Suivi temps-réel : matrices de covariance couleur-texture et commutation automatique de descripteur/opérateur

Synthèse du manuscrit en version française

0.1 Introduction

Cette thèse propose un système de vision par ordinateur capable de détecter et suivre plusieurs objets dans les séquences vidéo. L'algorithme proposé de recherche de correspondances s'appuie sur les matrices de covariance obtenues à partir d'un ensemble de caractéristiques extraites des images (couleur et texture principalement). L'idée initiale sur laquelle reposent les bases de notre algorithme a été publiée par Porikli *et al*. [109] et son principal avantage est l'utilisation d'un descripteur de l'objet à suivre fusionnant des sources d'information très hétérogènes. Cette représentation, compacte et discriminante est efficace pour le suivi d'objets mais également leur ré-identification.

Quatre contributions sont introduites dans cette thèse. Tout d'abord nous nous intéressons à l'invariance des algorithmes de suivi face aux changements de contexte. Nous proposons ici une méthodologie pour mesurer l'importance de l'information couleur en fonction de ses niveaux d'illumination et de saturation. Ensuite, une deuxième partie se consacre à l'étude des différentes méthodes de suivi, leurs avantages et limitations en fonction du type d'objet à suivre (rigide ou non rigide par exemple) et du contexte (caméra statique ou mobile). La méthode que nous proposons s'adapte automatiquement et utilise un mécanisme de commutation entre différentes méthodes de suivi en considérant leurs forces complémentaires [79]. Notre algorithme s'appuie sur un modèle de covariance qui fusionne les informations couleur-texture et le flot optique (KLT) modifié pour le rendre plus robuste face aux changements d'illumination [113]. Une deuxième approche propose l'analyse des différents espaces de réprésentation couleur afin d'obtenir un descripteur assurant un bon équilibre entre pouvoir discriminant et invariance photométrique. Nous avons proposé une nouvelle représentation par matrice de covariance qui par une nouvelle façon d'intégrer les caractéristiques de texture LBP (pour *Local Binary Patterns* en anglais) permet de réduire la taille des matrices tout en conservant un très bon pouvoir discriminant [114].

Une troisième contribution porte sur le problème de suivi multi-cibles où plusieurs difficultés apparaissent parmi lesquelles la confusion des identités de deux objets, les occultations, la fusion ou la division des trajectoires. Ici, un ensemble de matrices de covariance distribuées spatialement est utilisé pour ré-identifier les cibles. La solution finale que nous proposons utilise une fonction d'énergie discrète qui s'appuie sur le comportement de l'ensemble des trajectoires et sur les modèles d'apparence proposés [116]. Finalement, à partir des algorithmes développés, nous avons réalisé une étude

sur l'adéquation algorithme-architecture et implémenté les codes sur des architectures multi-cœur en atteignant des gains importants en terme de rapidité d'exécution [79, 117].

Ainsi la thèse se divise en six parties. Tout d'abord, nous introduisons dans la partie 0.2 les caractéristiques visuelles disponibles au niveau pixel et leur utilisation pour définir le descripteur plus global de l'objet à étudier. Nous abordons ensuite dans 0.3 le problème de la détection d'objet et du suivi. La troisième partie 0.4 détaille les premières contributions de la thèse : la commutation d'opérateurs et de caractéristiques pour le suivi. Ensuite, nos approches proposées en ce qui concerne la mise en correspondance par covariance font l'objet de la partie 0.5. Enfin, l'étude algorithme-architecture est développée dans la partie 0.6.

0.2 Information locale et représentation d'objets

La qualité d'un descripteur associé à un objet peut être mesurée suivant trois critères :

- la compacité de la représentation, pour permettre un stockage et une exécution efficaces ;
- le caractère distinctif ou discriminant, c'est-à-dire que chaque descripteur doit être unique pour chaque objet de la vidéo considérée ;
- la répétabilité : la représentation doit être constante au cours du temps malgré les modifications des conditions d'acquisition ;

À partir des multiples informations pixelliques disponibles (voir paragraphe 0.2.1), plusieurs descripteurs d'objets sont envisageables (paragraphe 0.2.2).

0.2.1 Information disponible au niveau pixel

La couleur, la luminance. La vision humaine est fondamentalement tri-chromique : elle est basée sur les réponses de trois types différents de cônes de photo-récepteurs situés dans notre rétine (LMS). Pour des raisons historiques, la plupart des caméras fournissent des informations de couleur dans un système RGB mais il existe bien d'autres espaces de représentation (par exemple XYZ, les systèmes de couleurs opposées, $L^*u^*v^*$, $L^*a^*b^*$, HSV, HSI, et HSL) [136, 21]. Remarquons également les modèles de couleurs gaussiens qui correspondent à une nouvelle théorie de la mesure de la couleur basés sur la théorie espace-échelle dans le domaine spatio-spectrale [51].

Contours et gradients. Les dérivées de l'image sont des données essentielles pour décrire la structure locale des images et leurs applications en vision par ordinateur sont vastes : détection des contours, extraction de caractéristiques, flux optique, la segmentation d'image et la détection d'objet. Les opérateurs de gradient les plus populaires sont ceux de Sobel [105] et de Canny [22] et il existe plusieurs approches dédiées à la couleur : l'approche vectorielle de Di Zenzo [35], l'approche de Carron [24] qui fusionne les gradients de teinte, saturation et luminance en accordant une importance plus grande aux couleurs de forte saturation.

Texture. Classiquement, les bancs de filtres tels que celui de Gabor [42] sont efficaces mais difficiles à utiliser dans des applications temps-réel. Récemment des méthodes plus économiques et tout aussi discriminantes telles que les Motifs Binaires Locaux LBP (pour *Local Binary Patterns*) [101] ont vu le jour. L'opérateur consiste à analyser un voisinage

circulaire autour d'un pixel central. La valeur du pixel central est utilisée comme un seuil pour ses voisins. Si un voisin est de valeur supérieure, alors le code qui lui sera associé dans le LBP est 1, et sinon 0. Enfin, chaque bit dispose d'un poids en fonction de sa position dans le voisinage, et l'on peut en déduire une valeur décimale caractéristique de ce motif local. Il s'agit donc d'une représentation à la fois compacte et peu sensible aux changements de contraste. Après une simple normalisation, ils deviennent invariants aux changements de rotation.

Primitives locales et points d'intérêt. Il existe un nombre considérable de méthodes de détection et de description de points d'intérêt : Harris [124], SIFT [89], SURF [10] et FAST [119].

Information de mouvement. Dans le cas d'une caméra fixe, les informations de mouvement peuvent être extraites aisément pas soustraction de fond. Dans un cadre plus général, et en particulier lorsque la caméra est mobile, on s'intéressera plutôt à des approches de calcul de flot optique [66, 90].

Information de profondeur. Enfin, avec l'émergence des capteurs RGB+D de type Kinect, il est désormais devenu aisé d'obtenir des cartes de profondeurs. Les méthodes alternatives, basées sur la stéréo-vision et la lumière structurée offrent par ailleurs des possibilités étendues dans certaines conditions d'acquisition, pour des distances capteur-objet supérieures à 5 mètres ou encore en extérieur.

0.2.2 Représentation d'un objet

À partir des informations disponibles au niveau pixel, différentes représentations plus globales sont possibles. L'objet peut être directement représenté par le bloc de pixels qui lui est associé dans l'image, où chaque pixel porte une information de luminance, de couleur (dans différents espaces de représentation) ou de profondeur par exemple. Ensuite, il peut être décrit par un histogramme, de couleur [54] ou de gradients orientés HOG [34] ou par son contour [76] ou bien encore par une matrice de covariance de caractéristiques pixelliques [139]. Comparée aux représentations par histogrammes, ces derniers descripteurs sont de dimension plus faible et leur taille ne dépend pas de la taille initiale de l'objet dans l'image. De plus, ils peuvent être calculés rapidement par l'usage d'images intégrales. C'est sur cette représentation que s'appuie une grande partie de nos travaux de thèse.

0.3 Détection et mise en correspondance

La reconnaissance visuelle est un problème étroitement lié à l'apprentissage des catégories visuelles à partir d'un ensemble limité d'instances. Typiquement deux approches sont utilisées pour résoudre ce problème: l'apprentissage des catégories génériques et la ré-identification d'instances d'un objet particulier. Dans le dernier cas, il s'agit de reconnaître un objet en particulier: le portrait sur un magazine, une liste de personnes enregistrées sur une base de données, des monuments connus tel que la tour Eiffel. D'autre part la reconnaissance générique consiste à retrouver toutes les instances d'objets qui appartiennent à la même catégorie conceptuelle : toutes les voitures, les piétons, les oiseaux, etc. Pour la reconnaissance spécifique, les algorithmes visent à trouver des correspondances ou à faire du *matching* en utilisant des descripteurs propres à l'instance observée. La reconnaissance générique utilise des modèles statistiques de l'apparence et la forme des objets, afin de chercher dans de nouvelles images inconnues les zones de forte similarité avec le modèle statistique. Ce type d'apprentissage requiert l'accumulation d'un ensemble d'images d'entraînement pour extraire ou construire le mod-

èle de chaque catégorie. Trouver les correspondances existantes entre deux (ou plusieurs) images n'est pas un problème aisé. Un objet peux générer deux instances complètement différentes en fonction des situations non contrôlées telles que l'illumination, le point de vue de la camera, la position de l'objet, les occultations et l'encombrement.

Les approches de classification les plus répandues dans la communauté de la vision par ordinateur sont les approches SVM (pour *Support Vector Machine* en anglais) et Adaboost. Dans le cadre de notre étude concernant l'utilisation des matrices de covariance, nous nous intéressons également à la classification dans les espaces Riemaniens. Des descriptions plus complètes aux SVM et à leur implémentation peut être trouvée dans [88] et [17]. Dans le domaine de la vision par ordinateur, on se réfère très souvent aux travaux de [34] sur la classification piéton/non piéton par HOG (histogrammes de gradients orientés) associés aux SVM. Concernant Adaboost, les travaux précurseurs concernent la détection de visages en utilisant des ondelettes de Haar [144].

Enfin, la classification piéton/non piéton a également été effectué dans un espace Riemanien [140] en utilisant une représentation par matrices de covariance. L'approche est inspirée par l'algorithme *LogitBoost* [48] qui opère sur les espaces vectoriels.

Concernant les métriques utilisées, de nombreuses méthodes ont été proposées par la communauté de la vision par odinateur. Quatre approches ont particulièrement inspiré nos travaux.

Appariement de blocs ou ***Template matching*****.** Il s'agit de comparer directement les images associées aux objets en utilisant de simples mesures, telles que l'erreur quadratique moyenne *MSE* (pour *Mean Squared Error*) , la corrélation normalisée croisés NCC (pour *Normalized Cross-Correlation*) ou encore l'indice de similarité de structure $SSIM$ (pour *Structural SIMilarity index*). Ces approches s'avèrent plus adaptées pour mettre en correspondance ou suivre temporellement des zones de petite taille et des objets planaires dont le mouvement est rigide.

Le flot optique médian plus populairement connu sous le nom de *Flock of Trackers* FoT est une approche capable de suivre des objets non-rigides par analyse du flot optique obtenu par l'approche de suivi KLT (Kanade-Lucas-Tommasi) en utilisant l'opérateur médian pour estimer la direction de déplacement de l'objet. Quand une cible est détectée, son rectangle englobant est partitionné en quadrants. Chacun d'entre eux est suivi par KLT, les 50% de points les plus erronés sont écartés, et le reste participe au calcul du déplacement global. Cette méthode apporte une certaine robustesse par rapport aux occultations partielles. Elle est adaptée dans le cas d'objets rigides montrant des points saillants.

Appariement d'histogrammes et ***Mean-Shift*****.** Dans le cas du suivi *Mean-Shift* [30], l'objet est modélisé par une représentation couleur-espace, et le suivi est fait par descente de gradient utilisant la distance de Bhattacharyya. L'histogramme est généralement quantifié afin d'assurer une exécution temps-réel et d'éviter les représentations creuses. Cette approche est particulièrement adaptée aux objets dont les couleurs sont caractéristiques par rapport à celles du reste de l'image. Elle permet une bonne résilience vis-à-vis des déformations de l'objet.

Le suivi par covariance. L'objet à suivre est représenté par sa boîte englobante et en chaque point (x, y) de l'objet est calculé un vecteur de caractéristiques $F(x, y)$. La combinaison originalement proposée par Tuzel *et al*. [139] continent la position des points (x, y), leur couleur (composantes RGB) et la norme des 1ère et 2nde dérivées d'intensité par rapport à x et y. Ainsi, chaque point est converti en un vecteur de dimension 9 :

$$F(x,y) = \left[\begin{array}{ccccccccc} x & y & R(x,y) & G(x,y) & B(x,y) & \left|\frac{\partial I(x,y)}{\partial x}\right| & \left|\frac{\partial I(x,y)}{\partial y}\right| & \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right| & \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right| \end{array} \right]^T,$$

L'objet est ensuite représenté par la matrice de covariance de taille 9×9 obtenue à partir de l'ensemble de ces vecteurs. Le suivi consiste alors à retrouver dans la nouvelle image la région dont la matrice de covariance est la plus similaire. Cela peut être fait par une recherche exhaustive [139] à la fois sur la position et sur l'échelle. Cette approche est performante pour suivre tout type d'objet, particulièrement lorsqu'il n'y a aucune information disponible sur la dynamique de l'objet ou que celui subit des mouvements erratiques entre deux images. Plusieurs approches ont également été proposées pour incorporer des données concernant la dynamique de l'objet et estimer le déplacement le plus probable.

0.4 Adaptation au contexte et commutation de méthode

Aucune des méthodes de suivi évoquées dans la partie 0.3 n'est applicable de manière universelle, chaque algorithme ayant ses avantages et ses limites. Les performances qu'ils offrent sont bien souvent dépendantes de l'application et il n'existe pas d'algorithme qui puisse être considéré comme *supérieur* aux autres. Quelques-uns des aspects à prendre en compte dans la définition d'un algorithme de suivi sont les suivants : la nature de la cible, les changements de conditions d'éclairage, les variations de pose ou d'apparence, les déformations non -rigides et des occlusions totales ou partielles. Malheureusement, dans de nombreuses applications telles que la vidéo-surveillance, il n'y a aucune information préalable disponible sur la nature de la cible (rigide ou non-rigide) et l'estimation de ces informations à la volée peut poser un problème très difficile, c'est particulièrement vrai lorsque les objets sont détectés en utilisant des algorithmes de soustraction de fond au lieu d'un classificateur d'objets. Cette thèse soutient qu'une *bonne* méthode de suivi doit résulter de la combinaison de plusieurs approches, l'algorithme qui en résulte devant être capable de gérer une plus grande liste de difficultés que tous les algorithmes originaux tout en étant suffisamment rapide. L'objectif est d'obtenir un bon équilibre entre la robustesse et la précision tout en conservant des besoins en puissance de calcul aussi faibles que possible afin de maintenir l'exécution en temps réel. Ici trois approches sont proposées.

0.4.1 Adaptation aux changements de saturation

Principe. Au cours d'une séquence d'images il est possible de rencontrer de fortes variations de luminosité, qui ne sont pas toujours uniformes dans l'image. Il n'est donc pas aisé de définir au préalable les meilleures caractéristiques à utiliser : couleur ou luminance.

Le traitement couleur est généralement plus coûteux en temps de calcul tandis que la luminance est par essence sensible aux variations photométriques. La couleur permet de définir de précieux invariants couleur [54] qui sont malheureusement peu fiables dans le cas de faibles saturations. Nous proposons une approche de suivi qui sélectionne automatiquement les caractéristiques en fonction de leur pertinence : la luminance sera utilisée aux pixels pour lesquels la saturation de la couleur est faible et la couleur sera utilisée dans les autres cas. Cette approche a été validée dans le cadre du suivi de points de type *KLT* [90, 133]. La composante couleur utilisée a été choisie de manière à être compacte et invariante aux changements d'illumination.

Choix de la composante couleur. Une façon directe de séparer l'information de chrominance (invariante) de celle de luminance est de normaliser (norme L_1) chaque composante (R, G, B) en les divisant par la somme des trois, qui correspond à l'intensité de la couleur. On obtient les composantes (r, g, b) qui ne dépendent plus que de l'*albedo*, propriété

intrinsèque de réflectance du matériau.

À partir de ces trois composantes, nous calculons la valeur scalaire $L_1 = max(r, g, b)$ qui représente la composante la plus saturée parmi les trois. Notons que cette information de couleur est plus facile à manipuler que les trois composantes et conserve l'invariance. Elle s'avère toutefois moins discriminante. Dans le cas du suivi KLT où la cohérence temporelle inter-image est pleinement exploitée, cet aspect a un impact relativement faible sur les résultats de suivi.

Mesure de pertinence de la couleur. L'approche est inspirée des travaux de [23] concernant la détection de contours dans l'espace *HSV* en fusionnant les information de teinte, saturation et luminance à l'aide d'une fonction de pertinence dépendant de la saturation, typiquement une fonction sigmoïde. Dans notre cas, la fonction de coût utilisée dans l'approche de suivi par KLT est modifiée de manière à accorder une importance plus forte à L_1 lorsque la saturation est élevée et à la luminance sinon.

Résultats. Á partir d'expériences menées sur plusieurs séquences et en comparant les résultats de KLT appliqué dans différents espaces couleur, nous avons pu constater un très bon compromis entre le nombre de points correctement suivis et le temps d'exécution. Utilisé seul, l'invariant L_1 échoue lorsque la saturation est faible. La méthode proposant une association entre L_1 et la luminance I est capable de s'adapter à cette situation. Elle permet de maintenir les performances du suivi dans le cas de séquences saturées et améliore les résultats de manières significatives dans le cas de séquences peu saturées. D'autre part, utiliser les trois composantes (r, g, b) ne permet pas toujours de meilleurs résultats que L_1 mais augmente toujours, et de manière significative, les temps de calcul. Enfin, nous avons montré la pertinence du choix de L_1 comparé à la teinte pour l'ensemble des séquences de test. Ceci est probablement du au caractère bruité de cette composante.

0.4.2 Première coopération FoT + CT

Afin de répondre au problème du suivi d'objet sur de longues durées, nous proposons une approche fondée sur le suivi *FoT* évoqué dans la partie 0.3, qui analyse de manière robuste le flot optique associé aux points de l'objet. D'autre part nous utilisons le descripteur par covariance permettant de modéliser l'apparence globale (couleur et texture).

L'approche *FoT* estime de manière robuste le mouvement apparent de l'objet et permet théoriquement de résister aux occultations partielles jusqu'à 50% de masquage [146]. La restriction principale de cette méthode est que l'objet doit montrer un nombre suffisant de points saillants pour que le suivi par KLT soit fiable. Elle s'avère plus adaptée pour suivre des objets dont le mouvement est spatialement uniforme, autrement dit les objets rigides.

Le suivi par covariance *CT* apparaît comme une alternative qui de manière compacte modélise la cible par les corrélations au sein d'un ensemble d'attributs préalablement sélectionnés (communément il s'agit des coordonnées spatiales, de la couleur et de la couleur). Contrairement à *FoT*, de bonnes performances sont atteintes même pour des objets faiblement texturés ou pour des objets au mouvement non-rigide. Sa faiblesse principale est le temps d'exécution et ceci devient plus important lorsque la recherche de la nouvelle position de l'objet se fait de manière exhaustive dans l'image. Notons toutefois que *CT* peut être accélérée par un couplage avec d'autres méthodes de suivi telles que le filtrage particulaire [87] qui permet de limiter l'espace de recherche.

Le mécanisme proposé est décrit graphiquement par la figure 3-7. Tout d'abord, *FoT* est exécuté et l'analyse de la variance des vecteurs de mouvement permet d'identifier l'objet comme rigide ou non-rigide. S'il est rigide, *FoT* est exécuté tant que les résidus du suivi KLT (caractéristiques de l'erreur) restent inférieurs à une valeur seuil $\mathcal{R}_{max}$ ou que la dissimilarité

$\mathcal{D}_t$ entre deux matrices de covariance successives est inférieure à une valeur seuil $\mathcal{D}_{max}$. Si l'objet est non-rigide ou que *FoT* a échoué (à cause d'une occultation par exemple), *CT* est exécuté pour retrouver l'objet pendant une période de temps T. Si l'objet n'est pas retrouvé après cette période il est considéré comme définitivement perdu.

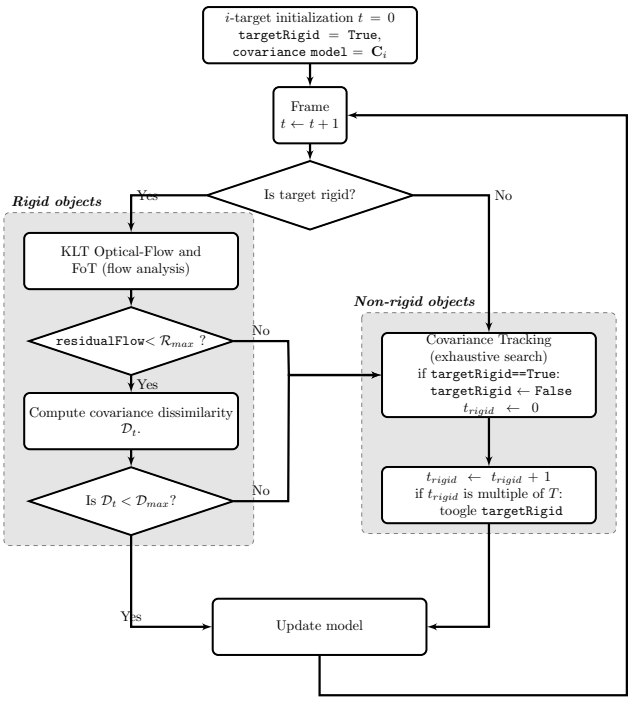

Figure 0-1: Mécanisme de coopération *FoT+CT*.

0.4.3 Deuxième coopération MS + CT

Les approces *MS* et *CT* possèdent également des avantages complémentaires. *MS* représente la cible par sa distribution couleur qui est quasiment insensible aux distorsions géométriques. *MS* est très rapide mais son pouvoir de séparation est relativement faible, ne permettant pas de faire face aux occultations et aux déplacements importants de l'objet dans l'image. Une association avec *CT* doit permettre de pallier cette faiblesse. Ici aussi la dissimilarité entre matrices de covariance est utilisée comme indicateur de précision de *MS* et l'algorithme *CT* est exécuté pour retrouver la cible après qu'elle soit perdue. La figure 3-12 illustre le mécanisme de commutation *MS+CT*.

Quatre séquences ont été utilisées lors des expérimentations. *Pedxing* et *Panda* sont des séquences où la caméra est quasiment fixe tandis que les séquences *Motocross* et *Carchase* correspondent à des vidéos prises d'une caméra mobile (soit fixée sur le cadre d'une moto, soit sur un hélicoptère). La tableau 3.3 analyse pour chacune d'entre elles le nombre de cycles par seconde (cpp) utilisé par chaque partie de l'algorithme de commutation *MS+CT* durant toute la durée de la séquence : *MS* et *CT* selon que *MS* ou *CT* est exécuté dans l'image, la mise à jour du modèle et le calcul du critère de similarité (qui est fait dans chaque image). Le pourcentage d'images pour lequel *MS* et *CT* sont sélectionnés par rapport à la durée totale de la séquence est également montré. Dans les séquences où la caméra est stable (*Pedxing* et *Panda*), *MS* est exécutée la plupart du temps étant donné que le mouvement est régulier. Lorsque l'objet n'est plus visible, *CT* est exécuté

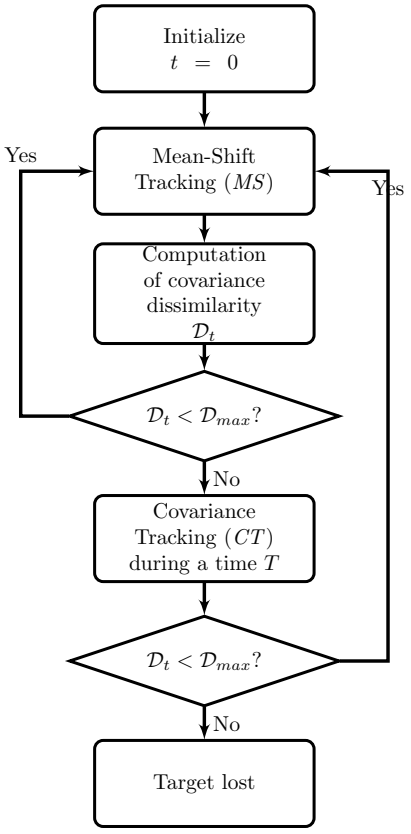

Figure 0-2: Mécanisme de commutation *MS*+*CT*.

pour retrouver la cible. Lorsque la caméra est placée sur un porteur mobile (*Motocross* et *Carchase*), le mouvement est complexe et imprédictible. Dans ce cas, *CT* est exécuté plus fréquemment.

0.4.4 Conclusion

Trois contributions ont été apportées en ce qui concerne le suivi adapté au contexte. Ces approches apportent l'un et/ou l'autre des avantages suivants :

1. *La robustesse*, soit par rapport aux occultations de l'objet, les mouvements abrupts, les changements de conditions d'acquisition.

2. *La rapidité*: dans la méthode de suivi adaptée aux changements de saturation, la couleur (qui requiert plus de temps de calcul) n'est utilisée que si nécessaire. Ensuite, les approches *MS* et *FoT* sont plus rapides mais moins robustes que *CT* seul. La combinaison de *CT* avec l'une des ces approches permet donc de réduire le temps d'exécution sans détériorer la qualité du suivi. Le suivi sera d'autant plus rapide que *MS* ou *FoT* seront appelées fréquemment. Cela correspond aux cas où la caméra est statique et que l'objet n'est pas occulté.

Table 1: Analyse de la procédure de coopération *MS+CT*. Ces résultats ont été obtenus sur un processeur *Nehalem*. Le tableau montre pour chaque séquence : la taille de l'objet suivi à l'initialisation ; le pourcentage d'images (%im) pour lequel les algorithmes *MS* et *CT* algorithmes sont exécutés ; le nombre total de cycles par pixel (*cpp*) passé par chaque partie de l'algorithme (*MS*, *CT*, mise à jour du modèle de covariance, calcul de la similarité) pendant toute la durée de la séquence.

Séquence	Taille	# im	*MS*		*CT*		M.à.J modèle	Similarité	Temps
	objet		*cpp* ($\times 10^6$)	%im	*cpp* ($\times 10^6$)	%im	*cpp* ($\times 10^6$)	(*cpp* $\times 10^6$)	[ms]/im
Pedxing	93×35	189	7.20	70	60.59	30	1.07	2.45	10.1
Panda	23×28	940	2.48	88	8.74	12	0.57	0.33	1.6
Motocross	64×47	2665	5.07	12	30.04	88	0.005	1.06	10.6
Carchase	45×97	2999	0.99	13	19.19	87	1.03	1.44	7.3

0.5 Suivi multiple temps-réel par covariance et ré-identification

L'un des objectifs de ce chapitre est de trouver la meilleure combinaison d'opérateurs de texture et de couleur qui améliorent à la fois la robustesse et le caractère distinctif du descripteur de covariance tout en gardant sa compacité et sa rapidité de calcul ainsi qu'une bonne robustesse vis-à vis des changements de couleur et d'illumination.

Ces matrices de covariance sont ensuite utilisées dans le cadre de la ré-identification d'objets et du suivi multiple. À cet effet, une méthode de suivi multi-objet originale est proposée, elle minimise une fonction d'énergie discrète combinant les probabilités d'association des trajectoires aux différentes cibles ainsi que leur concordance avec les modèles d'apparence décrits par des matrices de covariance.

0.5.1 Étude du choix des primitives texture

Plusieurs approches existent pour inclure des informations de texture au sein des matrices de covariance en vue d'améliorer leur pouvoir discriminant, citons *GRMC* (pour *Gabor Region Covariance Matrix* en anglais) utilisant les réponses issues des bancs de filtres de Gabor (voir partie 0.2.1) [104, 135], ce qui mène à une matrice de covariance de taille relativement importante et donc à des temps de mise en correspondance prohibitifs. Certaines publications remplacent les filtres de Gabor par des LBP (voir partie 0.2.1) comme dans la méthode *LBCM* (pour *Local Binary Covariance Matrix* en anglais) de [59], le *GLRCD* (pour *Gabor-LBP based Region Covariance Descriptor* en anglais) [154]). C'est également l'approche que nous avons privilégiée puisqu'elle répond à nos objectifs d'efficacité et de compacité.

Le choix de la *meilleure* façon de présenter l'information de texture sous la forme de motifs binaires locaux dans le descripteur de la matrice de covariance n'est pas évident à faire. *GLRCD* utilise une introduction brute des valeurs décimales des *LBP*s, valeurs très instables dans le cas des rotations. En outre, les opérations arithmétiques classiques ne sont plus applicables (ajouter ou sommer plusieurs valeurs décimales de *LBP* n'a pas de signification en termes de texture). *LBCM* utilise chaque bit dans le modèle de *LBP* comme un élément indépendant formant une chaîne de P bits. Ceci s'avère plus stable et il devient cohérent d'effectuer des opérations arithmétiques de façon indépendante pour chaque bit de la chaîne. Le problème est que le nombre de fonctions à l'intérieur de la matrice de covariance augmente avec le nombre P de bits considérés pour le codage des *LBP*s. Cela a un impact significatif sur la vitesse d'exécution .

Dans ces travaux de thèse, nous proposons une nouvelle façon d'introduire les LBPs dans les matrices de covariance

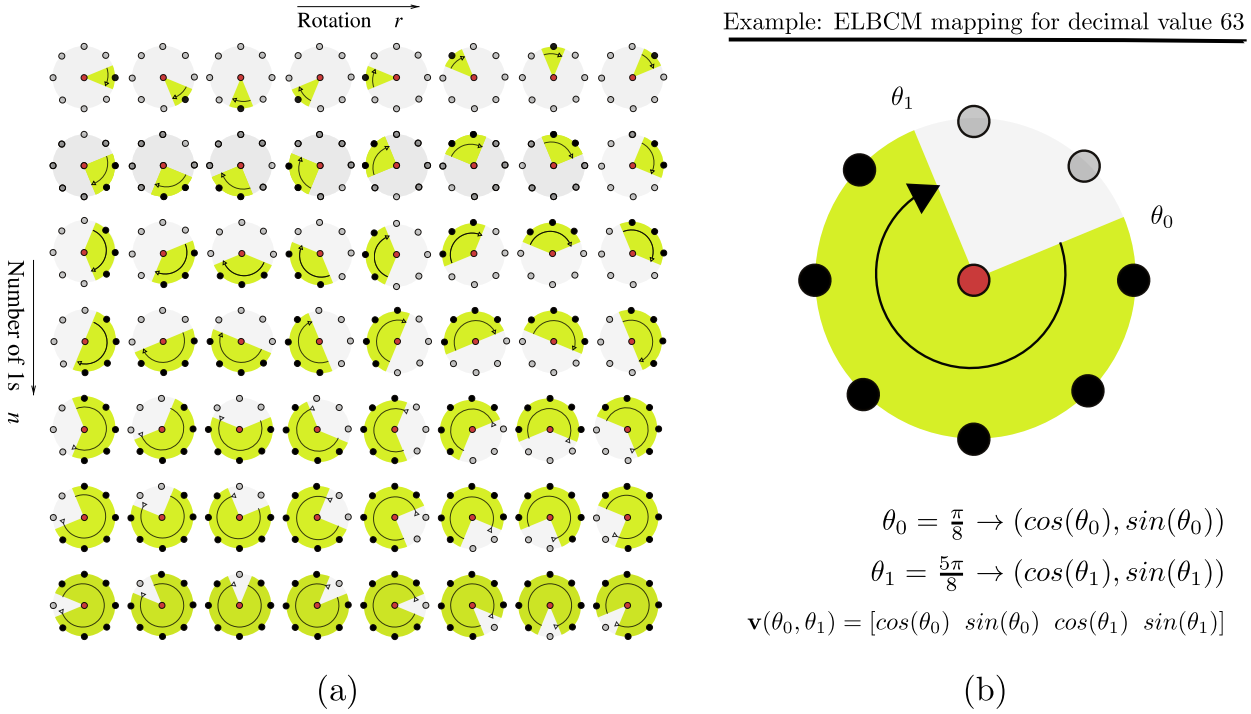

Figure 0-3: (a) Ensemble des 56 motifs *LBP* uniformes dans un voisinage $(8, R)$. Pour chaque motif, une flèche circulaire indique le début et la fin de l'angle (θ_0 and θ_1). (b) Dans la méthode *ELBCM*, un motif *LBP* défini par deux angles θ_0 et θ_1 est décrit par $\mathbf{v}(\theta_0, \theta_1) = [cos(\theta_0) \;\; sin(\theta_0) \;\; cos(\theta_1) \;\; sin(\theta_1)]$. Le motif *LBP* associé à la valeur décimale 63 prend la valeur $\mathbf{v}(\pi/8, 5\pi/8) = [cos(\pi/8) \;\; sin(\pi/8) \;\; cos(5\pi/8) \;\; sin(5\pi/8)]$.

par un nouveau descripteur de covariance appelé *ELBCM* (pour *Enhanced Local Binary Covariance Matrix*). Il utilise les angles définis par les motifs *LBP* uniformes présentés sur la figure 4-1. L'utilisation des formules trigonométriques est coûteuse en temps de calcul, mais ce problème est complètement résolu en utilisant une table de correspondance associant directement à chaque motif les valeurs de cosinus et sinus des angles θ_0 et θ_1.

ELBCM présente de multiples avantages par rapport aux précédents descripteurs de texture à base de covariance. Il est plus compact (7 composantes contre 11 pour *LBCM*) et plus stable car moins affecté par les petites rotations, qui impactent seulement les angles θ_0 et θ_1 alors que pour *LBCM* les mêmes rotations affectent irrégulièrement la valeur des bits dans le vecteur descripteur *LBP* en fonction de leur position. Un avantage supplémentaire du descripteur de la *ELBCM* est que sa taille est complètement indépendante du nombre de P voisins utilisés dans le modèle LBP.

L'apport de ELBCM a été montré sur trois applications : la classification de texture, la classification d'expressions faciales, le suivi d'objets. Les résultats obtenus démontrent la généricité de cette approche et les bonnes performances tant en termes de temps d'exécution qu'au niveau du pouvoir discriminant de la représentation.

0.5.2 Étude des primitives couleur

Après l'étude sur la texture, nous nous intéressons au choix des composantes couleur les plus pertinentes à introduire dans la matrice de covariance en vue d'une mise en correspondance efficace c'est-à-dire précise (qui produit peu d'ambiguïtés d'appariement) et rapide. Quatre représentations couleur sont utilisées : RGB, HSV, les modèles de couleur gaussiens

évoques dans la partie 0.2.1 et l'invariant L_1 introduit dans 0.4.1.

Deux applications sont visées : la classification de textures couleur et le suivi de visage dans le cas de conditions d'illumination variables. Les expérimentations réalisées dans le premier cas nous permettent de constater l'apport de l'approche $ELBCM$ par rapport aux approches comparables. Dans la seconde application, seules les combinaisons utilisant la composante invariant $L_1 = max(r, g, b)$ permet de résister aux forts changements d'illumination. Pour des conditions d'acquisition constantes, la méthode $ELBCM$ offre de bons résultats pour la plupart des espaces couleur utilisés.

0.5.3 Ré-identification d'objets

Principes. La mise en correspondance est rendue difficile lorsque les objets doivent être re-détectés et ré-identifiés à partir d'une seconde caméra sous des conditions d'acquisitions différentes en termes d'illumination et de perspective, à des instants très différents. Le descripteur de l'objet doit alors être suffisamment caractéristique de l'objet. À cet effet, Bak *et al*. ont proposé dans [6] un maillage dense de matrices de covariances [6] ainsi qu'une opération de moyenne appelé MRC (pour *Mean Riemannian Covariance* en anglais), qui permet de fusionner au cours du temps les informations d'apparences issues de ces multiples échantillons. En partant de cette approche, nous proposons de réduire le temps d'exécution par l'usage d'un nouvel arrangement de matrices *MRC*. Les régions d'intérêt associées aux objets sont redimensionnées (typiquement 96×128 pixels), puis des anneaux concentriques de rectangles sont positionnés autour du centre de l'objet, avec des aires qui augmentent de manière exponentielle pour permettre un recouvrement entre deux couches successives. Cela est inspiré des méthodes de FREAK et DAISY [1, 132] mais pour des régions rectangulaires, permettant des calculs rapides grâce à la méthode d'image intégrale.

Résultats. Quelques résultats sont visibles sur la figure 4-19. Afin de valider notre approche de ré-identification nous utilisons les courbes CMC (pour *Cumulative Matching Characteristic* en anglais) utilisées également dans [6] et [58]. Elles representent en ordonnée le pourcentage de fois où l'identité réelle apparaît parmi les n meilleurs appariements, avec n en abscisse. Ces tests sont réalisés sur les bases de données ETHZ [121] et PETS'09 L1-Walking-Sequence 1 [39]. Les taux de ré-identification obtenus sont comparables à ceux rapportés dans [6] mais en utilisant 75% de matrices de covariance en moins et des matrices plus compactes.

0.5.4 Suivi multiple

Le but de notre algorithme de suivi multi-objet est de détecter, d'identifier et de tracer les positions des objets. Idéalement le nombre de trajectoires par objet de un exactement, mais les inter-occlusions, les disparitions, l'apparition de faux positifs de la méthode de détection peut provoquer des dérives et des fragmentations des trajectoires. La possibilité d'inclure des informations relatives à l'apparence dans la fonction d'énergie a été laissée ouverte dans [3]. Cette idée est récupérée ici et enrichie en efficacité et en robustesse grâce à la représentation des cibles par les matrices de covariance.

Image par image l'algorithme évalue toutes les correspondances possibles entre les détections $\mathbf{d}_i$ et les trajectoires $\mathbf{p}_j$. De nouvelles pistes sont créées lorsque cela est nécessaire. Finalement, l'algorithme génère une cartographie dynamique contenant l'ensemble couples $M_T = \{(\mathbf{d}_i, \mathbf{p}_j)\}$ qui minimise une fonction d'énergie discrète. Cette fonction d'énergie $E(\mathbf{d}_i, textbf p_j)$ privilégie les configurations plausibles et pénalise celles qui sont incohérentes. Notre fonctionnelle d'énergie comprend la plupart des termes utilisés dans [3] : un terme de détection fondée sur la détection de données

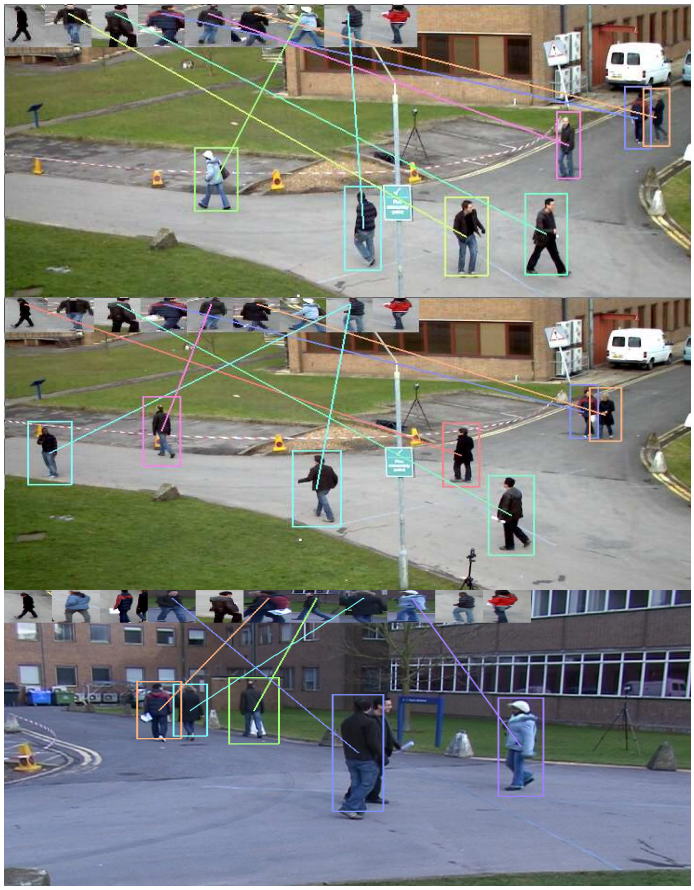

Figure 0-4: Exemples de résultats obtenus sur la base de données PETS'09, montrant une ré-identification à différents instants et à différents points de vue.

image E_{det}, un terme physique qui modélisant la dynamique des objets (vitesses linéaire et angulaire) E_{dyn} , un terme d' apparence E_{app} qui mesure la similarité entre la cible à l'emplacement actuel et le modèle de covariance associé aux trajectoires $\mathbf{p}_j$. Le dernier terme pénalise les créations (ou disparitions) de nouvelles pistes favorisant les trajectoires persistantes et continues E_{add}. Tous les termes mentionnés ci-dessus sont accumulés tel que l'exprime l'équation suivante

$$E(\mathbf{d}_i, \mathbf{p}_j) = \underbrace{E_{det}}_{\text{confiance de détection}} + \underbrace{E_{dyn}}_{\text{modèle dynamique}} + \underbrace{E_{app}}_{\text{similarité d'apparence}} + \underbrace{E_{add}}_{\text{modèle de persistance}} . \quad (1)$$

Une de nos contributions est de rajouter le terme d'énergie associé à l'apparence, celle-ci étant modélisée par les matrices de covariance. Ces matrices appartient à un espace Riemanien formé par l'ensemble des matrices définies positives (*SPD*). Une variété de métriques et de dissimilarités a été proposée dans la litérature afin de comparer deux matrices dans cet espace. Parmi elles, la mesure de dissimilarité *Jensen-Bregman LogDet Divergence* [29] s'avère particulièrement intéressante car elle est rapide à calculer et elle jouit de très bonnes propriétés de robustesse. Concernant la minimisation de (4.11), nous utilisons l'algorithm de Munkres [99] (la méthode Hongroise) utilisée par ailleurs dans [72].

Résultats. Nous avons évalué l'algorithme en utilisant les métriques proposées dans [127] et [83] et en partant de bases de données très largement utilisées dans la communauté ainsi que des séquences enregistrées dans notre laboratoire

en utilisant le premier point de vue : la base PETS 2009-S2L1-V1 [1], la base TUD [2] avec les séquences TUD-Campus, TUD-Crossings et TUD-Stadtmitte. La vérité-terrain est disponible pour toutes ces bases de données. Nos résultats sont visible en-ligne[3]. Sur la base de ces expériences, les résultats montrent que notre approche de suivi n'est pas aussi précise que celles proposées par [95] ou [12], mais elle a l'avantage important de travailler à la volée, en estimant les trajectoires image par image. Dans la méthode proposée dans [95], le suivi nécessite l'ensemble des images de la vidéo pour estimer le nombre de trajectoires optimal et leur tracé le plus probable.

0.5.5 Bilan

Dans cette partie, une étude a été menée pour proposer un nouveau descripteur par matrice de covariance en proposant une combinaison de composantes couleur et de texture, dans le but d'assurer un bon pouvoir discriminant tout en réduisant la taille de la représentation. Concernant la texture, le descripteur $ELBCM$ est proposé. Les résultats obtenus en classification de texture, suivi et ré-identification d'objets sont très prometteurs. Ce descripteur est également utilisé dans le cadre du suivi multi-cibles, où nous avons proposé une nouvelle fonction de coût intégrant un modèle d'apparence basé sur la divergence de Jensen-Bregman.

0.6 Implementation temps-réel: optimisations logicielles et transformations algorithmiques

Dans cette partie nous décrivons les transformations et accélérations opérées sur l'algorithme de suivi par covariance. Notons que ce travail a également été mené dans le cas du suivi *Mean-Shift* en vue de faire coopérer les deux méthodes [79].

Afin d'évaluer les performances des algorithmes et l'impact des optimisations, des comparaisons ont été faites sur trois générations de processeurs : Penryn/Yorfield, Nehalem/Bloomfield et Sandy-Bridge qui sont tous des processeurs multi-cœurs, excepté le Nehalem. Le processeur du Sandy-Bridge a la possibilité d'être *over-clocké*. La mémoire RAM peut être plus *over-clockée* que le processeur ce qui signifie qu'à une fréquence plus élevée (4.4 GHz) la RAM semble plus rapide ce qui résulte en un nombre réduits de nombre de cycles.

Deux stratégies peuvent être exploitées pour optimiser CT. La première consiste à réaliser du *multi-threading* en parallélisant la boucle principale de traitement. Cela est effectué avec OpenMP. La seconde nécessite une transformation de la mise en forme des données en mémoire SdM→MdS (*Structure de Matrices* vers *Matrices de structures*). Le but de cette manipulation est de transformer un ensemble de matrices indépendantes en une matrice unique, où chaque cellule combine les éléments de toutes les matrices dans une structure de données. La contribution d'une telle transformation est de tirer parti de la performance du cache en exploitant la localité spatiale et temporelle. L'algorithme de suivi par covariance se compose de trois parties :

1. le calcul des produits point à point pour toutes les primitives,

[1] http://www.cvg.rdg.ac.uk/PETS2009
[2] http://www.mis.tu-darmstadt.de/node/428
[3] http://andresromeromier.wikispaces.com/

2. le calcul de l'image intégrale des primitives,

3. le calcul de l'image intégrale des produits.

Le produit de caractéristiques et de sa transformation sont décrits dans les algorithmes 16 et 17. Les notations utilisées sont les suivantes :

- h et w : hauteur et largeur de l'image
- n_F : nombre de caractéristiques (primitives) image
- n_P : nombre de produits de caractéristiques, $n_P = n_F(n_F + 1)/2$,
- F : cube (*SdM*) ou matrice (*MdS*) de caractéristiques,
- P : cube (*SdM*) ou matrice (*MdS*) de produits de caractéristiques,
- I_F et I_P : deux cubes (ou matrices) d'images intégrales (à partir de F ou P).

Grâce à la commutativité de la multiplication, la moitié seulement des produits doivent être calculés (la boucle sur k_2 commence à k_1, ligne 3). Comme les deux dernières étapes sont similaires, nous ne présentons qu'une version générique du calcul de l'image intégrale (Algo. 18) et sa transformation (Algo. 19).

Algorithm 1: Optimisation ode CT - version *SdM* du produit de caractéristiques.

```
1 k ← 0
2 foreach k1 ∈ [0..nF − 1] do
3     foreach k2 ∈ [k1..nF − 1] do
4         [ point-to-point multiplication ]
5         foreach i ∈ [0..h − 1] do
6             foreach j ∈ [0..w − 1] do
7                 P[k][i][j] ← F[k1][i][j] × F[k2][i][j]
8                 k ← k + 1
```

Algorithm 2: Optimisation de CT - version *MdS* du produit de caractéristiques.

```
1 foreach i ∈ [0..h − 1] do
2     foreach j ∈ [0..w − 1] do
3         k ← 0
4         foreach k1 ∈ [0..nF − 1] do
5             foreach k2 ∈ [k1..nF − 1] do
6                 P[i][j × nP + k] ← F[i][j × nP + k] × F[i][j × nP + k]
7                 k ← k + 1
```

Une fois que cette transformation est faite, nous utilisons les instructions SIMD dans les différentes parties de l'algorithme. Pour ce qui concerne le produit, les deux boucles internes à k_1 et k_2 sont totalement déroulée de manière à montrer la liste de toutes les multiplications et la liste des vecteurs à construire par des instructions de permutations[4]. Par exemple, pour

[4] Cela est fait par `_mm_shuffle_ps` en SSE

Algorithm 3: Optimisation de CT - version *SdM* de l'image intégrale.

```
foreach k ∈ [0..n − 1] do
    [classical in place integral image]
    foreach i ∈ [0..h − 1] do
        foreach j ∈ [0..w − 1] do
            I[k][i][j] ← I[k][i][j] + I[k][i][j − 1] + I[k][i − 1][j] − I[k][i − 1][j − 1]
```

Algorithm 4: Optimisation de CT - version *MdS* de l'image intégrale.

```
foreach i ∈ [0..h − 1] do
    foreach j ∈ [0..w − 1] do
        foreach k ∈ [0..n − 1] do
            I[i][j×n+k] ← I[i][j×n+k] + I[i][(j−1)×n+k] + I[i−1][j×n+k] − I[k][i−1][(j−1)×n+k]
```

une valeur typique de $n_F = 7$, il y a $n_P = 28$ produits, elles sont effectuées dans l'ordre suivant (les nombres entre crochets sont les indices des caractéristiques) :

$$[0,0,0,0] \times [0,1,2,3]; \quad [0,0,0,1] \times [4,5,6,1];$$
$$[1,1,1,1] \times [2,3,4,5]; \quad [1,2,2,2] \times [6,2,3,4];$$
$$[2,2,3,3] \times [5,6,3,4]; \quad [3,3,4,4] \times [5,6,4,5];$$
$$[4,5,5,6] \times [6,5,6,6].$$

Dans ce cas, le septième vecteur est rempli à 100 %, mais il deviendra sous-optimal si n_P n'est pas divisible par le cardinal du vecteur (4 avec SSE, 8 avec AVX). Les permutations sont réalisées en une seule instruction pour certains d'entre eux, et en deux instructions dans le pire des cas. Parce que certaines permutations peuvent être réutilisées pour réaliser d'autres permutations, une factorisation est faite sur toutes les permutations nécessaires. Par exemple, avec $n_F = 7$, quinze remaniements sont nécessaires.

La quantité de mémoire requise par CT est égale à $(n_F + n_P) \times$ `sizeof(float))` $\times\, h \times w$ octets. En supposant $h = w = 1024$, $n_F = 7$ et $n_P = 28$, l'algorithme a besoin de 140 Mo, beaucoup plus que la taille de la plus grande mémoire cache disponible ! Afin d'évaluer l'impact des optimisations, les trois versions de l'algorithme (SdM, MdS, $MdS + SIMD$) ont été comparées pour des tailles variant de 128×12 à 1024×1024. Les temps d'exécution obtenus sont reportés dans la tableau 2.

Table 2: Optimisation de CT : temps d'exécution (ms) pour une image 1024×1024 avec 7 caractéristiques.

Processeur	*mono-threading*			*multi-threading*		
	SdM	MdS	MdS+SIMD	SdM	MdS	MdS+SIMD
Penryn	219.6	144.5	117.7	137.3	140.0	110.7
Bloomfield	219.3	118.2	65.5	91.0	116.5	69.9
SandyBridge	103.6	52.5	30.9	43.5	51.1	31.0
SandyBridge+OC	77.7	41.0	25.7	33.6	40.5	25.9
Nehalem	103.8	79.7	48.6	29.8	63.6	38.0

La transformation *SdM* $\rightarrow$ *MdS* est très efficace avec une accélération de près de $\times 2$. La version SIMD permet une accélération de $\times 2.5$ pour des images 128×128. Lorsque les données ne rentrent pas dans le cache (pour des images

1024×1024), cette accélération descend à une valeur moyenne de $\times 1,6$. Lorsqu'ils sont combinés ensemble, l'accélération atteint un maximum de $\times 5,11$ pour des tailles 128×128 et $\times 3,35$ pour des tailles 1024×1024. En ce qui concerne le Nehalem octo-cœurs, les accélérations sont plus faibles, mais le temps d'exécution est plus faible aussi, en raison de ses bus de cache rapides. Le gain apporté par OpenMP est moins important excepté pour Nehalem comparé au gain du codage SIMD. Concernant les temps d'exécution, la transformation *SdM*$\rightarrow$*MdS* + SIMD est nécessaire pour permettre une exécution temps-réel (40 ms par image) pour des images 1024×1024. Dans le cas de l'implémentation mono-thread, seul le SandyBridge (avec instructions SSE) est temps-réel et peut permettre un gain de $\times 4.0$ après augmentation de la fréquence d'horloge.

L'exécution de CT a été testée sur les séquences utilisées dans [79] à la fois en version monochrome et couleur. À cause des performances réduites des versions multi-threadées, l'algorithme est finalement implémenté et évalué sans multi-threading. Après les différentes transformations algorithmiques, les tests effectués montrent des temps d'exécution allant de 3 ms à 11ms, rendant donc l'algorithme rapide et compatible avec une exécution temps-réel sur un cœur.