Combinaison de caractéristiques pour la reconnaissance rapide, robuste et invariante d'objets spécifiques

Jérôme Revaud[1] Yasuo Ariki[2] Guillaume Lavoué[1] Atilla Baskurt[1]

[1] Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
[2]CS-17, Université de Kobe, Japon

Bât. Jules Verne, INSA-LYON, 69621 Villeurbanne CEDEX
jerome.revaud@insa-lyon.fr

Résumé

Nous présentons une approche rapide et robuste d'appariement de graphes destinée à la reconnaissance d'objets spécifiques 2D dans les images. A partir d'un petit nombre d'images d'apprentissage, un graphe modèle de l'objet à apprendre est automatiquement construit. Il contient ses caractéristiques locales (de différents types : points d'intérêts, contours et textures) ainsi que leurs relations de proximité spatiale. La détection est basée sur une pré-sélection des sous-graphes les plus performants avec l'information mutuelle puis sur la programmation dynamique avec un treillis. Les expériences démontrent que la méthode proposée surpasse les détecteurs d'objets spécifiques de l'état de l'art dans des conditions réalistes de bruit grâce à l'utilisation des textures et des contours.

Mots-clés

Reconnaissance d'objets spécifiques 2D, combinaison de caractéristiques, textures, contours, information mutuelle

Abstract

We present a fast and robust graph matching method specially designed for 2D object recognition. Using a small set of training images, a model graph of the object is automatically built, containing its different local features (from different types : keypoints, lines, textures) together with their proximity spatial relationships. A selection of the most significant model subgraphs using mutual information, a detection lattice and an efficient indexing of the image features enable a fast detection. Experiments demonstrate that the proposed method outperforms the state-of-the-art specific object detectors in realistic noise conditions thanks to the texture and contour features.

Keywords

2D specific object recognition, combination of features, textures, contours, mutual information

1 Introduction

La reconnaissance d'objets est un sujet de recherche très actif depuis au moins 30 ans. De plus en plus d'algorithmes se révèlent capables de rivaliser avec la vision humaine ; pourtant, le système parfait reste encore à inventer. Le problème principal des techniques actuelles telles que [14, 4] est qu'elles nécessitent de lourds calculs, ce qui exclue malheureusement les possibilités d'application temps-réel sur des systèmes embarqués ou des téléphones portables. Dans cet article, nous nous intéressons à une techniques de reconnaissance robuste basée sur l'appariement de graphe et associant plusieurs types de caractéristiques locales.

La reconnaissance d'objets spécifiques a reçue une attention décroissante dans la communauté scientifique depuis l'avènement des *points d'intérêt* dont SIFT [10] est l'avatar le plus connu. Certes, les méthodes par points d'intérêt présentent de nombreux avantages : elles sont invariantes en translation, échelle, rotation et occultation sans augmentation de la complexité ; grâce au pouvoir descriptif élevé des points, l'apprentissage est pratiquement inexistant ; elles sont pratiquement temps-réel ; enfin, elles sont simples à mettre en oeuvre et fonctionnent très bien sur des objets texturés. Malgré tout, ces méthodes se révèlent incapable de traiter la plupart des objets usuels car ils sont faiblement texturés et, comme nous le montrons dans la suite de cet article, elles sont assez sensibles au bruit typique d'une utilisation réaliste. C'est pourquoi la tendance actuelle est à la combinaison de différents types de caractéristiques, comme dans [12, 6, 7], ou en plus des points d'intérêts sont utilisées des informations de contours et de textures pour pouvoir traiter davantage de classes d'objets. Malheureusement, l'ajout de nouvelles caractéristiques se paye bien souvent par une augmentation notoire du temps de calcul et par la perte des invariances [12, 6] ; de plus il est difficile de combiner différents types en une seule approche pour diverses raisons. Dans [12], plusieurs méthodes sont utilisées séparément suivant le type d'objet (simple, texturé) sans aucune unification, d'où une complexité beaucoup plus élevée et la perte des invariances (translation, ro-

tation...) associées aux points d'intérêts.

Un grand nombre de travaux se sont intéressés à la manière même de détecter un objet à partir des points d'intérêt. L'algorithme RANSAC [5], sans doute le plus ancien, permet notamment de produire des résultats même si le processus de détection est interrompu, un point crucial pour les applications embarquées où la contrainte de temps est forte. Les techniques à fenêtre glissante (*sliding window*) comme [8, 11] sont aussi, dans une moindre mesure, capable de cette fonctionnalité malheureusement elles perdent généralement l'invariance en rotation ou à l'occultation. Néanmoins, le problème de RANSAC est que, sous des transformations complexes, le nombre d'itérations requis pour avoir une chance de détecter l'objet devient très élevé. Récemment, Philbin et al. [15] ont proposé une solution basé sur un RANSAC localement optimisé (LO-RANSAC) lui-même développé par Chum *et al.* [2] : dans la boucle principale, une transformation simplifiée basée sur la position, l'échelle et l'orientation d'un seul point d'intérêt est calculée et dans l'optimisation locale, la transformation complète est déduite des points trouvés précédemment. En comparaison, notre méthode débute également d'un seul point, mais sans faire de suppositions sur la transformation complète grâce au relâchement des contraintes spatiales entre points éloignés. De plus, l'hypothèse implicite de RANSAC ou des méthodes par votes comme [10] que chaque caractéristique locale apporte la même quantité d'information sur le modèle ne tient souvent pas en pratique : quelques unes seront très spécifiques au modèle, alors que les autres seront plus courantes. Au contraire, notre méthode peut estimer l'importance de chaque caractéristique et choisir un seuil de détection adapté à chacune en utilisant la théorie de l'information. Pour résumer, nous proposons un modèle moins restrictif en terme de contraintes spatiales et visuelles qui peut combiner différents types de caractéristiques locales (points d'intérêts, contours, textures) pour gagner en robustesse. Notre méthode est dédiée à la reconnaissance d'objets spécifiques robuste et invariante en translation, rotation, échelle et à l'occultation, tout en étant minimaliste en terme de calcul.

L'algorithme proposé prend en entrée un ensemble d'images du modèle (la position du modèle dans les images est supposée connue) et d'images négatives (sans le modèle). Il produit automatiquement un *graphe prototype* du modèle rassemblant toutes les caractéristiques des différentes images modèle (section 2.2). Pour la reconnaissance, un *treillis de détection* est construit ; il stocke plusieurs façons de construire le graphe prototype en ajoutant les caractéristiques une par une. Le cas de l'occultation est résolu en ne considérant la détection comme complète que lorsqu'une partie suffisamment spécifique du graphe prototype est reconnue (section 2.3). Au final, la détection est très rapide (quelques millisecondes par objet). Nous présentons également un descripteur de texture rapide en section 3.3 qui contribue pour une grande part à nos bons résultats (sous-section 4.4). Les différentes parties de l'algorithme sont détaillées dans la section suivante et les résultats sont présentés en section 4. La section 5 conclut et présente quelques perspectives.

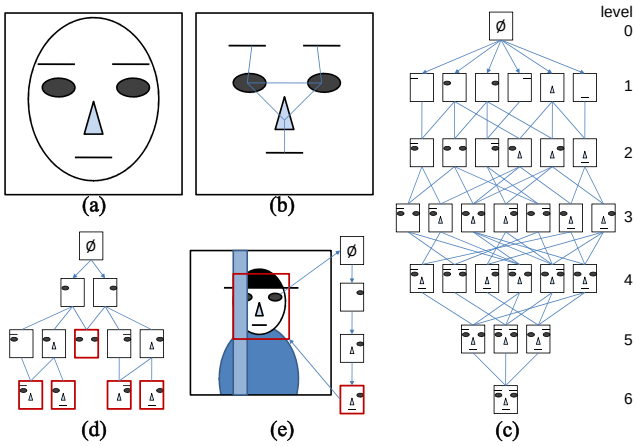

FIGURE 1 – Résumé de notre méthode. (a) Exemple d'une image modèle discrétisée. Les 3 types de caractéristiques locales sont figurées par des segments, des cercles et des triangles pour l'explication. (b) Le graphe prototype correspondant. (c) Le treillis de détection complet. (d) Le treillis de détection élagué. Les noeuds terminaux sont en rouge. (e) Un exemple de détection d'un visage semi-caché à partir d'une caractéristique amorce sur l'oeil droit.

2 Description de l'algorithme

L'algorithme est résumé dans la figure 1 : (a) (b) tout d'abord, un graphe prototype est extrait des images modèles discrétisées ; (c) (d) ensuite, le treillis de détection est construit à partir de ce graphe puis est élagué afin d'obtenir une robustesse à l'occultation et une détection rapide ; (e) finalement, le treillis est utilisé pour détecter des objets dans les images test en partant d'une seule caractéristique locale.

2.1 Extraction préliminaire des caractéristiques

Dans le but de faciliter les étapes suivantes, un grand nombre de caractéristiques locales de différents types sont extraites sur les images d'apprentissages puis indexées de manière à pouvoir les retrouver rapidement. En effet, pendant l'apprentissage et la détection, le système a besoin de tester la présence de ces caractéristiques par rapport aux hypothèses générées par le treillis.
Les caractéristiques, dénotées par $\mathbf{f} = (\mathbf{c}, \mathbf{h}, \mathbf{z})$, sont définies par un centre $\mathbf{c}$, un vecteur radial $\mathbf{h}$ et un descripteur (spécifique à chaque type) $\mathbf{z}$. Dans notre implémentation, nous avons utilisé les trois types de caractéristiques suivants : points d'intérêts SIFT [10], segments de contours et descripteur de texture car ils peuvent tous être extrait rapidement (cf. section 3). Une fonction de distance entre deux caractéristiques du même type est également définie

pour chaque type ; elle prend en compte leur position, leur descripteur, ou les deux.

Caractéristiques amorces. Afin de réduire l'espace de recherche au moment de la reconnaissance, nous avons limité le RANSAC initial à ne faire un choix que parmi les points d'intérêts SIFT [10]. Un autre type de caractéristique pourrait être utilisé à condition d'être saillant dans l'image.

2.2 Le graphe prototype

Le but du graphe prototype est double : premièrement, réduire le nombre de caractéristiques et deuxièmement, représenter le modèle sous une forme unique standard. Après avoir aligné les images modèle, une analyse de co-occurrence de chaque caractéristique est conduite : chaque caractéristique est recherchée dans les autres images au même endroit relatif et reçoit un score égal à la somme des distances correspondantes (donc les caractéristiques les plus redondantes possèdent les plus petits scores). Ensuite, pour chaque groupe de caractéristiques synonymes (i.e. celles qui représentent la même zone du modèle sur des images différentes), seule la plus redondante est retenue.
Après cette première sélection, le graphe prototype est construit avec les caractéristiques restantes regroupées en une seule image selon leur position relative. Dans ce graphe, deux caractéristiques sont connectées à la condition que la distance spatiale entre elles soit petite par rapport à leur rayon : $dist_{sp}(\mathbf{f}, \mathbf{f}') < \|\mathbf{h}\| \, \|\mathbf{h}'\|$. Par conséquent, chaque arrête du graphe représente idéalement une relation de voisinage stable ; en fait nous supposons ici que le bruit de position dépend de la taille de la caractéristique. La construction du graphe prototype est automatique ; la figure 1.(b) montre un exemple de graphe prototype construit à partir d'une seule image (un visage schématisé).

2.3 Le treillis de détection

Le treillis de détection représente chaque chemin possible pour construire le graphe prototype en ajoutant les caractéristiques une par une. Il est organisé de manière à ce que chaque niveau l contienne tous les sous-graphes de cardinalité l. Par exemple, le niveau $l = 1$ contient toutes les N caractéristiques unitaires du graphe prototype, etc. Par ailleurs, chaque arrête du treillis correspond à l'ajout d'une caractéristique au sous-graphe de départ. Globalement, le treillis comporte $N + 1$ niveaux. La figure 1.(c) illustre un exemple de treillis complet.

Procédure de reconnaissance. La procédure de reconnaissance découle logiquement de cette structure : tout d'abord, une caractéristique amorce est tirée au hasard dans l'image (RANSAC) ; puis elle est passée en entrée du treillis. De manière intuitive, l'algorithme va itérativement vérifier pour chaque arrête, en partant du premier niveau, si la caractéristique-modèle correspondante peut aussi être trouvée dans l'image test. Au fil de la progression dans le treillis se créent ainsi des agrégats de plus en plus gros (i.e. des sous-graphes du graphe prototype).
Concrètement, soit A un agrégat contenant l caractéristiques déjà trouvées, alors la condition pour passer au niveau $l + 1$ du treillis par l'arrête e_i (ce qui correspond à ajouter la caractéristique modèle $\mathbf{f}_i$) est :

$$d_i = dist_f(\mathbf{f}_i, pos(\mathbf{c}_i, \mathbf{h}_i; A)) \leq d_i^{max}$$

où d_i représente la distance entre $\mathbf{f}_i$ et ce qui peut être trouvé dans l'image test à la position $pos(\mathbf{c}_i, \mathbf{h}_i; A)$, et où d_i^{max} est une constante apprise pendant l'apprentissage (section 2.5). Ici, $pos(\mathbf{c}_i, \mathbf{h}_i; A)$ dénote la position prédite de la caractéristique $\mathbf{f}_i$ dans l'image test relativement à la position de l'agrégat courant, laquelle est simplement obtenue avec une isométrie (cf. 2.4). Bien que le calcul de d_i soit spécifique à chaque type de caractéristique, le fait que ces dernières soient ajoutées une par une permet de contourner le problème classique de combinaison de caractéristiques différentes. Quand une caractéristique n'est pas trouvée (i.e. $d_i > d_i^{max}$), le chemin est abandonné. Idéalement, quand le noeud terminal (au niveau $N + 1$) est atteint, les positions de toutes les caractéristiques du modèle sont connues et la reconnaissance est terminée (cf. figure 1.(c)). Nous montrons par la suite que l'on peut s'arrêter bien avant.
Il est à noter que dans notre approche les distances sont calculées en direct pendant l'appariement ; ainsi le graphe résultat peut être considéré comme continu dans l'espace de l'image test du fait de l'utilisation de caractéristiques non-saillantes. Par exemple, un nouveau segment peut être reconnu sur un plus grand (ou l'inverse : deux segments alignés peuvent être rassemblés), ou encore une texture peut être extraite sur demande n'importe où dans l'espace-échelle orienté de l'image. En comparaison, les approches classiques d'appariement de graphes ont d'abord besoin d'extraire les graphes des images (i.e. étape de discrétisation) avant de pouvoir passer à l'appariement en lui-même.

2.4 Croissance des agrégats

Comme les caractéristiques locales, les agrégats possèdent un centre et un vecteur radial calculés à partir des positions des caractéristiques contenues moyennés pour réduire le bruit de position. Soit un agrégat A contenant l caractéristiques locales $\mathbf{f}_j$, alors son centre $\mathbf{c}_A$ et son vecteur radial $\mathbf{h}_A$ sont définis comme :

$$\mathbf{c}_A = \frac{1}{l} \sum_{j=1}^{l} \mathbf{c}_j \tag{1}$$

$$\mathbf{h}_A = \frac{\sum_{j=1}^{l} \mathbf{h}_n}{\left\| \sum_{j=1}^{l} \mathbf{h}_n \right\|} \sqrt{\frac{1}{L} \left(\sum_{j=1}^{l} \|\mathbf{c}_j\|^2 + \|\mathbf{h}_j\|^2 \right) - \|\mathbf{c}_A\|^2} \tag{2}$$

Ces deux vecteurs peuvent être calculés en temps constant au prix de quelques variables cachées, ce qui rend cette opération très rapide. Ce mécanisme s'est montré stable

expérimentalement, même en cas de fortes déformations (voir par exemple la fig. 5). Il est utilisé indifféremment pendant l'apprentissage et pendant la détection : dans le premier cas, les agrégats sont composés de caractéristiques extraites des images modèle alors que dans le second elles viennent de l'image test.

2.5 Élagage du treillis

Même si le graphe prototype réduit le nombre de combinaisons possibles grâce à la contrainte de voisinage, le treillis complet reste malgré tout de taille exponentielle. De plus, le cas de l'occultation doit aussi être pris en compte. Puisqu'il est impossible de construire le treillis complet puis de l'élaguer, nous avons opté pour une solution sous-optimale qui consiste à itérativement ajouter un nouvel étage du treillis dans son intégralité, mesurer l'efficacité de chaque agrégat candidat et ne garder que les meilleurs.

Construction du premier niveau. Seul un sous-ensemble des caractéristiques amorces (les N_{seed} meilleures en terme de redondance et d'échelle) est retenu pour construire le niveau initial ($l = 1$) du treillis. Cette limitation vient du fait que la complexité de l'algorithme est approximativement linéaire avec N_{seed}, mais étrangement il est apparu dans les expériences que la reconnaissance atteint un maximum pour une petite valeur de N_{seed}, entre 12 et 20 (voir section 4.2).

Élagage itératif du treillis. Ensuite, pour chaque niveau à partir du second, tous les agrégats possibles sont ajoutés au treillis incomplet. A chaque fois, les micro-classifieurs embarqués sur les nouvelles arrêtes sont entraînés individuellement, puis les meilleurs agrégats sont sélectionnés par une procédure d'optimisation globale qui s'assure que toutes les zones du modèle puissent être détectées indépendamment et efficacement.

Calcul des micro-classifieurs Nous avons utilisé l'Information Mutuelle (IM) comme dans [3] pour trouver le seuil optimal d_i^{max} pour chaque arrête e_i. En pratique, l'IM mesure la corrélation (positive ou négative) entre deux variables aléatoires. Dans notre cas, comme nous voulons seulement prendre en compte la corrélation positive, nous utilisons une variante pondérée :

$$IM(A;O) = \sum_{a,o\in\{0,1\}^2} w_{ao}\, p(a,o) log \frac{p(a,o)}{p(a)p(o)}$$

$$\text{avec } w_{ao} = \begin{cases} 1 & \text{si } a = o, \\ 0 & \text{sinon} \end{cases}$$

où A et O sont des variables aléatoires binaires qui représentent respectivement la détection d'un agrégat et sa classe (vérité terrain). Sans perte de généralité, nous nous intéressons maintenant à la procédure d'optimisation pour une arrête e_i donnée. Tout d'abord, d_i^{max} est initialisé à l'infini et le processus de détection est lancé avec le treillis courant sur les images de classe et de non-classe, ce qui résulte en une liste de M agrégats détectés $\{\bar{A}_{i,j}\}_{j=1}^{M}$ (la notation $\bar{A}$ correspond à un agrégat détecté, par opposition à un agrégat modèle A). A chaque détection $\bar{A}_{i,j}$ est associé une distance $d_{i,j}$ correspondant à l'arrête e_i. Comme nous savons pour chaque distance si $\bar{A}_{i,j}$ est un agrégat valide ou pas (nous ne comptons comme positifs que les agrégats qui extrapolent correctement la position globale de l'objet) nous pouvons calculer le seuil d'acceptation d_i^{max} qui maximise l'IM (nous utilisons une fenêtre de Parzen avec noyau gaussien pour lisser les distributions de probabilité). Si une arrête ne laisse plus passer d'agrégats positifs, elle est supprimée automatiquement.
Finalement, la règle de Bayes est utilisé pour estimer l'efficacité de classification de chaque agrégat candidat A_k à partir des arrêtes le pointant :

$$\begin{aligned} p_{A_k} &= \min_{\forall i: e_i \to A_k} p_{e_i} \\ &= \min_{\forall i: e_i \to A_k} p(O = 1 | d_i \leq d_i^{max}) \end{aligned}$$

Si p_{A_k} dépasse le seuil p_{min} défini par l'utilisateur, alors A_k devient un agrégat *terminal* (carrés rouges dans la fig. 1.d). Ainsi, les agrégats peuvent devenir terminaux à différents niveaux selon la spécificité de leurs caractéristiques par rapport au modèle, de même qu'un oeil humain aura juste besoin de voir une petite partie très caractéristique d'un objet occulté pour le reconnaître et *vice versa*.

Sélection des meilleurs agrégats Une seconde étape d'optimisation sert à sélectionner les meilleurs agrégats d'un point de vue global : un seul paramètre r^{max} permet de contrôler le compromis entre une bonne couverture spatiale du modèle (i.e. pouvoir le reconnaître peu importe la zone occultée) et une vitesse de reconnaissance maximale (i.e. ne sélectionner que les agrégats les plus performants). Pour cela, chaque candidat est classé selon la quantité d'information qu'il peut apporter au treillis :

$$gain(A|\mathbf{M};\mathbf{O}) = IM(\mathbf{M}, \mathbf{M}^A; \mathbf{O}) - IM(\mathbf{M};\mathbf{O})$$

Ici, $\mathbf{M} = [r_1, \ldots, r_T]$ est un vecteur contenant les probabilités de détecter les différentes zones du modèle avec le treillis courant. Concrètement, $\mathbf{M}$ est simplement la concaténation de plusieurs pyramides d'espace-échelle discrétisées correspondant à chaque image modèle (4 échelles par pyramide, 1x1, 2x2, 3x3 et 4x4 = 30 valeurs/image) et d'une valeur supplémentaire servant à stocker la probabilité des faux positifs [1]. Les vecteurs $\mathbf{M}^A$ et $\mathbf{O}$ se rapportent respectivement à l'agrégat A (isolé du reste du treillis) et à la vérité terrain binaire, exprimés sous la même forme. Pour qu'il puisse jouer son rôle de mémoire des zones déjà prises en compte, le vecteur $\mathbf{M}$ est mis à jour chaque fois qu'un nouvel agrégat A devient terminal : $\mathbf{M}^A$ est ajouté à $\mathbf{M}$ et le résultat est seuillé à la valeur maximale r^{max} (le gain d'IM est donc intrinsèquement modulé par le paramètre r^{max}).
Ainsi, une valeur de $r^{max} = 0$ équivaut à désactiver cette mémoire (i.e. $\mathbf{M}$ = vecteur nul), donnant les meilleurs scores aux agrégats les plus efficaces sans tenir compte de

1. Ce modèle n'est qu'une approximation du modèle théorique rigoureux (mais irréalisable) où $\mathbf{M}$ serait exprimé dans l'espace des agrégats.

leur redondance spatiale, alors qu'une valeur de 1 mènerait à un résultat opposé (i.e. jamais deux agrégats traitant la même zone). Pour les expériences, nous avons fixé r^{max} à sa valeur optimale de 0.5.
Au final, un nombre limité des meilleurs candidats est conservé pour le niveau courant, et le processus se répète jusqu'à ce qu'il n'y ait plus de candidats ou que le niveau maximal L_{max} soit atteint. Dans nos expériences, nous avons fixé L_{max} à 6 et avons limité la largeur du treillis à $4N_{seed}$ au plus et le nombre d'enfants par agrégat à 4 au plus.

2.6 Programmation dynamique

Pendant la reconnaissance, la progression des agrégats dans le treillis est traitée en largeur. De cette manière, si deux agrégats représentant le même patch se rejoignent, nous gardons le meilleur des deux en terme de la somme des d_i. Quand un agrégat atteint un noeud terminal, il vote pour une position extrapolée à partir de sa position. Les votes sont regroupés dans l'espace-échelle et un seuil sur le nombre de votes est utilisé pour le résultat final.

3 Caractéristiques utilisées

Préalablement à la reconnaissance, l'image test est indexée sous une forme plus simple que pour l'apprentissage : au lieu d'extraire un grand nombre de caractéristiques, il suffit de pré-calculer quelques informations sous-jacentes nécessaires à une future extraction rapide. Dans notre implémentation, cela est spécialement le cas pour les segments et les textures. A chaque type de caractéristique est associé un système de requête de la forme "caractéristique requête idéale / caractéristique existante la plus proche" pour que le treillis puissent effectuer les multiples tests nécessaires à la détection. Finalement, nous n'avons pas utilisé l'information de couleur des images.

3.1 Points d'intérêt

Nous avons utilisé le détecteur et descripteur de points d'intérêt SIFT tel que décrit dans [10]. L'indexation des points se fait grâce à un k-d tree sur les 4 dimensions spatiales (position et vecteur radial) des points, ce qui permet de trouver le point existant dans l'image test le plus proche d'une position requête donnée. La fonction de distance entre deux points d'intérêt est définie comme la distance Euclidienne entre ces vecteurs 4D si leur descripteur SIFT sont assez proches ($\|\mathbf{z} - \mathbf{z}'\| < 0.5$) et $+\infty$ sinon.

3.2 Segments de contours

Pour extraire les contours, nous avons utilisé le détecteur de Canny [1] suivi par une étape de polygonisation. La fonction de distance entre deux segments est définie comme le maximum de la distance minimale entre chaque point du segment requête et tout segment ayant une orientation similaire dans l'image test. Nous avons utilisé 6 cartes de distance (une pour chaque orientation) pour accélérer la recherche. Cette technique est robuste à une segmentation bruitée car la distance varie peu en cas de segment brisé ou trop long. Le segment récupéré est la rétro-projection du segment requête sur le ou les segments de l'image les plus proches.

3.3 Textures

Nous avons dérivé un descripteur de texture de l'approche de Tola et Lepetit [16]. Comme dans l'article original, nous pré-calculons 8 pyramides de gradient pour couvrir tout l'espace-échelle de l'image dans toutes les orientations. Puisqu'une texture n'est pas une caractéristique saillante de l'image, la fonction de distance est seulement définie comme la distance Euclidienne entre le descripteur modèle et celui extrait sur l'image test à la position requête. Le descripteur texture est défini plus particulièrement comme la concaténation de trois descripteurs atomiques extraits au même endroit mais à des échelles légèrement différentes : $[s/1.54, s, 1.26\,s]$, où s est l'échelle requête. Chaque descripteur atomique est simplement un histogramme local du gradient (8 bins) extrait sur les pyramides.

4 Résultats expérimentaux

4.1 Apprentissage et base de test

Nous avons capturé une douzaine de vidéos avec une caméra SONY handycam (résolution : 720x480). Habituellement, les systèmes de reconnaissance sont testés sur des photos de bonne qualité comme dans [10] ou [2] ; cependant notre choix est motivé par la volonté de mieux simuler les conditions réalistes de la vision robotique, généralement beaucoup plus difficiles à cause de nombreux facteurs : luminosité d'intérieure criarde, bruits divers (capteur, flou de bougé, entrelacement vidéo) et objets en eux-même pas toujours fortement texturés. Les vidéos ont été échantillonnées à 10 fps (résultant en 2287 frames) où la vérité terrain a été manuellement étiquetée.
Dix objets à notre disposition ont servi à tester le système proposé (voir la fig. 2). Le choix de ces objets s'est basé sur leur degré de texture et leur forme dans l'optique de couvrir une vaste gamme d'objets possibles : les modèles n°3, 9 et 10 sont fortement texturés contrairement aux objets n°5, 7 et 8 ; les modèles n°1, 9 et 10 sont plats et rectangulaires alors que les modèles n°2, 3, 6, 7 et 8 ont des formes 3D complexes ; d'autres encore comportent des trous ; certains sont propices aux réflexions spéculaires etc. Nous avons utilisé entre 1 et 4 images d'apprentissage pour chaque objet modèle, mais ce nombre ne semble pas avoir une grande importance. La liste détaillée des modèles est présentée en table 1. Tous les objets ont été recherchés dans toutes les frames (les instances multiples par frame sont autorisées). Pour tester les limites de notre système, nous n'avons utilisé que 19 images négatives communes pour l'apprentissage [2], ce qui est ridiculement petit comparé à l'espace des textures ou des vecteurs SIFT et *a fortiori* à l'espace des agrégats.

2. Toutes les images d'apprentissage sont à http ://liris.cnrs.fr/jerome.revaud/PAMI09

#	modèle	#img	%kpt	%line	%tex	#frm
1	diplôme	1	18.6	26.5	**54.9**	373
2	peluche	4	18.0	15.4	**66.7**	319
3	bouteille	3	39.1	4.3	**56.5**	130
4	clavier	1	**83.7**	4.7	11.6	70
5	cintre	2	**51.8**	24.1	24.1	352
6	chaise	4	**46.2**	28.6	25.2	179
7	tasse	4	14.6	29.2	**56.2**	212
8	table	4	0.0	40.0	**60.0**	329
9	journal	3	31.1	24.4	**44.4**	201
10	vinyle	3	28.1	**43.8**	28.1	154

TABLE 1 – Détail de la base de test. *#img* : nombre d'images d'apprentissage ; *%kpt*, *%line*, *%tex* : pourcentage de points d'intérêt, de segments et de textures automatiquement choisis dans le treillis de détection ($N_{seed} = 20$) ; *#frm* : nombre d'images de test dans lequel le modèle apparaît.

FIGURE 2 – Objets modèles utilisés pour les expériences. Le degré de texture et la forme diffèrent significativement d'un objet à l'autre.

4.2 Comparaison avec des systèmes existants

Nous nous sommes comparés en terme de courbes rappel-précision (RP) avec les méthodes de détection d'objets spécifiques les plus utilisées :

- RANSAC standard avec une homographie. Les points d'intérêts de tous les modèles sont stockés dans un k-d tree pour un appariement rapide (basé sur le ratio des distances du premier au second plus proche voisin inférieur à 0.8). Comme une homographie nécessite au moins 4 points pour calculer une pose, cette méthode devient assez lente quand le ratio d'appariements corrects (inliers) est petit. Nous avons donc utilisé une probabilité surestimée de 0.25 (le vrai ratio étant généralement plus petit).
- RANSAC optimisé localement (LO-RANSAC) par Chum et al. [2] et adapté par Philbin et al. [15]. Il est similaire au RANSAC à part qu'une isométrie est utilisé dans la boucle principale et une homographie dans la boucle secondaire (à chaque fois qu'un nouveau maximum est trouvé). Cette solution permet de réduire considérablement le nombre d'itérations car elle ne tire qu'un seul point à chaque fois. Comme pour RANSAC, la probabilité de trouver/accepter l'objet sert à générer les courbes RP.

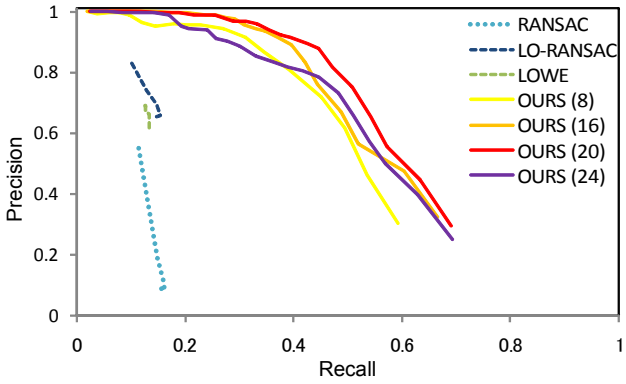

FIGURE 3 – Courbes rappel-précision pour tous les modèles. Pour notre méthode, le paramètre entre parenthèse indique la valeur de N_{seed}.

- Le système de détection de Lowe [9, 10] : les points d'intérêts SIFT sont utilisés pour l'appariement de caractéristiques locales avec un kd-tree. Ensuite, chaque paire vote pour une pose approximative du modèle, et les votes sont accumulés dans une table de hachage. Pour chaque groupe de 3 votes ou plus, l'hypothèse est vérifiée avec une transformation affine et un modèle probabiliste. Le seuil de probabilité est utilisé pour générer les courbes RP.
- La méthode proposée avec $N_{seed} \in [8, 24]$ et $p_{min} = 0.95$. Un seuil sur le nombre de votes sert à générer les courbes RP (nous avons aussi essayer de faire varier p_{min} mais les résultats étaient inférieurs).

Toutes les méthodes ont été entraînées avec les mêmes images modèle. Les résultats sont présentés sur la fig. 3 en terme de courbe RP. Le rappel et la précision sont définis comme N_c/N_g et N_c/N_d respectivement, où N_c est le nombre de détections correctes, N_g le nombre de boites de la vérité terrain et N_d le nombre total de détections. Quelques exemples de détections sont donnés en fig. 4 et 5 [3]. En moyenne, notre système surclasse les autres en terme de rappel-précision. Bien qu'il soit difficile de donner une liste exhaustive des causes de cette supériorité, voici les raisons principales :

- Même avec un seuil de détection très bas, les méthodes de Lowe et LO-RANSAC ($p \simeq 0$) ne génèrent pas plus de vrais positifs (cf. courbes RP pratiquement verticale). Cela est du au fait que ces méthodes requièrent au minimum 3 ou 4 points d'intérêt correctement appariés (mais souvent plus) pour authentifier une seule détection, ce qui confère à l'impossible sur des images bruitées.
- Quand le modèle est peu texturé, les descripteurs SIFT sont très peu spécifiques, ce qui diminue la probabilité d'un appariement valide par la méthode du ratio premier sur deuxième voisin. Comme notre méthode utilise une distance absolue entre les descripteurs SIFT, ce problème ne se présente pas.

3. Quelques vidéos exemples peuvent être trouvées sur http ://liris.cnrs.fr/jerome.revaud/PAMI09

– La différence d'échelle est souvent trop importante entre les images modèle et les instances dans la vidéos. Habituellement, les photos du modèle sont prises en gros-plan alors que pendant les tests, l'objet apparaît souvent petit et loin. C'est un problème pour les points d'intérêts, car leur invariance théorique au changement d'échelle a ses limites. Au contraire, notre descripteur de texture peut être extrait n'importe où, notamment à des petites échelles.
– Les vecteurs SIFT des points situés sur les bords d'un objet décrivent plus l'arrière-plan que l'objet lui-même. Au contraire, notre méthode peut se servir des segments de contours, moins sujets à l'arrière-plan.

A propos des autres méthodes, RANSAC produit les moins bons résultats dans toutes les situations. La méthode de Lowe est en moyenne légèrement inférieure à LO-RANSAC, ce qui s'explique par la supériorité de l'homographie sur la transformation affine. Bien que notre méthode n'utilise qu'une isométrie pour comparer les agrégats, elle reste robuste grâce à la petite taille des agrégats terminaux comparés à l'objet entier. Plus généralement, les performances de chaque système dépendent grandement du modèle en lui-même [4]. Par exemple, les méthodes de Lowe et LO-RANSAC échouent totalement sur les modèles n°6 et n°7, alors que notre système réalise des performances médiocres mais existantes.

Pour finir, le nombre optimal de points d'intérêt N_{seed} dans le niveau initial du treillis reste étonnamment petit dans l'absolu. Bien que notre méthode n'initie la détection qu'avec 10-20 points d'intérêts "amorce", à comparer aux 100-300 points d'intérêts stockés par les autres méthodes pour chaque objet modèle, des valeurs de rappel supérieures sont atteintes. Cela indique qu'un système de reconnaissance efficace peut être construit en deux parties : une première pour la génération rapide d'hypothèses en n'utilisant qu'une fraction de l'information du modèle, et une seconde pour vérifier les hypothèses avec le reste de l'information.

FIGURE 5 – Exemples de détections sur la classe "panneaux stop" de Caltech-101 avec une seule image modèle. Notre méthode est robuste aux déformations, aux changements de point de vue et aux légères variations intra-classes.

4.3 Vitesse de détection

Tous les tests ont été réalisés sur un ordinateur à 2.3 Ghz. Au final, l'extraction préliminaire des caractéristiques locales prend environ 2s sur une image 720x480 [5] (voir la table 2). En comparaison, le temps de détection avec la méthode proposée varie entre 3 et 7 ms par objet, ce qui est environ 5 fois plus rapide qu'avec RANSAC. La plupart du temps dépensé par LO-RANSAC et Lowe est en fait consommé dans le kd-tree pour apparier les points d'intérêt de l'image à ceux des modèles. Comme nous comparons chaque point de l'image à un petit nombre de points amorce, notre méthode est plus rapide mais génère plus d'hypothèses. Néanmoins, le faible nombre de caractéristiques par agrégat terminal (2.6 en moyenne) permet des vérifications pratiquement instantanées.

Dans notre implémentation actuelle, le temps de détection total est proportionnel au nombre d'objets modèle mais, comme énoncé par Messmer et al. [13], il pourrait devenir sous-linéaire si les différents treillis étaient réunis (mais ce n'est pas le propos de cet article).

4. les détails des courbes RP pour chaque objet sont sur http ://liris.cnrs.fr/jerome.revaud/PAMI09

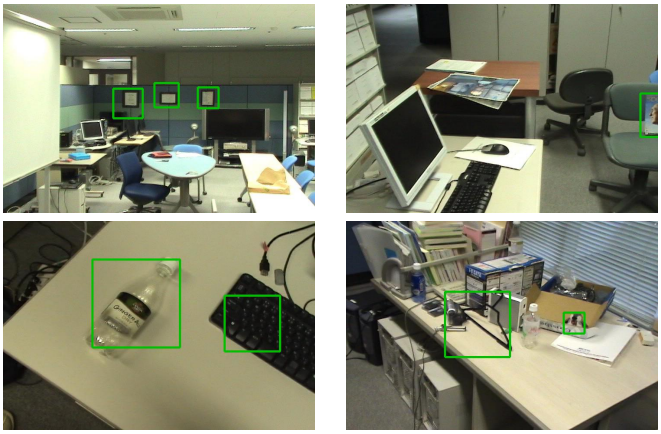

FIGURE 4 – Exemples de détections sur des images extraites de vidéos prises en intérieur avec une caméra SONY (la précision est fixée à 0.95 pour chaque objet). Notre méthode est robuste à l'occultation, aux changements d'échelle ou de point de vue et aux bruits divers comme le flou de bougé.

Opération		temps (ms)
Extraction des caractéristiques locales	points SIFT	1648
	segments	155
	textures	60
Détection des 10 objets	RANSAC	235.5
	LO-RANSAC	78.1
	LOWE	77.6
	Nous	53.6

TABLE 2 – Temps moyens de traitement sur une image 720x480 avec un ordinateur à 2.3 GHz

4.4 Importance de chaque type de caractéristique

Dans le but d'évaluer l'importance de chaque type de caractéristique locale, nous avons recueilli les performances

5. Nous avons utilisé l'exécutable de Lowe pour extraire SIFT mais d'autres implémentations sont plus rapides.

type de caractéristiques	P=R (10 objets)
points d'intérêt SIFT	27 %
segments de contour	40 %
textures de gradient	60 %
SIFT + segments	55 %
SIFT + textures	59 %
tous les 3	57 %

TABLE 3 – Comparaison des performances de détection quand le treillis est construit en excluant certains types de caractéristiques en terme de valeur précision = rappel ($N_{seed} = 20$).

de reconnaissance obtenues par des treillis construits en excluant certains type de caractéristiques [6]. Les résultats moyennés pour tous les objets sont présentés en terme de valeur rappel=précision sur le tableau 3. Cette mesure ne reflète pas nécessairement la supériorité d'un détecteur sur un autre, car en pratique on favorise plutôt la précision sur le rappel, mais les deux sont souvent corrélés.

Les résultats montrent que la combinaison de plusieurs types fonctionne généralement mieux que l'utilisation d'un seul type de caractéristiques, sauf pour les textures. Nous attribuons ce paradoxe au faible nombre d'image d'apprentissage qui ne permet pas à l'algorithme d'apprendre précisément l'importance de chaque caractéristique. Mais les performances dépendent en réalité beaucoup des objets : par exemple, la combinaison SIFT+segments fonctionne mieux que les textures pour les modèles n°3 et 7. En moyenne toutefois, utiliser les trois types permet de s'assurer de performances parmi les meilleures. Les bons résultats de notre méthode semblent donc issus en grande partie du type "texture" et en moindre part des segments. Cela s'explique par le fait que l'extraction des textures n'est pas sujette au bruit (contrairement aux points SIFT) : elles sont virtuellement toujours présentes. Cela confirme notre hypothèse initiale que les détecteurs de points saillants, important pour réduire l'espace de recherche et pour ancrer les agrégats pendant leur croissance, devraient toujours être utilisé en association avec des caractéristiques non-saillantes pour gagner en robustesse.

5 Conclusion

Nous avons présenté une approche novatrice qui permet une reconnaissance d'objet 2D robuste à la déformation et à l'occultation. Grâce à l'utilisation de différents types de caractéristiques (saillantes et non-saillantes), notre approche surpasse les détecteurs d'objets spécifiques de l'état de l'art dans des conditions réalistes d'utilisation. De plus, nous avons montré qu'atteindre des fortes valeurs de rappel ne nécessite pas forcément d'utiliser toute l'information du modèle ; seule une infime fraction des points d'intérêts du modèle utilisés pour la génération d'hypothèse détecte la plupart des instances dans les images de test. Finalement, l'appariement continu de graphe présenté dans le cadre de cet article est très général et pourrait être appliqué à d'autres type d'approche, par exemple en 3D si on modifiait la loi de croissance des agrégats. Le concept d'agrégat semble aussi intéressant pour la classification d'objets dans le cadre de la théorie des "bags-of-keypoints".

6. mais le niveau initial se compose toujours de points SIFT.

Références

[1] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6) :679–698, 1986.

[2] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. *Pattern Recognition*, pages 236–243, 2003.

[3] B. Epshtein and S. Ullman. Feature hierarchies for object classification. In *International Conference on Computer Vision (ICCV'05)*, pages 220–227. IEEE Computer Society, 2005.

[4] B. Epshtein and S. Ullman. Semantic hierarchies for recognizing objects and parts. In *Computer Vision and Pattern Recognition (CVPR'07)*. IEEE Computer Society, 2007.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395, 1981.

[6] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proceedings of the Twelfth IEEE International Conference on Computer Vision (ICCV)*, October 2009.

[7] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[8] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *International Conference on Computer Vision (ICCV'05)*, pages 446–453. IEEE Computer Society, 2005.

[9] D. G. Lowe. Local feature view clustering for 3d object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, 2001.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2) :91–110, 2004.

[11] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition (CVPR), 2008.*, pages 1–8, 2008.

[12] A. Mansur, M. Hossain, and Y. Kuno. Integration of multiple methods for class and specific object recognition. In *International Symposium on Visual Computing*, pages I : 841–849, 2006.

[13] B. T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5) :493–504, 1998.

[14] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *Computer Vision and Pattern Recognition (CVPR)*, pages 11–18. IEEE Computer Society, 2006.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[16] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.