UNIVERSITÉ DE LIÈGE
Faculté des Sciences Appliquées
Département d'Électricité, Électronique et Informatique
Institut Montefiore

Classification automatique d'images par arbres de décision

Thèse présentée par
Raphaël Marée
Docteur en Sciences
(orientation Informatique)

Année Académique 2004–2005

Résumé

Cette thèse porte sur la classification automatique d'images. À partir d'un échantillon d'images étiquetées par un être humain, l'objectif est de mettre au point de manière automatique un modèle qui permet de classer de nouvelles images dans l'une des classes établies au préalable.

L'intérêt principal de cette thèse est une nouvelle approche automatique de classification d'images qui s'est avérée précise sur une large gamme de problèmes. Elle repose sur des méthodes récentes d'apprentissage automatique supervisé. En particulier, nous avons proposé l'utilisation d'une méthode d'ensemble d'arbres de décision que nous avons appliquée directement aux valeurs des pixels des images comme variables descriptives. Elle a été combinée avec des techniques d'extraction et de transformation aléatoires de fenêtres au sein des images ce qui permet une meilleure robustesse à différentes conditions d'acquisition.

La méthode a été évaluée sur 7 bases de données d'images relatives à la classification de chiffres manuscrits, de visages, d'objets 3D, de textures, d'immeubles, de thèmes ou paysages. Certains de ces problèmes impliquent des perturbations telles que des changements de point de vue, d'échelle, d'orientation, d'illumination ; ou ils représentent des scènes réelles qui entraînent une visibilité partielle des objets ou encore la présence d'encombrement. En termes de précision, notre méthode est comparable aux meilleurs résultats de la littérature et ses temps de calcul sont très avantageux.

Abstract

The work presented in this thesis is motivated by the problem of automatic image classification. Image classification methods seek to automatically classify previously unseen images using databases of labeled images provided by human experts.

The main contribution of this thesis is a novel approach for image classification that has been shown to perform well on a variety of tasks. It uses some recent machine learning algorithms based on ensembles of decision trees that we applied directly on pixel values. We combine it with techniques of random extraction and transformation of subwindows from images so as to improve robustness to certain image transformations.

The method has been evaluated on 7 publicly available datasets corresponding to various image classification tasks : recognition of handwritten digits, faces, 3D objects, textures, buildings, themes, or landscapes. Some of these datasets contain images representing widely varying conditions : occlusions, cluttered background, illumination, viewpoint, orientation, and scale changes. The accuracy of our method is generally comparable with the state of the art and it is particularly attractive in terms of computational efficiency.

Remerciements

Je remercie en premier lieu Louis Wehenkel, promoteur de cette thèse, pour m'avoir orienté vers ce sujet de recherche, pour sa disponibilité tout au long de la réalisation de celle-ci et pour ses précieux conseils. Qu'il soit également sincèrement remercié pour sa patience, sa confiance, et pour les échanges d'égal à égal que nous avons eus. Le respect et la gentillesse dont il a témoigné à mon égard contribueront beaucoup aux souvenirs agréables que je garderai de cette expérience.

Je souhaite remercier Justus Piater, arrivé à l'Institut Montefiore il y a deux ans, et qui est devenu co-promoteur de cette thèse. Outre ses connaissances scientifiques, son enthousiasme et ses encouragements m'ont fait beaucoup de bien.

Je remercie tout particulièrement Pierre Geurts dont les recherches, les développements et les conseils ont rendû possible mes travaux. Je souhaite longue vie à ses arbres aléatoires.

J'adresse mes remerciements aux membres du jury de cette thèse qui ont accepté de l'évaluer.

Je remercie les personnes qui ont bien voulu consacrer du temps à la lecture complète ou partielle de ce document : Damien Ernst, Gilles Renard, mon frère Jean-François.

Parmi les membres anciens ou actuels de l'Institut Montefiore, je tiens à remercier, pour leur aide ou les moments passés ensembles : Claude Diepart, Ibtissam El Khayat, Jean-Marc François, Cédric Gauthy, Benjamin Jennes, Marie-Berthe Lecomte, Axel Mahy, Philippe Mack.

Merci à la Région Wallonne pour avoir financé ce travail dans le cadre du programme "FIRST Doctorat", et au Patrimoine de l'Université pour le subside relatif à ma participation à la conférence ACCV 2004 en Corée du Sud. Merci à André Renard et à l'équipe du GIGA pour la confiance qu'ils m'ont déjà accordée.

Pour de nombreuses très bonnes raisons, notamment leur présence, leur influence positive sur moi, leurs idées, leur soutien, leurs attentions et leur complicité, à diverses périodes, je remercie chaleureusement Antony, Damien, Ewo, Hugo, Laetitia, Sarah, les membres de l'ASBL Panoptic, et mes parents.

Enfin, je voudrais remercier, pour la motivation qu'elle ne manquera pas de susciter auprès de tout chercheur en vision par ordinateur, cette société qui publiait en 2004 une offre d'emploi pour l'interprétation et la classification manuelle de photographies aériennes et d'images satellites, aux conditions suivantes : *"Etre prêt à travailler en horaire décalé. Accepter de faire un travail répétitif. Ne pas être daltonien. Etre résistant au stress."*.

Table des matières

Chapitre 1

Introduction

La recherche présentée dans cette thèse concerne une nouvelle méthode de classification d'images. Elle utilise le paradigme de classification par apprentissage automatique supervisé. Dans ce chapitre, le problème de la classification d'images et le domaine de l'apprentissage automatique sont introduits au sein de la section 1.1. Les objectifs principaux de notre recherche dans ce contexte sont présentés dans la section 1.2 ainsi que l'approche que nous avons utilisée pour les atteindre. Nos principales contributions sont énumérées dans la section 1.3. Enfin, l'organisation de ce document est donnée en section 1.4.

1.1 Contexte et motivations

1.1.1 Classification d'images

Principes

La classification d'images consiste à répartir systématiquement des images selon des classes établies au préalable. Classer une image lui fait correspondre une classe, marquant ainsi sa parenté avec d'autres images.

En général, reconnaître une image est une tâche aisée pour un humain. Au fil de son existence, il a acquis des connaissances qui lui permettent de s'adapter aux variations qui résultent de conditions différentes d'acquisition. Il lui est par exemple relativement simple de reconnaître un objet dans plusieurs orientations, partiellement caché par un autre, de près ou de loin, et selon diverses illuminations. Toutefois, les progrès technologiques en termes d'acquisition d'images (microscopes, caméras, capteurs) et de stockage engendrent des bases de données riches en information et multiplient les domaines d'applications. Il devient alors difficile pour l'humain d'analyser ce nombre important d'images. Le temps requis, le caractère répétitif

de la tâche et la concentration nécessaire sont problématiques. Aussi, la classification automatique d'images est souhaitable. Toutefois, celle-ci n'est pas forcément aisée pour un programme informatique pour lequel une image est un ensemble de valeurs numériques. De plus, l'appartenance d'une image à une classe, autrement dit la similarité entre images d'une même classe, et la dissimilarité entre images de classes différentes, peuvent dépendre de nombreux critères : le domaine d'application, l'observateur, sa connaissance du domaine qui influence sa perception et son niveau d'abstraction, des lois physiques, des règles géométriques ou topologiques, etc. D'après ces critères et selon les conditions d'acquisition, l'apparence des images d'une même classe peut donc présenter une variabilité plus ou moins grande.

Nous considérons dans cette thèse le problème suivant : étant donné un échantillon d'images étiquetées, comment mettre au point de manière automatique un programme informatique qui permet de classer de nouvelles images, avec la plus grande précision possible, dans l'une des classes établies au préalable ? Si l'échantillon d'images est représentatif des différentes classes, il est raisonnable de penser qu'une généralisation au-delà de cet échantillon est possible. Le classificateur informatique construit à partir de ces images devrait donc permettre l'identification d'images inconnues. La précision de la classification et le temps requis pour la réaliser constituent les principaux critères de performance d'un tel système. Ils sont habituellement évalués sur un échantillon indépendant d'images.

Applications

La tâche de classification d'images apparaît dans de nombreux domaines. En pratique, une classe identifie une image représentant un objet particulier, un individu, un logo, une cellule vivante d'une certaine famille, un défaut de fabrication d'un type donné, etc. Les figures 1.1, 1.2 et 1.3 illustrent quelques exemples d'applications de classification d'images biologiques récemment abordées dans la littérature.[1] Nous énumérons quelques applications réelles :

- la médecine : classification de cellules [KDN01], de mammographies [AZC01], de radiographies [PKL+02],
- la biologie : reconnaissance d'espèces animales [RG04] [TG03], classification de pollen [RDCFG03],
- l'agriculture : classification de types de grains [GGVC03],
- la bureautique : reconnaissance de caractères d'alphabets multiples [LBBH98] [AHE03] [DKS03], de codes postaux, de pictogrammes [MENW99],
- la géologie : reconnaissance de types de sols ou de textures [MPV02], de volcans [BAS+98],
- l'astronomie : classification de galaxies [FWD93] [KCPFT02],

[1] Ces bases de données d'images ne sont pas disponibles publiquement.

- les processus industriels : reconnaissance de défauts de fabrication pour le contrôle qualité [PA03],
- la protection civile : détection d'anciennes zones de bombardement [MMFB04],
- l'environnement : reconnaissance de polluants [KHM98],
- la mobilité ou le tourisme : reconnaissance de panneaux de signalisation [LGF01], la localisation par la reconnaissance de bâtiments [SSV03],
- l'art : classification de peintures [LW04],
- la biométrie : reconnaissance de visages, d'empreintes, d'iris [RSQZ04],
- etc.

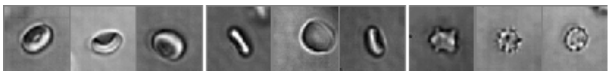

FIG. 1.1 – La classification de cellules vivantes [KDN01] selon l'une des trois classes : stomatocyte, discocyte, échinocyte.

FIG. 1.2 – La classification et la recherche d'images de radiographies [PKL+02] parmi 6 classes : abdomen, membre, sein, crâne, poitrine, colonne vertébrale.

FIG. 1.3 – L'identification de salamandres [RG04].

Méthodes existantes

Les systèmes informatiques de classification automatique d'images sont habituellement spécifiques au problème considéré. Après une phase d'acquisition des images, ces systèmes y vérifient la présence de caractéristiques. Celles-ci décrivent en général des propriétés colorimétriques et/ou géométriques des images ou des régions de celles-ci. Du choix de ces attributs dépendra en partie le succès avec lequel les images

sont classées. En effet, pour comparer et classer celles-ci, seule l'information correspondant à ces attributs sera utilisée. Cette approche présente plusieurs limitations. D'une part, il n'existe pas un ensemble de caractéristiques universelles autre que les valeurs de pixels qui pourront a priori convenir pour un problème donné. Cela implique souvent de devoir adapter manuellement cette phase de pré-traitement lorsqu'une nouvelle application voire un nouveau sous-problème se présente. L'automatisation du procédé est donc restreinte. En outre, il est parfois difficile pour un humain de caractériser des images par des attributs qui mèneront à leur reconnaissance. Autrement dit, pour un expert d'un domaine qui est capable d'identifier les images qu'il étudie, il n'est pas toujours évident de traduire les caractéristiques visuelles de haut niveau qu'il perçoit dans ces images en contraintes informatiques de bas niveau ([BAS+98] mentionne cette difficulté pour la reconnaissance de volcans sur Vénus). Pour certains problèmes, cette complexité à décrire les images par un nombre limité d'attributs peut provenir de la variabilité importante de ces images au sein d'une même classe. La minimisation du décalage entre l'encodage numérique et la sémantique des images est une difficulté de la tâche de caractérisation. De plus, il n'est pas évident d'établir un ensemble de caractéristiques qui seront correctement détectées quelles que soient les conditions d'acquisition des images. D'autre part, cette étape de pré-traitement peut causer la perte d'information potentiellement utile à la reconnaissance. Cette perte résulte des filtrages et de la sélection manuelle d'un sous-ensemble de caractéristiques symboliques par rapport à l'ensemble des données qui décrivent les images.

Il y a une quinzaine d'années sont apparues des méthodes de classification d'images qui ne reposent sur aucune représentation symbolique. Pour classer une nouvelle image, elles la comparent principalement à un échantillon de base (compressé ou non) d'images qui représentent les différentes classes sous différentes vues. En effet, [EB90] et [PE90] ont montré qu'il était possible de reconnaître des objets en 3 dimensions à partir d'une collection d'images en 2 dimensions. Ils ont donc montré que la structure 3D d'un objet pouvait être estimée si un nombre suffisant de vues en 2D de celui-ci étaient disponibles. Cette approche dispense ainsi de conserver un modèle en 3D de l'objet ou d'extraire des primitives à partir des images 2D réelles. [TP91] ont appliqué le même principe pour la représentation et la reconnaissance de visages, en utilisant un ensemble de vues 2D caractéristiques. Leur méthode projette les images dans un espace à dimension réduite par la transformation de Karhunen-Loeve (ou analyse à composantes principales, PCA), comme l'avait proposé au préalable [SK87] pour la caractérisation de visages. Il s'agit, autrement dit, d'établir une collection de modèles de différentes apparences en 2D des visages. La classification d'une nouvelle image se réalise en la décomposant également en composantes principales et en comparant dans cet espace sa position par rapport aux positions des images de l'échantillon de base. La distance entre les points de cet espace est ainsi vue comme une mesure de similarité des images et la classe de l'image la plus proche est assignée à la nouvelle image. [MN95] a appliqué ce type de projection à la reconnaissance d'une vingtaine d'objets quelconques. Ces études ont montré qu'il était possible de

réaliser la classification d'images dans des domaines divers selon une approche de base similaire : l'utilisation d'images représentatives et leur comparaison avec une nouvelle image pour classer celle-ci. On parle d'approche de classification fondée sur l'apparence globale.

Plus récemment, cette approche a été enrichie par des techniques qui améliorent la reconnaissance en situations réelles. De nouveaux procédés extraient à partir des images des points ou régions et cela de manière stable malgré un ensemble de transformations qui peuvent perturber les nouvelles images [MTS+04]. La description de ces points ou régions [MS03] est généralement conçue pour être invariante à ces perturbations par l'utilisation de propriétés mathématiques calculées sur une partie de la région considérée. Selon ce schéma, la classification d'une nouvelle image consiste à comparer les descriptions des points ou régions locales extraites en son sein à celles des images disponibles. La combinaison des résultats de ces comparaisons locales permet de classer l'image. Ces méthodes se comportent de manière satisfaisante sur des problèmes courants de classification. Mais comme c'était le cas avec l'utilisation d'un ensemble réduit de caractéristiques calculées sur les images globales, ces nouveaux procédés de détection et de description peuvent entraîner une perte d'information désavantageuse pour la reconnaissance. Certains types d'images comportent par exemple des régions homogènes ou des zones texturées qui peuvent être moins bien prises en compte par ce genre de techniques de détection.

1.1.2 Apprentissage automatique supervisé

Principes

L'apprentissage automatique supervisé regroupe un ensemble de méthodes d'analyse de données. L'un de ses objectifs est l'extraction automatique d'un modèle à partir d'un échantillon d'observations décrites par des variables. Dans ce contexte, on part généralement d'une base de données sous la forme d'un tableau dont les lignes sont des objets (un patient, un client, un état d'un système, etc.) et les colonnes des variables qui les décrivent (une mesure médicale, le montant d'une transaction, une dimension électrique, etc.). Le modèle construit automatiquement vise à expliquer le comportement observé. C'est-à-dire qu'il met en évidence ou synthétise les relations existantes entre les variables d'entrée qui décrivent les observations et une variable de sortie. On parle d'apprentissage automatique supervisé dans la mesure où la variable de sortie est donnée à la méthode. En classification, la variable de sortie est une classe et le modèle peut être utilisé pour classer des situations nouvelles.

Les fondements de l'apprentissage automatique sont les mathématiques, en particulier les statistiques et la théorie des probabilités, ainsi que la biologie, plus spécifiquement les neurosciences. Ses domaines d'applications sont très nombreux puisque de telles méthodes peuvent s'appliquer dès lors que l'on possède des bases de don-

nées. L'apprentissage automatique supervisé peut aider au diagnostic d'un nouveau patient sur base de mesures médicales de patients antérieurs et de leur état de santé. Il permet l'analyse du processus de fabrication d'un produit en exploitant l'historique des capteurs de sa chaîne de production, etc.

Aux cours des dernières décennies, parallèlement aux avancées dans le domaine de l'acquisition de données et aux progrès matériels (en termes de capacité de stockage et de puissance de calcul), de nouveaux développements dans le domaine de l'apprentissage automatique ont mené à des méthodes qui élargissent son champ d'application. Parmi celles-ci, citons les machines à support vectoriel et celles basées sur des ensembles d'arbres de décision. Elles permettent le traitement de problèmes de très grandes dimensions, en particulier dans les cas où les observations sont décrites par un très grand nombre de variables d'entrée. Grâce à l'utilisation de ces approches, l'apprentissage automatique trouve désormais son utilité dans des domaines aussi variés que la bioinformatique (traitement de séquences génétiques ou de spectres de masse [GFdS+04]), les réseaux électriques [Weh97], les procédés industriels, les réseaux informatiques [GKL04], l'économie et la finance, etc.

Application à la classification d'images

Le principe de l'apprentissage automatique supervisé peut être appliqué directement à la classification d'images [HVP02]. En effet, le problème de classification d'images peut être considéré comme un problème d'apprentissage supervisé où les exemples d'apprentissage sont les images disponibles étiquetées. Constituer un tel échantillon est réalisable par un expert du domaine. Son expérience est ainsi exploitée implicitement. L'idée est donc d'utiliser des images caractéristiques pour la représentation d'une classe. Ainsi, pour un problème donné, un modèle de classification (c'est-à-dire une règle de décision, une fonction) peut être construit de manière automatique à partir de cette collection d'images étiquetées. Il aura pour but de distinguer le mieux possible les différentes classes d'images à reconnaître. Appliqué à une nouvelle image, il prédit sa classe en fonction de l'information qu'elle contient. Comme pour tout autre problème de classification, il s'agit de modéliser les relations qui existent au sein d'une base de données d'apprentissage entre les variables d'entrée (l'information contenue dans les images) et la variable de sortie (la classe) et d'appliquer le modèle induit pour la classificaton de nouvelles instances.

Comme nous l'avons vu, la technique courante est de construire manuellement un ensemble de descripteurs pour chaque problème et d'utiliser cet ensemble d'informations en entrée d'une méthode d'apprentissage, généralement la méthode des plus proches voisins. Du point de vue de l'apprentissage, ce prétraitement spécifique permet de réduire la complexité du problème (sa haute dimensionnalité) qui résulterait de l'utilisation des pixels des images pour décrire celles-ci. En effet, dans un tel contexte, la plupart des méthodes d'apprentissage n'ont pas une précision satisfaisante et présentent des temps de calcul pénalisants. L'utilisation de méthodes

automatiques de réduction de dimension (dont l'analyse en composantes principales mentionnée précédemment) poursuit souvent le même objectif : la réduction de l'espace d'entrée pour en permettre le traitement par une méthode d'apprentissage automatique.

Ainsi, pour la classification d'images, l'utilisation de l'apprentissage automatique est souvent encore restreinte à la méthode des plus proches voisins ou à des techniques qui ne peuvent travailler efficacement que sur des bases de données dont les objets sont décrits par un nombre relativement réduit de variables. Toutefois, les récentes avancées en apprentissage automatique ont fait apparaître des méthodes capables de traiter des problèmes de plus en plus complexes sans utiliser aucune information a priori sur le domaine d'application. Elles rivalisent souvent avec les méthodes propres à ces domaines qui, elles, résultent pourtant d'une adaptation importante au problème d'application. En pratique, les problèmes de classification d'images étant de plus en plus nombreux et variés, l'utilisation de ces nouvelles méthodes peut s'avérer intéressante.

1.2 Objectifs

La recherche présentée dans cette thèse s'inscrit dans le contexte du problème de plus en plus vaste de la classification d'images.

Notre principal objectif est de proposer une méthode qui soit directement applicable aux multiples problèmes de classification d'images. Elle doit donc être générique dans le sens où elle doit pouvoir être appliquée directement et avec précision à des problèmes de classification d'images les plus divers.

Dans ce but, notre approche reposera sur des récents progrès en apprentissage automatique qui ont engendré des méthodes (telles que les machines à support vectoriel et les ensembles d'arbres de décision) qui peuvent être appliquées directement à une large gamme de problèmes sans connaissance a priori. Nous proposons donc d'évaluer leurs performances pour la famille de problèmes qui nous concerne : la classification d'images. Puisque ces méthodes récentes d'apprentissage automatique sont capables de traiter des problèmes de très grandes dimensions, nous allons étudier leurs performances pour la classification d'images sans réduction de dimension explicite (ni par pré-traitement spécifique, ni par méthode de décomposition). C'est-à-dire que nous proposons d'appliquer ces méthodes directement à partir de toutes les valeurs des pixels des images. A priori, ce choix présente plusieurs avantages. Par nature, cette représentation est premièrement générique : elle peut s'appliquer à tous types d'images et ne dépend pas de la détection de primitives. Cette approche ne fait aucune hypothèse implicite ou explicite sur les images et l'environnement et peut être appliquée directement aux nombreux problèmes de classification d'images. En outre, elle est potentiellement plus discriminante que le sous-ensemble de carac-

téristiques obtenues par pré-traitement spécifique car aucune information des images réelles n'est écartée a priori. En effet, les phases intermédiaires entre l'acquisition des images et leur apprentissage engendre potentiellement une perte d'information. Notre hypothèse est qu'un algorithme d'apprentissage devrait être capable de sélectionner automatiquement et efficacement l'information discriminante. Autrement dit, cela suppose qu'une méthode automatique peut remplacer le procédé manuel d'extraction de caractéristiques, ou une procédure élaborée de détection et de description de zones. Selon cette idée, la tâche parfois ardue de caractérisation des images à reconnaître n'est donc plus nécessaire : seules les images et leurs étiquettes seront utilisées. Nous voulons donc proposer une méthode automatique, générique et précise de classification d'images. Nous voulons démontrer empiriquement la pertinence de cette démarche et son champ possible d'application. Nous l'évaluerons sur des problèmes variés et de difficulté croissante.

1.3 Contributions

La principale contribution de cette thèse est une nouvelle méthode de classification d'images. Elle est automatique, souple et elle s'est avérée précise pour une large gamme de problèmes. Nous présentons ci-dessous progressivement nos contributions.

L'étude empirique des performances de 7 méthodes d'apprentissage automatique qui travaillent à partir des pixels, sur 4 problèmes typiques de classification d'images, montre tout d'abord l'intérêt de cette approche. L'utilisation de ces techniques directement à partir des pixels ne requiert pas d'intervention humaine importante au-delà de la phase préalable d'étiquetage des images commune à tous les systèmes de classification supervisée. En outre, l'approche est générique, car elle peut être appliquée sans adaptation à des problèmes divers. Aucune étude comparative n'avait évalué un tel ensemble de méthodes sur des problèmes variés de classification d'images, bien que l'approche ne soit pas nouvelle. Notre première évaluation systématique a été publiée dans [MGV+03]. Les résultats obtenus sont satisfaisants mais nous montrerons les limites de cette approche.

Suite à cette étude, nous avons étudié la combinaison d'une méthode d'apprentissage basée sur les ensembles d'arbres de décision aléatoires avec une technique simple d'extraction de fenêtres locales. Les fenêtres sont extraites en grand nombre à des positions aléatoires dans les images. L'information des pixels de ces fenêtres est utilisée par la méthode d'apprentissage. La combinaison des classifications des fenêtres permet la classification des images entières. Cette méthode est également automatique et générique comme sa variante globale dont elle améliore les performances en termes de précision. Ses temps de calcul sont acceptables et ses performances sont comparables aux meilleures méthodes de la littérature sur ces problèmes. Enfin, elle est souple car l'ajustement de quelques paramètres permet d'atteindre un compromis précision/rapidité qui peut s'avérer utile en pratique. La première version de

cette méthode a été présentée dans [MGW03].

Ensuite, nous avons étudié empiriquement la robustesse de cette approche par rapport aux changements de point de vue, d'orientation et d'échelle et sa tolérance aux occultations. Cette étude montre que l'approche locale est robuste à de faibles transformations. Une évaluation préliminaire de la robustesse a été publiée dans [MGPW04]. Une procédure simple de génération automatique de fenêtres transformées est ensuite proposée. Elle améliore sa robustesse par apprentissage.

Enfin, l'évaluation de l'approche sur trois problèmes récents montre que son utilisation pour des problèmes complexes est réaliste. Ces problèmes présentent des images où les variations sont importantes et où des perturbations et transformations sont naturellement présentes (phénomènes d'encombrement et d'occultations ainsi que des changements d'illumination, d'échelle, de point de vue, etc.).

1.4 Plan du document

Ce document est organisé en 7 chapitres. Après une présentation des notions d'apprentissage automatique, nous appliquons l'approche à des problèmes de classification de plus en plus complexes. L'approche est complétée au fur et à mesure par des techniques qui étendent son champ d'application.

Le chapitre 2 introduit l'apprentissage automatique et présente les méthodes de classification supervisée que nous utiliserons par la suite. Ainsi, nous présentons les méthodes d'arbre et d'ensembles d'arbres de décision (Bagging, Boosting, Random Forests, Extra-Trees), la méthode des plus proches voisins, et la méthode de machines à support vectoriel.

Au sein du chapitre 3, nous décrivons l'approche générique de classification qui repose sur l'application des méthodes d'apprentissage automatique à partir de l'ensemble des pixels des images. Quelques approches similaires sont présentées. Nous évaluons 7 méthodes d'apprentissage sur 4 problèmes de classification d'images. Nous considérons des problèmes typiques de la littérature : la classification de chiffres manuscrits, de visages, d'objets en 3D et de textures. Les figures 1.4, 1.5, 1.6, 1.7 montrent quelques images des bases de données en libre accès que nous avons utilisées. Notre protocole d'expérimentation est précis et constitue un cadre d'évaluation relativement large. Les critères retenus sont la précision et les temps de calculs. Nous retiendrons de cette première évaluation une méthode d'apprentissage parmi celles évaluées : les ensembles d'arbres de décisions aléatoires. Cette méthode sera combinée au sein des chapitres suivants à des techniques simples et génériques au fur et à mesure de nos évaluations sur des problèmes plus complexes.

Le chapitre 4 présente la combinaison de la méthode retenue au chapitre précédent avec une technique simple et générique d'extraction de fenêtres locales au sein

FIG. 1.4 – Quelques images de la base de données de chiffres manuscrits MNIST. Les 10 classes correspondent aux 10 chiffres.

FIG. 1.5 – Quelques images de la base de données de visages ORL comportant des images de 40 individus à identifier.

FIG. 1.6 – Quelques images de la base de données d'objets 3D COIL-100 qui propose 100 classes correspondant à 100 objets quelconques photographiés selon différents angles de vue.

FIG. 1.7 – Quelques images de la base de données OUTEX incluant 54 classes de textures.

des images. En début de chapitre, un parallèle est fait avec l'état de l'art. Notre méthode suit les étapes qui structurent généralement les méthodes actuelles de la littérature mais se distingue de la plupart de ces méthodes par au moins une des étapes. Les performances de cette nouvelle méthode sont étudiées selon ses paramètres. Cette approche améliore sensiblement tous nos résultats par rapport à la version globale sans extraction de fenêtres.

Le chapitre 5 étudie systématiquement la robustesse des deux approches en présence de changements de point de vue, d'orientation et d'échelle. Leur tolérance aux occultations est également évaluée. Les phénomènes d'encombrement et d'illumination sont abordés. Cette étude montre que la variante locale est plus robuste que sa version de base. Après une présentation des méthodes robustes de l'état de

l'art, nous proposons une procédure simple de génération automatique de fenêtres transformées qui améliorera encore sa robustesse.

Le chapitre 6 propose d'étudier les performances de l'approche sur des problèmes considérés comme encore plus complexes dans la littérature récente. Leur difficulté provient des plus grandes variations qui existent parmi les images d'une même classe dues aux conditions d'acquisition et aux différents niveaux d'abstraction. Nous appliquerons notre approche sur 3 bases de données récentes : la classification de catégories d'objets (voir la figure 1.8), la classification d'immeubles en situations réelles (voir la figure 1.9) et la classification de thèmes ou paysages (voir la figure 1.10). Nos résultats sont comparés avec l'état de l'art et confirment que l'utilisation de notre approche est réaliste pour ces problèmes également.

Le chapitre de conclusions résume nos résultats principaux. Il propose également des extensions possibles de l'approche en termes de méthodes et d'applications.

FIG. 1.8 – Quelques images de la base de données Cogvis ETH-80 constituées d'images réparties en 8 classes : pommes, poires, tomates, vaches, chiens, chevaux, tasses, voitures.

FIG. 1.9 – Quelques images de la base de données ZUBUD proposant des images, prises dans diverses conditions, pour la reconnaissance de 205 immeubles.

FIG. 1.10 – Quelques images de la base de données WANG avec une image pour chacune des 10 classes : villages et habitants d'Afrique, plage, bâtiments, bus, dinosaures, éléphants, fleurs, chevaux, montagnes et glaciers, nourriture.

Chapitre 2

Notions d'apprentissage automatique supervisé

Dans ce chapitre, nous présentons les notions d'apprentissage automatique supervisé que nous utiliserons au cours des chapitres suivants. En particulier, nous présentons les 7 méthodes de classification qui seront par la suite appliquées aux problèmes de classification d'images.

2.1 Principes

Avec les nouvelles technologies d'acquisition de données et les progrès matériels en termes de capacités de stockage et de puissance de calcul, de larges volumes de données brutes sont disponibles dans de nombreux domaines d'applications. Il est de plus en plus souhaitable d'en extraire automatiquement une information synthétique et utile. C'est l'objet de l'apprentissage automatique qui s'inscrit dans le processus plus large d'extraction de connaissances à partir de bases de données [FPSS96].

L'apprentissage automatique regroupe un ensemble de méthodes et théories d'analyse de données. Plusieurs familles de méthodes existent en fonction des différents types de problèmes. On parle d'apprentissage supervisé lorsque l'on dispose d'un échantillon d'exemples de paires entrée/sortie à partir duquel on veut concevoir une règle générale ou un modèle qui représente la relation entrée/sortie sous-jacente.

Dans ce domaine, on présente une base de données comme étant un tableau dont les lignes sont des objets (un patient, un client, un état d'un système, ...) et les colonnes des variables, aussi appelées attributs, qui les décrivent (une mesure médicale, le montant d'une transaction, une dimension électrique, ...). Ces attributs dénotent donc les paramètres utilisés pour décrire l'information en entrée. La valeur de sortie peut être soit une classe discrète c'est-à-dire une information symbolique

(le nom de la maladie d'un individu, le caractère stable ou instable d'un réseau, ...) soit une valeur numérique continue. Dans le premier cas on parle d'un problème de classification ou encore de discrimination ou d'apprentissage de concept à partir d'exemples. Dans le second cas, on parle de problème de régression. Dans de nombreuses applications, le nombre d'objets à analyser peut être de plusieurs centaines de milliers, et le nombre d'attributs qui les décrivent peut s'élever à plusieurs milliers ou dizaines de milliers. L'étude complète de ces données par un humain n'est pas réaliste ni souhaitable. L'automatisation complète ou partielle de l'analyse des données est donc nécessaire et c'est l'un des objectifs de l'apprentissage automatique. Une fois un modèle automatiquement construit à partir des données disponibles, celui-ci peut servir à prédire les valeurs de sorties inconnues de nouvelles entrées (c'est-à-dire d'objets non présentés au préalable). Ce modèle peut aussi être utilisé pour expliquer les paires observées par la mise en évidence des relations existantes.

Plusieurs méthodes permettent l'extraction automatique d'un modèle à partir d'un échantillon d'observations décrites par des variables. Les arbres de décision ou les réseaux de neurones en sont des exemples bien connus. Bien que ne construisant pas explicitement un modèle, la méthode des plus proches voisins est la méthode d'apprentissage la plus souvent utilisée en vision par ordinateur. Récemment de nouvelles méthodes d'apprentissage sont apparues. Elles permettent le traitement de problèmes de très grandes dimensions, en particulier dans les cas où les objets sont décrits par un très grand nombre de variables d'entrée. Parmi ces méthodes, les méthodes d'ensembles d'arbres et les SVM sont de plus en plus souvent utilisées.

Les fondements de l'apprentissage automatique sont les mathématiques, en particulier les statistiques et la théorie des probabilités, ainsi que la biologie, spécifiquement les neurosciences. Le processus complet d'extraction de connaissances à partir de bases de données intègre également des méthodes et théories pour le stockage des données (domaine des bases de données), leur visualisation et interprétation (interfaces homme-machine) et leur traitement efficace (algorithmique et parallélisme). Nous présentons ci-dessous les méthodes de classification par apprentissage supervisé que nous utiliserons pour le problème de la classification d'images au cours des chapitres suivants.

2.2 Définitions et notations pour la classification par apprentissage supervisé

L'apprentissage automatique a donc pour but d'extraire de l'information à partir d'une base de données. On considère qu'une base de données est une collection d'objets décrits par un certain nombre d'attributs.

Dans ce contexte, l'information fournie en entrée d'un algorithme d'apprentissage supervisé pour la classification est un échantillon d'objets ou d'exemples décrits

par des variables d'entrée et par une variable de sortie. Cet échantillon est appelé échantillon d'apprentissage. Les objets de celui-ci sont dits pré-classifiés puisque la valeur de leur variable de sortie est connue et fournie à la méthode. Le but est de modéliser les relations entre les entrées et sorties de cet échantillon. On dénote l'échantillon d'apprentissage de N objets par

$$LS = \{(\mathbf{a}^i, c^i), i = 1, \ldots, N\}$$

où $\mathbf{a}^i$ est un vecteur d'attributs d'entrée et $c^i \in \{1, \ldots, M\}$ sa classification parmi M classes discrètes. Selon les différentes méthodes d'apprentissage existantes, les éléments a_k^i $(k = 1, \ldots, m)$ de $\mathbf{a}^i$ peuvent prendre des valeurs numériques ou symboliques. Dans cette thèse, nous nous limiterons à des valeurs numériques. La figure 2.1 donne un exemple d'une telle base de données.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	Classe
60	19	18	17	0	1	1	1	C1
60	3	22	23	1	29	11	23	C1
75	9	2	1	3	77	46	3	C1
2	10	10	2	234	0	0	0	C2
3	7	9	18	5	0	0	0	C2
2	14	5	10	8	10	8	10	C3
65	3	20	21	2	0	1	1	?

FIG. 2.1 – Un exemple de base de données pour la classification : 6 objets décrits par 8 attributs d'entrée à valeurs numériques et une classe de sortie. À partir de cet échantillon, on désire construire automatiquement un modèle de classification qui permettra de prédire la classe du dernier objet.

Les différents algorithmes que nous allons présenter utilisent ces données pour construire un modèle de manière automatique lors de la phase d'apprentissage. Une fois le modèle construit, on peut vérifier si ce modèle permet une généralisation au-delà de l'échantillon d'apprentissage. Pour cela, on utilise un échantillon de test contenant des objets distincts de ceux de l'échantillon d'apprentissage. On exprime généralement la précision du modèle par son taux d'erreur (ou son taux de reconnaissance) sur cet échantillon de test. C'est-à-dire qu'on vérifie pour chaque objet de l'échantillon de test si la classe prédite par le modèle est la classe réelle. Cette information n'est pas disponible en situation réelle, mais la base de données d'objets pré-classifiés est généralement divisée de manière à disposer d'objets de test dont on connaît la classe de sortie. On appelle phase de test l'étape qui réalise cette évaluation. Nous reviendrons sur les protocoles et critères d'évaluation des modèles au sein de la section 2.4.

2.2.1 Sur-apprentissage et sous-apprentissage

Deux sources d'erreur peuvent influencer la précision d'un modèle construit par apprentissage automatique. D'une part, un modèle doit être suffisamment représentatif des cas d'apprentissage et exploiter au mieux l'information contenue dans ces données. Si le modèle construit est trop simple, on parle de sous-apprentissage et on exprime la composante dominante de l'erreur par le biais. D'autre part, un apprentissage trop précis (assimilé à un apprentissage "par cœur") peut ne pas être généralisable. Ce phénomène est appelé sur-apprentissage et l'erreur est exprimée sous la forme d'une variance. On parle de la variance du modèle car sa construction pour un même problème selon un algorithme donné sur différents échantillons d'apprentissage d'une même taille peut conduire à des règles de décision très différentes. Notons que la variance est généralement une fonction décroissante de la taille de l'échantillon d'apprentissage.

Un algorithme de construction automatique d'un modèle doit alors idéalement trouver un bon compromis entre l'adéquation aux données, c'est-à-dire la complexité du modèle, et sa fiabilité. Les deux sources d'erreur, le biais et la variance, sont liées à la complexité du modèle mais de manière opposée. On parle alors du compromis biais/variance (par exemple dans [Geu02]).

2.3 Algorithmes

2.3.1 Plus proches voisins

Le principe de la méthode des plus proches voisins est d'associer une situation inconnue (un nouvel objet) avec une ou plusieurs situation(s) similaire(s) trouvées dans l'échantillon d'apprentissage. Plus précisément, cette méthode consiste à attribuer à un nouvel objet la classe majoritaire parmi les classes de ses plus proches voisins dans l'échantillon d'apprentissage. La notion de voisin ou de similarité est caractérisée par une mesure de distance entre les attributs des objets. Différentes mesures de distance sont utilisées dans la littérature : la norme L1, la distance euclidienne, la distance de Mahalanobis, la distance tangente [SLDV00], etc. La règle du plus proche voisin étant sensible à la distance choisie, il est parfois nécessaire d'attribuer un poids moindre aux attributs peu pertinents et un poids plus élevé aux autres. Des techniques de normalisation aident parfois aussi à relativiser l'importance de certains attributs ou de leurs intervalles de variations. Un autre raffinement de la méthode peut consister à utiliser des définitions différentes de mesures de distance dans des régions différentes de l'espace d'attributs.

Dans sa version la plus simple, l'étape d'apprentissage requiert simplement le stockage de l'échantillon d'apprentissage dans un tableau. Le calcul des distances et

le tri des plus proches voisins sont effectués lors de la phase de prédiction d'un objet inconnu, en parcourant l'entièreté de l'échantillon d'apprentissage. La version de base de cette méthode associe au nouvel objet la classe du plus proche voisin ("nearest neighbor"). L'utilisation de plusieurs voisins ("k nearest neighbors") améliore parfois les résultats par diminution de la variance.

Cet algorithme fournit des informations intéressantes comme la distance aux plus proches voisins, les valeurs de leurs attributs (puisque le modèle est l'échantillon d'apprentissage lui-même) et de manière générale tout type d'information associée à ces voisins. Un désavantage important de cette approche est le temps nécessaire à la prédiction de la classe d'un nouvel objet. En effet, cela implique le parcours exhaustif de l'échantillon d'apprentissage. Des variantes de cette méthode proposent d'organiser cet échantillon de manière arborescente pour en accélérer l'accès, ou utilisent diverses approximations. Un autre désavantage, également très pénalisant, est la difficulté de traiter avec précision des bases de données de grandes dimensions en termes de nombre d'attributs. On parle du problème de "curse of dimensionality" car le nombre d'exemples nécessaires a tendance à croître exponentiellement avec le nombre de dimensions pour obtenir un taux d'erreur satisfaisant [DHD00].

Malgré ces désavantages, cette méthode est largement utilisée. Des structures de données arborescentes ([PKL+02] utilisent un "kd-tree"), des techniques d'approximation ([SSF+03] ou [AMN+98]) et de réduction de dimension rendent son utilisation acceptable, notamment en vision par ordinateur.

2.3.2 Arbre de décision

La principale motivation des méthodes d'induction d'arbre de décision [BFOS84] est la construction automatique d'une collection de règles mutuellement exclusives du type "si-alors" à partir de l'échantillon d'apprentissage. Ces règles sont structurées en arbre binaire et elles peuvent être facilement interprétées par un expert humain.

L'apprentissage automatique de ces règles vise à séparer récursivement les objets de l'échantillon d'apprentissage en sous-échantillons disjoints d'objets où la majorité des objets ont idéalement une même valeur pour la variable de sortie, c'est-à-dire la même classe dans le cas d'un problème de classification. Dans une telle structure arborescente, chaque nœud interne réalise un test sur un attribut d'entrée ("si") dont l'issue sépare l'échantillon d'objets en deux sous-ensembles non-vides et disjoints ("alors"). Ce test est déterminé automatiquement sur base de l'échantillon d'apprentissage et selon des procédures de sélection de test qui diffèrent d'un algorithme d'induction d'arbre à l'autre. Pour des attributs à valeurs numériques, un tel test consiste à comparer la valeur d'un attribut à une valeur numérique qu'on appelle seuil de discrétisation. Selon l'algorithme utilisé, les nœuds terminaux de l'arbre (aussi appelés feuilles) sont étiquetés soit par la classe majoritaire des objets de l'échantillon d'apprentissage qui ont atteint cette feuille suite aux séparations

successives, soit par une distribution de probabilités des classes estimées par la fréquence de ces objets dans chaque classe.

Une fois le modèle construit, on peut inférer la classe d'un nouvel objet en le propageant dans l'arbre de haut en bas selon les tests réalisés sur ses attributs d'entrée. Chaque test en un nœud permet de diriger tout objet vers l'un des deux successeurs de ce nœud sur base de la valeur de l'attribut testé en ce nœud. En commençant à la racine de l'arbre, tout objet traverse l'arbre le long d'un chemin unique et atteint un nœud terminal unique. La classe majoritaire de cette feuille qui avait été déterminée lors de la phase d'apprentissage est attribuée à l'objet. Autrement dit, un arbre de décision est vu comme une fonction qui attribue à tout objet la classe associée au nœud terminal vers lequel est dirigé l'objet consécutivement aux tests aux nœuds internes de l'arbre. Un exemple d'arbre de décision qui utilise un seul attribut d'entrée est donné à la gauche de la figure 2.2, ainsi que le partionnement de l'espace d'entrée qu'il implique (à droite). Les règles "si-alors" qui lui correspondent sont : si $a_1 >= \text{v1}$ alors C = c1, si $a_1 < \text{v1}$ et $a_1 < \text{v2}$ alors C = c1, si $a_1 < \text{v1}$ et $a_1 >= \text{v2}$ alors C = c2.

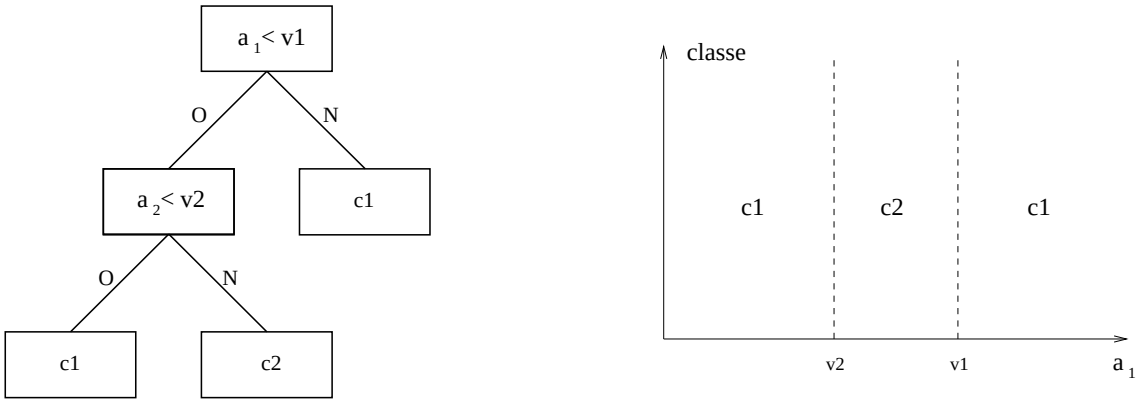

FIG. 2.2 – Un exemple d'arbre de décision et la partition qu'il implique.

La phase d'apprentissage est décrite par la table 2.1. Elle débute au sommet de l'arbre avec un nœud racine contenant tous les objets de l'échantillon d'apprentissage. Pour qu'un arbre soit facilement interprétable, on souhaite minimiser sa taille. Aussi, la procédure de sélection d'un test appliquée en chaque nœud a pour but de choisir le test (la paire attribut-seuil) qui sépare les objets du sous-échantillon courant de façon optimale c'est-à-dire qui diminue le plus possible l'incertitude liée à la variable de sortie au sein des nœuds successeurs. Une mesure d'entropie permet d'évaluer le gain d'information apporté par un tel test. Nous utiliserons un score basé sur une normalisation de la mesure d'information de Shannon [Weh97]. Le test choisi sépare alors l'échantillon courant d'objets en deux sous-échantillons qui se retrouvent dans les successeurs de ce nœud. L'algorithme développe l'arbre de manière récursive sur les successeurs de ce nœud. Typiquement, l'algorithme

Construire_arbre_de_décision(LS) :
- Si LS contient des objets appartenant tous à la même classe, renvoyer une feuille étiquetée avec cette classe.
- Sinon :
  1. Faire [a_k<a_{th}]=Choisir_test(LS) ;
  2. Diviser LS en LS_{left} et LS_{right} selon le test $[a_k < a_{th}]$ et construire les sous-arbres $\mathcal{T}_{left}$ = Construire_arbre_de_décision(LS_{left}) et $\mathcal{T}_{right}$ = Construire_arbre_de_décision(LS_{right}) à partir de ces sous-ensembles ;
  3. Créer un nœud avec le test $[a_k < a_{th}]$, attacher $\mathcal{T}_{left}$ et $\mathcal{T}_{right}$ comme successeurs de ce nœud et renvoyer l'arbre résultant.

Choisir_test(LS) :
1. Sélectionner un attribut a_k et un seuil a_{th} qui maximise la mesure de score calculée sur le LS.

TAB. 2.1 – Algorithme d'induction d'arbre de décision classique.

d'induction poursuit le développement d'un arbre jusqu'à ce que les nœuds terminaux contiennent des sous-échantillons d'objets qui ont une même valeur de sortie. D'autres critères d'arrêt peuvent être établis. L'étiquette associée à une feuille de l'arbre est déterminée à partir des objets de l'échantillon d'apprentissage qui ont été dirigés vers cette feuille. La classe majoritaire parmi les classes de ces objets peut être utilisée, ou encore la distribution de probabilités de classes si un critère d'arrêt a interrompu le développement avant d'atteindre des nœuds "purs".

L'objectif d'un algorithme d'induction est donc de construire un arbre le plus simple possible et dont la fiabilité est maximale (c'est-à-dire le taux d'erreur de classification est minimal) pour les objets de l'échantillon d'apprentissage. Mais un modèle très précis sur l'échantillon d'apprentissage n'est pas nécessairement généralisable à des objets inconnus, notamment si l'échantillon d'apprentissage présente des données bruitées. Comme nous l'avons dit, deux sources d'erreur (exprimées sous la forme du biais et de la variance) peuvent influencer la précision d'un modèle. Plusieurs auteurs ont montré (par exemple [Geu02] et les références citées dans ce travail) que la méthode d'arbre de décision classique souffre d'une variance importante qui pénalise la précision de cet algorithme. En effet, dans le cas d'un arbre, celle-ci peut être trop importante en conséquence d'un nombre trop grand de nœuds tests déterminés au bas de l'arbre sur des sous-échantillons d'objets de taille statistiquement peu fiable. En outre, [Geu02] a montré que le choix des tests (attributs et seuils) aux nœuds internes d'un arbre de décision peut fortement dépendre d'un échantillon à l'autre ce qui contribue également à la variance des modèles construits selon cette méthode. Le rôle des critères d'arrêt du développement d'un arbre ou de

techniques de simplification (élagage) a posteriori est de trouver un bon compromis entre la complexité du modèle et sa fiabilité sur un échantillon indépendant. Toutefois, ces techniques n'améliore que la première source de variance que nous avons mentionnée.

Différentes techniques de réduction de la variance sont proposées dans la littérature, notamment les méthodes d'ensembles d'arbres de décision.

2.3.3 Ensembles d'arbres

Les méthodes d'ensembles consistent à améliorer un algorithme d'apprentissage existant en combinant les prédictions de plusieurs modèles construits à l'aide de celui-ci à partir d'un même échantillon d'apprentissage. La prédiction qui résulte alors de l'agrégation selon une procédure de vote ou de moyenne est potentiellement plus stable, c'est-à-dire de variance plus faible, que la prédiction d'un seul modèle.

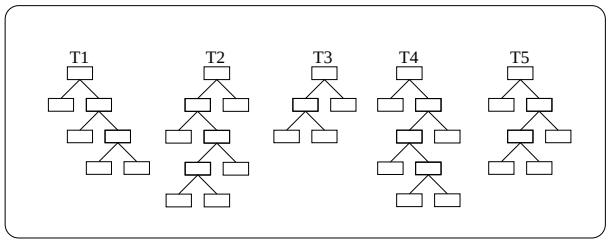

FIG. 2.3 – Une méthode d'ensemble d'arbres construit plusieurs arbres différents à partir d'un même échantillon d'apprentissage.

Ces méthodes sont particulièrement efficaces en combinaison avec la méthode d'arbre de décision qui, sinon, a souvent une précision peu compétitive avec d'autres méthodes. Les méthodes d'ensembles d'arbres de la littérature diffèrent par leur façon d'adapter l'algorithme d'induction d'arbre original et/ou d'agréger les résultats. En effet, à partir d'un échantillon d'apprentissage donné, un algorithme déterministe d'induction d'arbre produira le même modèle. De manière à produire différents modèles et à simuler la variabilité des données réelles, les méthodes d'ensembles perturbent d'une certaine façon l'algorithme original, soit en modifiant les données, soit en introduisant une composante aléatoire dans l'algorithme. Les arbres construits conservent toutefois leur structure hiérarchique avec séparations binaires. Nous présentons dans cette section plusieurs méthodes d'ensembles d'arbres de décision. Ces méthodes ont été appliquée avec succès à des nombreuses applications, notamment en bioinformatique [GFdS$^+$04] et en réseaux [GKL04].

Les trois premières méthodes que nous présentons (Bagging, Random Forests, Extra-Trees) présentent plusieurs similitudes. Elles construisent les T arbres constitutifs de l'ensemble de manière indépendante. Les prédictions des différents arbres

sont agrégées de la manière suivante : chaque arbre produit un vecteur de probabilités de classes. Les vecteurs de probabilités des T arbres sont additionnés en un vecteur de poids et la classe qui reçoit le plus de poids selon celui-ci est attribuée à l'objet. Notons que ces trois méthodes utilisent une composante aléatoire et leur précision peut alors légèrement varier d'une exécution à l'autre. La quatrième méthode présentée, le Boosting, produit l'ensemble d'arbres de manière séquentielle et déterministe contrairement à ces 3 méthodes, et elle utilise une procédure différente d'agrégation.

Bagging

Le Bagging [Bre96] ("boostrap aggregating") consiste à générer à partir de l'échantillon d'apprentissage T sous-échantillons à partir desquels un modèle sera construit selon l'algorithme classique d'induction d'arbre. Ces sous-échantillons sont obtenus par tirage aléatoire avec remplacement. Les études réalisées dans la littérature montrent que cette approche améliore sensiblement la méthode qui utilise un seul arbre de décision classique. Les temps de calcul sont proportionnels au nombre d'arbres de l'ensemble qui sont construits par l'algorithme classique d'induction d'arbre de décision.

Random Forests

La méthode Random Forests [Bre01] construit T arbres à partir de T sous-échantillons obtenus de la même manière que pour le Bagging. Cependant, durant la construction d'un arbre, à chaque nœud interne, un petit nombre k d'attributs est choisi aléatoirement parmi l'ensemble des attributs. Le meilleur test est recherché parmi ce sous-ensemble d'attributs au lieu d'être sélectionné à partir de tous les attributs comme c'est le cas dans le cas de l'algorithme de construction classique d'un arbre. Dans [Bre01], il a été montré que cette méthode donne de meilleurs résultats que le Bagging et qu'elle est souvent compétitive avec d'autres méthodes dont le Boosting (voir plus bas). Elle est aussi plus rapide que ces deux algorithmes puisque pour le développement de chaque nœud interne, seul un sous-ensemble d'attributs est considéré.

Extra-Trees

Cette méthode d'ensemble d'arbres[1] est davantage aléatoire dans le choix des tests aux nœuds internes d'un arbre et elle construit des arbres complètement développés. Elle utilise l'échantillon d'apprentissage au complet. Mais, au lieu de re-

[1] Dans ce document, le terme "arbres aléatoires" sera souvent utilisé pour dénoter les "Extra-Trees".

chercher le test optimal parmi tous les attributs d'entrée (dans le cas du Bagging) ou un sous-ensemble de ceux-ci (dans le cas des Random Forests), elle sélectionne aléatoirement un attribut parmi tous et tire aléatoirement le seuil de discrétisation. Plusieurs variantes du choix aléatoire de test ont été proposées [Geu03]. Celle que nous utiliserons par la suite est décrite par la table 2.2. Le seuil sur le score des tests, s_{th}, garantit qu'un test est relativement discriminant malgré sa sélection aléatoire. Avec ce seuil, si un test aléatoire n'est pas capable de bien séparer les instances selon la mesure de score choisie, alors un autre test aléatoire est considéré. La construction d'un arbre selon cette méthode est plus rapide que les autres méthodes d'ensembles qui se basent en chaque nœud sur la recherche d'un test optimal parmi les attributs d'entrée.

En termes de complexité, la recherche d'un test aléatoire pour séparer un sous-échantillon de taille N requiert le calcul de la moyenne et de l'écart-type pour un attribut tiré aléatoirement. Cette recherche est donc d'ordre N. En général, plusieurs essais sont nécessaires pour trouver un test suffisamment discriminant. Si s_{th} est suffisamment faible, seul un très petit nombre d'attributs (par rapport au nombre total d'attributs d'entrée) devront être considérés et la complexité de l'algorithme est pratiquement indépendante du nombre d'attributs. Augmenter la profondeur de l'arbre de un niveau implique de considérer des sous-échantillons dont le nombre total d'éléments est N. Si l'on suppose que l'arbre est relativement équilibré, alors sa profondeur est d'ordre $log_2(N)$ (elle est limitée par la taille de l'échantillon d'apprentissage) et l'algorithme de construction d'arbres aléatoires est alors d'ordre $N.log_2(N)$. La construction d'un ensemble de T arbres aléatoires demande donc de l'ordre de $T.N.log_2(N)$ opérations.[2] Une autre variante consiste en chaque nœud à tirer un test aléatoire pour un nombre fixé d'attributs choisis aléatoirement et de sélectionner parmi ces tests celui qui offre le meilleur score. Cette variante, très récemment proposée [GEW04], s'avère plus rapide en pratique que la variante avec seuil sur le score et présente un temps d'exécution mieux prévisible, pour une précision comparable.

Cette méthode se compare avec la méthode des Random Forests en termes de précision et est plus rapide que celle-ci et que toutes les autres méthodes d'ensembles d'arbres. Sa bonne précision est expliquée selon une étude du compromis biais/variance dans [Geu02] et [GEW04]. Avec cette méthode, chaque arbre est construit jusqu'à ce qu'il classe parfaitement l'ensemble d'apprentissage (les nœuds terminaux contiennent chacun un ou plusieurs objets de la même classe et ils sont étiquetés par celle-ci). Ainsi, le biais est réduit puisque la complexité de l'arbre n'est limitée que par la taille de l'ensemble d'apprentissage. En outre, le caractère aléatoire de cette méthode génère des arbres indépendants qui simulent la variabilité des données et la procédure d'agrégation réduit de façon importante la variance.

[2]Dans le pire des cas, en fonction de s_{th}, la complexité de l'algorithme de construction d'un arbre aléatoire est multipliée par un facteur correspondant au nombre total d'attributs et équivaut alors à la complexité de la méthode d'arbre classique.

Choisir_test(LS) :

1. Sélectionner aléatoirement un attribut a_k ;
2. Sélectionner aléatoirement un seuil a_{th} selon une distribution $N(\mu_k, \sigma_k)$, où μ_k et σ_k sont respectivement la moyenne et l'écart-type des valeurs de l'attribut a_k dans LS ;
3. Si le score de ce test est supérieur à un seuil donné s_{th}, renvoyer le test $[a_k < a_{th}]$;
4. Sinon, retourner à l'étape 1 et sélectionner un autre attribut. Si tous les attributs ont déjà été considérés, renvoyer le meilleur test obtenu jusqu'ici.

TAB. 2.2 – Algorithme d'induction d'arbre aléatoire.

Boosting

Par opposition aux méthodes d'ensembles précédentes, le Boosting produit un ensemble de T arbres de manière séquentielle et déterministe. Selon cette technique, l'algorithme de construction d'arbre est appliqué successivement à l'échantillon d'apprentissage original dont la pondération des objets est adaptée au fur et à mesure. Initialement, le poids des objets de l'échantillon d'apprentissage est uniforme. Ensuite, à chaque itération, le poids des objets qui sont mal classés par le nouvel arbre est augmenté. Les modèles sont donc en quelque sorte forcés de se concentrer sur les instances les plus difficiles qui reçoivent un poids de plus en plus élevé.[3] La prédiction d'un nouvel objet est réalisée selon un vote moyen pondéré. Lors de l'addition des vecteurs de probabilités de classes, le vecteur de chaque arbre est pondéré par un facteur propre à la précision individuelle de cet arbre sur l'échantillon d'apprentissage.

L'avantage de cette méthode est sa précision qui a été évaluée sur de nombreux problèmes dans la littérature. En général, cette méthode est meilleure que le Bagging et comparable aux Random Forests. Nous avons utilisé la variante Adaboost.M1 décrite dans [FRS96]. Cet algorithme requiert que chaque arbre construit ne soit pas parfait sur l'ensemble d'apprentissage (de manière à fournir des instances mal classées).[4] Puisque les nœuds internes sont déterminés selon la méthode classique de construction d'arbre, comme avec le Bagging, cette approche est habituellement moins rapide que des méthodes qui ne recherchent pas des tests optimaux pour tous les attributs aux nœuds internes des arbres : les Random Forests et les arbres aléatoires.

[3] La mesure de score pour le choix d'un test optimal aux nœuds internes tient alors compte du poids des objets.

[4] Nous utilisons alors des arbres de décision élagués avec le critère d'arrêt décrit par [Weh97]. Il se base sur un test d'hypothèse basé sur le G^2 [Kvå87] pour déterminer l'importance d'un test. Dans ce travail, le risque de non-détection est fixé à 0,005.

2.3.4 SVM

La méthode de machines à support vectoriel (SVM) est une méthode qui résulte d'avancées théoriques en statistiques et en apprentissage ([Cor02], [CST00]). La construction d'un modèle est réalisée en deux étapes. La première consiste à transformer l'espace d'entrée original vers un espace de représentation intermédiaire de grande dimension dans lequel il est généralement possible de séparer linéairement les données. La deuxième étape consiste à produire pour chaque paire de classe une frontière de décision linéaire, c'est-à-dire un hyperplan, qui classe les données d'apprentissage c'est-à-dire qui sépare chaque paire de classes dans ce nouvel espace. L'hyperplan optimal est défini comme celui dont la distance entre les points les plus proches et lui-même, qu'on appelle la marge, est maximisée. Il est construit de manière à se trouver le plus loin possible des objets d'apprentissage des 2 classes. Cette maximisation est intuitivement plus sûre. Une légère variation des données ne modifiera pas leur classification si la distance à l'hyperplan est grande. Les objets les plus proches de cet hyperplan sont appelés les vecteurs de support. La recherche de l'hyperplan dans l'espace intermédiaire de grande dimension (parfois infini) est rendue faisable par le fait que la solution peut s'exprimer uniquement en fonction du produit scalaire dans cet espace. On peut donc se contenter de définir une fonction noyau réalisant ce produit scalaire sans calculer explicitement le vecteur d'attributs dans l'espace intermédiaire. Cette simplification s'appelle le "kernel trick". Les fonctions noyaux peuvent être adaptées à chaque problème en fonction de la structure des données. La classification d'un nouvel objet implique le produit scalaire de cet objet avec les vecteurs de support. Cette phase dépend donc du nombre de vecteurs de support qui est généralement bien inférieur au nombre d'objets de l'échantillon d'apprentissage.

Depuis quelques années, plusieurs fonctions noyaux et plusieurs implémentations pour cette méthode ont montré de très bons résultats dans de nombreux domaines complexes. Dans ce travail, nous avons utilisé l'algorithme présenté dans [CL03].

2.4 Évaluation

Plusieurs critères peuvent déterminer le choix d'une méthode d'apprentissage pour un problème donné. Quatre critères sont principalement utilisés et un compromis entre eux est généralement recherché.

Précision

L'objectif d'une méthode de classification est généralement de prédire correctement la classe de nouveaux objets, c'est-à-dire d'objets non présentés lors de la phase d'apprentissage.

Pour réaliser l'évaluation de la précision d'une méthode, plusieurs protocoles existent. Souvent, on utilise un échantillon de test indépendant c'est-à-dire qui contient des objets distincts de ceux de l'échantillon d'apprentissage. La base de données d'objets disponibles est donc séparée en deux échantillons disjoints. On calcule le taux ou la proportion d'erreur par le comptage du nombre d'objets classés erronément dans l'échantillon de test par application du modèle induit à partir de l'échantillon d'apprentissage.

Dans le cas où la taille de la base de données est petite, les résultats peuvent être instables d'une exécution à l'autre c'est-à-dire d'un choix à l'autre des échantillons d'apprentissage et de test. On préfère alors généralement réaliser plusieurs exécutions sur différents échantillons afin d'avoir une estimation plus fiable de la précision.[5] La validation croisée est l'un de ces protocoles. Elle consiste à diviser la base de données en k sous-échantillons disjoints d'objets. Un modèle est construit à partir de l'union de ces sous-échantillons à l'exception d'un seul sous-échantillon qui servira d'échantillon de test. La procédure est répétée k fois, chaque sous-échantillon servant une fois d'échantillon de test et un modèle étant construit à chaque reprise à partir des $k-1$ autres sous-échantillons. L'erreur moyenne sur les k exécutions est alors utilisée comme estimation de l'erreur d'un modèle qui aurait été déterminée sur la totalité des données. La validation croisée stratifiée est une variante de ce protocole où les proportions des classes de la base de données sont conservées dans les sous-échantillons. Lorsque k est égal au nombre d'objets dans la base de données, le protocole est appelé "leave-one-out". L'apprentissage est dans ce cas réalisé sur tous les objets de la base de données à l'exception d'un seul, pour lequel on évaluera l'exactitude de la classification. Cette procédure est répétée pour tous les objets de la base de données.

Efficacité

On mesure l'efficacité d'une méthode par le temps nécessaire à l'apprentissage du modèle ainsi que le temps nécessaire à l'application de ce modèle à de nouveaux objets. L'importance respective de ces deux phases dépend du problème et de son application. Généralement, le temps de prédiction d'un nouvel objet est un critère particulièrement important.

Autonomie

L'autonomie d'une méthode c'est-à-dire sa capacité à être appliquée aux données sans adaptation (des données ou des paramètres de l'algorithme) est en pratique une

[5] Ces protocoles d'évaluation qui réalisent plusieurs exécutions sur différents échantillons d'apprentissage et de test afin de stabiliser l'estimation de l'erreur ne sont pas à confondre avec des méthodes d'ensembles qui construisent plusieurs modèles à partir d'un même échantillon d'apprentissage afin d'améliorer la précision évaluée sur un même échantillon de test. Ces protocoles ne visent pas à améliorer la précision mais à l'estimer de manière plus fiable.

notion complémentaire au temps d'apprentissage proprement dit. Elle est particulièrement importante dans des domaines où de nombreux sous-problèmes peuvent apparaître.

Interprétabilité

L'interprétabilité d'un modèle est un facteur qui peut être subjectif et dont l'importance dépend également du problème. Un modèle interprétable (et précis) augmente la connaissance du domaine étudié. Un arbre de décision qui ne comporte pas beaucoup de nœuds peut être considéré comme facilement interprétable par un humain. Un arbre avec plusieurs centaines de nœuds ou un ensemble d'arbres sont par contre moins faciles à comprendre. Certaines applications requièrent une bonne interprétabilité, ce qui n'est pas le cas de toutes les applications de prédiction.

Chapitre 3

L'apprentissage automatique pour la classification d'images

Dans ce chapitre nous traitons la classification d'images comme un problème d'apprentissage automatique supervisé. Nous appliquons les méthodes présentées au chapitre 2 directement à partir des valeurs des pixels des images. Afin de déterminer les éventuelles limites d'une telle approche générique, notre évaluation est réalisée sur quatre bases de données diverses : la classification de chiffres (MNIST), d'objets en 3D (COIL-100), de visages (ORL) et de textures (OUTEX). Les résultats principaux de ce chapitre ont été publiés dans [MGV+03].

3.1 Contexte

Nous avons vu au sein de l'introduction que les problèmes de classification d'images sont de plus en plus nombreux et variés. Une approche courante est de développer une méthode spécifique pour chaque problème. Celle-ci repose sur la sélection manuelle d'un ensemble de caractéristiques et l'utilisation de ces informations en entrée d'une méthode d'apprentissage (souvent, la méthode des plus proches voisins). Du point de vue de l'apprentissage, ce prétraitement spécifique permet de réduire la complexité du problème (sa haute dimensionnalité) qui résulterait de l'utilisation des pixels des images pour décrire celles-ci. En effet, dans un tel contexte, la plupart des méthodes d'apprentissage souffrent d'une grande variance qui dégrade leur précision et de plus elles présentent des temps de calcul pénalisants. L'utilisation de méthodes automatiques de réduction explicite de dimension (dont l'analyse en composantes principales) suit le même objectif : la réduction de l'espace d'entrée pour en permettre le traitement par une méthode d'apprentissage automatique.

Toutefois, les récentes avancées en apprentissage automatique ont fait apparaître des méthodes capables de traiter des problèmes de plus en plus complexes sans

utiliser d'information a priori sur le domaine d'application. Elles rivalisent souvent avec les méthodes spécifiques à ces domaines qui, elles, résultent pourtant d'une adaptation importante au problème d'application.

3.2 Description de l'approche

Dans ce contexte, comme première approche générique, nous proposons d'évaluer l'utilisation de méthodes récentes d'apprentissage automatique pour la classification d'images, méthodes qui peuvent travailler directement avec les valeurs des pixels des images globales comme variables descriptives de celles-ci. Premièrement, cette représentation est générique : elle peut s'appliquer à tous types d'images et ne dépend pas de la détection de primitives au sein des images. Deuxièmement, elle est potentiellement plus discriminante qu'un ensemble réduit de caractéristiques puisque toute l'information des images est utilisée. Au contraire, une phase intermédiaire entre l'acquisition des images et leur classification peut engendrer la perte explicite d'information.

La phase d'apprentissage consiste donc à construire automatiquement un modèle à partir des images étiquetées et des valeurs de pixels qui les constituent. Cette construction utilise un algorithme d'apprentissage automatique générique et nous évaluons dans ce chapitre plusieurs de ces méthodes.

La phase de prédiction d'une nouvelle image consiste à utiliser le modèle construit pour lui attribuer une classe.

3.2.1 Phase d'apprentissage

L'information fournie en entrée d'un algorithme d'apprentissage pour la classification d'images est un ensemble d'images pré-classifiées,

$$LS = \{(\mathbf{a}^i, c^i), i = 1, \ldots, N\}$$

où $\mathbf{a}^i$ est un tableau de dimension $W_x \times W_y$ qui décrit l'image et $c^i \in \{1, \ldots, M\}$ sa classification (parmi M classes). Les éléments $a^i_{k,l,ch}$ de $\mathbf{a}^i$ ($k = 1, \ldots, W_x$, $l = 1, \ldots, W_y$,$ch = 1, \ldots, CH$) décrivent les pixels aux positions (k, l) sous la forme d'une valeur entière dans le cas d'images en mode niveaux de gris ($CH = 1$) ou sous la forme de 3 valeurs entières des canaux rouge, vert et bleu (RGB) dans le cas d'images en mode couleurs 24 bits ($CH = 3$).[1] Les méthodes travailleront sur ces valeurs entières variant entre 0 et 255. Étant donné la nécessité pour la plupart des méthodes d'apprentissage automatique de travailler avec le même nombre d'attributs

[1] Au sein de la section 5.2.1, nous évaluerons la représentation HSV. Dans ce cas 3 valeurs flottantes seront utilisées.

Construire_arbre_aléatoire(LS) :
- Si LS contient des images appartenant toutes à la même classe, renvoyer une feuille étiquetée avec cette classe.
- Sinon :
  1. Faire $[a_{k,l,ch} < a_{th}]$ = Choisir_test_aléatoire(LS) ;
  2. Diviser LS en LS_{left} et LS_{right} selon le test $[a_{k,l,ch} < a_{th}]$ et construire les sous-arbres $\mathcal{T}_{left}$ = Construire_arbre_aléatoire(LS_{left}) et $\mathcal{T}_{right}$ = Construire_arbre_aléatoire(LS_{left}) à partir de ces sous-ensembles ;
  3. Créer un nœud avec le test $[a_{k,l,ch} < a_{th}]$, attacher $\mathcal{T}_{left}$ et $\mathcal{T}_{right}$ comme successeurs de ce nœud et renvoyer l'arbre résultant.

Choisir_test_aléatoire(LS) :
1. Sélectionner aléatoirement une position et un canal (k, l, ch) ;
2. Sélectionner aléatoirement un seuil a_{th} selon une distribution $N(\mu_{k,l,ch}, \sigma_{k,l,ch})$, où $\mu_{k,l,ch}$ et $\sigma_{k,l,ch}$ sont respectivement la moyenne et l'écart-type des valeurs $a_{k,l,ch}$ dans LS ;
3. Si le score de ce test est supérieur à un seuil donné s_{th}, renvoyer le test $[a_{k,l,ch} < a_{th}]$;
4. Sinon, retourner à l'étape 1 et sélectionner une autre position et un autre canal. Si toutes les positions et canaux ont déjà été considérés, renvoyer le meilleur test obtenu jusqu'ici.

TAB. 3.1 – Algorithme d'induction d'arbre aléatoire pour la classification d'images.

pour tous les objets, les images d'une même base de données devront avoir la même résolution, aussi bien en termes de dimensions (W_y et W_x) que de nombre de canaux (CH). Ainsi, un attribut dénote la même position de pixel et le même canal de couleur d'une image à l'autre.

Suivant ces définitions, les algorithmes d'apprentissage automatique présentés au chapitre précédent sont directement applicables au problème de classification d'images. La table 3.1 montre l'algorithme d'induction d'un arbre aléatoire (présenté au sein de la section 2.3.3) dans ce contexte. La figure 3.1 illustre un arbre de décision classique simple qui réalise les tests de comparaisons des valeurs de pixels à ses nœuds internes et attribue une classe aux feuilles de l'arbre. La figure 3.2 illustre la phase d'apprentissage pour les méthodes d'ensembles d'arbres.

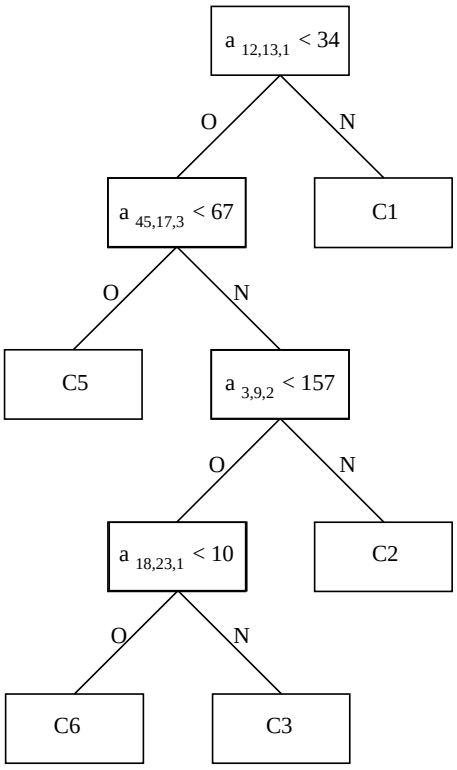

FIG. 3.1 – Un arbre de décision simple avec tests de valeurs de pixels qui constituent l'image.

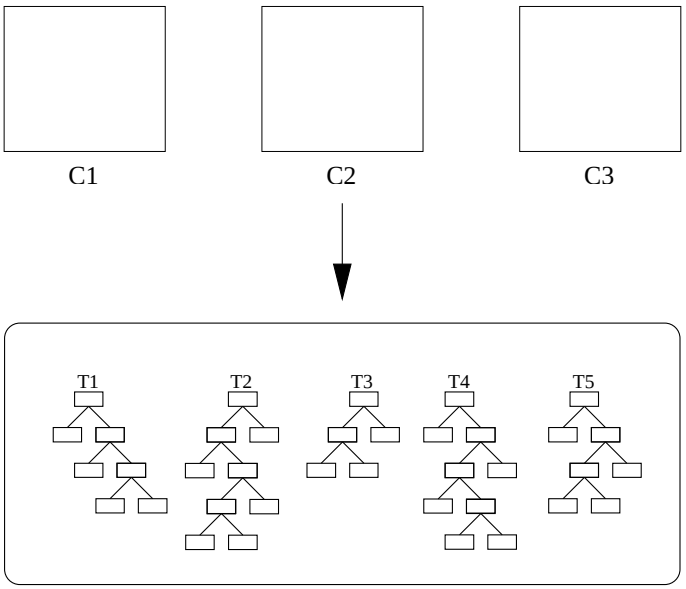

FIG. 3.2 – Phase d'apprentissage : construction d'un ensemble d'arbres ($T = 5$) à partir des images globales pré-classifiées.

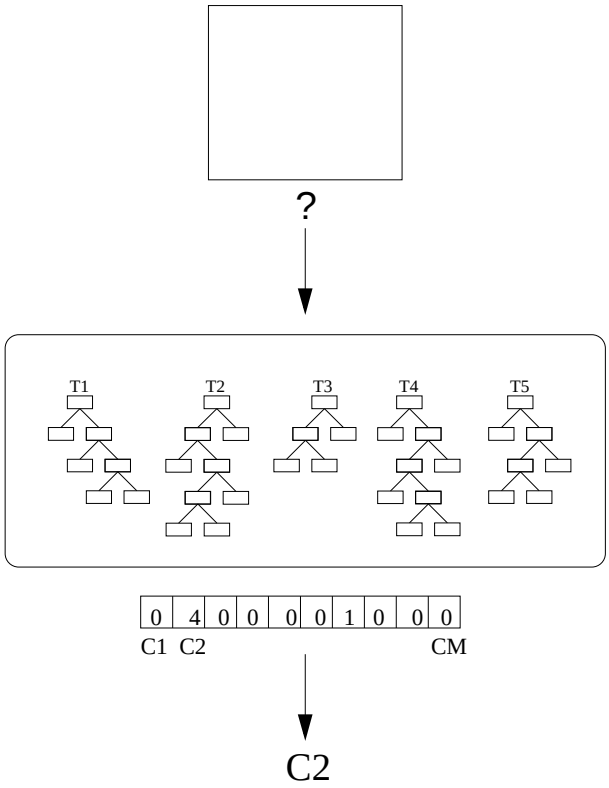

FIG. 3.3 – Phase de prédiction : propagation de l'image complète dans l'ensemble d'arbres et prédiction de la classe par recherche du maximum parmi les votes des T arbres pour les M classes.

3.2.2 Phase de prédiction

La phase de prédiction d'une nouvelle image consiste à utiliser le modèle construit lors de la phase d'apprentissage pour prédire sa classe, comme pour tout autre problème de classification. Les images sont décrites par les pixels qui les composent et les modèles utilisent cette information pour déterminer la classe de l'image. La figure 3.3 illustre cette phase pour une méthode d'ensemble d'arbres, en particulier les arbres aléatoires. Selon cette méthode, pour une image, chaque arbre vote pour une des M classes. Les T votes sont additionnés et la classe recevant le plus de votes est attribuée à la nouvelle image.

3.2.3 État de l'art

Dans la littérature, quelques études utilisant des méthodes récentes d'apprentissage à partir des pixels ont été menées. Pour la classification de chiffres manuscrits (sur le problème MNIST), [BS97] a utilisé les machines à support vectoriel. Pour

la reconnaissance d'objets 3D (sur le problème COIL-100), les machines à support vectoriel ont été utilisées par [PV98], [RZE01] et [dSG02], l'architecture SNoW par [YAR00]. [KJPK02] combine des SVM et un réseau de neurones pour la classification de textures à partir des niveaux de gris. Par ailleurs, [LBH04] a récemment comparé l'utilisation des SVMs, des plus proches voisins et des réseaux de neurones à convolution [2] sur une base de données de catégories d'objets.[3]

Des méthodes récentes d'apprentissage ont également été utilisées pour d'autres problèmes de classification ou de détection mais avec un pré-traitement spécifique ou l'application d'une méthode de réduction de dimensions. Le Boosting a été utilisé par [GZ01] pour la classification de visages avec analyse à composantes principales, par [VJ01] pour la détection de visages avec caractéristiques de voisinage basées sur une variante de filtres de Haar et par [How02] pour la recherche d'images avec plusieurs types de représentations dont les histogrammes et corrélogrammes. Les SVM ont été utilisées par [GLC00] pour la classification de visages avec analyse à composantes principales. Les arbres de décision sont utilisés dans [JF96] pour la classification d'objets de COIL-20 avec construction de codes de description du voisinage associé à chaque pixel.

3.3 Évaluation

3.3.1 Critères d'évaluation

Comme nous l'avons expliqué au chapitre précédent (voir la section 2.4), le principal critère d'évaluation d'une méthode d'apprentissage est son taux d'erreur sur un ensemble de test distinct de l'ensemble d'apprentissage. Puisque nous désirons évaluer la pertinence de l'approche par apprentissage à partir des pixels et comparer les méthodes, nous utiliserons ici ce critère. Les temps de prédiction seront donnés à titre d'information car l'implémentation des méthodes n'est pas forcément optimisée.

3.3.2 Bases de données et protocoles

Pour évaluer les méthodes d'apprentissage, nous avons choisi quatre bases de données fréquemment utilisées dans la littérature. Celles-ci sont publiques et gratuites, et nous pensons qu'elles constituent un panel représentatif des problèmes courants

[2] Ce type de réseaux de neurones se distingue essentiellement par sa topologie : un neurone d'une couche cachée est connecté à un sous-ensemble de neurones de la couche précédente plutôt qu'à tous les neurones de celle-ci dans un perceptron. Chaque neurone peut être considéré comme une unité de détection d'une caractéristique locale ([LBBH98], [PVG01]).

[3] Ce travail utilise en outre l'information "stereo" fournie par cette base de données (`http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/`).

de classification d'images. Les caractéristiques de ces bases de données sont résumées dans le tableau 3.2. MNIST concerne la classification de chiffres manuscrits, ORL consiste à reconnaître des visages humains, COIL-100 implique la classification d'objets 3D et OUTEX la classification de textures. Les résultats concernant ces bases de données sont largement disponibles individuellement, mais à notre connaissance aucune étude n'a comparé systématiquement une même méthode d'apprentissage sur quatre problèmes aussi divers. En effet, en général, une méthode est présentée pour un type de problème donné. Cela ne signifie pas qu'aucune méthode précédemment développée ne pourrait être appliquée à ces divers problèmes avec succès, mais aucune étude ne l'a réalisé.

Bases de données	# images	# attributs	# classes
MNIST	70000	784 $(28 * 28 * 1)$	10
ORL	400	10304 $(92 * 112 * 1)$	40
COIL-100	7200	3072 $(32 * 32 * 3)$	100
OUTEX	864	49152 $(128 * 128 * 3)$	54

TAB. 3.2 – Résumé des 4 bases de données.

MNIST

La base de données MNIST [4] contient 70000 images de chiffres écrits à la main par 500 personnes, de taille normalisée et centrée en images de 28×28 pixels avec 256 niveaux de gris par pixel. Le but est de construire un modèle qui classe les chiffres. Les différentes écritures se distinguent par des traits fins ou épais, des caractères plus ou moins penchés,... Quelques exemples d'images d'apprentissage pour le chiffre 9 sont donnés par la figure 3.4.

Dans la littérature, le protocole d'expérimentation pour cette base de données consiste généralement à utiliser les 60000 premières images pour l'ensemble d'apprentissage et les 10000 dernières pour l'ensemble de test. Nous avons suivi ce schéma, la validation croisée n'étant pas nécessaire en présence d'un si grand nombre de cas.

Plusieurs méthodes ont été proposées dans la littérature ([LBBH98]) pour aborder ce problème dont des méthodes d'apprentissage automatique comme les plus proches voisins et les réseaux de neurones, généralement précédées par une phase de pré-traitement, et les machines à support vectoriel. En effet, certaines de ces approches ont utilisé un pré-traitement spécifique à la reconnaissance de caractères destiné à augmenter la taille de l'échantillon en générant des versions déformées des images originales (combinaisons de décalages, inclinaisons, mises à l'échelle, compressions) ou consistant à corriger l'inclinaison des chiffres. Le taux d'erreur des

[4] http ://yann.lecun.com/exdb/mnist/

FIG. 3.4 – MNIST : quelques images d'apprentissage pour la classe "9".

méthodes référencées par [LBBH98] varie de 12% à 0.7%. L'utilisation par [BS97] de machines à support vectoriel sans information a priori a donné 1.1% d'erreur.

ORL

La base de données ORL (AT&T Laboratories Cambridge)[5] contient des images du visage de 40 personnes. Chacune de ces personnes a été photographiée à 10 reprises, sous différentes conditions relatives à l'éclairage, l'expression faciale (yeux ouverts ou fermés, sourire ou non, avec ou sans lunettes) et de faibles différences d'inclinaison de la tête. Cette base de données ne présente pas de variation en terme de taille ni de translation (c'est-à-dire les visages sont tous plus ou moins centrés). Chaque image est de taille 92×112 pixels avec 256 niveaux de gris par pixel. Le but est de reconnaître les personnes.

Dans la littérature, divers algorithmes ont été testés sur cette base de données tels que des modèles de Markov cachés [NH99], des réseaux de neurones à convolution [LGTB97], des machines à support vectoriel [GLC00] et des variantes des plus proches voisins [PPC01]. Ces études utilisent diverses étapes de pré-traitement pour réduire la dimensionnalité du problème Ces expériences utilisent souvent des méthodologies différentes (moyenne de 3, 4 ou 5 exécutions sur des groupes de données distincts) ce qui rend la comparaison délicate étant donné les variations présentes parmi les images d'une même personne photographiée à diverses reprises comme le montre la figure 3.5. Les résultats rencontrés dans la littérature varient de 7.5% à 0% mais le nombre d'exécutions n'est pas précisé dans ce dernier cas.

FIG. 3.5 – Variations des images d'une même personne pour la base de données ORL : port de lunettes, changement d'illumination, changement d'orientation et expressions variées du visage.

Étant donné le peu d'images qui constituent cette base de données et l'absence d'un protocole de test public, nous avons ici utilisé la validation croisée afin de mieux

[5] http ://www.uk.research.att.com/facedatabase.html

nous rendre compte des capacités de généralisation des méthodes étudiées.

Dans [MGW03], pour l'évaluation des arbres extrêmement aléatoires, nous avons moyenné les résultats de 100 exécutions en utilisant 5 images distinctes tirées aléatoirement par classe pour chaque ensemble d'apprentissage et l'autre moitié des images pour chaque ensemble de test. Ici, nous avons choisi un autre protocole, principalement à cause des temps de calcul trop importants de certaines méthodes qui auraient dû être exécutées 100 fois (notamment le Bagging et le Boosting). Dix ensembles d'apprentissage contenant 360 images (9 vues par personne) ont donc été construits tandis que les 40 images restantes (1 vue par personne) ont été utilisées pour les ensembles de test.

COIL-100

La bibliothèque d'images de l'université de Columbia [6] propose des bases de données d'images d'objets 3D. COIL-100 est un ensemble d'images représentant 100 objets. Il contient 7200 images prises par une caméra fixe avec les objets se trouvant sur une platine motorisée réalisant une rotation de 360 degrés par intervalles de 5 degrés. Cela correspond donc à 72 images par objet et chaque image est de taille 128×128 pixels en couleurs. Le but est de reconnaître les différents objets (boîtes, flacons, animaux et aliments en plastique, voitures miniatures, ...).

Ce problème a été abordé dans la littérature à l'aide de nombreuses techniques dont des méthodes de mise en correspondance de caractéristiques globales ou locales (histogrammes de couleurs, Local Affine Frames (LAFs) [OM02b]), les machines à support vectoriel [PV98], etc. Le sous-problème COIL-20 a également été traité par de nombreux chercheurs utilisant par exemple des techniques de réduction paramétrique de l'espace d'entrée [MN95] ou un pré-traitement qui caractérise structurellement les pixels des images [JF96]. Nous avons également traité ce sous-problème dans [MGW03].

Habituellement, l'ensemble d'apprentissage pour COIL-100 consiste en images de chaque objet prises à intervalles réguliers des angles de rotation. Dans cette étude, nous prenons 18 vues pour chacun des 100 objets, en démarrant à la pose 0° et continuant par pas de 20°, ce qui constitue un ensemble d'apprentissage de 1800 images. Un exemple des vues d'apprentissage d'un objet est donné à la figure 3.6. Le reste des images (54 par objet) sont utilisées pour l'ensemble de test comportant donc au total 5400 éléments. Les méthodes de la littérature fournissent des taux d'erreur variant de 12.5% à 0.1% en respectant ce protocole. Certaines études utilisent différentes résolutions pour les images. Nous avons utilisé des images de résolution 32×32 pixels.

[6] http ://www.cs.columbia.edu/CAVE/

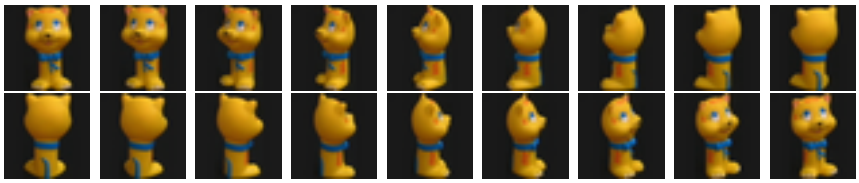

FIG. 3.6 – COIL-100 : les 18 vues d'un objet utilisées pour l'apprentissage.

OUTEX

Outex [OMP+02] est un cadre de travail pour l'évaluation empirique d'algorithmes d'analyse de textures. La base de données Contrib_TC_0006 incluse dans cet environnement [7] est construite à partir des textures VisTeX [8]. Elle contient 54 textures en couleurs avec 16 sous-images de taille 128×128 pour chaque texture VisTeX originale. Le but est de reconnaître les différentes textures (bois, métal, eau, sable, ...).

Cette base de données présente un petit nombre d'objets, un très grand nombre d'attributs et un nombre assez important de classes. L'environnement OUTEX spécifie les images utilisées pour l'ensemble d'apprentissage et l'ensemble de test (8 images par classe par ensemble), nous avons donc adopté ce protocole puisqu'il est précis et public. La figure 3.7 illustre les vues d'une texture de fleurs utilisées pour l'apprentissage.

Dans [OMP+02] sont évaluées plusieurs méthodes d'extraction de caractéristiques et de transformation des images en amont d'une méthode traditionnelle des plus proches voisins. Leurs résultats sur cet ensemble de données varient de 9.5% à 0.2% de taux d'erreur.

FIG. 3.7 – OUTEX : les 8 vues d'une texture utilisées pour l'apprentissage.

[7] http ://www.outex.oulu.fi/outex.php
[8] http ://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html

3.3.3 Méthodes d'apprentissage, paramètres et implémentation

Pour cette étude, nous désirons d'abord mettre en évidence les difficultés inhérentes à l'application des méthodes d'apprentissage à ces problèmes de grandes dimensions. Ensuite, nous profiterons de récentes avancées en apprentissage, présentées au chapitre précédent, qui ont fait apparaître des méthodes capables de gérer un nombre très large de variables en entrée.

Nous appliquerons donc premièrement deux méthodes de référence, une méthode d'arbre de décision [BFOS84] et la méthode des plus proches voisins, afin de mettre en avant leur inefficacité (en termes de précision ou de temps de calcul) lorsqu'elles sont appliquées à un problème complexe comme la classification d'images. Ensuite, nous utiliserons des méthodes plus récentes : le Bagging [Bre96], le Boosting [FRS96], les Random Forests [Bre01], les Extra-trees [Geu02] et les SVM [Vap95].

Comme nous l'avons vu au chapitre 2, les algorithmes d'apprentissage dépendent d'un nombre relativement restreint de paramètres qu'il est profitable d'adapter au problème afin d'obtenir les meilleurs résultats. Selon la méthode et le problème, une méthode peut donner des résultats différents en fonction des paramètres utilisés. Pour cette étude, nous avons donc tenu compte de cette variabilité en cherchant les paramètres qui donnent les meilleurs résultats, ou des résultats stables. Toutefois, le but principal de cette étude est d'évaluer l'applicabilité des méthodes d'apprentissage au problème de classification d'images en général. De légères améliorations pourraient probablement être apportées à nos résultats en approfondissant encore cette optimisation des paramètres, mais les résultats présentés ci-après fournissent une idée assez réaliste des capacités des différentes méthodes à traiter les problèmes considérés.

Pour tous les algorithmes utilisés ici, à l'exception des SVM, nous avons utilisé le logiciel développé par Pierre Geurts [9] qui est implémenté en C et auquel nous avons ajouté quelques fonctions relatives à notre tâche. Pour les SVM, nous avons utilisé l'outil LibSVM [10] qui est une implémentation en C++ de l'algorithme présenté dans [CL03].

Pour ces algorithmes, nous proposons de déterminer les valeurs optimales des paramètres sur base du taux d'erreur résultant sur l'échantillon d'images de test. Nous sommes conscients du fait que cela pourrait mener à des taux d'erreur légèrement sous-estimés. Cependant, nous pensons que ce fait n'est pas nuisible pour notre étude comparative, étant donné que le nombre de paramètres est relativement restreint.

[9] `http://www.montefiore.ulg.ac.be/~geurts/`
[10] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Arbre de décision

Avec l'implémentation que nous utilisons, les arbres de décision classiques sont complètement développés, c'est-à-dire qu'aucune technique d'élagage n'est utilisée [11]. La mesure de score utilisée pour évaluer les tests aux nœuds internes des arbres durant la phase d'induction est celle proposée par [Weh97] qui est une normalisation particulière du gain d'information. À l'exception de cette variante, l'algorithme que nous utilisons est similaire à la méthode CART [BFOS84].

Ensembles d'arbres

Les méthodes d'ensembles sont toutes influencées par le nombre d'arbres T qui sont construits et agrégés. Notons que habituellement, plus le nombre d'arbres est grand, meilleure est la précision. Dans nos expériences, nous utilisons donc pour chaque problème et pour chaque algorithme, un nombre d'arbres qui semblent être suffisamment grand pour donner une erreur stable sur l'ensemble de test. Habituellement, les arbres aléatoires qui sont construits de façon plus aléatoire que les autres méthodes d'ensembles d'arbres requièrent un nombre plus grand d'arbres que les autres variantes [Geu03].

La variante des arbres aléatoires que nous appliquons utilise le seuil sur la mesure de score du test au sein d'un noœud, s_{th}. Sauf mention contraire, nous avons fixé ce seuil dans nos expériences à 0.1 (ce qui correspond à 10% de la valeur maximale possible du score).

Les *Random Forests* dépendent d'un paramètre additionnel k qui est le nombre d'attributs choisis aléatoirement en chaque nœud de test. Dans nos expériences, sa valeur a été fixée à la valeur par défaut suggérée par l'auteur de l'algorithme et qui représente la racine carrée du nombre total d'attributs. En se référant à [Bre01], cette valeur donne habituellement le meilleur taux d'erreur.

SVM

Pour les machines à vecteur de support, nous avons utilisé les paramètres par défaut de la librairie LibSVM.[12] Nous avons essayé l'implémentation des noyaux linéaires, polynomiaux (de degré 2 et 3) et radiaux. Ici encore, le meilleur noyau est choisi sur base du taux d'erreur du modèle sur l'ensemble de test.

[11] L'utilisation d'une technique d'élagage permettrait probablement d'améliorer le taux d'erreur d'un arbre de décision mais sans doute pas de manière significative pour notre étude.

[12] type C-SVC ($svm\_type = 0$), gamma de la fonction noyau polynomiale et radiale ($gamma = 1/m$), coefficient de la fonction noyau polynomiale ($coef0 = 0.0$), coût ($C = 1.0$), tolérance du critère d'arrêt ($eps = 0.001d0$), heuristiques ($shrinking = 1$).

Plus proche voisin(s)

Pour les plus proches voisins, nous utilisons une mesure de distance euclidienne (après normalisation des valeurs de chaque pixel par l'écart-type) et le nombre optimal de plus proches voisins (évalué entre 1 et 15) est déterminé pour chaque problème selon le taux d'erreur obtenu sur l'ensemble de test.

3.3.4 Résultats

Nous avons appliqué les 7 méthodes d'apprentissage aux 4 problèmes de classification en respectant les protocoles définis précédemment.

Taux d'erreur

Les tables 3.3, 3.4, 3.5 et 3.6 résument les résultats obtenus sur chaque problème pour chaque méthode selon leurs paramètres qui ont donné les meilleurs résultats.

Méthode	**Taux d'erreur**
Arbre de décision	11.5%
3 Plus proches voisins	5.66%
Bagging ($T = 50$)	4.42%
Arbres aléatoires ($T = 100$)	3.17%
Random Forests ($T = 100$)	3.0%
Boosting ($T = 50$)	2.29%
SVMs (poly2)	1.95%
IDM, 3-NN [KGN04]	0.5%

TAB. 3.3 – Taux d'erreur pour la classification de chiffres manuscrits (MNIST).

Méthode	**Taux d'erreur**
Arbre de décision	29.25% ± 6.89
Bagging ($T = 50$)	9.5% ± 5.7
Boosting ($T = 50$)	3.75% ± 2.79
Plus proche voisin	2.25% ± 2.36
Random Forests ($T = 200$)	1.25% ± 1.68
Arbres aléatoires ($T = 500$)	1.25% ± 1.68
SVMs (linéaire)	1.25% ± 1.25

TAB. 3.4 – Taux d'erreur pour la classification de visages (ORL).

Méthode	Taux d'erreur
Arbre de décision	20.80%
Bagging ($T = 50$)	2.24%
Arbres aléatoires ($T = 500$)	1.96%
Plus proche voisin	1.94%
Random Forests ($T = 500$)	1.17%
Boosting ($T = 100$)	0.54%
SVMs (linéaire)	0.44%
Local Affine Frames [OM02b]	0.1%

TAB. 3.5 – Taux d'erreur pour la classification d'objets 3D (COIL-100).

Méthode	Taux d'erreur
Arbre de décision	89.35%
Plus proche voisin	80.79%
Bagging ($T = 50$)	73.15%
SVMs (linéaire)	71.99%
Boosting ($T = 50$)	69.44%
Random Forests ($T = 1000$)	66.90%
Arbres aléatoires ($T = 500$)	65.05%
RGB Histograms [MPV02]	0.2%

TAB. 3.6 – Taux d'erreur pour la classification de textures (OUTEX).

Comme prévu, pour chaque problème, un arbre de décision classique n'est pas satisfaisant en termes de taux d'erreur. Nous l'expliquons par la grande variance de cette méthode en particulier en présence d'un si grand nombre d'attributs (les pixels des images globales). Dans la littérature, les méthodes d'ensemble d'arbres sont bien connues pour l'amélioration qu'elles apportent par rapport à un arbre de décision seul, et ce gain en précision est confirmé par nos expériences. En effet, le Bagging et d'une façon plus impressionnante le Boosting donne des résultats beaucoup plus précis pour chaque problème. Les Random Forests et Extra-Trees sont comparables au Boosting. Les machines à support vectoriel sont très proches du Boosting et peut-être légèrement meilleures en moyenne. Enfin, l'algorithme des plus proches voisins [13] donne des résultats en deçà des méthodes les plus récentes mais plus satisfaisants que l'algorithme d'induction d'arbre de décision seul.

Les résultats obtenus sur trois des quatre bases de données (MNIST, ORL, COIL-100) sont comparables avec les techniques de l'état de l'art bien que légèrement inférieurs. Sur OUTEX, aucun des algorithmes ne donne de résultat satisfaisant.

[13] Pour cette méthode, l'utilisation de plus d'un voisin n'améliore pas les résultats si ce n'est que légèrement pour MNIST.

Nous expliquons ce mauvais résultat par la nature du problème. En effet, les motifs qui composent les textures se répètent à des positions différentes d'une image à l'autre au sein d'une même classe. Les vecteurs de pixels des images d'une même classe présentent une grande variabilité, principalement sous la forme de décalages. Les tests réalisés aux positions absolues par les modèles de classification sont alors peu concluants sur de nouvelles images. Cela d'autant plus que le nombre d'images d'apprentissage par classe disponibles lors de la phase d'apprentissage n'est que de 8, ce qui ne permet pas de couvrir la grande disparité des textures. Le faible nombre d'images d'apprentissage par rapport au grand nombre de pixels est une autre explication possible. Le nombre de tests le long d'une branche d'un arbre de décision étant dépendant du nombre d'objets d'apprentissage, lorsque les images ne sont pas nombreuses, les modèles construits individuellement sont très mauvais car le nombre de valeurs de pixels testées est faible (biais élevé).[14]

Le nombre d'arbres nécessaire pour stabiliser le taux d'erreur varie d'une méthode d'ensemble à l'autre. De manière générale, le nombre d'arbres utiles pour la méthode d'arbres aléatoires est plus élevé que les autres méthodes. Selon cette méthode, un arbre individuel étant construit aléatoirement, son taux d'erreur est plus élevé qu'un arbre construit selon les autres méthodes. Il faut alors plus d'arbres pour diminuer et stabiliser le taux d'erreur. La figure 3.8 illustre la décroissance du taux d'erreur pour les 4 problèmes en fonction du nombre d'arbres aléatoires moyennés, pour T allant de 1 à 1000.

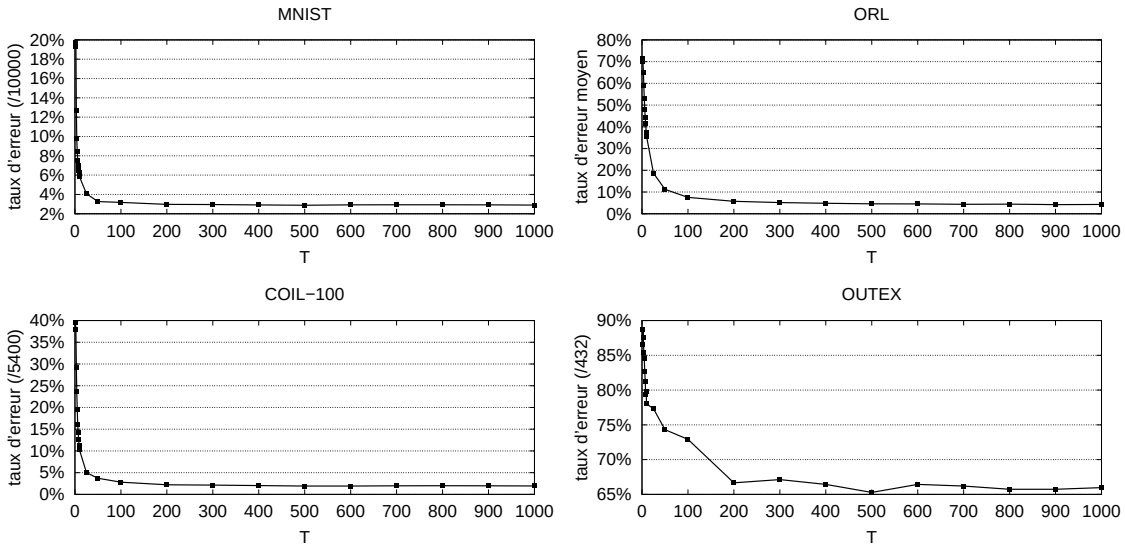

FIG. 3.8 – Influence sur le taux d'erreur de la variation du nombre d'arbres avec la méthode par arbres aléatoires, sur les 4 bases de données.

[14] Mais ce n'est pas le cas pour ORL où les visages à reconnaître sont relativement bien centrés dans l'image.

Temps d'exécution

La table 3.7 donne une idée du temps nécessaire à l'apprentissage d'un modèle de classification sur les problèmes COIL-100 et MNIST. [15] La méthode d'arbres aléatoires est sans aucun doute la plus rapide parmi les méthodes au taux d'erreur satisfaisant. Pour COIL-100, la construction de 500 arbres est d'ailleurs plus rapide que la construction d'un seul arbre de décision selon la méthode classique. Le Bagging et le Boosting, qui construisent T arbres selon l'algorithme d'induction classique sont les méthodes les plus lentes. La méthode des Random Forests est plus rapide que ces deux méthodes, mais elle est plus lente que les arbres aléatoires (surtout si le nombre d'attributs est grand). Les SVM sont très rapides sur COIL-100 mais plus lentes sur MNIST qui est composé d'un plus grand nombre d'images.

Pour l'utilisation pratique d'une telle approche, le temps de prédiction est un critère primordial. Il est négligeable avec un seul arbre de décision classique. Il est faible avec les méthodes d'ensemble d'arbres. En effet, avec ce type de méthode, la prédiction avec un arbre consiste à propager l'image le long d'une branche de l'arbre, selon les tests aux nœuds internes, jusqu'à une feuille. Chaque test implique la comparaison d'un attribut de l'image avec un seuil. L'opération est réalisée pour les T arbres de l'ensemble. Les prédictions de tous les arbres sont additionnées et le maximum est recherché parmi le vecteur de M classes. Avec les arbres aléatoires, il faut environ 20s pour prédire la classe des 5400 images de COIL-100 (soit moins de 4 msec par image), 6s pour les 10000 images de MNIST (soit moins de 1msec par image), moins de 2s pour les 432 images de OUTEX (soit environ 3 msec par image), 110msec par lot de 40 images de ORL (soit moins de 3 msec par image). Les autres méthodes d'ensemble sont au moins aussi rapides que cette méthode puisque le nombre d'arbres optimal est inférieur ainsi que la complexité de chacun d'entre eux. Avec les SVM, le temps de prédiction dépend du nombre de vecteurs de support. Cette méthode requiert 3m19s pour la prédiction des 5400 images de COIL-100, et 8m9s pour la prédiction des 10000 images de MNIST. La méthode des plus proches voisins est la plus lente de toutes puisque dans sa version de base elle consiste à comparer chaque image avec toutes les images de l'ensemble d'apprentissage. Il faut environ 20 minutes pour prédire la classe des 5400 images de COIL-100 et environ 2h30 pour les 10000 images de MNIST.

3.4 Conclusions

Dans ce chapitre, nous avons considéré la classification d'images comme un problème d'apprentissage automatique supervisé. Selon ce paradigme, un échantillon d'images pré-classifiées est donc utilisé pour inférer automatiquement un modèle de classification. La prédiction d'une nouvelle image consiste à lui appliquer ce modèle.

[15] Expériences réalisées sur un Pentium IV 2.53Ghz.

Méthode	Temps
Boosting ($T = 50$)	6h22m29s
Bagging ($T = 50$)	5h01m11s
SVMs (poly2)	28m28s
Random Forests ($T = 100$)	20m16s
Arbres aléatoires ($T = 100$)	11m39s
Arbre décision	7m17s

Méthode	Temps
Boosting ($T = 100$)	5h25m01s
Bagging ($T = 50$)	1h53m25s
Random Forests ($T = 500$)	51m34s
Arbre décision	3m08s
SVMs (linéaire)	1m02s
Arbres aléatoires ($T = 500$)	9s

TAB. 3.7 – Temps d'apprentissage pour MNIST (à gauche) et COIL-100 (à droite).

Comme certaines méthodes récentes d'apprentissage automatique peuvent traiter des problèmes dont les objets sont décrits par un grand nombre de variables, nous avons évalué l'application de ces méthodes à la représentation pixels des images. Cette approche n'implique aucune perte explicite d'information et le but de notre évaluation était de déterminer si une telle approche générique est également satisfaisante en termes de précision. Sept méthodes d'apprentissage automatique supervisé ont été comparées sur quatre problèmes de classification d'images. Des taux d'erreur satisfaisants sont atteints en comparaison avec ceux de l'état de l'art. Les temps d'apprentissage sont particulièrement rapides avec les arbres aléatoires et les temps de prédiction sont acceptables quelle que soit la méthode d'ensemble d'arbres utilisée.

Pour le quatrième problème, la classification de textures, aucune méthode d'apprentissage automatique n'offre un résultat satisfaisant. Les résultats que nous obtenons et la particularité de ce problème, où des motifs sont répétés en des positions non spécifiques à chaque classe, justifient l'introduction d'une approche locale de classification. C'est le propos du chapitre 4. En outre, nous verrons au chapitre 5 que diverses transformations peuvent perturber les images et rendre difficile leur classification par l'approche que nous avons présentée dans ce chapitre.

Chapitre 4

Ensembles d'arbres et extraction aléatoire de fenêtres

Dans ce chapitre, nous proposons et évaluons une nouvelle méthode de classification d'images basée sur la combinaison de la méthode d'ensembles d'arbres aléatoires avec une technique simple d'extraction et de prédiction de fenêtres dans les images. Au sein de la section 4.1 nous présentons la méthode selon un schéma semblable aux étapes des méthodes de l'état de l'art : la détection de régions saillantes, la description de celles-ci, et la stratégie de reconnaissance. Notre méthode est ensuite décrite de manière plus formelle (4.2) puis évaluée systématiquement sur les 4 problèmes que nous avons considérés jusqu'ici (4.3). Les résultats sont discutés (4.3.6) et une variante de la méthode est présentée (4.4) pour en accélérer les temps de prédiction.

4.1 Motivations

Les expériences du chapitre précédent ont montré que les méthodes d'apprentissages récentes travaillant avec une représentation basée sur les pixels donnaient de bons résultats sur trois problèmes sur quatre, mais de très mauvais résultats sur le quatrième problème : la classification de textures. Nous avons souligné la complexité de ce problème, en particulier expliquée par la répétition de motifs au sein des textures qui ne se retrouvent pas à des positions fixes d'une image à l'autre. Du point de vue de l'apprentissage automatique par arbres de décision, nous avons également vu qu'un ensemble d'apprentissage trop petit en nombre d'objets ne permettait pas de construire des modèles suffisamment complexes. Ces premières observations suggèrent qu'une approche locale et non globale devrait être envisagée, et cela pour la classification de textures en particulier. On peut également supposer que d'autres problèmes pourraient bénéficier d'une telle approche. Certains objets peuvent se distinguer par des caractéristiques locales plutôt que par leur forme générale. Un exemple est la reconnaissance de visages, où il peut sembler pertinent d'identifier

une personne en se concentrant sur certaines parties du visage (les yeux, la bouche, le nez, ...).

En outre, la littérature en vision par ordinateur est riche en études qui proposent des méthodes locales pour la classification d'images. Ces approches sont généralement moins sensibles aux occultations partielles, à la présence d'un fond encombré et elles sont également robustes aux translations c'est-à-dire insensibles à la position de l'objet à reconnaître dans l'image. De plus, différentes techniques permettent une sensibilité moindre à plusieurs types de transformations ou changements de configuration (échelle, orientation, point de vue). Nous reviendrons sur les problèmes de robustesse au sein du chapitre 5.

Les méthodes locales existantes procèdent généralement tout d'abord par une étape de détection de régions jugées pertinentes au sein des images. Ces zones sont ensuite chacune décrites par un ensemble de caractéristiques sous la forme d'un vecteur. Localisés et encodés à partir d'un ensemble d'images d'apprentissage, ces vecteurs constituent alors un nouvel ensemble d'apprentissage. Ces vecteurs sont alors utilisés par apprentissage pour construire un modèle de classification de régions ou, le plus souvent, ils sont stockés au sein d'une base. La même procédure de détection et de description est appliquée à chaque nouvelle image dont il faut prédire la classe. Les vecteurs ainsi extraits sont alors propagés dans les modèles construits par apprentissage ou comparés avec les vecteurs issus des images d'apprentissage stockés dans une base. La combinaison des prédictions de ces régions est utilisée pour réaliser la classification de l'image. Les méthodes locales dépendent donc principalement de trois étapes :

1. la détection de points utiles avec l'extraction de fenêtres autour de ceux-ci,
2. la description de ces fenêtres,
3. la stratégie de reconnaissance étant donné ces descriptions.

Ci-dessous, nous décrivons brièvement quelques-unes des approches existantes et nous donnons pour chaque étape un aperçu des techniques que nous proposons dans cette thèse. [MO04] a récemment proposé une revue de l'état de l'art de méthodes de reconnaissance d'objets selon ces 3 étapes. À la section 4.1.4, le tableau 4.1 résume les principales méthodes locales de la littérature, et notre approche, selon ces 3 étapes. Notre algorithme sera décrit plus formellement au sein de la section 4.2.

4.1.1 Extraction de fenêtres

État de l'art

La détection de points ou régions au sein des images a habituellement pour objet l'extraction d'informations locales discriminantes. Cet ensemble d'informations est ensuite utilisé et combiné pour la reconnaissance. Des parties sont ainsi extraites des images d'apprentissage et de test, en suivant souvent le même principe d'extraction.

Cette détection est souvent basée sur la localisation de points d'intérêts autour desquels des fenêtres sont extraites.

[SM97] détecte des points saillants à l'aide d'un opérateur de Harris. D'autres méthodes courantes sont comparées dans [MTS+04].

La méthode exposée dans [PPC01] et [PKL+02] propose d'extraire à partir des images de nombreuses fenêtres carrées, avec recouvrement éventuel, dont le contenu est discriminant. La mesure d'information utilisée pour sélectionner ces fenêtres significatives est la variance locale des valeurs de pixels au sein d'une petite fenêtre qui entoure chaque pixel. Tous les pixels d'une image sont considérés et seuls ceux qui possèdent une variance locale au-dessus d'un certain seuil sont sélectionnés. La fenêtre environnante de chaque pixel ainsi choisi est utilisée pour représenter l'image. Dans [KPNV02] qui considère la classification de caractères manuscrits, l'intensité des pixels en niveaux de gris est utilisée comme critère de sélection. Les pixels sombres (avec une faible valeur d'intensité) sont sélectionnés de manière à déterminer les points qui se trouvent sur la trace du caractère. Ici encore, la fenêtre environnante de chaque pixel retenu est utilisée.

[JD00] utilise un détecteur de points d'intérêt à l'aide d'un opérateur d'attention basé sur la symétrie. Parmi ces points sont sélectionnés ceux dont l'information au sein des sous-fenêtres avoisinantes est jugée abondante. L'écart-type des valeurs d'intensité d'une sous-fenêtre est calculé pour déterminer l'information qu'elle apporte : les sous-fenêtres dont l'écart-type est inférieur à un certain seuil sont rejetées. Un second critère est utilisé par ces auteurs pour rejeter des sous-fenêtres : celles qui ont un degré maximal de recouvrement avec les sous-fenêtres déjà choisies. Cela permet de sélectionner des sous-fenêtres à travers une portion importante de l'objet à reconnaître plutôt que des sous-fenêtres concentrées autour de quelques points d'intérêt. Cinquante points d'intérêts sont détectés au sein d'une image.

Dans [OI97], les auteurs définissent trois critères pour sélectionner les fenêtres utiles parmi l'ensemble de toutes les fenêtres possibles d'une taille donnée : la "détectabilité", l'unicité et la fiabilité.

La méthode [SSTV03] extrait des régions elliptiques construites autour d'extrema d'intensités colorimétriques.

L'algorithme proposé par [OM02b] ne se fonde pas sur la détection de points d'intérêts mais sur l'extraction de régions d'intensités homogènes. Une région est ici vue comme des composants de pixels connectés qui sont tous plus clairs ou plus sombres que tous les pixels qui se trouvent sur le contour de la région. À partir de ces régions sont alors construites de nombreuses fenêtres par l'application de divers constructeurs qui normalisent ces régions en fonction de l'orientation et de l'échelle. Cette méthode sera décrite plus amplement lors du chapitre 5 consacré à la robustesse puisqu'elle est particulièrement conçue dans ce but.

Les méthodes que nous venons de survoler permettent de détecter un nombre

réduit de points jugés pertinents au sein d'une image. Le voisinage de cet ensemble partiel de points est ensuite décrit pour la reconnaissance. Selon la technique de détection mise en place, le nombre de points retenus varie. Notamment, [JD00] souligne qu'utiliser seulement 5 points d'intérêt est suffisant dans 70% des cas de leurs expériences. La région couverte par les 5 sous-fenêtres de 10×10 qui sont extraites autour de ces points (soit 500 pixels) représente alors approximativement seulement 0.16% de chacune de leurs images en 640*480 pixels. Les trois critères de la méthode de [OI97] permettent d'extraire en moyenne environ 50 fenêtres de taille 15×15 à partir des images de taille 128×120 du problème qu'ils considèrent. Par contre, la méthode [OM02b] détecte un nombre assez important de régions et génère environ 100 000 fenêtres de taille 21×21 pour une configuration de 400 images en 128×128 pixels de la base de données COIL-100. En termes de pixels, cela représente donc plus de 6 fois ($21 * 21 * 100000 = 44100000$) le nombre de pixels de l'ensemble des images d'apprentissage ($128 * 128 * 400 = 6553600$). Le nombre de points détectés et de fenêtres extraites dépend donc des techniques utilisées mais aussi du contenu des images à reconnaître.

Notre approche

L'approche que nous proposons réalise également une extraction de fenêtres locales dans les images. Néanmoins, le principe de notre approche sera de laisser à l'algorithme d'apprentissage le soin d'apprendre à les discriminer et à en sélectionner les informations pertinentes. En effet, nos sous-fenêtres seront extraites à des positions aléatoires à partir des images, sans utiliser de critère d'information a priori et sans dépendre de la facilité de détection de points d'intérêts ni de la précision avec laquelle ils sont localisés. Le nombre de fenêtres à extraire des images d'apprentissage et de test sera un paramètre de la méthode qui pourra être fixé pour atteindre un compromis satisfaisant entre le temps d'apprentissage, le temps de prédiction, et la précision. On s'attend à ce que la précision soit une fonction croissante du nombre de fenêtres utilisées. Cette technique simple d'extraction est rapide et elle permet de représenter une portion importante des objets. Elle peut être appliquée à tous les types d'images sans adaptation importante au problème. La taille de ces fenêtres sera toutefois spécifique au problème comme nous le verrons.

4.1.2 Description de fenêtres

État de l'art

La modélisation des régions entourant les points détectés consiste à décrire chaque zone par un vecteur d'attributs, comme illustré par la figure 4.1. L'objectif premier de ce descripteur est d'être suffisamment discriminant c'est-à-dire de permettre la distinction entre les caractéristiques des différents objets à reconnaître.

Comme nous le verrons dans le chapitre suivant, un des aspects à prendre en compte lors de la conception de cette représentation est également la robustesse, aux variations de l'apparence d'un objet (changement d'illumination, de point de vue, etc.).

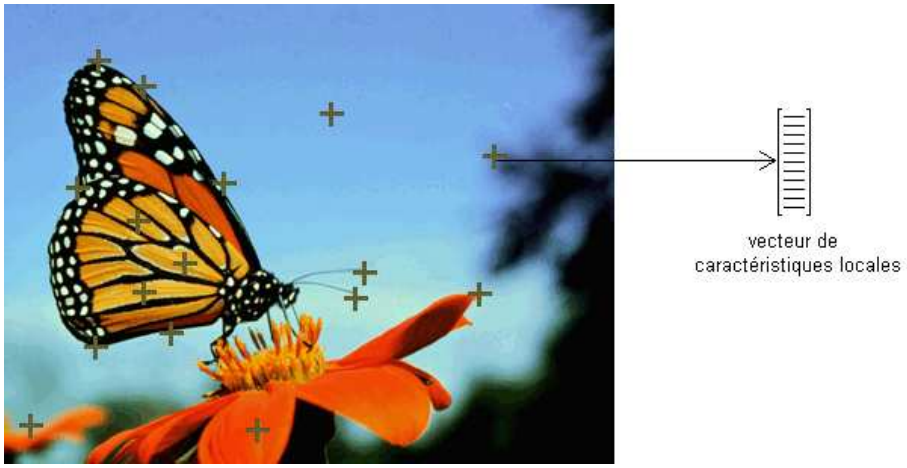

FIG. 4.1 – Représentation d'une image par détection de points d'intérêts et description par vecteurs de caractéristiques locales (adapté de [Sch96]).

[MS03] a montré l'importance de la représentation utilisée pour modéliser les régions environnantes des points d'intérêts. Cinq descripteurs différents ont été évalués. Les meilleurs résultats ont été obtenus en utilisant la description SIFT [Low99] dont chaque vecteur comporte 128 éléments calculés dans le voisinage de chaque point d'intérêt. [KS04] propose d'utiliser directement l'analyse à composantes principales dans la fenêtre centrée autour de chaque point d'intérêt. En augmentant la dimension du vecteur d'attributs (c'est-à-dire le nombre de composantes principales retenues), les auteurs obtiennent de meilleurs résultats jusqu'à une certaine taille. Une fois cette taille dépassée, la précision de l'algorithme de mise en correspondance (basé sur la distance euclidienne) se dégrade. Selon les auteurs, les vingt premières composantes sont suffisantes pour encoder le voisinage des points d'intérêt tandis que les suivantes représentent des détails qui ne sont pas utiles et qui sont potentiellement nuisibles pour la reconnaissance. On peut également expliquer ces résultats par le compromis biais/variance que nous avons mentionné au chapitre 2.

Récemment, [EC04] a comparé pour la catégorisation d'objets plusieurs descripteurs locaux : la méthode *SIFT* [Low99], les *locals JETs* (qui décrivent un point d'intérêt par l'ensemble des dérivées en ce point [SM97] [KvD87]) et les intensités des pixels d'une fenêtre de 6x6 pixels centrée autour de chaque point détecté. Il apparaît dans cette étude que la représentation par pixels égale en termes de taux d'erreur les performances de la représentation SIFT à l'aide des SVM.

Les régions extraites par [SSTV03] sont décrites par un vecteur de 9 caractéristiques basées sur les moments couleur.

La méthode [OM02b] représente les fenêtres construites consécutivement à la détection de régions par les intensités normalisées des pixels qui les composent. Ils proposent toutefois d'utiliser l'analyse à composantes principales pour réduire la dimension des fenêtres construites et ainsi accélérer le calcul de distance entre les différentes fenêtres qui sont particulièrement nombreuses avec cette méthode. Ils rapportent des résultats qui montrent l'influence sur la précision du nombre de composantes principales retenues : leur méthode est moins précise lorsque ce nombre diminue. Sur un autre problème [OM03], ils utilisent une représentation des fenêtres basée sur les coefficients DCT [1]. Les résultats obtenus sont légèrement meilleurs que ceux fournis par la comparaison directe des intensités des pixels, et, surtout, la représentation est plus compacte. Les auteurs expliquent cette précision accrue par une sensibilité moins importante de ces coefficients aux décalages de pixels résultant de la détection imprécise de régions et la construction des fenêtres à partir d'images différentes.

L'analyse en composantes principales est également appliquée aux fenêtres détectées par [PKL+02], [PPC01] et [KPNV02]. Les fenêtres projetées par cette méthode dans un espace à dimension réduite sont appelées *eigenwindows* par [OI97]. [JD00] applique au préalable un filtre gaussien à deux dimensions et une transformation de Fourier à ces fenêtres.

Les méthodes de représentation de régions sont donc nombreuses. Elles sont souvent motivées par un encodage compact des informations afin de réduire les temps de calcul et l'espace mémoire nécessaire. Elles réduisent la dimension du problème (le nombre d'attributs qui décrivent les fenêtres) car les méthodes d'apprentissage utilisées par ces auteurs ne sont pas capables de traiter des problèmes de si grandes dimensions de façon efficace et précise. Récemment, ces représentations ont également eu pour but de proposer des caractéristiques robustes aux variations de l'apparence des images. En effet, les méthodes traditionnelles de calcul de distance ou de corrélation sont très sensibles à ces variations si les intensités des pixels sont directement comparées entre images qui présentent des différences d'illumination ou de point de vue par exemple.

Notre approche

Nous pensons que les différents travaux présentés ci-dessus ne remettent pas en question l'utilisation directe des valeurs des pixels. Avec cette représentation, l'inconvénient semble plutôt être la stratégie de reconnaissance habituellement utilisée qui est basée sur une recherche du plus proche voisin au sens d'une mesure de distance euclidienne (voir section 4.1.3) : la mesure de distance donne la même importance à tous les attributs sans apprendre à utiliser les plus discriminants. De plus, elle est très sensible aux décalages (résultant de problème de localisation ou

[1] transformation en cosinus discrète ou "Discrete Cosine Transform" utilisée aussi en compression d'images.

d'alignement des points d'intérêts). En apprentissage, il est connu qu'en présence d'un grand nombre de variables, ces méthodes basées sur la distance souffrent la plupart du temps d'une variance importante. On parle du problème de "curse of dimensionality" car le nombre d'exemples nécessaires a tendance à croître exponentiellement avec le nombre de dimensions [DHD00]. Il est donc logique de réduire la dimension des fenêtres si ces méthodes de comparaison sont utilisées, mais d'autres méthodes d'apprentissage pourraient être utilisées.

Étant donné les résultats encourageants obtenus avec une approche globale utilisant directement les informations des pixels, au chapitre précédent, nous continuerons à utiliser la représentation par valeurs de pixels. Les sous-fenêtres extraites aléatoirement seront donc décrites par les valeurs des pixels qui les composent. Cette représentation est potentiellement la plus discriminante puisque aucune information n'est écartée, aucune opération de réduction de dimension n'étant appliquée aux fenêtres. Comme nous l'avons déjà dit, cette représentation est également indépendante du problème.

4.1.3 Stratégie de reconnaissance

État de l'art

Une fois les régions détectées et décrites, la troisième étape consiste à exploiter les caractéristiques locales extraites pour la classification. Il s'agit, étant donné les fenêtres extraites à partir de l'image de test, de prédire sa classe en les comparant à l'information fournie par l'ensemble de fenêtres extraites préalablement à partir des images d'apprentissage. Comme nous l'avons dit, deux approches sont envisageables. L'approche par apprentissage consiste à construire un modèle à partir des fenêtres d'apprentissage puis à utiliser ce modèle pour classer les nouvelles fenêtres de test. La seconde approche, par mise en correspondance, consiste à utiliser directement les représentations des fenêtres d'apprentissage, sans construction de modèle, pour la comparaison des fenêtres. Cette dernière approche est la plus couramment utilisée.

Pour réaliser cette classification par appariement, de nombreux travaux utilisent une mesure de distance entre les fenêtres, suivie d'une étape de combinaison des votes. En effet, une distance peut être calculée pour déterminer la similarité entre chaque paire de fenêtres. Pour chaque fenêtre de l'image de test, il convient de parcourir la liste des fenêtres issues des images d'apprentissage et d'en calculer la distance pour chaque paire. La classe de la fenêtre d'apprentissage la plus proche peut être attribuée à cette fenêtre de test. Pour combiner les résultats, on peut alors attribuer à l'image de test la classe qui aura été la plus souvent attribuée à ses fenêtres. Autrement dit, le score d'une classe équivaut au nombre de correspondances établies entre les fenêtres de l'image de test et les fenêtres d'apprentissage de cette classe d'images. Suivant cette approche, les différentes méthodes de la littérature

se distinguent par la mesure de distance utilisée, la manière de combiner ces mesures, et/ou par la façon dont sont indexées les fenêtres. Les mesures de distance couramment utilisées sont la distance euclidienne, de Mahalanobis et la distance tangente [SLDV00] [2]. Parmi les travaux cités précédemment, [OM02b], [KS04] utilisent la distance euclidienne, [SM97] et [SSTV03] utilisent la distance de Mahalanobis, [PKL+02] utilise la distance tangente. Différentes méthodes de combinaison des mesures sont envisageables. Pour chaque fenêtre de test, seule la classe du voisin le plus proche peut être retenue ou bien on peut retenir la classe majoritaire parmi les classes prédites par plusieurs voisins. La classe la plus fréquemment retenue est ensuite attribuée à l'image. Il est également possible de retenir pour chaque fenêtre le vecteur des classes des voisins les plus proches plutôt que la classe majoritaire uniquement et d'additionner tous les vecteurs correspondant à l'ensemble des fenêtres de l'image de test. Pour parcourir la base de données de fenêtres d'apprentissage, autrement dit pour chercher les plus proches voisins d'une fenêtre de test, divers travaux utilisent des structures arborescentes [PPC01] ou réalisent des approximations [SSTV03] pour en accélérer l'accès, car le nombre de fenêtres à comparer est parfois très grand. [OM02b] se distingue des autres méthodes par le fait que toutes les fenêtres ne sont pas comparées entre elles. En effet, chaque fenêtre de l'image de test est uniquement comparée avec les fenêtres d'apprentissage qui ont été obtenues par l'application du même constructeur. Cette méthode conserve donc, en plus des fenêtres, une série de paramètres qui permettent d'identifier le procédé utilisé pour la génération de chacune de ces fenêtres.

Très récemment, quelques travaux ont combiné des approches locales aux méthodes nouvelles d'apprentissage plutôt qu'à la méthode des plus proches voisins. [EC04] propose de combiner plusieurs type de représentations locales avec les SVM pour la catégorisation et il en est de même de [CWN04]. [WCG03] a comparé l'utilisation de SVM entre une représentation globale par pixels (comme nous l'avons fait au chapitre précédent) et une représentation locale par un vecteur de 9 caractéristiques (JETs) calculées autour de points d'intérêt.

Notre approche

Lors de notre première étude où nous avons utilisé une représentation globale par pixels des images (chapitre 3), les arbres extrêmement aléatoires se sont montrés très rapides (pour la phase d'apprentissage en particulier) et ils offraient une précision comparable aux autres méthodes récentes d'ensembles d'arbres et aux SVM. La méthode des plus proches voisins, basée sur une distance euclidienne, offrait des résultats moins satisfaisants. Nous utiliserons donc les ensembles d'arbres aléatoires pour évaluer notre approche locale. Un modèle sera donc construit à partir des vecteurs de pixels des fenêtres extraites aléatoirement à partir des images d'ap-

[2] Cette mesure de distance est construite de manière à être insensible à un ensemble de transformations.

prentissage. Ce modèle a comme but de distinguer les fenêtres issues d'images de différentes classes. Pour la prédiction, la règle de décision du modèle sera appliquée à chaque fenêtre de l'image de test. Une couche supplémentaire combinera les classifications des différentes fenêtres qui constituent une image pour prédire la classe de celle-ci.

4.1.4 Résumé

Le tableau 4.1 résume les principales méthodes de classification de la littérature selon les 3 étapes principales.

Référence	**Détection**	**Description**	**Reconnaissance**
[SM97]	opérateur de Harris	local jets, invariants	dist. Mahalanobis
[PPC01], [PKL+02]	variance locale	PCA	dist. tangente
[OI97]	détectabilité, unicité, fiabilité	PCA	norme L1
[SSTV03]	extremas locaux	9 moments couleur	dist. Mahalanobis
[OM02b], [OM03]	régions homogènes	pixels/PCA/DCT	dist. euclidienne
Notre approche	fenêtres aléatoires	valeurs de pixels	ens. arbres aléatoires

TAB. 4.1 – Résumé des principales méthodes locales de la littérature.

4.2 Description de la méthode proposée

Nous venons de motiver une approche locale de classification d'images utilisant l'apprentissage automatique pour classer des fenêtres extraites aléatoirement à partir des images. Dans cette section nous la présentons de manière plus formelle. Cette approche a été proposée pour la première fois dans [MGW03] et ensuite publiée dans [MGV+03] puis [MGPW04]. Nous utilisons donc une extraction aléatoire de fenêtres, une description par valeurs d'intensités des pixels, et une stratégie de reconnaissance qui utilise la méthode des ensembles d'arbres de décision aléatoires.

4.2.1 Phase d'apprentissage

La première étape de notre nouvelle approche est de générer à partir de l'échantillon d'apprentissage d'images, un nouvel échantillon d'apprentissage composé de fenêtres. C'est à partir de ce nouvel échantillon que sera construit un modèle de classification de fenêtres. Chaque fenêtre de ce nouvel échantillon est décrite par tous les pixels qui la constituent et par la classe de l'image à partir de laquelle elle a été extraite. La figure 4.2 illustre cette phase [3].

[3] L'exemple utilise $N_w = 15$ fenêtres extraites et $T = 5$ arbres pour un problème à $M = 3$ classes.

Algorithme d'apprentissage de base

- Pour une taille de fenêtre fixée $w_1 \times w_2$, on extrait aléatoirement un certain nombre N_w de fenêtres à partir des images de l'ensemble d'apprentissage. On associe à chacune de ces fenêtres la classe de l'image dont elle est extraite.
- On construit un ensemble de T arbres aléatoires à partir du nouvel ensemble d'apprentissage constitué des N_w fenêtres décrites par les valeurs des $w_1 \times w_2$ pixels qui les composent.

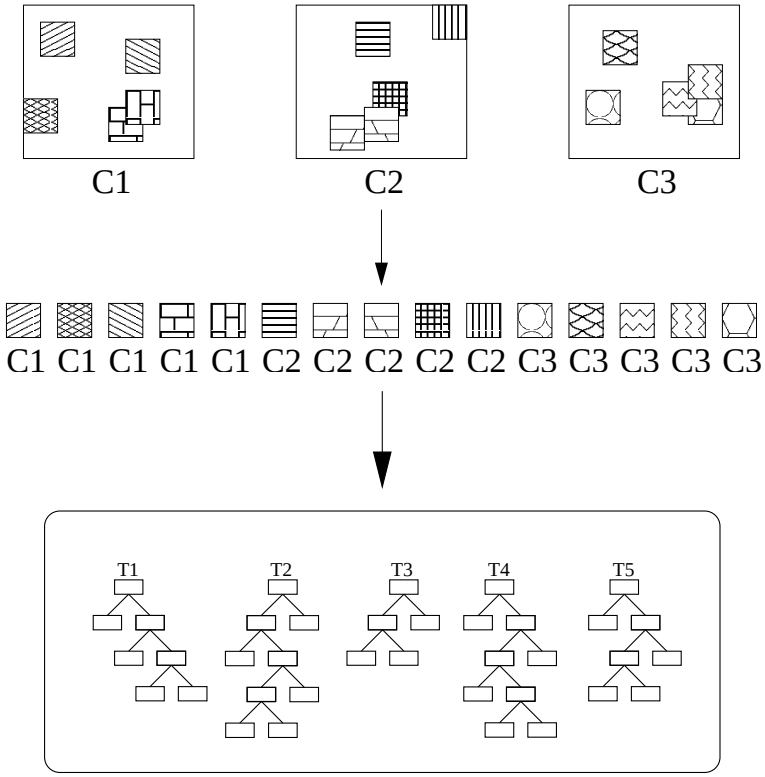

FIG. 4.2 – Phase d'apprentissage : extraction aléatoire de fenêtres à partir des images originales (premier et deuxième niveau), construction d'un ensemble d'arbres à partir des fenêtres (troisième niveau).

L'extraction aléatoire peut être réalisée en tirant aléatoirement à N_w reprises une image de l'ensemble d'apprentissage et en y prenant aléatoirement une fenêtre. Ou bien, on peut pour chaque image originale de l'ensemble d'apprentissage lui extraire un certain nombre (égal d'une image à l'autre) de fenêtres pour atteindre le total N_w de fenêtres de manière plus équilibrée. Nous n'avons pas remarqué d'effet sur le taux d'erreur selon ces différentes méthodes. La sélection aléatoire d'une fenêtre au sein d'une image est réalisée de manière à extraire des fenêtres de la taille fixée sans coupure aux bords. C'est-à-dire qu'étant donnée la taille de la fenêtre, on détermine aléatoirement la position du coin supérieur gauche de la fenêtre de manière à ce que

la fenêtre soit entièrement comprise dans l'image. Autrement dit, ce coin supérieur gauche est tiré aléatoirement entre les positions horizontales 0 et $W_x - w_1 - 1$ et les positions verticales 0 et $W_y - w_2 - 1$. Nous ne mémorisons pas pour une même image les positions déjà tirées, les fenêtres choisies pour une même image peuvent donc selon ce processus aléatoire se superposer. Il est toutefois raisonnable de penser que les fenêtres extraites de la sorte en nombre suffisant couvrent une partie importante des images originales. Cette couverture dépend aussi de la taille des fenêtres.

Nous réaliserons le choix optimal des paramètres N_w, w_1 et w_2 de manière empirique pour les différents problèmes. Le choix du nombre N_w dépend évidemment du nombre d'images originales dont on dispose et d'une certaine façon des dimensions w_1 et w_2. Bien que nous n'ayons pas réellement mis en place un méthode pour déterminer automatiquement ces dimensions, celles-ci peuvent être déterminées en évaluant le taux d'erreur résultant du modèle sur l'ensemble de test par validation croisée. L'influence de ces paramètres sur le temps d'apprentissage et le temps de prédiction, découlant de la complexité des arbres, n'est pas évident et est à étudier.

Dans notre cas, le modèle construit sera un ensemble d'arbres aléatoires. Le nombre d'arbres, T, reste un paramètre de cette nouvelle méthode. Ce nombre influence directement le temps d'apprentissage et le temps de prédiction. Son influence sur le taux d'erreur est comme précédemment à étudier pour chaque problème. Comme nous l'avons vu au chapitre 2, la complexité de l'algorithme de construction d'un arbre aléatoire est liée au nombre d'images (ici de fenêtres) dans l'échantillon d'apprentissage : $O(N_w log N_w)$.

4.2.2 Phase de prédiction d'une nouvelle image

Le modèle a été construit à partir de fenêtres de taille $w_1 \times w_2$ et permet donc de classer des nouvelles fenêtres de cette taille. Pour la classification d'une image complète, il faut donc lui extraire des fenêtres de cette taille, les propager dans le modèle, et combiner les prédictions des différentes fenêtres.

Une approche directe, qui est également celle de [HF00], est d'extraire toutes les fenêtres possibles de l'image de test en la parcourant de haut en bas et de gauche à droite, par un décalage d'un pixel à la fois, d'abord sur une même ligne, puis en passant à la ligne suivante, comme illustré à la figure 4.3. Lors de la phase d'apprentissage, toutes les fenêtres des toutes les images n'ont pas été utilisées car le nombre de fenêtres qui en découlerait pourrait devenir prohibitif. Un tirage aléatoire en a réduit le nombre. Lors de la phase de prédiction, nous considérerons dans l'algorithme de base toutes les fenêtres d'une image, une variante proposera d'en extraire un nombre réduit au sein de la section 4.4.

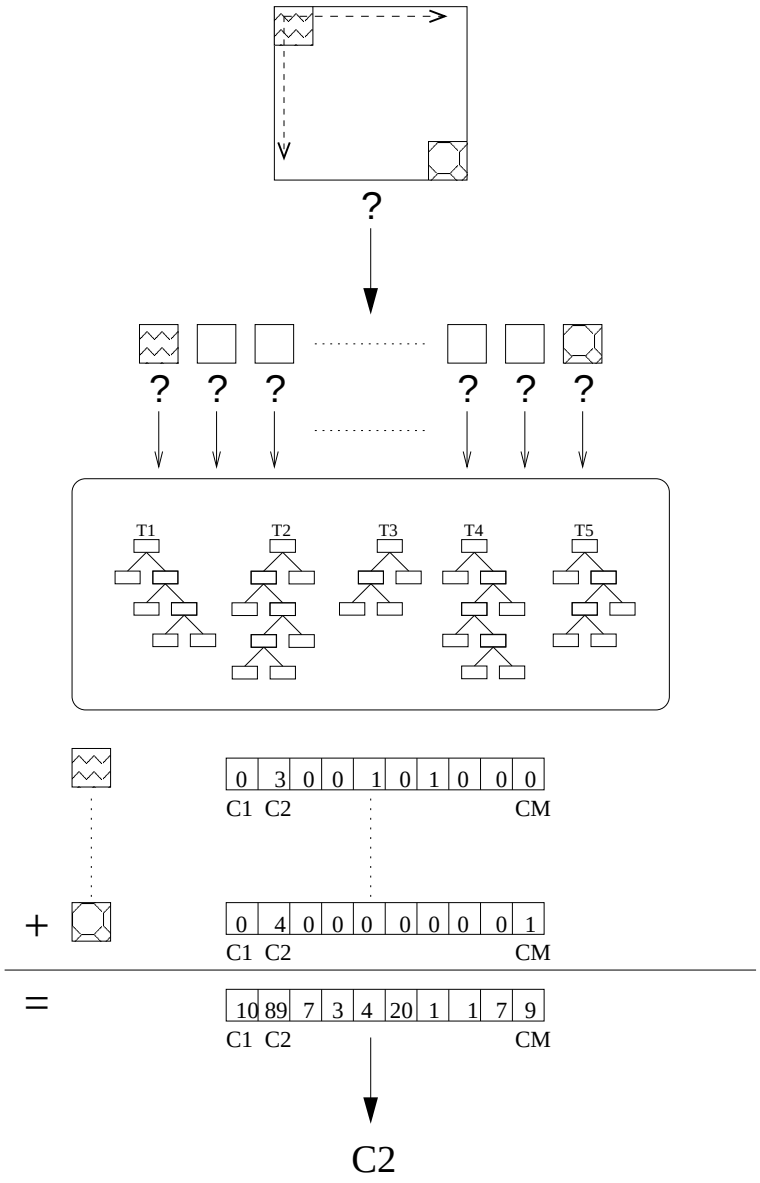

FIG. 4.3 – Phase de prédiction : parcours de l'image à classer pour l'extraction de toutes ses fenêtres (premier et deuxième niveau), propagation de chacune des fenêtres dans l'ensemble d'arbre (troisième niveau), addition des vecteurs de probabilités de M classes (quatrième niveau), prédiction de la classe par recherche du maximum (cinquième niveau)

Algorithme de prédiction de base

- On extrait toutes les fenêtres possibles de taille $w_1 \times w_2$ à partir de l'image qu'on désire classer,
- On propage chaque fenêtre dans l'ensemble d'arbres aléatoires,

– On combine les classes attribuées aux sous-fenêtres par l'ensemble d'arbres aléatoires et on associe à l'image la classe majoritaire parmi celles-ci.

De manière à ne pas perdre d'information lors de la combinaison des classes attribuées aux sous-fenêtres, nous combinons les vecteurs de distributions de classes pour chaque fenêtre plutôt que de retenir uniquement la classe majoritaire pour chacune des fenêtres. Autrement dit, nous additionnons les votes assignés aux fenêtres par les arbres (T votes par fenêtre) et ce sur l'ensemble des fenêtres de l'image. La classe majoritaire parmi les $T \times (W_x - w_1) \times (W_y - w_2)$ votes est alors associée à l'image. Avec un arbre aléatoire, la prédiction d'une fenêtre requiert en moyenne $logN_w$ tests qui consistent chacun à comparer la valeur d'un pixel à un seuil, comme nous l'avons vu au chapitre 2.

4.2.3 Paramètres de la méthode

Par rapport à la méthode de construction d'arbres aléatoires sur les images globales, les paramètres additionnels sont la taille des fenêtres $w_1 \times w_2$ et le nombre N_w de sous-fenêtres extraites lors de la phase d'apprentissage. L'influence du nombre d'arbres T de l'ensemble est à étudier à nouveau dans ce contexte. Bien qu'a priori il n'est pas évident de déterminer les paramètres optimaux pour chaque problème étant donné leur caractère distinct, on s'attend à ce que la précision (le taux d'erreur) soit une fonction croissante (décroissante) de T et N_w.

4.3 Évaluation

Dans cette section, nous évaluons la nouvelle méthode proposée sur les 4 problèmes pour lesquels les méthodes d'apprentissage ont déjà été évaluées au sein de la section 3.3. Dans [MGW03] cette approche locale a été comparée aux ensembles d'arbres aléatoires appliqués aux images globales pour les 4 problèmes MNIST, COIL-20, ORL et OUTEX. Dans [MGV$^+$03] elle a été comparée aux mêmes méthodes que celles utilisées dans le chapitre précédent, à l'exception de la méthode des plus proches voisins, sur les problèmes MNIST, COIL-100, ORL et OUTEX.

Les résultats principaux sont résumés en fin de section par les tableaux 4.2 et 4.3, page 67.

Dans cette section nous étudions empiriquement l'influence des paramètres pour chaque problème. Nous représentons l'influence de la dimension des fenêtres sur le nombre d'erreurs, le temps de prédiction d'une image, le temps d'apprentissage et la complexité du modèle construit. En fonction de la dimension optimale des fenêtres observée pour chaque problème, nous traçons ensuite l'influence sur le taux d'erreur du nombre de fenêtres d'apprentissage et du nombre d'arbres de l'ensemble. Dans

notre évaluation, nous allons nous restreindre à des valeurs égales de w_1 et w_2 c'est-à-dire à des fenêtres carrées.

4.3.1 MNIST

Pour ce problème de classification de chiffres manuscrits, la taille optimale des fenêtres, 24×24 pixels, est légèrement inférieure à la taille des images globales, 28×28 pixels, comme le montre la figure 4.4 (en haut à gauche). L'amélioration du taux d'erreur est légère par rapport aux arbres aléatoires sur les images globales : 2.63% contre 3.17% d'erreurs. Comme on le constate à la figure 4.6, avec ces paramètres, on réalise principalement uniquement une augmentation de l'ensemble d'apprentissage avec des chiffres légèrement décalés. À partir d'une taille de fenêtres de 12×12 pixels la précision est satisfaisante. En deçà, de trop petites fenêtres ne donnent pas de résultat convenable car il est alors plus difficile de distinguer des régions ainsi extraites d'un chiffre à l'autre, comme le montre la figure 4.7 pour des tailles de fenêtres de 8×8 pixels. En outre, pour ce problème, l'échantillon d'apprentissage est très grand et les images sont bien centrées et dimensionnées. Le nombre de variables (les valeurs de pixels) est inférieur au nombre d'images qui constituent l'échantillon d'apprentissage. L'augmentation de l'échantillon d'apprentissage par extraction de fenêtres n'offre alors pas d'amélioration nette de la précision pour ce problème.

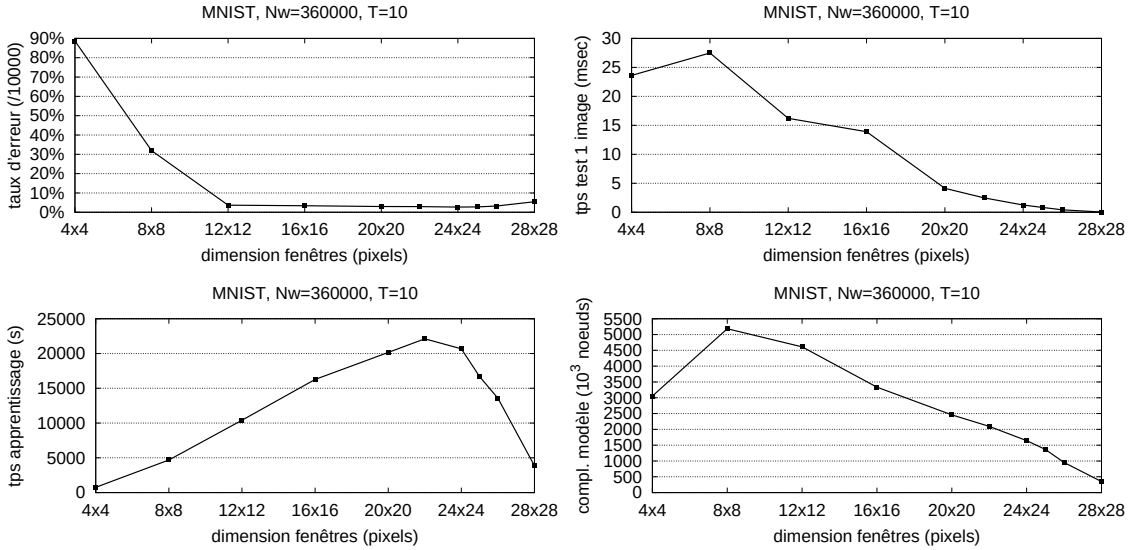

FIG. 4.4 – MNIST : influence de la dimension des fenêtres sur le nombre d'erreurs, le temps de prédiction d'une image, le temps d'apprentissage, la complexité du modèle construit.

Le temps pour prédire la classe d'une nouvelle image est dépendant du nombre de fenêtres qui lui sont extraites. En outre, comme chaque fenêtre est propagée dans

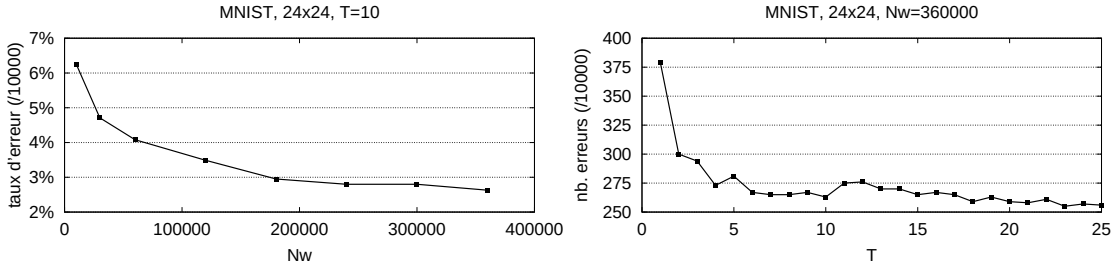

FIG. 4.5 – MNIST : influence sur le nombre d'erreurs de la variation du nombre de fenêtres d'apprentissage avec 10 arbres (à gauche) et variation du nombre d'arbres avec 360000 fenêtres (à droite), avec une dimension de fenêtres de 24×24.

FIG. 4.6 – MNIST : Quelques fenêtres de taille 24×24 pixels extraites des images de tailles 28×28, pour 3 chiffres, à raison de 4 fenêtres par objet.

l'ensemble d'arbres, le temps de prédiction est aussi dépendant de la complexité du modèle c'est-à-dire du nombre de tests (aux nœuds internes des arbres de l'ensemble) auxquels la fenêtre est soumise. Chaque test au nœud d'un arbre implique la comparaison d'une composante de pixel de la fenêtre à un seuil de discrimination. Le temps requis pour combiner les prédictions des différentes fenêtres est supposé négligeable puisqu'il s'agit simplement d'additionner le vecteur de probabilités pour chaque fenêtre (la taille de ce vecteur étant le nombre M de classes du problème traité, dans ce cas 10 chiffres). Le graphique montre donc que le temps de prédiction est principalement une fonction décroissante de la taille des fenêtres et de la complexité du modèle : plus les fenêtres sont grandes, moindre est le nombre d'entre elles à classer, puisque nous balayons complètement l'image [4]. Pour la taille de fenêtre optimale, 24×24 pixels, il faut moins de 2 msec pour prédire la classe d'une nouvelle image. Selon ces paramètres, le test d'une image implique la propagation de 16 fenêtres dans l'ensemble d'arbres, puis l'addition des 16 vecteurs de probabilités et la détermination du maximum parmi les $M = 10$ sommes obtenues.

Le temps de construction de modèle est lui aussi dépendant de la taille des fenêtres mais de manière moins directe. Avec des fenêtres de petites tailles, comme

[4] Néanmoins, le temps nécessaire à la prédiction pour une taille de 8×8 pixels est supérieur à celui avec fenêtres de 4×4 pixels. Dans ce cas, nous observons que le modèle construit avec fenêtres de 8×8 pixels est un peu moins que 2 fois plus complexe qu'avec fenêtres de 4×4 pixels. Le nombre moins important de fenêtres à balayer en 8x8 ne compense pas cette complexité plus grande.

FIG. 4.7 – MNIST : Une fenêtre de taille 8×8 (ici agrandies pour plus de visibilité) pixels extraite pour chacun des chiffres 0, 6 et 8.

4×4 pixels, le temps de construction est faible car le nombre maximum d'attributs, à considérer pour l'élaboration d'un test au sein d'un nœud, est petit. En effet, même si l'algorithme ne trouve pas d'attribut suffisamment discriminant, la liste d'attributs candidats à considérer est de taille réduite. L'algorithme, ayant rapidement parcouru toute la liste, renvoie le meilleur test obtenu. Le temps de construction augmente alors jusqu'à une certaine taille de fenêtres puisque la liste d'attributs s'agrandit, mais diminue ensuite car il devient plus facile de trouver un test discriminant parmi un ensemble large d'attributs. Enfin, avec des fenêtres de 28×28 pixels, plusieurs fenêtres parmi les $N_w = 360000$ sont identiques et la complexité des arbres est alors plus faible.

La figure 4.5 illustre la décroissance du taux d'erreur en fonction du nombre de fenêtres d'apprentissage pour une taille de fenêtres et un nombre d'arbres fixés d'une part (à gauche), et d'autre part (à droite) en fonction du nombre d'arbres pour une taille et un nombre fixés de fenêtres. Sur ce problème, la taille importante de l'échantillon de base (60000 images) motive un nombre élevé de fenêtres. Une valeur supérieure à 360000 pourrait vraisemblablement encore légèrement améliorer la précision mais pourrait devenir rédhibitoire en termes de temps de calcul, en particulier pour l'apprentissage du modèle.[5] Le taux d'erreur ne diminue pas considérablement lorsqu'on utilise plus de 10 arbres : 7 images supplémentaires sur 10000 sont correctement classées (2.56% d'erreur) avec $T = 25$ arbres par rapport à l'utilisation de 10 arbres (2.63% d'erreur).

En conclusion, notre nouvelle approche locale améliore légèrement la précision des arbres aléatoires sur ce problème mais elle reste toutefois légèrement inférieure au Boosting et aux SVM appliqués aux images globales au sein du chapitre précédent. Notre méthode reste aussi en-dessous des performances de [KGN04] qui a récemment obtenu un taux d'erreur de 0.5%.

4.3.2 ORL

Pour la classification de visages, nous avons obtenu dans [MGV+03] un taux d'erreur de 0.5% $\pm$ 1.0 par validation croisée avec notre méthode locale contre 1.25% $\pm$ 1.68 avec les arbres aléatoires sur les images globales, selon le même protocole que celui utilisé précédemment. Ce protocole nous a permis d'ordonner les performances

[5]Pour ce problème, le nombre maximal de fenêtres différentes de tailles 24×24 pixels que l'on peut extraire parmi les 60000 images d'apprentissage est de 960000.

des différentes méthodes mais il n'est pas adéquat pour effectuer une analyse plus fine de l'influence des paramètres de notre nouvelle méthode sur le taux d'erreur. C'est pourquoi nous utilisons dans cette section le protocole que nous avions déjà utilisé dans [MGW03] et [MGPW04]. C'est-à-dire que nous moyennons les résultats de 100 exécutions en utilisant pour chacune d'elle 5 images distinctes tirées aléatoirement par classe pour chaque ensemble d'apprentissage et l'autre moitié des images pour chaque ensemble de test. Au total, cela représente un taux d'erreur moyenné sur 20000 prédictions.

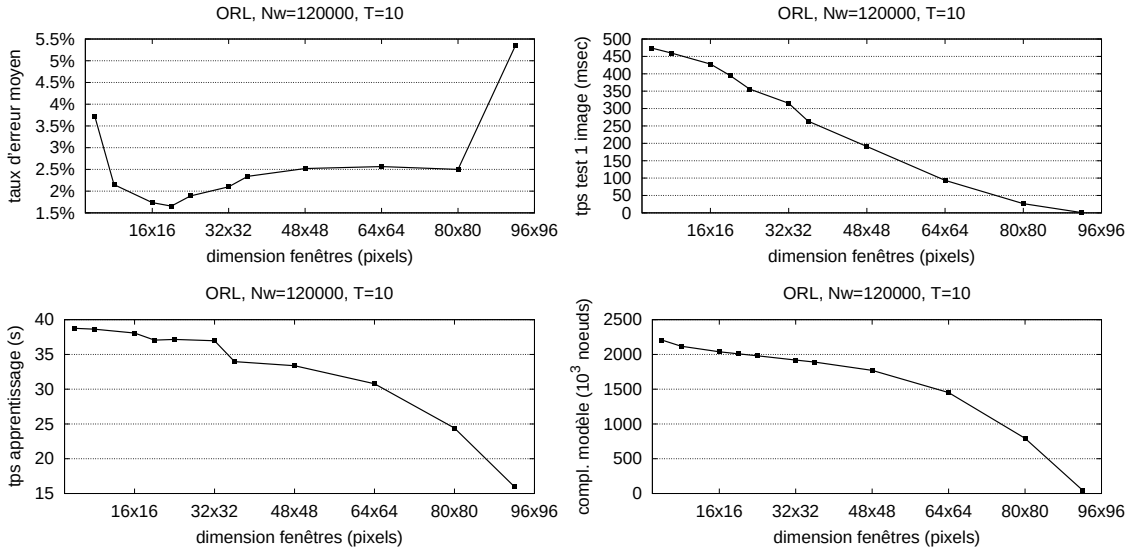

FIG. 4.8 – ORL : influence de la dimension des fenêtres sur le nombre d'erreurs, le temps de prédiction d'une image, le temps d'apprentissage, la complexité du modèle construit.

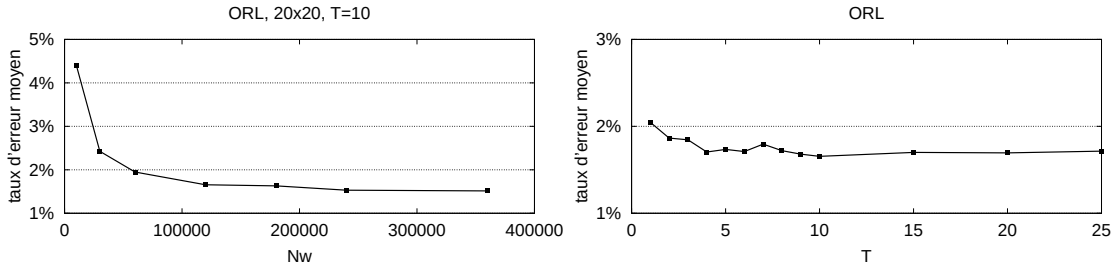

FIG. 4.9 – ORL : influence sur le nombre d'erreurs de la variation du nombre de fenêtres d'apprentissage avec 10 arbres (à gauche) et variation du nombre d'arbres avec 360000 fenêtres (à droite), sur le problème ORL avec une dimension de fenêtres de 24×24.

Selon ce protocole, nous obtenons un taux d'erreur de $4.56\% \pm 1.43$ avec la méthode d'ensemble d'arbres aléatoires sur les images globales. Avec l'approche locale,

la taille de fenêtres qui donne les meilleurs résultats est de 20×20 pixels (pour un nombre d'arbres $T = 10$ et un nombre de fenêtres d'apprentissage $Nw = 120000$) comme le montre la figure 4.8, avec un taux d'erreur de $1.66\% \pm 1.08$. [6] On peut sans doute expliquer cette amélioration en considérant que des fenêtres de cette taille permettent de capturer des caractéristiques locales des visages comme les yeux, la bouche ou encore le nez qui sont plus discriminantes entre les différentes personnes et moins variables entre différentes images d'une même personne photographiées dans des conditions changeantes. Des fenêtres trop petites ou trop grandes ne permettent pas d'atteindre une si bonne précision. Quelques fenêtres de taille optimale sont illustrées par la figure 4.10.

FIG. 4.10 – ORL : Quelques fenêtres de taille 20×20 pixels extraites des images originales de tailles 92×112, pour 3 individus, à raison de 4 fenêtres par individu.

Le temps de prédiction est ici encore principalement dépendant de la taille des fenêtres et de la complexité du modèle. Il faut environ 354 msec pour prédire la classe d'une nouvelle image en utilisant la taille de fenêtres optimale de 20×20 pixels. Une prédiction implique la propagation d'environ 6500 fenêtres dans l'ensemble de 10 arbres.

Les courbes du temps d'apprentissage et de la complexité du modèle montrent une décroissance selon la taille des fenêtres. Cela est principalement expliqué par le seuil de 0.01 de rejet des tests aléatoires.[7] Avec ce seuil, la plupart des tests sont acceptés, et plus la taille des fenêtres augmente, moins le nombre de tests à effectuer pour distinguer les classes des fenêtres des images d'apprentissage est important.

La figure 4.9 illustre la décroissance du taux d'erreur en fonction du nombre de fenêtres d'apprentissage pour une taille de fenêtres et un nombre d'arbres fixés d'une part (à gauche), et d'autre part (à droite) en fonction du nombre d'arbres pour une taille et un nombre fixés de fenêtres. Un nombre de 120000 fenêtres d'apprentissage est suffisant pour atteindre le taux d'erreur présenté. Ici encore, augmenter ce nombre

[6] Ces résultats résultent d'expériences avec un seuil s_{th} de 0.01 plutôt que 0.1 au sein de l'algorithme d'arbres aléatoires, cela pour accélérer le temps de construction étant donné que nous réalisons 100 tirages différents.

[7] Si nous traçons la courbe pour un seuil de 0.1, nous obtenons la même allure que celle pour le problème MNIST : augmentation du temps, passage par un maximum puis diminution jusqu'à la taille globale.

améliore légèrement le taux d'erreur. Utiliser plus de 10 arbres n'est à nouveau pas très intéressant puisque le taux d'erreur ne diminue pas sensiblement. En fait, utiliser un seul arbre permet déjà d'atteindre un taux d'erreur moyen intéressant de $2.05\% \pm 1.3$ pour ce problème. Une conclusion directe de l'observation de ces deux courbes pourrait consister en l'utilisation de moins d'arbres et moins de fenêtres, afin de réduire la complexité du modèle et donc le temps de prédiction d'une nouvelle image, en conservant une précision voisine.

En conclusion, le résultat que nous obtenons sur ce problème est considérablement meilleur que celui des arbres aléatoires appliqués aux images globales. Comme nous l'avons déjà souligné, la comparaison avec d'autres méthodes de la littérature n'est pas aisée. Très récemment, [Rav04] a utilisé le même protocole et selon les méthodes utilisées obtient un taux de reconnaissance moyen variant de 95% à 98%.

4.3.3 COIL-100

Pour cette base de données d'objets 3D dont on utilise un ensemble de 1800 images régulièrement espacées pour l'apprentissage, et les autres 5400 images pour le test, 16×16 pixels est la taille de fenêtres qui donne les meilleurs résultats sur l'ensemble de test avec 20 erreurs (0.37%) sur 5400 images de test pour un nombre d'arbres $T = 10$ et un nombre de fenêtres d'apprentissage $Nw = 120000$. La figure 4.11 illustre ces résultats et la figure 4.13 montre quelques fenêtres de cette taille. Le taux d'erreur obtenu avec cette dimension de fenêtres est donc bien meilleur que celui résultant de l'application des arbres aléatoires sur les images globales où le nombre d'erreurs était de 106 (1.96%). [8]

Les courbes du temps de prédiction et du temps d'apprentissage peuvent être expliquées de manière identique aux courbes des problèmes précédents. Pour la taille de fenêtre optimale, 16×16 pixels, le temps de prédiction d'une image est approximativement de 14 msec, ce qui requiert la classification et la combinaison de 256 fenêtres.

La figure 4.12 représente (à gauche) l'évolution du nombre d'erreurs en fonction du nombre de fenêtres d'apprentissage pour un nombre d'arbres fixé ($T = 10$) et une taille de fenêtres de 16×16. On voit qu'au-delà de $Nw = 120000$ l'amélioration n'est plus évidente. À droite, on observe le nombre d'erreurs en fonction du nombre d'arbres construits pour un nombre de fenêtres fixé ($Nw = 120000$) et une taille de fenêtres de 16×16. On voit qu'il n'est pas nécessaire d'augmenter le nombre d'arbres au-delà de $T = 10$, et, en fait, le taux d'erreur est identique avec $T = 5$ arbres et ne varie que d'une image sur 5400 entre ces deux valeurs. Il est donc envisageable, sans perte importante de précision, de réduire pour ce problème également

[8] Lorsque la taille des fenêtres est identique à la taille des images globales, puisque nous construisons seulement 10 arbres, nous obtenons un taux d'erreur supérieur à celui de l'approche globale où 500 arbres étaient construits.

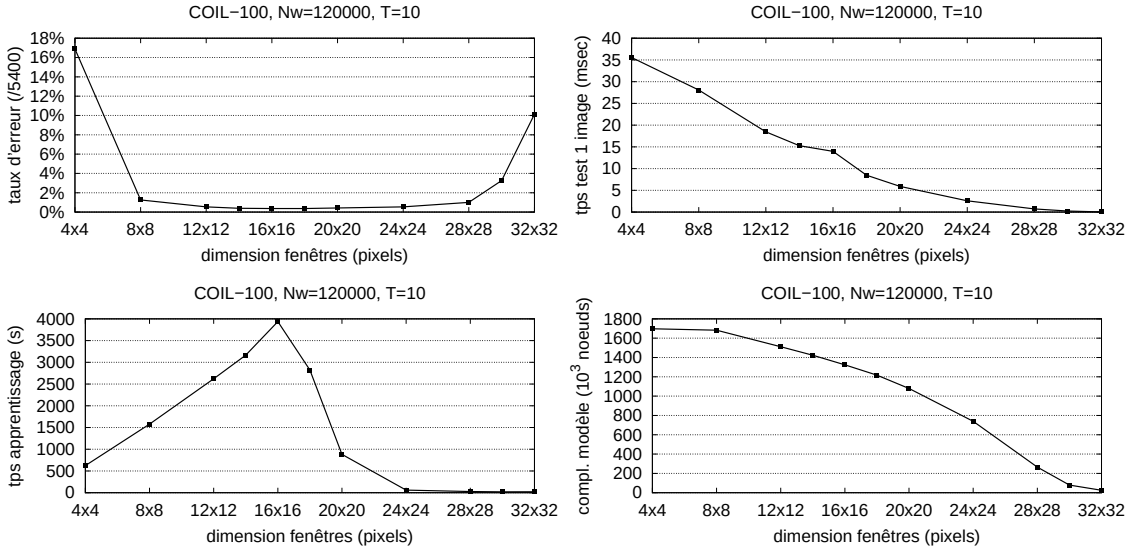

FIG. 4.11 – COIL-100 : influence de la dimension des fenêtres sur le nombre d'erreurs, le temps de prédiction d'une image, le temps d'apprentissage, la complexité du modèle construit.

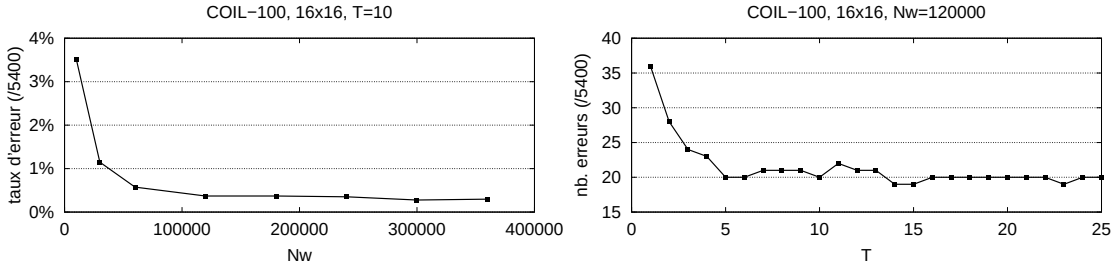

FIG. 4.12 – COIL-100 : influence sur le nombre d'erreurs de la variation du nombre de fenêtres d'apprentissage avec 10 arbres (à gauche) et variation du nombre d'arbres avec 120000 fenêtres (à droite), sur le problème COIL-100 avec une dimension de fenêtres de 16×16.

FIG. 4.13 – COIL-100 : Quelques fenêtres de taille 16×16 pixels extraites des images de tailles 32×32, pour 3 objets, à raison de 4 fenêtres par objet.

le nombre d'arbres construits et par conséquent le temps nécessaire à la prédiction d'une nouvelle image.

Pour conclure, comme c'était le cas pour la base de données ORL, le taux d'erreur est notablement réduit par notre approche locale. De plus, nous obtenons des résultats supérieurs aux autres méthodes d'apprentissage (Boosting, SVM). Selon le protocole utilisé, notre méthode talonne également le meilleur résultat connu à ce jour [OM02b] qui obtient 0.1% d'erreur.

4.3.4 OUTEX

Pour le problèmes de textures OUTEX, l'approche globale donnait de mauvais résultats, ce qui a notamment motivé l'introduction d'une approche locale par classification de fenêtres. Les textures étant généralement constituées de petits motifs qui se répètent à des positions non identiques au sein des images représentant la même texture, ce sont des fenêtres de petites tailles, 4×4 pixels, qui présentent les meilleurs résultats sur cette base de données. Nous obtenons un taux d'erreur de 2.78% sur l'ensemble de test de 432 images, avec un nombre d'arbres $T = 10$ et un nombre de fenêtres d'apprentissage $Nw = 120000$. Comme le montre la figure 4.14, le taux d'erreur est de moins en moins satisfaisant au fur et à mesure que la taille des fenêtres augmente, puisque des motifs plus grands sont moins souvent identiquement répétés d'une image à l'autre d'une classe. Au sein de cette même figure (en bas à gauche), la courbe du temps d'apprentissage montre très clairement que l'algorithme de construction d'arbres aléatoires a de plus en plus de mal à trouver des tests discriminants lorsque la taille des fenêtres augmente.

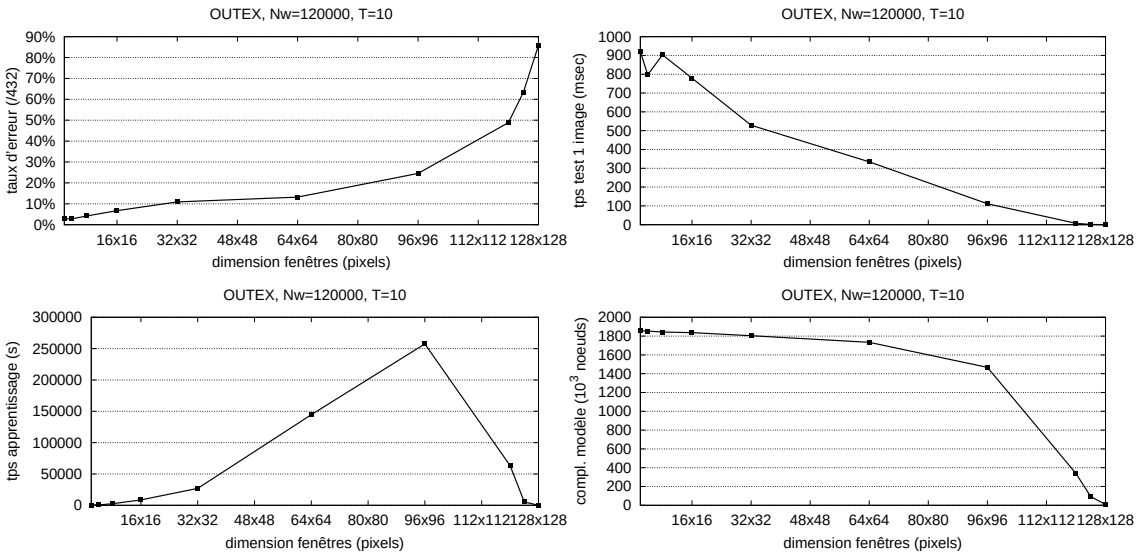

FIG. 4.14 – OUTEX : influence de la dimension des fenêtres sur le nombre d'erreurs, le temps de prédiction d'une image, le temps d'apprentissage, la complexité du modèle construit.

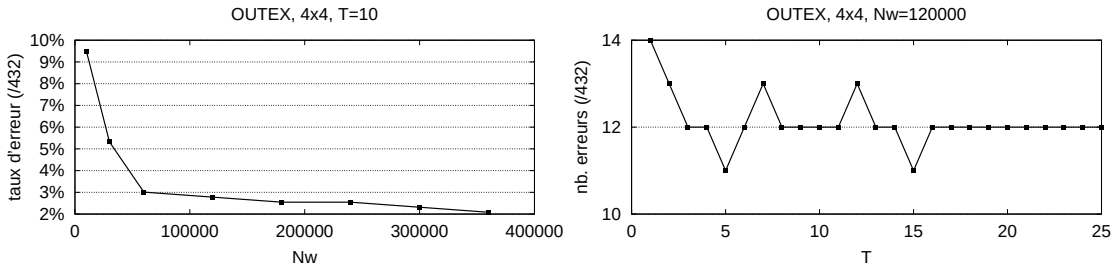

FIG. 4.15 – OUTEX : influence sur le nombre d'erreurs de la variation du nombre de fenêtres d'apprentissage avec 10 arbres (à gauche) et variation du nombre d'arbres avec 120000 fenêtres (à droite), sur le problème OUTEX avec une dimension de fenêtres de 4×4.

FIG. 4.16 – OUTEX : Quelques fenêtres de taille 4×4 pixels (agrandies ici pour une meilleure visibilité) extraites des images originales de tailles 128×128, pour 3 textures, à raison de 4 fenêtres par texture.

Étant donné la petite taille des fenêtres par rapport à la taille globale des images, le temps de prédiction d'une image est le plus élevé sur ce problème puisque nous balayons toujours entièrement chaque image de test. Avec la dimension de fenêtres optimale, 4×4 pixels, classer une image consiste à prédire la classe d'environ 15000 fenêtres ce qui nécessite un peu moins de 800 msec par image.

La figure 4.15 (à gauche) montre à nouveau la décroissance du taux d'erreur relativement à l'augmentation du nombre de fenêtres d'apprentissage, pour un nombre d'arbres $T = 10$ et la taille optimale de 4×4 pixels. Une centaine de milliers de fenêtres semble à nouveau être un bon compromis. Le nombre d'arbres construits n'influence que très légèrement le taux d'erreur. Nous avons d'ailleurs constaté que plusieurs arbres de l'ensemble atteignent individuellement un taux d'erreur similaire au taux d'erreur de l'ensemble des 10 arbres. Il est donc une fois de plus concevable de construire moins d'arbres pour réduire le temps de prédiction tout en conservant un taux d'erreur analogue.

En conclusion, l'approche présentée dans ce chapitre améliore grandement les résultats sur ce problème, passant d'environ 65% d'erreurs obtenus avec les meilleures méthodes d'apprentissage du chapitre précédent, à moins de 3%. Comparativement à la littérature, notre méthode reste inférieure à la meilleure des méthodes évaluées

par [MPV02] qui obtient un taux d'erreur de 0.2%.

4.3.5 Résumé des résultats

Nous récapitulons ici les résultats obtenus avec la méthode d'arbres aléatoires combinée à l'extraction aléatoire de fenêtres. La table 4.2 compare cette nouvelle méthode avec les arbres aléatoires et avec l'état de l'art. La table 4.3 récapitule le temps de prédiction par image, pour la méthode locale et la méthode globale, pour chaque problème. [9]

BDs	*Arbres aléatoires*	*Arbres aléatoires et fenêtres*	État de l'art
MNIST	3.26%	2.63%	12% à 0.5% [DKN04b]
ORL	4.56% ± 1.43	1.66% ± 1.08	5% à 2% [Rav04]
COIL-100	1.96%	0.37%	12.5% à 0.1% [OM02b]
OUTEX	65.05%	2.78%	9.5% à 0.2% [MPV02]

TAB. 4.2 – Récapitulatif des taux d'erreur pour les 4 problèmes.

BDs	*Arbres aléatoires*	*Arbres aléatoires et fenêtres*
MNIST	0.6 msec	2 msec
ORL	3 msec	354 msec
COIL-100	4 msec	14 msec
OUTEX	3 msec	800 msec

TAB. 4.3 – Récapitulatif des temps de prédiction (par image) pour les 4 problèmes.

4.3.6 Discussion

Grâce à l'approche locale, la précision est améliorée plus ou moins nettement selon le problème. OUTEX bénéficie très largement de cette nouvelle approche, ORL et COIL-100 en profitent également de manière sensible, tandis que les gains sur MNIST sont proportionnellement moins importants.

Par rapport aux méthodes d'apprentissage appliquées aux images globales et en particulier les arbres aléatoires, les deux principales objections que l'on peut émettre a priori sont un nombre de paramètres à ajuster plus important, ce qui pourrait questionner son caractère générique, ainsi qu'un temps d'exécution plus grand (aussi bien pour la phase d'apprentissage que celle de prédiction) et une complexité de modèle plus importante, ce qui pourrait rendre moins évidente sa mise en place.

[9] Dans ces expériences, les images et les arbres sont déjà chargés en mémoire.

Paramètres de la méthode

Concernant les paramètres à régler, notre étude a montré que le paramètre le plus influent sur le taux d'erreur est la taille des fenêtres extraites, les autres paramètres pouvant être bien souvent fixés à priori.

En effet, nous avons retenu un nombre d'arbres $T = 10$ pour chaque problème. Quant au nombre N_w de fenêtres d'apprentissage, il est d'une centaine de milliers pour 3 problèmes sur 4, la taille de l'échantillon de base du problème MNIST justifiant une valeur plus élevée. Ces deux paramètres peuvent prendre des valeurs plus élevées mais un gain important en termes de taux d'erreur n'est pas observé.

Au sujet de la taille des fenêtres, bien que nous n'ayons pas mis en place une procédure automatique de sélection de la taille optimale des fenêtres, il est assez aisé d'exécuter notre algorithme selon différentes tailles de fenêtres et d'en évaluer l'impact par validation croisée. Nous pensons que certaines tailles de fenêtres peuvent intuitivement être exclues pour un certain nombre de problèmes. Par exemple, comme nous l'avons dit, il ne semble pas évident pour un humain de reconnaître des chiffres sur base de fenêtres trop petites, et il en est de même pour notre méthode. D'autre part, lorsque les motifs d'une image ne se répètent pas à des positions identiques d'une image à l'autre, ou encore lorsque l'élément à reconnaître a une forme générale et/ou un contenu flexible, il est probablement plus opportun d'utiliser une taille de fenêtre petite relativement à la taille de l'élément, ce que nous avons observé pour la classification de textures. Par ailleurs, étant donné la diversité des objets à reconnaître au sein d'un même problème, il pourrait être intéressant de considérer plusieurs tailles de fenêtres, certains objets se distinguant par leur forme générale, d'autres par leur texture ou plus petites régions particulières. Nous développerons cette idée au sein du chapitre suivant (section 5.4.1). Enfin, il est à noter que nos expériences ont considéré des fenêtres carrées uniquement. Néanmoins, aucune adaptation n'est nécessaire pour extraire des fenêtres rectangulaires. Une brève étude [Mah04] n'a cependant pas mis en évidence l'amélioration du taux d'erreur à l'aide de quelques tailles de fenêtres rectangulaires.

Temps d'exécution

À propos du temps d'apprentissage et de prédiction, nos expériences ont été réalisées sur un ordinateur de gamme personnelle, et notre code de recherche n'a pas été optimisé. Il nous semble que le temps de prédiction est plus contraignant que le temps d'apprentissage pour beaucoup d'utilisations pratiques de l'approche. Par contre, si un système de classification doit réagir à l'ajout fréquent de nouvelles classes d'images, le temps d'apprentissage devient décisif également.

La prédiction d'une fenêtre avec un arbre requiert en moyenne seulement $O(logN_w)$ tests (chacun implique la comparaison d'un composant de pixel de cette fenêtre à

un seuil) et reste donc rapide même pour un grand nombre de fenêtres.

Lors de nos expériences, nous avons à plusieurs reprises souligné que le nombre d'arbres $T = 10$ assurait une certaine stabilité du taux d'erreur, mais un nombre moins élevé d'arbres (jusqu'à un seul arbre pour OUTEX) suffit parfois pour atteindre un même taux d'erreur. Utiliser moins d'arbres réduit le temps d'apprentissage et le temps de prédiction. Il est aussi envisageable, pour une application donnée, de réaliser un apprentissage incrémental c'est-à-dire d'ajouter des arbres en cours de route en fonction des erreurs de reconnaissance. L'algorithme offre donc beaucoup de possibilités pour diminuer les temps de calcul.

Par ailleurs, la méthode peut être facilement parallélisée, aussi bien pour la phase d'apprentissage que pour la phase de test. Pour l'apprentissage, la construction de chaque arbre étant indépendante des autres, ils peuvent être construits en parallèle pour autant que chaque processus ait accès à l'échantillon d'apprentissage. Pour la phase de prédiction, un processus peut appliquer la règle de décision d'un arbre à l'ensemble des fenêtres. Une autre approche consiste à attribuer à un processus l'application de l'ensemble des arbres à un sous-ensemble des fenêtres de l'image. Une approche hybride est envisageable. Dans tous les cas, la prédiction de l'image se réalise en combinant les vecteurs de probabilités calculés par chaque processus : il s'agit d'additionner les éléments des $T \times (W_x - w_1) \times (W_y - w_2)$ vecteurs de dimension M puis de rechercher le maximum c'est-à-dire la classe la plus probable.

Mais avant d'envisager la parallélisation, nous présenterons au sein de la section 4.4 deux approches qui réduisent considérablement les temps de prédiction sur un matériel tel que celui que nous avons utilisé.

Stockage du modèle

Par construction, le nombre de nœuds dans un arbre de décision construit selon l'algorithme 2.2 est borné par $2 * N_w - 1$. Dans nos expériences, pour chaque problème et pour chaque taille optimale de fenêtre, le nombre de nœuds dans un arbre n'a pas dépassé 200000. Chaque arbre peut être représenté par un tableau de nœuds. Chaque nœud d'un arbre contient 3 valeurs : un numéro d'attribut (l'attribut utilisé en ce nœud pour le test), un seuil (le seuil de discrétisation de l'attribut), un index vers le nœud fils gauche ou un numéro de classe si il s'agit d'un nœud terminal. Le nombre de byte nécessaire pour encoder le numéro d'un attribut dépend du nombre d'attributs, c'est-à-dire de la taille des fenêtres et du nombre de canaux par pixel. Nous avons utilisé de 1 à 4 bytes. Comme nous représentons les pixels par des valeurs d'intensité et que celle-ci varie entre 0 et 255, les valeurs de seuil peuvent donc être représentées par un seul byte. Étant donné la complexité des modèles, l'index de la troisième valeur du nœud doit être encodé par au-moins 3 byte. Toutefois, le code que nous utilisons encode un arbre sous la forme d'un tableau de flottants. Si l'espace mémoire est restreint, il est facilement possible de sauvegarder sur disque les arbres

durant la phase d'apprentissage. Pour la phase de test, on peut charger en mémoire un arbre à la fois et appliquer sa règle à l'ensemble des fenêtres, puis totaliser les prédictions.

Variantes

La méthode que nous avons présentée suit un schéma classique. Elle se compose d'une phase de détection de régions, d'une description de celles-ci et de la stratégie de reconnaissance. Chacune de ces étapes pourrait être modifiée et étudiée. Notre détection de régions est aléatoire, donc très rapide, mais il est envisageable de réaliser une étude comparative de différents détecteurs, notamment en termes de nombre de régions extraites et leur incidence sur le taux d'erreur. Ensuite, nous décrivons les fenêtres extraites directement à l'aide des valeurs d'intensité des pixels qui les composent, il est concevable d'étudier l'influence d'autres représentations courantes (comme les PCA ou DCT), en particulier si l'espace mémoire est restreint, et d'en évaluer les conséquences sur la précision. Enfin, d'autres algorithmes d'apprentissage pourraient remplacer la méthode d'ensemble d'arbres aléatoires comme le Boosting qui présenterait possiblement une légère amélioration du taux d'erreur, et une complexité moindre des modèles, au détriment d'un temps d'apprentissage beaucoup plus long.

Les résultats que nous avons obtenus avec notre méthode de base sont très positifs, mais l'étude de ces variantes pourraient constituer un extension intéressante de ce travail.

Images de tailles différentes

Il est intéressant de noter qu'avec cette approche locale, on peut traiter des images de tailles différentes, contrairement aux méthodes d'apprentissage sur les images globales qui travaillent en positions absolues des pixels dans une matrice. La seule contrainte est que la taille des images soit supérieure à la dimension des fenêtres.

4.4 Variante pour la phase de prédiction d'une nouvelle image

La phase de prédiction d'une image par notre méthode consiste à la parcourir entièrement par glissement d'une fenêtre selon un décalage d'un pixel à la fois. Toutes les fenêtres d'une taille donnée sont donc considérées. Par contre, la phase d'apprentissage consiste à extraire un nombre réduit de fenêtres parmi l'ensemble possible de toutes les fenêtres des images. Une idée, qui a été validée dans une étude parallèle à celle-ci et que nous avons supervisée [Mah04], est de ne pas considérer

toutes les fenêtres d'une image de test mais seulement un sous-ensemble de celles-ci. Deux approches ont été proposées. D'une part, la modification du pas horizontal et du pas vertical dans le parcours systématique de l'image. D'autre part, l'extraction aléatoire, à l'instar du procédé utilisé au départ de la phase d'apprentissage, d'un nombre réduit de fenêtres au sein de l'image, comme illustré à la figure 4.17.

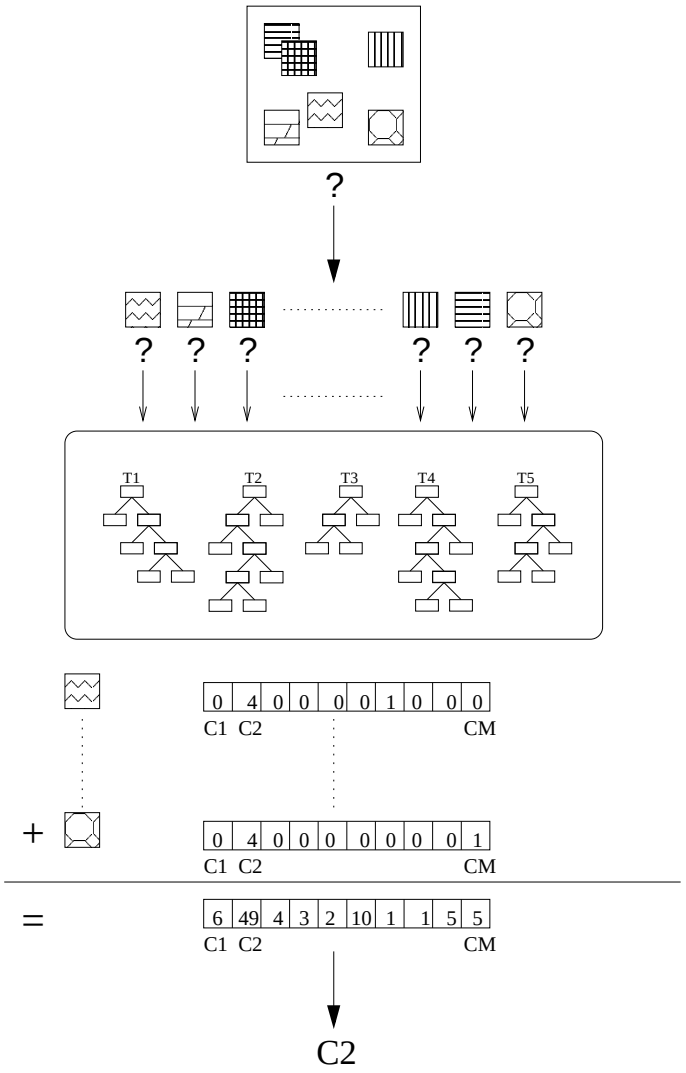

FIG. 4.17 – Variante pour la phase de prédiction : extraction aléatoire d'un certain nombre de fenêtres au sein de l'image à classer (premier et deuxième niveau), propagation de ces fenêtres dans l'ensemble d'arbre (troisième niveau), addition des vecteurs de probabilités de M classes (quatrième niveau), prédiction de la classe par recherche du maximum (cinquième niveau)

Il a été montré dans [Mah04] que ces deux méthodes pouvaient diminuer de manière nettement perceptible le temps de prédiction (puisque moins de fenêtres sont propagées dans le modèle) en maintenant un taux d'erreur voisin du taux

d'erreur obtenu par un parcours exhaustif de l'image. Cette étude a été menée sur les 4 problèmes que nous avons considéré.

Pour la première approche, une modification propre à chaque problème du pas est optimale. Il s'agit d'une modification du nombre de pixels de décalage dans le parcours horizontal et vertical entre chaque fenêtre. L'algorithme de base présenté précédemment a un pas horizontal d'un pixel et un pas vertical d'un pixel. En résumé de cette étude, pour un taux d'erreur semblable, il est possible de diviser au-moins par 3 le nombre de fenêtres et donc le temps de prédiction des images sur MNIST et COIL-100, d'au-moins par 25 sur OUTEX et jusqu'à 40 fois sur ORL.

La seconde approche, par tirage aléatoire, nécessite également un ajustement pour chaque problème, ici le pourcentage de fenêtres extraites dans l'image de test. Des résultats similaires à l'approche par modification de pas sont obtenus en tirant aléatoirement dans l'image un nombre de fenêtres correspondant au nombre de fenêtres qui résulte de la modification optimale du pas de parcours. Autrement dit, on obtient un taux d'erreur semblable avec environ 40% des fenêtres tirées aléatoirement par image pour MNIST, 35% pour COIL-100, 4% pour OUTEX, 3% pour ORL.

Ces résultats montrent donc qu'il n'est pas nécessaire de considérer toutes les fenêtres d'une image pour prédire avec succès sa classe. Avec ces variantes, il devient possible de classer les images de chaque problème à un taux assez élevé : environ 1000 images par seconde pour MNIST, 200 images pour COIL-100, 100 images pour ORL, 30 images pour OUTEX. Cette variante introduit deux nouveaux paramètres dans le cas de la modification du pas (pour le décalage vertical et horizontal) et un nouveau paramètre dans le cas du tirage aléatoire (le pourcentage de fenêtres extraites parmi toutes les fenêtres possibles).

4.5 Conclusions

Dans ce chapitre, nous avons motivé et présenté une nouvelle approche de classification d'images par la combinaison d'une part de l'algorithme d'apprentissage automatique d'ensembles d'arbres aléatoires, et d'autre part l'extraction aléatoire de fenêtres dans les images. Cette approche locale a été motivée par les résultats obtenus au chapitre précédent et se distingue des autres travaux de recherche dont nous avons présenté un état de l'art. Elle s'en différencie essentiellement par sa simplicité conceptuelle. Elle n'utilise pas de procédure élaborée de détection de points d'intérêts, la description des fenêtres est triviale, et la stratégie de reconnaissance est une méthode récente de classification qui repose sur l'utilisation d'un ensemble d'arbres de décision.

Nous avons réalisé une évaluation systématique de cette nouvelle approche selon ses différents paramètres et cela sur les problèmes déjà considérés au chapitre

précédent. Les résultats obtenus améliorent nettement les taux d'erreur obtenus précédemment par la méthode d'arbres aléatoires sur les images globales pour 3 problèmes sur 4, et légèrement pour le dernier. Notre méthode générique est comparable sur ces problèmes bien que légèrement inférieure (pour 3 problèmes sur 4) aux meilleurs méthodes de la littérature. Nos expériences montrent qu'un seul paramètre est réellement déterminant et dépendant du problème : la taille des fenêtres. Le temps de prédiction d'une image est acceptable sur une machine classique de bureau.

Chapitre 5

Évaluation de la robustesse et extraction de fenêtres transformées aléatoirement

Bien que les méthodes d'apprentissage automatique que nous avons présentées, et en particulier la méthode avec extraction de fenêtres, proposent déjà une solution partielle au problème de la classification d'images, nous désirons étudier la possibilité d'élargir leur champ d'application. Dans ce chapitre, nous évaluons la robustesse de l'approche en présence de perturbations ou transformations non contrôlées (section 5.2). C'est-à-dire que nous considérons la reconnaissance lors de changement de point de vue, de rotation ou d'échelle ; la reconnaissance en présence d'occultations partielles ou d'encombrement ainsi que la reconnaissance d'objets multiples. Nous montrons que la méthode avec extraction de fenêtres est plus robuste que la version globale de base. Après une revue de l'état de l'art de méthodes robustesse (section 5.3), nous proposons une procédure simple de génération de fenêtres transformées pour améliorer encore la robustesse de notre approche (section 5.4).

5.1 Problèmes de robustesse

Comme nous l'avons vu, des problèmes variés de classification d'images peuvent déjà être traités avec la méthode que nous avons présentée. En effet, elle propose telle quelle une solution partielle à la classification d'images, en particulier en présence d'images dont les conditions d'acquisition sont relativement bien contrôlées. Autrement dit, dans le cas où les variations entre les images de tests et d'apprentissage ne sont pas trop importantes.

Nous désirons maintenant élargir son champ d'application à des problèmes où les conditions d'acquisition ne sont pas toutes contrôlées, c'est-à-dire étudier la robus-

tesse de la méthode à des perturbations ou transformations qui peuvent se présenter en situations réelles.

Dans cette section nous présentons brièvement quelques transformations qui pourraient affecter les nouvelles images dont il faut prédire la classe, par rapport à des images d'apprentissage acquises en conditions "propres".

Illumination

D'une vue à l'autre, les images d'un même objet peuvent présenter différentes illuminations. Ces différences peuvent provenir de la nature des objets qui composent les images (objets réfléchissants, ...), de la position de la caméra et de la source lumineuse (autrement dit la géométrie de l'éclairage et la couleur de la lumière), de l'environnement (en intérieur ou extérieur, ombres, ...), du moment où l'acquisition est opérée (jour, nuit, saisons). Les méthodes de classification qui se basent sur l'apparence visuelle des objets pour les reconnaître peuvent donc être troublées par ces perturbations. L'effet combiné de tous ces changements n'est pas évident à modéliser.

Translation

Pour 3 des 4 problèmes considérés jusqu'à présent, les éléments à reconnaître (chiffres, visages, objets) sont plus ou moins centrés au sein des images d'apprentissage et de test. En conditions réalistes, les méthodes globales telles que celles que nous avons utilisées risquent d'être pénalisées par un décalage puisqu'elles se basent sur des pixels à des positions fixes au sein de l'image. La méthode locale par extraction de fenêtres est elle par principe moins affectée par la translation. Lors de la phase d'apprentissage et de test, des fenêtres sont extraites aléatoirement au sein de toute l'image. Des fenêtres issues d'une nouvelle image peuvent donc coïncider avec des fenêtres extraites lors de l'apprentissage, même si elles n'ont pas été extraites à des positions absolues identiques.

Occultations

Le phénomène d'occultation dénote le fait qu'une partie d'un objet peut ne pas être visible. Par exemple suite au passage d'un objet derrière un autre. La résistance aux occultations partielles est une fonctionnalité requise d'un système utilisé en conditions réelles. Il s'agit de pouvoir reconnaître un objet même si sa visibilité est partielle. Les approches locales sont en principe plus robustes aux occultations. En effet, même si toute l'information d'un objet n'est pas disponible, on peut considérer que certaines parties de l'objet seront encore intégralement visibles. La classification de ces parties n'étant pas affectée, leur reconnaissance permet la reconnaissance de

l'objet. Si l'information disponible est suffisante, l'élément pourra donc encore être reconnu. La résistance aux occultations dépend aussi du problème. Il semble par exemple possible de reconnaître un visage si sa moitié droite est cachée, cela semble moins aisé pour la reconnaissance de chiffres.

Encombrement

L'encombrement peut également faire obstacle à la reconnaissance. En effet, l'accumulation d'objets ou la présence d'un arrière-plan engendre une information supplémentaire qui peut perturber une méthode de classification. Le comportement souhaité du système de classification peut dépendre de la situation. Il peut être intéressant de reconnaître les multiples objets présents dans la scène, ou au contraire d'ignorer les éléments qui ne sont pas essentiels (et qu'on appelle parfois le "fouillis") pour la tâche en question.

Bruit

La capture de l'image d'un objet peut engendrer un bruit, selon les conditions d'acquisition, le matériel utilisé, la procédure de numérisation, etc. La nature du bruit peut fluctuer d'une image à l'autre. Par exemple, la mise au point incorrecte de la caméra induit un flou dans l'image. Le "fouillis" est une autre forme de bruit. Il est non-recouvrant car ce bruit est constitué par l'arrière-plan de l'image qui ne recouvre pas l'objet à reconnaître.

Rotation

La variation de l'orientation 2D c'est-à-dire la rotation dans le plan de l'image est une autre transformation susceptible d'influencer la classification d'une image. En effet, elle modifie la position des pixels au sein de l'image. Les méthodes locales sont en général également plus robustes. Notamment, pour des fenêtres de petites tailles incluses dans des régions homogènes, l'effet induit par la rotation peut être vu comme un simple décalage.

Échelle

Un changement d'échelle est la conséquence de l'éloignement ou du rapprochement d'un objet et/ou des réglages de la caméra. Ces phénomènes impliquent en outre une forme d'occultation partielle (en cas de rapprochement) et d'encombrement (en cas d'éloignement). Le changement de dimensions d'une image implique également une forme de changement d'échelle.

Point de vue

Le changement de point de vue peut affecter la reconnaissance d'images à cause des déformations (étirement, biais) qui en découlent. Pour un problème de classification d'objets comme COIL-100, l'effet de cette perturbation est visible en déplaçant le capteur sur un cercle autour de l'élément à reconnaître (variation de l'angle azimutal), comme illustré par la figure 3.6, page 36.

5.2 Évaluation de la robustesse

Nous venons de voir que de nombreuses transformations et perturbations peuvent modifier l'apparence d'un objet. Ces phénomènes peuvent influencer les méthodes de reconnaissance. En particulier, comme nous nous basons sur l'apprentissage par apparence à partir des pixels, notre approche est susceptible d'être sensible à diverses transformations.

L'idéal serait évidemment d'être robuste à toutes les transformations possibles combinées. Mais nous proposons dans ce chapitre d'évaluer la sensibilité de notre méthode à une transformation à la fois. En décomposant cette étude, nous pouvons mieux comprendre les faiblesses de l'approche et éventuellement l'améliorer par étapes. Aussi, certains environnements permettent de considérer que certaines perturbations n'auront pas lieu (par exemple en milieu industriel). Puisque la robustesse à une transformation entraîne généralement des performances moindres en situations contrôlées, il est souhaitable de proposer une méthode flexible. Aussi, l'activation ou la désactivation des procédures qui permettent la robustesse à une transformation particulière devrait être rendue possible. En effet, ces procédures requièrent parfois des opérations coûteuses en temps de calcul ou impliquent une perte de précision. De plus, les transformations peuvent être typiques d'une classe à reconnaître.[1]

Dans cette section, nous proposons d'étudier la robustesse de notre approche en utilisant principalement la base de données COIL-100 comme cadre d'évaluation.

5.2.1 Variations d'illumination

Les images des bases de données OUTEX et COIL-100 ne présentent pas de grandes variations d'illumination. Toutefois, les niveaux de couleurs peuvent varier entre les images des échantillons d'apprentissage et de test puisqu'un changement de point de vue est réalisé entre chaque image.

[1] On peut imaginer une application où le but d'un modèle de classification est par exemple d'identifier des visages qui sont à l'envers.

Une approche possible pour traiter les variations d'illumination consiste à normaliser les images avant de leur appliquer un algorithme de classification pour la phase d'apprentissage et/ou de prédiction. De nombreuses techniques de normalisation existent dans la littérature. Elles permettent de se défaire d'un ou plusieurs facteurs d'éclairage ([FSC98], [GS99]) avant d'appliquer la méthode de classification proprement dite (construction et comparaison d'histogrammes de couleurs, analyse en composantes principales, ...) à la reconnaissance d'objets ou de visages. Les différentes études ne semblent pas permettre d'élire une technique universelle de pré-traitement de la couleur.

Une autre stratégie consiste à extraire des caractéristiques indépendantes de l'illumination. Cette extraction impliquerait néanmoins une perte d'information.

Une autre méthode consiste à augmenter l'échantillon d'images d'apprentissage par des images qui subissent un éclairage différent, par exemple en déplaçant la source lumineuse. Cela de manière à rendre le modèle indépendant de la position de la source. Il est également envisageable de générer automatiquement un échantillon d'images à l'illumination variable en adoptant un modèle d'éclairage. Mais le modèle adopté doit alors être suffisamment complexe pour tenir compte, par exemple, des propriétés des objets réfléchissants.

Il n'est pas dans l'objectif de cette thèse d'étudier les nombreuses méthodes de la littérature qui traitent du problème large des variations d'éclairage. Il s'agit d'être conscient des faiblesses possible de notre méthode qui travaille directement à partir des pixels. Étendre notre approche à l'aide de techniques de normalisation est une extension possible. L'utilisation d'une représentation (autre que RGB) plus robuste à ce type de variations pourrait donc apporter un gain en précision en général, et sur ces bases de données en particulier. La sensibilité de la décomposition RGB de la couleur est en effet bien connue [GS99].

Dans cette section, nous proposerons donc comme première approche simple et générique de remplacer la représentation RGB des pixels par l'espace de couleurs HSV qui est moins sensible aux variations d'illumination. La conversion entre les modes RGB et HSV est établie [FvDFH93]. Nous la réalisons pour toutes les images de la base de données (d'apprentissage et de test). Le canal H (Hue) correspond à la teinte. La Saturation (canal S) représente la pureté de la couleur, c'est-à-dire son caractère terne ou vif. Enfin, le canal V (Value) représente l'intensité, c'est-à-dire son aspect sombre ou clair. Dans nos expériences, nous utilisons le triplet H,S,V, mais il est envisageable de n'utiliser qu'une ou deux composantes de cette représentation. Dans [OSI98], seule la valeur de H est utilisée pour un problème de reconnaissance d'objets. Cette composante représente l'information couleur.

La table 5.1 compare les taux d'erreur sur les bases de données OUTEX et COIL-100 selon les représentations HSV et RGB. L'approche globale par arbres aléatoires et l'approche locale par extraction supplémentaire de fenêtres sont évaluées selon les protocoles identiques à ceux des chapitres précédents. C'est-à-dire que 18

vues sont utilisées par objet pour l'apprentissage sur COIL-100, et 8 images par texture pour l'apprentissage sur OUTEX. Les paramètres des méthodes sont ceux qui donnaient les résultats retenus précédemment. Pour OUTEX et COIL-100, 500 arbres aléatoires sont construits pour la méthode globale. Pour la méthode locale, sur OUTEX, $T = 10$ arbres sont construits à partir de $Nw = 120000$ fenêtres de dimensions 4×4 pixels. Sur COIL-100, $T = 10$ arbres sont construits à partir de $Nw = 120000$ fenêtres de dimensions 16×16 pixels. On constate une amélioration sensible de la précision pour les deux problèmes avec la représentation HSV aussi bien pour la méthode globale que pour l'approche locale.

Méthode	**Taux d'erreur OUTEX**	**Taux d'erreur COIL-100**
Arbres aléatoires RGB	64.35% (278/432)	1.96% (106/5400)
Arbres aléatoires HSV	41.20% (178/432)	1.59% (86/5400)
Arbres aléatoires et fenêtres RGB	2.78% (12/432)	0.37% (20/5400)
Arbres aléatoires et fenêtres HSV	1.16% (5/432)	0.26% (14/5400)

TAB. 5.1 – Comparaison des résultats en représentation RGB et HSV, sur les base de données Outex et COIL-100.

Au sein de la section 5.2.2, nous proposons de poursuivre cette comparaison sur la base de données COIL-100 en faisant varier le nombre d'images d'apprentissage. La section 5.2.2 consiste à étudier l'effet du changement de point de vue sur la précision en utilisant moins d'images pour l'apprentissage. Mais la diminution de la taille de l'échantillon d'apprentissage implique également une possibilité moindre pour l'algorithme d'apprentissage d'apprendre les variations des valeurs de pixels d'un même objet. On s'attend donc à une précision comparativement meilleure en HSV et ce de manière plus nette au fur et à mesure que le nombre d'images d'apprentissage diminue.

5.2.2 Changement de point de vue

Comme pour tout problème d'apprentissage, un échantillon d'apprentissage représentatif des variations possibles des instances d'une classe permet une meilleure généralisation, autrement dit une meilleure précision. Dans le cas de la classification d'objets dont les vues d'apprentissage sont espacées régulièrement, si le nombre de vues disponibles est faible, les variations de l'apparence d'un objet entre une image de test et les image d'apprentissage seront grandes. On s'attend donc à ce que la reconnaissance soit moins précise lorsque le nombre d'images d'apprentissage diminue.

Nous proposons ici d'évaluer les capacités de généralisation de notre approche en considérant des tailles différentes de l'échantillon d'apprentissage. Sur COIL-100,

réduire le nombre d'images d'apprentissage implique pour la plupart des objets des déformations de perspective plus importantes entre les vues d'apprentissage et les nouvelles vues dont il faut prédire la classe.

La table 5.2 présente les taux d'erreur de notre approche selon différentes configurations de l'ensemble d'apprentissage. La figure 5.1 illustre ces résultats. Selon ce protocole, nv dénote le nombre de vues par objet sélectionnées pour l'apprentissage. Les $(72 - nv)$ vues restantes sont utilisées pour le test de chaque objet. Les angles azimutaux qui correspondent aux vues d'apprentissage sont mentionnés dans la première colonne de la table 5.2 de manière à permettre la répétition et la comparaison des résultats. Les paramètres des méthodes restent les mêmes. En particulier, avec l'approche locale, le nombre de fenêtres de 16×16 pixels extraites pour l'apprentissage est encore 120000 quel que soit le nombre de vues d'apprentissage. Le nombre d'arbres T est toujours fixé à 10.

Généralisation	*Arbres* RGB	*Arbres* HSV	*Arbres* + *Fenêtres RGB*	*Arbres* + *Fenêtres HSV*	LAFs [OM02b]
$nv = 36$; $k * 10°$	0.33%	0.11%	0.06%	0.03%	-
$nv = 18$; $k * 20°$	1.96%	1.59%	0.37%	0.26%	0.1%
$nv = 8$; $k * 45°$	7.55%	6.94%	1.53%	0.94%	0.6%
$nv = 4$; 45°+ k*90°	12.46%	10.13%	4.94%	2.72%	5.3%
$nv = 2$; 0°, 90°	24.91%	19.44%	12%	8.01%	12.2%
$nv = 1$; 0°	36.1%	32.61%	24.83%	19.58%	24%

TAB. 5.2 – Changement de point de vue (COIL-100) : taux d'erreur pour différentes tailles de l'ensemble d'apprentissage avec les arbres aléatoires, et l'approche avec extraction de fenêtres, pour les représentations RGB et HSV. Et comparaison avec la méthode LAFs [OM02b].

Comme prévu, le taux d'erreur augmente lorsque la taille de l'échantillon d'apprentissage diminue. C'est la méthode d'arbres aléatoires avec extraction de fenêtres et représentation HSV qui donne les meilleurs résultats. Cette approche est toujours meilleure que l'approche par arbres aléatoires sur les images globales. La représentation HSV se montre plus robuste que le codage RGB comme nous l'avions imaginé. Le taux d'erreur reste très faible, en deçà du pour-cent, jusqu'à 8 vues d'apprentissage par objet. Notre méthode est de plus au-moins aussi robuste que [OM02b] dont nous présentons les résultats [2] selon les mêmes protocoles.

Afin d'évaluer le rôle de la couleur pour la discrimination des objets de cette base de données, le tableau 5.3 donne les taux d'erreur de la méthode globale et

[2] Une communication personnelle avec l'auteur indique que des améliorations ont été apportées à la méthode publiée dans [OM02b], notamment pour la détection de régions. Selon ces améliorations, ces auteurs obtiennent un taux d'erreur de 11.5% pour $nv = 2$ et 19.69% pour $nv = 1$.

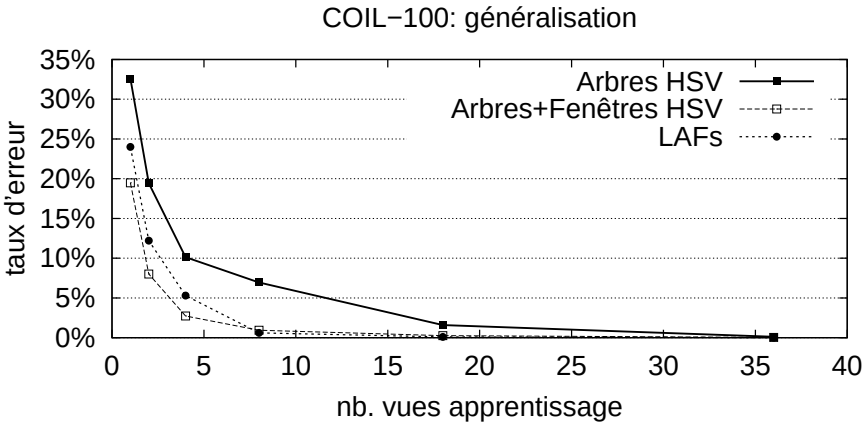

FIG. 5.1 – Changement de point de vue (COIL-100) : taux d'erreur pour différentes tailles de l'ensemble d'apprentissage avec les arbres aléatoires et l'approche avec extraction de fenêtres en HSV, ainsi que la méthode LAFs [OM02b].

de la méthode par extraction de fenêtres selon les mêmes paramètres et les mêmes protocoles que précédemment mais pour des images (d'apprentissage et de test) en niveaux de gris [3]. Nous utilisons donc encore $N_w = 120000$ fenêtres de dimensions 16×16 pixels à partir desquelles $T = 10$ arbres sont construits pour chacune des configurations. Utiliser des images en niveaux de gris au lieu d'images en couleurs implique une perte d'information et rend la tâche plus difficile car plusieurs objets de cette base de données se distinguent essentiellement par leurs couleurs. Nos méthodes se comportent donc moins bien qu'en représentation RGB et HSV mais un nombre d'images assez important sont toutefois encore bien classées.

Généralisation	*Arbres Niveaux de Gris*	*Arbres + Fenêtres Niveaux de gris*
$nv = 36$; $k * 10°$	0.89%	0.81%
$nv = 18$; $k * 20°$	3.93%	1.78%
$nv = 8$; $k * 45°$	11.39%	6.03%
$nv = 4$; 45°+ k*90°	23.47%	13.10%
$nv = 2$; 0°, 90°	32.6%	25.46%
$nv = 1$; 0°	44.49%	39.7%

TAB. 5.3 – Changement de point de vue (COIL-100) : taux d'erreur pour différentes tailles de l'ensemble d'apprentissage avec les arbres aléatoires, et l'approche avec extraction de fenêtres, pour la représentation en niveaux de gris.

Pour mieux comprendre le phénomène de changement de point de vue et se rendre compte des conséquences pour la reconnaissance, la figure 5.2 représente le

[3] La conversion a été réalisée à l'aide de l'utilitaire *convert* de *ImageMagick* : `http://www.imagemagick.org/`.

taux d'erreur sur l'ensemble des 100 objets de la base de données pour chaque vue de test. L'apprentissage a été réalisé à partir d'une seule vue d'apprentissage par objet ($nv = 1$) selon la méthode d'arbres aléatoires ainsi qu'avec la méthode locale avec extraction de fenêtres, en représentation HSV.[4] Cette figure permet de se rendre compte de l'influence sur l'erreur des déformations dues au changement de point de vue. En effet, en partant de la vue initiale (angle 0°sur le graphique) où le taux d'erreur est nul (il s'agit de la vue utilisée pour l'apprentissage), on observe une augmentation du nombre d'erreurs quand la vue de test s'écarte de la vue initiale et cela jusqu'à un certain point (environ $\pm$ 90°). Ensuite, le taux d'erreur diminue au fur et à mesure que l'on se rapproche de la vue du milieu ($\pm$ 180°). Cela s'explique par le fait qu'un certain nombre d'objets de la base de données sont quasi symétriques. Néanmoins, ils ne le sont pas tous, et l'apparence de certains objets vus de l'arrière est sensiblement différente de la vue de face, ce qui explique en partie les erreurs pour la reconnaissance selon cette vue. Enfin, nous observons que la méthode avec arbres aléatoires et extraction de fenêtres se comporte relativement bien (moins de 10% d'erreurs) jusqu'à un changement de point de vue de $\pm$ 45° alors que la méthode globale présente déjà un taux d'erreur proche de 30% pour un changement de point de vue de cet ordre.

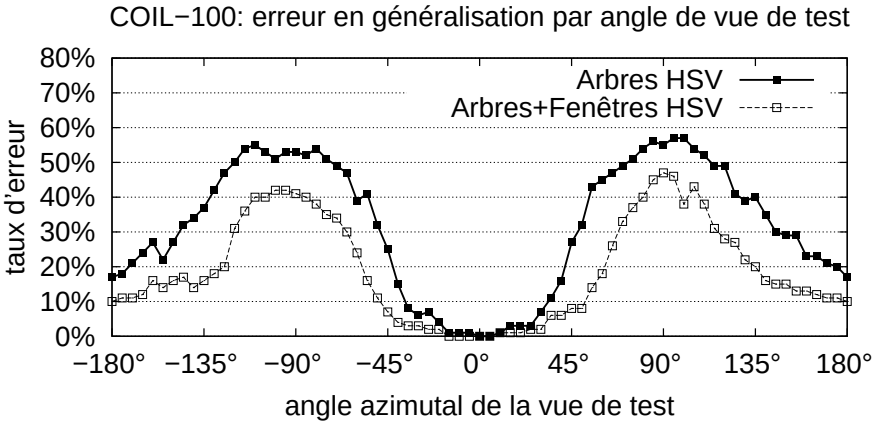

FIG. 5.2 – Changement de point de vue (COIL-100) : nombre d'erreurs en fonction de l'angle de vue de test pour la méthode par arbres aléatoires et la méthode avec extraction de fenêtres, en représentation HSV. Apprentissage avec $nv = 1$ (vue 0°).

À titre d'exemple, les objets mal classés pour la vue de test à 180° sont donnés par la figure 5.3 pour la représentation RGB et la représentation HSV. Pour cette vue, 18 objets parmi les 100 objets sont mal classés en représentation RGB contre 10 en représentation HSV. Les objets mal classés en HSV sont un sous-ensemble des objets mal classés en RGB. Autrement dit, pour cette vue, la représentation HSV permet de bien classer certains des objets mal classés par la représentation

[4]En représentation RGB, les courbes suivent la même forme, avec un taux d'erreur supérieur pour chaque vue. Mais nous ne les reproduisons pas pour alléger la figure.

RGB, sans introduire de nouvelle erreur (c'est-à-dire aucun objet mal classé en HSV n'était bien classé en RGB). Toutefois, les deux modèles construits à partir de la représentation HSV et RGB se trompent de manière différente pour 3 images, c'est-à-dire que la classe prédite erronément n'est pas la même. Ces observations sont à prendre comme un exemple et ne se généralisent pas nécessairement à d'autres problèmes.

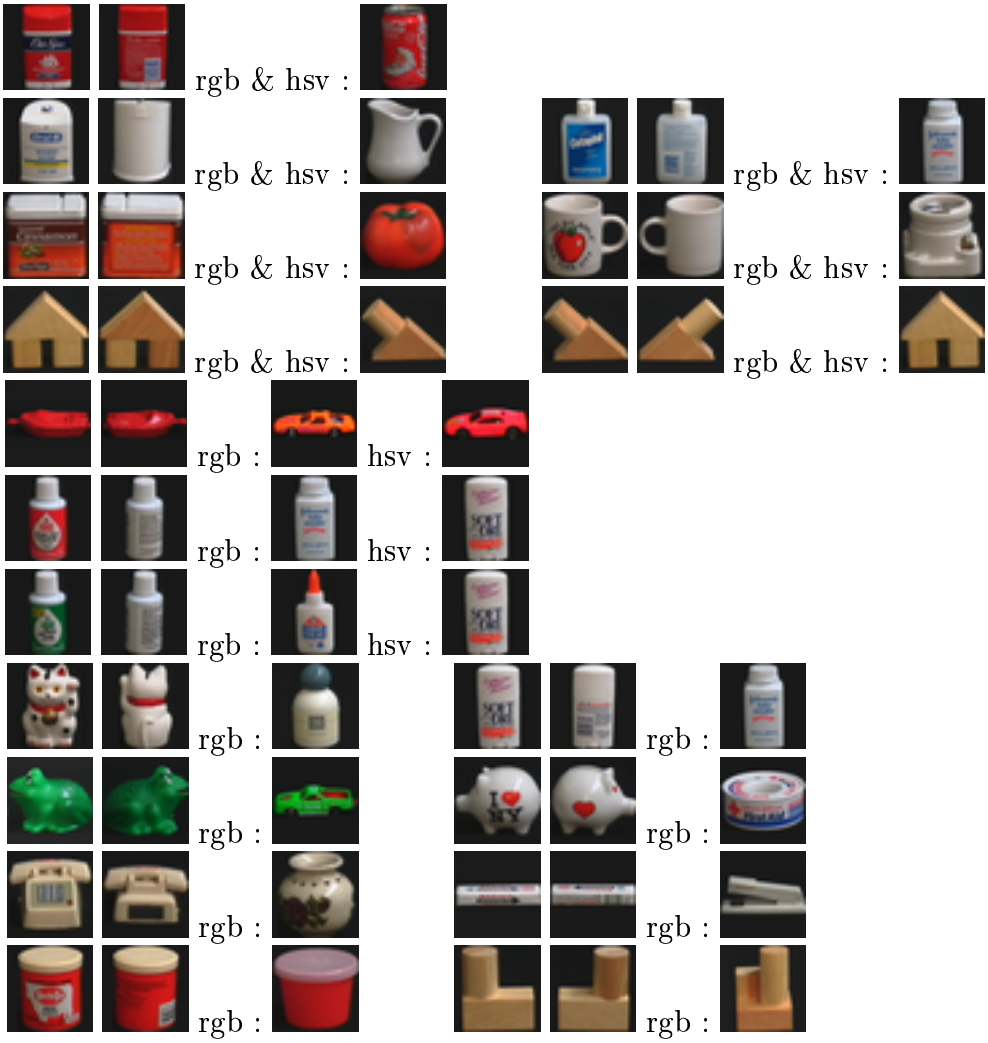

FIG. 5.3 – Changement de point de vue (COIL-100) : Pour chaque objet, la classe dont la vue frontale est dans l'ensemble d'apprentissage (à gauche) et la vue de test avec angle de vue de 180° de cet objet. À côté, en fonction de la représentation (rgb : et/ou ou hsv :), la vue frontale de la classe prédite erronément par la méthode d'arbres aléatoires avec extraction de fenêtres en RGB et/ou HSV.

5.2.3 Changement d'échelle

Pour évaluer la robustesse au changement d'échelle, nous testons le modèle de classification construit à partir des images d'apprentissage en 32×32 pixels sur des versions redimensionnées des images de test. Comme les arbres aléatoires sur images globales doivent travailler sur des images contenant le même nombre de pixels pour la cohérence des tests de pixels en positions absolues, cette méthode ne permet pas de classer des images de tailles différentes. Nous évaluons donc uniquement la robustesse de la méthode d'arbres avec extraction de fenêtres. Avec cette méthode, toute taille d'image peut être considérée, pour autant qu'elle soit supérieure à la taille des fenêtres extraites, 16×16 pixels dans notre cas.

Nous avons donc testé le modèle construit selon les paramètres habituels ($T = 10$, $Nw = 120000$), sur les images de test variant de 16×16 pixels à 48×48 pixels. Un exemple de redimensionnement est donné par la figure 5.4.[5]

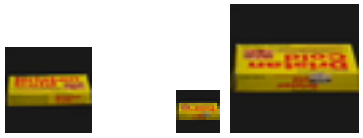

FIG. 5.4 – Changement d'échelle (COIL-100) : une image d'apprentissage en 32x32 avec angle de vue 0 (gauche), une image de test en 16x16 avec angle de vue 180°(milieu), la même image de test en 48x48 (droite)

Le protocole utilisé sépare comme précédemment la base de données en 18 vues par objet pour l'apprentissage, et 54 vues de test pour chacun d'entre eux. Nous avons reproduit les taux d'erreur sur les échantillons de test aux images redimensionnées au sein de la table 5.4. Les résultats sont relativement bons pour des variations modérées de l'échelle. Le très mauvais résultat obtenu avec des images de test de la taille des fenêtres (16×16) est dû au fait que seule une fenêtre peut dans ce cas de figure être extraite de l'image de test. En plus du changement d'échelle proprement dit, la classification correspond donc uniquement à la moyenne de 10 votes (un par arbre) ce qui n'est pas suffisant pour que l'effet de réduction de la variance des arbres aléatoires puisse se réaliser.

Remarquons que cette évaluation a été réalisée avec des images en représentation RGB. Le redimensionnement d'une image impliquant des moyennes de pixels et donc des modifications légères de couleurs, il est possible que le mode de représentation

[5]Selon ce protocole, il suffit de redimensionner les images de test dans la taille des images d'apprentissage pour obtenir une précision identique au cas de base. En pratique, la taille des images et la proportion occupée par les objets au sein de celles-ci n'est pas nécessairement connue. Notre protocole est une simplification d'un changement d'échelle en conditions réelles où de l'encombrement (dans le cas d'un éloignement ou zoom arrière) et une forme d'occultations (dans le cas d'un rapprochement ou zoom avant) peuvent apparaître. Il permet toutefois de se rendre compte des erreurs qui sont la conséquence d'un redimensionnement seul.

Changement d'échelle	*Arbres+Fenêtres*
16x16	71.72 %
24x24	2.67 %
32x32	0.37 %
40x40	1.43 %
48x48	7.15 %

TAB. 5.4 – Changement d'échelle (COIL-100) : taux d'erreur sur des images de test redimensionnées.

HSV soit plus robuste.

5.2.4 Changement d'orientation 2D

Pour l'évaluation de la robustesse au changement d'orientation (ou rotation 2D selon le plan de l'image), nous testons le modèle construit à partir des images originales sur des versions pivotées des images de test. La figure 5.5 donne un exemple de changement d'orientation pour un objet de COIL-100.

FIG. 5.5 – Changement d'orientation 2D (COIL-100) : une image d'apprentissage avec angle de rotation 0°(gauche), des images de test avec angle de rotation 45°(milieu et droite).

Le tableau 5.5 indique les taux d'erreur obtenus pour des changements d'orientation de 0°à 45°. La figure 5.6 illustre des résultats. Les arbres avec extraction de fenêtres se montrent plus robustes aux rotations que les arbres sur les images globales. Les taux d'erreur sont acceptables jusqu'à 20° de rotation, la méthode est donc robuste à des petites rotations. Cette évaluation a été réalisée avec des images en représentation RGB. La rotation d'une image peut impliquer de faibles changements de couleurs qui pourraient être mieux pris en compte par une représentation HSV.

5.2.5 Occultations partielles

Nous étudions la sensibilité aux occultations en effaçant des zones de taille croissante au sein des images de test. Un exemple où 50% de la partie droite d'une image est effacée est montré à la figure 5.7. Pour nos tests, nous remplaçons une certaine portion de la partie droite de l'image par des valeurs de pixels noirs, comme l'ont

Rotation 2D	*Arbres*	*Arbres+Fenêtres*
$ra = 0°$	2.04 %	0.37 %
$ra = 10°$	5.31 %	0.85 %
$ra = 20°$	22.80 %	3.20 %
$ra = 30°$	43.11 %	8.85 %
$ra = 45°$	76.61 %	31.69 %

TAB. 5.5 – Changement d'orientation 2D (COIL-100) : taux d'erreur avec images de test pivotées de ra degrés.

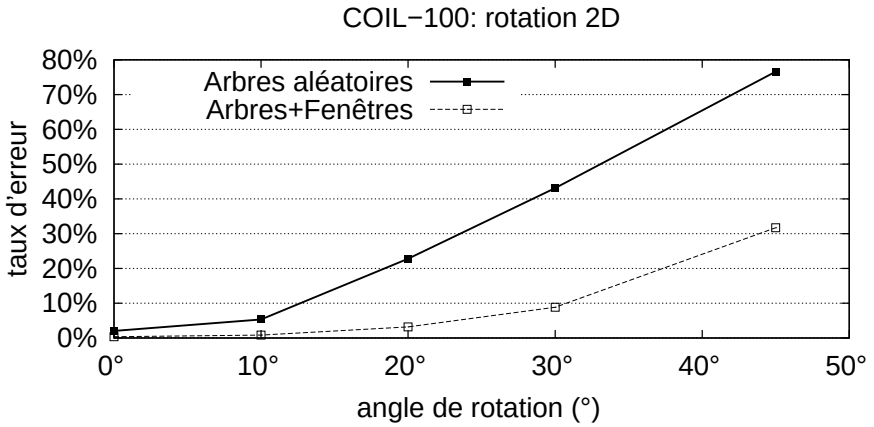

FIG. 5.6 – Changement d'orientation 2D (COIL-100) : taux d'erreur pour des images de test avec angle de rotation 2D.

fait [OM02b]. "of=x%" signifie que x% des pixels de la partie droite des images de test sont remplacées. Nos résultats sont résumés à la table 5.6 et illustré par la figure 5.8. Les arbres aléatoires avec extraction de fenêtres se comportent à nouveau mieux que les arbres sur images globales. Un certain nombre de fenêtres restent visibles et sont correctement classées. La précision de l'approche locale reste acceptable jusqu'à environ 40% d'occultation. Ces résultats sont toutefois inférieurs à ceux de [OM02b] qui obtient un taux d'erreur de 7.4% pour 50% d'occultation, ce qui est comparable à notre résultat obtenu pour 40% d'occultation.

FIG. 5.7 – Occultations partielles (COIL-100) : occultation de la moitié droite d'une image par remplacement noir.

Occultation	*Arbres aléatoires*	*Arbres+Fenêtres*
$of = 0\%$	2.04 %	0.37 %
$of = 25\%$	3.72 %	0.37 %
$of = 37.5\%$	12.10 %	4.72 %
$of = 40.625\%$	15.96 %	7.07 %
$of = 43.75\%$	21.85 %	14.04 %
$of = 46.875\%$	33.93 %	26.76 %
$of = 50\%$	49.26 %	47.69 %

TAB. 5.6 – Occultations partielles (COIL-100) : of % de la partie droite des images de test est remplacée par des pixels noirs.

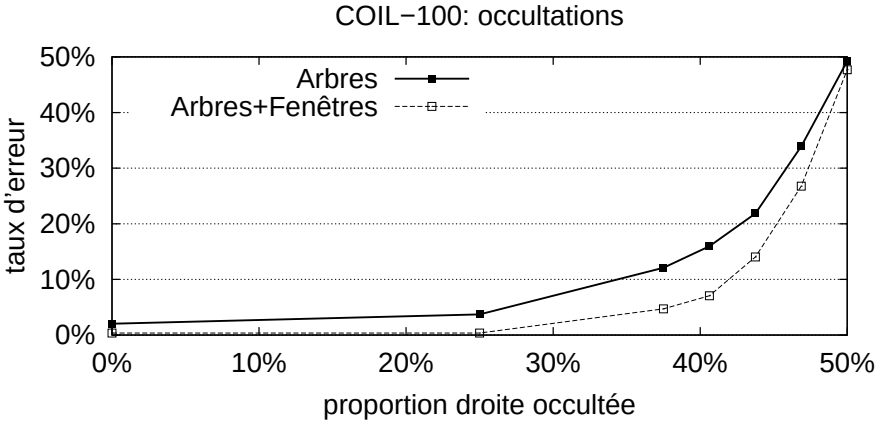

FIG. 5.8 – Occultations partielles (COIL-100) : taux d'erreur pour une portion occultée croissante des images de test.

5.2.6 Encombrement et objets multiples

Pour le phénomène d'encombrement et d'objets multiples au sein d'une image, nous ne réalisons pas de test systématique car les degrés de liberté sont nombreux. Nous évaluerons la méthode sur quelques exemples construits manuellement à partir de la base de données COIL-100. Afin d'étudier particulièrement la robustesse à ces perturbations, les images de test utilisées présentent un changement de point de vue minime de 5° par rapport aux images d'apprentissage. Les modèles sont ici encore construits à partir d'un ensemble d'apprentissage constitué de 18 vues par classe, selon des paramètres identiques aux expériences précédentes : 120000 fenêtres de tailles 16×16, $T = 10$ arbres.

Encombrement

Pour l'encombrement, nous proposons de construire le modèle à partir des images originales (sur fond noir) de COIL-100 mais d'ajouter un fond (une texture d'OUTEX) à quelques images de l'échantillon de test de COIL-100.[6] Le but est de toujours reconnaître les objets de ces images malgré les perturbations introduites dans les images de test. La figure 5.9 illustre les 5 exemples que nous avons créés manuellement. Ce protocole est assez irréaliste dans la mesure où les images d'apprentissage ont été segmentées. En quelque sorte, nous avons l'impression de créer un problème plus compliqué que la réalité où les images d'apprentissage pourraient contenir des images d'un objet sur différents fonds. L'algorithme d'apprentissage devrait alors pouvoir faire abstraction du fond qui n'est pas discriminant d'une classe à l'autre. Nous constaterons au sein du chapitre suivant (section 6.3) que la méthode se comporte en effet très bien sur un problème réaliste de reconnaissance de bâtiments avec encombrement au sein des images d'apprentissage et de test (le ciel, des arbres, des véhicules, ...).

FIG. 5.9 – Encombrement (COIL-100) : images de test contenant un objet de COIL-100 avec un fond encombré.

Malgré la présence d'un fond jamais présenté lors de la phase d'apprentissage, la méthode avec arbres aléatoires sur images globales se trompe seulement pour le premier et le dernier exemple et reconnaît correctement les 3 autres exemples. En utilisant la méthode d'arbres aléatoires avec extraction de fenêtres, les objets des quatre premières images de test sont bien reconnus. La présence d'un fond introduit un vecteur de probabilités de classe plus uniforme qu'avec le fond original noir comme pour les images d'apprentissage, mais la classe de l'objet à reconnaître reste la classe majoritaire. L'objet de la cinquième image n'est pas correctement reconnu. Pour cette image, le fond est assez complexe et coloré. De plus, l'objet n'occupe pas une grande partie de l'image. Il n'est d'ailleurs pas si évident que cela de distinguer cet objet du fond à l'œil nu.

Objets multiples

Pour la présence d'objets multiples, nous avons créé manuellement 5 exemples d'images au sein desquelles plusieurs objets de la base de données se chevauchent, de manière à présenter les différents cas problématiques. Ils sont illustrés par la figure

[6] En pratique, nous avons découpé une image d'un objet de COIL-100 faisant partie de l'ensemble de test par sélection de l'arrière-plan inversée, et nous avons collé l'objet sélectionné de cette manière sur une image d'une texture de OUTEX.

5.10. Certains impliquent en outre des occultations partielles, et comme nous l'avons dit un léger changement de point de vue (5°) par rapport aux images d'apprentissage de ces objets. Le résultat que nous espérons obtenir est la présence des classes de ces objets parmi les premières classes prédites par le modèle.

FIG. 5.10 – Multiples objets (COIL-100) : image de test contenant plusieurs objets à reconnaître de la base de données.

Avec la méthode d'arbres aléatoires sur images globales, un des deux objets du premier exemple est reconnu (la boîte jaune) tandis que le second objet ne fait pas partie des 5 premières classes prédites par le modèle. Il en est de même pour le cinquième exemple (c'est-à-dire le fruit en plastique est reconnu mais l'autre objet ne fait pas partie des 5 premières classes). Pour le deuxième, le troisième et le quatrième exemple, aucun des deux objets ne fait partie des 5 premières classes prédites par le modèle.

Avec la méthode d'arbres aléatoires et extraction de fenêtres, les deux objets des trois premiers exemples sont bien reconnus. En effet, pour chaque image, les deux premières classes recevant le plus de votes sont celles des objets présents dans ces images Pour le quatrième exemple, un des deux objets, la canette rouge, est reconnu comme première classe. C'est une canette rouge d'une autre marque qui est prédite en deuxième position, et le second objet est prédit comme troisième classe. Pour le cinquième exemple, un des deux objets de l'image est bien reconnu comme la première classe (le fruit en plastique jaune), tandis que la classe de l'autre objet (le récipient) est en cinquième position dans la liste des classes prédites par le modèle. Notamment, deux autres objets de couleur jaune s'intercalent entre ces prédictions.

Suite à ces quelques expériences, il s'avère que l'utilisation directe de l'approche pour la détection et la classification de plusieurs objets au sein d'une image n'est pas efficace. En effet, la méthode telle quelle ne permet pas de déterminer combien d'objets se trouvent au sein de l'image. Il est envisageable d'établir un seuil sur le nombre de votes reçus par chaque classe et de ne retenir que les classes qui reçoivent un nombre de votes supérieur à ce seuil. Mais cette approche ne semble pas suffisante car l'analyse des votes pour ces quelques exemples montre que le système attribue une probabilité similaire à plusieurs (plus de 2) classes. En outre, l'approche est incapable de se rendre compte de la présence multiple éventuelle d'un objet de la même classe.

Pour un problème de reconnaissance d'objets mutiples au sein d'une image dont la segmentation n'est pas aisée, il pourrait être utile, comme le propose [DKP+03], de détecter dans l'image des positions voisines qui correspondent à des fenêtres qui sont classées de manière similaire par le modèle. Par regroupement de ces régions, on pourrait alors déterminer la présence d'un objet. L'évaluation systématique sur

une base de données plus importante d'objets multiples et d'images encombrées est une autre extension de ce travail.

5.2.7 Discussion

Dans cette section, nous avons réalisé des tests systématiques pour évaluer la robustesse de la méthode globale d'arbres aléatoires ainsi que celle de la méthode locale par extraction de fenêtres.

Nous avons observé que les variations d'illumination peuvent être atténuées en utilisant une représentation HSV au lieu du mode RGB.

La méthode locale s'est montrée plus robuste que la méthode globale face au changement de point de vue, d'échelle, d'orientation 2D et en présence d'occultations partielles. Des tests ont également montré une meilleure robustesse à l'encombrement et à la présence d'objets multiples mais ces tests étaient moins nombreux et moins systématiques. Comme les images contiennent une certaine redondance et présentent des régions homogènes, certaines transformations spatiales impliquent que le contenu des images soit simplement décalé. Cela perturbe de manière plus importante l'algorithme global qui travaille directement aux positions absolues du vecteur de pixels. Globalement, les transformations causent beaucoup de changements au sein d'une image. De nombreux pixels sont déplacés ou modifiés et les variations entre les tableaux de pixels d'apprentissage et de test sont nombreuses. Au contraire, l'approche locale y est moins sensible. Un certain nombre de vecteurs de pixels des fenêtres restent en effet plus ou moins semblables suite à une transformation donnée. Cela explique donc la robustesse plus grande : certaines fenêtres intactes sont toujours bien classées par le modèle. La meilleure tolérance aux occultations partielles résulte également du fait qu'un certain nombre de fenêtres restent totalement visibles et il n'est pas requis que toutes les fenêtres soient correctement classées.

L'approche locale est suffisamment robuste à des petites rotations et à des changements d'échelle modérés pour permettre un traitement à échelle et/ou orientation multiples, lors de l'apprentissage ou de la phase de test, selon des échelles et orientations discrètes.

Mais si une application de classification d'images est susceptible de présenter des changements importants d'échelle et d'orientation 2D, l'approche présentée jusqu'ici ne sera peut-être pas suffisamment robuste. En effet, nous décrivons nos fenêtres à l'aide des valeurs de pixels qui la composent, et aucune information n'est fournie à l'algorithme d'apprentissage pour tenir compte de probables transformations.

Nous présentons dans la section suivante un état de l'art de méthodes locales robustes à des transformations de ce type. Ensuite, nous proposerons d'améliorer la robustesse de notre méthode à certaines transformations.

5.3 État de l'art de méthodes robustes

La tolérance aux occultations partielles et la robustesse aux transformations telles que la rotation, le changement d'échelle ou les changements de point de vue sont des difficultés abordées activement depuis quelques années en vision par ordinateur. Nous présentons brièvement quelques unes des méthodes qui traitent de ces problèmes en les regroupant en 3 familles.

5.3.1 Détection robuste de points d'intérêt et descripteurs invariants

Dans la littérature, de nombreuses méthodes sont des approches locales basées sur la détection robuste de points d'intérêts ou régions [MTS+04]. Ces régions sont détectées de manière stable d'une image à l'autre malgré un ensemble de transformations qui peuvent agir sur ces images. Les régions entourant ces points d'intérêts sont alors généralement représentées à l'aide de descripteurs calculés à partir de l'information qu'elles contiennent [MS03]. Généralement, ces descripteurs utilisent des propriétés mathématiques qui permettent l'invariance à certaines transformations locales. Ces caractéristiques sont appelées des descripteurs invariants. En effet, ils sont construits de manière à être insensibles à un certain nombre de transformations géométriques et photométriques. Ces techniques sont utilisées pour la reconnaissance d'objets ou de manière plus fondamentale pour l'appariement de caractéristiques qui correspondent au même point physique sur un objet vu selon deux points de vue arbitraires ("point matching" ou "stereo matching"). En effet, grâce à ces techniques invariantes, des régions identiques sont supposées être détectées puis décrites de la même façon d'une image à l'autre c'est-à-dire que leur vecteur est supposé identique ou proche. Certaines applications impliquent la connaissance de la configuration de la caméra pour les deux vues, d'autres supposent que le déplacement et l'orientation de la caméra sont inconnus mais relativement faibles. Le problème le plus complexe est celui où les variations des paramètres de la caméra sont larges et inconnus ("wide baseline stereo matching", [Bau00]). Ces approches peuvent également être utilisées pour l'identification d'objets 3D ou de scènes au sein de séquences vidéo [SZ02].

Plusieurs types de détecteurs de points d'intérêts et de descripteurs ont été proposés. Notamment, [SSTV03] utilisent des moments couleurs pour caractériser les régions elliptiques détectées autour d'extrema d'intensités. Seuls les extremas locaux qui sont détectés à plusieurs échelles de l'image (l'image originale ainsi que des versions redimensionnées selon 4 facteurs) sont considérés comme stables et sont utilisés pour extraire les régions. Chaque région est alors décrite par un ensemble de 9 invariants. La mise en correspondance de régions est réalisée selon le principe de classification basé sur la méthode des plus proches voisins.

[MS02] calule les dérivées gaussiennes des régions normalisées entourant les points d'intérêts détectés grâce à une méthode basée sur le détecteur de Harris. Le vecteur

de caractéristiques utilisé comporte 12 éléments. La distance de Mahalanobis est utilisée lors de la phase de prédiction pour mettre en correspondance les descripteurs des images. L'utilisation de critères supplémentaires permet de rejeter des appariements moins significatifs ou inconsistants.

[Bau00] détecte des points d'intérêt selon un détecteur de Harris multi-échelle. Chaque point d'intérêt est caractérisé par un vecteur d'invariants comportant entre 20 et 40 caractéristiques basées sur une variante de la transformation de Fourier-Mellin pour l'invariance aux rotations. Ce travail normalise au préalable les fenêtres entourant les points d'intérêt par rapport aux déformations de biais ("skew") et d'étirement ("stretch") ainsi que pour les changements d'intensités. La distance de Mahalanobis est ensuite utilisée pour mesurer la similarité entre deux vecteurs de caractéristiques. Cette approche a été évaluée sur quelques images de boîtes et de jouets.

5.3.2 Fenêtres locales affines

Une autre approche proposée par [OM02b] ne réalise pas le calcul de descripteurs invariants. Elle repose tout d'abord sur l'extraction de régions d'intensités homogènes. Ces régions sont invariantes aux transformations affines et de perspectives ainsi qu'aux transformations monotones de l'intensité de l'image. À partir de ces régions, plusieurs constructeurs génèrent des fenêtres normalisées localement par transformations affines ("Local Affine Frames", LAFs). Ces fenêtres sont alors décrites par les valeurs de pixels qui les composent et par les paramètres de leur normalisation c'est-à-dire un identifiant du constructeur utilisé et les paramètres géométriques et photométriques de la normalisation.

Lors de la phase de reconnaissance qui implique la mise en correspondance de fenêtres locales, les paramètres de normalisation des paires de fenêtres sont utilisés pour rejeter certaines comparaisons candidates. Notamment, les fenêtres qui ont été générées via des constructeurs différents ne sont pas comparées. En outre, une dizaine de conditions sur les coefficients des transformations photométriques et géométriques sont utilisées pour rejeter les correspondances jugées peu probables entre deux fenêtres. Autrement dit, les paramètres des transformations de normalisation de chacune des deux fenêtres sont comparés et si leur différence dépasse un certain seuil, ces fenêtres ne sont pas comparées entre elles. Les seuils qui établissent cette invraisemblance sont des paramètres de la méthode qui peuvent être appris à partir des images d'apprentissage. Les fenêtres qui n'ont pas été filtrées sont alors directement comparées par corrélation de leurs valeurs de pixels ou des représentations compactes de ces fenêtres (coefficients DCT dans [OM03] et PCA dans [OM02b]). Le nombre de correspondances établies donne alors une idée raisonnable de la similarité entre objets : plus le nombre de fenêtres locales sont similaires, plus grand est le score. Cette approche a également été utilisée avec succès pour la recherche d'images dans [OM02a] (recherche de publicités) et [OM03] (recherche de bâtiments).

5.3.3 Génération de nouveaux échantillons et apprentissage automatique

Un autre type d'approche consiste à augmenter l'échantillon d'apprentissage à l'aide de versions transformées des images d'apprentisssage. [RZE01] applique ce principe pour une meilleure robustesse à la présence d'un arrière-plan à partir des images globales et en utilisant les pixels comme attributs d'entrée de la méthode SVM. L'ensemble d'apprentissage est simplement augmenté d'images avec fond blanc, en plus du fond noir initial de la base de données COIL-100. Cette approche a été validée sur un sous-ensemble des images de cette base de données.

[DKN01] propose de générer des versions transformées par décalages de pixels des images globales de chiffres manuscrits (bases de données MNIST et USPS). Ces transformations sont appliquées aux images d'apprentissage ("virtual training data") mais également aux images de test ("virtual test sample method"). Les prédictions de ces versions décalées d'images sont combinées pour prédire la classe de l'image originale. Cette approche améliore les résultats, en particulier lorsqu'elle est combinée à la mesure de distance tangente.

[DS02] propose d'utiliser uniquement les vecteurs de supports obtenus par une première application de l'algorithme des SVM pour augmenter l'échantillon d'apprentissage à l'aide de versions transformées de ces vecteurs de support uniquement. L'algorithme est alors appliqué une seconde fois sur le nouvel échantillon d'apprentissage composé des vecteurs de support de l'application précédente et de leurs versions transformées. Cette approche a été validée sur des problèmes de reconnaissance de chiffres manuscrits (USPS et MNIST) en incorporant l'invariance aux translations uniquement.

Récemment, [LPF04] a appliqué cette approche dans le contexte de la mise en correspondance de points pour l'estimation de pose et le suivi d'un objet dans une séquence vidéo. Dans leur cas, les images d'une classe correspondent à une collection de vues d'un point d'intérêt. Il y a autant de classes que de points d'intérêt pour un objet donné. Un nouvel échantillon d'apprentissage est généré automatiquement à partir de fenêtres entourant les points d'intérêt détectés dans les images d'apprentissage. Ces nouvelles fenêtres sont générées selon des paramètres aléatoires de transformations. L'algorithme de classification apprend donc à reconnaître des points d'intérêt à partir d'un échantillon d'apprentissage de fenêtres obtenues par application d'un certain nombre de transformations relatives à l'orientation, la taille, la translation. Cette approche a été appliquée à l'estimation de pose et le suivi de quelques objets (un livre, un voilier, un visage, une boîte).

En dehors de l'amélioration de la robustesse, l'application de modèles de déformations appliqués aux images d'apprentissage d'un problème donné permet généralement d'améliorer le taux de reconnaissance par augmentation de l'échantillon d'images. Notamment, [KGN04] et [SSP03] appliquent différents modèles de défor-

mations aux images pour la reconnaissance de chiffres manuscrits.

Généralement, l'augmentation de l'ensemble d'apprentissage implique un temps de calcul supplémentaire nécessaire pour la phase d'apprentissage. Cet effet dépend de la méthode employée. Avec une méthode basée sur un ensemble d'arbres, le coût est quasi linéaire par rapport à cette augmentation. Pour réduire la dimension du problème, [LPF04] applique sur l'ensemble d'images l'analyse en composantes principales et un algorithme de clustering (K-Means). Un algorithme du plus proche voisin est ensuite utilisé pour la phase de prédiction. [DS02] propose de générer des versions transformées des images à partir des images de support uniquement et non à partir de la totalité des images d'apprentissage pour réduire ce coût.

5.4 Arbres aléatoires et fenêtres transformées

Nous proposons dans cette section d'utiliser une approche qui génère des fenêtres transformées à partir des images originales. Il s'agit d'un cas particulier des méthodes présentées au sein de la section 5.3.3. Cette approche est directe et notre schéma de classification reste ainsi identique et générique. La procédure d'extraction de fenêtres est toujours réalisée à des positions aléatoires. La description des fenêtres consiste encore à utiliser tous les pixels qui les décrivent plutôt qu'un nombre réduit d'invariants. L'approche par invariants fonctionne bien dans la littérature. Mais selon notre thèse qui vise à minimiser la perte explicite d'information, nous proposons plutôt d'augmenter la diversité au sein de l'échantillon d'apprentissage en appliquant des transformations aux fenêtres extraites. L'algorithme d'apprentissage devrait alors apprendre à classer les images selon différentes transformations qui peuvent les affecter. Nous utilisons encore la méthode d'apprentissage de construction d'arbres aléatoires. Avec cet algorithme, la phase d'apprentissage est assez rapide et pourrait donc supporter une augmentation de l'échantillon d'apprentissage.

Suite à notre évaluation dans la section 5.2, les transformations qui affectent le plus la précision de notre approche et qui peuvent se présenter régulièrement dans des cas réels sont les changements d'orientation 2D et d'échelle, ainsi que le changement de point de vue. Nous désirons donc mettre en place une procédure qui permet une meilleure robustesse à ces transformations par extension de la méthode d'extraction de fenêtres.

5.4.1 Extraction de fenêtres de tailles aléatoires

Dans la version de base de l'algorithme d'extraction de fenêtres (cfr. 4.2.1), pour un problème donné, des fenêtres d'une taille fixée sont extraites pour constituer l'échantillon d'apprentissage. Pour chaque problème, nous avions déterminé la dimension optimale de ces fenêtres en comparant les taux d'erreur obtenus sur la

totalité de l'échantillon de test. Lors de l'évaluation de la robustesse de la méthode, nous avons également utilisé cette taille de fenêtres.

Dans cette section, nous présentons une variante de l'extraction de fenêtres qui considère plusieurs tailles de fenêtres. Plusieurs raisons motivent ce choix. D'une part, si pour un problème donné il existe une taille optimale de fenêtres sur la totalité de l'échantillon de test, il est possible que différentes tailles conviennent mieux à certaines classes d'images. D'autre part, se limiter à l'extraction de fenêtres d'une seule taille revient à considérer implicitement que les images sont représentées à la même échelle, ce qui n'est le cas qu'en conditions contrôlées. Aussi, nombreuses sont les approches qui extraient des caractéristiques à plusieurs échelles. Notamment, [KKN04] propose d'extraire des fenêtres de tailles multiples dont la variance des pixels dépasse un certain seuil. Toutes les positions de pixels sont considérées. En chaque point, ils évaluent la variance de fenêtres de tailles variant par pas entre une taille minimale et une taille maximale (la taille de la plus petite image de leur base de données). Ces fenêtres sont ensuite redimensionnées dans une taille unique pour permettre la classification basée sur une méthode de plus proches voisins. Cette approche améliore leurs résultats sur une base de données de radiographies.

Inspiré par ces motivations et reposant sur notre méthode actuelle, nous proposons d'extraire au sein de chaque image (des échantillons d'apprentissage et de test) des fenêtres de tailles aléatoires entre une taille minimale (1×1 pixels) et une taille maximale (la taille de l'image considérée), et cela toujours à des positions aléatoires.[7] Toutes ces fenêtres sont ensuite redimensionnées dans une taille unique de manière à pouvoir être traitées directement par notre méthode qui travaille sur un tableau de données (comme c'est le cas de la plupart des méthodes d'apprentissage automatique). Ce redimensionnement est l'étape qui devrait permettre de reconnaître des caractéristiques semblables à des échelles différentes. L'algorithme de la phase d'apprentissage devient le suivant, et sa première étape est illustrée par la figure 5.11.

- On extrait aléatoirement un certain nombre N_w de fenêtres à partir des images de l'ensemble d'apprentissage. La taille $w'_1 = w'_2$ de chaque fenêtre est tirée aléatoirement entre 1×1 et $min(W_x, W_y) \times min(W_x, W_y)$ pixels. Sa position est ensuite déterminée aléatoirement pour être contenue entièrement dans l'image (entre 0 et $W_x - w'_1 - 1$ d'une part, et 0 et $W_y - w'_2 - 1$ d'autre part). La fenêtre est ensuite redimensionnée en $w_1 \times w_2$ pixels. On associe à chacune de ces fenêtres la classe de l'image dont elle est extraite.
- On construit un ensemble de T arbres aléatoires à partir du nouvel ensemble d'apprentissage constitué des N_w fenêtres décrites par les valeurs des $w_1 \times w_2$ pixels qui les composent.

[7] Cette extraction est générique mais, en pratique, il pourrait être intéressant d'adapter la taille minimale et maximale des fenêtres au problème considéré. Par exemple, des fenêtres de taille 1×1 sont probablement rarement intéressantes.

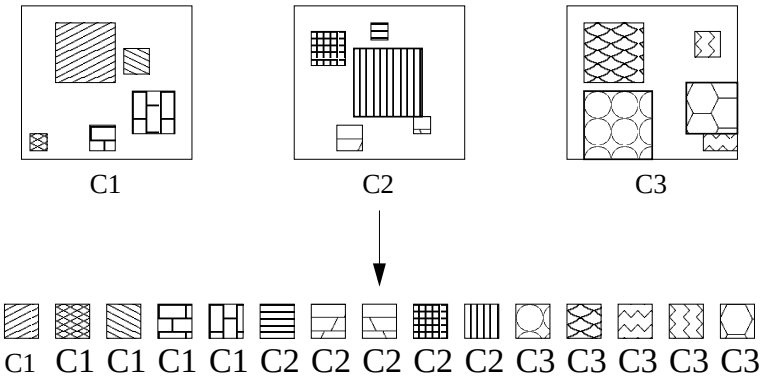

FIG. 5.11 – Première étape de la phase d'apprentissage avec extraction de fenêtres de tailles aléatoires à partir des images originales.

La version de base de la phase de test (cfr. 4.2.2) réalisait un parcours exhaustif des fenêtres de taille fixe d'une image. Il serait coûteux de réaliser autant de parcours de l'image qu'il y a de tailles de fenêtres possibles. Avec des fenêtres de tailles multiples, nous proposons d'utiliser la variante de la phase de test (cfr. 4.4) qui consiste à extraire un nombre réduit de fenêtres au sein de l'image à tester plutôt que la totalité d'entre elles. Cette variante est similaire à la procédure d'extraction réalisée lors de la phase d'apprentissage. La figure 5.12 illustre quelques fenêtres de tailles aléatoires et redimensionnées, pour une image du problème COIL-100.

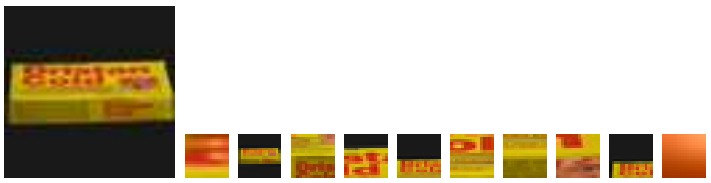

FIG. 5.12 – Une vue d'un objet de COIL-100 (à gauche) avec quelques exemples de fenêtres de tailles aléatoires redimensionnées en une taille unique

Changement d'échelle

Selon un protocole similaire à celui de la section 5.2.3, nous observons dans le tableau 5.7 que cette variante permet la robustesse aux changements d'échelle considérés. Le modèle est construit à partir d'un échantillon d'apprentissage constitué de $Nw = 120000$ fenêtres de tailles aléatoires extraites des 5400 images en 32×32 pixels, en mode RGB. Les fenêtres sont redimensionnées[8] vers la taille qui donnait les meilleurs résultats dans la version de l'algorithme utilisant une seule taille :

[8]Pour redimensionner les fenêtres, nous utilisons l'utilitaire *convert* de *ImageMagick* : `http://www.imagemagick.org`.

16×16 pixels. Cent fenêtres de tailles aléatoires sont extraites de chaque image de test, puis redimensionnées avant de les propager dans l'ensemble de $T = 10$ arbres. Le taux d'erreur de la variante de base pour des images de test de tailles 48×48 était de 7.15% (cfr. tableau 5.4). Avec cette nouvelle variante, le taux d'erreur reste semblable pour toutes les tailles d'images considérées. De plus, l'extraction de fenêtres de tailles aléatoires ne dégrade pas le taux d'erreur par rapport à la méthode de base appliquée au cas particulier d'images de test de la même taille que les images d'apprentissage. Il est intéressant de souligner que la taille de l'échantillon d'apprentissage est identique à celle de la version non robuste, $Nw = 120000$ fenêtres. Une meilleure robustesse est donc obtenue sans augmentation de la taille de l'échantillon d'apprentissage.

Changement d'échelle	*Arbres+Fenêtres Tailles aléatoires RGB*
16x16	0.44 %
32x32	0.50 %
48x48	0.56 %
128x128	0.44 %

TAB. 5.7 – Changement d'échelle (COIL-100) : taux d'erreur sur des images de test redimensionnées avec la variante avec fenêtres de tailles aléatoires. Le modèle est construit sur des images d'apprentissage de taille 32×32 pixels, en mode RGB.

Changement de point de vue

Bien que cette variante avec fenêtres de tailles aléatoires ait comme première motivation la robustesse au changement d'échelle, il est intéressant d'évaluer sa robustesse au changement de point de vue. En effet, certaines fenêtres de plus petites tailles sont susceptibles d'être mieux reconnues que des fenêtres de tailles plus grandes lors d'un changement de point de vue. En effet, certaines régions de petites tailles peuvent se répéter malgré un changement de point de vue. D'autre part, le redimensionnement de grandes fenêtres en fenêtres plus petites atténue certaines des déformations dues au changement de point de vue.

Nous avons évalué cet effet à partir du cas le plus difficile de la base de données COIL-100 qui consiste à utiliser une seule vue pour chaque objet dans l'échantillon d'apprentissage. Pour rappel (cfr. tableau 5.2), le taux d'erreur obtenu avec la version de base était de 24.83% en représentation RGB et 19.58% en représentation HSV pour les 7100 images de test en 32×32 pixels. Avec cette nouvelle variante utilisant des fenêtres de tailles aléatoires, nous obtenons 20.63% en représentation RGB et 13.58% en représentation HSV. À notre connaissance, il s'agit du meilleur résultat obtenu pour ce protocole sur cette base de données. La figure 5.13 représente le taux d'erreur sur l'ensemble des 100 objets de la base de données pour chaque vue de test (de la même manière que la figure 5.2, page 83). La méthode qui réalise l'extraction

de fenêtres de tailles aléatoires se montre plus robuste aux changements de point de vue que la méthode qui extrait des fenêtres de tailles fixes. Le nombre d'erreurs pour les vues autour de $\pm$ 180°est similaire pour les deux méthodes, mais la méthode avec fenêtres de tailles aléatoires se comporte mieux pour les vues proches de $\pm$ 90°.

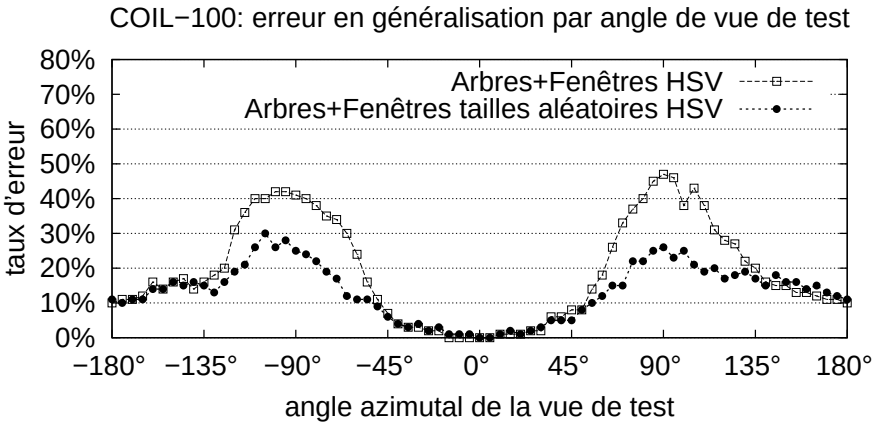

FIG. 5.13 – Changement de point de vue (COIL-100) : nombre d'erreurs en fonction de l'angle de vue de test pour la méthode avec extraction de fenêtres de tailles fixes et la méthode avec extraction de fenêtres de tailles aléatoires, en représentation HSV. Apprentissage avec $nv = 1$ (vue 0°).

Changement d'orientation 2D

Nous avons également évalué cette nouvelle variante au cas le plus difficile de notre protocole d'évaluation de la robustesse au changement d'orientation, c'est-à-dire dans le cas d'images de test ayant subi une rotation de 45° par rapport aux images d'apprentissage. Cette variante ne montre ni une amélioration ni une dégradation sensibles de la robustesse. Le taux d'erreur obtenu est de 30.86% contre 31.69% obtenu précédemment (cfr. tableau 5.5), en représentation RGB.

5.4.2 Extraction de fenêtres de tailles et d'orientations aléatoires

Dans cette section nous présentons une extension de l'extraction de fenêtres de tailles aléatoires. Elle applique une transformation de rotation aléatoire à toutes les fenêtres extraites afin d'améliorer la robustesse à cette transformation.

Nous proposons, pour chaque fenêtre de taille aléatoire extraite à une position aléatoire, et avant de la redimensionner, de lui appliquer une rotation d'angle aléa-

toire (entre 0° inclus et 360°). Nous la redimensionnons ensuite vers la taille unique. La figure 5.14 illustre cette variante. L'algorithme d'apprentissage devient donc :

- On extrait aléatoirement un certain nombre N_w de fenêtres à partir des images de l'ensemble d'apprentissage. La taille $w'_1 = w'_2$ de chaque fenêtre est tirée aléatoirement entre 1×1 et $min(W_x, W_y) \times min(W_x, W_y)$ pixels. Sa position est ensuite déterminée aléatoirement pour être contenue entièrement dans l'image (entre 0 et $W_x - w'_1 - 1$ d'une part, et 0 et $W_y - w'_2 - 1$ d'autrepart). Une rotation d'angle aléatoire (entre 0° inclus et 360°) est appliquée à la fenêtre. Celle-ci est ensuite redimensionnée en $w_1 \times w_2$ pixels. On associe à chacune de ces fenêtres la classe de l'image dont elle est extraite.
- On construit un ensemble de T arbres aléatoires à partir du nouvel ensemble d'apprentissage constitué des N_w fenêtres décrites par les valeurs des $w_1 \times w_2$ pixels qui les composent.

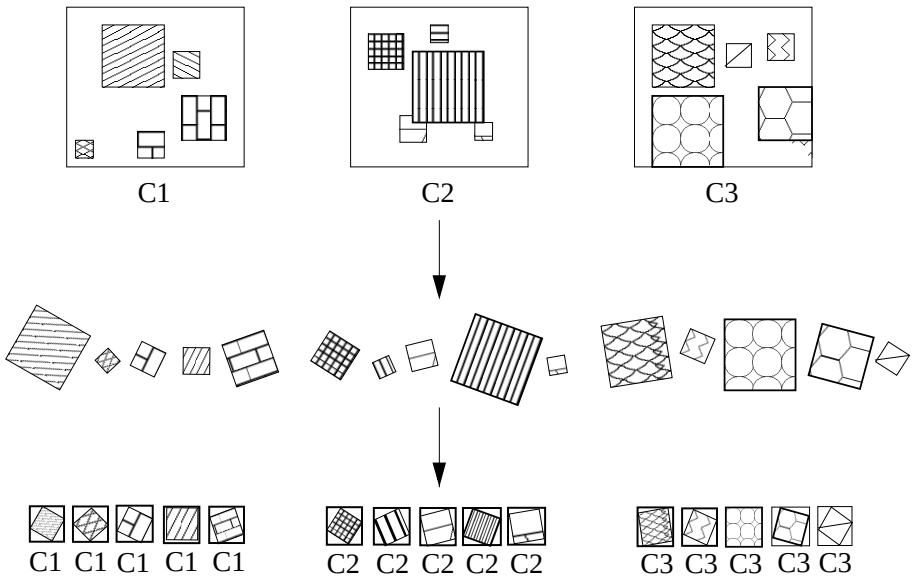

FIG. 5.14 – Première étape de la phase d'apprentissage avec extraction de fenêtres de tailles et d'orientations aléatoires à partir des images originales.

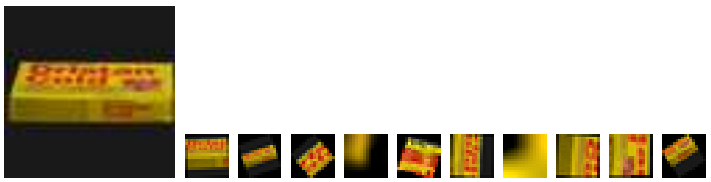

FIG. 5.15 – Une vue d'un objet de COIL-100 (à gauche) avec quelques exemples de fenêtres de tailles aléatoires et d'orientation aléatoires, redimensionnées en une taille unique.

La phase de test d'une nouvelle image implique le même processus de tirage d'un

nombre réduit de fenêtres de tailles aléatoires à des positions aléatoires, transformées par une rotation aléatoire, puis redimensionnées en une même taille pour être propagées dans le modèle de classification. La figure 5.15 illustre quelques fenêtres de tailles et d'orientations aléatoires et redimensionnées, pour une image du problème COIL-100.

Changement d'orientation 2D

Selon un protocole similaire à celui de la section 5.2.3, nous observons dans le tableau 5.8 que cette variante améliore la robustesse aux changements d'orientation. Le modèle est construit à partir d'un échantillon d'apprentissage constitué de $Nw = 120000$ fenêtres de tailles et d'orientations aléatoires [9] extraites des 5400 images en 32×32 pixels. Les fenêtres sont redimensionnées en 16×16 pixels.

Cent fenêtres de tailles et d'orientations aléatoires sont extraites de chaque image de test, puis redimensionnées avant de les propager dans l'ensemble de $T = 10$ arbres. Le taux d'erreur de la variante de base pour des images de test avec rotation de 45° était de 31.69% en mode RGB (cfr. tableau 5.5). La robustesse est donc nettement améliorée puisque nous obtenons maintenant 5.87% en mode RGB et 1.56% en représentation HSV. Le mode HSV est plus robuste que le mode RGB. Cela s'explique sans doute par les modifications des valeurs de pixels dues à la combinaison des redimensionnements et rotation des fenêtres. Ces transformations impliquent des interpolations et échantillonnages des pixels. Ces modifications affectent donc davantage le mode RGB. Le taux d'erreur dans le cas particulier (0°) est légèrement moins bon avec cette variante qu'avec la version de base et la variante avec tailles de fenêtres aléatoires. Mais ce cas ne se rencontre que dans un environnement aux conditions contrôlées. En situations réelles, la meilleure robustesse obtenue est préférable malgré cette légère perte. Ici encore, la taille de l'échantillon d'apprentissage est identique à celle de la version non robuste, $Nw = 120000$ fenêtres.

Changement d'échelle

Comme l'extension de la méthode pour la robustesse au changement d'orientation induit une légère diminution de la précision pour le changement de point de vue, nous évaluons également son effet en présence de changement d'échelle seul. Nous testons donc le modèle construit à partir des 1800 images en 32×32 pixels aux 5400 images de test en 48×48 pixels. Ce protocole menait à un taux d'erreur de 7.15% avec la méthode de base et 0.56% avec la méthode utilisant des fenêtres de tailles aléatoires, en représentation RGB. Le taux d'erreur obtenu avec des fenêtres de tailles et d'orientations aléatoires est de 3.57%. Ce résultat est similaire à celui

[9] Ces opérations de redimensionnement et rotation sont réalisées à l'aide de l'utilitaire *convert* de *ImageMagick*.

Rotation 2D	*Arbres+Fenêtres Tailles+Orientations aléatoires RGB*	*Arbres+Fenêtres Tailles+Orientations aléatoires HSV*
$ra = 0°$	3.72%	0.87%
$ra = 45°$	5.87%	1.56%
$ra = 90°$	4.39%	1.24%
$ra = 135°$	8.19%	2.56%

TAB. 5.8 – Changement d'orientation 2D (COIL-100) : taux d'erreur avec images de test pivotées de ra degrés, avec la variante utilisant des fenêtres de tailles et orientations aléatoires. Le modèle est construit sur des images d'apprentissage de taille 32×32 pixels, en mode RGB et HSV.

obtenu lorsque les images de tests sont de même taille et de même orientation 2D que les images d'apprentissage (cfr. $ra = 0°$ dans le tableau 5.8). L'utilisation de fenêtres d'orientations aléatoires n'est donc pas sensiblement moins robuste au changement d'échelle que les autres variantes.

Changement de point de vue

Comme pour la variante avec extraction de fenêtres de tailles aléatoires uniquement, nous évaluons la robustesse au changement d'échelle de cette nouvelle version à partir du cas le plus difficile de la base de données COIL-100. Cela consiste à utiliser une seule vue pour chaque objet dans l'échantillon d'apprentissage et à prédire la classe des 7100 images restantes. Nous obtenons un taux d'erreur de 22.92% en représentation RGB, et 16.68% en HSV. Ce résultat est moins bon que la variante avec tailles de fenêtres aléatoires uniquement (20.63% et 13.58% respectivement) mais meilleur que la version de base (24.83% et 19.58%). La robustesse aux rotations introduit donc une légère perte de précision par rapport au cas où seule la procédure de robustesse au changement d'échelle est utilisée, pour le cas particulier où les images de test ont la même orientation que les images d'apprentissage.

5.4.3 Discussion

Dans cette section, nous avons proposé une extension de la méthode d'extraction de fenêtres à des positions aléatoires. Elle consiste à extraire des fenêtres de tailles et d'orientations aléatoires. Nous l'avons évaluée pour les cas de transformations auxquelles la méthode de base était la plus sensible. Une meilleure robustesse est obtenue sans augmentation de l'échantillon d'apprentissage mais par simples transformations appliquées à toutes les fenêtres extraites, lors de la phase d'apprentissage et lors de la phase de test. D'autres études, présentées au sein de la section 5.3.3, ont montré, comme nous, que l'apprentissage à partir d'images générées permet une

meilleure robustesse. Dans le cas de ces études, il s'agissait de la robustesse à l'encombrement [RZE01], aux translations [DS02] et au changement de pose [LPF04].

Une extension possible de notre étude est d'évaluer l'utilisation d'un nombre beaucoup plus grand de fenêtres d'apprentissage. Nous avons réalisé quelques essais dans ce sens mais notre implémentation actuelle du stockage en mémoire de l'échantillon d'apprentissage limite le nombre de fenêtres utilisables. Avec $N_w = 200000$ fenêtres, nous n'avons pas observé une amélioration de la robustesse.

Temps de calcul

Les extensions apportées à l'approche de base pour une meilleure robustesse n'impliquent pas d'augmentation de la taille de l'échantillon d'apprentissage. Les temps d'apprentissage, la complexité des modèles et les temps de prédiction d'une fenêtre par un modèle sont comparables à la méthode de base.

Par contre, l'extraction de fenêtres implique davantage d'opérations avec ces variantes. Comme nous avons utilisé un utilitaire externe à notre code, une implémentation qui intègre directement ces opérations permettrait de réduire les temps de calcul et les échanges de données. En pratique, si il est nécessaire de réduire davantage les temps de calcul pour la prédiction d'une nouvelle image, il est envisageable d'appliquer les transformations uniquement aux images d'apprentissage. La prédiction d'une nouvelle image n'impliquerait alors pas les opérations de rotations et de redimensionnements appliquées à ses fenêtres. Appliquer la méthode d'extraction de fenêtres de tailles et orientations aléatoires uniquement aux images d'apprentissage pour l'évaluation de la robustesse au changement d'orientation donne un taux d'erreur de 6.8% pour les images de tests avec 0°de rotation (contre 3.72% si la procédure est appliquée aux images de test également) et 10.87% pour 45° de rotation (contre 5.87%), en mode RGB. Ce résultat est aussi à comparer avec 31.69% d'erreur obtenus avec la méthode de base (fenêtres de taille et d'orientation fixe) pour 45°de rotation. Ces variantes offrent un compromis entre le temps de prédiction et la précision. Leur utilité peut être déterminée en pratique pour chaque application.

Extensions

Notre méthode simple offre de bons résultats. Plusieurs extensions sont possibles. Une transformation supplémentaire pourrait être appliquée aux fenêtres extraites pour synthétiser un effet d'étirement ou de biais qui peut résulter d'un changement de point de vue (inclinaison, ...). Les résultats que nous avons obtenus pour la robustesse au changement de point de vue sur la base de données COIL-100 sont toutefois déjà satisfaisants.

Une autre extension consiste à utiliser les fenêtres locales affines construites par la méthode [OM02b] présentée au sein de la section 5.3.2. Une façon directe de com-

biner notre approche à cette méthode est de remplacer notre extraction aléatoire de fenêtres par la détection de régions et la construction de fenêtres présentés par ces auteurs. Pour la phase d'apprentissage, leur programme de génération de fenêtres normalisées peut être appliqué aux images d'apprentissage. Les fenêtres ainsi générées constituent l'ensemble d'apprentissage et un ensemble d'arbres aléatoires peut être construit à partir de celles-ci. Pour la phase de test, le même procédé est utilisé pour les images à classer. Chaque fenêtre générée est propagée dans l'ensemble d'arbres et les prédictions attribuées à toutes ces fenêtres sont additionnées et combinées. La construction et l'utilisation d'un modèle d'ensemble d'arbres remplace donc le procédé d'appariement par rejets de correspondances improbables et mesure de distance de la méthode originale [OM02b]. Quelques exemples de fenêtres extraites par cette méthode sont donnés pour une image de la base de données COIL-100 par la figure 5.16.

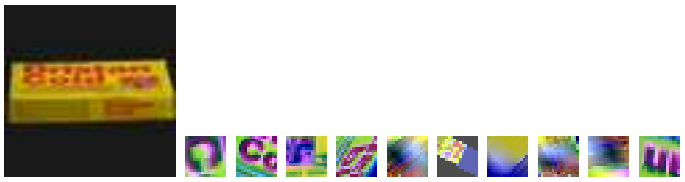

FIG. 5.16 – Une vue d'un objet de COIL-100 (à gauche) avec quelques exemples de fenêtres extraites par la méthode [OM02b].

Nous avons réalisé quelques expériences sur COIL-100. La méthode de [OM02b] n'a jamais été appliquée à des problèmes de reconnaissance de chiffres, de visages, de textures. Sur COIL-100, nous obtenons un taux d'erreur de 0.37% en combinant les arbres aléatoires et les fenêtres affines pour les 1800 images d'apprentissage et les 5400 images de test en résolution 32×32 pixels. Ce résultat est comparable à notre approche. Notons que nos temps de calcul sont inférieurs à cette approche grâce à notre extraction aléatoire de fenêtres et cela en nombre relativement réduit. Par exemple, l'application du programme de génération de fenêtres affines aux 1800 images d'apprentissage de COIL-100 de dimensions 32×32 pixels, produit plus de 390000 fenêtres.[10]

Pour la robustesse au changement d'échelle, cette combinaison donne 9.57% pour les images de test en dimension 16×16 pixels, 1.13% en 48×48 pixels et 2.85% en 64×64 pixels. Les résultats de notre approche étaient proposés au tableau 5.7, page 98.

Pour la robustesse au changement d'orientation, cette combinaison donne 8.93% pour des images de test pivotées de 45°et 3.63% pour une rotation de 90°. Les résultats de notre approche étaient reproduits dans le tableau 5.8, page 102.

[10]La méthode [OM02b] propose de nombreux constructeurs de fenêtres à partir des régions. Il est possible d'activer ou désactiver ceux-ci. Notre expérience utilise tous les constructeurs de cette méthode, comme l'ont fait leurs auteurs. Les fenêtres normalisées sont représentées en RGB.

La combinaison des fenêtres affines et des arbres aléatoires est aussi moins robuste au changement de point de vue : 36.42% d'erreurs sont obtenus sur les 7100 images de test dans le cas où une seule vue est utilisée pour l'apprentissage. Notre meilleur résultat était de 13.58%.

Les résultats obtenus sont inférieurs à ceux de notre approche. Ces expériences préliminaires suggèrent que cette méthode d'extraction de régions requiert des images de dimensions plus grandes pour offrir de meilleurs résultats. Les transformations que nous avons simulées impliquent des dégradations d'images qui perturbent sensiblement la détection de régions de cette méthode. Certains détails des surfaces des objets sont en effet diminués ou perdus suite aux transformations d'images, à leur redimensionnement et à leur encodage en tableau de pixels.[11] Nous n'avons pas pu réaliser directement toutes ces expériences avec des tailles d'images plus grandes, car le nombre de fenêtres générées occupe alors un espace mémoire prohibitif pour notre implémentation lorsqu'elles sont réunies en un seul échantillon d'apprentissage. Toutefois, pour la robustesse au changement de point de vue, un taux d'erreur de 27.73% est obtenu si les images sont de taille 128×128 pixels au lieu de 36.42% en 32×32 pixels.[12] Ce résultat reste inférieur à ceux de notre approche. Il l'est également par rapport à celui obtenu selon la combinaison de fenêtres affines avec le processus complet d'appariement décrit au sein de la section 5.3.2. Ce processus filtre les fenêtres en rejetant les comparaisons qui sont improbables sur base des détecteurs de régions et constructeurs de fenêtres utilisés ainsi qu'en fonction des paramètres de transformations de normalisation qui ont été réalisées. Selon notre combinaison, ces critères ne sont pas pris en compte. Il semble en outre que les fenêtres générées par leur approche ont une variabilité plus importante au sein d'une même classe suite à l'utilisation de nombreuses techniques différentes de normalisation. En termes de valeurs de pixels par exemple, la figure 5.16 montre des fenêtres générées par plusieurs constructeurs. Celles-ci comportent des pixels dans de nombreux tons (bleus, jaunes, rouges, verts, mauves, roses, gris, blancs) alors que l'objet dont elles sont extraites est principalement rouge et jaune sur fond noir. Plusieurs objets se partagent alors de mêmes tons pour leurs fenêtres affines contrairement à leurs images globales ou aux fenêtres extraites selon notre méthode. Cette variabilité au sein de chaque classe diminue alors la possibilité pour une méthode d'apprentissage (qui considère toutes les fenêtres de la même façon) d'établir des règles de discrimination précises. La façon de combiner les fenêtres affines avec une méthode d'apprentissage en général et une évaluation plus approfondie de cette approche sont une extension possible de ce travail.

[11] Par exemple, à partir de la première image de COIL-100, 201 régions sont détectées et 1040 fenêtres construites en 128×128 pixels. Si une rotation de 45°est appliquée à cette image, 177 régions sont alors détectées et 1048 fenêtres générées. Tandis qu'en dimensions 32×32 pixels, 44 régions sont détectées pour l'orientation originale et 132 fenêtres générées, mais seulement 22 régions sont détectées et 76 fenêtres sont construites pour une rotation d'image de 45°.

[12] Avec cette taille d'images, environ 118321 fenêtres sont générées par cette méthode car il y a seulement 100 images d'apprentissage.

Autres protocoles d'évaluation

D'autres expériences auraient pu être menées pour une évaluation de la robustesse plus approfondie. Notamment, des cas plus complexes où plusieurs transformations sont combinées n'ont pas été évalués. Plutôt que de simuler de tels cas en créant des images artificiellement transformées, nous préférons appliquer au sein du chapitre suivant notre méthode à des problèmes réels où plusieurs transformations et perturbations se présentent naturellement.

L'évaluation de la robustesse aurait également pu être réalisée sur d'autres bases de données. Aussi, l'utilisation de l'une ou l'autre extension de la méthode présentée dans cette section pourrait peut-être améliorer la précision de l'approche sur les problèmes présentés au chapitre précédent. Par exemple, l'utilisation de fenêtres d'orientations et de tailles aléatoires sur le problème de chiffres manuscrits (MNIST) pourrait peut-être améliorer la précision. Notamment, les chiffres de cette base de données présentent des différences d'inclinaisons. La spécialisation de la méthode pour tenir compte des particularités d'un problème n'est pas le sujet de cette thèse mais elle est néanmoins envisageable. Les extensions proposées peuvent être ainsi activées ou désactivées en fonction de la connaissance a priori des conditions d'acquisition des images.

5.5 Conclusions

Au cours de ce chapitre, nous avons poursuivi l'étude de notre approche en évaluant sa robustesse à une série de transformations qui peuvent se présenter entre les images d'apprentissage et les images de test. Une robustesse à ces perturbations permettrait d'étendre le champ d'application de la méthode à des problèmes dont les conditions d'acquisition sont moins contrôlées.

Nos expériences se sont principalement basées sur des images de la base de données COIL-100 auxquelles nous avons artificiellement appliqué des transformations. Selon cette étude, la méthode par extraction aléatoire de fenêtres s'est avérée plus robuste que la méthode de base travaillant sur les images globales pour toutes les perturbations considérées : changement de point de vue, d'échelle, d'orientation 2D, ainsi que la présence d'occultations partielles et d'encombrement. Sans utiliser de techniques particulières pour la robustesse, elle est donc robuste à de faibles transformations. Par ailleurs, l'utilisation de la représentation HSV au lieu du RGB offre globalement de meilleurs résultats car cette représentation est moins sensible aux changements d'illumination.

Après une revue des méthodes de la littérature, nous avons ensuite proposé d'étendre notre méthode par l'extraction de fenêtres transformées. Des fenêtres de tailles et d'orientations aléatoires sont extraites à partir des images d'apprentissage

et de test. Elles sont toujours décrites par les pixels qui les composent et classées par un ensemble d'arbres aléatoires. Ces extensions améliorent sensiblement la robustesse de la méthode par la simple application de transformations à toutes les fenêtres extraites, sans réaliser d'augmentation de l'échantillon de fenêtres d'apprentissage. De manière plus générale, nos résultats confirment ceux d'autres études qui atteignent une meilleure robustesse à diverses transformations par apprentissage.

Au sein du chapitre suivant, nous appliquerons notre méthode et ses extensions à des problèmes réels où plusieurs transformations sont naturellement combinées.

Chapitre 6

Autres applications

Dans ce chapitre, nous évaluons les performances de notre approche sur des problèmes où les variations entre les images d'une même classe sont en général plus importantes que dans le cas des problèmes considérés lors des chapitres précédents. Ces problèmes sont considérés plus complexes dans la littérature. Notre évaluation est réalisée sur trois bases de données (ETH-80, ZuBuD, WANG). Les résultats principaux sont repris dans [MGPW05].

6.1 La catégorisation et la recherche d'images

Dans la littérature, les notions de catégorisation et de recherche d'images englobent généralement des problèmes où l'apparence des images présente des variations importantes. Les différences entre les images d'apprentissage et de test sont généralement conséquentes pour ces problèmes. En effet, outre les variations qui peuvent résulter de transformations comme celles que nous avons vues au chapitre 5 (changements d'illumination, de point de vue, d'orientation, etc.), ces images peuvent notamment représenter de nouveaux objets voire des scènes ou paysages dont l'apparence varie considérablement par rapport aux images préalablement vues par le système. La classification de telles images est donc considérée comme un problème plus complexe que l'identification d'un objet isolé déjà rencontré par le système lors de la phase d'apprentissage (comme c'était le cas par exemple avec la base de données COIL-100).

Dans ce chapitre, nous proposons donc d'étudier les performances de notre méthode sur 3 problèmes de ce type qui représentent des images d'objets miniatures (ETH-80), des photographies d'immeubles d'une ville (ZuBuD) et de scènes ou paysages (WANG). Les images de ces bases de données étant déjà classées par un être humain comme pour les problèmes étudiés précédemment dans ce travail, nous pouvons directement appliquer notre approche pour réaliser la classification de ces images et évaluer sa précision en termes de taux d'erreur sur un ensemble de test.

6.2 Évaluation sur la base de données ETH-80

6.2.1 Présentation et protocole

La base de données Cogvis ETH-80[1] comporte des images de dimensions 128×128 pixels, en couleurs, de 80 objets répartis en 8 classes : pommes, voitures, poires, tomates, chiens, chevaux, vaches, tasses. Chaque classe compte 10 objets. Chaque objet est représenté par 41 images qui correspondent à autant de points de vue régulièrement espacés. La figure 6.1 montre une vue pour chacun des 80 objets de la base de données. Malgré le fait que des objets miniatures sont utilisés et non des images en conditions réelles, les variations au sein d'une même classe sont assez importantes. Par exemple, les animaux de chaque classe présentent individuellement des variations importantes de couleurs (les crins et poils des chevaux), de textures (le pelage des chiens) et de formes (l'attitude des vaches).

FIG. 6.1 – Base de données Cogvis ETH-80 : une vue pour chacun des 80 objets répartis en 8 classes.

Le protocole suggéré par les créateurs de cette base de données et auteurs de [LS03] est un "leave-one-out". Chaque étape de ce protocole consiste à réaliser un apprentissage sur tous les objets (toutes leurs vues) de la base de données à l'exception d'un seul objet (toutes ses vues). Chacun des 80 tests consiste alors à vérifier que les

[1] `http://www.vision.ethz.ch/projects/categorization/eth80-db.html`

41 vues de l'objet qui n'ont pas été utilisées lors de l'apprentissage sont classées dans la bonne classe parmi les 8 établies. Cette étape est donc répétée pour tous les objets de la base de données. Pour chacune des 8 classes, on effectue donc 10 apprentissages basés sur 9 objets $*$ 41 vues $+$ 7 classes $*$ 10 objets $*$ 41 vues $= 3239$ images. Après chaque apprentissage on teste chacune des 41 vues de l'objet restant. Le taux de reconnaissance moyen pour une classe est donc calculé sur $10 * 41 = 410$ vues.

6.2.2 Résultats

Le tableau 6.1 reprend les résultats de l'application de la méthode d'arbres aléatoires sur les images globales, la méthode par extraction de fenêtres locales (de taille fixe), et les résultats de plusieurs méthodes évaluées par [LS03]. Leur étude a comparé des approches qui utilisent soit la couleur, soit la forme globale ou locale, soit la texture des images. Certaines requièrent une segmentation dont les masques sont fournis avec la base de données mais que nous n'avons pas utilisés. Les images de cette base de données ont été acquises en conditions contrôlées, l'objectif premier étant d'évaluer des méthodes habituelles de classification en présence de variations importantes dues aux différents objets et au niveau d'abstraction des classes, avant de considérer leur utilisation en conditions réelles. La présence de variations supplémentaires d'illumination, d'échelle ou de rotation rendrait la tâche encore plus complexe. Les performances en telles situations ne sont pas étudiées par [LS03]. En plus de la méthode globale, nous utiliserons donc uniquement la version de base de notre méthode avec extraction de fenêtres.

Pour nos deux approche (les arbres aléatoires sur les images globales, et les arbres aléatoires avec extraction de fenêtres), nous avons utilisé les paramètres habituels. Le nombre d'arbres pour la méthode globale est de $T = 100$. Augmenter ce nombre (jusqu'à 500) n'a pas donné un meilleur taux d'erreur. Pour la méthode avec fenêtres, nous utilisons $T = 10$ arbres, $Nw = 120000$ fenêtres et la taille optimale des fenêtres a été déterminée (sur les 80 ensembles de test), elle est de 108×108 pixels. La figure 6.2 illustre la variation du taux d'erreur en fonction de la taille des fenêtres, pour l'ensemble de la base de données et pour chacune des classes en particulier [2]. Avec la méthode sur images globales, nous obtenons un taux d'erreur de 21.4% tandis que la méthode avec extraction de fenêtres donne 18.87%, en représentation RGB. Comme le montre le tableau 6.1, certaines classes sont facilement reconnues (pommes, poires, tomates), tandis que d'autres sont beaucoup moins bien distinguées (vaches et surtout chiens et chevaux). Les principales erreurs de notre méthode avec

[2] Étant donné le protocole utilisé qui implique la construction de 80 ensembles d'arbres pour l'évaluation de la précision de chaque taille de fenêtres, nous avons utilisé un seuil s_{th} de 0.01 pour la construction d'arbres aléatoires, ce qui réduit sensiblement les temps d'apprentissage. La taille optimale des fenêtres a été déterminée selon ce paramètre. Ensuite, cette taille a été utilisée avec le seuil de 0.1. Le taux d'erreur global est légèrement supérieur en 0.01 avec 22.47% contre 18.87% en 0.1 repris dans le tableau 6.1.

extraction de fenêtres proviennent de la confusion des quadrupèdes (chevaux, chiens, vaches) entre eux. Les erreurs liées aux voitures et tasses sont diverses. Les 5 erreurs pour les fruits résultent de la confusion de pommes (rouges) avec des tomates et d'une poire avec une pomme.

Les méthodes utilisées dans [LS03] présentent également des différences importantes d'une classe à l'autre. Globalement, la méthode basée sur des histogrammes de couleurs est la moins satisfaisante (35.15% d'erreurs), et la méthode utilisant des contours locaux est la plus précise (13.6% d'erreurs). Leur précision est variable d'une classe à l'autre. Par exemple, une méthode basée sur les contours distingue difficilement les pommes des tomates. Par contre, une méthode utilisant des histogrammes de couleurs permet de reconnaître les tomates, mais moins bien les pommes et les poires dont les tons sont souvent proches. Aucune méthode utilisée n'est donc supérieure pour toutes les classes. Les auteurs suggèrent alors de combiner différentes techniques pour une précision globalement meilleure. Notre méthode, qui a été évaluée précédemment sur des problèmes divers impliquant des formes, des objets colorés quelconques et des textures, n'est jamais la moins précise quelle que soit la classe considérée. Il est raisonnable de penser que le modèle construit par notre approche combine implicitement les notions de couleurs, de formes et de textures.

Classe	*Arbres aléatoires RGB*	*Arbres+Fenêtres RGB*	[LS03]
Pommes	4.89% (20/410)	0.98% (4/410)	42.44% à 11.71%
Voitures	21.46% (88/410)	15.12% (62/410)	37.07% à 0%
Vaches	30.49% (125/410)	26.1% (107/410)	37.56% à 5.61%
Tasses	13.17% (54/410)	10% (41/410)	33.9% à 0.24%
Chiens	50.73% (208/410)	46.34% (190/410)	65.37% à 17.07%
Chevaux	49.51% (203/410)	58.54% (214/410)	67.32% à 15.37%
Poires	0.98% (4/410)	0.24% (1/410)	33.9% à 0.24%
Tomates	0% (0/410)	0% (0/410)	32.32% à 1.46%
Total	21.40% (702/3280)	18.87% (619/3280)	35.15% à 13.60%

TAB. 6.1 – Base de données ETH-80 : taux d'erreur global et par classe pour les arbres aléatoires ($T = 100$), les arbres avec fenêtres ($w_1 = w_2 = 108$), et différentes méthodes évaluées par [LS03].

6.2.3 Autres protocoles d'évaluation

Comme dit précédemment, les images de cette base de données ne présentent pas de variations importantes (par exemple d'échelle) autres que celles résultant de la variabilité des objets regroupés au sein des classes. Ces transformations supplémentaires rendraient la tâche encore plus difficile. Comme nous l'avons vu, l'utilisation par exemple de fenêtres de tailles aléatoires permettrait d'être plus robuste aux éventuels changements d'échelle. L'évaluation de cette variante donne 25.49% d'erreur.

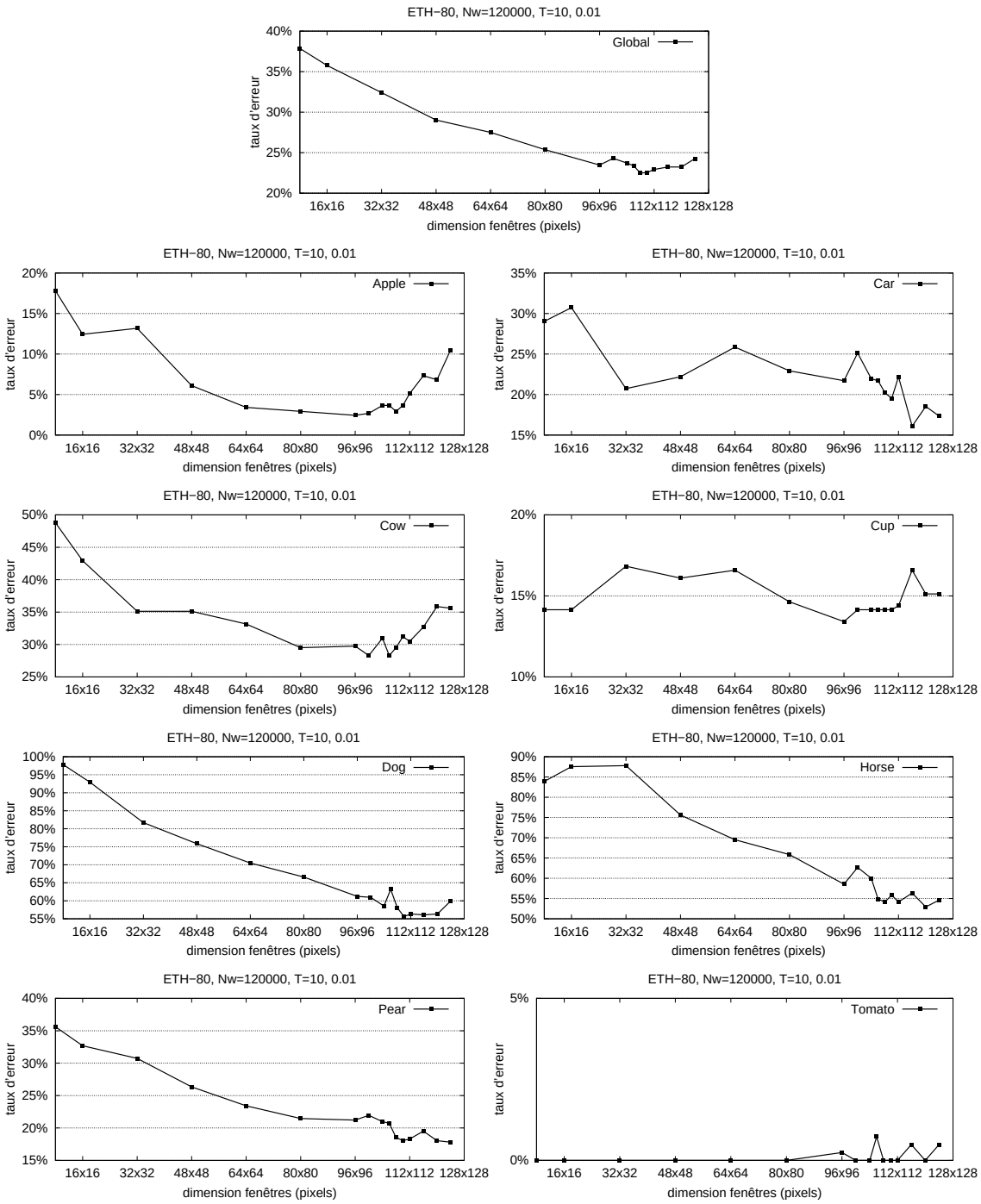

FIG. 6.2 – ETH-80 : influence de la dimension des fenêtres sur le taux d'erreur global (en haut) et pour chaque classe.

Une procédure d'évaluation supplémentaire consisterait à évaluer les modèles construits à partir des images d'objets miniatures sur des images acquises en condi-

tions réelles contenant un élément de ces classes. [CWN04] a réalisé une évaluation de ce type pour les classes de vaches et de voitures, avec un ensemble de test de 239 images. La méthode qui donne les meilleurs résultats combine les machines à support vectoriel avec la détection de points d'intérêt et descripteurs du type "local jet". Les résultats obtenus sont proches de 90% de reconnaissance, ce qui est un très bon résultat au regard des variations entre les images de test (acquises en conditions réelles) et les images d'apprentissage (acquises en milieu contrôlé et avec fond uniforme). Nos résultats de la section 6.4 montreront qu'il est possible d'apprendre à reconnaître des classes (par exemple des bus, chevaux et éléphants) à partir d'images acquises en conditions réelles.

6.3 Évaluation sur la base de données ZuBuD

6.3.1 Présentation et protocole

La base de données publique ZuBuD [SSV03][3] propose un ensemble d'images de bâtiments de Zurich. La figure 6.3 montre quelques unes de ces images. Cette base de données est divisée en un ensemble d'apprentissage composé de 201 immeubles représentés chacun par 5 images. L'ensemble de test comprend au total 115 images d'un sous-ensemble de ces immeubles photographiés selon d'autres conditions. Le but est donc de classer correctement ces 115 images dans l'une des 201 classes. La localisation d'un touriste parcourant les rues d'une ville par la reconnaissance du bâtiment le plus proche de lui est une application envisageable.

FIG. 6.3 – Quelques images de la base de données d'immeubles ZuBuD correspondant à cinq immeubles.

Les images de cette base de données présentent un nombre assez important de variations pour les images d'un même immeuble. Des sapins, branches d'arbres et parfois de la neige constituent différents types d'occultations partielles. Le ciel, la route et les voitures qui circulent autour des bâtiments représentent une forme d'encombrement. La position du photographe par rapport aux bâtiments photographiés impliquent des changements de point de vue parfois assez conséquents. Naturellement, la prise de photos implique de faibles rotations. En outre, certaines photographies ont été prises en orientation portrait plutôt que paysage, mais toutes les

[3] `http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html`

images sont fournies en orientation paysage, ce qui implique une rotation de 90° pour certaines d'entre elles (voir l'image à droite de la figure 6.3). La distance entre le photographe et le bâtiment est à l'origine de changements d'échelle et cet effet est augmenté par des résolutions d'images différentes entre les images d'apprentissage (en 640×480 pixels) et les images de test (en 320×240 pixels). Des changements d'illumination sont également présents puisque les photos ont été prises à différentes périodes de la journée et en différentes saisons de l'année. On peut également constater un léger flou pour certaines images, ce qui est une forme de bruit. Enfin, les images d'apprentissage et de test n'ont pas été prises avec le même appareil photo, et le format des images n'est pas identique (PNG pour l'échantillon d'apprentissage et JPG pour l'échantillon de test). La figure 6.4 illustre quelques unes de ces variations pour un même immeuble, notamment des changements d'illumination, de point de vue, d'échelle, et des occultations partielles.

FIG. 6.4 – Les images d'un immeuble de la base de données ZuBuD. En haut, les 5 images utilisées lors de l'apprentissage, en bas 4 images de test.

6.3.2 Résultats

La méthode de [OM03] classe parfaitement les 115 images de l'ensemble de test. Elle utilise les fenêtres locales affines décrites par coefficients DCT et le procédé de mise en correspondance dont nous avons parlé au sein de la section 5.3.2. L'approche de [GTV04] obtient 106 classifications correctes. Les deux méthodes évaluées dans [DKN04b] répondent correctement à 103 et 97 requêtes. La méthode de [SSTV03] identifie 99 images parmi les 115. La technique présentée dans [SSF+03] reconnaît 47 immeubles. En tolérant que la classe correcte soit parmi les 5 classes les plus probables prédites par le modèle, ces deux dernières méthodes reconnaissent respectivement 109 et 97 images. Étant donné le grand nombre de classes (201 immeubles) pour ce problème, cette tolérance est considérée comme acceptable.

Nous avons évalué plusieurs variantes de notre approche. La méthode d'arbres aléatoires combinés à l'extraction de fenêtres de taille fixe reconnaît 69 images parmi les 115, en représentation RGB. Les paramètres $T = 10$, $Nw = 120000$ et $w1 =$

$w2 = 16$ ont été utilisés, et 100 fenêtres sont extraites au sein de chaque image de test [4]. Avec le même nombre d'arbres construits et de fenêtres extraites, la variante avec extraction de fenêtres de tailles et d'orientations aléatoires (redimensionnées en 16×16 pixels) donne 33.91% d'erreurs, c'est-à-dire 76 images correctement classées, en RGB, et 18.26% (94 images) en représentation HSV. La variante avec extraction de fenêtres de tailles aléatoires (redimensionnées en 16×16 pixels) mais d'orientation fixe donne 4.35% d'erreur soit 110 images correctement classées parmi 115, en HSV.

En résumé, le meilleur résultat que nous obtenons résulte de l'utilisation de fenêtres de tailles aléatoires mais sans transformations de rotation, en combinaison avec la représentation HSV. Ce résultat est seulement inférieur à [OM03]. Comme nous l'avions remarqué lors de notre étude de la robustesse, l'utilisation supplémentaire de fenêtres d'orientations aléatoires dégrade les résultats lorsque les conditions d'acquisition n'impliquent pas de changements d'orientations importants, ce qui est le cas pour la majorité des images de cette base de données. La représentation HSV permet d'améliorer sensiblement les résultats étant donné les variations d'illumination.

À titre d'exemple, toutes les images de test de l'immeuble représenté à la figure 6.4 sont correctement classées. Malgré les perturbations et transformations manifestes, un certain nombre de régions des images de test peuvent être reconnues, ce qui permet la reconnaissance de l'immeuble par une méthode locale comme la nôtre. Pour chacune de ces 4 images, parmi les 1000 votes qui résultent de l'application de notre modèle de $T = 10$ arbres aux 100 fenêtres extraites, la bonne classe reçoit respectivement 132, 358, 403 et 344 votes. La deuxième classe prédite par le modèle et les suivantes reçoit dans les 4 cas moins de 70 votes. La figure 6.5 illustre quelques fenêtres de tailles aléatoires puis redimensionnées selon notre méthode, extraites à partir d'une image de test de la figure 6.4. La première ligne de fenêtres de cette figure donne des fenêtres qui sont individuellement correctement reconnues, c'est-à-dire pour lesquelles la classe correcte reçoit un nombre de votes numériquement supérieur parmi l'ensemble des $T = 10$ arbres. Pour cette image de test, 28 fenêtres sur les 100 extraites sont correctement classées (avec 5, 4, 3 ou 2 votes sur les 10 arbres). La deuxième ligne de fenêtres montre des fenêtres individuellement mal classées. Certaines de ces fenêtres reçoivent toutefois quelques votes corrects (les deux premières fenêtres de la deuxième ligne en reçoivent 3 sur 10, une autre classe recevant 4 votes pour ces fenêtres). Pour cette image de test, la deuxième classe prédite par le système reçoit au total 59 votes sur 1000, et 13 fenêtres sur les

[4] Une étude plus approfondie de la taille optimale n'a pas été réalisée sur ce problème car les changements d'échelle et de résolution d'images suggèrent d'utiliser des fenêtres de tailles aléatoires. Toutefois, on le voit avec cette taille de fenêtres 16×16 pixels, la méthode de base se comporte assez bien par rapport à la variante avec fenêtres de tailles et d'orientations aléatoires, malgré ces variations. Les images sont d'assez grandes tailles et les objets à reconnaître occupent une portion assez importante en leur sein. Il est probable que, quelle que soit la résolution, des zones homogènes se répètent (par exemple les portions de murs des immeubles) et les fenêtres qui y sont extraites sont correctement classées.

100 sont individuellement classées comme faisant partie de cette classe. On constate que plusieurs fenêtres extraites dans des régions occultées ou encombrées sont mal classées et les votes qui leur sont attribués sont généralement imprécis. Mais notre méthode qui totalise l'ensemble des votes pour toutes les fenêtres est ici robuste à ces perturbations.

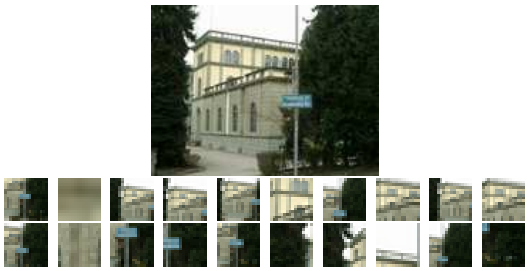

FIG. 6.5 – Quelques exemples de fenêtres de tailles aléatoires redimensionnées, pour une image de test de la base de données ZuBuD. Les fenêtres de la première ligne sont individuellement correctement classées contrairement à celles de la seconde.

Sur l'ensemble de test, les 5 images mal classées par le modèle sont données à la première ligne de la figure 6.6. La deuxième ligne de cette figure montre une image de la classe eronnément prédite pour ces images. Les trois premières images de test présentent le même immeuble qui a été photographié en mode paysage pour les images de test, alors que les images utilisées pour l'apprentissage (voir un exemple à la troisième ligne de la figure) de cette classe étaient orientées en mode portrait. La rotation de 90° qui en découle perturbe la classification. Ces 3 erreurs sont également présentes avec la variante qui génère des fenêtres d'orientations aléatoires. La variation très importante d'illumination rend en outre la tâche plus difficile. Pour chacune de ces trois images, le vecteur de probabilités produit par le modèle est particulièrement uniforme : moins de 35 votes sur 1000 sont attribués à la classe majoritaire. De nombreuses classes reçoivent un nombre de votes de cet ordre. Le même phénomène se produit pour la quatrième image mal classée. Cette image de test est également orientée en paysage alors que les images d'apprentissage pour cet immeuble étaient orientées en mode portrait.[5] De plus, quelques éléments présents dans les images d'apprentissage (les pointes au-dessus de l'immeuble) qui permettent de mieux reconnaître cet immeuble ne sont pas visibles dans cette image de test. L'immeuble erronément prédit présente plusieurs similitudes avec l'image présentée, particulièrement en termes de couleurs et de textures des murs.[6] La dernière image

[5]Les appareils numériques les plus récents permettent de conserver l'information d'orientation pour chacune des photos. Il est alors envisageable de décider d'une orientation pour un problème donné et d'appliquer une rotation aux images qui ne sont pas orientées de la sorte.

[6]Concernant ce problème d'orientation, nous avons constaté pour un des immeubles de la base de données que les deux orientations étaient représentées par ses images d'apprentissage. Dans l'ensemble de test, deux images le représentent, l'une en orientation paysage, l'autre en orientation portrait. Elles sont correctement reconnues par notre modèle.

pour laquelle le modèle se trompe est toutefois bien classée en rang 2. Un vote de moins seulement sur les 1000 est attribué à la bonne classe. Il n'est d'ailleurs pas évident pour un être humain d'affirmer rapidement que ces deux images représentent des immeubles différents.

FIG. 6.6 – Erreurs sur la base de données ZUBUD : les 5 images des immeubles non reconnues (en haut), une image d'apprentissage de l'immeuble prédit erronément par notre approche (au milieu), une image d'apprentissage de l'immeuble tel qu'il aurait dû être reconnu (en bas).

6.3.3 Espace mémoire et temps de calcul

Pour ce genre d'application, l'espace mémoire et les temps de calculs peuvent constituer des critères importants.

Pour la base de données ZuBuD, 1005 images en 640×480 au format PNG[7] composent l'ensemble d'apprentissage. L'ensemble de ces fichiers occupent 488Mb sur disque.

Selon notre approche, $Nw = 120000$ fenêtres redimensionnées en 16×16 pixels sont extraites à partir des 1005 images d'apprentissage. 904MB sont nécessaires pour stocker ces fenêtres en représentation HSV dans des fichiers "texte brut". La compression de la totalité de ces fichiers avec gzip occupe entre 250Mb et 300Mb.

Après extraction des fenêtres, notre approche construit un ensemble d'arbres. Pour le meilleur résultat que nous obtenons, la moyenne du nombre de nœuds par

[7]Ce format d'image intègre un algorithme de compression compatible avec l'outil de compression gzip. L'application de gzip aux images au format PNG de cette base de données ne réduit pas leur taille.

arbre de l'ensemble est de 211413. Dix arbres sont construits. L'encodage direct sur disque d'un arbre occupe environ 84Mb. L'utilisation directe d'un outil de compression comme gzip sur les fichiers d'arbres réduit cette taille entre 1,5Mb et 1,7Mb, c'est-à-dire au total entre 15Mb et 17Mb pour les 10 arbres. Contrairement à la plupart des méthodes qui utilisent une stratégie de reconnaissance par mesure de distance, une fois que les arbres sont construits, il n'est pas nécessaire de conserver les fenêtres extraites à partir des images d'apprentissage, ni les images elles-mêmes.

Sur cette base de données, les résultats présentés par [OM03] nécessitent de 1.3Gb à 290Mb pour conserver les représentations des 1005 images, selon la méthode de représentation choisie (valeurs de pixels ou nombre variable de coefficients DCT). Le meilleur résultat obtenu utilise 15 coefficients DCT par fenêtres locales affines, ce qui requiert 470Mb. Toutes ces fenêtres doivent être conservées pour la phase de prédiction d'une nouvelle image. Le temps nécessaire à la classification d'une image n'est pas donné par [OM03].

Pour cette application, le temps de prédiction d'une nouvelle image avec notre méthode implique tout d'abord l'extraction à des positions aléatoires de 100 fenêtres de tailles aléatoires ensuite redimensionnées. Chacune des fenêtres est propagée dans l'ensemble des 10 arbres. En moyenne, chaque arbre réalise moins de 20 tests aux nœuds internes avant d'attribuer une classe en un nœud terminal. Chaque test consiste à comparer un pixel avec un seuil. Après propagation des fenêtres dans l'ensemble d'arbres, la recherche du maximum du nombre de votes reçus par les 205 classes est l'opération qui achèvent la phase de prédiction.[8]

6.4 Évaluation sur la base de données WANG

6.4.1 Présentation et protocole

La base de données WANG [9] comporte des images en couleurs de 10 classes issues de la base de données COREL. Chaque classe est constituée de 100 images qui illustrent un des thèmes suivants : "villages et habitants d'Afrique", "plage", "bâtiments", "bus", "dinosaures", "éléphants", "fleurs", "chevaux", "montagnes et glaciers", "nourriture". La figure 6.7 présente une image de cette base de données par classe. Cette base de données est disponible publiquement et gratuitement. Elle comporte 1000 images, de taille 384×256 (ou 256×384 pour les images disposées verticalement).

[8] Nous ne communiquons pas ici de temps de calcul car l'implémentation utilisée dépend d'un utilitaire externe pour l'extraction de fenêtres de tailles aléatoires et celles-ci sont stockées dans des fichiers. En outre, un seul arbre à la fois est stocké en mémoire, ce qui implique des chargements supplémentaires. Le tableau 4.3 page 67 donnait des temps de prédiction de la méthode de base pour 4 bases de données, lorsque les fenêtres et les arbres étaient chargés en mémoire.

[9] `http://wang.ist.psu.edu/docs/related`

FIG. 6.7 – Quelques images de la base de données WANG : une image pour chaque classe.

Comme le montre la figure 6.8 pour la classe "plage", les images au sein d'une même classe présente une variabilité assez importante. Notamment, au sein de la classe "plage", on retrouve des images avec ou sans humains, selon un angle de vue au niveau du sol ou en hauteur, certaines images montrent principalement la mer, d'autres surtout le sable, certaines plages sont bordées par des rochers, des arbres, d'autres par des immeubles, etc. Les couleurs et illuminations varient également de manière significative au sein d'une même classe et les différentes classes peuvent partager des mêmes tons. En outre, certains éléments rendent l'appartenance à une classe moins triviale : des images de la classe "villages et habitants d'Afrique" présente des aliments, des images de la classe "plage" comporte des "rochers", des images de la classe "montagnes et glaciers" présentent un lac ou une rivière.

FIG. 6.8 – Quelques images de la classe "plage" de la base de données WANG.

Le protocole d'évaluation utilisé par [DKN04a] consiste à utiliser l'ensemble des images de la base de données à l'exception d'une seule pour l'apprentissage. L'image restante est alors classée par le modèle. Cette étape est répétée pour les 1000 images. Le résultat est donc un taux d'erreur en mode "leave-one-out". [CW04] propose un autre protocole qui consiste à séparer les images de la base de données en un ensemble d'apprentissage composé de 500 images (50 images par classe) tirées aléatoirement parmi les 1000, les images restantes étant utilisées pour l'ensemble de test. Cette procédure a été réalisée 5 fois et les résultats ont été moyennés. Ce protocole est moins fastidieux pour les approches qui nécessitent une phase d'apprentissage coûteuse en temps. En effet, le protocole de "leave-one-out" nécessite ici 1000 constructions de

modèles. La phase d'apprentissage de notre approche étant rapide [10], nous appliquerons ce protocole.

6.4.2 Résultats

Le travail de [DKN04a] compare 9 méthodes de la littérature selon le protocole "leave-one-out". Les résultats varient de 62.5% à 15.9% d'erreurs. Les différentes méthodes utilisent des histogrammes locaux ou globaux construits à partir des intensités de pixels, à partir des réponses de filtres (de Gabor et de Tamura) ou de caractéristiques invariantes, la comparaison de régions détectées après application de techniques de segmentation, la distance euclidienne entre les images globales (avec ou sans modèle de distortion), entre des fenêtres locales ou entre des caractéristiques de Fourier Mellin. Le meilleur résultat est obtenu par l'application d'une méthode d'histogrammes pondérés construits à partir de fonctions qui représentent des caractéristiques invariantes. L'utilisation d'histogrammes globaux construits à partir des valeurs de pixels donne 17.9% d'erreur. L'utilisation d'une mesure de distance euclidienne entre les valeurs de pixels des images donne 55.1% d'erreurs. Trois méthodes sont comparées dans [CW04], elles combinent les SVM à des histogrammes ou à la description de régions extraites par segmentation. Les taux d'erreur moyens présentés varient de 33.3% à 19.5%. Parmi la douzaine de méthodes appliquées à cette base de données, certaines travaillent à partir d'une représentation RGB des images, d'autres en LUV ou en HSV.

Selon le protocole "leave-one-out", l'application de notre méthode avec extraction de fenêtres de tailles et d'orientations aléatoires, en représentation HSV, donne 21.4% d'erreur, avec $T = 10$ arbres, $Nw = 120000$ fenêtres et un redimensionnement de toutes les fenêtres en 16×16 pixels. Si seule la taille est aléatoire, le résultat est de 15.9%, le même taux d'erreur que le meilleur présenté dans [DKN04a].

Le tableau 6.2 montre la répartition des erreurs par classes et le taux d'erreur global. La somme des éléments d'une ligne est égale à 100 c'est-à-dire le nombre d'images dans cette classe. Chaque ligne représente le nombre d'images de cette classe qui sont classées par notre modèle dans la classe de chaque colonne. Les classifications correctes sont celles qui se trouvent sur la diagonale. On observe que les images des classes "villages et habitants d'Afrique" et "plage" sont les moins bien classées, ce qui est également le cas pour [CW04] qui a utilisé une autre méthode et un autre protocole sur cette base de données. À titre d'exemple, les deux premières images de la figure 6.8 ne sont pas reconnues par notre modèle, tandis que les suivantes le sont. Les images des classes "Dinosaures" et "Chevaux" sont parfaitement classées. La classe "Fleurs" est la classe qui présente le troisième meilleur taux d'erreur. On constate un bon nombre d'erreurs entre les classes "Plage" (C1)

[10] En particulier, nous avons utilisé la variante (voir la section 2.3.3, page 21) des arbres aléatoires qui choisit un test aléatoire parmi un nombre fixé d'attributs choisis aléatoirement.

Classe Réelle/Prédite	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Taux erreur
C0 Afrique	**68**	3	6	2	1	4	0	3	0	13	32%
C1 Plage	5	**67**	7	1	0	4	0	2	13	1	33%
C2 Bâtiments	4	5	**80**	0	0	5	0	0	3	3	20%
C3 Bus	6	1	1	**83**	0	0	0	1	3	5	17%
C4 Dinosaures	0	0	0	0	**100**	0	0	0	0	0	0%
C5 Éléphants	1	0	2	0	0	**86**	0	6	2	3	14%
C6 Fleurs	0	0	1	2	0	0	**94**	2	0	1	6%
C7 Chevaux	0	0	0	0	0	0	0	**100**	0	0	0%
C8 Montagnes et glaciers	1	10	6	3	0	1	1	1	**77**	0	23%
C9 Nourriture	4	2	0	4	0	0	3	1	0	**86**	14%
Total											15.9% (159/1000)

TAB. 6.2 – Base de données WANG : taux d'erreur global et par classe pour les arbres aléatoires avec fenêtres de tailles aléatoires.

et "Montagnes et glaciers" (C8), ce qui est également observé par [CW04]. La figure 6.9 permet d'observer les 10 images de la classe "Montagnes et glaciers" mal classées en "Plage". On note la présence d'un ciel bleu, de lacs ou rivières, d'humains, ou de rochers aux tons jaunes (comme le sable d'une plage) ou bleus. Certaines images de cette classe contenant des rivières ou lacs sont toutefois bien classées par le modèle.

Les 13 images de la classe "Plage" erronément reconnues comme "Montagnes et glaciers" sont proposées par la figure 6.10. Certaines présentent des rochers, des petits cailloux, des collines et montagnes à l'horizon. On remarque aussi l'écume de mer assez importante sur certaines d'entre elles, ainsi que des nuages blancs. Les fenêtres de petites tailles qui sont extraites dans ces régions peuvent être confondues avec des fenêtres contenant de la neige. L'une des images mal classées présente un château de sable en forme de montagne ou pyramide, le sable sur cette photo est en outre très clair. La figure 6.11 montre une image de la classe "Plage" qui présente des nuages blancs et une écume de mer assez importante. Quelques fenêtres qui sont lui extraites et qui sont individuellement mal classées comme "Montagnes et glaciers" sont également illustrées. Pour cette image, 29 fenêtres sont individuellement bien classées, mais toutes les autres sont reconnues comme appartenant à la classe "Montagnes et glaciers".

Les 13 images de la classe "Afrique" qui sont mal classées comme faisant partie de la classe "Nourriture" sont illustrées à la figure 6.12. Elles sont toutes très colorées, ce qui était le cas aussi d'un bon nombre d'images de plats de nourriture comportant des fruits et légumes d'origines variées, comme le montre la figure 6.13. On remarque des costumes fabriqués à partir de plumes ou de plantes. On voit par exemple sur l'une de ces images (la première de la deuxième ligne de la figure), un habitant qui

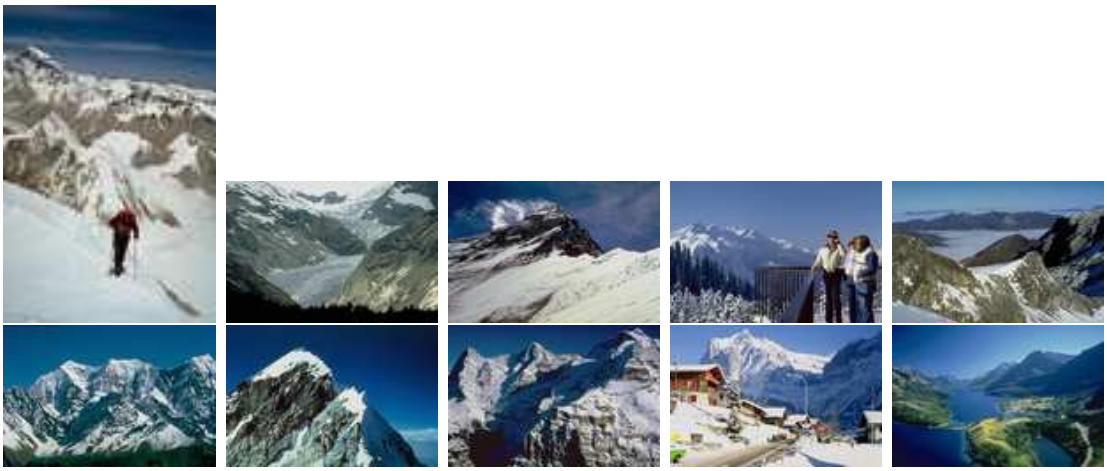

FIG. 6.9 – Base de données WANG : les 10 images de la classe “Montagnes et glaciers” incorrectement classées dans la classe “Plage”.

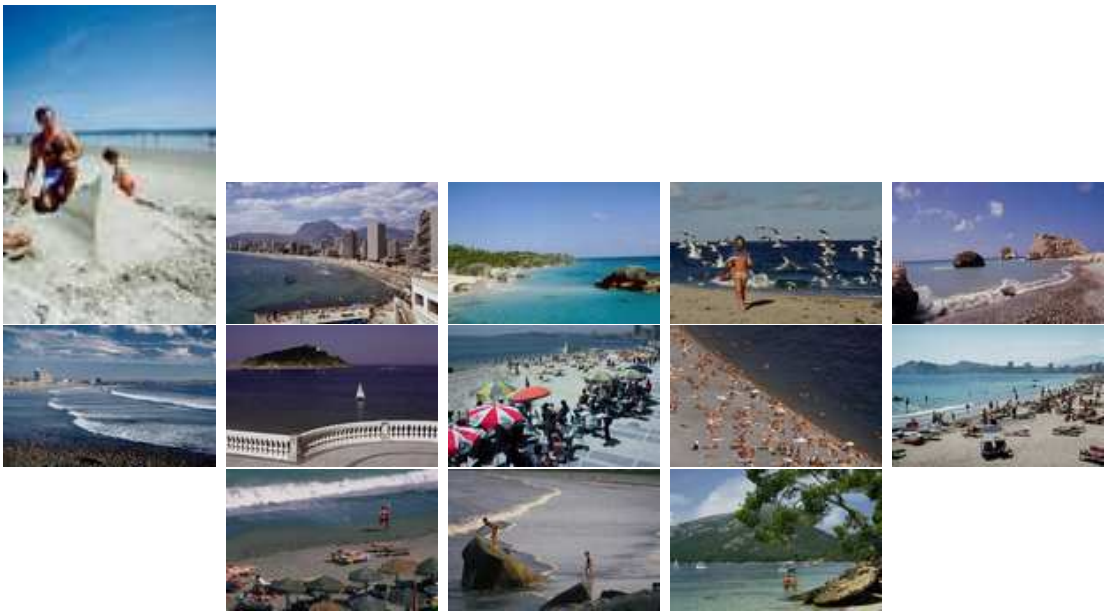

FIG. 6.10 – Base de données WANG : les 13 images de la classe “Plage” incorrectement classées dans la classe “Montagnes et glaciers”.

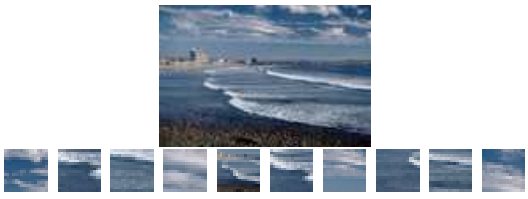

FIG. 6.11 – Base de données WANG : Une image de la classe “Plage” incorrectement classée dans la classe “Montagnes et glaciers”, ainsi que quelques fenêtres dont les votes contribuent à cette mauvaise classification.

FIG. 6.12 – Base de données WANG : les 13 images de la classe “Afrique” incorrectement classées dans la classe “Nourriture”.

FIG. 6.13 – Base de données WANG : Quelques images de la classe “Nourriture”

porte des bananes sur la tête, ce qui contribue peut-être à la mauvaise classification. Au sein de la classe “Nourriture”, plusieurs images comportent des fruits dont des bananes. Une autre image (à gauche de la première ligne de la figure 6.12) représente une table sur laquelle sont déposés des aliments.

6.5 Conclusions

Dans ce chapitre, nous avons appliqué notre approche à des problèmes considérés comme plus complexes dans la littérature. Ces trois problèmes (ETH-80, ZuBuD, WANG) présentent des variations entre les images d’apprentissage et de test plus importantes que dans le cas de la classification d’images d’objets individuels acquises en situations contrôlées.

Nos résultats se comparent avantageusement à de nombreuses méthodes et riva-

lisent avec ceux de l'état de l'art. Ils confirment que notre méthode peut s'appliquer avec succès à une large gamme de problèmes, même lorsque les images qui les composent présentent des apparences très variées. Nos résultats témoignent également de la robustesse de la méthode, et de ses variantes, à des perturbations et transformations présentes naturellement dans le cas d'images acquises en conditions réelles.

Chapitre 7

Conclusions

7.1 Résumé et contributions principales

Dans cette thèse, nous avons proposé une nouvelle méthode de classification d'images dont l'originalité provient de la combinaison de plusieurs techniques simples issues d'une part des récents progrès en apprentissage automatique et d'autre part de connaissances en vision par ordinateur. Nous avons montré empiriquement sur 7 problèmes divers de classification d'images que cette approche est automatique et précise. Des expériences supplémentaires ont confirmé que notre méthode est suffisamment robuste pour être appliquée à des problèmes réels. Notre évaluation a impliqué la construction de centaines de milliers d'arbres de décision et la génération et le traitement de centaines de gigas de données.

Dans un premier temps, nous avons proposé (chapitre 3) l'évaluation de méthodes d'apprentissage supervisé sur quatre problèmes typiques : la classification de chiffres manuscrits, de visages, d'objets 3D et de textures. Puisque les méthodes récentes d'apprentissage automatique sont capables de traiter des problèmes de très grandes dimensions, nous avons étudié leurs performances sur ces problèmes en les appliquant directement à la représentation pixels des images. Par nature, cette représentation est générique ; de plus elle est potentiellement très discriminante puisque aucune information des images n'est écartée a priori. Sept méthodes d'apprentissage supervisé ont donc été évaluées sur ces problèmes : arbre de décision, plus proches voisins, SVM et 4 méthodes d'ensembles d'arbres de décision (Bagging, Boosting, Random Forests, Extra-Trees). À l'exception de l'ajustement d'un nombre restreint de paramètres inhérents à chacune de ces méthodes, aucune adaptation importante de ces méthodes n'a été réalisée. Nous avons ainsi considéré que la classification d'images est un problème comme un autre de classification par apprentissage supervisé. Le modèle construit automatiquement par apprentissage est alors utilisé pour prédire la classe de nouvelles images. Cette approche contraste avec bon nombre d'approches existantes qui impliquent la construction ou la sélection d'attributs per-

tinents par un expert ou par des méthodes à ajuster, au préalable de l'application d'une méthode de classification comme les plus proches voisins. Les résultats qui ont été obtenus par notre première approche sont acceptables pour trois problèmes sur quatre. Mais les variations introduites par le dernier problème, la classification de textures, ont mis en évidence les limites d'une approche qui travaille à partir des images globales avec des attributs qui correspondent à des positions fixes au sein des images. La contribution de cette première étude est d'avoir montré la pertinence de l'application directe, à partir d'une représentation qui utilise les pixels des images, des récentes méthodes d'apprentissage automatique pour la classification d'image, tout en montrant ces limitations.

Suite à ces observations et suivant un schéma courant en classification d'images (tout en se distinguant par plusieurs aspects), nous avons proposé de combiner la méthode d'apprentissage par ensemble d'arbres aléatoires à l'extraction de fenêtres locales au sein des images (chapitre 4). Contrairement aux méthodes courantes, notre méthode réalise d'une part l'extraction de fenêtres à des positions aléatoires et ne dépend pas de la détection de points d'intérêt ou régions. D'autre part, ces fenêtres sont décrites par l'ensemble des pixels qui les composent et non par un nombre restreint d'attributs calculés en leur sein. Enfin, la classification de ces fenêtres individuelles par un ensemble d'arbres aléatoires et la combinaison des prédictions ainsi attribuées aux fenêtres d'une image constituent la version de base de la nouvelle méthode de classification d'images que cette thèse présente. L'évaluation sur les 4 problèmes étudiés jusqu'alors a montré une amélioration sensible de la précision de l'approche, en particulier pour le problème de textures pour lequel de mauvais résultats avaient été obtenus en suivant l'approche globale. Les opérations d'extraction de fenêtres étant très simples et la classification par arbres de décision impliquant un nombre raisonnable d'opérations, les temps de calculs sont très avantageux.

Bien que les 4 problèmes considérés présentent des images de nature différente, celles-ci ont été acquises en situations relativement contrôlées. En pratique, des perturbations peuvent modifier l'apparence des images et influencer la classification réalisée par un modèle construit à partir d'images acquises dans d'autres conditions. L'évaluation de notre nouvelle méthode a donc été poursuivie afin d'en étudier la robustesse à diverses transformations probables en situations réelles. La méthode a été évaluée en présence d'occultations partielles, de changements d'orientation 2D, d'échelle, et de point de vue. Elle est robuste à de faibles transformations de ce type et est aussi meilleure de ce point de vue que l'approche globale. Nous avons également pu vérifier que la représentation HSV des valeurs de pixels était plus robuste aux changements d'illumination que la représentation RGB. Une extension de la méthode a été proposée pour la robustesse au changement d'échelle et d'orientation. Elle génère aléatoirement des fenêtres selon ces deux transformations avant la phase d'apprentissage et lors de la phase de prédiction. Son évaluation a montré une amélioration sensible de la robustesse.

Enfin, la méthode a été évaluée sur des problèmes présentant des variations en-

core plus importantes dues à la classification des images en concepts sémantiquement plus larges ou suite à l'acquisition des images en situations réelles. La méthode a été évaluée sur trois bases de données récemment publiées : la classification de familles d'objets miniatures, la classification de bâtiments dans une ville, la classification d'images de thèmes dont l'apparence des images est très variable. Les images de ces différents problèmes présentent des changements importants de point de vue, d'échelle, d'orientation, des occultations partielles, de l'encombrement et des changements d'illumination. Notre méthode s'est révélée robuste aux variations ainsi introduites et offre des performances comparables aux meilleures méthodes de la littérature.

7.2 Perspectives

Notre thèse (et la méthode qu'elle présente) ouvre plusieurs voies en termes de méthode proprement dite mais aussi en termes d'applications.

7.2.1 Méthode et implémentation

La méthode proposée utilise des ensembles d'arbres aléatoires combinés à un procédé aléatoire d'extraction de fenêtres au sein des images qui sont décrites par les valeurs des pixels qui les composent. Comme nous l'avons vu au chapitre 4, notre méthode s'inscrit dans un schéma habituel pour la classification d'images. Ce schéma présente trois étapes. Notre méthode se distingue de toutes les méthodes existantes par au moins une de ces étapes. Il est toutefois envisageable de remplacer chacune de ces étapes par d'autres techniques existantes. L'utilisation de détecteurs [MTS+04] et de descripteurs robustes [MS03] pourrait être envisagée, en particulier pour tenter d'améliorer encore la robustesse de l'approche puisque ces techniques ont été conçues dans ce but. Nous avons initié une telle comparaison au sein de la section 5.4.3 en combinant les ensembles d'arbres aléatoires avec la méthode de fenêtres locales affines [OM02b]. Toutefois, le nombre de fenêtres extraites par cette méthode est très élevé et le code que nous avons utilisé nécessite que la totalité des données d'apprentissage soit dans un premier temps dans un même fichier texte qui doit ensuite être chargé totalement en mémoire. Ces détails d'implémentation ont limité nos expérimentations. Il serait intéressant de reprogrammer ce code, par exemple avec un système de cache, pour permettre le traitement d'un nombre plus élevé de fenêtres d'apprentissage.

Enfin, l'utilisation d'autres variantes des ensembles d'arbres aléatoires ou d'autres méthodes d'apprentissage est possible. Dans cette thèse, notre choix s'est porté sur les ensembles d'arbres aléatoires étant donné leur efficacité et leur bonne précision. Toutefois, les méthodes de Boosting ou de SVM sont parmi les méthodes les plus

précises de la littérature. Au sein du chapitre 3 leur utilisation s'est avérée plus précise que les arbres aléatoires sur les images globales pour certains problèmes, mais les temps de calcul étaient davantage importants. [GEW04] a également proposé de nouvelles variantes des ensembles d'arbres aléatoires. Les futures variantes et nouvelles méthodes issues de la communauté d'apprentissage automatique supervisé pourront facilement être intégrées dans notre schéma de classification d'images.

7.2.2 Robustesse

Nous avons réalisé l'évaluation de la robustesse de notre approche par rapport à plusieurs transformations d'une part en transformant des images acquises en conditions contrôlées (voir le chapitre 5), d'autre part en appliquant l'approche sur des images réelles (voir le chapitre 6). Certaines images présentent des variations importantes au sein d'une même classe. Les résultats que nous avons obtenus sur ces différents problèmes sont comparables aux meilleurs résultats de la littérature et ils montrent que notre méthode peut être appliquée à une large gamme de problèmes. Toutefois, ces résultats ne sont pas parfaits et il nous semble qu'un être humain peut classer ces images avec davantage de précision (avec, peut-être, plus de temps). Il doit donc être possible de mieux classer ces images. Sur ces problèmes, il pourrait être intéressant de comparer la combinaison des différentes techniques existantes pour la détection de régions, leur description et l'apprentissage d'un modèle. Les récentes techniques de détection [MTS+04] et de description de régions ont été conçues de manière à améliorer la robustesse à diverses transformations, mais elles n'ont encore été combinées qu'à une méthode de classification par plus proches voisins.

En outre, la robustesse à toutes les transformations possibles n'a pas été systématiquement évaluée. Aussi, la combinaison de plusieurs transformations n'a pas été simulée, nous avons préféré évaluer la méthode sur des images où des perturbations apparaissent naturellement, comme c'est le cas avec les bases de données ZuBuD et WANG. Bien que plusieurs de ces transformations sont présentes dans les bases de données que nous avons utilisées, la robustesse à des changements plus importants d'illumination et au bruit (induit par un flou ou une compression d'image conséquents) reste à évaluer, de même que la tolérance à des encombrements plus grands. La première étape de ces travaux futurs consistera donc à évaluer la méthode telle quelle sur de nouvelles bases de données encore plus complexes qui commencent à être proposées à la communauté. Nous pensons notamment à la base de données "Butterflies" [1] très récemment proposée (septembre 2004) qui présente des familles de papillons sur fonds encombrés. La base de données NORB [2] proposée à la même époque inclut des images de familles d'objets avec encombrement, la base de données "Object Recognition Database" [3] est également intéressante pour la reconnaissance

[1] http://www-cvr.ai.uiuc.edu/ponce_grp/data/
[2] http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/
[3] http://www-cvr.ai.uiuc.edu/ponce_grp/data/

d'objets multiples avec encombrement important, de même que la base de données utilisée par [DKP+03].

Pour améliorer encore la robustesse, il pourrait être intéressant de générer davantage de fenêtres aléatoirement transformées suivant plusieurs paramètres, notamment pour une meilleure robustesse à l'illumination (en incluant des fenêtres dont la luminosité a été transformée) et au bruit (en générant des fenêtres ayant subi différents types de bruit). Une nouvelle implémentation de notre code devrait permettre de traiter plusieurs centaines de milliers de fenêtres d'apprentissage sur un ordinateur classique.

7.2.3 Objets multiples et détection

Le problème de la reconnaissance d'objets multiples au sein d'une même image (voir section 5.2.6) devrait également être abordé plus en profondeur. Nous avons vu que l'application directe de notre approche sur quelques images contenant plusieurs objets ne permettait pas toujours de déterminer le nombre d'objets présents ni de les reconnaître parfaitement. En outre, la présence de plusieurs objets d'une même classe n'est pas détectable selon ce schéma.

Des tests plus systématiques devront être réalisés, par exemple sur les bases de données citées ci-dessus. Différentes approches sont envisageables pour cette problématique. Tout d'abord, des techniques de segmentation existantes permettent d'extraire à partir des images les objets qui la composent avant leur classification individuelle. Toutefois, il n'est pas toujours aisé d'extraire un objet en présence d'un fond encombré ou d'une image faiblement contrastée.

Une autre méthode consisterait à détecter les régions de l'image qui reçoivent des votes similaires. Il s'agirait tout d'abord de conserver la position des fenêtres extraites au sein de l'image de test et les votes qui leur sont attribués. Ensuite, en combinant les votes des fenêtres qui se trouvent dans un certain voisinage, on peut déterminer si la région ainsi considérée contient un même objet. La taille minimale ou maximale des régions est à étudier. Cette approche devrait permettre de localiser des objets au sein d'une image.

7.2.4 Recherche d'images

La recherche d'images similaires dans une base de données ("image retrieval") est une autre famille de problèmes présentée dans la littérature en vision par ordinateur. Un système de ce type recherche généralement des images similaires à celle présentée par l'utilisateur, sans que les images aient été nécessairement classées au préalable. Au sein du chapitre 6, nous avons appliqué notre méthode à un problème présenté comme tel dans la littérature (WANG). Nous avons considéré ce problème comme

un problème de classification où toutes les images sont déjà classées, ce qui n'est pas toujours le cas en pratique. Pour un problème où cette information de classification n'est pas disponible, le but est de retrouver des images proches de l'image proposée par l'utilisateur et celui-ci doit pouvoir indiquer ses propres critères de similarité au fur et à mesure en interagissant avec le système. Habituellement, il lui est possible de pondérer les images qui ont été retournées par le programme.

L'adaptation de notre approche au problème de la recherche d'images similaires est une perspective possible et intéressante. D'une part, les ensembles d'arbres aléatoires peuvent être construits de manière non supervisée [GEW04]. C'est-à-dire qu'ils peuvent être construits totalement aléatoirement, sans tenir compte de l'information de la classe de sortie aux nœuds internes de l'arbre. On peut alors envisager d'utiliser un ensemble d'arbres aléatoires comme une structure de données pour le stockage d'images dans laquelle les feuilles des arbres sont étiquetées par le nom des images qui ont été amenées dans cette feuille par construction. Lorsque l'on veut obtenir les images proches d'une nouvelle image, on peut propager cette image dans l'ensemble d'arbres et déterminer sa similarité en comptant le nombre de fois que l'image tombe dans la même feuille (ou dans un même sous-arbre parent) que les images de la base de données utilisées pour construire les arbres. Les images d'apprentissage les plus proches selon cette mesure sont alors retournées à l'utilisateur. Il est également possible d'appliquer cette approche pour déterminer la similarité de deux images sur base de la similarité de ses sous-fenêtres.

Il semble en outre intéressant de réaliser la comparaison algorithmique et empirique de cette approche avec d'autres méthodes de recherche d'images basées sur la méthode des plus proches voisins et des structures arborescentes.

7.2.5 Problèmes de régression

En apprentissage supervisé, on parle d'un problème de régression plutôt que de classification lorsque la variable de sortie à prédire est à valeur numérique plutôt que symbolique. Des variantes des méthodes d'arbres de décision existent pour traiter ces problèmes. Il est donc également envisageable d'appliquer notre approche à des problèmes de ce type, comme l'estimation de la pose d'un visage ou d'un objet.

7.2.6 Interprétabilité et régions pertinentes

Au sein d'une image de test, par souci d'interprétabilité, il peut être intéressant de déterminer quelles sont les régions qui comportent le plus d'information et qui ont permis de classer correctement l'image. Pour cela, on peut utiliser les fenêtres individuelles qui ont été bien classées. Il est également envisageable d'ordonner les fenêtres extraites au sein d'une image de test par le calcul de l'entropie du vecteur de probabilité de classes : les fenêtres qui ont été classées avec le plus de certitude sont

les plus pertinentes pour une classe donnée. Cette extension est directement réalisable puisque les votes attribués par les différents arbres aux fenêtres individuelles sont disponibles.

7.2.7 Applications

La méthode que nous avons proposée a été évaluée sur 7 problèmes divers de classification d'images. Ils impliquent la classification de visages, de chiffres manuscrits, d'objets 3D, de textures, d'immeubles, de paysages, de situations réelles, etc. Nous avons montré par ces études empiriques que le champ d'application de la méthode était large. Au sein de l'introduction, nous avons par ailleurs énuméré quelques exemples d'applications de la classification d'images dans de nombreux domaines variés : médecine, biologie, agriculture, bureautique, géologie, astronomie, environnement, processus industriels, tourisme, art, biométrie, etc. Notre méthode se comportant sur 7 problèmes variés de manière comparable à des méthodes souvent adaptées au problème considéré, il est raisonnable d'envisager son application aux nombreux problèmes de classification d'images qui existent et existeront. Son caractère générique et automatique permet d'obtenir rapidement des premiers résultats sur tout type de problème. Par exemple, citons quelques applications réelles qui nous ont été suggérées pour lesquelles il n'existe pas de méthode de classification spécifique connue a priori et pour lesquelles une méthode automatique et générique comme la nôtre pourrait directement être appliquée : la classification de logos sur des bouchons de bouteilles de vin, la classification de poudres élémentaires pour améliorer la composition de médicaments, la classification de défauts sur des réflecteurs de phares de voiture, etc.

D'autre part, les résultats obtenus dans cette thèse confirment l'applicabilité d'une méthode d'apprentissage automatique, augmentée d'une technique simple d'extraction de fenêtres, à des problèmes complexes comme ceux de classification d'images, et cela en utilisant directement l'information de bas niveau (les pixels). Il semble intéressant d'envisager l'application de ce genre de combinaison à d'autres problèmes complexes. Une approche similaire a été appliquée à la classification de signaux temporels [Geu02]. La classification de textes pourrait elle aussi bénéficier d'une généralisation de la technique d'extraction de régions. Au lieu d'extraire des régions voisines de pixels pour classer une image, on peut envisager d'extraire des phrases ou suites de mots au sein d'un texte pour le classer par la combinaison des prédictions de ses parties. Les problèmes de classification présentant des objets décrits par un très grand nombre d'attributs structurés sont nombreux. La classification de séquences génétiques (ou de données biologiques au sens large) est un autre exemple d'application qui pourrait bénéficier d'une telle approche.

Bibliographie

[AHE03] S. Alma'adeed, C. Higgins, and D. Elliman. Off-line recognition of handwritten arabic words using hidden Markov models. In *Proc. 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, December 2003.

[AMN+98] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45 :891–923, 1998.

[AZC01] M.-L. Antonie, O. R. Zaïane, and A. Coman. Application of data mining techniques for medical image classification. In *Proc. of Second Intl. Workshop on Multimedia Data Mining (MDM/KDD'2001) in conjunction with Seventh ACM SIGKDD*, pages 94–101, August 2001.

[BAS+98] M. C. Burl, L. Asker, P. Smyth, U. M. Fayyad, P. Perona, L. Crumpler, and J. Aubele. Learning to recognize volcanoes on Venus. *Machine Learning*, 30(2-3) :165–194, 1998.

[Bau00] A. Baumberg. Reliable feature matching across widely separated views. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[BFOS84] L. Breiman, J. Friedman, R. Olsen, and C. Stone. *Classification and Regression Trees*. Wadsworth International (California), 1984.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2) :123–140, 1996.

[Bre01] L. Breiman. Random forests. *Machine learning*, 45 :5–32, 2001.

[BS97] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of SVM. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 375. The MIT Press, 1997.

[CL03] C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. Technical report, Computer Science and Information Engineering, National Taiwan University, 2003.

[Cor02] A. Cornuéjols. Une nouvelle méthode d'apprentissage : Les SVM, séparateurs à vaste marge. *Bulletin de l'AFIA*, (51), juin 2002.

[CST00] C. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*. MIT Press, Cambridge, MA, 2000.

[CW04] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5 :913–939, August 2004.

[CWN04] B. Caputo, C. Wallraven, and M. E. Nilsback. Object categorization via local kernels. In *Proc. 17th International Conference on Pattern Recognition (ICPR)*, August 2004.

[DHD00] R. Duda, P. Hart, and S. D.G. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000.

[DKN01] J. Dahmen, D. Keysers, and H. Ney. Combined classification of handwritten digits using the 'virtual test sample method'. In *Proc. Second International Workshop, MCS 2001 Cambridge, UK*, pages 99–108, July 2001.

[DKN04a] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval : A quantitative comparison. In *Proc. 26th DAGM Symposium on Pattern Recognition (DAGM 2004)*, volume LNCS 3175, pages 228–236, August/September 2004.

[DKN04b] T. Deselaers, D. Keysers, and H. Ney. Classification error rate for quantitative evaluation of content-based image retrieval systems. In *Proc. 17th International Conference on Pattern Recognition (ICPR)*, 2004.

[DKP+03] T. Deselaers, D. Keysers, R. Paredes, E. Vidal, and H. Ney. Local representations for multi-object recognition. In *Proc. 25th DAGM Symposium*, volume LNCS 2781 of *Computer Science*, pages 305–312, September 2003.

[DKS03] J. X. Dong, A. Krzyzak, and C. Suen. High accuracy handwritten chinese character recognition using support vector machine. In *Proc. of International Workshop on Artitifical Neural Networks on Pattern Recognition*, September 2003.

[DS02] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3) :161–190, 2002.

[dSG02] E. M. dos Santos and H. M. Gomes. A comparative study of polynomial kernel SVM applied to appearance-based object recognition. In *Proc. First International Workshop on Pattern Recognition with Support Vector Machines*, pages 408–418, London, UK, 2002. Springer-Verlag.

[EB90] S. Edelman and H. H. Bülthoff. Viewpoint-specific representations in three-dimensional object recognition. Technical Report AIM-1239, MIT, 1990.

[EC04] J. Eichhorn and O. Chapelle. Object categorization with SVM : kernels for local features. Technical report, Max Planck Institute for Biological Cybernetics, 2004.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *Ai Magazine*, 17 :37–54, 1996.

[FRS96] Y. Freund and E. Robert Schapire. Experiments with a new boosting algorithm. In *Proc. Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[FSC98] G. D. Finlayson, B. Schiele, and J. L. Crowley. Comprehensive colour image normalization. *Lecture Notes in Computer Science*, 1406 :475–490, 1998.

[FvDFH93] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practices*, chapter Achromatic and Colored Light, pages 593–594. Addison-Wesley, 1993.

[FWD93] U. M. Fayyad, N. Weir, and S. Djorgovski. Skicat : A machine learning system for automated cataloging of large scale sky surveys. In *Proc. of the Tenth International Conference on Machine Learning (ICML-93)*, pages 112–119, 1993.

[Geu02] P. Geurts. *Contributions to decision tree induction : bias/variance tradeoff and time series classification.* Phd. thesis, Department of Electrical Engineering and Computer Science, University of Liège, May 2002.

[Geu03] P. Geurts. Extremely randomized trees. Technical report, Department of Electrical Engineering and Computer Science, University of Liège, Belgium, June 2003.

[GEW04] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Submitted to Journal of Machine Learning Research*, 2004.

[GFdS+04] P. Geurts, M. Fillet, D. de Seny, M.-A. Meuwis, M.-P. Merville, and L. Wehenkel. Proteomic mass spectra classification using decision tree based ensemble methods. *Submitted to Bioinformatics*, 2004.

[GGVC03] P. Granitto, P. Garralda, P. Verdes, and H. Ceccatto. Boosting classifiers for weed seeds identification. *Journal of Computer Science Technology*, 3(1), April 2003.

[GKL04] P. Geurts, I. E. Khayat, and G. Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Proc. of IEEE International Conference on Data Mining (ICDM-2004)*, 2004.

[GLC00] G.-D. Guo, S. Li, and K. Chan. Face recognition by support vector machines. In *Proc. International Conference on Automatic Face and Gesture Recognition, 196-201.*, 2000.

[GS99] T. Gevers and A. W. Smeulders. Color based object recognition. *Pattern Recognition*, 32 :453–464, March 1999.

[GTV04] T. Goedeme, T. Tuytelaars, and L. Van Gool. Fast wide baseline matching for visual navigation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 1, pages 24–29, 2004.

[GZ01] G.-D. Guo and H.-J. Zhang. Boosting for fast face recognition. In *Proc. IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 96–100, 2001.

[HF00] M. Hoque and M. C. Fairhurst. A moving window classifier for off-line character recognition. In *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam*, pages 595–600, September 2000.

[How02] N. Howe. Boosted image classification : An empirical study. In *ICML Workshop on Machine Learning in Computer Vision*, July 2002.

[HVP02] B. Heisele, A. Verri, and T. Poggio. Learning and vision machines. In *Proceedings of the IEEE, Visual Perception : Technology and Tools*, volume 90, pages 1164–1177,, 2002.

[JD00] D. Jugessur and G. Dudek. Local appearance for robust object recognition. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2000.

[JF96] B. Jedynak and F. Fleuret. Reconnaissance d'objets 3d à l'aide d'arbres de classification, 1996.

[KCPFT02] C. Kamath, E. Cantu-Paz, I. Fodor, and N. Tang. Classifying bent-double galaxies. *Computing in Science and Engineering*, 4(4) :52–60, 2002.

[KDN01] D. Keysers, J. Dahmen, and H. Ney. Invariant classification of red blood cells. In *Proc. Bildverarbeitung für die Medezin (BVM)*, pages 367–371, March 2001.

[KGN04] D. Keysers, C. Gollan, and H. Ney. Local context in non-linear deformation models for handwritten character recognition. In *Proc. International Conference on Pattern Recognition*, 2004.

[KHM98] M. Kubat, R. C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3) :195–215, 1998.

[KJPK02] K. Kim, K. Jung, S. Park, and H. Kim. Support vector machines for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11) :1542–1550, 2002.

[KKN04] T. Kölsch, D. Keysers, and H. Ney. Enhancements for local feature based image classification. In *Proc. International Conference on Pattern Recognition*, 2004.

[KPNV02] D. Keysers, R. Paredes, H. Ney, and E. Vidal. Combination of tangent vectors and local representations for handwritten digit recognition. In *Proc. 4th International workshop on Statistical Techniques in Pattern Recognition*, 2002.

[KS04] Y. Ke and R. Sukthankar. Pca-sift : A more distinctive representation for local image descriptors. In *Proc. Computer Vision and Pattern Recognition*, 2004.

[Kvå87] T. Kvålseth. Entropy and correlation : Some comments. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-17(3) :517–519, 1987.

[KvD87] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55 :367–375, 1987.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11) :2278–2324, 1998.

[LBH04] Y. LeCun, L. Bottou, and J. HuangFu. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. of Computer Vision and Pattern Recognition*, Washington, D.C., 2004. IEEE.

[LGF01] Y. B. Lauzière, D. Gingras, and F. P. Ferrie. A model-based road sign identification system. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 2001.

[LGTB97] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition : A convolutional neural network approach. *IEEE Transactions on Neural Networks*, 8(1) :98–113, 1997.

[Low99] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[LPF04] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2004)*, June 2004.

[LS03] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, WI, June 2003.

[LW04] J. Li and J. Z. Wang. Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Transactions on Image Processing*, 13(3) :340–353, 2004.

[Mah04] A. Mahy. Classification automatique d'images. Master's thesis, Institut Montefiore, Université de Liège, Belgique, 2004.

[MENW99] S. Muller, S. Eickeler, C. Neukirchen, and B. Winterstein. Segmentation and classification of hand-drawn pictograms in cluttered scenes - an integrated approach. In *Proc. IEEE Intern. Conference on Acoustics, Speech, and Signal Processing*, pages 3489–3492, March 1999.

[MGPW04] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. A generic approach for image classification based on decision tree ensembles and local subwindows. In *Proc. 6th Asian Conference on Computer Vision*, January 2004.

[MGPW05] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005.

[MGV+03] R. Marée, P. Geurts, G. Visimberga, J. Piater, and L. Wehenkel. A comparison of generic machine learning algorithms for image classification. *Proc. 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Research and Development in Intelligent Systems*, XX, 2003.

[MGW03] R. Marée, P. Geurts, and L. Wehenkel. Une méthode générique pour la classification automatique d'images à partir des pixels. *Proc. XXX-Vèmes Journées de Statistique 2003, Session spéciale "Méthodes statistiques pour l'image"*, 2003.

[MMFB04] M. Martinelli, S. Merler, C. Furlanello, and L. Bruzzone. A regularized version of AdaBoost for pattern classification in historic air photographs. In *V Meeting degli Utenti Italiani di GRASS*, Padova, Italy, 5–6 February, 2004.

[MN95] H. Murase and S. K. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14(1) :5–24, 1995.

[MO04] J. Matas and S. Obdržálek. Object recognition methods based on transformation covariant features. In *Proc. 12th European Signal Processing Conference (EUSIPCO 2004)*, Vienna, Austria, September 2004.

[MPV02] T. Mäenpää, M. Pietikäinen, and J. Viertola. Separating color and pattern information for color texture discrimination. In *Proc. 16th International Conference on Pattern Recognition*, 2002.

[MS02] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002. Copenhagen.

[MS03] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 257–263, June 2003.

[MTS+04] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Submitted to International Journal of Computer Vision*, 2004.

[NH99] A. Nefian and M. Hayes. Face recognition using an embedded HMM. In *Proc. IEEE Conference on Audio and Video-based Biometric Person Authentication*, pages 19–24, March 1999.

[OI97] K. Ohba and K. Ikeuchi. Detectability, uniqueness and reliability of eigen windows for stable verification of partially occluded objects.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9) :1043–1048, September 1997.

[OM02a] S. Obdržálek and J. Matas. Local affine frames for image retrieval. In M. S. Lew, N. Sebe, and J. P. Eakins, editors, *Proc. Image and Video Retrieval, International Conference (CIVR)*, volume 2383, 2002.

[OM02b] S. Obdržálek and J. Matas. Object recognition using local affine frames on distinguished regions. In *Electronic Proceedings of the 13th British Machine Vision Conference, University of Cardiff*, 2002.

[OM03] S. Obdržálek and J. Matas. Image retrieval using local compact DCT-based representation. In *Proc. 25th DAGM Symposium, Magdeburg, Germany*, volume 2781. Springer, 2003.

[OMP+02] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllönen, and S. Huovinen. Outex - new framework for empirical evaluation of texture analysis algorithms. *Proc. 16th International Conference on Pattern Recognition, Quebec, Canada, 1 :701-706*, 2002.

[OSI98] K. Ohba, Y. Sato, and K. Ikeuchi. Appearance based visual learning and object recognition with illumination invariance. In *Proc. Third Asian Conference on Computer Vision*, volume 2, pages 424–431, 1998.

[PA03] D. T. Pham and R. J. Alcock. *Smart Inspection Systems - Techniques and Applications of Intelligent Vision*. Elsevier Science, 2003.

[PE90] T. Poggio and S. Edelman. A network that learns to recognize 3d objects. *Nature*, (343) :263–266, 1990.

[PKL+02] R. Paredes, D. Keysers, T. M. Lehmann, B. B. Wein, H. Ney, and E. Vidal. Classification of medical images using local representations. In *Bildverarbeitung fur die Medizin*, pages 171–174, 2002.

[PPC01] R. Paredes and A. Perez-Cortes. Local representations and a direct voting scheme for face recognition. In *Proc. 1st International Workshop on Pattern Recognition in Information Systems*, pages 71–79, July 2001.

[PV98] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6) :637–646, 1998.

[PVG01] E. Poisson and C. Viard-Gaudin. Réseaux de neurones à convolution : reconnaissance de l'écriture manuscrite non contrainte. In *Proc. Journées Valgo*, number 01-02, Octobre 2001.

[Rav04] S. Ravela. Shaping receptive fields for affine invariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2004)*, june 2004.

[RDCFG03] M. Rodriguez-Damian, E. Cernadas, A. Formella, and A. Gonzales. Automatic identification and classification of pollen of the urticaceae

family. In *Proc. of Advanced Concepts for Intelligent Vision Systems (ACIVS 2003)*, pages 38–45. Ghent University, Belgium, September 2003.

[RG04] S. Ravela and L. Gamble. On recognizing individual salamanders. In *Proc. 6th Asian Conference on Computer Vision*, January 2004.

[RSQZ04] A. A. Rad1, R. Safabakhsh1, N. Qaragozlou1, and M. Zaheri. Iris recognition based on random texture analysis. In *Proc. 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Research and Development in Intelligent Systems*, 2004.

[RZE01] D. Roobaert, M. Zillich, and J.-O. Eklundh. A pure learning approach to background-invariant object recognition using pedagogical support vector learning. In *Proc. Computer Vision and Pattern Recognition Conference*, December 2001.

[Sch96] C. Schmid. *Appariement d'images par invariants locaux de niveaux de gris. Application à l'indexation d'une base d'objets.* PhD thesis, Institut National Polytechnique de Grenoble, 1996.

[SK87] L. Sirovich and M. Kirby. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3) :519–524, 1987.

[SLDV00] P. Y. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2000.

[SM97] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *Pattern Analysis and Machine Intelligence*, 19(5) :530–534, 1997.

[SSF+03] H. Shao, T. Svoboda, V. Ferrari, T. Tuytelaars, and L. Van Gool. Fast indexing for image retrieval based on local appearance with re-ranking. In *Proc. IEEE International Conference on Image Processing (ICIP 2003)*, September 2003.

[SSP03] P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. Intl. Conf. Document Analysis and Recognition*, pages 958–962, 2003.

[SSTV03] H. Shao, T. Svoboda, T. Tuytelaars, and L. Van Gool. HPAT indexing for fast object/scene recognition based on local appearance. In *Proc. International Conference on Image and Video Retrieval (CIVR)*, pages 71–80, July 2003.

[SSV03] H. Shao, T. Svoboda, and L. Van Gool. Zubud - Zurich building database for image based recognition. Technical Report TR-260, Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland, 2003.

[SZ02] F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. In *Proceedings of the Challenge of Image and Video Retrieval, London*, LNCS 2383, pages 186–197. Springer-Verlag, 2002.

[TG03] S. Tiwari and S. Gallager. Machine learning and multiscale methods in the classification of bivalve larvae. In *Proc. 9th International Conference on Computer Vision (ICCV)*, pages 494–501, 2003.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1) :71–86, 1991.

[Vap95] V. Vapnik. *The nature of statistical learning theory.* Springer Verlag, 1995.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.

[WCG03] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features : the kernel recipe. In *Proc. 9th International Conference on Computer Vision (ICCV)*, pages 257–264, October 2003.

[Weh97] L. A. Wehenkel. *Automatic Learning Techniques in Power Systems.* Kluwer Academic Publishers, Boston, November 1997.

[YAR00] M.-H. Yang, N. Ahuja, and D. Roth. View-based 3d object recognition using SNoW. In *Proc. European Conference on Computer Vision*, 2000.