Suivi de motifs texturés : approche hybride texture/contours

Contour/texture approach for visual tracking

Lucie Masson Frédéric Jurie Michel Dhome

LASMEA - CNRS UMR 6602 - Université Blaise-Pascal

Campus Scientifique des Cézeaux 63170 Aubière
masson,jurie,dhome@lasmea.univ-bpclermont.fr

Résumé

Dans cet article, nous décrivons une méthode rapide hybride contour/ texture pour le suivi de motifs planaires en temps réel. Ces travaux constituent une extension intéressante des travaux que nous avons précédemment réalisés sur le suivi de motifs à partir de leur texture.

Cette nouvelle méthode possède les mêmes avantages de simplicité et d'efficacité que celle utilisant les textures : connaissant la différence entre deux images, il suffit d'un calcul élémentaire (multiplication matricielle) pour déduire les paramètres de mouvement du motif. Cette simplicité de traitement est rendue possible par l'utilisation d'une phase d'apprentissage hors ligne.

La contribution principale de cet article est l'ajout d'informations décrivant les contours ; cela nécessite un apprentissage et un calcul de la différence entre images spécifiques aux points de contour.

Mots Clef

suivi de motifs, suivi de contours, vision temps réel

Abstract

In this paper, we describe a real-time contours/texture tracking method for planar objects. This is an interesting extension of our work on texture tracking.

This new method is as simple and efficient that the one with texture only. Knowing the difference between two frames, we only need a matrix mutiplication to compute the movement parameters. This is due to the use of a off line learning stage before tracking.

The main contribution of this article is the addition of informations about edges. Informations about textures and egdes are mixed, and so tracking needs a specific learning phase and a specific difference between images computing.

Keywords

texture tracking, contour tracking, real-time vision

1 Introduction

Nous nous intéressons, depuis plusieurs années, au suivi temps réel de motifs texturés. Nous avons en particulier proposé [8] une méthode originale pour le suivi d'objets texturés. Elle permet de suivre le déplacement de motifs texturés avec une grande rapidité et une grande précision. Malheureusement, elle n'est pas applicable dans le cas de motifs peu texturés ou de couleur uniforme.

Or, dans de nombreuses applications industrielles ce type de suivi peut avoir un intérêt (contrôle de qualité sur des pièces manufacturées, par exemple). Certains motifs ne sont pas suffisamment texturés pour pouvoir être suivis par des algorithmes basés sur la luminance des motifs ; pour ces objets, il est plus indiqué de les suivre en utilisant leurs contours dans les images, alors que pour d'autres objets, c'est le contraire.

C'est pourquoi nous proposons ici une méthode hybride permettant de suivre des motifs visuels rigides en prenant en compte les caractéristiques les plus adaptées au motif à suivre.

1.1 Suivi de contour / suivi de texture

D'une manière générale, les algorithmes de suivi d'objets dans des images vidéo peuvent être vus comme des algorithmes d'optimisation. Il s'agit de déterminer des paramètres d'un vecteur d'état manière à optimiser le *recalage* du modèle sur une image. Ces paramètres d'état peuvent comporter des informations sur la position, l'orientation, les vitesses de l'objet et peuvent éventuellement décrire des changements d'aspect ou de propriétés des objets.

Il est possible de classer les algorithmes de suivi existant dans l'état de l'art en deux catégories, en fonction de la nature de la fonction à optimiser. Pour le premier type, il s'agit de minimiser des distances entre des primitives de l'image (points, lignes, b-splines, *etc.*) et des primitives d'un modèle de l'objet. Dans le second cas il s'agit de mesurer la ressemblance entre la fonction de luminance globale du motif et celle dans l'image étudiée.

Nous allons présenter quelques travaux significatifs illus-

trant ces deux classes d'algorithmes.

Suivi par minimisation d'une distance. Pour illustrer cette classe de méthodes, nous pouvons mentionner les travaux de Lowe [12] dans lesquels un modèle polyhédrique 3D d'objet est suivi dans une séquence d'images. L'objectif est ici de trouver la pose de l'objet telle que la distance entre les segments 3D du modèle projeté dans l'image et les segments de l'image soit la plus faible possible (maximisation d'une probabilité de correspondance entre primitives). Plus récemment, les travaux de Kollnig et Nagel [10] portent sur le suivi de véhicules à partir de modèles polyhédriques 3D des véhicules. L'originalité de ces travaux est d'utiliser l'information du gradient plutôt que de détecter les contours. La pose 3D du modèle est optimisée de manière à maximiser la probabilité qu'un point du modèle se projette sur un point de contour (gradient élevé) de l'image. Drummond et Cippola [2] présentent un système de suivi temps réel d'objets 3D à partir des contours. Ils proposent d'utiliser un système *hardware* de rendu temps réel permettant d'obtenir très rapidement l'aspect d'objets complexes pour une attitude donnée. La fonction à minimiser est la distance des points projetés aux contours de l'image. La distance est calculée en mesurant la distance du contour projeté au contour le plus proche dans l'image, dans la direction de la normale au contour. On peut aussi citer les travaux de Marchand *et al.* [3] qui utilise les contrastes d'intensité dans l'image pour recaler le modèle CAO de l'objet suivi.

Suivi par recalage d'un modèle de luminance. L'ensemble des travaux basés sur du *template matching* rentrent dans cette catégorie. Nous pouvons mentionner par exemple les travaux de La Cascia *et al.* [11] sur le suivi de visage. La texture du visage est plaquée sur un cylindre 3D et la pose de ce cylindre est optimisée de manière à ce que la différence entre la texture du visage dans l'image et la texture de la projection du cylindre ait une norme Euclidienne la plus faible possible.

Hager *et al.* [4] ont développé une approche de suivi de texture permettant de suivre des motifs texturés en temps réel. Limitée à des objets planaires et à de petits déplacements, elle a été étendue par Jurie et Dhome au cas de mouvements importants [9] et aux cas de surfaces gauches [7].

Jepson *et al.* [6] proposent une approche qui permet de mettre à jour le motif suivi. Le modèle du motif est, en effet, obtenu par application d'une version *en ligne* de l'algorithme EM. Cet algorithme est utilisé pour modéliser les réponses de filtres orientables multi-échelles, appliqués en chaque point du motif.

Dans le cas de l'algorithme *Mean Shift* [1], l'objet est modélisé par un histogramme couleur. L'algorithme de suivi consiste alors à déplacer une fenêtre d'analyse de telle manière que l'histogramme contenu dans la fenêtre d'analyse coïncide au mieux avec celui du motif.

1.2 Contribution et plan de l'article

A notre connaissance, peu de méthodes de suivi d'objet combinent différents types d'informations, tels que les contours et la texture. La contribution principale de cet article réside justement dans la proposition d'une méthode hybride, capable de mélanger des informations de distances aux contours et des informations de texture, en s'adaptant ainsi au mieux aux différentes caractéristiques visuelles d'un objet.
Cet article comporte trois parties. La section 2 présente les bases de la méthode proposée. Cette méthode repose sur l'utilisation de deux phases ; une phase dite d'apprentissage et une phase de suivi ; elles seront présentées en détail dans la section suivante. Enfin nous présenterons des résultats expérimentaux montrant la validité de l'approche.

2 Une méthode hybride pour le suivi de motif

Comme nous l'avons remarqué précédemment, pour suivre un motif visuel particulier il peut être plus intéressant d'utiliser des distances aux contours, alors que pour d'autres la prise en compte de la texture sera préférable.
Nous proposons ici d'utiliser un critère mixte contour/texture. La méthode que nous proposons dans cet article constitue une extension des travaux de Jurie et Dhome [8].

2.1 Représentation du motif

Nous supposons que le motif planaire à suivre est composé des N points $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N)$, où $\mathbf{p}_i = (x, y)$, réunis dans le vecteur $\mathbf{P}$. Ces coordonnées, exprimées dans un repère lié au motif (différent du repère de l'image), sont supposées constantes.
Ces points sont choisis de manière à satisfaire deux contraintes :
- être le mieux possible répartis sur le motif
- être placés dans des lieux de l'image où le gradient spatial est fort

En pratique, le motif est découpé en baquets, et un nombre minimal de points est choisi dans chaque baquet. Le choix se fait en tirant des points au hasard dans le baquet et en choisissant ceux ayant le gradient le plus fort.
Chacun des points ainsi sélectionnés est ensuite classé comme point de contour ou point de texture, à partir du critère suivant : soient x et y les coordonnées d'un point $\mathbf{p}$ de l'image, et $I(x, y) = I(\mathbf{p})$ l'intensité lumineuse en ce point. Nous définissons M comme :

$$M = \begin{pmatrix} \sum_w \frac{\partial I(\mathbf{p})}{\partial x}^2 & \sum_w \frac{\partial I(\mathbf{p})}{\partial x}\frac{\partial I(p)}{\partial y} \\ \sum_w \frac{\partial I(\mathbf{p})}{\partial x}\frac{\partial I(\mathbf{p})}{\partial y} & \sum_w \frac{\partial I(\mathbf{p})}{\partial y}^2 \end{pmatrix} \quad (1)$$

où w est une fenêtre carrée centrée sur le point. Soient α et β les deux valeurs propres de M. Alors, le point est considéré comme point de contour si $\alpha + \beta$ est grand et

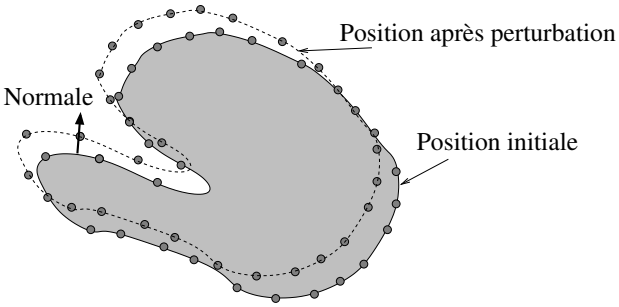

FIG. 1 – Distance du point au contour le plus proche

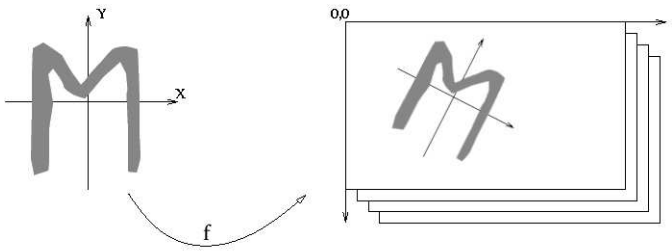

FIG. 2 – Repère image / repère motif

si, simultanément, $\alpha * \beta$ est faible. Sinon, il est considéré comme point de texture. Cela revient à dire que la fonction d'autocorélation de la surface n'est courbée que dans une direction. Ce critère est très proche du critère proposé par Harris et Stephens [5] pour la détection de coins.
A l'issue de cette étape, chaque point $\mathbf{p}_i$ du motif est classé comme point de contour ou point de texture, ce que nous notons $\mathbf{C}(\mathbf{p}_i) = 1$ et $\mathbf{C}(\mathbf{p}_i) = 0$, respectivement.
Le motif est caractérisé par un vecteur de forme $\mathbf{V} = (v_1, \ldots, v_N)$ où les v_i sont définis par :

$$v_i = \mathbf{V}(\mathbf{p}_i) = \begin{cases} I(\mathbf{p}_i, t) & \text{si } \mathbf{C}(v_i) = 0 \\ DI(\mathbf{p}_i, t, nx, ny) & \text{si } \mathbf{C}(v_i) = 1 \end{cases} \quad (2)$$

où $I(\mathbf{p}, t)$ représente le niveau de gris de l'image I au temps t. $DI(\mathbf{p}_i, t, nx, ny)$ représente une distance du point à un contour, mesurée dans l'image I au temps t, dans une direction (nx, ny) perpendiculaire au contour. Cette distance est représentée figure 1.
Nous noterons par la suite $\mathbf{V}$ par $V(\mathbf{P}, t)$ pour montrer le lien existant entre le vecteur de forme et, d'une part, les coordonnées des points du motif, et, d'autre part, le temps. Le vecteur de forme est donc un ensemble de mesures décrivant le motif ; il contient d'une part l'intensité lumineuse de certains points de l'image et d'autre part des distances de points à des contours.

2.2 Représentation du mouvement

Il nous faut maintenant modéliser les déplacements du motif dans l'image.
Soit $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_n(t))$ le vecteur des paramètres représentant la position du motif à l'instant t. Ces paramètres sont les paramètres d'une fonction de changement de repère, amenant un point $\mathbf{p}$ du repère *motif* au point $\mathbf{p}'$ dans l'*image* (figure 2). On considère qu'il existe une fonction f décrivant la position du motif, fonction telle que

$$\mathbf{p}' = f(\mathbf{p}, \mu(t)) \quad (3)$$

Nous n'introduisons pas de restriction sur cette fonction de mouvement. Elle peut représenter des transformations simples tels que des translations planaires, mais aussi des transformations plus complexe comme des homographies, ou des mouvements impliquant des déformations, même si cette dernière possibilité n'a pas été étudiée.
L'expression 3 signifie que le point $\mathbf{p}$ vient en $\mathbf{p}'$ si on lui applique le mouvement de paramètres μ. Nous noterons également :

$$\mathbf{V}(\mu(t), t) = \mathbf{V}(f(\mathbf{P}, \mu(t)))$$

où $f(\mathbf{P}, \mu(t)) = (f(p_1, \mu(t)), \ldots, f(p_N, \mu(t)))$. Nous supposons que $\mathbf{V}$ est dérivable par rapport au temps.
Nous appellerons par la suite *mouvement* de l'objet sa différence de position entre deux images, et nous noterons $\delta\mu(t)$ ce mouvement. Ainsi, nous avons par définition :

$$\delta\mu(t) = \mu(t) - \mu(t-1)$$

Par la suite, nous supposerons que la transformation réalisée est linéaire dans l'espace du plan projectif. Cela nous permettra de représenter des déplacements 2D tels que translations, rotations planaires, affinités, mais aussi des mouvements 3D du plan au moyen d'homographies.
La transformation pourra donc s'exprimer par un produit de matrices, après avoir écrit les coordonnées des points avec des coordonnées homogènes. Nous noterons par la suite cette matrice $\mathbf{F}(\mu(t))$. La relation (3) s'écrira donc

$$\mathbf{p}' = \mathbf{F}(\mu(t)) \times \mathbf{p} \quad (4)$$

où $\mathbf{p}$ et $\mathbf{p}'$ sont écrits en coordonnées homogènes.

2.3 Suivi du motif

En utilisant les définitions précédentes, nous proposons un suivi basé sur la minimisation d'un critère de type SSD. Suivre l'objet revient à donner à chaque instant t sa position $\mu(t)$ telle que $\mu(t)$ minimise le critère :

$$\epsilon(t) = \| \mathbf{V}(\mu_0, t_0) - \mathbf{V}(\mu(t), t) \| \quad (5)$$

où $\mu_0 = \mu(t_0)$ est la position du motif à la première image de la séquence. Nous supposons que μ_0 est connu à la première image de la séquence.
Nous recherchons une méthode itérative, donnant la position $\mu(t)$ en fonction de la position $\mu(t-1)$ et de l'image à l'instant t. Le principe de base consiste à considérer qu'au voisinage d'une certaine position du contour suivi, la fonction permettant de passer des paramètres du mouvement

effectué par le contour à la modification de l'image est linéaire.
En supposant que la position à l'instant $t-1$ soit parfaitement connue, c'est à dire que $\epsilon_{t-1} = 0$,

$$\mathbf{V}(\mu_0, t_0) = \mathbf{V}(\mu(t-1), t-1) \quad (6)$$

$\mathbf{V}$ est une fonction dépendant de t et de μ, nous pouvons écrire son développement limité au premier ordre comme :

$$\begin{aligned} & \mathbf{V}(\mu(t-1) + \delta\mu, t-1+\delta t) = \mathbf{V}(\mu(t), t) \\ = \quad & \mathbf{V}(\mu(t-1), t-1) + (\mu(t) - \mu(t-1))\mathbf{V}_\mu(t) + \delta t \mathbf{V}_t(t), \end{aligned}$$

où $\mathbf{V}_\mu(t)$ représente la dérivée de $\mathbf{V}$ par rapport à μ et $\mathbf{V}_t(t)$ sa dérivée par rapport au temps, ces deux dérivées étant calculées au temps t. Nous supposons que le temps séparant deux images est par définition de 1.
En supposant que

$$\mathbf{V}_t(t) = -\mathbf{V}(\mu(t-1), t-1) + \mathbf{V}(\mu(t-1), t),$$

noté par la suite $\delta\mathbf{V}(t)$, en prenant en compte le fait que le minimum sera obtenu pour $\bigtriangledown\epsilon(t) = 0$ dans (5), et utilisant (6), nous obtenons :

$$\begin{aligned} 0 &= \mathbf{V}(\mu(t-1), t-1) - \mathbf{V}(\mu(t), t) \\ &= (\mu(t) - \mu(t-1))\mathbf{V}_\mu(t) - \delta\mathbf{V}(t) \end{aligned}$$

ce que nous pouvons également mettre sous la forme :

$$\mu(t) = \mu(t-1) + \mathbf{V}_\mu^{-1}(t)\delta\mathbf{V}(t)$$

ou encore :

$$\delta\mu(t) = \mathbf{V}_\mu^{-1}(t)\delta\mathbf{V}(t)$$

où $\delta\mu(t)$ représente le mouvement de l'objet à l'instant t, et $\mathbf{V}_\mu^{-1}(t)$ la matrice Jacobienne inverse. Cette relation nous permet donc de déduire le mouvement de l'objet à partir d'une variation du vecteur de forme. La matrice Jacobienne $\mathbf{V}_\mu(t)$ donne les variations des paramètres du vecteur de forme en fonction d'un mouvement $\delta\mu$ de l'objet. Il s'agit d'une matrice de taille $N \times n$ où n est le nombre de paramètres décrivant le mouvement, et N la dimension du vecteur de forme. La matrice Jacobienne n'est donc pas inversible. Nous avons montré, et c'est une contribution importante de notre approche, qu'une utilisation de la pseudo-inverse $\mathbf{V}_\mu^\dagger(t)$, au moyen de l'équation :

$$\delta\mu(t) = \mathbf{V}_\mu^\dagger(t)\delta\mathbf{V}(t), \quad (7)$$

donne des résultats bien moins intéressants que de calculer $\mathbf{H}(t)$ telle que :

$$\delta\mu(t) = \mathbf{H}(t)\delta\mathbf{V}(t), \quad (8)$$

par apprentissage direct.
La méthode de suivi peut être résumée de la façon suivante : nous commençons par une phase d'*apprentissage* durant laquelle nous déterminons une matrice H pour de petits déplacements du contour autour de la position initiale. Si le contour a déjà été appris, nous pouvons passer directement à la deuxième phase, le *suivi* en lui-même, où nous calculons la différence entre deux images de la séquence pour obtenir $\delta\mathbf{V}(t)$ puis nous en déduisons $\delta\mu(t)$ par l'équation (8).
On peut considérer que cette équation correspond à la modélisation de points dans un espace de dimension N par n hyperplans dont les coefficients $h_{i1}\ldots h_{iN}$ peuvent s'écrire

$$\left\{ \begin{array}{rcl} 0 &=& (h_{11}, \ldots, h_{1N}, -1)(\delta i_1, \ldots, \delta i_N, \delta\mu_1)^T \\ 0 &=& (h_{i1}, \ldots, h_{iN}, -1)(\delta i_1, \ldots, \delta i_N, \delta\mu_i)^T \\ 0 &=& (h_{1n}, \ldots, h_{nN}, -1)(\delta i_n, \ldots, \delta i_N, \delta\mu_n)^T \end{array} \right.$$

Bien évidemment il serait totalement inefficace de recalculer à chaque image la matrice $\mathbf{H}$(t). En effet, le calcul d'optimisation impliqué dans cette phase est coûteux en temps de calcul.
La matrice $\mathbf{H}$ est calculée une seule fois, dans le repère du motif, et nous effectuons des changements de repère de façon à toujours nous trouver au voisinage de la position initiale. Nous développons ce point par la suite.

3 Les phases d'apprentissage et de suivi

Nous venons de voir que l'algorithme proposé repose sur deux phases : une phase d'apprentissage et une phase de suivi. Nous allons détailler chacune de ces deux phases et les modifications apportées par la prise en compte des contours.
Mais auparavant, nous allons expliquer comment la mesure du vecteur de forme est réalisée.

3.1 Une mesure relative du vecteur de forme

Nous avons en effet signalé que, pour éviter de calculer la matrice $\mathbf{H}$ pour chaque nouvelle position du motif dans l'image, nous proposons de la calculer dans le repère du motif.
A l'issue de l'étape de modélisation du motif, chaque point $\mathbf{p}_i$ est classé comme point de contour ou point de texture ($\mathbf{C}(\mathbf{p}_i) = 1$ ou $\mathbf{C}(\mathbf{p}_i) = 0$, respectivement).
Le motif est caractérisé par un vecteur de forme $\mathbf{V} = (v_1, \ldots, v_N)$ où les v_i sont définis par l'équation (2).
La direction de la normale au contour est mémorisée à l'aide d'un point $\mathbf{n} = (n_x, n_y)^t$: la normale en $\mathbf{p}$ est définie comme la droite passant par $\mathbf{p}$ et $\mathbf{n}$. La transformation f étant représentée par la matrice $\mathbf{F}$ (comme expliqué section 2.2), la direction de la normale est donnée par les points $\mathbf{p}' = \mathbf{F} \times \mathbf{p}$ et $\mathbf{n}' = \mathbf{F} \times \mathbf{n}$.
L'étendue de cette zone d'exploration est déterminée par l'étendue de la zone d'apprentissage. Par exemple, si l'on choisi de perturber le motif jusqu'à 20% de sa taille, la zone d'exploration sera déterminée de telle façon qu'un déplacement de 20% puisse être retrouvé. Dans les expériences exposées dans cet article, la zone d'exploration est indexée sur la longueur de la "diagonale" du quadrilatère

suivi. Pour éviter des problèmes d'échelle, on ajoute un terme qui prend en compte le déplacement du motif. Ainsi, si l'on appelle l_{zone} la longueur de cette zone et $d(\mathbf{p}', \mathbf{n}')$ la longueur de la normale au moment considéré, on a

$$l_{zone} = d(\mathbf{p}', \mathbf{n}') \times l_{apprentissage} \times l_{diagonale}$$

Nous mesurons alors la distance d au contour le plus proche, dans l'image, et obtenons directement la distance dans le repère du modèle, en remarquant que la distance D dans le repère modèle est $\frac{d}{\|\mathbf{n}'\|}$.
Ainsi, avec une quantité de calcul très minime, nous obtenons le vecteur de forme que nous obtiendrions si nous appliquions la transformation f^{-1} à l'image.

3.2 Apprentissage

Comme nous l'avons dit ci-dessus, il faut éviter à tout prix de devoir recalculer $H(t)$ à chaque nouvelle image. Il est possible de supposer que $\mathbf{H}$ est constante en établissant la relation (8) dans le repère lié au motif et non pas dans le repère lié à l'image. Ainsi, dans la phase de suivi, il suffira de ramener l'image dans le repère du motif pour pouvoir utiliser la relation (8). C'est pour cette raison que nous avons dû supposer que f était inversible.
L'estimation de la matrice $\mathbf{H}$ se fera donc dans un voisinage de la fonction *identité*. Pour ceci, effectuons N_{Pert} variations de position aléatoires $\delta\mu^i$ et obtenons les N_{Pert} valeurs de $\delta\mathbf{V}^i$ correspondantes. Les mouvements $\delta\mu^i$ sont stockés dans une matrice $\mathbf{DMU}_{N_{Pert} \times n}$ et les vecteurs de forme dans une matrice $\mathbf{DI}_{N_{Pert} \times N_{Pt}}$.
Il suffit ensuite de calculer la "meilleure" matrice $\mathbf{H}$, c'est-à-dire la matrice pour laquelle la somme

$$\sum_{i=1}^{i \leq N_{Pert}} (\delta\mu^i - \mathbf{H} \times \delta V^i)^2$$

est la plus faible possible. Cela revient à minimiser un système au sens des moindres carré, ce qui peut être fait au moyen de la relation suivante :

$$\mathbf{H} = \mathbf{DMU}.\mathbf{DI}^{\dagger}$$

où $\mathbf{DI}^{\dagger}$ représente la pseudo-inverse de $\mathbf{DI}$.
$\mathbf{H}$, résultat de notre apprentissage, est une matrice qui va nous permettre de retrouver le mouvement correspondant à une différence de vecteur de forme (différence de texture et distance au contour) trouvée.
Les données sont obtenues lors de l'apprentissage de la même façon qu'elles le seront lors du suivi. On suppose ainsi que si l'algorithme effectue une erreur en localisant un point pendant l'une des phases, il reproduira cette erreur lors de la seconde. Cela permet de minorer les problèmes dus à de mauvaises mises en correspondances des points de contour.

3.3 Suivi

Une fois la matrice $\mathbf{H}$ connue, nous pouvons passer à la phase de suivi proprement dite.

Lors du suivi, nous n'utilisons pas d'étape de prédiction de mouvement. Il est supposé que dans l'image suivante la position du motif sera la même qu'à l'image actuelle. Pour que le programme de suivi fonctionne correctement, il faut donc que le déplacement du contour entre deux images soit assez faible pour que la "prédiction" soit correcte. En pratique, le domaine de convergence est suffisamment large pour que le motif soit suivi malgré des amplitudes de déplacement importantes.
Nous construisons le vecteur de forme en mesurant dans l'image, pour chaque point du motif, le niveau de gris du point ou la distance au contour (en fonction du type de point). Nous stockons la différence entre le vecteur ainsi formé et le vecteur de forme obtenu pour un mouvement nul dans un vecteur δV. La construction de ce vecteur de forme se fait dans le repère lié au motif, comme expliqué section 3.1.
Après une multiplication de ce vecteur avec la matrice $\mathbf{H}$, nous obtenons un vecteur $\delta\mu$ correspondant à l'estimation des paramètres de mouvement, dans le repère du motif. Toutefois cette matrice ne permet de connaître le déplacement que pour un objet se situant au voisinage de sa position initiale. Pour pouvoir continuer d'appliquer l'algorithme même après un changement de position du motif, Nous devons ramener le mouvement dans le repère de l'image. Nous obtenons ainsi le mouvement

$$f(\mu(t)) = f(\mu(t-1)) \circ f(\delta\mu(t))$$

Dans le cas où le déplacement f est modélisé par une transformation linéaire dans le plan projectif, $f(\mu(t))$ est représenté par une matrice $\mathbf{F}(\mu(t))$, dont l'estimation est :

$$\mathbf{F}(\mu(t)) = \mathbf{F}(\mu(t-1)) \circ \mathbf{F}(\delta\mu(t))$$

Notre algorithme de suivi peut s'écrire en pseudo-langage de la façon suivante, en appelant mu le déplacement global jusqu'à l'image n et pt les points du contour sur l'image initiale.

```
## changement de référentiel
# pour les points
PTI = F(MU)* PT
# pour les normales
NORMI = F(MU)*NORM

# recherche des nouvelles positions
pour I de 1 à N_PT faire
   si point de contour
      DNI = NORMI - PTI
      GRADMAX = 0
      pour K le long de la normale
         P1 = PT(I) + (K-1) * DNI
         P2 = PT(I) + (K+1) * DNI
         GRAD = Image(P2) - Image(P1)
         si GRAD>GRADMAX
            DI(I) = K
            GRADMAX = GRAD
```

```
         finsi
      finpour
   sinon si point texture
      DI(I) = Image(PT(I))
   finsi
finpour

# calcul du déplacement
DMU = H * DI

# calcul du déplacement global
F(MU) = F(MU) * F(DMU)
```

On constate que la complexité algorithmique du calcul du vecteur de mesure est plus élevée pour les points de contour que pour les points de texture. Si le nombre de pas dans la zone d'exploration est élevé, le temps de traitement pour chaque image sera d'autant plus important.
Il ne faut pas oublier de prendre en compte l'affichage et les temps d'accès à la caméra et à la mémoire, qui ont une grande importance dans les temps d'exécution. Toutefois cela reste relativement raisonnable, et l'exécution de l'algorithme s'effectue bien en temps réel vidéo (25 images/s).

4 Validation expérimentale

Nous avons implanté l'algorithme proposé sur un ordinateur de type PC, équipé d'une caméra numérique SONY délivrant des images en temps réel vidéo, au moyen d'une communication FireWire (IEEE 1394).
Les motifs sont représentés à l'aide d'une centaine de points. La transformation utilisée est une homographie du plan projectif. La figure 3 présente des images issues du traitement de séquences en temps réel.
Pour démontrer l'apport de la composante de contour, nous avons utilisé l'algorithme de suivi hybride sur une séquence générée artificiellement. Cette séquence contient un motif non texturé se déplaçant rapidement. Une image extraite de cette séquence est visible à la figure 4.
Nous avons suivi le motif de cette séquence en n'utilisant que des points de texture, puis uniquement des points de contours, et enfin avec notre méthode hybride. La méthode hybride a sélectionné 80 points de contours et 20 points de textures. Cette prédominance des points de contour sur les points de texture est due au fait que le motif n'est pas "texturé".
La figure 5 montre l'estimation de la position de l'un des coins de la zone suivie pour chacun des algorithmes, ainsi que la position réelle.
Nous constatons que l'algorithme de suivi n'utilisant que des points de texture décroche au bout de 28 itérations, tandis que le suivi à base de points de contour suit le motif jusqu'à la fin de la séquence. Nous constatons aussi que la précision de l'algorithme hybride n'est pas subpixellique, contrairement à l'algorithme classique, du fait de la plus faible précision de la méthode de suivi de contour employée.

Un autre point important est le temps d'exécution. Etant donné que la complexité algorithmique du suivi avec des points de contour est plus élevée, le temps d'exécution est plus important si l'algorithme utilise de nombreux points de contour.
Dans le tableau suivant qui montre, dans des conditions similaires, le temps de traitement moyen d'une frame suivant l'algorithme utilisé, nous constatons bien une baisse des performances avec l'utilisation de points de contour.

	contour	texture	hybride
temps d'exécution	16.18 ms	9.53 ms	14.14 ms

Nous avons aussi testé la taille du domaine de convergence du suivi pour un motif donné. Pour cela, nous avons fait apprendre le même motif à chacun des algorithmes et nous avons testé l'étendue des variations permises avant que l'algorithme ne perde le motif.
Les zones en noir de la figure 5 montre les translations pour lesquelles l'algorithme converge vers la bonne solution.
Le suivi des points de texture est moyennement efficace dans cette configuration (graphique de gauche), tandis qu'avec les points de contours uniquement le résultat est plus mauvais (graphique du milieu). C'est avec la méthode hybride que la zone de convergence est la plus importante.
La zone de convergence obtenue avec des points de contours présente des discontinuités (les "blancs"), très visibles dans le cas des contours seuls. On peut voir, de plus, que le motif n'est pas suivi de manière homogène : la zone de convergence n'est pas symétrique en x et y. C'est dû au fait que le motif soit quelconque et présente des problèmes de régularité : l'algorithme confond certaines dispositions du motif.
L'algorithme de suivi a choisi 45 points de texture et 55 points de contour pour ce motif. On peut constater sur ces graphiques l'apport de l'utilisation des deux méthodes : le suivi de texture donne une zone de convergence restreinte mais sans discontinuités, tandis que le suivi de contour donne une zone de convergence plus étendue mais aussi plus irrégulière.
Nous pouvons donc dire que si nous avons réussi à étendre le domaine de convergence de notre algorithme à des motifs non texturés, ceci se fait au prix d'une augmentation du temps d'exécution et d'une légère baisse de la précision.

5 Conclusion

Dans cet article, nous avons présenté une nouvelle méthode de suivi de motifs hybride texture / contours.
L'idée consiste à représenter le motif au moyen d'un nombre limité de points. Les points sont choisis de manière à être répartis sur le motif et à se trouver dans des zones où la variation de la luminance est importante. Ces points sont ensuite classés comme point de contour ou point de texture. Un vecteur de forme contenant l'intensité du motif (pour les point de texture) et la distance au contour le plus proche (pour les points de contours) est ensuite calculé.
Lorsque le motif se déplace, une différence dans le vecteur de forme apparaît, et c'est cette différence qui est utilisée

FIG. 3 – Exemples de suivi de motifs texturés

FIG. 4 – Séquence de test générée atificiellement (images 5 et 15)

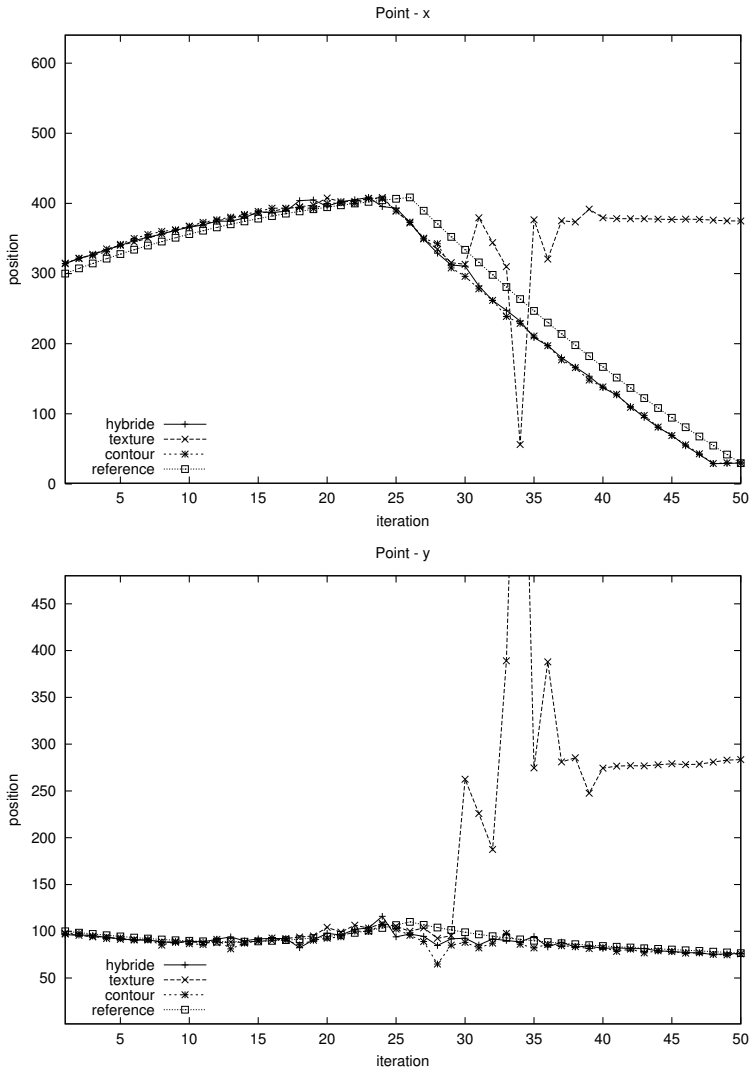

FIG. 5 – Estimation des coordonnées du coin supérieur droit du motif par les différents algorithmes

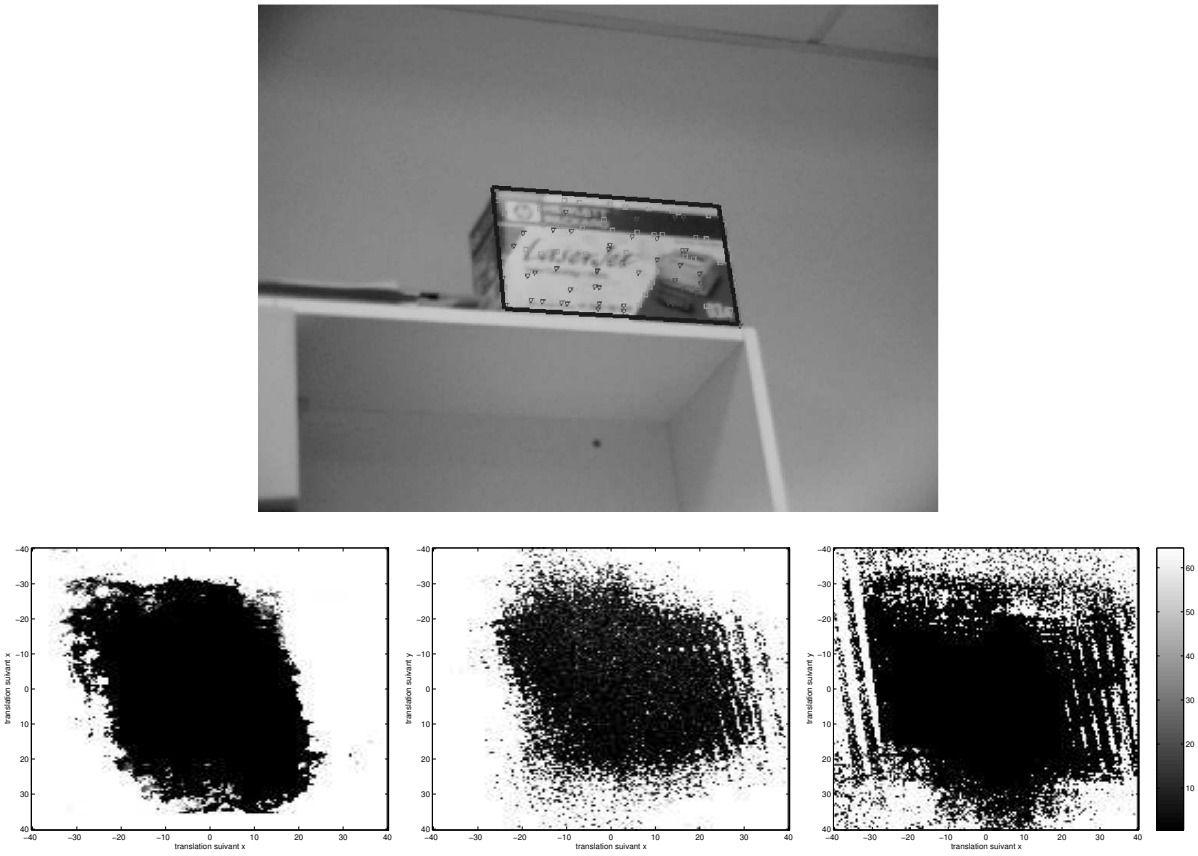

FIG. 6 – Zones de convergence pour des translations suivant x et y. (haut) objet suivi, (gauche) uniquement des textures, (milieu) uniquement des points de contour, (droite) méthode hybride

pour déduire le déplacement de l'objet. Cet algorithme permet de suivre un motif rigide en l'absence d'occultations. Cette méthode, basée sur un algorithme connu pour son efficacité et sa robustesse, permet d'élargir le domaine d'utilisation de cet algorithme à des motif pas ou peu texturés. Cette extension se fait au prix d'une augmentation du temps d'exécution.

Références

[1] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR00*, pages II : 142–149, 2000.

[2] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. *IVC*, 20(5-6) :427–433, March 2002.

[3] É. Marchand et al. A 2D–3D model-based approach to real-time visual tracking. In *Image and Vision Computing*, pages 19(13) :941–955, novembre 2001.

[4] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(10) :1025–1039, October 1998.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, Mancherster, 1988.

[6] A. D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust on-line appearance models for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages pp. 415–422, 2001.

[7] F. Jurie and M. Dhome. Real time 3d template matching. In *Computer Vision and Pattern Recongition*, pages (I)791–797, Hawai, December 2001.

[8] F. Jurie and M. Dhome. Real time template matching. In *Proc. IEEE International Conference on Computer vision*, pages 544–549, Vancouver, Canada, July 2001.

[9] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7) :996–1000, 2002.

[10] H. Kollnig and H.H. Nagel. 3d pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23(3) :283–302, 1997.

[11] M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination : An approach based on registration of textured-mapped 3d models. *PAMI*, 22(4) :322–336, April 2000.

[12] D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2) :113–122, 1992.