$ . We again refer to figure 3.3 for an example.

## 2.2 Finding the Best FLHs for Image Classification

We want to use the FLH set  $\Upsilon$  as a new set of mid-level features to represent an image. To this end, we first need to select the most useful FLH patterns from  $\Upsilon$  because i) the number of generated FLH patterns is huge (several millions) and ii) not all discovered FLH patterns are equally relevant for the image classification task. Usually, relevant pattern mining methods select patterns that are *discriminative* and *not redundant*. On top of that, we introduce a new selection criterion, *representativity*, that takes into account that, when using LBOW, a single image generates multiple transactions. As a result, some patterns may be frequent and considered discriminative but they may occur in very few images (e.g. due to repetitive structures). We believe that such features are not representative and therefore not the best choice for image classification. A good FLH pattern should be at the same time discriminative, representative and non-redundant. In this section we discuss how we select such patterns.

**Relevance criterion:** We use two criteria for pattern relevance: a *discriminativity score*  $D(t)$  [61] and a new *representativity score*  $O(t)$ .

The overall relevance of a pattern  $t$  is denoted by  $S(t)$  defined as:

$$S(t) = D(t) \times O(t) \quad (3.18)$$

We claim that if a pattern  $t$  has a high relevance score  $S(t)$ , it is likely to be discriminative and repeatable across images, hence suitable for classification.

**Discriminativity score:** To find discriminative patterns, we follow the entropy-based approach of [61], where a *discriminativity score*  $D(t)$  ( $0 \leq D(t) \leq 1$ ) for a pattern  $t$  is defined as:

$$D(t) = 1 + \frac{\sum_c p(c|t) \cdot \log p(c|t)}{\log C}, \quad (3.19)$$

with  $p(c|t)$  the probability of class  $c$  given the pattern  $t$ , computed as follows:

$$p(c|t) = \frac{\sum_{j=1}^N F(t|I_j) \cdot p(c|I_j)}{\sum_{j=1}^N F(t|I_j)}. \quad (3.20)$$

Here,  $I_j$  is the  $j^{\text{th}}$  image and  $N$  is the total number of images in the dataset.  $p(c|I) = 1$  if the class label of  $I_j$  is  $c$  and 0 otherwise. A high value of  $D(t)$  implies that the pattern  $t$  occurs only in very few classes. Note that in Eq. 3.19, the term  $\log C$  is used to make sure that  $0 \leq D(t) \leq 1$ .

**Representativity score:** The second factor for the relevance  $S(t)$  is the representativity  $O(t)$ . To compute it, we compare the distribution of the patterns over all the images with the optimal distribution with respect to a class  $c$ . A pattern having an optimal distribution is called an optimal pattern and denoted by  $t_c^*$  for class  $c$ . This optimal distribution is such that i) the pattern occurs only in images of class  $c$ , i.e.  $p(c|t_c^*) = 1$  (giving also a discriminativity score of 1), and ii) the pattern instances are equally distributed among all the images of class  $c$ , i.e.  $\forall I_j, I_k$  in class  $c$ ,  $p(I_j|t_c^*) = p(I_k|t_c^*) = (1/N_c)$  where  $N_c$  is the number of images of class  $c$ .

To find patterns with distributions close to the optimal one, we define the *representativity score* of a pattern  $t$  denoted by  $O(t)$ . It considers the divergence between the optimal distribution for class  $c$   $p(I|t_c^*)$  and the distribution for pattern  $t$   $p(I|t)$ , and then takes the best match over all classes:

$$O(t) = \max_c \{\exp\{-[D_{KL}(p(I|t_c^*)||p(I|t))]\}\} \quad (3.21)$$

where  $D_{KL}(\cdot||\cdot)$  is the Kullback-Leibler divergence between two distributions. The quantity  $p(I|t)$  is computed empirically from the frequencies  $F(t|I_j)$  of the pattern  $t$ :

$$p(I|t) = \frac{F(t|I)}{\sum_j F(t|I_j)} \quad (3.22)$$

**Redundant patterns:** We propose to remove redundant patterns in order to obtain a compact representative set of FLHs. We take a similar approach as in [128] to find affinity between patterns. Two patterns  $t$  and  $s \in \mathcal{T}$  are redundant if they follow similar document distributions, i.e if  $p(I|t) \approx p(I|s) \approx p(I|\{t, s\})$  where  $p(I|\{t, s\})$  gives the document distribution given both patterns  $\{t, s\}$ .

$$p(I|\{t, s\}) = \frac{F(t|I) + F(s|I)}{\sum_j F(t|I_j) + F(s|I_j)} \quad (3.23)$$

We define the redundancy  $R(s, t)$  between two patterns  $s, t$  as follows:

$$R(s, t) = \exp\{-[p(t) \cdot D_{KL}(p(I|t)||p(I|\{t, s\})) + p(s) \cdot D_{KL}(p(I|s)||p(I|\{t, s\}))]\} \quad (3.24)$$

where  $p(t)$  is the probability of pattern  $t$ :

$$p(t) = \frac{\sum_{I_j} F(t|I_j)}{\sum_{t_j \in \mathcal{T}} \sum_{I_j} F(t_j|I_j)} \quad (3.25)$$

Note that  $0 \leq R(s, t) \leq 1$  and  $R(s, t) = R(t, s)$ . For redundant patterns,  $D_{KL}(p(I|t)||p(I|t, s)) \approx D_{KL}(p(I|s)||p(I|t, s)) \approx 0$  which increases the value of  $R(s, t)$ .

**Finding the most suitable patterns for classification:** We are interested in finding the most suitable pattern subset  $\chi$  where  $\chi \subset \Upsilon$  for classification. To do this we define the *gain* of a pattern  $t$  denoted by  $G(t)$  s.t.  $t \notin \chi$  and  $t \in \Upsilon$  as follows:

$$G(t) = S(t) - \max_{s \in \chi} \{R(s, t) \cdot \min(S(t), S(s))\} \quad (3.26)$$

In Eq. 3.26, a pattern  $t$  has a higher gain  $G(t)$  if it has a higher relevance  $S(t)$  (*i.e. it is discriminative and representative*) and if the pattern  $t$  is non redundant with any pattern  $s$  in set  $\chi$  (*i.e.  $R(s, t)$  is small*). To find the best  $k$  patterns we use the following greedy process. First we add the most relevant pattern to the relevant pattern set  $\chi$ . Then we search for the pattern with the highest gain (non redundant but relevant) and add this pattern into the set  $\chi$  until  $k$  patterns are added (or until no more relevant patterns can be found).

### 2.3 Kernel Function for Effective Pattern Classification

After computing the  $k$  most relevant and non-redundant FLHs, we can represent each image using a new representation called *bag-of-FLHs* by counting the occurrences of such FLHs in the image. Let  $L$  be such a *bag-of-FLHs* for the image  $I_L$  and  $M$  be the *bag-of-FLHs* for the image  $I_M$ . We propose to use the kernel function

$$K(L, M) = \sum_i \min(\sqrt{L(i)}, \sqrt{M(i)}) \quad (3.27)$$

to find the similarities between the *bag-of-FLHs* of  $L$  and  $M$ . Here  $L(i)$  is the frequency of the  $i^{th}$  selected pattern in histogram  $L$ . This kernel provides good classification accuracies for our frequent pattern-based image representation. It is a standard histogram intersection kernel but with non-linear weighting. This reduces the importance of highly frequent patterns and is necessary since there is a large variability in pattern frequencies. Similar power-low normalization methods are used in improved Fisher Vector-based methods [109, 63].

### 2.4 GRID-FLH: Incorporating Global Spatial Information to FLH

Finally, we propose a variant of *bag-of-FLHs* that incorporates both global and local spatial information. We build on the spatial pyramid idea [90] and apply it in our FLH mining framework. First we create LBOW for all features in the image. Then we discover grid-specific relevant FLH patterns by employing the process described in Section 2.2. For each image, we concatenate these grid-specific *bag-of-FLH* representations to create a new representation called *GRID-FLH*. The *GRID-FLH* is a more structured local-global representation with more flexibility than traditional spatial pyramids [90]. Note that we mine FLHs specific to a grid cell from all the images and then create a *bag-of-FLHs* in a grid specific way. As a result each grid-cell uses a different set of FLH patterns.

## 3 Experiments

In order to assess the efficiency of our two dictionary creation methods, the GMM-based method called *GEMP* presented in Section 1 and the FLH-based method presented in Section 2, we carried out experiments on different very well known datasets of the computer vision domain.



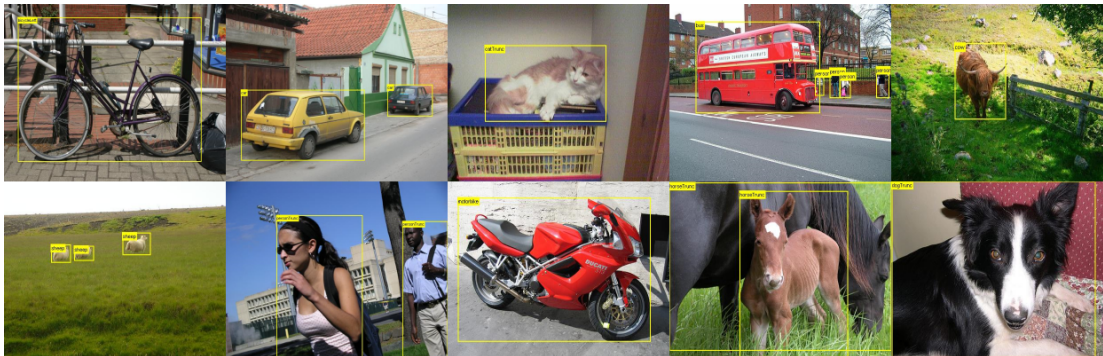


Figure 3.4: Sample images drawn from the 10 classes of PASCAL VOC-2006 dataset.

### 3.1 Image Datasets and Data Preparation

For the experiments which concern *GEMP* we used:

1. the well-known challenging *PASCAL-VOC-2006* dataset [72] which contains 5,304 images, 2,618 for training and 2,686 for testing including 9,507 annotated objects for 10 classes equally distributed between the training and the test sets.
2. the *Caltech-10* [89, 92] dataset which contains 400 images for training and 2644 for testing for 10 classes equally distributed between the training and the test sets.

For the experiments which concern the FLH-based method, we used:

3. *GRAZ-01* [106] which consists of two object classes (bike and person) and a complex yet representative background class. For each object class (bike or person) we randomly sample 100 negative images (50 from the background class and 50 from the other object class) and 100 positive images, as done in [90].
4. *Oxford-Flowers17* [105] which contains 17 flower categories where each category contains 80 images. We randomly select 60 images from each category for training and 20 images for testing as in [105].
5. *15-Scenes* [90] which contains 15 scene categories. This dataset is useful for evaluating scene classification. From each scene category, 100 randomly selected images are used for training and the rest are used for testing as in [90].
6. *Land-Use* [130] is a new dataset consisting of 2100 images of area imagery of various urban areas. There are 21 classes including various spatial structures and homogeneous textures. For this dataset, we also keep 50% of the images for training and 50% for testing.
7. *PASCAL-VOC2007* dataset [71] consists of 20 object classes and 9,963 images. This dataset is one of the most interesting image classification benchmarks. The data has been split into 50% for training/validation and 50% for testing.

We use classification accuracy to evaluate our results on the Oxford-Flower and Land-Use datasets and the mean classification accuracy computed over per-class-based classification accuracies for the 15-scenes dataset as done in the literature (and for comparison purpose). For the GRAZ-01 dataset we report ROC equal error rate. For

*PASCAL-VOC-2006*, *Caltech-10* and *Pascal-VOC-2007*, the dataset, we report the mean average precision or mAP.

For all datasets, we start from SIFT descriptors [95] that either describe patches found by a Harris-Laplace [101] key point detector using the implementation provided in <sup>2</sup> or densely sampled over the image with patches of size  $16 \times 16$  pixels and a grid spacing of 8 pixels (for the FLH-based method). As pointed out in [73], dimensionality reduction is an important step in GM-based BoW image representation. This makes sure that the most relevant directions of the input feature space are identified and that the remaining noisy directions discarded. Moreover, this allows us to substantially reduce the computational complexity of the algorithm. Therefore, for *GEMP*, we performed a principal component analysis to reduce the dimension  $D$  of the descriptors from 128 to 32.

To be able to achieve this classification task, an image  $P$  has to be represented in the form of a feature vector  $H$ . For the hard assignment-based methods (*e.g.* *K-means*), we used a standard normalized term frequency histogram approach. In this case, a component  $H_i$  of  $H$  corresponds to the proportion of times the  $i^{\text{th}}$  visual word (representing cluster  $s_i$ ) has been assigned to the descriptors extracted from the image  $P$ . More formally,

$$H_i = \frac{1}{|P|} \sum_{x_k \in P} \mathbb{1}_{[s_i=NN(x_k)]}, \quad (3.28)$$

where  $\mathbb{1}_{[s_i=NN(x_k)]}$  is an indicator function which takes the value of 1 if the center of  $s_i$  is the nearest neighbor  $NN(x_k)$  (using the Euclidean distance) of the descriptor  $x_k$  and 0 otherwise, and where  $|P|$  is the number of descriptors extracted from the image  $P$ .

For the soft-assignment methods based on a GM model, each component is obtained by summing (and normalizing) the conditional probabilities  $\hat{p}(s_i|x_k)$  for a descriptor  $x_k$  to belong to cluster  $s_i$ , that exactly corresponds to the observed frequency of descriptors in cluster  $s_i$ . More formally,

$$H_i = \frac{1}{|P|} \times \sum_{x_k \in P} \hat{p}(s_i|x_k). \quad (3.29)$$

We then use LIBSVM [57]<sup>3</sup> to train an SVM over image representation vectors. We use the square root intersection kernel in the SVM for FLH-based methods as presented in Section 2.3.

## 3.2 Experimental Results for the GMM Approach

### 3.2.1 Other Approaches

We compared our algorithm *GEMP* with three other approaches:

- The *K-means* clustering algorithm which will be used as a baseline.

<sup>2</sup><http://staff.science.uva.nl/~ksande/research/colordescriptors/>

<sup>3</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

- The *SDLM* model introduced in [92] which combines an unsupervised model (a GM) and a supervised model (a logistic regression model) in a probabilistic framework. To avoid having to implement this method and to allow us to simply report the results presented in [92], we used the same experimental setup.
- The incremental gradient descent-based clustering algorithm (*SA* for short) presented in [29] which optimizes the visual word detection by the use of the class label of training examples. Unlike *SDLM*, note that *SA* is a supervised hard-assignment method.

Note that the choice of these methods has been driven by our desire to compare *GEMP* with the state of the art dictionary methods, that is with either (i) unsupervised methods (*K-means*) or (ii) supervised methods with hard-assignment (*SA*) or (iii) supervised methods with soft-assignment (*SDLM*).

### 3.2.2 Pre-processing and Setting

For both datasets, we built the visual dictionary using 20,000 SIFT descriptors per class leading to a whole training set  $X$  of 200,000 examples. Among these data, we only used 40,000 labeled descriptors (*i.e.* 20% of  $X$ ) to create the dictionary. These labeled examples have been obtained from training images using a bounding box surrounding the considered object (see Figure 3.4) for the PASCAL-VOC-2006 dataset and using the boundaries of the object provided by the authors for the *Caltech-10* dataset. Note that the 80% remaining descriptors have been generated from the whole image without bounding boxes and their posterior probability  $\hat{p}(c_j|x_k)$  (required in our algorithm) has been estimated after having learned a Random Forest classifier [52] (with 20 trees and 8 random features used during the induction process).

To compare the different visual dictionaries learned by the four approaches and assess their respective discriminative power, we performed an image classification task. For both datasets (*PASCAL-VOC-2006* and *Caltech-10*), we learned 10 classifiers (one for each class against the others) using a Support Vector Machine algorithm and a chi-square kernel as used in [92].

### 3.2.3 Results

The best mean average precisions (Mean AP) are reported in Tables 3.1 and 3.2 for each binary classification problem. In each table, we indicate the number of visual words (parameter  $I$ ) required to reach the best performance. Note that we used a value  $\alpha = 0.5$  for *GEMP*. Several remarks can be made:

- First, for both datasets, our *GEMP* algorithm allows us to outperform all the other supervised and unsupervised methods. Using a Student paired-t test, we can show that the difference is significant in favor of our method. We obtain a precision of 0.8257 for *GEMP* versus 0.6425, 0.6715 and 0.7516 for *K-means*, *SDLM* and *SA* respectively for the PASCAL-VOC-2006 dataset and 0.9293 for *GEMP* versus 0.8373, 0.8928 for *K-means* and *SDLM* respectively for the *Caltech-10* dataset.
- Second, on the PASCAL-VOC-2006 dataset, for all the binary classification problems, *GEMP* is better than *SDLM*. Better still, for 9 binary problems out of 10, *GEMP* significantly outperforms *SDLM* using a Student paired-t test with a Type

	K-Means 2,000	SA 1,000	SDLM 400	GEMP 400
Object class				
Bicycle	0.7295	0.7081	0.8198	0.8267
Bus	0.4994	0.6457	0.8538	<b>0.8959</b>
Car	0.6822	0.6538	0.8122	<b>0.9038</b>
Cat	0.7131	0.7119	0.7537	<b>0.8639</b>
Cow	0.6358	0.6469	0.7581	<b>0.8582</b>
Dog	0.5917	0.6635	0.7234	<b>0.7617</b>
Horse	0.6431	0.6966	0.6322	<b>0.7604</b>
Motorbike	0.6890	0.6216	0.7946	<b>0.8332</b>
Person	0.5773	0.6686	0.6307	<b>0.6818</b>
Sheep	0.6905	0.6982	0.7376	<b>0.8714</b>
Mean AP	0.6425	0.6715	0.7516	<b>0.8257</b>

Table 3.1: Mean average precision evaluated on PASCAL VOC-2006 dataset. An average precision in bold font means that GEMP significantly outperforms all the other methods using a Student paired- $t$  test with a Type I error of 5%.

I error of 5% (the difference is not significant only for the class *Bicycle*). For the *Caltech-10* dataset, *GEMP* is better than both other methods on 9 out of 10 dataset and statistically significantly better on 5 out of 10 datasets. This constitutes an experimental evidence of the efficiency of *GEMP* since *SDLM* has already been proven to be very competitive in comparison with state of the art approaches [92]. The main reason of this improvement comes from the fact that we do not assume in our approach that each image descriptor has been generated from a single object category. By taking into consideration the two-fold uncertainty in our GM model, *GEMP* is able to improve the discriminative power of the created visual words.

- Finally, we can note that like *SDLM*, *GEMP* is sparse in terms of visual words required to reach the optimal performance. Indeed, for the PASCAL-VOC-2006 dataset, 400 visual words are sufficient for *GEMP* and *SDLM* while 1,000 clusters are necessary for *SA*, and 2,000 for *K-means*. The same remark can be made for the *Caltech-10* dataset for which *SDLM* and *GEMP* need a small number of visual words ( $I = 200$ ) to outperform the other methods. Through this classification in terms of required visual words to reach the optimal behavior, we can confirm that (i) using supervised information is better ( $\gg$ ) than resorting to a standard K-Means clustering (*i.e.*  $SA \gg K\text{-means}$ ), (ii) allowing a soft-assignment is better than hardly assigning a descriptor to the nearest centroid (*i.e.*  $SDLM \gg SA$ ) and finally (iii) taking into consideration in a GM model the two-fold uncertainty provides better results (*i.e.*  $GEMP \gg SDLM$ ).

To show the behavior of our method on various sizes of visual dictionaries, we reported on Figure 3.5 the results obtained with *GEMP* according to an increasing number of visual words from 50 to 1,000. We can see that whatever the size, *GEMP* is very competitive and thus, an interesting sparse method. A comparison with the behavior of *K-Means* (see Figure 3.6) confirms that the use of supervised information for building dictionaries dramatically improves the quality of the resulting visual words. *K-Means*

	K-Means	SDLM	GEMP
	200	200	200
Object class			
airplanes	0.8253	0.8803	<b>0.9253</b>
bonsai	0.8020	0.8649	0.8721
car	0.7981	0.8000	<b>0.8981</b>
chandelier	0.8286	0.9310	0.9486
faces	0.8988	0.9772	<b>0.9989</b>
hawksbill	0.7816	0.7375	<b>0.8217</b>
ketch	0.7948	0.9153	0.9248
leopards	0.8956	0.9157	<b>0.9957</b>
motorbikes	0.8781	0.9314	0.9381
watch	0.8697	0.9750	0.9697
mAP	0.8373	0.8928	<b>0.9293</b>

Table 3.2: Mean average precision evaluated on Caltech-10 dataset.

requires much more visual words (1,000 for PASCAL-VOC-2006 - in fact it needs 2,000 clusters to reach its optimum- and 400 for Caltech-10) to reach a rather poor average precision (0.6425). We also made experiments to evaluate the purity of the clusters according to number of visual words (using  $\alpha = 0.5$ ). We noted that the average purity increases showing that clusters generally tend to be specialized for a given class. Interestingly, we can also note that the standard deviation of the purity grows as we increase the number of words. This can be explained by the nature of our objective function which aims at jointly optimizing the likelihood and the entropy. Therefore, this leads to some clusters having a high purity and some others having moderate purity but high likelihood. In order to give an idea about which kind of words contributes the most to a good performance in image categorization, we estimate the mean average precision with (i) only words of high purity, (ii) only words of low purity and (iii) “intermediate” words representing a compromise between high purity and high likelihood. To achieve this task, we split the optimal set of visual words obtained for PASCAL VOC-2006 and Caltech-10 (i.e. 400 and 200 words respectively) into three balanced subsets: the most pure words, the less pure words, and the remaining intermediate ones. The results showed that using only words of high purity or of low purity is not sufficient to obtain a good precision. Though a good behavior is obtained with the intermediate words, we can see that one needs the three categories to reach the best performance (with the same number of words) that provides an experimental evidence of the interest of our algorithm.

The effects of the parameter  $\alpha$  in *GEMP* are explained in details in [9]. In particular, we show that:

- As expected, for small values ( $\alpha < 0.15$ ), the level of supervision is not sufficient to take advantage of *GEMP*. We obtain more or less the same results as *K-Means*. On the other hand, for large values ( $\alpha > 0.85$ ), the clusters become purer and purer leading to overfitting phenomena. Therefore, even if in both situations the obtained average precisions are almost the same (smaller than 0.65), this is definitely not due to the same reasons. In the first case, the resulting clusters are too general to be discriminative, while in the second case, the visual words are too specific to allow us to generalize.

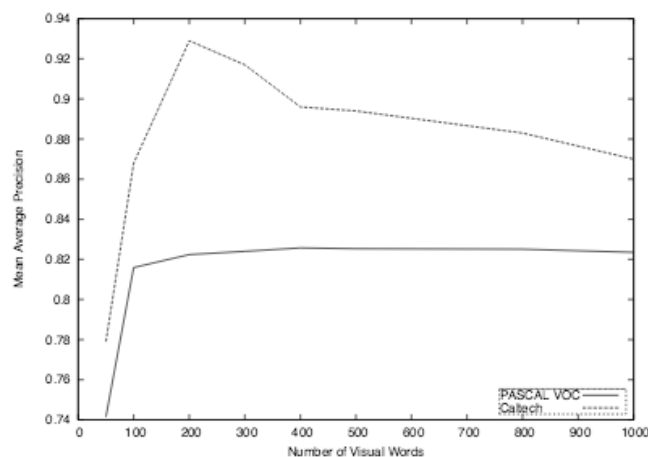


Figure 3.5: Mean average precision of GEMP (using  $\alpha = 0.5$ ) evaluated on PASCAL VOC-2006 and Caltech-10 on different dictionary sizes (from 50 to 1,000 visual words).

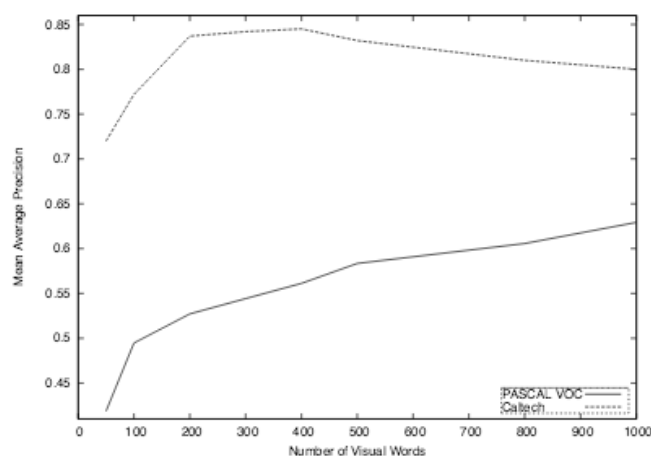


Figure 3.6: Mean average precision of K-Means evaluated on PASCAL VOC-2006 and Caltech-10 on different dictionary sizes (from 50 to 1,000 visual words).

- The best results are obtained with middle values ( $\alpha \approx 0.5$ ). Even if we are aware that the optimal  $\alpha$  obviously depends on the application we deal with, we can note that a good balance between the two criteria (*i.e.* likelihood and purity) of the objective function seems to be a relevant way to obtain good results.

### 3.3 Experimental Results for the FLH-based Approach

Experimental results which assess the interest of frequent itemset mining with different parameter settings (in particular the binarization method, the size of the original descriptor, the dictionary size, the neighborhood of the local BOW) compared to traditional BOW-based image classification settings can be found in [11]. Because of the size of the *Pascal-VOC-2007* and Land-Use datasets, we did not perform any baseline comparisons nor parameter optimizations for them, we simply report the results obtained with the parameters optimized for the other datasets.

### 3.3.1 Comparison with Non-mining Methods

In this section we compare FLH with non-mining methods that exploit local structural statistics. Specifically we compare our method with the spatial pyramid co-occurrence method (SPCK) of Yang *et al.* in [130] and the unbounded-order spatial features method of Chen and Zhang [131]. We also compare our method with the mid-level feature construction method of Boureau *et al.* in [51] and the PDK method [93] which uses proximity distribution kernels based on geometric context for category recognition. Results are reported in Table 3.3. GRID-FLH outperforms all other non-mining methods for both GRAZ-01 and 15-Scenes datasets. Note that for these methods, no results were reported for the Oxford-flower dataset.

The mid-level features method of [51] using macro-features seems to work quite well on 15-Scenes. This method also uses dense SIFT features but a visual dictionary of 2048. In this method the sparsity and supervision is enforced during the feature construction. The Bag-of-FLH representation, on the other hand, is quite sparse after the relevant pattern mining step. For example, in the case of Oxford-Flower dataset, there were 17.6% non-zero bins before the relevant pattern mining step and 5.13% non-zeros bins after. One of the key differences between the macro-features and FLH is that macro-features capture very small neighborhood of  $2 \times 2$  while FLHs capture comparatively larger neighborhoods of  $5 \times 5$ . Secondly for macro-features larger discriminative and supervised dictionaries seem to work well. For an unsupervised smaller dictionary of size 1048 macro-features reported only 83.6%.

Neither the spatial pyramid co-occurrence method of Yang *et al.* in [130] nor the unbounded-order spatial features method of Chen and Zhang [131] work as good as our GRID-FLH method. This could be due to the fact that none of these methods capture database wide spatial statistics.

Table 3.3: Comparison with non-mining methods

Dataset	GRAZ-Bike	GRAZ-Person	15-Scenes
GRID-FLH	<b>95.8</b>	<b>91.4</b>	<b>86.2</b>
FLH+BOW	95.0	90.1	83.0
FLH	94.0	89.2	70.4
SPCK [130]	91.0	87.2	82.5
PDK [93]	95.0	88.0	-
Higher Order Features [131]	94.0	84.0	-
Mid-Level Features [51]	-	-	85.6

### 3.3.2 Comparison with State-of-the-art Methods

In this section we compare *FLH* using the parameters optimized as above with, to the best of our knowledge, the best results reported in the literature.

**GRAZ-01:** The results reported in Table 3.4 show that on average all *FLH*-based methods outperform the state-of-the-art. The *GRID – FLH* representation, combining local and global spatial information, yields the best results. For the “Bike” class, the higher order features [131] seem the best. But on average FLH outperforms the higher order features [131] and the co-occurrence spatial features [130].

Table 3.4: Equal Error Rate (over 20 runs) for categorization on GRAZ-01 dataset

Method	Person	Bike	Average
SPCK+ [130]	87.2	91.0	89.1
NBNN [49]	87.0	90.0	88.5
Higher Order Features [131]	84.0	<b>94.0</b>	89.0
FLH	94.0	89.2	91.6
FLH + BOW	95.0	90.1	92.6
GRID-FLH	<b>95.8</b>	91.4	<b>93.8</b>

**Oxford-Flower:** The results are reported in Table 3.5. Note that only using SIFT features we get a classification accuracy of **92.9%**, reaching the state-of-the-art. GRID-FLH only gives an insignificant improvement of 0.4% compared to FLH. Note that we use only SIFT features for classification. Most of the other works such as [105, 114, 127], [8] use multiple features such as Hue and ColorName descriptors [120]. To the best of our knowledge the best results on Oxford-Flower17 using a single feature is reported by Rematas et al. [111], 85.3%. We should mention that when we combine SIFT with color information (using the ColorName descriptor [120]) we obtain a **classification accuracy of 94.0%** outperforming the state-of-the-art.

Table 3.5: Classification accuracy (over 20 runs) on the Flower dataset

Method	Accuracy
Nilsback [105]	88.3
CA [114]	89.0
$L_1 - BRD$ [127]	89.0
LRFF [8]	<b>93.0</b>
Pooled NBNN Kernel [111]	85.3
FLH	92.5
FLH + BOW	92.7
GRID-FLH	92.9

**15-Scenes:** Results are shown in Table 3.6. This dataset is strongly aligned. *FLH* does not exploit this and therefore by itself cannot obtain state-of-the-art results. However, the *GRID-FLH* method described in section 2.4 does take the global spatial information into account and achieves close to state-of-the-art results (86.2%). This is only outperformed by [132] who report 87.8% using CENTRIST and SIFT features along with LLC coding. In our defense, our method uses only SIFT features. As far as we know the previous best classification accuracy using SIFT features was reported by Tuytelaars *et al.* in [117] combining a NBNN kernel and a SPM method.

**Land-Use:** Yang and Newsam proposed a spatial pyramid co-occurrence method called SPCK [130] to classify Land-Use images. Most of these images are texture dominant. They use two types of spatial predicates: the proximity predicate and the orientation predicate, to define the SPCK method. We obtain a best result of **79.2%** for this dataset, again outperforming best results reported in [130]. The results for Land-Use dataset are shown in Table 3.7.



Table 3.6: Results on 15-Scenes dataset

Method	Accuracy
SPM	80.9
<i>SPCK</i> ++ [130]	82.5
NBNN Kernel+SPM [117]	85.0
(AND/OR) [132]	<b>87.8</b>
FLH	70.4
FLH+BOW	83.0
GRID-FLH	86.2

Table 3.7: Results on recent Land-Use dataset





















Method	Accuracy
BOW	71.9
SPM	74.0
<i>SPCK</i> <sub>SP1</sub> [130]	72.6
<i>SPCK</i> <sub>SP3+</sub> [130]	76.1
FLH	76.8
FLH+BOW	77.2
GRID-FLH	<b>79.2</b>

**Pascal-VOC2007:** Results are reported in Table 3.8. For this dataset Fisher Vector [109, 60] is the best performing method so far and the authors report a mAP of 61.7. The FLH-based method alone gives a mAP of 60.4. In combination with BOW of SIFT-128 and 5000 visual word vocabulary (with a weighted average kernel with weights learned using train/validation set), we obtain a state-of-the-art mAP of 62.8. Note that the score for each individual class often varies a lot between the FLH+BOW and the Fisher Vector [109] method. Our method does especially well on what are known to be 'hard' classes such as bottle (+34% improvement), dining table (+11%), potted plant (+16%), or tv monitor (+23%). This suggests that both methods are complementary. To evaluate this claim we also performed another experiment in which we average the output score of Fisher vector method [109, 60] with the output scores of (FLH+BOW) method. This yields a mean average precision of **72.2**. This approach clearly outperforms the state-of-the art by a significant margin. Not only this confirms the complementary nature of FLH and Fisher vectors but the improvement is consistent over every PASCAL-VOC class.

## 4 Conclusion

We have presented two very successful methods to improve BOW-based image classification. The first one, called *GEMP* is a soft-clustering method which integrates the use of background class information to improve the BOW representation. The second one, called *FLH*, is a mid-level representation built from the BOW to create more discriminant features for image classification. Note that both methods could be combined to further improve the classification results since our experiments with the first method have proved that the *K-means* algorithm, used in Section 2 is not the ideal clustering algorithm when trying to quantize descriptions of images. Our experiments have shown

Table 3.8: Results on PASCAL-VOC 2007 (Mean average precision)

Class											
Fisher Vectors(FV)	78.8	67.4	51.9	70.9	30.8	72.2	79.9	61.4	56.0	49.6	
FLH	67.9	70.6	41.0	54.6	64.9	60.9	85.8	56.6	59.6	40.0	
FLH+BOW	69.2	73.0	42.7	56.3	64.9	60.9	86.6	58.9	63.3	41.8	
FLH+FV	78.6	76.3	55.7	75.0	74.9	75.6	87.4	66.2	65.7	50.6	
FLH+BOW+FV	80.0	78.0	55.9	76.2	75.5	75.6	88.1	67.0	67.3	51.8	
Class											m.AP
Fisher Vectors(FV)	58.4	44.8	78.8	70.8	85.0	31.7	51.0	56.4	80.2	57.5	61.7
FLH	64.7	47.3	56.6	65.7	80.7	46.3	41.8	54.6	71.0	77.6	60.4
FLH+BOW	74.3	48.4	61.8	68.4	81.2	48.5	41.8	60.4	72.1	80.8	62.8
FLH+FV	75.7	52.4	78.7	77.0	88.8	53.9	51.7	68.7	83.6	82.0	70.9
FLH+BOW+FV	80.9	51.9	78.9	77.3	89.8	58.5	51.8	72.0	83.9	84.5	<b>72.2</b>

that our results highly depends on some parameters. For example *GEMP* is dependant on the parameters  $\alpha$  which gives the trade-off between the cluster and the class uncertainties in the optimization process or simply on the parameter that set the number of Gaussians in the mixture. For *FLH*, without mentioning the countless parameters associated to the BOW creation, the user also needs to decide the number of (good) local histograms to keep for his image description and could want to use other quality criteria than the ones described in Section 2 of this chapter. To achieve the best results, the user should be able to run multiple experiments and use the parameters as constraints to improve the efficiency of the algorithms.

In both cases, we tried experiments to combine our descriptors with other ones (for example adding a color-descriptor as a low level representation or at the BOW level) and we showed that this can further increase the end classification results. This also confirms that the image classification process needs to be an iterative and interactive process where a user should be able to run experiments, store intermediate results, combine them, run the algorithms with different parameters, etc. To integrate these requirements, one would need all those algorithms to be integrated in an inductive database such as the one presented in Chapter 2.

## Chapter 4

# Data mining for Object Tracking in Videos

In Chapter 2, we have presented a plane graph mining algorithm called `PLAGRAM` and its extension to be used on an ordered sequence of graphs called `DYPLAGRAM`. We also have shown how spatio-temporal constraints could be integrated to the previous algorithm to obtain the `DYPLAGRAM-ST` algorithm. In this chapter, we present an application of this algorithm to the analysis of videos and, in particular, the task of tracking objects in videos. The work presented in this chapter has been mainly published in [28],[20]. Note that the first problem when applying our algorithm to videos is to transform the video into a sequence of graphs. Different solutions have been explored in [20] but also in [113, 66]. The first section of this chapter explains our chosen representation.

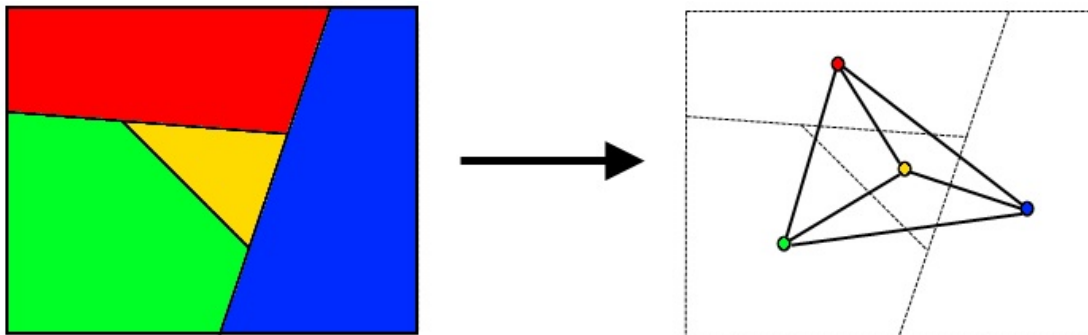


Figure 4.1: Example of segmented images with 4 regions (left) and its corresponding RAG (right)

## 1 Images and Videos as Graphs

There exist various methods (described in details in [69]) to represent images in ways that capture more semantic information than a simple matrix of pixels. A video can be seen as a sequence of such images (frames) and does not necessarily need a different processing.

## 1.1 RAG and Triangulation

To create our dynamic plane graph, we decided to first pre-process each frame of the video using a segmentation algorithm. Segmentation is the process of partitioning an image into multiple segments, i.e., sets of neighbor pixels that share a common property (e.g., their color). From the segmentation and for each frame we create a region adjacency graph (RAG) ([58]). In a RAG, each node represents a region and an edge connects two nodes if the corresponding two regions are adjacent in the image (see Figure 4.1). We associate to each node the coordinates  $(x, y)$  of the barycenter of its representing region. One special node is added to represent an unbounded region encompassing all the image. Informations on the regions (e.g., size, average color etc...) and on their borders (e.g, length for example) can be added to nodes and edges as labels. In our experiments we mostly used two types of node labels based either on a discretization of the size of the regions or a discretization of their average color. The inclusion relationship can also be represented by a label on the edges.

As our RAGs greatly depend on the segmentation, we tried two types of segmentation. The first segmentation (static) is done independently on each frame using the algorithm<sup>1</sup> presented in [74]. This algorithm has three parameters for which we use the default values. The second segmentation is the (dynamic) video segmentation algorithm<sup>2</sup> presented in [78]. This algorithm outputs regions that are identified through time, i.e, it provides a correspondence between regions in different frames. Figure 4.2 shows examples of RAGs representing a frame of our videos.

For the sake of comparison, we give some experiments using another plane graph representation. This representation, called *triangulation*, consists in triangulating the nodes (barycenter of the segmented regions) using a Delaunay triangulation [91].

## 1.2 Video Datasets

There exist multiple video benchmarks for object tracking in the literature (see [69]). However, the existing benchmarks are not entirely satisfactory because most of them focus on video surveillance setups or are composed of videos with too few frames for the mining step to extract meaningful patterns. Note that in these cases, our algorithms could also be used but may perform worse than the optimized dedicated ones. To assess the qualities of our algorithm, we thus introduce our own datasets. We used 4 videos for these experiments. They can be downloaded from <sup>3</sup>. The two first ones are synthetic videos. They allow us to avoid the possible segmentation problems by using the true colored regions. The two last ones are real videos.

**Synthetic videos** The original video has 721 frames in total. We made two versions out of it. In the first version, called *Anim1*, three identical objects (X-Wings) are moving in the video such that they may overlap or even get partially out of the video frames (this helped us to evaluate how well spatio-temporal patterns can be used to represent the trajectory of the X-Wings individually, as reported in subsection 2.2.2 of this chapter.

We first transformed this video into a *Triangulated* dataset. The final graphs had, on average, 197.33 nodes with an average degree of 2.93. The labels of the nodes were generated based on the size of the regions (in number of pixels). The size of the regions

<sup>1</sup>Efficient graph base segmentation source code available here: <http://cs.brown.edu/~pff/segment/>

<sup>2</sup>Video segmentation web service at this address: <http://videosegmentation.com/>

<sup>3</sup><http://perso.univ-st-etienne.fr/frel9915/Diot/interface.html>

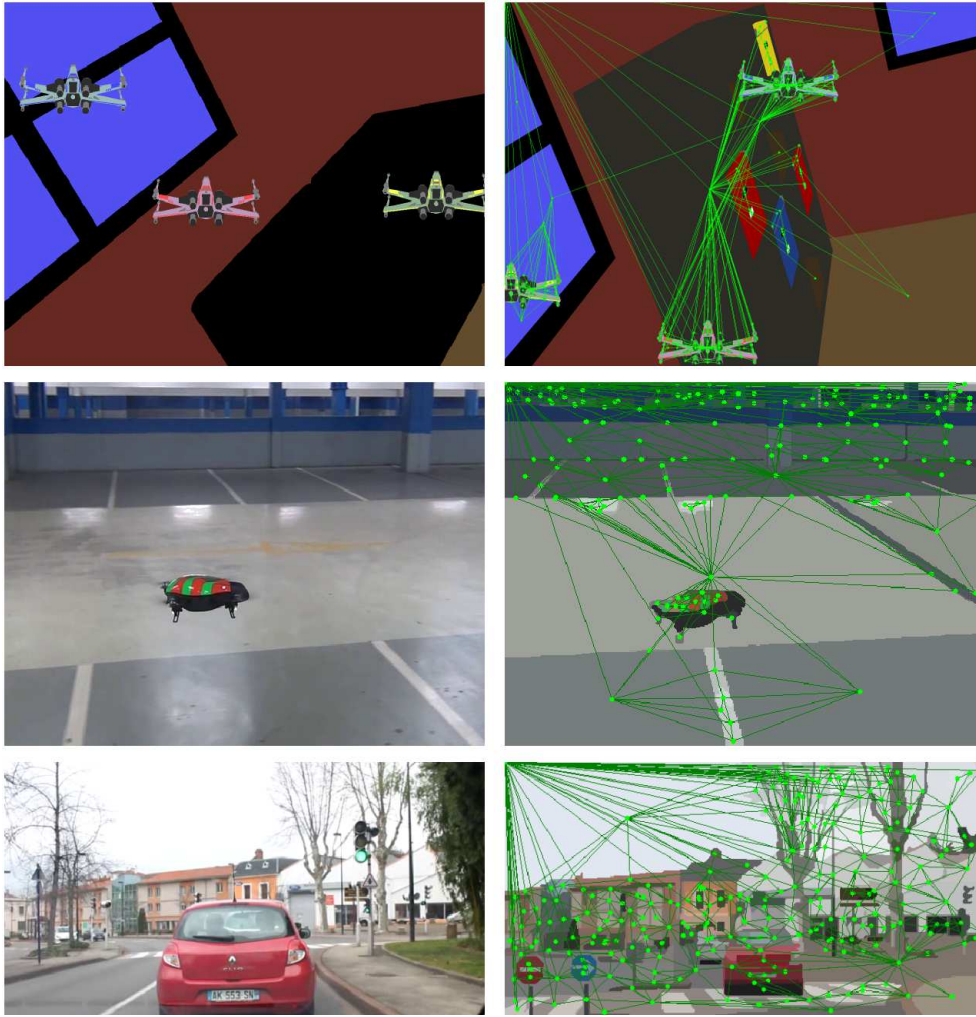


Figure 4.2: Example of frames and RAGs obtained from the synthetic videos (top), from the segmented real drone video (middle), and from the segmented car video (bottom).

were discretized into 10 bins containing the same number of regions, which led to 10 possible node labels. Note that, in this dataset, each graph is a 2-connected graph.

We also represented each frame as a RAG (Region Adjacency Graph). More precisely, the nodes are computed in the same way as for the *Triangulated* dataset, except that there is also one node representing the outer region. Each continuous frontier between two regions is represented by one edge. On average, each frame led to a graph with 245.2 nodes, with an average degree of 2.23, and the labels of the nodes were discretized in the same way as for the *Triangulated* dataset. Contrarily to the graphs in the *Triangulated* dataset, the edges of the target graphs are more meaningful, since they represent adjacencies between regions. Moreover, if different regions have the same barycenters, they are not discarded as for the *Triangulated* dataset. This explains the higher number of nodes in this new dataset.

One disadvantage of the *RAG* dataset, however, is that the generated graphs may not be 2-connected. Since PLAGRAM mines only 2-connected patterns, it is not able to find a pattern that spans on several 2-connected components. Indeed, in the experiments, we found bigger patterns in the *Triangulated* dataset. Nevertheless, interesting patterns

were also found by PLAGRAM in the *RAG* dataset.

Some example frames (left) along with their triangulated (middle) and RAG (right) representations are illustrated in Figure 4.3.

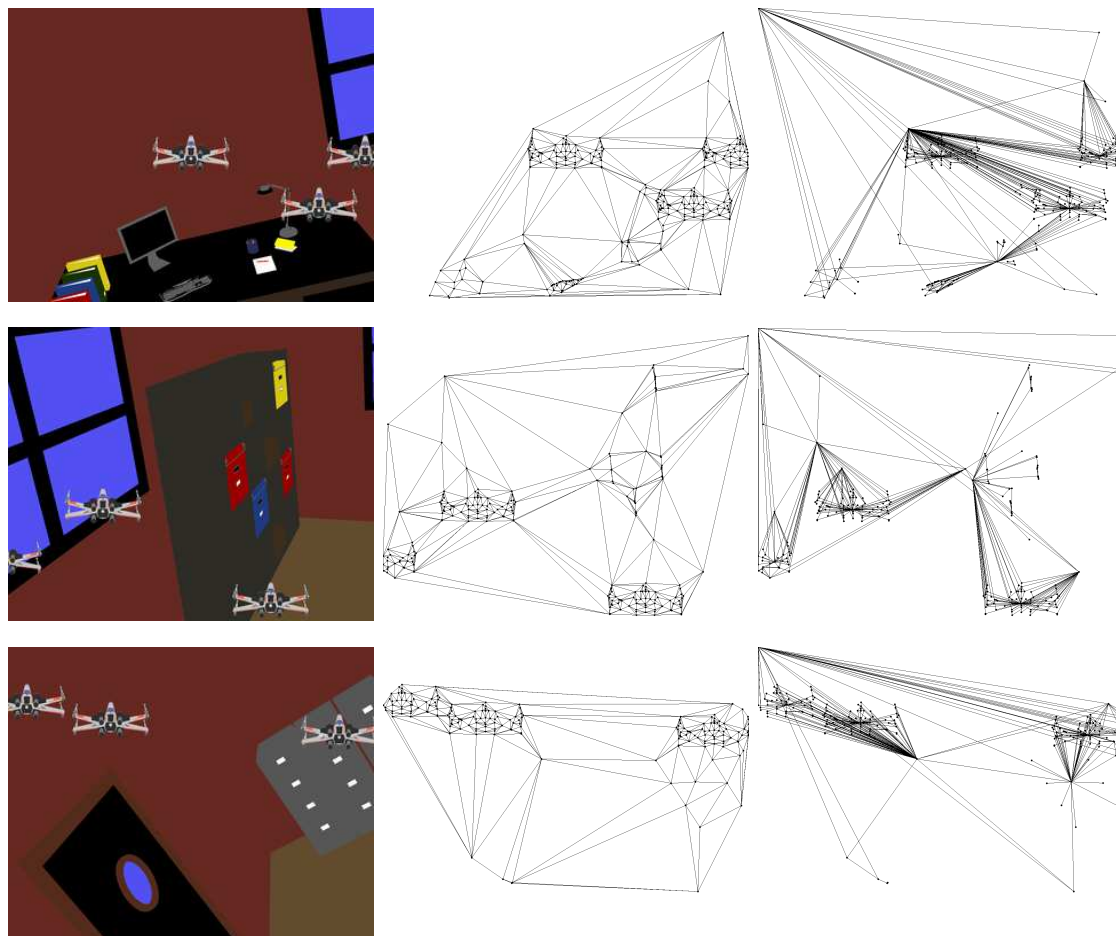


Figure 4.3: Example video frames (left) along with their corresponding triangulated (middle) and RAG (right) representations. In the latter, the upper-left node represents the outer region.

Based on *Anim1*, we produced *Anim2* which is identical except for the color of the X-Wings that is different for each one of them (cf. top of Figure 4.2). These videos were used to assess whether our approach can deal with scenes involving several objects occluding each others and moving out of the field of view.

**Real Videos** The first real video (cf. middle of Figure 4.2) is composed of 950 frames, each RAG has on average 194.5 nodes with an average degree of 5.35. This video shows a drone flying across a covered parking lot. This video is simple but the segmentation still suffers from the illumination changes. The second real video (cf. bottom of Figure 4.2) is made of 5619 frames, each RAG has on average 207.5 nodes with an average degree of 5.5. This video is shot from a car while following another car (the main object). In this video the main object goes out of the field of view, its scale changes, the global illumination changes all the time and it is also longer than the other ones which

allows us to test the efficiency of our approach. This video has been divided into 3 parts (car1000, car2000, car3000) which correspond to the 1000, 2000 and 3000 first frames of the car video. This has been done since the tracking difficulty gradually increases along the video.

For both videos, we use the same modified segmentation algorithm with standard parameters to segment the images. With these videos, we want to assess whether our approach can deal with changing appearances and with the segmentation inaccuracies.

## 2 Object Tracking Using Graph Mining

In Chapter 2, we described the PLAGRAM algorithm. It can be used to efficiently extract frequent plane graph patterns from a database of plane graphs. We also demonstrated that using spatio-temporal constraints could further increase the efficiency of the mining process. This chapter studies how meaningful the extracted patterns are in the context of object tracking.

The appearance of moving objects changes over time and so does their graph representation. Besides, the instability of the segmentation process also results in different graph representations of the same object from one frame to another. Consequently, it is very unlikely to be able to follow an object using one single spatio-temporal pattern. Instead we have to find a way of combining several patterns to obtain complete tracks of the interesting objects. To do so, we use two different strategies described in Section 2.1. Both of them use the spatio-temporal patterns discovered by the DYPLAGRAM\_ST algorithm and described in Chapter 2.

### 2.1 Tracking with Patterns

We first describe a strategy that uses spatio-temporal paths in a global occurrence graph to track object. Then we detail an alternative hierarchical clustering approach.

#### 2.1.1 Spatio Temporal Path

When tracking an object in a real video, we cannot expect the object to be represented by the same graph pattern during the whole video (e.g., due to changes in view point or instability of the segmentation). Thus, if we want to track it, we need to use several spatio-temporal patterns. To do so, we propose to merge the occurrence graph of each pattern into a global occurrence graph, and add similarity edges to it so that similar occurrences of different patterns that appear in the same frame are connected. In this way, it is possible for a path in the occurrence graph, called a spatio-temporal path, to “jump” from a spatio-temporal pattern to another one that has similar occurrences. The similarity between two occurrences is derived from their overlap in term of nodes. It can be efficiently computed by counting how many regions they have in common. Indeed, since the similarity edges connect only occurrences that appear in the same frame, their set of common nodes can be obtained by computing the intersection between their node list.

**Definition 26 (Similarity of two occurrences)** *Let  $o = (i, f)$  and  $o' = (i, f')$  be two occurrences of two different patterns  $P = (V, E, F, f_e, L)$  and  $P' = (V', E', F', f'_e, L')$  with  $f, f'$  the mappings of the nodes of the patterns to the graph of the  $i$ th frame. The similarity between these occurrences is defined as  $sim(o, o') = \frac{|f(V) \cap f'(V')|}{|f(V)|}$ .*

This similarity is used to weight the similarity edges of the global occurrence graph in combination with the spatial distance between occurrences. Note that this similarity is not symmetric in order to favor the transition from smaller patterns to bigger ones. The other edges of the global occurrence graph, i.e., the ones connecting occurrences of the same pattern if they are close in space and time, are weighted according to the temporal distance between the occurrences.

**Definition 27 (Global occurrence graph)** *Given a set of patterns  $\mathcal{P}$ , temporal and spatial thresholds  $\tau$  and  $\varepsilon$ , a similarity threshold  $\mu$ , the global occurrence graph is a weighted oriented graph: its node set is  $V = \cup_{P \in \mathcal{P}} Occ(P)$  and its edge set is  $E = E_{\mathcal{P}} \cup E_{sim}$  where :*

- $E_{\mathcal{P}}$  is the union of the edge sets of all patterns' occurrence graphs. The weight of an edge  $((i, f), (i', f'))$  is  $w = \frac{(i' - i - 1)}{\tau}$ .
- $E_{sim} = \{(o, o', w) \mid o = (i, f), o' = (i', f'), sim(o, o') > \mu\}$  is the set of similarity edges with

$$w = \begin{cases} 0 & \text{if } |V| < |V'| \\ \frac{1}{2} \left( \frac{1 - sim(o, o')}{1 - \mu} + \frac{d}{\varepsilon} \right) & \text{otherwise.} \end{cases}$$

where  $V$  and  $V'$  are the node sets of the patterns corresponding respectively to occurrences  $o$  and  $o'$ , and  $d$  is the distance between the barycenters of  $o$  and  $o'$ .

A spatio-temporal path is a path in the global occurrence graph.

In definition 27, the edges in  $E_{\mathcal{P}}$  are edges between 2 occurrences of the same pattern that are not in the same frame. If these two occurrences are in consecutive frames, the weight is 0 (when  $i' = i + 1$ ) otherwise the weight increases with the number of frames between them (normalized by the temporal threshold  $\tau$ ).

The edges in  $E_{sim}$  are *similarity edges* between 2 occurrences of different patterns that are in the same frame and whose similarity is above  $\mu$ . To favor paths that use large patterns we set the weight of edges going from smaller occurrences to bigger ones to 0. The weight of an edge from an occurrence of a large pattern to a smaller one increases as the similarity decreases and the spatial distance increases. Otherwise the weight on the edges increases as the spatial distance increases and the similarity between the occurrences decreases.

### 2.1.2 Clusters of Spatio-Temporal Patterns

In this second approach, we want to cluster the spatio-temporal patterns. To do so, we first need to define a distance function between them and then a clustering algorithm that can regroup them in clusters corresponding to interesting objects. One of the main difficulties here is to estimate how many clusters to produce, and how to choose the ones that are more likely to represent an interesting object.

**Dissimilarity between spatio-temporal patterns** Each spatio-temporal pattern  $p$  can be represented as a trajectory  $p_{tr} = \{(x_i^p, y_i^p) \mid f_s^p \leq i \leq f_e^p\}$  with  $f_s^p$  and  $f_e^p$  respectively the starting and ending frames of  $p$ . For each spatio-temporal pattern, the coordinates  $(x_i^p, y_i^p)$  of the points of its trajectory are obtained by computing the barycenters of its occurrences in each frame  $i$ . For example, in Figure 4.4, we would



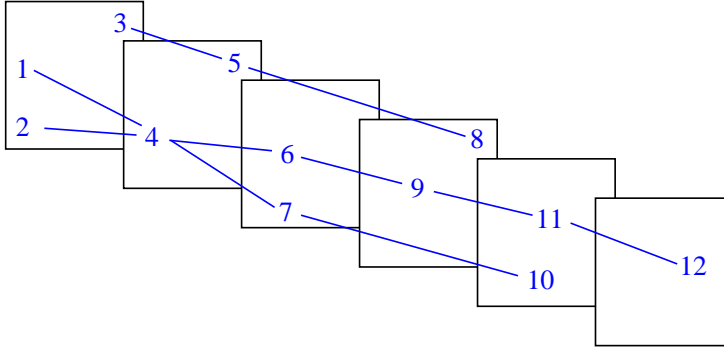


Figure 4.4: Example of an occurrence graph for a given pattern  $P$  which occurs 12 different times in 6 frames of a given video.

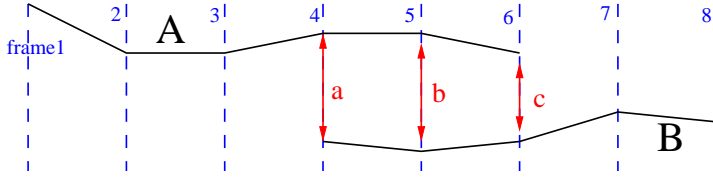


Figure 4.5: Example of two overlapping (on 3 frames) spatio-temporal patterns  $A$  and  $B$

compute the barycenter of occurrences 1 and 2 for the second spatio-temporal pattern. Since the temporal threshold  $\tau$  allows spatio-temporal patterns to have gaps in the sequence of their occurrences, the coordinates of the points of the trajectory in those frames are interpolated between the previous and the next known points.

Let  $A$  and  $B$  be two patterns. Let  $f_s = \max(f_s^A, f_s^B)$  and  $f_e = \min(f_e^A, f_e^B)$ . The distance between two spatio-temporal patterns is defined as:

$$d(A, B) = \begin{cases} \text{if } f_e - f_s > 0 \\ \text{then } d_{traj}(A, B) * (2 - d_{ov}(A, B)) \\ \text{else } \infty \end{cases}$$

$$\text{where } d_{traj}(A, B) = \sum_{f_s}^{f_e} \frac{\sqrt{(x_i^A - x_i^B)^2 + (y_i^A - y_i^B)^2}}{f_s - f_e + 1}$$

$$\text{and } d_{ov}(A, B) = \frac{f_s - f_e + 1}{\min(f_s^A, f_s^B) - \max(f_e^A, f_e^B) + 1}$$

In words, if two patterns never belong to the same frames, their distance is infinite. Otherwise, their distance is the normalized (over the number of common frames) sum of the Euclidean distances between the barycenters of the patterns that appear in common frames. We added a penalty between 1 and 2 to take into account the proportion of common frames compared to the span of the union of the two spatio-temporal patterns. For example, in Figure 4.5, the distance between the two patterns is  $\frac{(a+b+c)}{6-4+1} * (2 - \frac{3}{8-1+1})$ .

**Clustering algorithm** To cluster our spatio-temporal patterns without knowing in advance the number of interesting clusters, we decided to use a simple hierarchical clustering algorithm ([45]) with the distance function previously defined. The main

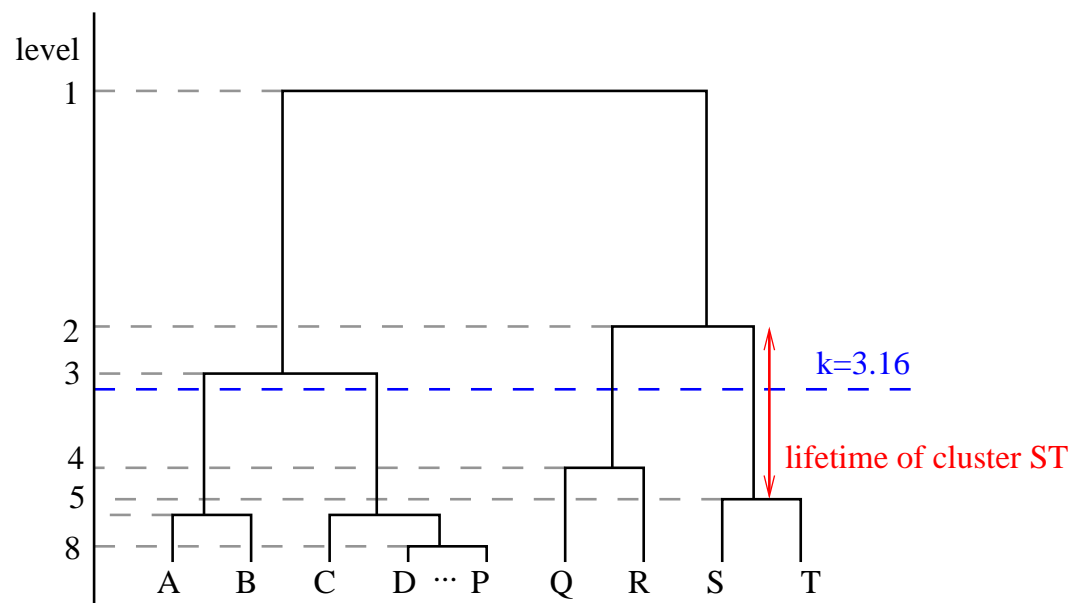


Figure 4.6: Example of hierarchy of clusters with the lifetime of cluster ST depicted with a red arrow. The hierarchy is cut at level 5 because cluster ST has the longest lifetime among those below level  $k = \sqrt{20/2} = 3.16$ .

problem of this algorithm is the choice of the criterion to cut the hierarchy of the dendrogram without any information a priori about the quality of the resulting clustering. We decided to cut the hierarchy at the level of the creation of the cluster with the highest *lifetime* ([44]). The *lifetime* of a cluster corresponds to the difference between the similarity at which it has been formed and the similarity at which it is merged with another cluster. However, the lifetime criterion tends to behave badly in the presence of outliers which are fused at the top of the hierarchy and often have the maximum lifetime (the hierarchy is thus cut at a high level with very few clusters). To overcome this drawback, we decided to ignore the 10% first levels of the hierarchy (note that there are  $i$  clusters at level  $i$ ) before computing the lifetime. A visual example of the lifetime is depicted in Figure 4.6. We also tried an other criteria, called the *gain* ([85]), but this criteria tended to cut the hierarchy at the top, even when ignoring the 10% first levels, which resulted in clusters with low precision.

**Selection of the best clusters in the clustering** Since with our approach, a lot of the spatio-temporal patterns of the background are not part of any precise cluster, the optimal number of clusters is usually much higher than the true number of main objects.

Therefore, after having cut the hierarchy, we still have to decide which clusters are the most interesting. The idea is to rank the clusters and only keep the best ranked ones. In the rest of this document, the *size* of a cluster refers to the number of spatio-temporal patterns composing it and the *length* of a cluster refers to the number of frames it covers. More precisely, the length is computed as the difference between the frame number of the first and last frames the cluster has an occurrence in. We tried different strategies to rank the clusters. We first ranked them according to their length only or size only but the strategy of ranking the clusters according to their length first and then

according to their size gave better results overall. This third strategy still has problems. In particular, in the case where interesting objects do not appear in all the frames of the video, top ranked clusters do not always represent those interesting objects. Instead they often are small clusters composed of few patterns with low discriminative power that cover all the video but do not represent anything interesting. To deal with this problem we changed our ranking strategy to favor the biggest clusters among the ones that covered the majority of the video. More precisely, we first keep all the clusters with length  $l$  such that  $l_{max} \geq l \geq l_{max} - 0.1 \times |\mathcal{D}|$ , where  $l_{max}$  is the length of the longest cluster and  $|\mathcal{D}|$  is the number of graphs in the database (i.e., the number of frames in the video). Within the clusters of this length, we select the one (or randomly among the ones) with the highest number of spatio-temporal patterns and then the highest number of occurrences. This cluster is called the *longest* in the rest of this document.

To decide how many interesting objects should be tracked in the video in a completely unsupervised manner (without selecting them in the first frame), we could either assume that there is only one object, or find among the longest clusters the ones that are sufficiently far from each other. However, in our experiments, we select for each video the  $n$  longest clusters, with  $n$  being the number of main objects in the video. We then measure their precision and recall with respect to the ground truth.

## 2.2 Meaningfulness of the (Spatio-Temporal) Patterns

To evaluate how meaningful our (spatio-temporal) patterns are, before constructing any spatio-temporal path or clusters, we study whether they can be used to track a given object in a video.

We start by introducing two measures which assess how precisely a (spatio-temporal) pattern  $p$  corresponds to a given target object  $o$  in the video frames. These measures, also used later on to evaluate our more elaborate tracking strategies, are adaptations of the popular measures *precision* and *recall* as described below:

- **precision:** fraction of the occurrences of  $p$  (in the target graphs) for which every node maps to the object  $o$  in the corresponding video frames. The intuition behind this measure is to evaluate the *purity* of  $p$ , that is,  $p$  has the maximum precision if it maps only to the object  $o$  and nothing else.
- **recall:** Let  $n$  be the number of frames in which  $o$  is present. The recall is defined as the fraction of  $n$  in which there exists at least one occurrence of  $p$  where every node maps to  $o$ . Here, the intuition is to evaluate the *completeness* of  $p$ . More precisely, the idea is to check whether the occurrences of  $p$  map to all occurrences of  $o$  in the set of video frames.

Since our algorithms are exhaustive, that is, they mine for all frequent (spatio-temporal) patterns in the graph database without supervision, the mining results may consist of different (spatio-temporal) patterns corresponding to different objects, or even to no specific one (w.r.t. the proposed measures). Therefore, to follow a specific object in the video, the user should be able to select from the entire set of output (spatio-temporal) patterns those that correspond to this object. A basic strategy for this task is the following:

1. First, the user selects a frame area where there exists an object he or she is interested in tracking, that is, the target object. This is done in a user selected frame, referred to here as  $f$ , where this object occurs.

Support	<i>Triangulated</i>		<i>RAG</i>	
	precision (%)	recall (%)	precision (%)	recall (%)
721	96.2	99.8	97.1	99.9
711	97.6	98.9	97.2	99.8
701	99.3	97.7	96.5	99.0
691	99.7	96.3	93.9	95.6
681	99.8	95.0	92.8	93.9
671	99.8	93.7	92.5	93.5
661	99.9	92.5	92.5	93.5
651	99.9	91.0	91.8	92.6

Table 4.1: Average precision and recall (in percentage) computed for the patterns selected at step 3 of the proposed object tracking strategy.

2. Afterwards, the user starts the graph mining process by executing either PLAGRAM with a given minimum support or DYPLAGRAM with a given minimum support and time constraint as input, followed by the post-processing step described in section 3.2.4 with a spatial constraint. Alternatively, spatio-temporal patterns can be directly extracted using DYPLAGRAM\_ST.
3. Next, the (spatio-temporal) patterns that have no occurrences in the user-selected area, in frame  $f$ , are discarded. The remaining patterns are considered the target patterns, i.e., those that characterize the target object.
4. Finally, all the occurrences of the target (spatio-temporal) patterns are mapped to the video frames, allowing the user to detect the position of the target object through the video.

### 2.2.1 Output of Plagram (plane graph patterns)

We first evaluated our strategy on the patterns returned by PLAGRAM (the plane patterns). In those experiments we used the simple video with only one X-Wing. We checked whether it would be possible to follow the X-Wing in this basic video by considering the patterns that matched it in the first frame, i.e., patterns that were inside the user selected area. As might be expected, those patterns had different precision and recall with respect to the X-Wing. Some examples are given in Figure 4.7. In (b) and (c), we show 2 different occurrences of a pattern with 100% support, 52% precision, and 100% recall in the *RAG* dataset. Now, consider the graph in (d). It illustrates an occurrence of a pattern with support of 378 frames in the *RAG* dataset. Note that this occurrence had a node outside of the X-Wing area; this decreased the precision of the corresponding pattern. Indeed, it had 0% precision and recall.

After executing step 3, we got the patterns whose average precision and recall (in percentage) are shown in Table 4.1.

Observe that the selected patterns had, on average, very good quality, making step 4 successful. Considering the *Triangulated* dataset, the average precision increased in inverse proportion to the minimum support, while the average recall decreased with the minimum support. Here, lower minimum support led to bigger patterns with higher precision and lower recall. In the *RAG* dataset, the behavior was different: big patterns had nodes that did not map to the X-Wing. In addition, small patterns with low support did not have good precision nor recall. As a consequence, the average precision and recall decreased with the minimum support.

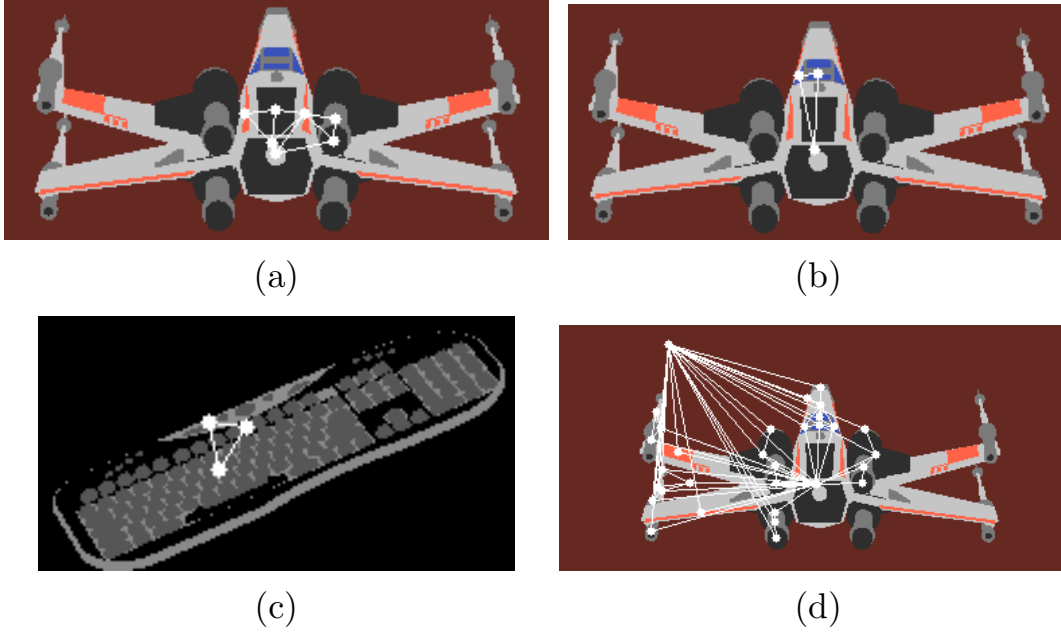


Figure 4.7: (a): pattern with 100% precision and recall in the Triangulated dataset. (b) & (c): 2 occurrences of the same pattern (the X-Wing and a keyboard, respectively) with 52% precision and 100% recall in the RAG dataset. (d): example pattern with 0% precision and recall in the RAG dataset.

However, in a less simplistic context (e.g., when multiple identical objects are present), the precision and recall of the patterns returned by PLAGRAM drops dramatically. This is due to the fact that, in the video used for these experiments (see Figure 4.3), the 3 X-Wings may overlap and two of them can partially go out of the video frames. Besides, as the target X-Wings are identical, some ambiguities may happen w.r.t the target patterns. For example, a target pattern may be very frequent just because it maps to multiple X-Wings and thus appear in almost every frame, but imprecise (i.e., with a low precision) with respect to a given X-Wing  $o$  if it maps not only to  $o$ , but to different X-Wings through the video. Therefore, to track a given object in our more complex video, the use of spatio-temporal constraints becomes necessary.

### 2.2.2 Output of DyPlagram and DyPlagram\_st

To check whether the defined spatio-temporal patterns can well represent the individual trajectory followed by several similar objects, or the trajectory of objects in real videos, we used the previously described strategy on the video Anim1 that shows 3 identical X-Wings moving in a room and using the size discretization to label the nodes of the RAGs. We also performed experiments on the real video with the drone to assess the meaningfulness of the spatio-temporal patterns when the appearance of the target changes and segmentation errors are introduced.

**Experiments on Anim1** For DYPLAGRAM, we first extracted all frequent patterns with a frequency threshold of  $\sigma = 721$ , corresponding to the number of frames in the video, and a temporal constraint  $\tau = 1$  to focus on patterns that appear in every frame (note that some occurrences of the same pattern may correspond to different

	DYPLAGRAM with post-processing			DYPLAGRAM_ST		
	Precision(%)	Recall(%)	# ST patterns	Precision(%)	Recall(%)	# ST patterns
$\epsilon = 10, \sigma_{st} = 10$						
X-Wing 1	78	7	151	78	7	<b>114</b>
X-Wing 2	72	3	129	<b>95</b>	<b>3</b>	71
X-Wing 3	87	2	<b>131</b>	88	2	<b>84</b>
$\epsilon = 20, \sigma_{st} = 50$						
X-Wing 1	77	15	73	82	17	65
X-Wing 2	93	26	43	<b>100</b>	29	<b>39</b>
X-Wing 3	100	10	60	100	10	60
$\epsilon = 170, \sigma_{st} = 50$						
X-Wing 1	45	38	27	<b>51</b>	42	24
X-Wing 2	51	10	15	49	8	17
X-Wing 3	60	12	21	<b>69</b>	13	19

Table 4.2: Evaluation of the spatio-temporal patterns issued from all patterns with  $\sigma = 721$  and  $\tau = 1$  for DYPLAGRAM and for DYPLAGRAM\_ST. The labels are created from the size of the region. For both algorithms, the third column indicates how many spatio-temporal patterns have been discovered. Those experiments were conducted on the synthetic video Anim1 with 3 identical X-Wings.

spatio-temporal patterns). Then, we post-processed them to generate spatio-temporal patterns using the strategy described in 3.2.4. The same temporal threshold  $\tau = 1$  and frequency threshold  $\sigma = 721$  were used to directly extract comparable spatio-temporal patterns with DYPLAGRAM\_ST.

Next, the precision and recall of every spatio-temporal pattern whose first occurrence mapped to an object  $o$  in a video frame  $i$  were computed with respect to the object  $o$ . The precision allows to assess if spatio-temporal patterns are robust, i.e, if they tend to follow the same object on all frames they have an occurrence in. The recall tells us how much of a target, in term of number of frames, spatio-temporal patterns cover.

Table 4.2 summarizes the results obtained with the spatio-temporal patterns generated by post processing the frequent patterns of DYPLAGRAM, and the spatio-temporal patterns of DYPLAGRAM\_ST. The spatio-temporal patterns were generated with a minimum  $\text{freq}_{st}$  threshold  $\sigma_{st} = 10$  and 50, and 3 different spatial thresholds  $\epsilon$  of 10, 20, and 170 pixels (from 20 to 160 pixels, the results were quite similar and thus are not reported here). For each experimented pair  $(\sigma_{st}, \epsilon)$  and for each target X-Wing in the video, the first two columns give the average precision and recall computed for its associated spatio-temporal patterns (as defined in our strategy). In addition, the third column shows the total number of such patterns.

Table 4.2 shows that the spatio-temporal patterns obtained with DYPLAGRAM\_ST are in general less numerous, more precise and have a better recall than the ones obtained with DYPLAGRAM. The distance threshold  $\epsilon$  has an important impact on the obtained results. Indeed, if it is set too low (to 10 pixels, in our example), we obtain spatio-temporal patterns with high average precision for each X-wing as different occurrences of patterns which map to different X-wing are very well distinguished. However, this leads to a low average recall: since only very close occurrences of the same pattern are linked, the spatio-temporal patterns tend to be short (i.e., have low  $\text{freq}_{st}$ ). When using a distance threshold  $\epsilon = 10$ , no spatio-temporal patterns with  $\text{freq}_{st} \geq 50$  were found for X-wing2 for DYPLAGRAM, which explains why we used  $\sigma_{st} = 10$  in this case. Conversely,

Table 4.3: Precision and recall computed for the spatio-temporal patterns produced by DYPLAGRAM\_ST on the real video with  $\sigma = \sigma_{st}$ , and  $\mu = 0.65$

$\tau$	$\sigma_{st}$	$\epsilon = 10$			$\epsilon = 20$		
		Precision(%)	Recall(%)	# ST patterns	Precision(%)	Recall(%)	# ST patterns
10	100	<b>100</b>	26.18	10	<b>92.48</b>	22.97	13
	50	93.55	17.40	20	91.35	15.44	25
	10	89.78	2.87	294	89.70	2.72	334
25	100	91.28	35.34	11	89.02	30.03	14
	50	90.28	25.12	18	83.79	20.14	24
	10	88.90	3.18	307	89.47	2.94	358
100	100	89.52	<b>38.21</b>	14	89.02	<b>31.03</b>	19
	50	92.27	24.38	27	90.30	22.45	30
	10	89.01	4.03	258	89.88	3.63	302

for a higher  $\epsilon$  of 170 pixels, the average precision drops as the different X-wings are not well distinguished anymore. For example, it was possible to obtain spatio-temporal patterns with higher recall for the X-Wing 1 (when comparing to the other experiments), but, they had low average precision. Since the X-Wing 1 gets partially out of the video frames around 6 times, a higher number of spatio-temporal patterns were derived for this X-wing for  $\sigma_{st} = 50$  and  $\epsilon$  of at least 20, which represent the different time intervals where this X-wing is visible through the video. As another example, the X-Wing 2 is hidden only twice by the X-Wing 3 (during around 15 frames) and never goes out of the video frames. This explains the lower number of spatio-temporal patterns found for this object, also for  $\sigma_{st} = 50$  and  $\epsilon \geq 20$ . Note that, in the case of our example video, increasing the time constraint  $\tau$  could increase the length of the spatio-temporal patterns and thus their recall but this would lead to a lower precision.

**Experiments on the drone video** The aim of those experiments is to demonstrate that spatio-temporal patterns can serve as a basis to track objects in real videos. We experimented on the influence of different values for the spatio-temporal thresholds, using  $\tau = 10, 25, 100$  and  $\epsilon = 10, 20$  (above 20 the precision started to drop significantly which is expected for large  $\epsilon$  values), and different values for the frequency threshold with  $\sigma_{st} = 10, 50$  and 100. The precision and recall results for the spatio-temporal patterns returned by DYPLAGRAM\_ST under those parameters are presented in Table 4.3.

As expected, the precision is a little higher with  $\epsilon = 10$  (100% for  $\epsilon = 10$  when  $\tau = 10$  and  $\sigma_{st} = 100$  against 92.48% for  $\epsilon = 20$ ). The fact that the average recall also decreases with a higher distance is more surprising at first glance. This is explained by the fact that most of the time,  $\epsilon = 10$  is enough to follow the drone, but sometimes the drone or the camera movement accelerates. In those cases a higher distance might give longer and better spatio-temporal patterns but also might introduce some noisy ones which would decrease the average recall and precision.

The average recall also decreases when we lower  $\sigma_{st}$ . This is due to the fact that when using a low  $\sigma_{st}$  DYPLAGRAM\_ST outputs short spatio-temporal patterns that necessarily have a low recall. Lowering  $\sigma_{st}$  slightly reduces the precision of the spatio-temporal patterns but increases their number.

As also expected, higher gaps lead to better recall (38.21% for  $\tau = 100$  when  $\epsilon = 10$

and  $\sigma_{st} = 100$  against 26.18% for  $\tau = 10$ ) as well as improve the coverage of the spatio-temporal patterns in the whole video. The precision does not seem to be influenced by  $\tau$  when we allow small spatio-temporal patterns (i.e., a low  $\sigma_{st}$ ).

Overall this series of experiments have shown that spatio-temporal patterns are robust and can follow a target with high precision. The fact that a single spatio-temporal pattern is not enough to track an object across all the frames of a video is also confirmed by the low recall results. In the next sections we will present the experiments we conducted to show how this problem can be solved with the spatio-temporal paths or clusters of spatio-temporal patterns.

### 2.3 Spatio-Temporal Paths for Object Tracking

To assess the effectiveness of the spatio-temporal paths for object tracking, we apply the following strategy. We first build the occurrence graph and then, for each target object, we select the occurrences matching it in the first frame. Then we compute the path of lowest cost starting from those occurrences and reaching the last frame using Dijkstra’s shortest path algorithm. This means that this strategy is better suited to cases where the target appears in the last frame. Although it could still follow a target that is not present in the end of the video, in the last frames, this strategy would drift to some pattern that does not represent the object. In all experiments reported here we use a similarity of  $2/3$  ( $\mu = 0.65$ ). We also tried with a similarity of  $3/4$  ( $\mu = 0.75$ ) but in this caused the occurrence graph to have too few edges to find a complete track of any object. With a similarity of  $1/2$  ( $\mu = 0.5$ ) the occurrences graphs took a lot of space in memory because of the number of edges between the occurrences and the track obtained drifted easily to elements of the background.

In practice the minimum support threshold  $\sigma$  can be set, for example, to  $1/5$  of the total number of frames (to make sure that the patterns occur enough and help the mining process). By default, it will be equal to the  $\sigma_{st}$  threshold.  $\sigma_{st}$  should be set as low as possible (depending on available memory). The  $\tau$  should, in general, be set as high as possible (as will be shown in the experiments). The  $\epsilon$  constraint depends on the motion speed of the target object and on the resolution of the video. Most of the time we use 20 pixels.

**Evaluation of the Spatio-Temporal Path for Object Tracking** For the synthetic videos Anim2 with the 3 different airplanes, and for the drone video, we report the precision and recall results for the spatio-temporal paths. Each time we selected the spatio-temporal pattern with lowest weight starting from an occurrence in the area selected by the user and ending in the last frame. The precision and recall results are computed on the occurrences taken by the spatio-temporal path with lowest weight.

**Experiments on the Synthetic Video Anim2** The experiments reported in Table 4.4 show the precision and recall results for the paths obtained on the synthetic video when varying the gap between 10 and 100.

Because of the nature of the video, we use a global minimum support  $\sigma$  of 250 in order to prune the number of frequent patterns. Indeed, since the synthetic video has been especially made to produce stable graphs, DYPLAGRAM\_ST returns a lot of frequent patterns on this dataset which leads to a huge global occurrence graph that possibly does not fit into memory for processing. To be able to perform various experiments, especially with the size discretization which does not permit to distinguish the three



Table 4.4: Evaluation of the spatio-temporal path with  $\sigma = 250$ ,  $\sigma_{st} = 150$ ,  $\mu = 0.65$ ,  $\epsilon = 20$ . The numbers between parenthesis correspond to the best precision and recall of the best path in term of recall, and the emphasized results are the best results for each X-Wing

	$\tau$	Size Discretization			Color Discretization		
		Precision(%)	Recall(%)	Paths	Precision(%)	Recall(%)	Paths
X-Wing 1	10	<b>98.32</b> (99.72)	<b>97.50</b> (99.30)	34	93.92 (99.74)	93.60 (99.86)	21
X-Wing 2		<b>99.63</b> (99.73)	<b>97.26</b> (98.19)	24	98.65 (100)	<b>96.82</b> (99.02)	17
X-Wing 3		<b>9.49</b> (16.64)	<b>8.70</b> (15.39)	4	- (-)	- (-)	0
X-Wing 1	25	95.79 (100)	94.59 (99.02)	38	<b>99.17</b> (99.73)	<b>98.40</b> (100)	21
X-Wing 2		65.66 (99.61)	64.61 (98.05)	32	98.54 (100)	96.34 (99.02)	20
X-Wing 3		2.93 (9.09)	2.50 (8.59)	29	31.95 (31.95)	29.54 (29.54)	2
X-Wing 1	100	79.05 (100)	74.37 (94.31)	42	97.76 (100)	95.36 (99.30)	29
X-Wing 2		72.57 (97.53)	67.05 (93.62)	35	<b>98.87</b> (100)	96.30 (99.02)	39
X-Wing 3		5.42 (18.46)	4.82 (16.36)	31	<b>86.27</b> (90.52)	<b>75.92</b> (82.80)	23

X-Wings at the mining step, we set the  $\sigma_{st}$  to 150 (although as already discussed, it is better to set it as low as possible).

Overall, we obtain very good results for the first two X-Wings (precision and recall close to 100%). We can clearly see the lack of discriminative power of the size discretization when the gap increases. Indeed the paths start to follow different X-Wings, reducing their precision and their recall. For those two X-Wings the color discretization always shows good results, with average precisions and recalls close to the ones of the best paths (values in brackets). Since the 3rd X-Wing moves back and forth horizontally across the field of view (getting almost completely out every 120 frames), only few paths starting on this X-Wing manage to reach the end of the video when we use a low gap. The paths which uniquely follow this X-Wing are thus more expensive than other paths on which the algorithm can "jump" using the similarity edges, decreasing the precision and recall. As we can see, increasing the gap allows to overcome this problem with the color discretization while keeping good results for the other two X-Wings.

**Real Video with the drone** The experiments reported in Table 4.5 were made without using a global minimum support threshold (which is equivalent to set  $\sigma = \sigma_{st}$ ). Because of the segmentation, this dataset is a lot less stable than the synthetic one resulting in less frequent patterns. For this one, only the color discretization gave good precision/recall results (we also tried the size and some other color discretizations using the HSV color space but our simple discretization of the RGB space worked better).

Table 4.5 shows the results for the spatio-temporal patterns of DYPLAGRAM\_ST on the real dataset for the color discretization.

A distance  $\epsilon$  equal to 20 gives the best results in most cases with high precision and good recall (99.0380.63 $\sigma_{st} = 10$  for example). However, the values for  $\tau = 10$  show the limits of the use of the shortest path algorithm to tackle our problem. Similarly to what was happening with the third X-Wing in the synthetic video, the shortest path might not always be following the object we want to track if elements in the background or other objects offer better stability than the original target and are close enough to "jump" on them.

The results with our preferred setting (low  $\sigma_{st} = 10$ , high  $\tau = 100$  and a distance

Table 4.5: Precision and recall computed for the spatio-temporal paths for the real video with  $\sigma = \sigma_{st}$  and  $\mu = 0.65$ , using the color discretization to label the nodes of the graphs.

$\tau$	$\sigma_{st}$	$\epsilon = 10$			$\epsilon = 20$		
		Precision(%)	Recall(%)	Paths	Precision(%)	Recall(%)	Paths
10	100	96.30 (96.30)	67.89 (67.89)	1	98.23 (100)	80.94 (82)	2
	50	98.25 (100)	70.00 (71.26)	2	26.16 (38.96)	24.03 (36.21)	3
	10	91.93 (93.34)	69.60 (70.63)	8	18.75 (36.09)	17.88 (34.73)	8
25	100	98.43 (100)	68.89 (70)	6	98.51 (100)	78.68 (79.68)	6
	50	98.66 (100)	69.05 (70)	7	98.72 (100)	78.82 (79.68)	7
	10	99.06 (100)	69.36 (70.21)	10	<b>99.03 (100)</b>	<b>80.63 (81.36)</b>	10
100	100	100 (100)	67.42 (67.78)	8	100 (100)	77.52 (79.68)	9
	50	100 (100)	67.36 (67.68)	9	100 (100)	77.54 (79.68)	9
	10	100 (100)	67.21 (67.78)	10	99.26 (100)	79.17 (79.78)	10

$\epsilon = 20$ ) show that the spatio-temporal paths can indeed be used to follow an object in the video. The similarity edges introduced are very useful to increase the recall of the patterns and experiments with a higher similarity constraint (for example with  $\mu = 0.8$ ) provide worst results. This shows the importance of this "inexact" matching phase in the process. On the downside, the choice of the labels on the node (here it is a color information) seems to play a very important role to get interesting spatio-temporal patterns although it is difficult to evaluate in an unsupervised setting what could be the best ones. One solution could be to attach more diverse informations on the labels of the nodes to overcome this problem.

## 2.4 Clusters of Spatio-Temporal Patterns for Tracking

In this section we present the experiments we conducted to show that the top clusters, according to our ranking strategy, correspond to interesting objects and can be used to follow those objects.

### 2.4.1 Experimental Design

To assess the quality of the tracks returned by our approach, we compare our algorithm to two other state-of-the-art algorithms called TLD ([87]) and CT ([134]). We also apply our algorithm on the video segmentation of [78]. To summarize we compare the 4 following approaches:

- TLD (Track Learn Detect) is a tracking algorithm [87] that requires manual selection of the target.
- CT (Compressive Tracking) is a tracking algorithm [134] that also requires manual selection of the target.
- TRAP is our tracking algorithm which mines frequent spatio-temporal patterns and clusters them. It uses the simple segmentation algorithm presented in [74] for the real video (and the original regions for the synthetic ones). The value for the three parameters of the algorithm ( $\tau$ ,  $\sigma_{st}$  and  $\epsilon$ ) are discussed below.

- TRAP + VS (Video Segmentation) uses the second type of segmentation presented in [78].

For the clusters obtained with our approach, the *precision* corresponds to the proportion of occurrences of the cluster that have all their nodes in the bounding box of the ground truth (at the corresponding frame). The *recall* of a cluster is the number of frames in which at least one occurrence of this cluster has all its nodes in the bounding box of the ground truth.

*TLD* and *CT* are given the ground truth of the first frame of each video as input. Both algorithms return a sequence of bounding boxes representing the track of the followed objects. The *precision* is the area the bounding boxes of the track and of the ground truth have in common, divided by the area of the bounding boxes of the track. The *recall* of the algorithm is the number of frames in which the center of the bounding box of the track is inside the bounding box of the ground truth.

As explained in Section 2.1.2, the choice of the clusters that are used to track the objects of interest is an important problem. In the experiments, we will show the results for the *Longest* cluster as defined in Section 2.1.2 but also the results for the *Best* cluster in the hierarchy (we chose the best cluster for all possible cut of the clustering hierarchy). This best cluster is the one for which the  $Precision * Recall * 100$  is the highest. Of course, these “best” results are just given to assess the possible improvements for our algorithm since they cannot be used in an unsupervised setting. In some experiments, the two criteria we use (cut with the lifetime and keep the longest cluster) are not always the best but we can show that a very good cluster exists and could be found using a different criteria.

**Parameters of DyPlagram.st** The spatial threshold  $\epsilon$  should be high enough depending on the motion of the objects and the motion of the camera. This can be estimated on the first frames of the video using optical flow techniques (see [69] for more details). However, setting this to 20 (pixels) for all experiments gave sufficiently good results. In general, giving a high value for this parameters will increase the mining time but will not harm the results. Similarly the time threshold  $\tau$  is set for all videos to 25 frames (1 second of the video). Again, this may not be the best set of parameters especially for the car video which is the most complex to deal with. The frequencies thresholds ( $\sigma$  and  $\sigma_{st}$ ) should be set after having found a working  $\tau$  and  $\epsilon$  to obtain a significant number of spatio-temporal patterns ( $600 < \#patterns < 2000$ ). A too large number would also slow down the algorithm. By default  $\sigma = \sigma_{st}$ . Note that  $\sigma$  controls the frequency of the patterns from which the spatio-temporal patterns can be generated. However, a very high  $\sigma_{st}$  threshold (for example, more than 20% of the length of the video) means that the structure of the object (and thus of the patterns representing it) should not change at all during 20% of the frames which is not very reasonable for most of the real videos that are recorded by amateurs. Thus, we impose that  $\sigma_{st}$  is always bellow 20% of the length (in frames) of the video. If the number of patterns is still too big with this bound, we can increase  $\sigma$  to get inside the  $\#patterns$  bounds.

## 2.4.2 Results

We now present the results obtained for our clusters of spatio-temporal patterns in term of tracking quality and efficiency.

		Anim 1: Identical Objects						Animation 2: $\neq$ Objects					
		Obj 1		Obj 2		Obj 3		Obj1		Obj2		Obj3	
		P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
TLD		22	14	<b>90</b>	17	0	0	14	13	36	5	0	0
CT		39	52	0	0	0	0	68	96	0	0	0	0
TRAP	Longest	<b>97</b>	<b>90</b>	41	<b>99</b>	<b>21</b>	<b>63</b>	<b>100</b>	<b>99</b>	<b>91</b>	<b>99</b>	8	12
	Best	99	90	92	88	87	49	100	99	91	99	72	92
VS+C	Longest	14	14	47	55	35	38	91	95	67	84	18	43
	Best	100	52	97	63	100	21	91	95	97	63	53	45

Figure 4.8: Precision and Recall of the CT, TLD and TRAP algorithms using the standard color segmentation, the TRAP algorithm using the video segmentation (TRAP+VS) and the video segmentation alone with a clustering phase (VC+C) on the two synthetic videos Anim1 and Anim2.

		Drone		Car 1000		Car 2000		Car 3000	
		P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
TLD		63	88	65	68	<b>55</b>	<b>46</b>	<b>55</b>	<b>31</b>
CT		<b>84</b>	<b>99</b>	9	14	8	8	5	5
TRAP	Longest	81	99	92	83	10	98	4	52
	Best	97	99	90	98	90	51	90	34
VS+TRAP	Longest	24	95	<b>92</b>	<b>90</b>	5	82	5	65
	Best	95	94	93	98	85	54	85	36
VS+C	Longest	<b>90</b>	<b>100</b>	0	0	0	0	0	0
	Best	100	100	95	100	84	100	98	79

Figure 4.9: Percentage of Precision (P) and Recall (R) of the CT, TLD and TRAP algorithms using the standard color segmentation, the TRAP algorithm using the video segmentation (TRAP+VS) and the video segmentation alone with a clustering (VC+C) on the two real videos.

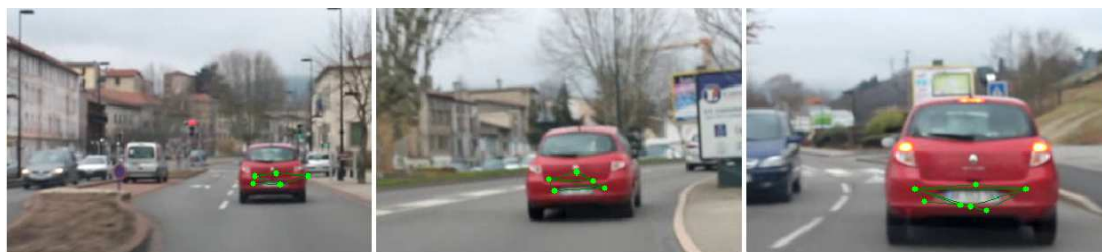


Figure 4.10: Occurrences of frequent patterns (in green) in the longest cluster for the first 1000 frames of the car video

## Tracking quality

**Synthetic videos** In Figure 4.8 we can see that CT and TLD do not give good results on the synthetic video especially when the 3 planes are identical. Indeed, the initial bounding box given in the ground truth includes a lot of background between the wings of the planes which corrupts the appearance model learned. Besides, there are occlusions between the objects and their rapid changes in direction make them hard to track. Our approach gives good results (97/90 P/R for Anim1 and 100/99 for Anim2) on the first plane which is the most stable. However there is no really good cluster (where the recall

and the precision would be both above 90%) in all the hierarchy representing the second and the third object. For the second object, the best cluster has 92% precision and 88% recall but this cluster exists only when cutting the hierarchy at 11 clusters whereas our lifetime criteria cuts the hierarchy at 252 clusters and thus does not allow us to find the best one. For the third object, the best cluster was only 361 frames long so it was not selected as the longest one. Because the third object goes almost completely out of the field of view for 3 to 4 seconds several times in the video, the clusters representing this object were easily split. When directly clustering the patterns extracted from the video segmentation (VS+C) and for Anim1, the best clusters have a high precision but their recall is low, reaching 63 (less for the longest cluster). This comes from the fact that the video segmentation is of course less accurate than the original segmentation and tends to over-segment regions due to the frame-to-frame region matching which decreases the relevance of the patterns.

The results for Anim2 show that the color difference between the three objects usually helps all the trackers (except for TLD). For our approach, the longest clusters at the highest lifetime were the best ones in the hierarchy as can be seen in Figure 4.11 (top). The difference between the objects was discriminative enough to be able to follow the third object with a best cluster with 72% precision and 92% recall. Unfortunately this good cluster was at the 14th level of the hierarchy while it was cut at the 70th level as can be seen in Figure 4.11 (middle). In this later case the size of the best cluster decreases around the 50th level of the hierarchy which causes it to be ranked lower than bigger clusters that do not match the third object.

The video segmentation did less mistakes on this animation and the results are thus better for the VS+C method. As we can see, the best cluster for the first X-Wing was the longest one with a precision of 91% and a recall of 95%. For the second X-Wing, the recall drops to 84% and the precision to 67%. For the last object, there is not enough patterns to build good clusters (the best one only has 53% precision and 42% recall).

**Real videos** TLD and CT both track the drone for almost all the video, the former with 63/88% (P/R) and the later with 84/99% (P/R) (see Figure 4.9). TLD loses it for some frames which results in a lower recall. Due to the large size of the output bounding box in some frames, the precision is lower than for our approaches for both algorithms. TRAP also follows the drone with 99% recall, but the longest cluster is less precise than the best cluster (81% versus 97%). Note that just reducing  $\sigma_{st}$  to 10 in this case would allow us to find the best cluster. Clustering the spatio-temporal patterns extracted from the video segmentation (VS+TRAP) also produces some good clusters but not at the level the lifetime cuts the hierarchy. Thanks to a high number of spatio-temporal patterns, VS+C obtains good results (the longest cluster has 100% recall and 90% precision).

From Figure 4.9, we can confirm that the car video is a much more difficult tracking problem. TLD follows the car until the frame 1305, losing it occasionally, but never with a good precision. CT never succeeds in following the car. For both types of segmentation, the longest cluster returned by TRAP follows the car until the frame 1200 and then loses it. At this point of the video the car is small and both segmentation segmented it in only one region. Since occurrences of frequent spatio-temporal patterns have at least 3 nodes (1 face, this is imposed by the DYPLAGRAM\_ST algorithm), there is none matching the car in this part of the video. The best patterns for the first 2000 and 3000 frames all end at this frame, and, since there is no other long pattern matching the car, the

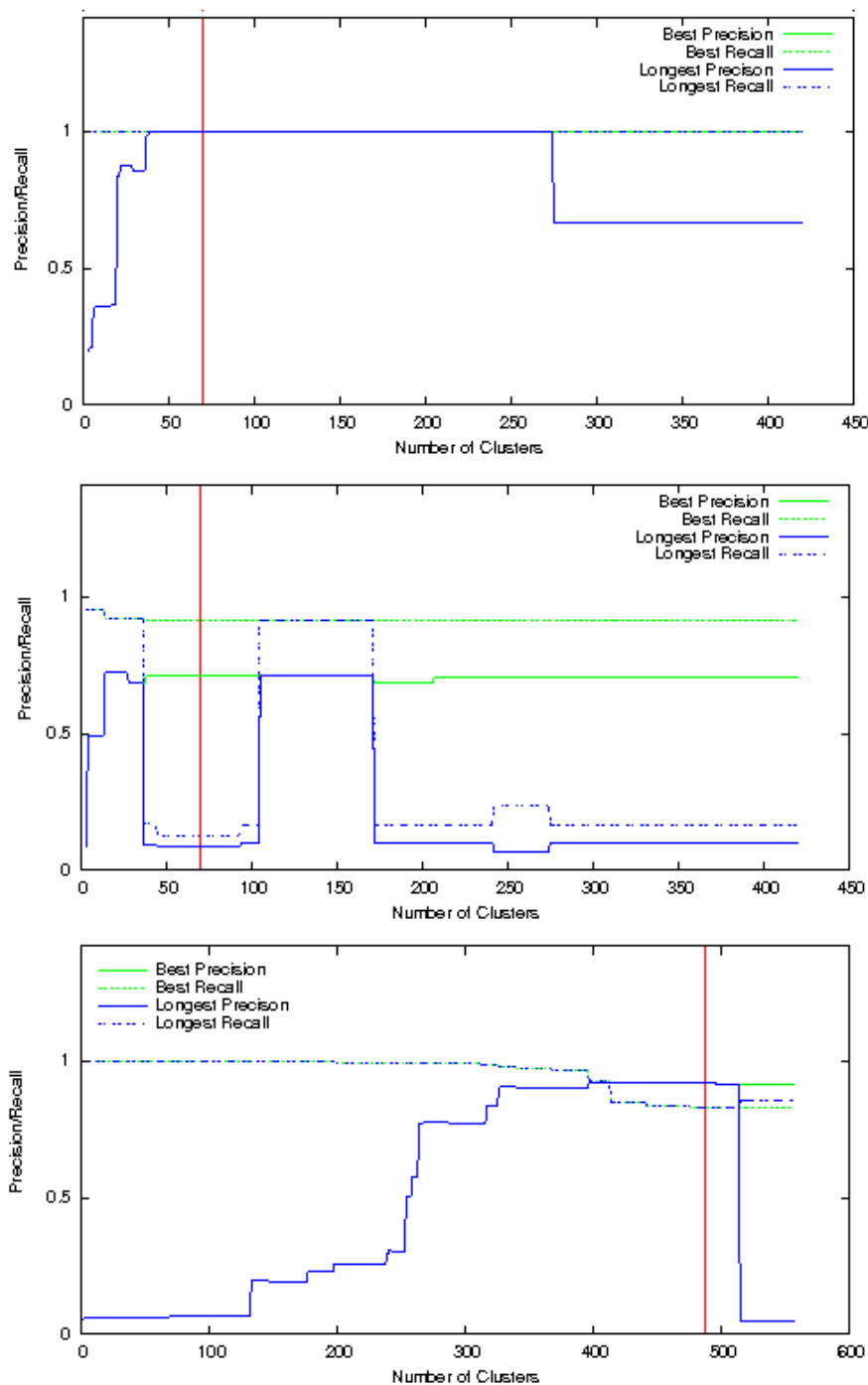


Figure 4.11: Precision and recall results of the best and longest clusters output by TRAP for the object 1 (top) and object 3 (middle) of animation 2 and for the car (bottom) for car1000. The vertical red line is the lifetime cut.

longest clusters has a bad quality. As shown in tab 4.12, augmenting the gap allows us to skip the frames of the video where the car is too small which produces better results. However, the algorithm faces the same situation for a longer time at the frame 2300. This shows that if the gap threshold  $\tau$  can allow us to deal with some situations where the object is hard to detect, it would be better to introduce a mechanism specifically

		Car1000		Car2000		Car3000	
		P	R	P	R	P	R
TRAP	Long	90	99	96	73	7	83
	Best	90	99	93	87	92	61
VS +TRAP	Long	90	99	93	87	7	84
	Best	92	98	88	91	22	39

Figure 4.12: Percentage of Precision ( $P$ ) and Recall ( $R$ ) obtained for the car video when increasing the gap  $\tau$  to 75 for the TRAP algorithm.

		Exec Time (s)			# patterns
		Mine	Clust	Total	
Anim1	TRAP	11	1042	1053	1708
	VS+C	0	7	7	116
Anim2	TRAP	9	1180	1189	1667
	VS+C	0	6	6	113
Drone	TRAP	28	952	980	1421
	VS+TRAP	9	722	731	1349
	VS+C	0	521	521	1095
Car1000	TRAP	109	231	340	575
	VS+TRAP	212	204	416	520
	VS+C	0	4560	4560	2923
Car2000	TRAP	153	1005	1158	1046
	VS+TRAP	153	954	1107	985
	VS+C	0	28524	28524	5196
Car3000	TRAP	153	1758	1911	1232
	VS+TRAP	153	1981	2134	1237
	VS+C	0	69866	69866	6543

Figure 4.13: Execution time and number of patterns output by the TRAP algorithm and the method which uses the video segmentation followed by a clustering step (VS+C)

designed to deal with long term occlusions. Figure 4.11 shows that, on the first 1000 frames, the longest cluster returned by TRAP is always the best one until the lowest levels of the hierarchy. This shows that the length criterion can be very good to find the best cluster when sufficient patterns representing the objects can be extracted and when no long disappearance of the targets splits the clusters. The VS+C method builds good clusters but only at the higher levels of the hierarchy, they are thus not found using our lifetime criterion.

In conclusion, our unsupervised methods give comparable (and most of the time better) results than the state-of-the-art trackers TDL and CT. However, we do not need to select the objects of interests in the first frame of the video which makes this method usable in practice to treat batches of off-line recorded videos such as Youtube ones.

**Efficiency** For the synthetic videos, when keeping the default parameters for  $\tau$  and  $\epsilon$  we fell into the number of patterns problem mentioned in Section 2.4.1. For both animations,  $\sigma_{st}$  was set to 150 but  $\sigma$  was set to 250 for the first animation and to 220 for the second one. As can be seen in the Figure 4.13 it takes more than 15 minutes to

process 1700 patterns in both cases. For the video segmentation (VS), we can not control the number of output patterns and for the simple animation video, we get only around 100 of them which made the clustering process very fast. Because of many changes in appearance for the real videos, there were less frequent patterns so we could keep the default setting for all parameters (except  $\sigma_{st}$  as discussed in Section 2.4.1). For TRAP, we used  $\sigma_{st} = 15$  for the drone and  $\sigma_{st} = 25$  for the car, and we set it to 35 for both videos when mining the more stable video segmentation. We mined the 5600 frames of the car video at once and then restricted the occurrence graph to the first 1000, 2000, and 3000 frames, this explains why the time results for the mining step of this video are constant. This is also why the number of patterns can be as low as 500 when processing only the first 1000 frames. As we can see, the VS+C approach produces a lot more patterns which greatly increased the computation time.

In conclusion, the mining phase can give better results and is more efficient than directly using the output of the dynamic segmentation for real videos. However, both methods are far from usable in real time although the clustering step could easily be improved by designing an optimized algorithm.

### 3 Conclusion

We have presented experiments which show the usefulness of the patterns mined by the algorithms presented in Chapter 2. In particular, the experiments we conducted on the spatio-temporal patterns show that the patterns are meaningful in a video tracking context: when the first occurrence of a spatio-temporal pattern matches an object, the rest of its occurrences tend to also match the same object with high precision.

We also described two strategies that use spatio-temporal patterns to track the main objects of videos. In a first strategy, we build a graph which is the concatenation of the occurrence graphs of all frequent patterns. We add edges to this graph connecting the occurrences of different patterns that are similar. Then, we look for paths called *spatio-temporal paths* in this global occurrence graph. With this method, the user needs to select a region of interest in the first frame of the video by drawing a bounding box. We then select the shortest spatio-temporal path (with respect to some weights on the edges) that starts from one of the occurrences contained in the selected area.

The second strategy exploits a similarity measure between the trajectories of the spatio-temporal patterns to group them into clusters representing the objects of the scene. The clusters are obtained with a hierarchical clustering algorithm. With this approach, we cut the hierarchy to obtain the best clusters (i.e., clusters that best match the main objects) automatically.

The spatio-temporal paths suffer from multiple limitations. They still require the user to select the target himself. They also tend to drift from the target, especially if the video is long. Indeed, it is sometimes less costly to move from the original selected occurrences to occurrences of a pattern that can be more easily followed. Therefore, sometimes, the computed shortest path starts by taking multiple similarity edges until it reaches the occurrence of a more stable pattern than the ones matching the target.

Our clustering approach solved the problem of the target selection. However, it is still unclear where exactly to cut the hierarchy to obtain the clusters and how many of them are matching an interesting object. Nonetheless, the highest ranked cluster often corresponds to the main object of the video.



## Chapter 5

# Conclusion

In this “Habilitation à Diriger des Recherches” thesis, I presented the research domains I have explored for the last 10 years, as a post-doc in the Machine Learning team in Leuven (Belgium) and as an associate professor in the Data Intelligence team at University Jean Monnet, Saint-Etienne. These domains include the design of an inductive database framework and the development of new exact constraint-based algorithms for this type of databases, as well as the development of new data mining algorithms to solve computer vision problems such as image classification and object tracking in videos.

As already discussed in the conclusion of Chapter 3, computer vision problems are complex tasks which usually necessitate important pre-processing, parameter tuning, and post-processing steps that would benefit from integrated systems such as inductive databases that could allow us to perform an entire knowledge discovery process. The usefulness of data mining techniques (in the broad sense including machine learning) to help computer vision has been proven a decade ago and this is confirmed every year by looking at the content of the top conferences in both the computer vision (*CVPR*, *ICCV* and *ECCV*) and the data mining (*NIPS*, *KDD*, *ICML*) domains. However, if the trend in domains such as pattern mining goes toward the elaboration of more and more generic (exact) algorithms triggered by more and more declarative query languages, it seems that when computer vision is concerned, the trend goes towards the opposite: very specific algorithms with ad hoc parameters that tend to be difficult to compare. Difficult then to integrate such algorithms to generic systems such as inductive databases... When trying to extract a meaningful representation for images, the main reasons for these countless algorithms lie in the fact that it is difficult to define what an exact solution to an image representation problem could be or which constraints one could add to his/her algorithms to obtain better classification results (apart from the class information that we used for the *GEMP* algorithm presented in Chapter 3). Furthermore, the number of data available in computer vision is so big than looking for and storing an exhaustive descriptive solution for such problems does not seem reasonable.

A generic alternative to these countless algorithms to extract better representations from images or videos might come from the recent use of deep learning architectures. These architectures and, in particular, convolution neural networks, learn automatically in their early layers a mid-level representation of the data suitable to achieve impressive classification results directly from the pixel information. The results shown since 2012 in the literature using those networks are far better than the results obtained by semi hand-crafted methods such as the ones presented in Chapter 3 (which rely on low level image descriptions). The main reasons why these “old” (in the history of machine

learning algorithms) networks are so effective nowadays come from the fact that we are able to gather a huge amount of labelled image data to learn the equally huge number of parameters of these networks and also from the recent advances of computer architectures and in particular the now common use of GPUs to speed up the learning process. Whatever the reasons, they have finally make everybody agree on a common solution to learn a suitable representation for images. Besides, the datasets used to learned those networks (such as Imagenet<sup>1</sup>) are so big and so diversified that the features learned by one network are easily (and successfully) transferable to many other problems. However, one still needs to train a network for days to get good performance and needs to train it almost from scratch (with a new architecture) when a new label needs to be added. Because of these drawbacks, this solution is nowadays still not suitable for an interactive process and thus for an inductive database system. Another drawback of this technique is that it does not take into account structural information in images or videos that could maybe further improve the results.

As far as using pattern mining for discovering better image representation (as shown in Chapter 3) is concerned, my conclusion is that one needs to work hand-in-hand with these deep architectures. For example, it would be possible to use neural architectures such as auto-encoders to automatically discover temporal patterns without unsupervised information. Or, one could imagine building patterns on the features extracted from the deep architectures. Another direction could be to mine directly the architectures to better understand the features that are extracted. Exploring this is one of the goals of the two ANR projects I am involved in: *Solstice*, mentioned in the introduction of this thesis and *Lives (Learning with Interacting ViEwS)*. Within the *Solstice* project, we hope to further explore the possibilities of deep learning architectures to solve new computer vision tasks such as, for example, outdoor scene labelling. This task would be particularly useful to devices such as self-driving cars or for visually-impaired persons wearing smart glasses. Within this project, we will also try to evaluate how, more structured patterns, could help building better representation of images. Our first attempt in this line is the development of a geometric graph mining algorithm called *GRIMA* which can mine interesting substructures in a grid of descriptors. For *GRIMA* to be competitive with the state-of-the-art, the description of the images could be made using the information provided by the descriptive layers of a convolutional neural network. Within the *Lives* project, we will work on integrating different views (for example depth maps in addition to colour images or, in this particular project, multiple brain images acquired with different medical imaging devices to build a computer-aided diagnosis tool for neurological disorders) to learn better representation of a problem e.g. using deep learning architectures.

In addition to these computer vision applications, I am currently starting new projects on different domains: a fraud detection, a bio-medical and two language processing applications. In the bio-medical domain, we are interested in predicting adverse reactions when transfusing blood samples according to proteins and other molecular markers contained in the samples. If simple prediction models such as decision trees can be used to give some interesting results [17, 5], by adding additional knowledge about the patient being transfused, because the data are not independent and identically distributed (i.i.d.) and because the classes to predict are highly unbalanced, we now need to design new types of algorithms that could cope with these specific data. This is also true for the bank fraud detection application. I will start working on this problem with

---

<sup>1</sup><http://www.image-net.org/>

the new PhD thesis of Guillaume Metzler funded by an industrial contract with the *Blitz* company. In this context, our goal is to model precisely what a fraud is in terms of basket analysis and user buying behavior. In addition, we want to model user profiles (e.g. by using information about loyalty cards) to reduce as much as possible the number of false alarms triggered by an automatic fraud detection system that would monitor the use of credit cards or cheques when paying in a supermarket. In this context, a single customer might make multiple (correlated) transactions along the time (the data are thus non i.i.d.) and the fraction of frauds compared to the total number of transactions in a supermarket is very low (the classes are again unbalanced).

In the context of language processing, I started projects in two different directions. In the first one, we want to explore how language models can be inferred by taking into account the precise context in which the sentences are pronounced [22]. In the second one, we work on building tools for automatic meeting management. In particular, we would like to answer questions such as: i) how can we build, manipulate and dynamically update a semantical representation of all information exchanged during a meeting? ii) how can we conceive new data mining algorithms able to learn models about the interactions between the meeting participants? iii) how can we take into account the contextual information of a meeting to facilitate the previous tasks? This last point is related to our first language processing problem. We would like to automatically learn the constraints of a CSP (*Constraint Satisfaction Problem*) using methods at the crossroad between inductive logic programming and pattern mining.

This document shows the diversity of the applications that I have tackled and of the researches (in data mining) that I have conducted. These researches have produced a good number of high quality publications. I have acquired a good experience in supervising PhD students (2 of them have already defended their thesis, 3 have started) but also master students (14 up to this day) during the past ten years. I am also involved in the research administration through project proposal writing, conference organizations, programme committee memberships, jury memberships, university council memberships and research management. And least but not last, I have spent a fair amount of time trying to create and teach interesting lectures to my students at all university levels. I am particularly involved in the management of the masters in computer science in Saint-Etienne.

## Other Publications Cited in the Document

- [42] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.
- [43] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.
- [44] LNF Ana and Anil K Jain. Robust data clustering. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–128. IEEE, 2003.
- [45] Michael R Anderberg. Cluster analysis for applications. Technical report, DTIC Document, 1973.
- [46] Nicos Angelopoulos and James Cussens. Exploiting informative priors for bayesian classification and regression trees. In Leslie Pack Kaelbling and Alessandro Saffioti, editors, *IJCAI-05, Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 641–646. Professional Book Center, 2005.
- [47] Jaume Baixeries, Laszlo Szathmary, Petko Valtchev, and Robert Godin. Yet a faster algorithm for building the Hasse diagram of a concept lattice. In Sébastien Ferré and Sebastian Rudolph, editors, *ICFCA'09, Proceedings of the 7th International Conference on Formal Concept Analysis*, Lecture Notes in Computer Science, pages 162–177. Springer, 2009.
- [48] G. Blanchard, C. Schäfer, Y. Rozenholc, and K. R. Müller. Optimal dyadic decision trees. *Machine Learning*, 66(2-3):209–241, 2007.
- [49] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.
- [50] Francesco Bonchi, Fosca Giannotti, Cludio Lucchese, Salvatore Orlando, Raffaele Perego, and Roberto Trasarti. A constraint-based querying system for exploratory pattern discovery information systems. *Information System*, 2008. Accepted for publication.
- [51] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.
- [52] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [53] Leo Breiman, Journal H. Friedman, R. A. Olshen, and C. Journal Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.
- [54] Björn Bringmann, Siegfried Nijssen, and Albrecht Zimmermann. Pattern-based classification: A unifying perspective. In Arno Knobbe and Johannes Fürnkranz, editors, *LeGo'09, Proceedings of the ECML PKDD 2009 Workshop 'From Local Patterns to Global Models'*, 2009.

- [55] W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.
- [56] Toon Calders, Bart Goethals, and Adriana Prado. Integrating pattern mining in relational databases. In *Proc. PKDD*, pages 454–461, 2006.
- [57] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.
- [58] Ruey-Feng Chang, Chii-Jen Chen, and Chen-Hao Liao. Region-based image retrieval using edgeflow segmentation and region adjacency graph. In *International Conference on Multimedia and Expo (ICME)*, volume 3, pages 1883–1886. IEEE, 2004.
- [59] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055, 1999.
- [60] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.
- [61] Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725, april 2007.
- [62] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–947, 1998.
- [63] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Image categorization using fisher kernels of non-iid image models. In *CVPR*, 2012.
- [64] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.
- [65] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Work. on Statistical Learning in CV*, pages 1–22, 2004.
- [66] G. Damiand. *Définition et étude d’un modèle topologique minimal de représentation d’images 2d et 3d*. Thèse de doctorat, Université Montpellier II, Décembre 2001.
- [67] Guillaume Damiand, Colin De La Higuera, Jean-Christophe Janodet, Émilie Samuel, and Christine Solnon. A polynomial algorithm for submap isomorphism. In *Graph-based Representation in Pattern Recognition (GBR)*, pages 102–112. Springer, 2009.
- [68] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society*, 39(1):1–38, 1977.
- [69] Fabien Diot. *Fouille de Graphes pour le Suivi d’Objets dans les Vidéos*. PhD thesis, Université Jean Monnet de Saint Etienne, 2014.

- [70] Saher Esmeir and Shaul Markovitch. Anytime induction of cost-sensitive trees. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS'07, Proceedings of the 21st Conference on Neural Information Processing Systems*, pages 425–432. MIT Press, 2007.
- [71] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>, 2007.
- [72] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. <http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf>.
- [73] J.D.R. Farquhar, Sandor Szedmak, Hongying Meng, and John Shawe-Taylor. Improving bag-of-keypoints image categorisation: Generative models and pdf-kernels. *Technical report, University of Southampton*, 2005.
- [74] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.
- [75] Arik Friedman, Assaf Schuster, and Ran Wolff.  $k$ -anonymous decision tree induction. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *PKDD'06, Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 151–162. Springer, 2006.
- [76] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.
- [77] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, and Murali Venkatrao. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. *Data Mining and Knowledge Discovery*, pages 152–159, 1996.
- [78] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2141–2148. IEEE, 2010.
- [79] Jiawei Han, Yongjian Fu, Wei Wang, Krzysztof Koperski, and Osmar Zaiane. DMQL: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.
- [80] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, September 2000.
- [81] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, *SIGMOD'00, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12. ACM Press, 2000.

- [82] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.
- [83] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39:58–64, 1996.
- [84] Tomasz Imielinski and Aashu Virmani. Msql: A query language for database mining. *Data Mining Knowledge Discovery*, 3(4):373–408, 1999.
- [85] Yunjae Jung, Haesun Park, Ding-Zhu Du, and Barry L Drake. A decision criterion for the optimal number of clusters in hierarchical clustering. *Journal of Global Optimization*, 25(1):91–111, 2003.
- [86] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 604–610, Washington, DC, USA, 2005. IEEE Computer Society.
- [87] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 49–56. IEEE, 2010.
- [88] Arno Knobbe, Bruno Crémilleux, Johannes Fürnkranz, and Martin Scholz. From local patterns to global models: the LeGo approach to data mining. In Johannes Fürnkranz and Arno Knobbe, editors, *LeGo'08, Proceedings of the ECML PKDD 2008 Workshop 'From Local Patterns to Global Models'*, pages 1–16, 2008.
- [89] R. Fergus L. Fei-Fei and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision.*, 2004.
- [90] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.
- [91] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.
- [92] Xiao-Chen Lian, Zhiwei Li, Changhu Wang, Bao-Liang Lu, and Lei Zhang. Probabilistic models for supervised dictionary learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pages 2305–2312, 2010.
- [93] Haibin Ling and S. Soatto. Proximity distribution kernels for geometric context in category recognition. In *ICCV*, 2007.
- [94] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86. AAAI Press, 1998.

- [95] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV 1999)*, volume 2, pages 1150–1157 vol.2. IEEE Computer Society, 1999.
- [96] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.
- [97] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkatasubramanian.  $l$ -diversity: Privacy beyond  $k$ -anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3, 2007.
- [98] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.
- [99] Rosa Meo, Giuseppe Psaila, and Stefano Ceri. An extension to sql for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195–224, 1998.
- [100] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision*, 60(1):63–86, 2004.
- [101] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Computer Vision*, 60:63–86, 2004.
- [102] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [103] Siegfried Nijssen and Joost N. Kok. Multi-class correlated pattern mining. In *Revised selected papers of the workshop KDID'05*, pages 165–187, 2006.
- [104] Siegfried Nijssen and Luc De Raedt. Iql: A proposal for an inductive query language. In *Revised selected papers of the workshop KDID'06*, pages 189–207, 2007.
- [105] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729, dec. 2008.
- [106] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak hypotheses and boosting for generic object detection and recognition. In *ECCV*, pages 71–84, 2004.
- [107] D. Ormoneit and V. Tresp. Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks*, 9(4):639–650, 1998.
- [108] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.
- [109] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, pages 143–156, 2010.
- [110] Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [111] Konstantinos Rematas, Mario Fritz, and Tinne Tuytelaars. The pooled nbnn kernel: Beyond image-to-class and image-to-image. In *ACCV*, volume 7724, pages 176–189, 2012.



- [112] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.
- [113] E. Samuel. *Recherche de motifs dans des images : apport des graphes plans*. PhD thesis, Université Jean Monnet de Saint Etienne, 2011.
- [114] F. Shahbaz Khan, J. van de Weijer, and M. Vanrell. Top-down color attention for object recognition. In *ICCV*, pages 979–986, 2009.
- [115] Zhao H. Tang and Jamie MacLennan. *Data Mining with SQL Server 2005*. John Wiley & Sons, 2005.
- [116] P. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.
- [117] Tinne Tuytelaars, Mario Fritz, Kate Saenko, and Trevor Darrell. The nbnn kernel. In *ICCV*, pages 1824–1831, 2011.
- [118] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In Roberto Bayardo, Bart Goethals, and Mohammed J. Zaki, editors, *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, CEUR Workshop Proceedings. CEUR-WS.org, 2004.
- [119] A. Vailaya, M.A.T. Figueiredo, A.K. Jain, and H.J. Zhang. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1):117–130, 2001.
- [120] Joost van de Weijer and Cordelia Schmid. Applying color names to image description. In *ICIP*, pages 493–496, 2007.
- [121] Jan C. van Gemert, Cor J. Veenman, Arnold W.M. Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1271–1283, 2010.
- [122] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 281, Washington, DC, USA, 2003. IEEE Computer Society.
- [123] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110, 2000.
- [124] H. Wang and C. Zaniolo. Nonmonotonic reasoning in ldl++. *Logic-based artificial intelligence*, pages 523–544, 2001.
- [125] Haixun Wang and Carlo Zaniolo. Atlas: A native extension of sql for data mining. In *Proc SDM*, pages 130–144, 2003.
- [126] Jö Wicker, Lothar Richter, Kristina Kessler, and Stefan Kramer. Sinbad and sql: An inductive database and query language in the relational model. In *Proc ECML PKDD*, pages 690–694, 2008.

- [127] Nianhua Xie, Haibin Ling, Weiming Hu, and Xiaoqin Zhang. Use bin-ratio information for category and scene classification. In *CVPR*, pages 2313–2319, june 2010.
- [128] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In *ACM SIGKDD*, 2005.
- [129] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *International Conference on Data Mining (ICDM)*, pages 721–724. IEEE, 2002.
- [130] Yi Yang and Shawn Newsam. Spatial pyramid co-occurrence for image classification. In *ICCV*, 2011.
- [131] Tsuhan Chen Yimeng Zhang. Efficient kernels for identifying unbounded-order spatial features. In *CVPR*, 2009.
- [132] Junsong Yuan, Ming Yang, and Ying Wu. Mining discriminative co-occurrence patterns for visual recognition. In *CVPR*, pages 2777–2784, june 2011.
- [133] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In David Heckerman, Heikki Mannila, and Daryl Pregibon, editors, *KDD'97, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.
- [134] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *European Conference on Computer Vision (ECCV)*, pages 864–877. Springer, 2012.