Segmentation et partition de séquences aériennes pour le diagnostic du trafic routier et autoroutier

Segmentation and partition of aerial scenes for road traffic diagnosis

K. Kaâniche[1] P. Vasseur[1]

[1] Centre de Robotique et d'Electrotechnique et d'Automatique - EA3299
Université de Picardie Jules Verne

CREA, 7 rue du moulin neuf 80000 Amiens
khaled.kaaniche@crea.u-picardie.fr

Résumé

Nous proposons une approche originale qui a pour but le traitement de séquences aériennes présentant un trafic routier. Nous considérons les séquences vidéo comme étant une succession d'images fixes. Le système proposé consiste, premièrement, en la détection de contours de ces images, ensuite, à créer un graphe dans lequel les primitives issues de la segmentation sont connectées entre elles via des critères perceptifs et de mouvement. Enfin, la bipartition du graphe par la technique des coupes normalisées est réalisée dans le but d'extraire les éléments importants de la scène, c'est à dire les véhicules. Les paramètres de l'algorithme proposé sont choisis grâce à une étape d'apprentissage pour laquelle nous utilisons les algorithmes génétiques.

Mots Clef

Analyse de scène, traitement d'image, apprentissage, algorithmes génétiques.

Abstract

We propose an original approach that aims to treat aerial scenes which present a road traffic. We consider the scene as a succession of fixed images. Our system consists essentially of three parts : the segmentation of the acquired images, the modelization of a graph in which primitives provided by the segmentation are connecting according to perceptual organization and motion criteria, and the bi-partition of the graph by normalized cuts technique. The final aim consists of extracting vehicles from the background. Parameters of the proposed algorithm will be chosen after a learning stage in which we maximize the similarity between manual cut and normalized cut results. Genetic Algorithm are used for this phase of optimization.

Keywords

Scene analysis, image processing, learning, genetic algorithms.

1 Introduction

L'analyse des séquences vidéo est l'un des sujets de recherche les plus tentants. Peut être que ceci découle de la richesse en informations que présente une scène ou encore de la variété des applications (médicales, sécuritaires, militaires, ...etc.). Les spécialistes du domaine ont tenté, et tentent encore d'imiter via le système caméra/calculateur le comportement humain en terme de vision. Pour y parvenir différents axes ont été développés. Ainsi apparaît le terme de " groupement spatio-temporel " [1]. C'est le terme qui englobe à la fois l'identification des objets et la détection du mouvement. Nous nous en sommes inspirés pour proposer dans cet article une approche basée, d'une part sur le traitement statique, image par image, de la séquence et d'autre part sur l'observation du mouvement effectué par les primitives au cours du temps. Notre but est de créer en premier lieu un graphe. Les noeuds de ce graphe sont les primitives issues de la détection de contours d'une image. Ces noeuds se connectent entre eux par des liens basés sur des critères d'organisation perceptive, en l'occurrence géométriques, ainsi que par des liens qui tiennent compte du mouvement effectué par chaque noeud et sa configuration dans l'image suivante. Ce graphe est donc considéré comme la racine de tout le processus de groupement. C'est sur ce graphe que la partition s'exécutera par la suite. Nous avons choisi la technique des coupes normalisées [2] pour cette phase de partition. Shi et Malik considèrent le groupement visuel comme étant un problème de partition de graphe où les noeuds sont tout simplement les pixels et où les connexions correspondent aux degrés de similarité en intensité ou en couleur de ces pixels. Dans notre application, ce sont les primitives issues de la détection des

contours qui sont utilisées comme noeuds. Il est clair que le nombre de ces primitives est beaucoup moins important que le nombre de pixels présents dans l'image originale. Nous espérons par ce choix gagner énormément en terme de temps de calcul. Cependant, ce choix peut entraîner une perte d'informations car une image contient plus d'informations que sa dérivée binaire. Nous justifions cette prise de risque par le fait que le processus de groupement proposé compte essentiellement sur l'aspect géométrique qui se manifeste davantage dans les contours. L'application qui se cache derrière ces propositions est des plus complexes. Extraire les routes et les véhicules à partir de séquences aériennes acquises à partir d'un hélicoptère (comportement dynamique de l'outil d'acquisition), n'est pas une tache aisée. En effet, la littérature ne semble pas être généreuse pour ce genre d'applications. Les travaux déjà existants s'intéressent surtout à des images statiques. Burlina, Parameswaran et Chellappa [3], Liu et Haralick [4] et Moon, Chellappa et Rosenfeld [5] considèrent les véhicules comme des objets 2D ; un véhicule est modélisé par un rectangle. Une fois le modèle établi, on commence à le rechercher dans l'image binaire (celle résultante de la segmentation). Zhao et Nevatia [6] ont tenté d'améliorer ce processus en formulant le problème en 3D ; ainsi l'erreur sur la décision finale est plus petite car le modèle est plus riche en informations. Nous remarquons que ces méthodes partent avec un modèle et le cherchent sur l'image alors que nous ne définissons aucun modèle mais nous laissons les critères géométriques et de mouvement décider de la façon de grouper les entités de la séquence. L'analyse du trafic routier a été l'objet de plusieurs projets [7][8][9]. La différence entre ces travaux et notre application se situe dans le comportement dynamique de l'outil d'acquisition que nous utilisons. Les séquences analysées dans ces projets sont acquises à partir d'un capteur statique : une soustraction entre les images acquises et une image référence (sans véhicule et systématiquement mise à jour) est suffisante pour extraire les entités en mouvement (les véhicules dans ce cas). Cohen et Medioni [10] se sont intéressés à la détection d'objets en mouvement à partir de séquences aériennes prises par des capteurs dynamiques. Ils proposent d'estimer puis de compenser le mouvement de la caméra. Les éléments mobiles sont ensuite détectés en exploitant les différences d'images déplacées. Ceci suppose que tout véhicule présent dans la séquence doit être en mouvement pour qu'il soit détecté. Nous abordons le problème différemment en nous intéressant à l'aspect perception.

Ce papier est organisé de la façon suivante. Dans la section 2 nous présentons la théorie de l'organisation perceptive et sa contribution dans ce travail. Les coupes normalisées sont développées dans la section 3. Dans la section 4 nous détaillons notre approche. La phase d'apprentissage, les fondements d'une telle étape ainsi qu'une étude approfondie sur les algorithmes génétiques, sont les sujets abordés dans la section 5. Dans la section 6 nous présentons quelques résultats expérimentaux.

2 L'organisation perceptive

Sarkar et Boyer[1] définissent l'organisation perceptive par la capacité d'un système de vision à organiser les primitives détectées dans une image tout en se basant sur des critères gestaltistes par exemple. Les psychologues avaient offert un ensemble de lois qui semblent être déterminantes dans le système humain de perception. Parmi ces lois nous trouvons le parallélisme, la continuité, la proximité, la similarité, la symétrie ...etc [11]. C'est ce type de loi que nous allons utiliser pour établir les connexions entre différents noeuds (primitives) de notre graphe. Nous traduisons ceci par un simple processus basé sur le calcul d'un score (ou probabilité) entre primitives pour une loi donnée. Par exemple, si nous choisissons la proximité comme un critère de groupement, le processus calcule un score qui tend vers *1* si les primitives se touchent et qui s'annule dès que la distance entre deux primitives dépasse une certaine valeur prédéfinie. Ainsi pour favoriser un groupement de deux noeuds selon un critère, le score doit être élevé. Cependant, l'utilisation en même temps de plusieurs lois ou critères semble être une piste intéressante surtout pour des applications complexes ; il est évident qu'une loi ne peut à elle seule dégager plusieurs types de formes ou d'entités. Il est donc nécessaire de combiner plusieurs scores. Ceci peut être assuré par des opérateurs logiques et des pondérations selon les applications et les tests expérimentaux. Enfin le choix des lois ou critères peut être lui aussi justifié par la nature de l'application. Dans le cadre d'un trafic routier, où nous cherchons des routes et des véhicules, le parallélisme et la proximité sont des candidats potentiels.

3 Les coupes normalisées

Etant l'une des nouvelles techniques conçues pour la partition des graphes, la méthode des coupes normalisées [2] est basée sur la minimisation d'un critère qui traduit la similarité des connexions entre noeuds de différentes parties. Nous pouvons encore citer la technique des coupes minimales[12] ainsi que la technique des coupes moyennes[11]. Ces processus de partition de graphes ont fait l'objet d'une étude comparative par Soundararajan et Sarkar [13]. Les coupes minimales ont montré des limites. Wu et Leahy [12] ainsi que Shi et Malik [2] confirment ceci en constatant que cette technique favorise le groupement des petits ensembles de noeuds isolés dans le graphe. Les deux autres techniques montrent des performances presque similaires en terme de résultats corrects. D'une manière explicite, le problème de la partition d'un graphe est formulé par la technique des coupes normalisées comme suit :

Etant donné un graphe *G(V,E)*, nous cherchons à trouver *A* et *B* / $A \bigcup B = V$ et $A \bigcap B = \emptyset$ et qui minimise :

$$Ncut(A,B) = \frac{cut(A,B)}{Assoc(A,V)} + \frac{cut(A,B)}{Assoc(B,V)} \qquad (1)$$

Avec :

cut(A,B) = $\sum$ *w(u,v)* ; $u \in A$ et $v \in B$

Assoc(A,V) = $\sum$ *w(u,v)* ; $u \in A$ et $v \in V$

w(u,v) est le poids de la connexion qui relie le noeud *u* au noeud *v*. Le vecteur propre correspondant à la deuxième plus petite valeur propre du système (2) traduit par l'équation suivante :

$$Wx = \lambda Dx \quad (2)$$

divise d'une manière presque optimale le graphe *G(V,E)* en deux parties [2] avec :

$$W(i,j) = w_{ij} \qquad et \qquad D(i,i) = \sum w_{ij}, j \in V \quad (3)$$

w_{ij} est le score calculé par le processus d'organisation perceptive pour les deux noeuds (primitives) *i* et *j*. Quand nous utilisons les primitives comme noeuds, la dimension de la matrice *W* est petite et par conséquent la résolution du système (2) est quasi-immédiate. Ce fait est d'une importance majeure pour notre application car il est question de traiter des séquences vidéo et par conséquent, de traiter une succession rapide d'un grand nombre d'images.

4 L'approche

Notre algorithme est destiné à être appliqué sur des séquences aériennes qui présentent un trafic routier. La séquence acquise à partir d'hélicoptère est découpée en images fixes successives. Le schéma du système proposé dans la figure 1 (FIG. 1.), décrit les différentes étapes de l'algorithme.

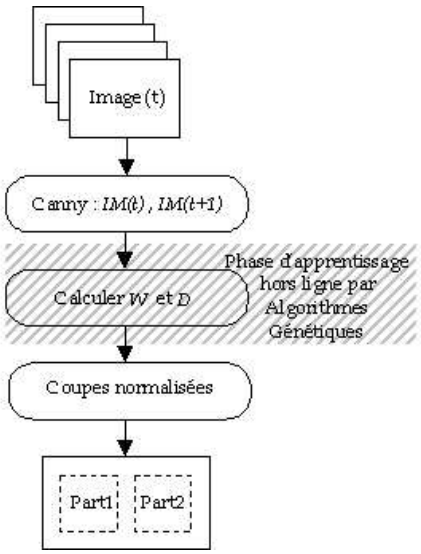

FIG. 1 – **Description de l'algorithme.**

Tout d'abord le système commence à extraire les contours des images successives grâce à la technique de Canny[14](FIG. 2.). L'analyse se concentre dès lors sur les images binaires résultantes. Chaque primitive est décrite

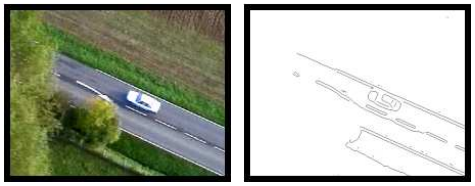

FIG. 2 – **Exemple de détection de contours.**

par un nombre de caractéristiques telles que l'orientation, les coordonnées des extrémités ...etc.
Nous créons par la suite le graphe pour une image donnée. Les primitives présentes sur l'image binaire sont considérées comme les noeuds de ce graphe. Les connexions entre noeuds sont stockées dans la matrice *W* appelée matrice des liens. Un élément w_{ij} (*i* et *j* étant deux noeuds du graphe) de la matrice *W* peut s'écrire comme le produit des scores des critères de liaisons. Dans notre cas :

$$w_{ij} = Spar_{ij} \times Sprox_{ij} \times Smot_{ij} \quad (4)$$

- $Spar_{ij}$ décrit le degré de parallélisme entre *i* et *j*.
- $Sprox_{ij}$ décrit le degré de proximité entre *i* et *j*.
- $Smot_{ij}$ décrit la similarité de mouvement entre *i* et *j*.

4.1 Calcul de $Spar_{ij}$

Un segment, appelé aussi une primitive, est un ensemble de pixels qui forment un contour, il n'est pas forcément linéaire. Pour pouvoir établir un lien de parallélisme entre deux segments, il faut tout d'abord calculer leurs orientations. L'idée est d'estimer les paramètres *a* et *b* de la droite Δ : *y = a x + b*. Cette estimation, effectuée par la méthode des moindres carrés, a pour but de minimiser la distance entre Δ et l'ensemble des pixels formant le segment. Soient *X* le vecteur contenant les abscisses des pixels d'un segment (ou d'une primitive) et *Y* le vecteur contenant les ordonnées de ces mêmes pixels : *a* et *b* sont les résultats du système suivant :

$$[a \quad b]^T = ([X \quad 1]^T[X \quad 1])^{-1}[X \quad 1]^T Y \quad (5)$$

Ainsi, pour chaque primitive, *a* indique son orientation. Pour établir le lien de parallélisme entre deux primitives il suffit de comparer leurs orientations respectives :

$$Spar_{ij} = \begin{cases} 1 - \frac{|a_i - a_j|}{seuilpar} & si \quad |a_i - a_j| < seuilpar \\ 0 & ailleurs \end{cases} \quad (6)$$

a_i et a_j sont les orientations respectives des noeuds *i* et *j*. *seuilpar* est une constante prédéfinie.

4.2 Calcul de $Sprox_{ij}$

Si la distance entre deux noeuds dépasse un certain seuil prédéfini noté *seuilprox*, le lien de proximité entre ces noeuds noté $Sprox_{ij}$ doit être nul. La figure 3 (FIG. 3.) montre comment cette distance est calculée. En fait, nous

choisissons, pour un segment *i*, trois points caractéristiques (dans ce cas les extrémités et le point milieu) et nous calculons toutes les distances possibles entre ces points et leurs correspondants dans le segment *j* : d_1,...,d_9. Nous gardons la plus petite de ces distances qui désigne ainsi la distance entre les deux primitives. Pour un résultat exact, le calcul doit s'effectuer sur tous les points des deux segments, mais ceci demande un important temps de calcul. L'approximation proposée est suffisante pour notre application et fournit des résultats satisfaisants.

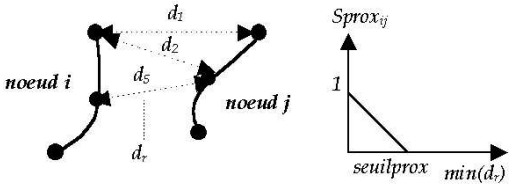

FIG. 3 – **Calcul de** $Sprox_{ij}$**.**

$Sprox_{ij}$ est calculé de la façon suivante :

$$Sprox_{ij} = \begin{cases} 1 - \frac{min(d_r)}{seuilprox} & si \quad min(d_r) < seuilprox \\ 0 & ailleurs \end{cases} \tag{7}$$

4.3 Calcul de $Smot_{ij}$

Pour le calcul du lien de mouvement entre deux noeuds du graphe représentant l'image binaire *IM(t)*, nous nous intéressons à la fois à ce graphe et au graphe qui représente l'image binaire suivante *IM(t+1)*. A chaque noeud *i(t)*, on fait correspondre (s'il existe) un noeud *i(t+1)* qui doit théoriquement être l'évolution de *i(t)* au cours du temps. Notre seul connaissance étant les segments des deux images, les critères de mise en correspondance sont plus restreints et essentiellement géométriques. Nous partons de l'hypothèse suivante : le mouvement effectué d'une image *IM(t)* à l'image suivante *IM(t+1)* est faible. Ainsi, il est inutile de rechercher l'élément correspondant dans toute l'image *IM(t+1)*, il suffit d'établir une fenêtre *f* et de limiter la recherche à l'intérieur de son périmètre. Si la dimension de *f* est bien choisie et que l'hypothèse de base est vérifiée, la possibilité de trouver le bon segment *i(t+1)* est forte. Le choix entre plusieurs candidats à cette correspondance est effectué par comparaison de leurs tailles par rapport à celle de *i(t)*. Il est possible qu'un segment *i(t)* ne trouve pas de correspondant, dans ce cas il est éliminé du graphe. Le centre de la fenêtre *f* est celui du segment *i(t)* ; la fenêtre est ensuite calquée sur l'image *IM(t+1)* pour entamer la recherche. Dans la figure 4 (FIG. 4.) nous montrons comment calculer le degré de similarité de mouvement entre deux noeuds *i(t)* et *j(t)* appartenant à l'image *IM(t)*. *seuilmot* est une constante prédéfinie. Dans l'exemple présenté (FIG. 4.), nous pouvons constater que les deux mouvements sont simples (uni-dimensionnels). Dans le cas général(mouvements bi-dimensionnels), chaque mouvement est décomposé selon les deux axes de l'image en deux composantes ; la comparaison s'effectue deux à deux entre composantes de mouvements différents.

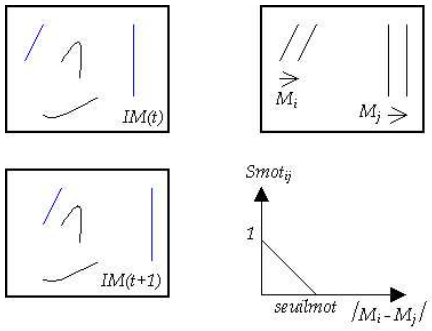

FIG. 4 – **Calcul de** $Smot_{ij}$**.**

$Smot_{ij}$ s'écrit de la façon suivante :

$$Smot_{ij} = \begin{cases} 1 - \frac{|M_i - M_j|}{seuilmot} & si \quad |M_i - M_j| < seuilmot \\ 0 & ailleurs \end{cases} \tag{8}$$

Une fois la matrice *W* calculée, nous appliquons la technique des coupes normalisées pour partager le graphe en deux parties en calculant le deuxième plus petit vecteur propre du système (2). Le but de cette séparation en deux, est d'extraire les véhicules présents dans la séquence d'une part et la route d'autre part. Si le fait que nous traitions des scènes routières peut expliquer le choix du critère de parallélisme, la proximité a pour objectif de favoriser le groupement des contours détectés par la présence d'un véhicule. Ces contours sont, en effet, extrêmement proches. Enfin la comparaison des différents mouvements peut dans certains cas favoriser le groupement de segments de véhicules qui bougent en général d'une façon similaire quand il s'agit d'une même voie.

5 Phase d'apprentissage

L'approche proposée comme elle se présente sur la figure 1 (FIG. 1.) est en boucle ouverte. Comment s'assurer donc que la partition finale est correcte ? c'est à dire que l'algorithme a réussi à extraire les véhicules présents dans la séquence. Les essais effectués montrent une sensibilité des résultats finaux au choix des seuils. Ces seuils sont d'ailleurs le seul moyen d'agir sur les résultats. Les critères ainsi que les techniques de calcul des scores ont été déjà fixés et argumentés dans les sections précédentes. Le but est donc de bien choisir les seuils pour que la partition effectuée par l'algorithme soit identique ou presque à une partition référence ; une partition référence qui va faire appel à la vision humaine. En effet un observateur désigne manuellement les véhicules à partir d'une paire d'images

successives binaires. Un algorithme d'optimisation aura pour objectif de minimiser l'erreur entre la bi-partition manuelle et celle de l'algorithme proposé. Nous sommes donc devant un problème d'optimisation où les variables à déterminer sont, dans notre cas, les trois seuils (*seuilpar*, *seuilprox* et *seuilmot*) des trois critères d'organisation choisis. Ce processus d'optimisation est appliqué sur plusieurs images pour en déduire à la fin un ordre de grandeur des seuils qui puisse justifier les valeurs utilisées d'une part et qui assure une bipartition adéquate en boucle ouverte d'autre part : C'est la phase d'apprentissage. Les seuils recueillis grâce à cet apprentissage seront utilisés pour des applications en boucle ouverte qui présentent le même type d'images (luminosité, altitude,... etc.) que lors de la phase d'optimisation schématisée dans la figure 5 (FIG. 5.).

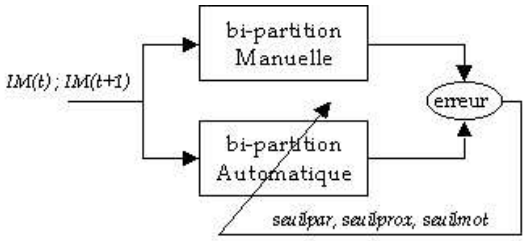

FIG. 5 – **Système d'apprentissage.**

5.1 Les Algorithmes Génétiques

Pour résoudre d'une manière déterministe le problème d'optimisation précédemment décrit, il faut établir un lien explicite entre les seuils utilisés pour le calcul des scores et le deuxième plus petit vecteur propre responsable de la partition. Une telle modélisation semble compliquée car les scores calculés dépendent à la fois des seuils mais aussi de la configuration de chaque segment. A ceci on peut rajouter le nombre d'approximations faites par Malik et Shi [2] pour construire le système (2). Nous avons par conséquent choisi une technique stochastique, en l'occurrence les Algorithmes Génétiques [15]. En effet, ces algorithmes n'ont besoin, pour fonctionner, que de couples entrées (commandes)/sorties (observations), ce qui correspond dans notre cas à des couples seuils/parties [16]. Cette technique stochastique est adéquate à notre application vu l'absence d'un modèle mathématique qui peut traduire le processus proposé (FIG. 5.). Elle est aussi réputée par sa robustesse et par la globalité des optimums qu'elle trouve [17][18]. La fonction à minimiser et qui traduit l'erreur entre les deux bi-partitions s'écrit de la façon suivante :

$$\varepsilon = \begin{cases} +\infty & si \quad p-r > tol \quad ou \quad p-r < 0 \\ \sum_{i=1}^{r} \prod_{j=1}^{p} d(x_i, z_j) & sinon \end{cases} \tag{9}$$

où :
- ε est l'erreur calculée.
- Mp_i (*i=1,2*) sont les deux parties issues de la bi-partition manuelle.
- Ap_i (*i=1,2*) sont les deux parties issues de la bi-partition automatique.
- $Mp_1 = (x_1,...,x_r)$; x_i un segment de Mp_1.
- $Mp_2 = (y_1,...,y_s)$; y_i un segment de Mp_2.
- $Ap_1 = (z_1,...,z_p)$; z_i un segment de Ap_1.
- $Ap_2 = (v_1,...,v_q)$; v_i un segment de Ap_2.
- *n* est le nombre de segments total dans l'image binaire *IM(t)*.
- $r + s = n$; $p + q = n$.
- $d(x_i,z_j)$ est la distance entre x_i et z_i.
- *tol* représente une valeur de tolérance selon laquelle l'optimisation est plus ou moins sévère.

la figure 6 (FIG. 6.) montre un exemple (*n=4*) dans lequel nous calculons pour différentes bi-partitions automatiques, la fonction ε : (*tol = 0* : bi-partition idéale)

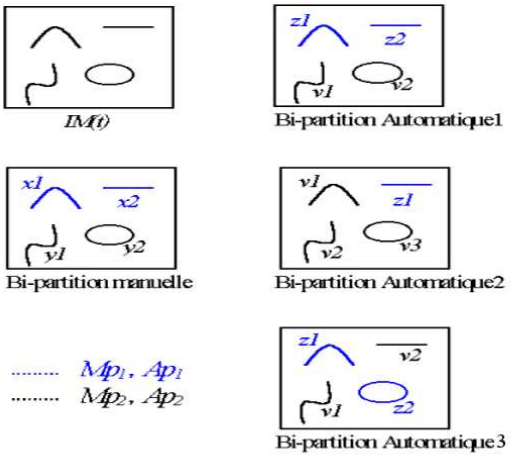

FIG. 6 – **Exemple de calcul pour ε.**

$\varepsilon 1 = 0$.
$\varepsilon 2 = +\infty$.
$\varepsilon 3 = d(x_2,z_1) \times d(x_2,z_2)$
Pour optimiser cette fonction nous avons donc choisi les Algorithmes Génétiques. La question est maintenant, comment décider d'un code génétique unique pour toute notre application ? comment s'assurer qu'il est assez rapide et performant ? plus précisément, sur quels critères allons nous choisir le type de l'algorithme (réel ou codé), le type des opérateurs génétiques mis en oeuvre (sélection, croisement et mutation) et les probabilités de leurs interventions... etc ? L'algorithme choisi doit être robuste, rapide et ayant un taux de réussite élevé.

5.2 Etude d'un Algorithme Génétique réel

Dans cette partie, nous étudions les performances d'un Algorithme Génétique à codage réel en l'appliquant sur des "fonctions test" de la littérature [Annexe]. La particularité de ces fonctions est le fait qu'elles sont

multivariables, qu'elles ont généralement des optimums locaux et qu'elles présentent des non-linéarités. Elles sont considérées comme un bon testeur pour n'importe quel algorithme d'optimisation. L'avantage de l'Algorithme Génétique à codage réel est le fait qu'il manipule directement les valeurs des variables contrairement à un Algorithme Génétique classique qui nécessite deux étapes supplémentaires, une pour coder et l'autre pour décoder les valeurs. Nous gagnons ainsi en terme de temps de calcul.
La figure 7 (FIG. 7.) décrit le "potentiomètre" grâce auquel nous effectuons les tests et nous concluons sur le meilleur code à utiliser pour notre application. Les types d'opérateurs les plus courants dans la littérature sont [16] : pour la séléction : Troncation , Tournoi et RWS (Roulette Wheel Selection).
pour le croisement : 1-point , 2-points et arithmétique.
pour la mutation : constante et variable.

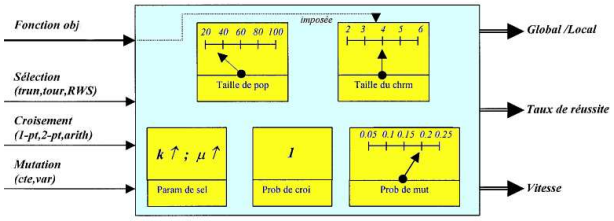

FIG. 7 – **Les paramètres de réglage de l'Algorithme Génétique.**

Nous appliquons toutes les combinaisons possibles (18) sur toutes les fonctions tests en agissant sur les paramètres présents dans le "potentiomètre" décrit sur la figure 7(FIG. 7.) sachant que :
- *k* représente les *k* individus choisis d'une façon aléatoire lors de la sélection par Tournoi.
- μ représente les μ meilleurs individus lors de la sélection par Troncation.
- La probabilité de croisement est fixée à l'unité (valeur courante).
- Le taux de réussite est calculé après avoir effectué dix essais. Le nombre maximal de génération pour chaque essai est fixé à *100*.
- L'algorithme converge lorsqu'il arrive à détecter l'optimum global avec une erreur maximale égale à *1%* avant d'effectuer *100* générations.
- La vitesse de l'algorithme est exprimée de la façon suivante : $VIT = 2 \times P \times NG$ où : *VIT* est la vitesse de l'algorithme, *P* la taille de la population initiale et *NG* le nombre de générations nécessaire pour la convergence.
Le tableau de la figure 8 (FIG. 8.) présente les meilleurs résultats obtenus pour chaque fonction test.

Les caractéristiques de l'algorithme d'optimisation retenu après cette étude sont les suivantes :

Fonction objective	Nombre de variables	Taille de la population	Type de la sélection	Type de croisement	Type de mutation	Taux de réussite	Vitesse de convergence	Optimum trouvé
Z1	2	20	Tronc : μ = 6	Arith : 0.2	Cte : prb = 0.2	100%	680	44.35
Z2	2	20	Tourn : k = 4	Arith : 0.2	Cte : prb = 0.2	100%	520	-35.30
Z3	5	20	Tronc : μ = 6	1-point	Cte : prb = 0.2	100%	2960	-517.71
Z3	5	20	Tourn : k = 6	2-points	Cte : prb = 0.2	100%	2800	-517.65
Z4	2	20	Tronc : μ = 6	Arith : 0.2	Cte : prb = 0.2	100%	1360	186.59
Z4	2	10	Tronc : μ = 4	Arith : 0.2	Cte : prb = 0.25	80%	580	186.58
Z5	4	20	Tronc : μ = 6	2-points	Cte : prb = 0.25	70%	2840	$4.74\,10^{-4}$
Z6	10	40	Tronc : μ = 12	2-points	Cte : prb = 0.35	20%	7280	$1.41\,10^{-3}$
Z7	5	20	Tronc : μ = 6	Arith : 0.2	Cte : prb = 0.1	100%	2240	1.9
Z8	10	20	Tronc : μ = 5	2-points	Cte : prb = 0.2	90%	3600	-14.43
Z9	2	20	Tronc : μ = 6	Arith : 0.2	Cte : prb = 0.2	100%	340	-33.54
Z10	10	20	Tronc : μ = 6	2-points	Cte : prb = 0.2	40%	3240	-59.01

FIG. 8 – **Résultats des tests effectués.**

- Type de codage : réel.
- Taille de la population = *20*.
- Type de la sélection : troncation ; $\mu = 6$.
- Type de croisement : arithmétique avec un rapport d'échange égal à *0.2*.
- Type de mutation : constante ; probabilité de mutation = *0.25*.
La figure 9 (FIG. 9.) décrit l'algorithme mis en oeuvre pour calculer les seuils optimaux. L'algorithme converge quand $\varepsilon = 0$, ou lorsque la population n'évolue plus. La figure 10 (FIG. 10.) montre le type de résultats qu'on peut trouver suite à cette phase d'apprentissage.

```
//   Initialisation aléatoire de la population ;
     Population : [ seuilpar_i   seuilprox_i   seuilmot_i ] ;
                  i : 1,...,P.

//   Evaluation de chaque individu : ε_i ?

//   Tant que condition d'arrêt = 0  Faire

          generation = generation + 1 ;
          Evaluation de chaque individu : ε_i ;
          Selection ;
          Croisement ;
          Mutation ;

//   Fin Tant que.
```

FIG. 9 – **Le code d'apprentissage.**

Voici les seuils calculés pour chaque bi-partition :
Image (1) ⟶ Image(2) : *[0.0425 51.7298 202.6092]*.
Image (2) ⟶ Image(3) : *[0.0356 46.8767 200.8307]*.
Image (3) ⟶ Image(4) : *[0.2124 59.1049 85.8303]*.

6 Résultats expérimentaux

Dans cette partie, nous exposons quelques résultats des essais effectués en boucle ouverte sur des successions d'images. La taille des images traitées est de [300×400].

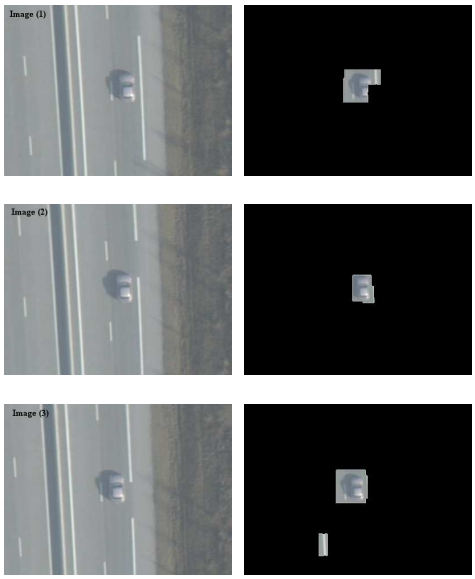

FIG. 10 – **Exemples de résultats expérimentaux fournis par apprentissage.** *tol = 1*.

Les seuils sont inspirés de la phase d'apprentissage. Le temps nécessaire pour toute l'opération (hors détection de contour) est inférieur à une seconde sachant que l'outil de développement est Matlab 5.3.

La figure 11 (FIG. 11.) présente les résultats sur un extrait du trafic routier dans lequel un seul véhicule est présent. Les conditions d'acquisition d'images sont les suivantes :
- Altitude moyenne : *452 m*.
- Latitude moyenne : *46°31'N*.
- Longitude moyenne : *03°19'E*.
Le vecteur des seuils utilisé est : *[0.1 50 200]*.

La figure 12 (FIG. 12.) présente les résultats sur un extrait du trafic routier dans lequel plusieurs véhicules sont présents. Les conditions d'acquisition d'images sont les suivantes :
- Altitude moyenne : *415 m*.
- Latitude moyenne : *46°33'N*.
- Longitude moyenne : *03°23'E*.
Le vecteur des seuils utilisé est : *[0.02 100 80]*.

La figure 13 (FIG. 13.) présente les résultats sur un extrait du trafic routier dans lequel des véhicules circulent dans les deux sens. Les conditions d'acquisition d'images sont les suivantes :
- Altitude moyenne : *413 m*.
- Latitude moyenne : *46°35'N*.
- Longitude moyenne : *03°23'E*.
Le vecteur des seuils utilisé est : *[0.05 50 200]*.

Nous remarquons que la détection des véhicules par

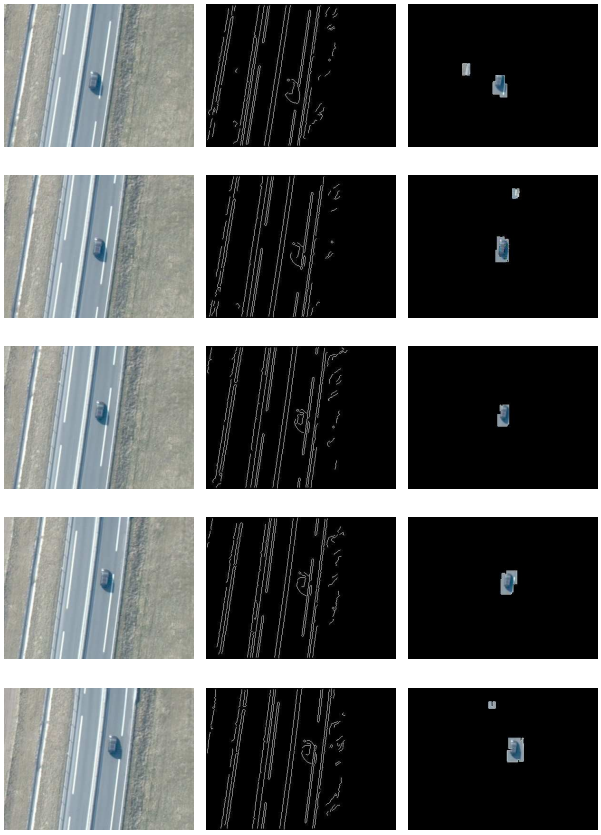

FIG. 11 – **Trafic présentant un véhicule :** La colonne du milieu montre les segments (primitives) détectés par la méthode du canny et utilisés comme noeuds pour la construction du graphe. Dans cet exemple le nombre de segments varie entre 55 et 63.

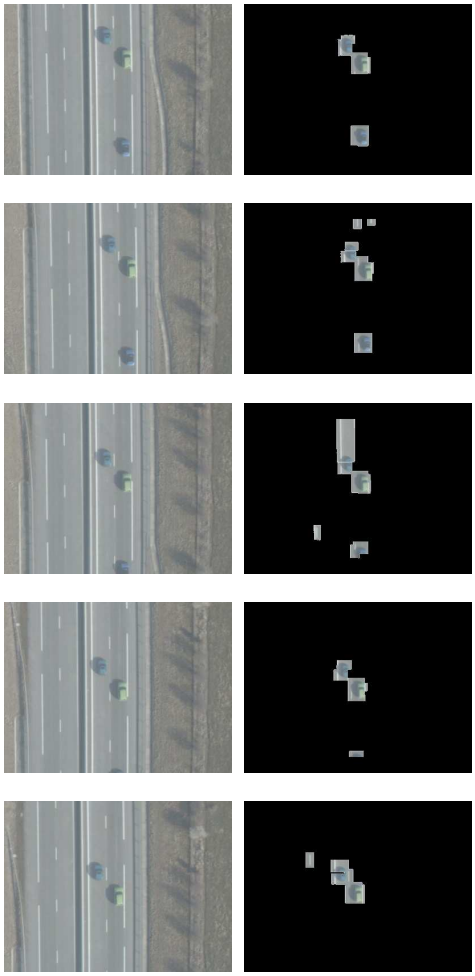

FIG. 12 – **Trafic présentant plusieurs véhicules :** Les images de cette séquence sont plus sombres comparés aux images des autres séquences présentées. Le vecteur des seuils utilisé est par conséquent assez différent.

FIG. 13 – **Trafic présentant deux sens de circulation.**

l'algorithme proposé a réussi. Cependant, des petites zones non significatives sont considérées comme des véhicules sur quelques images. Ceci est dû principalement à la mise en correspondance effectuée lors du calcul du critère de mouvement. En effet, dans certains cas, deux images successives ne sont pas forcément de la même netteté à cause des vibrations de l'hélicoptère, par conséquent la qualité de la segmentation diffère d'une image à l'autre. Ainsi plusieurs segments de l'image binaire *IM(t)* peuvent s'associer à des faux correspondants de *IM(t+1)*. L'une des perspectives de ce travail est l'amélioration de la technique de mise en correspondance pour surmonter les méfaits du bruit.

7 Conclusion

Dans ce papier nous avons présenté une approche originale qui analyse des séquences aériennes présentant un trafic routier. Nous employons la théorie d'organisation perceptive et la méthode des coupes normalisées pour extraire les véhicules à partir d'images successives. Nous proposons d'utiliser les primitives issues de la segmentation comme noeuds du graphe à partitioner. Nous étudions et nous utilisons enfin, les Algorithmes Génétiques pour une phase d'apprentissage qui va permettre un choix adéquat des seuils de l'algorithme proposé. Les résultats expérimentaux sont très encourageants. Ce travail ouvre des perspectives très intéressantes. Tout d'abord, améliorer la mise en correspondance des segments permettra d'améliorer les performances de l'algorithme. Ensuite, établir une base de données plus large pour affiner l'apprentissage et avoir des seuils valables pour des conditions d'acquisition différentes. Enfin, ce travail peut nous servir comme base pour un diagnostic du trafic routier ou encore pour l'implantation d'un système d'asservissement visuel qui vise la poursuite d'un véhicule.

Remerciement

Ce travail entre dans le cadre du projet **ROVA** du pôle **DIVA** : **DI**agnostic et **V**éhicules **A**vancés soutenu par la Région Picardie.
Les auteurs tiennent à remercier Dr. Sudeep Sarkar pour sa collaboration, ses conseils et ses remarques.

Annexe

$Z1 = 43.62 - 1.16x_1 - 1.17x_2 - 1.15x_1^2 - 0.61x_2^2 - 0.31x_1x_2$
$-1.5 \leq x_i \leq 1.5 \quad i = 1, 2$
$X1_{opt} = (-0.39, -0.86) \quad Z1_{opt} = 44.36$

$Z2 = x_1^2 + x_2^2 + 40 \sin x_1 \sin x_2$
$-3 \leq x_i \leq 3 \quad i = 1, 2$
$X2_{opt} = (1.5, -1.5) \quad Z2_{opt} = -35.30$

$Z3 = 0.01x_1^2(x_1+2)(x_1-5)^2(x_1-12) + \sum_{i=1}^{5} x_i^2$
$-5 \leq x_i \leq 1 \quad i = 1, ..., 5$
$X3_{opt} = (10.37, 0, 0, 0, 0) \quad Z3_{opt} = -517.727$

$Z4 = -(\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$
$-10 \leq x_i \leq 0 \quad i = 1, 2$
$X4_{opt} = (-1.425, -7.083) \quad Z4_{opt} = 186.731$

$Z5 = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + \tan(x_3 - x_4)^4 + x_1^8$
$-3 \leq x_i \leq 3 \quad i = 1, ..., 4$
$X5_{opt} = (0, 1, 1, 1) \quad Z5_{opt} = 0$

$Z6 = (1-x_1)^2 + (1-x_{10})^2 + \sum_{i=1}^{9}(x_1^2 - x_{i+1})^2$
$0 \leq x_i \leq 5 \quad i = 1, ..., 10$
$X6_{opt} = (1, ..., 1) \quad Z6_{opt} = 0$

$Z7 = 2 - \frac{1}{1200}(x_1 x_2 x_3 x_4 x_5)$
$0 \leq x_i \leq i \quad i = 1, ..., 5$
$X7_{opt} = (1, 2, 3, 4, 5) \quad Z7_{opt} = 1.9$

$Z8 = -\sum_{i=1}^{10}(x_i^2 + 0.5x_i)$
$-1 \leq x_i \leq 1 \quad i = 1, ..., 10$
$X8_{opt} = (1, ..., 1) \quad Z8_{opt} = -15$

$Z9 = -x_1^2 - 2.1x_1^4 - 0.33x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
$-5 \leq x_i \leq 5 \quad i = 1, 2$
$X9_{opt} = (2.24, 0.81) \quad Z9_{opt} = -33.5507$

$Z10 = \sum_{i=1}^{10}((\ln(x_i - 2))^2 + (\ln(10 - x_i)))^2 - (\prod_{i=1}^{10} x_i)^{0.2}$
$2.9 \leq x_i \leq 9.9 \quad i = 1, 10$
$X10_{opt} = (9.9, ..., 9.9) \quad Z10_{opt} = -59.227$

Références

[1] K. L. Boyer, S. Sarkar, *Perceptual Organization for Artificial Vision Systems*, The KLUWER international series in engineering and computer science, 2000.

[2] J. Shi, J. Malik, Normalized Cuts and Image Segmentation, *Proceeding of the IEEE computer society conference on computer vision and pattern recognition*, pp. 731-737, 1997.

[3] P. Burlina , V. Parameswaran, R. Chellappa, Sensitivity Analysis and Learning Strategies for Context-based, *Vehicle Detection Algorithms. DARPA IU Workshop 97*, pp. 577-584, 1997.

[4] G. Liu, R. M. Haralick, Vehicle Ground-truth Database for the Vertical-view Ft. Hood Imagery, *International Conference on Pattern Recognition, ICPR00*, Vol. 1, pp. 342-345, 2000.

[5] H. Moon , R. Chellappa, A. Rosenfeld, Performance Analysis of a Simple Vehicle Detection Algorithm, *IVC(02)*, Vol. 1, pp. 1-13, 2002.

[6] T. Zhao , R. Nevatia, Car Detection in Low Resolution Aerial Image, *USC computer vision, ICCV 01*, pp. 710-717, 2001.

[7] J. Malik , S. Russell, A Machine Vision Based Surveillance System for California Roads, *University of California. California Path Research*, Mars 1995.

[8] S. Milan , H. Vaclav, B. Roger, Image Processing, Analysis and Machine Vision, *Chapman and Hall*, pp. 524-533, 1993.

[9] I. Rafael, Traffic Monitoring and Control using Machine Vision : A survey, *IEEE Transactions on Industrial Electronics*, vol. 32, No. 3, pp. 177-185, August 1985.

[10] I. Cohen, G. Medioni, Detecting and Tracking Moving Objects in Video Surveillance, *CVPR99*, vol. 2, pp. 319-325, 1999.

[11] S. Sarkar , P. Soundararajan, Supervised Learning of Large Perceptual Organization, *Graph Spectral Partitioning and Learning Automata, IEEE transactions on pattern analysis and machine intelligence*, vol. 22, No. 5, Mai 2000.

[12] Z. Wu , R. Leahy, An Optimal graph theoretic approach to data clustering : theory and application to image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1101-1113, Nov 1993.

[13] P. Soundararajan , S. Sarkar, Investigation of Measures for Grouping by Graph Partitioning, *CVPR01*, vol. 1, pp. 239-246, 2001.

[14] J. Canny, A Computational Approach to Edge Detection, *PAMI(8)*, No. 6, pp. 679-698, Nov 1986.

[15] J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

[16] D.E. Goldberg, *Genetic Algorithms in search Optimization and Machine Learning*, Addison Wesley, 1989.

[17] T. Bäck, F. Hoffmeister, Global optimization by means of evolutionary algorithms, *Random Search as a Method for Adaptation and Optimization of Complex Systems*, pp. 17-21, Mars 1991.

[18] T. Bäck, Self-Adaptation in Genetic Algorithms, *Proceedings of the 1st European Conference on Artificial Life*, pp. 263-271, 1992.