Université de Liège
Faculté des Sciences Appliquées
Département d'Électricité, Électronique et Informatique
Institut Montefiore

Détection et reconnaissance de signaux routiers dans un flux vidéo

Travail de fin d'études présenté en vue de l'obtention du grade d'ingénieur civil électricien (orientation électronique)

Michaël Freylinger et Jean-François Verdin

Promoteur : Prof. Marc Van Droogenbroeck

Année académique 2005–2006

Résumé

L'objectif de ce travail est la réalisation d'un outil de détection, de localisation et de reconnaissance de signaux routiers par acquisition vidéo, en vue de construire une base de données de signaux routiers géoréférencés à l'aide d'un système GPS.

Nous avons d'abord réalisé une étude bibliographique. Nous y avons décrit les méthodes les plus couramment utilisées et celles dont nous nous sommes inspirés pour ce travail.

La détection et la localisation sont accomplies grâce à une cascade de classeurs entraînés sur des filtres de Haar, avec une variante de l'algorithme AdaBoost. Le taux de détection obtenu sur des images isolées est supérieur à 90 % pour environ 1 fausse alarme toutes les 5 images. L'entraînement de la cascade de classeurs a nécessité la création d'une base de données contenant 1988 images de signaux routiers.

Les signaux détectés et localisés sont suivis à l'aide de filtres de Kalman. Un suivi multicible est géré par un algorithme d'association de données qui permet par la même occasion d'éliminer un nombre important de fausses alarmes.

Le programme de détection, implémenté en C sous Linux, est destiné à fonctionner en temps réel. Actuellement non optimisé, il fonctionne à une vitesse de 9 images (640×480) par seconde sur une machine équipée d'un processeur Intel Pentium IV 2.8 GHz. Le système a été testé avec succès sur des séquences vidéo obtenues en environnement urbain et suburbain, dans des conditions météorologiques variables, à des vitesses atteignant 70 km/h.

La reconnaissance des signaux est effectuée entièrement *off-line*. Nous avons testé plusieurs techniques sur des images provenant de 5 classes. La première utilise un modèle génératif dans un espace réduit par une analyse en composantes principales suivie par une analyse discriminante linéaire. La seconde est celle du plus proche voisin dans ce même espace. La troisième méthode est basée sur une extraction aléatoire de fenêtres et sur des ensembles d'*Extra-Trees*. La méthode est implémentée dans le logiciel PiXiT de la société PEPITe. Nous atteignons pour toutes les méthodes des taux de reconnaissance supérieurs à 90 %.

Suite à ce travail exploratoire, de nombreuses perspectives sont envisageables, la plus importante étant l'apport d'un système GPS qui permettrait de générer une base de données de signaux routiers géoréférencés.

Remerciements

En premier lieu, je tiens à remercier le Professeur Marc Van Droogenbroeck, promoteur de ce travail, pour nous avoir proposé ce sujet et nous avoir suivi tout au long de la réalisation de ce travail.

Je remercie ensuite Jean-François Verdin, collègue et ami, avec qui j'ai réalisé ce travail. Une bonne entente et une bonne communication nous ont permis de surmonter bien des difficultés.

Je remercie ensuite Olivier Barnich, Renaud Dardenne et Cédric Demoulin pour avoir participé aux différentes réunions et avoir émis leurs avis aussi bien positifs que négatifs.

Je remercie Raphaël Marée et la société PEPITe pour nous avoir permis d'utiliser leur logiciel PiXiT.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et en particulier à mes parents, à ma sœur Julie, et à ma grand-mère maintenant décédée, qui m'ont toujours soutenu et encouragé au cours de mes études.

Michaël

Je tiens tout d'abord à remercier Monsieur Marc Van Droogenbroeck, promoteur et initiateur de ce travail, pour nous avoir proposé ce sujet et nous avoir guidés tout au long de sa réalisation.

Je remercie chaleureusement Michaël Freylinger, avec qui j'ai réalisé ce travail. Notre collaboration s'est toujours déroulée dans les meilleures conditions. Nous nous sommes soutenus l'un l'autre dans les moments difficiles. Grâce à lui, je retire une expérience enrichissante de ce travail en équipe.

Je tiens également à dire toute ma gratitude à Olivier Barnich, Renaud Dardenne et Cédric Demoulin. Il ont souvent proposé spontanément des idées et sont toujours restés à notre écoute, prêts à nous aider. Je suis aussi reconnaissant envers les professeurs de l'Institut Montefiore qui ont pu nous enseigner les matières indispensables à la réalisation de ce travail.

Je remercie Raphaël Marée pour sa grande disponibilité, et la société PEPITe pour nous avoir permis d'utiliser le logiciel PiXiT.

Merci aussi à Marie-Eve et à Jean-Marc pour leur soutien moral et matériel. Ils ont toujours été disponibles, à l'écoute.

Enfin, je remercie mes parents. Ils m'ont soutenu tout au long de mes études et c'est en grande partie à eux que je dois l'aboutissement de celles-ci. Ils ont tout mis en œuvre pour que je sois dans les conditions les plus favorables. Je leur en serai éternellement reconnaissant.

Jean-François

Notations et symboles

Nous fournissons ci-dessous les notations et symboles les plus souvent utilisés et leur signification, sauf indications contraires mentionnées dans le document.

Généralités

$\mathbf{x}$, $\mathbf{y}$, ...	les lettres minuscules droites et grasses représentent des vecteurs colonnes
$\underline{\mathbf{u}}$, $\underline{\mathbf{v}}$, ...	vecteurs unitaires
$\boldsymbol{A}$, $\boldsymbol{B}$, ...	les lettres majuscules italiques et grasses représentent des matrices, sauf la matrice identité $\mathbf{I}$ qui est droite
$\mathbf{x}^T$, $\boldsymbol{A}^T$	transposée de $\mathbf{x}$, $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	déterminant de la matrice $\boldsymbol{A}$
$\lvert\mathcal{E}\rvert$	taille (cardinal) de l'ensemble $\mathcal{E}$
$\mathcal{R}^n$	espace euclidien des vecteurs réels de dimension n

Espaces de couleurs

r, g, b	valeurs des canaux d'un pixel RGB
h, s, v	valeurs des canaux d'un pixel HSV
Y, u, v	valeurs des canaux d'un pixel YUV

Apprentissage et reconnaissance

$\mathcal{LS}$	ensemble d'apprentissage (*learning set*)
$\mathcal{TS}$	ensemble de test (*test set*)
N	nombre d'objets dans $\mathcal{LS}$
M	nombre de classes dans $\mathcal{LS}$
$\mathbf{a}$	vecteur d'attributs
D	taille du vecteur d'attributs $\mathbf{a}$
$\tilde{\mathbf{a}}$	nouveau vecteur d'attributs obtenu par transformation de $\mathbf{a}$
K	taille du vecteur d'attributs $\tilde{\mathbf{a}}$
$\mathbf{a}^i$	vecteur d'attributs de l'objet i
a_k^i	attribut k de l'objet i
$\mathbf{a}^{i\|j}$	vecteur d'attributs de l'objet i de la classe j
$\mathbf{a}(o)$	vecteur d'attributs de l'objet o
$\boldsymbol{\mu}_j$	vecteur d'attributs moyen de la classe j
$\boldsymbol{\Sigma}_j$	covariance de la classe j
$y(o)$	classe de l'objet o

$\hat{y}(o)$ classe estimée (prédite) de l'objet o

Suivi

$\mathbf{x}$	vecteur d'état de la cible
$\hat{\mathbf{x}}^-$	estimation a priori du vecteur d'état
$\hat{\mathbf{x}}$	estimation a posteriori du vecteur d'état
$\mathbf{z}$	vecteur de mesure
$\mathbf{u}$	vecteur de commande
$\boldsymbol{A}$	matrice de transition d'état
$\boldsymbol{B}$	matrice de commande
$\boldsymbol{H}$	matrice de mesure
$\boldsymbol{Q}$	covariance du bruit du processus à estimer
$\boldsymbol{R}$	covariance du bruit de mesure
$\boldsymbol{P}^-$	estimation a priori de la covariance de l'erreur sur l'estimation de l'état
$\boldsymbol{P}$	estimation a posteriori de la covariance de l'erreur sur l'estimation de l'état

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

La détection et la reconnaissance automatiques de signaux routiers sont des domaines de recherche dont l'activité s'est amplifiée ces dernières années. Cela est dû conjointement à l'augmentation de la puissance des processeurs et à la large gamme d'applications offertes par un système pourvu de ces capacités :

1. *Aides à la conduite et véhicules intelligents.* Les systèmes d'aide à la conduite informent ou assistent le conducteur afin d'éviter l'apparition de situations dangereuses. Les exemples sont nombreux : le système anti-blocage de freins ABS, le contrôle de trajectoire ESP, les systèmes de navigation GPS, l'allumage automatique des feux de croisement, le radar de recul, etc. Un système reconnaissant automatiquement la signalisation routière permettrait par exemple d'informer le conducteur s'il dépasse la vitesse autorisée ou s'il se trouve à proximité d'une école.
2. *Inventaire des signaux et leur maintenance.* Sans système automatique, maintenir une signalisation routière cohérente et en bon état requiert l'intervention d'un opérateur humain qui doit parcourir les routes, ou visionner attentivement de longues séquences vidéo. C'est un travail fastidieux.
3. *Robots autonomes.* Ceux-ci se déplacent sans l'intervention d'êtres humains et utilisent des points de repères pour se localiser (balises, marquages au sol, etc.). Ces points de repère artificiels peuvent être conçus de la même manière que des signaux routiers, en incluant une information visuelle, par exemple une tâche particulière à accomplir par le robot.

Ce travail s'intéresse plus particulièrement à la seconde application. Une base de données contenant une liste de signaux géoréférencés permettrait d'envisager un éventail d'applications :

1. *Maintenance de la signalisation.* À partir d'une base de données existante, des véhicules munis d'un système de reconnaissance pourraient s'assurer de l'intégrité et de la bonne visibilité des signaux. Ces véhicules pourraient être équipés du système sans devoir être nécessairement dédiés à cette application.
2. *Vérification de la cohérence de la signalisation.* Suite à des travaux de voirie ou de modifications de la signalisation, celle-ci devient parfois incohérente (signaux

contradictoires) ou trop abondante. Cela perturbe et met en danger les usagers de la route. Un logiciel utilisant la base de données pourrait débusquer les endroits à risque.

3. *Support aux systèmes d'aide à la conduite.* Une base de données pourrait épauler les systèmes de reconnaissance de signaux qui équiperont les véhicules grand public du futur. Une base de données complète et à jour serait même en mesure de se substituer à ces systèmes.
4. *Perfectionnement de certains logiciels.* L'utilisation de la base de données par les logiciels de calcul d'itinéraires ou par les systèmes de navigation GPS améliorerait leurs performances.
5. *Services à divers organismes.* Un accès à la base de données par les forces de l'ordre ou les compagnies d'assurances pourrait être utile en cas d'accident de la route ou de confrontation entre parties.

L'objectif de ce travail est la réalisation d'un outil de détection, de localisation et de reconnaissance de signaux routiers par acquisition vidéo, en vue de construire une base de données de signaux routiers géoréférencés à l'aide d'un système GPS.

L'ouvrage est organisé de la façon suivante :

- Tout d'abord, une étude bibliographique résume les techniques usuelles qui sont utilisées pour la détection et la reconnaissance des signaux routiers.
- Ensuite, nous décrivons la structure générale que nous avons adoptée pour la conception de notre programme.
- Les techniques que nous avons utilisées et implémentées pour la détection, le suivi et la reconnaissance sont décrites dans les chapitres suivants. Les résultats et performances obtenus sont communiqués progressivement.
- Nous exposons ensuite une série d'améliorations possibles et les perspectives ouvertes par ce travail.
- L'ouvrage se termine par les conclusions.

Chapitre 2

Étude bibliographique

De nombreuses recherches ont été réalisées au sujet de la détection et de la reconnaissance automatiques de signaux routiers. La plupart travaillent image par image et adoptent une approche en deux étapes. Une première phase consiste à détecter et à localiser les régions d'intérêt dans lesquelles se situent les signaux routiers. Ensuite, une phase de reconnaissance permet de déterminer le type de signal contenu dans chaque région d'intérêt. L'efficacité de la méthode de détection est très importante. On doit tenter de détecter tous les signaux présents dans l'image et de rejeter au maximum les détections d'autres objets, car celles-ci rendraient le travail de reconnaissance plus difficile. Certains travaux ajoutent une étape entre la détection et la reconnaissance : le suivi des signaux au travers d'un flux vidéo. Le suivi permet de faciliter grandement la reconnaissance puisque grâce à lui, celle-ci peut se baser sur une série d'images pour un même signal.

Le présent chapitre va tenter de donner un aperçu des méthodes utilisées dans les récents travaux de recherche. Premièrement, la section 2.1 présentera brièvement les moyens mis en place pour l'acquisition des images. Ensuite, dans la section 2.2, nous examinerons les techniques utilisées pour la détection et la localisation des signaux. Dans la section 2.3, nous nous pencherons sur le problème du suivi des signaux. Enfin, nous traiterons des techniques utilisées pour la reconnaissance des signaux dans la section 2.4. Les méthodes dont nous nous servirons seront décrites plus en détails dans les chapitres suivants.

2.1 Acquisition des images

Le but des travaux de recherche en détection et reconnaissance automatiques de signaux routiers est généralement de construire un outil d'assistance à la conduite. Cet outil devrait être capable de travailler en temps réel et devrait comporter un module dédié à l'acquisition du flux vidéo. Cependant, les recherches se concentrent essentiellement sur les parties détection et reconnaissance des signaux. Les travaux effectués se basent soit sur des images fixes, soit sur des séquences vidéo acquises en conditions réelles.

Étant donné la complexité du problème de détection et de reconnaissance automatiques des signaux, certains auteurs se limitent à travailler sur des images fixes. Les images peuvent être obtenues en se rendant directement sur le terrain pour prendre des photographies des signaux désirés [ZKS98]. Ce travail est évidemment long et fastidieux. D'autres auteurs travaillent sur des images fixes obtenues à partir de séquences vidéo [Shi04]. À la suite de certains travaux, des bases de données d'images de signaux sont mises en ligne à disposition du public [Datb][Data]. Elles ne sont malheureusement pas toujours représentatives et de taille importante.

Pour acquérir les séquences vidéo, on trouve différentes variantes. Fang et al. [FFY+04] utilisent un caméscope placé au niveau du pare-brise. Ils travaillent sur autoroute à des vitesses supérieures à 90 km/h. Ils utilisent le mode entrelacé, qui donne de meilleurs résultats à ces vitesses. D'autres placent la caméra sur le toit du véhicule [Shn05]. Certains auteurs travaillent avec un zoom et une orientation fixes [VPPS01]. Miura et al. [MKS00] travaillent avec deux caméras : une caméra grand angle et une caméra à téléobjectif. La caméra grand angle détecte et localise les signaux. Lorsqu'un signal est suivi dans la séquence vidéo depuis un certain temps, on focalise la caméra à téléobjectif sur lui. Grâce à cette caméra, on continue le suivi et on réalise la reconnaissance. Une illustration du système complet est représentée sur la figure 2.1.

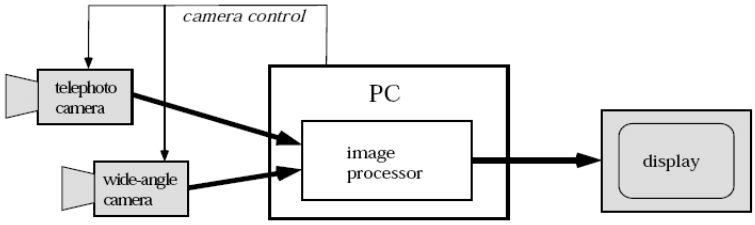


Fig. 2.1 – Vue d'ensemble du système utilisé dans [MKS00].

Certains auteurs travaillent directement sur des séquences vidéo, plusieurs d'entre eux utilisent l'information temporelle disponible pour effectuer un suivi des signaux au travers des séquences [BZR+05]. La plupart des auteurs travaillent *off-line*, peu de travaux comportent des tests temps réel comme ceux réalisés par Franke et al. [VPH01, chapitre 6].

Lorsque l'on réalise l'acquisition sur un véhicule en mouvement (parfois à grande vitesse), la qualité des séquences obtenues est variable. On peut obtenir des images floues. Certains travaux visent à régler ce problème, par exemple en se servant des marques longitudinales qui délimitent les bandes de circulation [LTC+04].

Pour notre travail, nous réalisons l'acquisition de séquences vidéo à partir d'un caméscope monté dans un véhicule. Cela nous permet d'obtenir rapidement une quantité importante de données et de pouvoir ainsi effectuer un suivi des signaux au travers des séquences vidéo.

2.2 Détection et localisation des signaux

Quand nous conduisons notre voiture, nous n'avons aucun mal à déterminer où se situent les signaux routiers dans notre champ de vision. Il est malheureusement difficile de comprendre comment notre cerveau réalise cette tâche. Néanmoins, il paraît évident que plusieurs éléments jouent un rôle : les signaux routiers ont des couleurs et des formes particulières qui dénotent du reste du paysage. Aussi les méthodes classiques de détection et de localisation des signaux routiers utilisent deux types d'information : les couleurs et la forme des signaux. La section 2.2.1 traite des méthodes liée à l'information couleur et la section 2.2.2 présente des techniques utilisant la forme. Dans la section 2.2.3, nous décrivons l'application aux signaux routiers d'une technique générale de détection d'objets : le détecteur de Viola et Jones. Enfin, dans la section 2.2.4, nous présentons une méthode de détection qui permet d'inclure des connaissances a priori sur les images.

2.2.1 Utilisation de l'information couleur

Les signaux routiers ont des couleurs caractéristiques même si celles-ci varient, par exemple avec l'âge du signal. De nombreux auteurs travaillent sur les signaux à bord rouge. Généralement, ils utilisent l'information couleur pour segmenter leurs images. Certains isolent directement les pixels à forte composante rouge, d'autres segmentent les images en différentes régions et labellisent ces régions selon leur couleur afin de se servir de la disposition spatiale entre régions de couleurs différentes pour une étape ultérieure. Pour la segmentation, certains auteurs utilisent des réseaux de neurones ou des classeurs polynomiaux [VPH01, chapitre 6], mais la plupart optent pour une approche plus intuitive et relativement simple.

2.2.1.1 Espace RGB

En raison des variations de l'illumination suivant l'heure, la saison ou les conditions météorologiques, il n'est pas conseillé d'utiliser les canaux classiques RGB. Certains auteurs se basent quand même sur l'espace RGB, principalement pour économiser le temps de calcul lié à la conversion entre espaces de couleur. Ces auteurs doivent alors se prémunir des effets des variations de luminosité. Dans la plupart des cas, les auteurs fixent des seuils α, β et γ et des conditions à respecter pour les valeurs rouge r, verte g et bleue b d'un pixel pour que celui-ci soit considéré comme rouge. Les auteurs fixent les valeurs des seuils le plus souvent après un échantillonnage des valeurs des pixels des signaux dans différentes images. Voici les conditions utilisées par différents auteurs :

– Shneier [Shn05] :

$$\frac{r}{g} > \alpha_m \text{ ET } \frac{r}{b} > \beta_m \text{ ET } \frac{g}{b} > \gamma_m. \tag{2.1}$$

– Escalera et al. [dlEM97] et Shojania [Sho02] :

$$\alpha_M > r > \alpha_m \text{ ET } \beta_M > \frac{g}{r} > \beta_m \text{ ET } \gamma_M > \frac{b}{r} > \gamma_m. \tag{2.2}$$

– Torresen et Sekanina [TBS04] ainsi que Benallal et Meunier [BM03] :

$$r > \alpha_m \text{ ET } (r-g) > \beta_m \text{ ET } (r-b) > \gamma_m, \tag{2.3}$$

pour des luminosités extrêmes (faible ou forte), les seuils sont adaptés dans [TBS04] pour plus de robustesse.

Dans le même ordre d'idée, Zadeh et al. [ZKS98] définissent une région cylindrique dans l'espace RGB pour effectuer la segmentation. Yang [YLLH03], pour sa part, utilise un seuillage très léger défini par : $2r > g + b$. Ce seuillage lui permet néanmoins d'éliminer une grande partie des pixels de l'image.

2.2.1.2 Espace HSV

Certains auteurs utilisent l'espace HSV [1], espace de couleurs moins sensible aux variations d'illumination. Dans cet espace, la teinte h permet de caractériser la couleur d'un pixel, la saturation s sa pureté, et la valeur v sa luminosité. L'avantage est que la teinte est peu dépendante de la luminosité, mais elle est instable lorsque la saturation est faible.

Vitabile et al. [VPPS01] utilisent une technique d'agrégation dynamique des pixels. Ils filtrent les pixels de l'image qui ont des coordonnées h, s et v proches des coordonnées standard pour les pixels des signaux. Ils divisent l'image en régions rectangulaires et choisissent, dans chaque région, un pixel source [2] parmi les pixels filtrés. Dans chaque région, un pixel est considéré comme similaire au pixel source si la distance euclidienne dans l'espace cylindrique HSV entre les deux pixels est inférieure à un seuil t, non linéaire avec la saturation et défini par :

$$t = k - \sin(s_{seed}), \tag{2.4}$$

où k est un paramètre de normalisation et s_{seed} est la saturation du pixel source, comprise entre 0 et 1. Le seuil devient donc plus grand pour les faibles valeurs de saturation. Ceci permet de réduire les problèmes liés à l'instabilité de la teinte lorsque la saturation est faible. Les pixels similaires aux pixels sources sont conservés. Dans chaque région, la segmentation de l'image s'effectue donc par agrégation des pixels à partir du pixel source.

Escalera et al. [dlEAM03] travaillent uniquement sur les composantes h et s. Ils appliquent aux images deux *lookup tables* (voir figure 2.2) et multiplient ensuite

[1] *Hue Saturation Value.*
[2] *Seed pixel.*

les images obtenues. Par rapport au classique ET logique entre images obtenues, la multiplication permet aux deux composantes (teinte et saturation) de « s'entraider » lorsqu'une des deux a une faible valeur, afin de passer le seuillage appliqué après la multiplication.

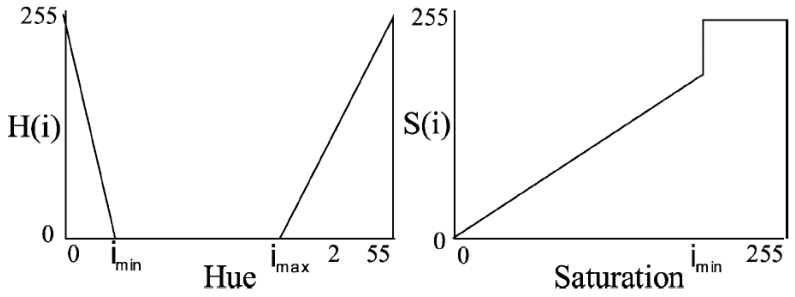


FIG. 2.2 – *Lookup tables* sur la teinte (à gauche) et la saturation (à droite) utilisées dans [dlEAM03].

SHADEED [SAANM03] allie les informations de l'espace HSV et de l'espace YUV [3]. Il effectue un seuillage combiné sur la teinte h et sur les composantes u et v. Ce seuillage est précédé par une égalisation d'histogramme et un contrôle de la luminosité effectué sur la canal luminance Y.

FANG et al. [FFY+04] supposent de leur côté que chaque couleur particulière d'un signal peut être représentée par une valeur de teinte distribuée de manière gaussienne avec une variance σ^2. L'ensemble de toutes ces valeurs de teinte est noté $\{h_1, h_2, ..., h_q\}$. Les auteurs calculent un degré de similarité z entre la teinte h d'un pixel et les teintes h_k des couleurs de signaux :

$$z = \max_{k=1,...,q}(z_k) \qquad \text{avec} \qquad z_k = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(h-h_k)^2}{2\sigma^2}} \tag{2.5}$$

Ainsi, ils obtiennent, non pas une image segmentée, mais une image où chaque valeur de pixel représente la similarité entre la couleur du pixel et les couleurs standard des pixels de signaux.

2.2.1.3 Conclusions

Dans la plupart des cas, les traitements qui viennent d'être décrits donnent en sortie une image segmentée. On ne peut pas vraiment comparer les performances des méthodes, elles sont liées le plus souvent aux seuils choisis. Ceux-ci ne doivent d'ailleurs pas être fixés précisément. Le but est d'éliminer une grande partie des pixels

[3] Espace colorimétrique notamment utilisé dans les systèmes de transmission télévisuelle PAL et NTSC. L'information couleur est contenue dans les composantes u et v et l'information luminance est contenue dans la composante Y.

de l'image tout en conservant presque tous les pixels des signaux, ceci afin de faciliter les opérations qui suivront. La proportion de pixels que l'on souhaite supprimer est d'ailleurs fonction de ces opérations. D'autres objets que les signaux sont segmentés, des phares de voitures ou des enseignes par exemple. Ceci n'est pas grave tant que l'on se donne les moyens d'éliminer ces objets par d'autres traitements.

2.2.2 Utilisation de la forme

Les signaux routiers ont des formes de figures géométriques remarquables : cercle, triangle, carré, rectangle, losange, octogone. La forme particulière d'un signal permet de le distinguer de nombreux autres objets présents sur une image. L'information donnée par la forme est généralement utilisée à la suite d'une première segmentation basée sur l'information couleur. Généralement, on diminue d'abord le bruit dans l'image segmentée par des opérations morphologiques. La plupart des auteurs calculent ensuite les composantes connexes de l'image et labellisent les objets obtenus. Certains auteurs travaillent sur la totalité des pixels de l'objet. D'autres se contentent de l'information fournie par les contours.

Beaucoup de méthodes sont intéressantes, mais les auteurs sont rarement précis sur les résultats obtenus. Nous présentons ici, parfois très brièvement, certaines méthodes. Même s'il n'existe pas de technique universelle, nous nous forgerons ainsi une idée sur l'éventail de techniques disponibles.

2.2.2.1 Contraintes sur l'entièreté de l'objet

Pour trouver les signaux routiers dans l'ensemble d'objets segmentés, certains auteurs [Shn05][Shi04] utilisent des contraintes sur des caractéristiques géométriques simples des objets :

- l'aire ;
- la hauteur et la largeur ;
- le rapport hauteur sur largeur ;
- le rapport entre l'aire de l'objet et l'aire du rectangle dans lequel il est inscrit ;
- ...

Malheureusement, ces contraintes ne suffisent pas à dissocier les signaux routiers de certains autres objets comme les feux de voiture par exemple.

Franke et al. [VPH01, chapitre 6] travaillent avec des régions labellisées suivant 6 classes de couleurs. En plus des contraintes imposées ci-dessus, ils ajoutent des contraintes sur les relations de couleur entre :

- les régions adjacentes ;

– les régions incluses dans une autre région.

Dans une autre approche, VITABILE et al. [VPPS01] travaillent sur des objets considérés comme rouges. Ils calculent sur chacun des objets une mesure de similarité avec un ensemble d'images binaires représentant chacune une forme de signal routier. Les modèles étant de taille fixe (36×36 pixels), les régions d'intérêt sont remises à la bonne échelle avant la mesure de similarité. La fonction de similarité utilisée est le coefficient de TANIMOTO : si $\mathcal{X}$ et $\mathcal{Y}$ sont les ensembles de pixels des images binaires comparées, le coefficient de TANIMOTO s est défini par :

$$s = \frac{|\mathcal{X} \cap \mathcal{Y}|}{|\mathcal{X} \cup \mathcal{Y}|}. \tag{2.6}$$

Remarquons que $0 \leq s \leq 1$ et que s augmente avec la similarité entre le modèle et la région d'intérêt testée. Les auteurs ont effectué des tests sur 620 images de 24 classes. Ils obtiennent plus de 86 % de détections.

TORRESEN et SEKANINA [TBS04] travaillent sur des signaux de limitation de vitesse, à partir d'images dans lesquelles les objets sont labellisés suivant 3 classes : rouge, noir et blanc. Ils appliquent un algorithme de *template matching*. Ils déterminent si au moins 50 % des pixels rouges correspondent au cercle du modèle. Pour que la méthode soit valable pour plusieurs tailles de signaux, plusieurs modèles sont utilisés (voir figure 2.3).

Des contraintes supplémentaires sont imposées sur la zone circulaire intérieure du modèle, afin d'éviter les fausses détections. Les pixels rouges doivent être distribués symétriquement et ne doivent pas se situer au centre du cercle. Des pixels noirs et blancs doivent se trouver à l'intérieur du cercle dans une certaine proportion. Ceci sert notamment à éliminer les signaux circulaires autres que les signaux de limitation de vitesse.

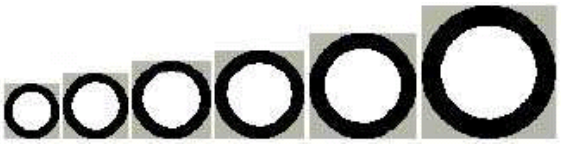


FIG. 2.3 – Modèles utilisés pour le *template matching* dans [TBS04]. La taille des modèle varie de 32×32 à 78×78 pixels.

La méthode utilisée a détecté et localisé correctement les 115 images de signaux parmi les 198 images de l'ensemble de test. Cinq détections de signaux ont été relevées dans des images n'en contenant pas. Les résultats obtenus sont intéressants, cependant la méthode utilisée ne s'applique qu'aux signaux de limitation de vitesse.

2.2.2.2 Contraintes sur les contours de l'objet

De nombreux auteurs se basent sur une détection de contours effectuée sur une image déjà segmentée grâce à l'information couleur.

La méthode utilisée par Gavrila [Gav98] est basée sur une transformée en distance. Une transformée en distance convertit une image binaire, composée de pixels caractéristiques et de pixels non caractéristiques, en une image où chaque valeur de pixel représente la distance du pixel au plus proche pixel caractéristique (voir figure 2.4). La distance utilisée peut être la distance euclidienne, mais d'autres distances existent. Les caractéristiques utilisées par Gavrila sont les contours.

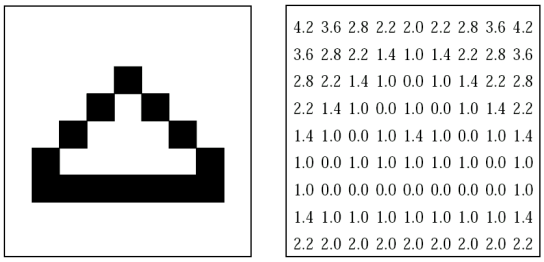


Fig. 2.4 – Une image binaire et sa transformée en distance (distance euclidienne). [Gav98]

À partir d'un modèle binaire de signal routier T, où les pixels « *on* » (pixels caractéristiques) représentent les contours, Gavrila calcule la distance de chanfrein entre T et l'image I, définie par :

$$D_{\text{chanfrein}}(\mathrm{T},\mathrm{I}) = \frac{1}{|\mathrm{T}|}\sum_{t\in\mathrm{T}} d_{\mathrm{I}}(t) \tag{2.7}$$

où $|\mathrm{T}|$ est le nombre de pixels « *on* » dans le modèle et $d_{\mathrm{I}}(t)$ la valeur de la transformée en distance de l'image I au pixel t. La distance de chanfrein évalue la ressemblance entre le modèle et l'image [4].

Pour rendre la méthode robuste, l'auteur répartit les caractéristiques en différentes catégories et effectue le calcul de correspondance pour chacune d'entre elles. Les contours sont répartis en bords de même orientation. Ainsi, les correspondances entre le modèle et l'image ne se font plus sur des distances par rapport au plus proche contour, mais sur des distances par rapport au plus proche bord de même orientation.

Plusieurs modèles sont utilisés. Il faut un modèle pour chaque forme de signal : triangulaire sur base, triangulaire sur pointe et circulaire. De plus, pour être capable de détecter les signaux à différentes échelles, il faut utiliser différents modèles, ce qui augmente le temps de calcul. Pour diminuer ce temps de calcul, l'auteur adopte une approche pyramidale avec une hiérarchie de modèles.

[4] La distance de chanfrein est sensible aux occlusions. D'autres mesures permettent de réduire les effets des occlusions, comme la distance de Hausdorff.

La hiérarchie de modèles est obtenue de la façon suivante. On construit chaque étage de la hiérarchie en partant du plus bas, où tous les modèles sont séparés. À chaque étage, on applique un algorithme semblable aux *K-means*. L'entrée de l'algorithme est l'ensemble des N modèles présents à l'étage courant $\{\mathrm{T}_1, ..., \mathrm{T}_N\}$, leur matrice de similarité $D_{\text{chanfrein}}(i, j)$ et le nombre de groupements voulus K. En sortie, on obtient K groupements $\mathcal{S}_1, ..., \mathcal{S}_K$ et un ensemble de modèles prototypes $\{\mathrm{P}_1, ..., \mathrm{P}_K\}$. Le groupement est accompli par itération en minimisant la fonction objectif suivante :

$$E = \sum_{k=1}^{K} \max_{\mathrm{T}_i \in S_k} D_{\text{chanfrein}}(\mathrm{T}_i, \mathrm{P}_k^*), \tag{2.8}$$

où P_k^* est le prototype du groupe k à l'itération courante. Sur 1000 images testées, Gavrila rapporte 90 % des signaux détectés et 4 à 6 fausses détections par image.

Miura et al. [MKS00] adoptent une autre approche basée sur les contours. À partir de ceux-ci, ils appliquent une transformée de Hough pour les signaux circulaires. Le principe de la transformée de Hough est très simple. Chaque pixel d'un contour peut appartenir à un cercle dont le centre est situé sur la droite passant par le pixel et perpendiculaire au contour en ce pixel. Chaque pixel vote pour les positions possibles du centre du cercle dont il pourrait être issu. On choisit pour centre du cercle l'endroit qui a reçu le plus de votes. Pour les signaux rectangulaires, les auteurs font des projections sur les axes vertical et horizontal et cherchent les pics correspondant aux contours des signaux.

Yang et al. [YLLH03] appliquent une détection de contours avec un LoG [5]. Ils travaillent uniquement sur les composantes connexes d'objets de couleur rouge obtenus via une segmentation couleur ; ils déterminent les contours connectés qui forment ces objets. Les auteurs s'occupent seulement des signaux triangulaires. Ils imposent des contraintes sur les aire, position, hauteur, largeur et rapport hauteur sur largeur des contours des objets [6]. En plus, ils effectuent une détection de coins et vérifient que les positions relatives des 3 coins de l'objet correspondent bien à une forme triangulaire.

Zadeh et al. [ZKS98] font une détection de contours et cherchent des bords circulaires ou des polygônes (les angles et le rapport entre les longueurs des côtés sont connus sauf pour les signaux rectangulaires). Les auteurs vérifient les différentes couleurs des régions connexes. Celle-ci doivent respecter certaines relations pour que l'ensemble soit considéré comme un signal, par exemple : un cercle rouge avec un intérieur blanc.

Escalera et al. [dlEAM03] utilisent un algorithme génétique en travaillant sur les contours des objets déjà détectés. Les algorithmes génétiques cherchent une solution approchée à un problème d'optimisation en utilisant la notion de sélection naturelle développée au XIX$^{\text{e}}$ siècle par le scientifique Darwin et l'appliquent à une population de solutions potentielles au problème donné. L'algorithme génétique donné dans [dlEAM03] est décomposé en 6 étapes :

[5] *Laplacian of Gaussian.*

[6] Nous l'avons vu dans la section 2.2.2, des contraintes similaires ont été appliquées sur l'entièreté des objets [Shn05][Shi04].

1. Initialisation de la population.
2. Évaluation du score d'adaptation de chaque individu.
3. Sélection des bons individus (solutions) pour générer une nouvelle population.
4. Génération de la nouvelle population.
5. Évaluation des résultats de la nouvelle population.
6. Échange de la vieille population contre la nouvelle.

Les étapes 3 à 6 sont répétées un certain nombre de fois (générations) ou jusqu'à ce que la solution soit trouvée par un individu.

Pour déterminer la population initiale, les auteurs partent d'une image contenant un certain nombre d'objets. Chaque objet se voit assigner un nombre fixé d'individus. Chaque individu possède son propre code génétique. On part d'un modèle de signal. Les signaux apparaissant dans les images sont des versions déformées de ce modèle. Ces déformations sont approchées par une transformation affine :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} E_x \cos\theta & E_x \sin\theta & T_x \\ -E_y \sin\theta & E_y \cos\theta & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_m \\ y_m \\ 1 \end{pmatrix}. \tag{2.9}$$

(x, y) et (x_m, y_m) sont les positions des pixels de l'individu et du modèle. Le code génétique de l'individu reprend les paramètres suivants : (T_x, T_y) le vecteur de déplacement, E_x et E_y les facteurs d'échelle horizontal et vertical, θ est l'angle de rotation.

On calcule le score d'adaptation de chaque individu sur base de la distance partielle de Hausdorff [7] entre les contours de l'objet et la transformation affine codée dans les chromosomes de l'individu. Les bons individus (score d'adaptation important) sont sélectionnés pour la génération suivante (loi de la sélection naturelle). La population de la génération suivante est obtenue en croisant les gênes des individus sélectionnés. Ce croisement doit être positif, autrement dit, le score d'adaptation doit augmenter.

De génération en génération, on obtient des individus plus proches de l'objet détecté. Un exemple est montré sur la figure 2.5. Sur la figure (a), on voit le meilleur individu lors de l'initialisation. Sur la figure (d), on montre le meilleur individu de la génération finale. Les figures (c) et (d) montrent les meilleurs individus pour des générations intermédiaires. Les pixels bleus sont les pixels du modèle recherché. Les pixels jaunes sont ceux de l'image [8] et les pixels blancs sont les pixels communs.

Certains auteurs se basent seulement sur une partie des contours des signaux. Ainsi, le détecteur optimal de coins de Rangarajan et al. [RSvB88] est utilisé par Escalera et al. [dlEM97] et par Hassan Shojania [Sho02] pour détecter les signaux circulaires, triangulaires et octogonaux. Ce détecteur consiste en un noyau de convolution qui fournit une réponse maximale à l'endroit du coin en présence de bruit gaussien.

[7] La distance partielle de Hausdorff a l'avantage d'être relativement insensible aux occlusions.

[8] Ce sont les pixels des contours de l'objet segmenté de l'image. Les pixels ont subi la transformation affine codée dans les gênes de l'individu.

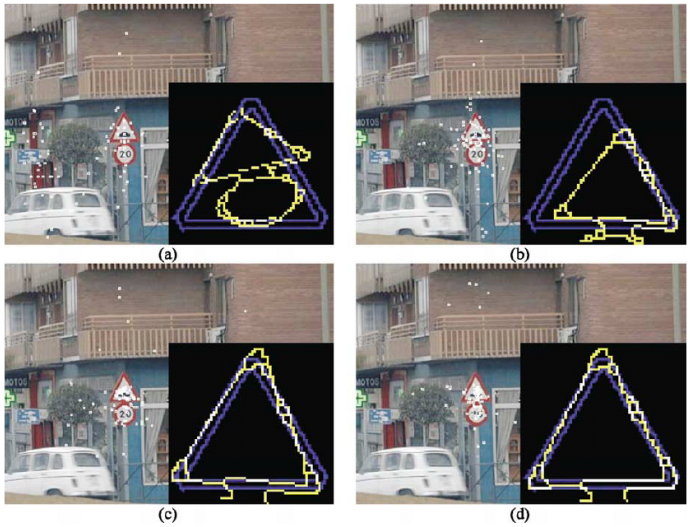


FIG. 2.5 – Évolution du meilleur individu au fil des générations. [dlEAM03].

2.2.3 Le détecteur de VIOLA et JONES

VIOLA et JONES apportent dans [VJ01] trois contributions importantes dans le domaine de la détection d'objets. Premièrement, les auteurs introduisent « l'image intégrale », qui permet de calculer très rapidement les attributs utilisés par leur détecteur. Ces attributs évoquent les ondelettes de HAAR. Ensuite, un algorithme d'apprentissage basé sur l'algorithme AdaBoost est présenté. Il sélectionne les attributs les plus intéressants parmi un grand nombre d'attributs disponibles, ce qui mène à des classeurs extrêmement efficaces. Enfin, une méthode pour combiner des classeurs de complexité croissante sous forme de cascade est introduite. Celle-ci permet d'éliminer rapidement les régions de l'image ne contenant pas d'objet pour passer plus de temps de calcul sur les régions intéressantes.

L'article de Viola et Jones a inspiré de nombreuses recherches dans le domaine de la détection d'objets, entre autres pour la détection des signaux routiers.

Dans [BZR+05], les auteurs reprennent l'idée de VIOLA et JONES, mais à la différence de ces derniers, ne travaillent pas sur des images en niveaux de gris. Ils utilisent 7 canaux couleurs :

1. Les canaux classiques r, g et b.
2. Les canaux normalisés $R = r/s$, $G = g/s$ et $B = b/s$, avec $s = r + g + b$.
3. Le canal en niveaux de gris $s/3$.

L'apport de la couleur leur permet de diminuer le taux de fausses détections par un facteur 10 par rapport à l'utilisation en niveaux de gris de la méthode de VIOLA et JONES.

Dans [BV04], les auteurs se basent également sur l'article de VIOLA et JONES [VJ01]. Ils soulignent le problème de la technique Adaboost. Celle-ci nécessite de

construire de nombreux étages pour que la cascade de classeurs possède un taux de fausse alarme raisonnable. Ceci entraîne un surapprentissage. La cascade classe presque parfaitement les échantillons d'apprentissage, mais n'a aucun pouvoir de généralisation sur le classement d'échantillons inconnus. La solution adoptée par les auteurs a été de combiner une cascade avec moins d'étages avec d'autres méthodes pour éliminer les fausses détections. Le schéma utilisé est représenté figure 2.6.

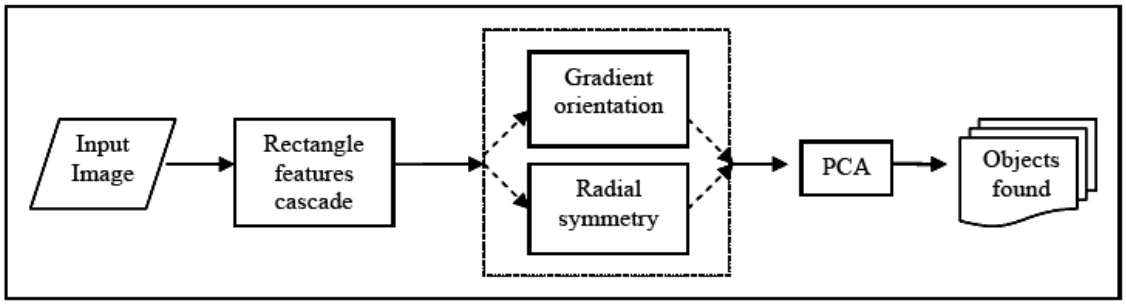


FIG. 2.6 – Structure du détecteur utilisé dans [BV04].

Après la cascade, le traitement est séparé selon la forme des signaux (triangulaire ou circulaire). Les signaux circulaires étant difficilement descriptibles par des attributs rectangulaires, la cascade fournit en sortie un grand nombre de fausses détections. Les auteurs emploient alors une méthode rapide qui utilise la symétrie radiale locale pour extraire des points d'intérêt [LZ03]. Cette méthode est sensible aux relations entre l'intensité du signal et l'intensité du fond de l'image. Cela oblige donc les auteurs à utiliser un critère de convergence basé sur la convergence du gradient.

Pour les signaux triangulaires, les auteurs se basent sur le fait que la direction du gradient est constante sur les trois côtés des signaux et qu'elle change d'une manière connue aux pointes du triangle. Ils classent les valeurs de la direction du gradient en 8 groupes. Ils cherchent ensuite les variations dans la direction du gradient pour trouver les pointes du triangle et utilisent les relations géométriques pour trouver la forme complète.

Les auteurs terminent par une approche globale (valable pour les signaux triangulaires et circulaires) pour éliminer les fausses détections. Ils appliquent une analyse en composantes principales. Celle-ci permet d'éliminer les fausses détections, car en tenant compte uniquement des premières composantes principales, l'erreur de reconstruction d'une fausse détection est plus grande que celle d'une image contenant un vrai signal.

2.2.4 Fusion de connaissances

Quand un signal est détecté, BAHLMANN et al. [BZR+05] suivent le signal à l'aide d'un modèle dynamique simple. Une détection robuste est obtenue en fusionnant

des informations contenues dans les images consécutives du flux vidéo. Ce principe de fusion est décrit dans les travaux de Zhu et al. [ZCR+05] dans le cadre de la détection de véhicules.

Au contraire des détecteurs basés uniquement sur l'aspect et prenant une décision instantanée, Zhu et al. proposent de fusionner les informations d'aspect, de géométrie et de mouvement, provenant des images successives du flux vidéo. Ils montrent ainsi que la fusion de connaissances améliore considérablement la robustesse et la fiabilité du système de détection.

Les explications suivantes concernent la détection de véhicules [ZCR+05], mais s'étendent facilement aux signaux routiers.

Soient $\{I_k\}_{k=1}^m$ m images consécutives d'un flux vidéo, $(\mathbf{x}_k, \mathbf{s}_k)$ la position $(\mathbf{x}_k)$ et la taille $(\mathbf{s}_k)$ du véhicule détecté dans la k^{e} image, et $I_k(\mathbf{x}_k, \mathbf{s}_k)$ la sous-image de taille $\mathbf{s}_k$ à la position $\mathbf{x}_k$ de la k^{e} image. La suite $\{(\mathbf{x}_k, \mathbf{s}_k)\}_{k=1}^m$ définit une trajectoire du véhicule sur le plan image de la caméra (figure 2.7). Étant données les observations

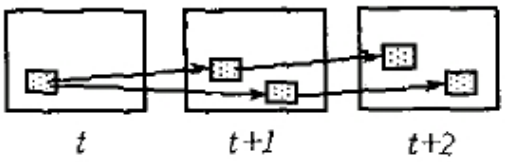


Fig. 2.7 – Trajectoires de véhicules sur le plan image. [ZCR+05]

de m images consécutives $\{I_k\}_{k=1}^m$ et la connaissance de la géométrie de la scène, la vraisemblance du véhicule dans la suite d'images est

$$\begin{aligned} & p_m\left((\mathbf{x}_1, \mathbf{s}_1), \ldots, (\mathbf{x}_m, \mathbf{s}_m) \mid I_1, \ldots, I_m\right) \\ & \cdot \prod_{k=1}^{m} p_g\left((\mathbf{x}_k, \mathbf{s}_k) \mid \text{géométrie de la scène}\right) \\ & \cdot \prod_{k=1}^{m} P_a\left(I_k(\mathbf{x}_k, \mathbf{s}_k) \in \text{véhicule}\right). \end{aligned} \tag{2.10}$$

Le premier facteur de 2.10 est la vraisemblance de la trajectoire du véhicule. L'indice m signifie que ce terme fait appel à l'information que l'on a sur le mouvement du véhicule. La connaissance sur le mouvement est utilisée pour associer les apparitions du véhicule dans les images successives. Ce facteur dépend de la façon dont on modélise le mouvement du véhicule.

Le second facteur de 2.10 est la vraisemblance de la trajectoire du véhicule étant donné l'information que l'on possède sur la géométrie (indice g) de la scène. La connaissance de la scène et de la géométrie des véhicules est utilisée afin d'imposer des contraintes sur la taille et la position du véhicule dans l'image, en supposant que

les paramètres de la caméra sont connus et que le véhicule se déplace sur une route plane. L'utilisation de cette connaissance a priori évite par exemple de détecter des véhicules à des endroits où il est peu probable d'en trouver (par exemple le haut de l'image).

Le troisième facteur de 2.10 est la probabilité que les sous-images représentent le véhicule (indice a pour « aspect »). La connaissance a priori sur l'aspect des véhicules est utilisée pour entraîner des classeurs faibles $\{h_j(I)\}$ avec l'algorithme AdaBoost pour distinguer les véhicules (classe $y = +1$) des non-véhicules (classe $y = -1$). Le classeur résultant est une combinaison linéaire de ces classeurs,

$$H(I) = \text{sign}\,(f(I)) \quad \text{où} \quad f(I) = \sum_j \alpha_j\, h_j(I) \quad \text{et} \quad h_j(I) \in \{+1, -1\}. \tag{2.11}$$

Il a été montré [FHT98] que la probabilité a posteriori pouvait être déduite de la réponse $f(I)$,

$$P_a(y = 1 \mid I) = \frac{e^{f(I)}}{e^{-f(I)} + e^{f(I)}}. \tag{2.12}$$

Le troisième facteur $\prod_{k=1}^{m} P_a\,(I_k(\mathbf{x}_k, \mathbf{s}_k) \in \text{véhicule})$ peut donc être évalué facilement.

L'utilisation de la fusion de connaissances est illustrée sur la figure 2.8. Pour

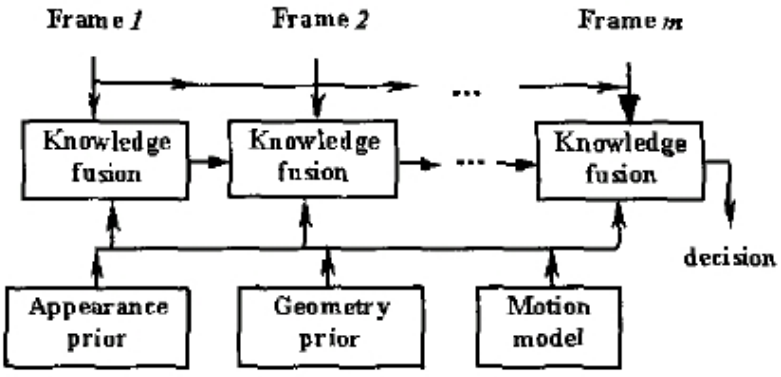


FIG. 2.8 – Fusion de connaissances sur la géométrie de la scène, l'aspect et le mouvement des véhicules. [ZCR+05]

détecter un véhicule, le système commence par utiliser les modèles d'aspect P_a et de géométrie de la scène p_g. Ensuite, en utilisant le modèle de mouvement p_m, le véhicule est suivi. Après quelques images, par exemple une dizaine, la trajectoire est considérée comme étant celle d'un véhicule ou non. Si oui, le véhicule continue d'être suivi par le système.

2.3 Suivi des signaux

Suivre les signaux routiers que l'on a détectés une première fois est une opération indispensable pour notre application. Chaque signal présent dans la scène doit générer une et une seule entrée dans l'inventaire. Il faut donc que nous puissions associer entre elles les détections successives dans le flux vidéo qui correspondent au même signal. Pour cela, nous devons au moins être capable de prédire la position (x_k, y_k) du signal dans la k^{e} image étant donné les postions précédentes. De cette façon, si un signal proche de la prédiction est détecté dans la k^{e} image, nous pouvons considérer que c'est le même signal.

La plupart des travaux relatifs à la détection et à la reconnaissance de signaux routiers dans un flux vidéo décrivent très peu, voire pas du tout, la manière dont les signaux sont suivis. Nous ne parlerons ici que de quelques méthodes qui ont été appliquées aux suivis de signaux routiers.

2.3.1 Localisation 2D des signaux

Miura et al. [MKS00] se placent dans la situation simple où le véhicule se déplace à vitesse constante sur une route plane et rectiligne (figure 2.9). Notons f la

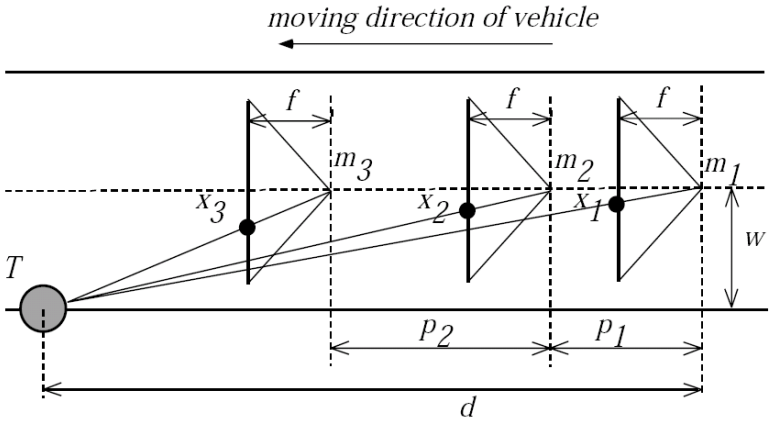


Fig. 2.9 – Prédiction du mouvement sur le plan image d'une cible fixe lorsque le véhicule se déplace à vitesse constante vers cette cible. [MKS00]

distance focale de la caméra, m_k la position de la caméra à l'instant t_k, x_k la position horizontale de la cible sur le plan image à l'instant t_k, $p_k = m_{k+1} - m_k$ la distance entre deux positions successives de la caméra. Le véhicule se déplace à une vitesse constante v et la cible se trouve à une distance d de la caméra à l'instant t_1 et en retrait par rapport à l'axe optique d'une distance w. La similarité des triangles sur

la figure permet d'écrire

$$\frac{f}{x_1} = \frac{d}{w}, \tag{2.13}$$

$$\frac{f}{x_2} = \frac{d-p_1}{w}, \tag{2.14}$$

$$\frac{f}{x_3} = \frac{d-(p_1+p_2)}{w}. \tag{2.15}$$

La résolution de ce système d'équations fournit

$$x_3 = \frac{p_1 x_1 x_2}{(p_1+p_2)x_1 - p_2 x_2} = \frac{x_1 x_2}{(1+\alpha)x_1 - \alpha x_2}, \tag{2.16}$$

où $\alpha = p_2/p_1 = [v(t_3-t_2)]/[v(t_2-t_1)] = (t_3-t_2)/(t_2-t_1)$. La position de la cible sur le plan image peut donc être prédite sans la connaissance de f ni de v mais seulement en connaissant deux positions et instants antérieurs. Miura et al. ne prétendent pas suivre les signaux en usant uniquement de cette technique mais l'utilisent pour capturer une image haute résolution du signal routier à l'aide d'une seconde caméra à téléobjectif[9] : la première caméra détecte le signal et la seconde est dirigée dans la bonne direction grâce à la position prédite. Il faut un certain temps pour diriger correctement la caméra, d'où l'intérêt de la prédiction.

Fang et al. [FCF03] utilisent un filtre de Kalman pour suivre les signaux détectés. Un rappel sur le fonctionnement du filtre de Kalman est disponible en annexe A. Ils supposent également que le véhicule se déplace en direction du signal routier à vitesse constante sur une route plane et rectiligne.

Prédiction de la taille du signal. Comme sur la figure 2.10, notons R_r le rayon du signal circulaire suivi, f la distance focale de la caméra, et v la vitesse du véhicule. $r(t)$ et $r(t+1)$ représentent la projection du rayon du signal sur le plan image aux temps t et $t+1$. $d(t)$ et $d(t+1)$ sont les distances entre le signal et la caméra (dans le sens du déplacement) aux temps t et $t+1$. En se référant à la figure nous pouvons écrire

$$\begin{aligned} \frac{R_r}{r(t+1)} &= \frac{d(t+1)}{f} \\ \frac{R_r}{r(t)} &= \frac{d(t)}{f} = \frac{d(t+1)+v\Delta t}{f}. \end{aligned} \tag{2.17}$$

De ces équations, on tire

$$r(t+1) = \frac{R_r f r(t)}{R_r f - r(t) v \Delta t}. \tag{2.18}$$

La taille du rayon sur le plan image à l'instant $t+1$ peut donc être prédite à partir de sa taille sur l'image précédente à condition de connaître R_r, f et v.

[9] Un téléobjectif est un objectif à longue focale permettant un cadrage beaucoup plus serré grâce à un angle de vision étroit.

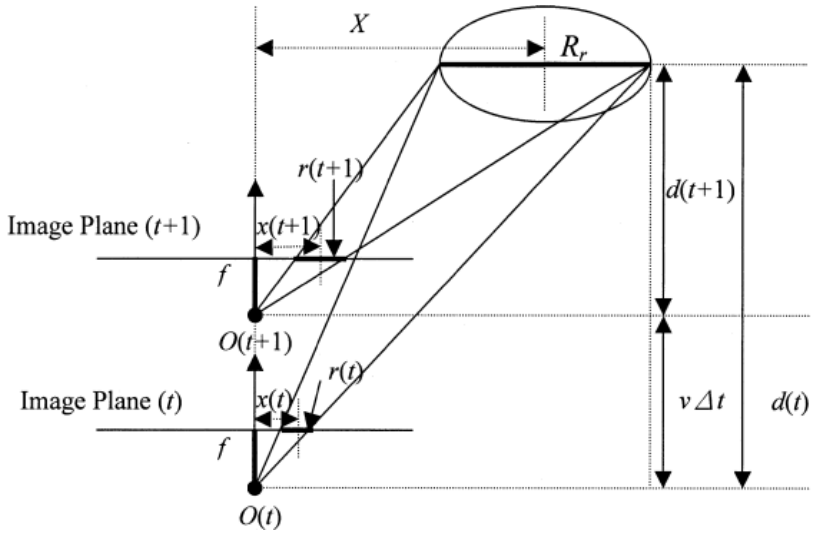


FIG. 2.10 – Prédiction du mouvement sur le plan image d'une cible fixe lorsque le véhicule se déplace à vitesse constante vers cette cible. [FCF03]

Prédiction de la position du signal. Notons X la distance latérale entre le signal et la caméra et $x(t)$ et $x(t+1)$ les distances horizontales sur le plan image entre le signal et le centre de l'image aux instants t et $t+1$. On a alors

$$\begin{aligned} \frac{X}{x(t+1)} &= \frac{d(t+1)}{f} \\ \frac{X}{x(t)} &= \frac{d(t)}{f} = \frac{d(t+1)+v\Delta t}{f}. \end{aligned} \tag{2.19}$$

On tire de 2.17 et 2.19

$$x(t+1) = \frac{r(t+1)}{r(t)}x(t) = \frac{R_r f x(t)}{R_r f - r(t)v\Delta t}. \tag{2.20}$$

Modèle d'état. Le modèle d'état utilisé pour le filtrage KALMAN est le suivant :

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ 1/r_k \\ 1 \end{bmatrix} \tag{2.21}$$

$$\mathbf{z}_k = \begin{bmatrix} x_k \\ y_k \\ 1/r_k \\ 1 \end{bmatrix} \tag{2.22}$$

$$\boldsymbol{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

$$\boldsymbol{A}_k = \begin{bmatrix} \alpha_k & 0 & 0 & 0 \\ 0 & \alpha_k & 0 & 0 \\ 0 & 0 & 1 & -\beta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{avec} \quad \begin{cases} \beta = \dfrac{v\Delta t}{R_r f} \\ \alpha_k = \dfrac{1}{1 - \beta\, r_{k-1}} \end{cases} \tag{2.24}$$

Remarquons que la matrice de transition d'état dépend de la variable d'état r_k. L'équation aux différences n'est donc pas linéaire. De plus, nous devons connaître R_r, f et v afin de prédire l'état suivant, ce qui nous semble assez contraignant par rapport à la méthode proposée par Miura et al. [MKS00].

2.3.2 Localisation 3D des signaux

Arnoul et al. [AVGM96] ont monté une caméra sur un véhicule équipé d'un odomètre et de capteurs gyroscopiques, ce qui permet de connaître le déplacement de la caméra entre deux acquisitions d'image. Comme le signal routier est un objet statique dans la scène, il est donc possible de le localiser en trois dimensions. La position 3D du signal est estimée à l'aide d'un filtre de Kalman étendu (EKF [10]) qui permet de traiter des processus non linéaires. Un rappel sur le fonctionnement du filtre de Kalman étendu est disponible en annexe B.

La caméra est modélisée par une projection perspective (figure 2.11). Les axes x_p

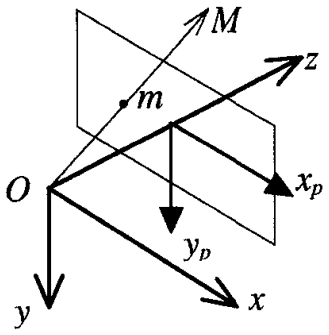


Fig. 2.11 – Modèle sténopé d'une caméra (*pinhole model*). [AVGM96]

et y_p forment le plan image, le point O est le centre optique, et l'axe Oz l'axe optique qui est perpendiculaire au plan image. La distance entre le centre optique et le plan image est la distance focale f. La projection d'un point 3D $\mathbf{x} = [x, y, z]^T$ exprimé dans le repère de la caméra est un point $\mathbf{x}_p = [x_p, y_p]^T$ sur le plan image dont les coordonnées sont :

$$\mathbf{x}_p = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{bmatrix}. \tag{2.25}$$

Le mouvement de la caméra est modélisé par une rotation $\boldsymbol{R}$ (matrice de rotation) autour du centre optique suivi d'une translation $\mathbf{t} = [t_x, t_y, t_z]^T$. À la suite d'un tel

[10] EKF : *extended* Kalman *Filter*.

mouvement, la position 3D de la cible dans le repère de la caméra devient

$$\begin{aligned} \mathbf{x}' &= \boldsymbol{R}^T\mathbf{x} - \boldsymbol{R}^T\mathbf{t} && (2.26)\\ &= Rot(\mathbf{r})^T\mathbf{x} - Rot(\mathbf{r})^T\mathbf{t} && (2.27)\end{aligned}$$

où $Rot(\cdot)$ est la fonction (non linéaire) qui retourne la matrice $\boldsymbol{R}$ qui correspond à une rotation d'un angle $\theta = \|\mathbf{r}\|$ autour du vecteur $\mathbf{r}$.

Nous avons maintenant tout ce qu'il faut pour décrire le modèle d'état. Le vecteur d'état est la position 3D de la cible,

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}. \tag{2.28}$$

Le vecteur de commande contient l'information relative au déplacement de la caméra,

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{r_k} \\ \mathbf{t_k} \end{bmatrix}. \tag{2.29}$$

L'équation d'état est

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) && (2.30)\\ &= Rot(\mathbf{r}_k)^T\mathbf{x}_k - Rot(\mathbf{r}_k)^T\mathbf{t}_k + \tilde{\mathbf{w}}_k && (2.31)\end{aligned}$$

où $\mathbf{w}_k$ représente le bruit sur la commande $\mathbf{u}_k$ fournie par les capteurs, qui devient un autre bruit $\tilde{\mathbf{w}}_k$ après être passé par la fonction $\mathbf{f}$. Ce bruit peut être vu comme le bruit du processus à estimer.

L'équation de la mesure est

$$\begin{aligned} \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) && (2.32)\\ &= \begin{bmatrix} f\frac{x_k}{z_k} \\ f\frac{y_k}{z_k} \end{bmatrix} + \mathbf{v}_k. && (2.33)\end{aligned}$$

L'inconvénient de ce modèle est que le mouvement de la caméra et la distance focale doivent être connus.

2.4 Reconnaissance des signaux

La phase de reconnaissance consiste à assigner une classe à chaque signal détecté. Généralement, les signaux de formes différentes ont été dissociés lors de la phase de détection. Les réseaux de neurones artificiels tiennent une place importante dans les travaux réalisés sur la reconnaissance de signaux routiers. À côté de cela, de nombreuses autres méthodes de classification existent. Nous en présentons certaines, la plupart ayant déjà été utilisées pour la reconnaissance des signaux.

2.4.1 Réseaux de neurones

Yang et al. [YLLH03] utilisent des réseaux de neurones pour la reconnaissance. Les meilleurs taux de détection obtenus sont proches de 70 %. Comme entrée aux réseaux de neurones, ils fournissent les valeurs des pixels et les coefficients de la transformée en cosinus discrète DCT [11] des images. Comme traitement préalable à l'image, les auteurs sélectionnent le pictogramme et font une normalisation en luminance.

Torresen et Sekanina [TBS04] utilisent un perceptron multicouches MLP [12]. Ils se concentrent sur les signaux de limitation de vitesse. Pour déterminer la vitesse indiquée par le signal, les auteurs ne prennent en considération que le premier chiffre du signal (le deuxième étant toujours 0). L'intérieur du signal a été segmenté et les pixels des chiffres sont dissociés des pixels du fond. Il est donc facile de trouver la région d'intérêt contenant le premier chiffre. Cette région d'intérêt est remise à une échelle 7×5 pixels (voir figure 2.12). Ces petites images sont présentées à l'entrée d'un réseau de neurones *feed-forward* entraîné avec l'algorithme de rétropropagation du gradient. Le réseau de neurones possède 3 couches. La couche d'entrée contient 35 (7×5) neurones, la couche cachée en contient 35 et 6 neurones forment la couche de sortie (un pour chaque chiffre présent sur les signaux).

Fig. 2.12 – Exemples de chiffres de taille 7x5 utilisés pour lors de la reconnaissance des signaux de limitation de vitesse. Ces images de 35 pixels sont fournies à l'entrée du réseau de neurones. [TBS04]

Franke et al. [VPH01, chapitre 6] utilisent des fonctions à base radiale RBF [13]. On utilise les réseaux RBF à simple couche, car ils ne nécessitent pas les longs apprentissages des MLP. Les neurones des RBF utilisent généralement des fonctions d'activation gaussiennes.

Vitabile et al. [VPPS01] utilisent un MLP. Les signaux de même forme ont été précédemment groupés. Ils travaillent en RGB. Pour chaque canal, l'image est divisée en blocs de 3×3 pixels. Les entrées du réseau sont les valeurs moyennes des blocs. Les taux de reconnaissance obtenus sont supérieurs à 84 %.

Escalera et al. [dlEM97] testent également un MLP (à 3 ou 4 couches). Ils fournissent directement les valeurs des pixels au MLP. Les images sont remises à une résolution de 30×30 pixels qui correspond à la taille de la couche d'entrée du MLP. La couche de sortie comporte 1 neurone de plus que le nombre de classes des signaux, pour pouvoir indiquer que l'image ne contient pas de signal connu. Pour

[11] *Discrete Cosine Transform.*
[12] *Multilayer perceptron.*
[13] *Radial Basis Function.*

entraîner le MLP, les auteurs utilisent un ensemble d'apprentissage basé sur des images de signaux idéaux (schémas). Ils construisent leur ensemble d'apprentissage en modifiant ces images par :

- des rotations ;
- ajout de bruit gaussien ;
- seuillage afin d'obtenir une information localisée dans la partie intérieure du signal.

Dans d'autres travaux [dlEAM03], les mêmes auteurs utilisent un réseau de neurones ART1. Ce réseau est capable de développer un *clustering* stable à partir de séquences d'entrées arbitraires en s'auto-organisant. Grâce à cela, lorsque de nouvelles classes de signaux lui sont présentées, il ne nécessite pas de nouvel entraînement.

Les réseaux ART [14] sont des réseaux à apprentissage par compétition, dont le problème majeur est le dilemme « stabilité/plasticité ». Afin d'assurer la stabilité, on doit faire tendre le coefficient d'apprentissage vers zéro, mais le réseau perd alors toute sa plasticité. Les réseaux ART permettent de contourner ce problème. Les vecteurs de poids n'y seront adaptés que si l'entrée fournie est semblable à un prototype déjà connu par le réseau. À ce moment, on utilisera le terme : résonance. Par contre, si l'entrée est trop différente des prototypes existants, une nouvelle catégorie va se créer, avec pour prototype, l'entrée qui a engendré sa création. Il y a deux principaux types de réseaux ART : les ART-1 pour des entrées binaires et les ART-2 pour des entrées continues.

Fang et al. [FFY+04] utilisent deux types de réseaux de neurones : un réseau CART [15] et un réseau CHAM [16]. Le premier permet de séparer les signaux en catégories et le second classe les signaux à l'intérieur des différentes catégories.

Franke et al. [VPH01, chapitre 6] emploient un réseau ATDNN [17] qui permet de classer un objet non pas sur base d'une image, mais sur base d'une série temporelle d'image de cet objet. L'architecture de ce type de réseaux est montré sur la figure 2.13.

Pour conclure cette section sur les réseaux de neurones, remarquons que ceux-ci sont couramment utilisés pour la reconnaissance des signaux. Les réseaux de neurones sont un outil puissant pour la classification des objets. Cependant, il faut une certaine expérience pour les construire et les entraîner.

2.4.2 Corrélation normalisée

Miura et al. [MKS00] utilisent une méthode de *pattern matching* basée sur la corrélation normalisée. La corrélation normalisée possède l'avantage d'être robuste

[14] *Adaptive Resonance Theory* [CG83].
[15] *Configurable Adaptive Resonance Theory*.
[16] *Configurable Heteroassociative Memory*.
[17] *Adaptive Time Delay Neural Network*.

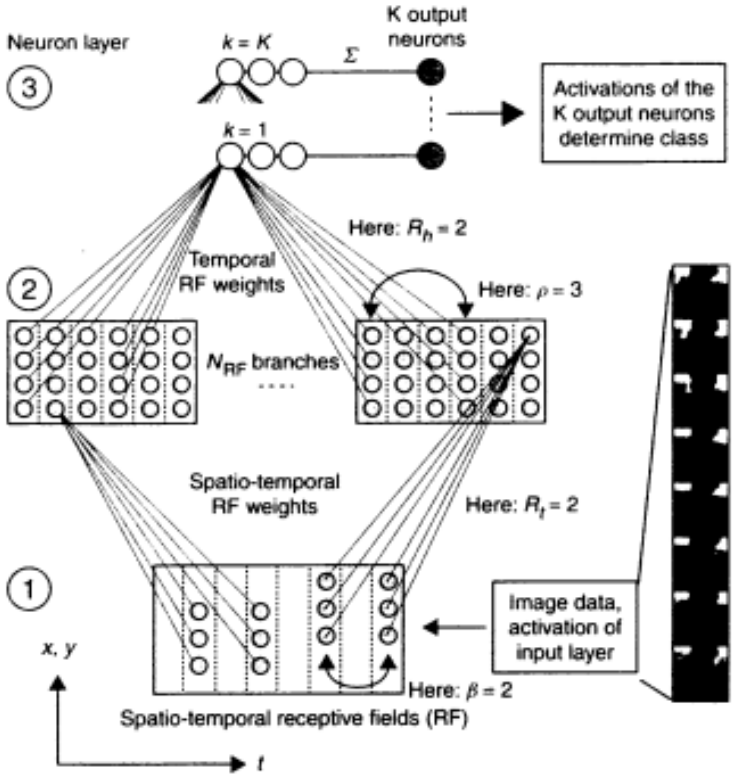


FIG. 2.13 – Architecture du réseau de neurones adaptatif à temps de retard utilisé dans [VPH01].

par rapport aux variations d'illumination, ce qui est intéressant lorsque l'on travaille en extérieur.

2.4.3 Classeurs polynomiaux

FRANKE et al. [VPH01, chapitre 6] proposent de nombreuses solutions pour la reconnaissance de signaux. Par exemple l'utilisation de classeurs polynomiaux dont le schéma est représenté sur la figure 2.14. La structure fait penser aux réseaux de neurones. Le principe est le suivant. Le classeur transforme l'espace d'attributs d'entrée. Il crée une liste de structure polynomiale en effectuant des produits des attributs d'entrées. La seconde couche est une combinaison linéaire de ces nouveaux attributs définie par les coefficients de la matrice $\boldsymbol{W}$ (voir figure 2.14). La liste de structure polynomiale est choisie par l'utilisateur et les coefficients de la matrice $\boldsymbol{W}$ sont ajusté lors d'un apprentissage.

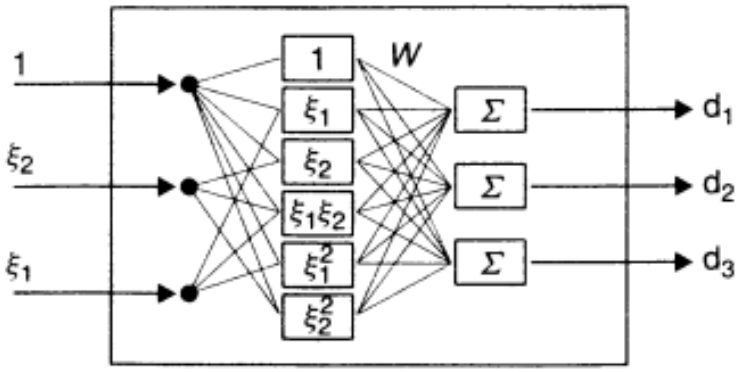


FIG. 2.14 – Architecture d'un classeur polynomial quadratique utilisé dans [VPH01, chapitre 6].

2.4.4 Machine à vecteurs supports

Une autre méthode de classification proposée par FRANKE et al. [VPH01, chapitre 6] : les machines à vecteurs supports SVM [18]. Le principe des SVM est de déterminer une frontière entre les classes, dans un espace d'attributs, en utilisant uniquement les échantillons frontières qui sont en fait les seuls échantillons déterminants (voir figure 2.15). La frontière est fixée de manière à maximiser la marge, c'est-à-dire l'écart entre les vecteurs supports et la frontière.

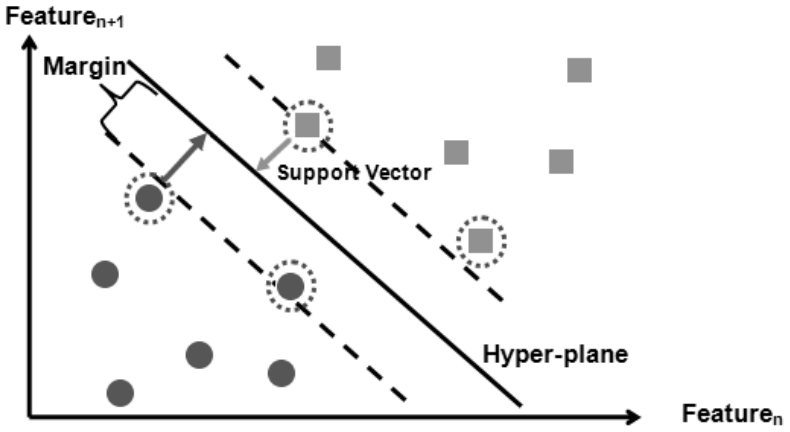


FIG. 2.15 – Principe d'une SVM. Un faible nombre d'échantillon détermine l'allure de la frontière entre les classes. [Equ]

[18] *Support Vector Machine*.

2.4.5 Modèle génératif

Bahlmann et al. [BZR+05] utilisent un modèle génératif, c'est-à-dire qu'ils supposent que les objets de l'ensemble d'apprentissage qu'ils possèdent ont été générés par une certaine distribution de probabilité. L'architecture globale du système est illustrée sur la figure 2.16.

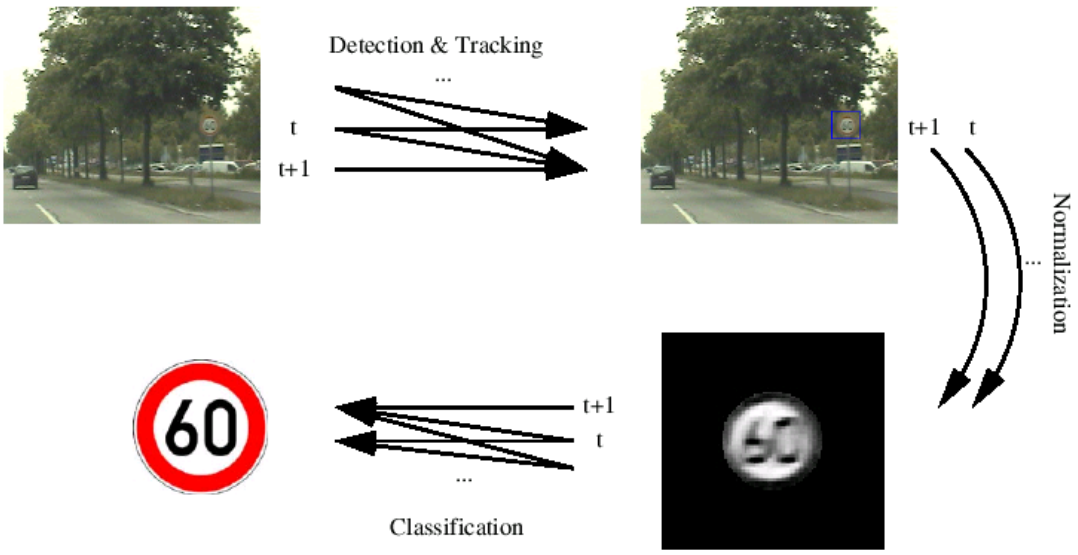


Fig. 2.16 – Architecture du système de reconnaissance utilisé par Bahlmann et al. [BZR+05]. Une cascade de classeurs, entraînés sur des filtres de Haar avec une variante de l'algorithme AdaBoost, détecte les signaux dans chaque image t. Une fois détectés, les objets sont suivis, et les détections individuelles des images $\{1, \ldots, t_0\}$ sont fusionnées pour une détection globale plus robuste. Sur la figure, ceci est représenté par des flèches entrelacées. Ensuite, le signal est masqué par une forme circulaire et normalisé selon la position, l'échelle et la luminosité. Finalement, le signal est classé en maximisant une fonction de vraisemblance en tenant compte de dépendances temporelles.

Entraînement. Avant la modélisation, l'espace des attributs (les pixels en niveaux de gris) est transformé par une analyse discriminante linéaire (LDA [19]). Le nouveau vecteur d'attributs $\mathbf{a} \in \mathcal{R}^{25}$ est constitué des 25 premiers facteurs les plus discriminants de LDA. Ensuite, pour chaque classe $j \in \{1, 2, \ldots, M\}$, ils estiment les paramètres d'une loi normale multivariée et unimodale

$$p(\mathbf{a} \mid y) = \mathcal{N}(\mathbf{a};\, \boldsymbol{\mu}_y,\, \boldsymbol{\Sigma}_y). \tag{2.34}$$

[19] LDA : Linear Discriminant Analysis.

Le classeur est alors entièrement déterminé par M couples de moyenne et de covariance,

$$\left\{(\boldsymbol{\mu}_j,\, \boldsymbol{\Sigma}_j)\right\}_{j=1}^{M}. \tag{2.35}$$

Normalisation. Une fois un signal localisé par le détecteur basé sur des filtres de Haar, il est normalisé :

- une région circulaire est extraite de l'image contenant le signal (seuls les signaux de forme circulaire sont considérés) ;
- l'image est convertie en niveaux de gris, la luminosité est normalisée par une égalisation de l'histogramme ;
- l'image résultante est redimensionnée afin d'être compatible avec le classeur.

Classification. Étant donnée un observation $\mathbf{a}^t(o)$ dans l'image t de la séquence vidéo de test, une approche « maximum de vraisemblance » [20] implique de choisir la classe

$$\hat{y}(o) = \underset{y\,\in\,\{1,\,2,\,\ldots,\,M\}}{\arg\max}\; p(\mathbf{a}^t(o) \mid y) \tag{2.36}$$

$$= \underset{y\,\in\,\{1,\,2,\,\ldots,\,M\}}{\arg\max}\; \log p(\mathbf{a}^t(o) \mid y). \tag{2.37}$$

Afin d'augmenter la robustesse de la classification, celle-ci prend en compte les différentes détections dans t_0 images successives du flux vidéo. Étant donné une suite d'observations supposées indépendantes $\{\mathbf{a}^t\}_{t=1}^{t_0}$, relatives au même signal, la classification est effectuée en maximisant la vraisemblance,

$$\hat{y}(o) = \underset{y\,\in\,\{1,\,2,\,\ldots,\,M\}}{\arg\max} \prod_{t=1}^{t_0} p(\mathbf{a}^t(o) \mid y) \tag{2.38}$$

$$= \underset{y\,\in\,\{1,\,2,\,\ldots,\,M\}}{\arg\max} \sum_{t=1}^{t_0} \log p(\mathbf{a}^t(o) \mid y). \tag{2.39}$$

Comme le signal provenant de l'image 1 est plus petit que celui de l'image t_0 (on suppose que le véhicule avance vers le signal), la classification est améliorée en maximisant une log-vraisemblance pondérée,

$$\hat{y}(o) = \underset{y\,\in\,\{1,\,2,\,\ldots,\,M\}}{\arg\max} \sum_{t=1}^{t_0} \pi_t \log p(\mathbf{a}^t(o) \mid y) \tag{2.40}$$

où les π_t sont des poids croissants,

$$\pi_t = b^{t_0 - t} \quad \text{avec} \quad b < 1. \tag{2.41}$$

[20] L'approche « maximum de vraisemblance » (ML : *Maximum Likelihood*) est équivalente à l'approche « maximum a posteriori » (MAP) si les probabilités a priori $\{p(y)\}_{y=1}^{M}$ sont égales.

Cela permet d'accorder moins de poids à la reconnaissance des premières images qui sont plus petites.

Une mesure de confiance dans la classification est obtenue par la probabilité a posteriori

$$p(y \mid \mathbf{a}^t) = \frac{p(\mathbf{a}^t \mid y)P(y)}{\sum_{j=1}^{M} p(\mathbf{a}^t \mid y = j)P(y = j)}. \tag{2.42}$$

La probabilité a priori $P(y)$ peut être choisie uniforme ou être choisie de façon à refléter l'information que l'on a sur l'environnement (milieu urbain, autoroutes, etc.).

Cette méthode a l'avantage d'être intuitive, conceptuellement simple et de donner de bons résultats. Un classeur entraîné sur 23 classes de signaux circulaires allemands commet 6 % d'erreur sur des images isolées [BZR+05].

2.4.6 Ensembles d'arbres

Marée [Mar05] introduit, dans sa thèse sur la classification automatique d'images, une méthode d'apprentissage supervisé capable de s'appliquer à de nombreux types de problèmes de classification. Elle est en outre très rapide et présente une précision comparable aux meilleurs résultats obtenus dans la littérature.

Partant d'un ensemble d'apprentissage composé d'une série d'images des différentes classes, il sélectionne aléatoirement dans les images des sous-fenêtres afin de se créer un nouvel ensemble d'apprentissage. Sur base de celui-ci, il construit un ensemble d'arbres extrêmement aléatoires [GEW06] basé uniquement sur les valeurs de pixels des sous-fenêtres. Cette méthode possède plusieurs avantages. L'extraction de sous-fenêtres implique une approche locale. Celle-ci permet notamment d'augmenter la taille de l'ensemble, ce qui permet une meilleure généralisation du classeur lorsque l'ensemble de départ est trop petit. De plus, l'approche locale possède une meilleure robustesse qu'une approche globale par rapport aux occlusions et aux changements de point de vue. D'autre part, l'utilisation directe des valeurs des pixels lors la construction de l'ensemble d'arbres permet une conservation de toute l'information des sous-fenêtres sans nécessiter les calculs inhérents à d'autres techniques. Enfin, la construction d'arbres extrêmement aléatoire est très peu gourmande en temps de calcul.

Marée [MGPW05a] précise qu'un meilleur taux d'erreur peut être obtenu en utilisant du *boosting* si l'on accepte de sacrifier la rapidité.

Chapitre 3

Structure générale du programme

Dans ce chapitre, nous décrivons la structure générale de notre programme.

La première question à se poser est de savoir quelles tâches nous allons effectuer en temps réel.

Une première solution est de procéder comme le font les prototypes de systèmes d'aide à la conduite pour lesquels les étapes de détection et de reconnaissance sont réalisées en temps réel. L'inconvénient de cette méthode est qu'elle demande une grande puissance de calcul et des algorithmes rapides qui sont peut-être moins efficaces que d'autres.

Une autre solution, complètement à l'opposé de la première, est d'accomplir toutes les tâches en temps différé. À bord du véhicule, le flux vidéo capturé par la caméra serait entièrement sauvé avec les coordonnées GPS correspondantes, la détection et la reconnaissance se faisant entièrement *off-line*. L'inconvénient de cette méthode est le besoin d'une importante mémoire de stockage.

Une solution intermédiaire est de réaliser la détection des signaux en temps réel et la reconnaissance en temps différé. À bord du véhicule, on sauvegarderait uniquement des suites de sous-images contenant les signaux détectés avec leurs coordonnées GPS. Ce moyen de procéder combine les avantages des deux solutions précédentes, c'est-à-dire une puissance de calcul et une mémoire modérées. C'est sans doute la solution la moins onéreuse si un grand nombre de véhicules devaient être équipés du système. Nous avons donc choisi de procéder de la sorte en séparant totalement les phases de détection et de reconnaissance.

3.1 Programme de détection

Le rôle du programme de détection est de détecter les signaux routiers présents dans la scène et de sauvegarder pour chacun d'eux une suite de sous-images contenant le signal. Le programme a été écrit en C sous Linux en utilisant la bibliothèque de vision OpenCV [Ope]. L'algorithme général est illustré sur la figure 3.1 et est décrit ci-dessous.

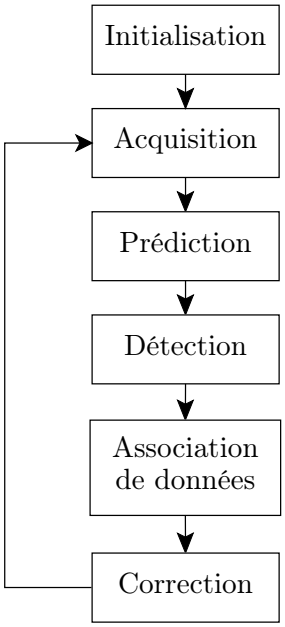


FIG. 3.1 – Structure du programme de détection et de suivi.

Initialisation. L'initialisation de notre programme peut être divisée en deux parties :

1. Initialisation vidéo : une structure d'acquisition est initialisée selon l'origine du flux vidéo, qui peut provenir soit d'un fichier, soit d'une caméra IEEE 1394. La mémoire nécessaire aux différents traitements est allouée de manière statique.
2. Initialisation des structures relatives au suivi : notre programme doit être capable de suivre plusieurs cibles simultanément, cela demande une bonne organisation. Chacun des signaux potentiels possède sa propre structure contenant entre autres :
   - un filtre de KALMAN ;
   - le nombre d'images depuis lequel il est suivi ;
   - le nombre d'images depuis lequel le signal n'a pas été associé à une observation ;
   - une zone mémoire contenant les dernières observations qui ont été associées au signal, ou les dernières prédictions dans le cas où certaines observations sont manquantes.

Acquisition. Une nouvelle image est acquise soit :
- à partir d'un fichier vidéo via la bibliothèque FFmpeg utilisée par OpenCV ;
- depuis une caméra IEEE 1394 via les bibliothèques libraw1394 et libdc1394 utilisées par OpenCV.

Prédiction. Le vecteur d'état de chaque signal actuellement suivi est estimé par son filtre de KALMAN.

Détection. Les signaux routiers sont localisés, dans l'image courante convertie en niveaux de gris, par notre détecteur qui est une cascade de classeurs entraînés sur des filtres de HAAR avec une variante de l'algorithme AdaBoost. Nous éliminons ensuite certaines fausses alarmes en imposant une contrainte sur la couleur.

Association de données. Les observations dans l'image courante sont associées aux signaux suivis en utilisant la méthode du plus proche voisin. Les conditions de validation, d'abandon et de création de pistes sont décrites dans le chapitre relatif au suivi.

Correction. Le filtre de KALMAN de chaque signal actuellement suivi corrige le vecteur d'état en utilisant la mesure courante.

3.2 Programme de reconnaissance

Le rôle du programme de reconnaissance est de classer chacun des signaux représentés par les suites de sous-images obtenues via le programme de détection. Actuellement, le programme n'est pas complètement opérationnel. Nous avons testé diverses méthodes de classification qui sont décrites dans le chapitre relatif à la reconnaissance.

Chapitre 4

Acquisition des images

4.1 La signalisation routière belge

La signalisation routière belge est riche. Elle est composée d'environ 200 signaux routiers. Dans l'arrêté royal du 1er décembre 1975 [Arr75], les signaux routiers sont répartis en 6 catégories :

A. Signaux de danger.

B. Signaux relatifs à la priorité.

C. Signaux d'interdiction.

D. Signaux d'obligation.

E. Signaux relatifs à l'arrêt et au stationnement.

F. Signaux d'indication.

Pour désigner un signal routier en particulier, nous utiliserons le plus souvent le code adopté par l'arrêté royal. Quelques exemples de signaux routiers sont repris sur la figure 4.1. De gauche à droite, on peut voir les signaux A14, B1, C43 et E1. Nous

Fig. 4.1 – Exemples d'images de signaux routiers.

avons travaillé sur une partie de l'ensemble des signaux routiers. Nous avons décidé de nous limiter aux signaux qui comportent une composante rouge au moins sur leur bord et qui sont de forme triangulaire, circulaire ou octogonale. Nous avons choisi ces formes et couleur de signaux, car ce sont les plus représentées sur les routes que nous empruntons quotidiennement. Nous avons tenté d'obtenir un maximum de ces signaux lors de l'acquisition de nos séquences vidéo.

4.2 Matériel utilisé et conditions d'acquisition

Pour l'acquisition, nous avons disposé d'un caméscope DV. Le format DV utilise certains éléments du standard JPEG pour le codage de la vidéo. La compression DV ne joue que sur les redondances spatiales à l'intérieur de l'image, sans chercher à réduire les redondances temporelles comme le fait le MPEG. Le facteur de compression est de 5 : 1. Le signal vidéo est compressé de manière telle qu'il ne nécessite « plus que » 3.2 Mo/s pour être lu. Il faut donc environ 11 Go pour stocker 1 heure de vidéo.

Nous avons installé le caméscope dans une voiture (voir figure 4.2) et enregistré environ 1h30 de séquences vidéo prises dans la région de Liège sur des routes locales et nationales. Il y a une quarantaine de séquences de quelques minutes. Les conditions météorologiques sont variées : ensoleillé, temps couvert et sec, couvert et humide, nuit. Nous avons enregistré des séquences vidéo avec différents zooms afin de déterminer

FIG. 4.2 – Caméscope embarqué.

lequel serait le plus adéquat pour notre travail. Avec un zoom, l'avantage est que les signaux apparaissent avec une taille plus grande sur l'image, ce qui facilite la reconnaissance. D'un autre côté, le suivi du signal au travers de la séquence vidéo est rendu plus difficile, car la vitesse de déplacement du signal entre deux images successives est plus importante. Ceci est aggravé par plusieurs problèmes : la stabilité de l'installation est précaire (figure 4.2) et l'état des routes dans notre région n'est pas irréprochable. Pour ces raisons, nous travaillerons le plus souvent sans zoom.

Les séquences obtenues sont des séquences DV entrelacées. De plus, le flux de données du format DV est très important. Ceci rend les séquences originales difficilement utilisables. Nous désentrelaçons le flux en ne conservant qu'une seule trame sur deux et en dupliquant chaque ligne. Ensuite, nous convertissons les séquences DV au format MPEG-4. La taille des fichiers est ainsi divisée par 7 sans trop de perte de qualité visible. La figure 4.3 illustre la qualité des images obtenues avec les séquences DV originales et les séquences désentrelacées et compressées au format MPEG-4. La qualité de nos séquences vidéo est limitée par un certain nombre de facteurs. La compression MPEG-4 diminue la qualité. Le désentrelacement diminue la résolution : nous travaillons avec séquences 640×480, mais le désentrelacement diminue la résolution verticale réelle par 2. La vitesse du véhicule est parfois importante ce qui

amène du flou dans les images, celles-ci n'étant pas stabilisées. Enfin, les conditions météorologiques variables entraînent une illumination non maîtrisée.

(a) Image extraite d'une séquence vidéo DV entrelacée.

(b) Image extraite d'une séquence vidéo MPEG-4.

FIG. 4.3 – Illustration de la qualité des séquences vidéo. La séquence vidéo MPEG-4 est obtenue à partir de la séquence DV, avec désentrelacement préalable.

Chapitre 5

Détection et localisation des signaux

Nous travaillons, comme beaucoup d'auteurs, uniquement sur les signaux à bords rouges. Nous l'avons vu dans l'étude bibliographique, il est courant d'utiliser pour la détection une segmentation basée sur la couleur. La méthode est intuitive et simple à mettre en place. Dans une première approche, nous réalisons une segmentation basée sur la couleur rouge dans l'espace HSV. Nous tentons ensuite de localiser les signaux parmi les zones segmentées. Les résultats obtenus sont repris en annexe C. Nous ne poursuivrons pas dans cette voix principalement pour une raison : la difficulté de réduire le nombre de fausses détections en milieu urbain. Nous nous tournons donc vers une méthode générale.

La méthode utilisée dans [BZR+05] et [BV04] nous semble intéressante. Elle est basée sur le détecteur introduit par Viola et Jones [VJ01], qui connaît un certain succès depuis quelques années. La méthode est rapide, élégante et générale. De plus, elle est implémentée dans la bibliothèque OpenCV. Nous optons donc pour celle-ci. Les fonctions disponibles dans la bibliothèque OpenCV découlent des travaux effectués dans [LM02] et [LKP03].

Le détecteur introduit par Viola et Jones est en fait une cascade de classeurs. Nous réalisons la détection et la localisation grâce à cette cascade. Ensuite, pour améliorer les performances, nous imposons une contrainte simple sur la répartition spatiale de la couleur dans les fenêtres détectées, pour éliminer celles qui ne contiennent pas de signaux routiers. Nous allons d'abord décrire les principes de construction de la cascade de classeurs dans la section 5.1. Ensuite, nous montrerons comment nous avons procédé en pratique dans la section 5.2. Enfin, dans la section 5.3, nous terminerons par l'explication de la contrainte imposée à la répartition spatiale de la couleur.

5.1 Détection et localisation par cascade de classeurs

Tout d'abord, décrivons l'approche suivie pour la détection et la localisation des signaux dans une image. Nous parcourons l'image avec une fenêtre glissante de taille déterminée. En chaque position de l'image, le détecteur détermine si oui ou non un signal routier est présent dans la fenêtre. Ainsi, la détection et la localisation des signaux sont réalisées en même temps. Les signaux routiers d'une taille voisine de celle de la fenêtre glissante sont détectés. Pour être capable de détecter les signaux routiers aux différentes échelles auxquelles ils apparaissent dans une image, la procédure est répétée pour différentes tailles de la fenêtre glissante.

Le détecteur est une cascade, c'est-à-dire un arbre de décision dégénéré. On propage la fenêtre testée dans la cascade. Si la fenêtre est rejetée par un étage de la cascade, on considère qu'il n'y a pas de signal routier dans la fenêtre. Si celle-ci traverse la cascade, autrement dit si elle est acceptée par tous les étages de la cascade, on considère qu'un signal routier est présent dans la fenêtre. Ceci est illustré sur la figure 5.1. L'intérêt de la cascade est de réduire considérablement le temps de calcul

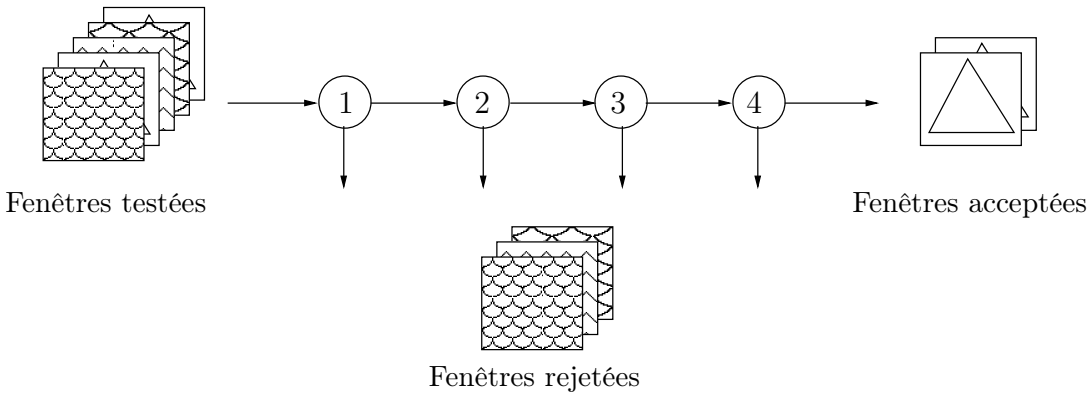


FIG. 5.1 – Détecteur : cascade de classeurs à 4 étages.

et d'augmenter la performance de la détection. Lors de la détection, la plupart des fenêtres ne contenant pas de signal routier sont rejetées dès les premiers étages et demandent peu de temps de calcul. L'essentiel du temps de calcul est consacré aux fenêtres les plus prometteuses qui doivent traverser tous les étages de la cascade.

Nous allons maintenant décrire plus en détails la cascade. Chaque étage de la cascade est un classeur. Chaque classeur est construit par le même algorithme. Nous allons maintenant présenter la construction d'un classeur.

En toute généralité, pour construire un classeur, on se choisit un algorithme de construction de classeur A et un ensemble d'apprentissage $\mathcal{LS}$ (pour *learning set*) composé de N objets :

$$\mathcal{LS} = \{(\mathbf{a}^i, y^i), i = 1, ..., N\} \tag{5.1}$$

où $\mathbf{a}^i$ est un vecteur d'attributs et y^i la classe se rapportant à l'objet o^i. L'algorithme A construit ensuite le classeur $\mathrm{C}_{\mathrm{A}}^{\mathcal{LS}}$ que nous noterons simplement C. Une fois le classeur construit, on peut l'appliquer sur tout vecteur d'attribut $\mathbf{a}(o)$ décrivant un objet o de classe $y(o)$. Le classeur prédit une classe :

$$\hat{y}(o) = \mathrm{C}(\mathbf{a}(o)). \tag{5.2}$$

Idéalement, la classe prédite devrait être correcte pour tout objet, qu'il soit ou non dans l'ensemble d'apprentissage $\mathcal{LS}$: $\hat{y}(o) = y(o), \forall o$.

Dans notre cas, les objets traités sont des images, y prend 2 valeurs pour représenter ces situations : présence ou absence de signal routier. On n'utilisera pas directement la valeur des pixels de l'image comme vecteur d'attributs $\mathbf{a}$, mais des caractéristiques liées aux ondelettes de Haar. Nous décrirons plus loin ces caractéristiques. Leur but principal est de coder une certaine connaissance de ce qu'est un signal routier, connaissance qu'il est difficile d'apprendre à partir d'un ensemble de pixels bruts. La complexité de calcul des caractéristiques est un aspect très important. Nous verrons que les attributs utilisés ici ont l'avantage de posséder un temps de calcul très faible qui correspond à quelques appels dans une *lookup table* et à quelques opérations élémentaires.

La section suivante présente les attributs manipulés par les classeurs de la cascade. Ils sont inspirés des ondelettes de Haar (du nom d'Alfred Haar, mathématicien du début du XX$^{\mathrm{e}}$ siècle).

5.1.1 Des attributs inspirés des ondelettes de Haar

La théorie des ondelettes repose sur des résultats de l'algèbre linéaire et sur la notion d'espace vectoriel. Elle fournit un outil mathématique pour la décomposition hiérarchique d'une fonction (une fonction pouvant être vue comme vecteur d'un espace vectoriel). On peut ainsi décomposer une fonction en une forme grossière et une série de détails de plus en plus fins. Les ondelettes offrent de cette façon une technique élégante pour décrire les niveaux de détails présents dans une fonction : la transformée en ondelettes. Les ondelettes n'ont qu'une influence locale ou finie, à la différence des sinusoïde et cosinusoïde pour la transformée de Fourier. La forme la plus simple d'ondelette est l'ondelette de Haar [1].

Soit $\mathcal{V}$ l'espace vectoriel des fonctions définies sur $[0, 1)$, constantes par morceaux et possédant 4 morceaux de longueur égale . La figure 5.2 représente les 4 fonctions (ou ondelettes) formant une base de Haar pour l'espace vectoriel $\mathcal{V}$. Toute fonction $\mathrm{f} \in \mathcal{V}$ est donc une combinaison linéaire de ces 4 fonctions :

$$\mathrm{f} = c_0\mathrm{b}_0 + c_1\mathrm{b}_1 + c_2\mathrm{b}_2 + c_3\mathrm{b}_3. \tag{5.3}$$

[1] Le lecteur intéressé trouvera des informations sur la théorie des ondelettes de Haar dans [SDS95].

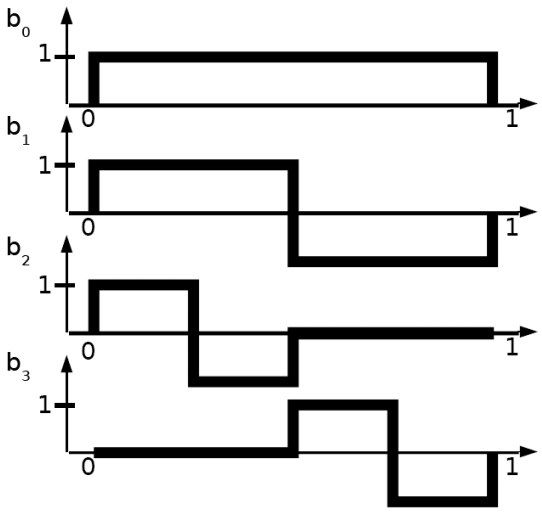


FIG. 5.2 – Ondelettes de HAAR pour l'espace vectoriel $\mathcal{V}$. [SDS95]

Les images pouvant être considérées comme des fonctions de deux variables (ligne et colonne), les ondelettes peuvent également s'appliquer sur celles-ci. On utilise alors des ondelettes à deux dimensions. La figure 5.3 représente les ondelettes qui forment la base de HAAR pour une image de dimension 4×4. On y trouve 16 ondelettes, car l'espace vectoriel est de dimension 16. Le pouvoir de représentation des ondelettes

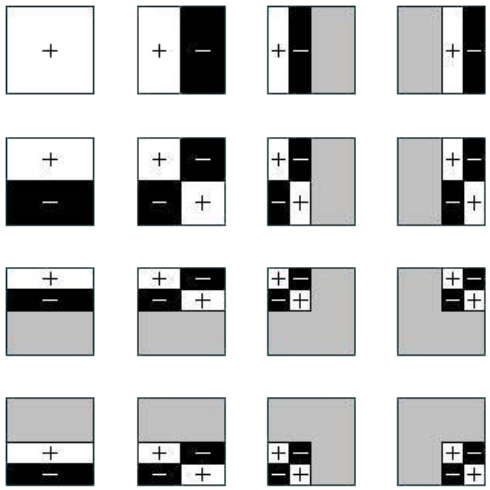


FIG. 5.3 – Ondelettes de HAAR pour l'ensemble des images de taille 4×4. [SDS95]

de HAAR est couramment utilisé pour la compression d'images. Il a aussi été utilisé pour la détection d'objets. La réponse d'une image à une ondelette peut être considérée comme caractéristique de cette image et être utilisée comme attribut par un

algorithme d'apprentissage dans le cadre de la détection d'objets.

Les ondelettes de Haar sont des fonctions linéairement indépendantes. Dans [POP98], Papageorgiou dérive un ensemble de fonctions à partir des ondelettes de Haar. Cet ensemble de fonctions sera étendu successivement dans les travaux de Viola et Jones [VJ01] et de Lienhart et Maydt [LM02]. Ainsi, sont obtenus des ensembles de fonctions linéairement dépendantes. Le terme ondelette n'est dès lors plus approprié puisqu'il est réservé à des fonctions linéairement indépendantes. Par la suite, nous utiliserons le terme filtres de Haar pour parler de ces fonctions. Pour les attributs obtenus, on voit notamment dans la littérature les termes *Haar-like features*, *Haar wavelet features*, nous parlerons de réponses aux filtres de Haar.

La figure 5.4 montre les prototypes de filtres de Haar implémentés dans OpenCV [LKP03]. Les prototypes (1a), (1b) et (4) sont utilisés dans [POP98]. Viola et Jones y ont ajouté les prototypes (2a) et (2c) [VJ01]. Les autres prototypes, ajoutés par Lienhart et Maydt dans [LM02], permettent d'augmenter encore le pouvoir de représentation de l'ensemble de filtres de Haar.

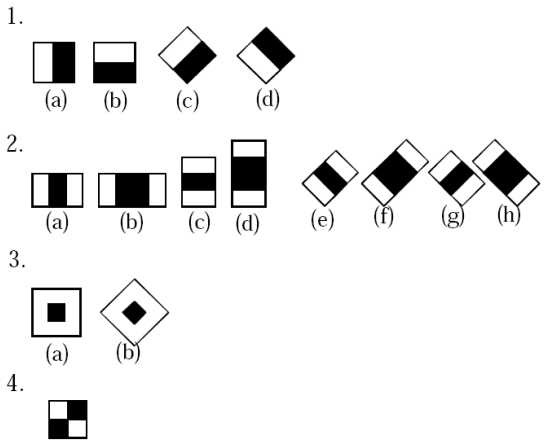


Fig. 5.4 – Prototypes de filtres de Haar implémentés dans OpenCV. [LKP03]

Pour la détection, nous travaillons dans des fenêtres de taille fixe $W \times H$. À partir des prototypes de filtres de Haar de la figure 5.4, on génère un très grand nombre de filtres, environ 800 000 pour une fenêtre 32×32, en faisant varier le prototype, sa position (a, b) dans la fenêtre et sa taille (w, h), comme représenté sur la figure 5.5. Comme cela a été déjà dit, la détection est réalisée à l'aide d'un classeur C. Chaque image I de taille $W \times H$ présentée au classeur est décrite par un vecteur d'attributs $\mathbf{a}$. Chaque attribut de ce vecteur est une réponse de l'image I à un des filtres de Haar.

Pour obtenir un attribut, on réalise le produit scalaire entre l'image et le filtre de Haar. Pour illustrer, la réponse d'une image à un filtre de prototype (1) est la différence entre la somme des pixels de l'image recouverts par le rectangle blanc et la somme des pixels de l'image recouverts par le rectangle noir. Pour les autres types de filtres, on pondère cette différence de manière à compenser les différences de taille entre les zones noires et blanches.

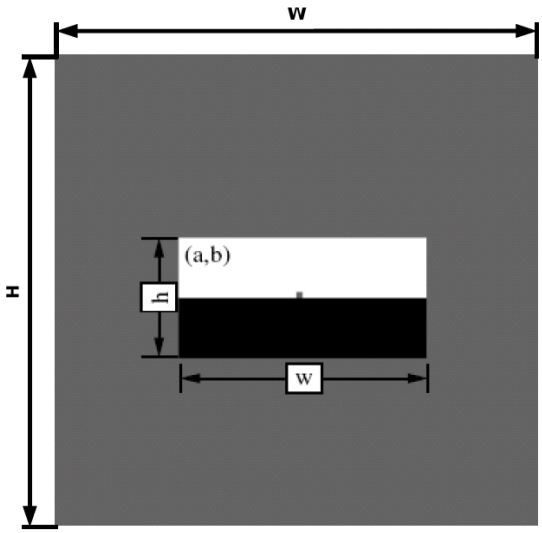


FIG. 5.5 – Exemple de filtre de HAAR. [BZR+05]

5.1.2 Calcul des attributs

L'utilisation de dizaines, voire de centaines de milliers d'attributs nécessite un calcul extrêmement rapide de ceux-ci. Dans ce but, VIOLA et JONES ont introduit l'image intégrale [VJ01]. La valeur de l'image intégrale en (x,y), est la somme de tous les pixels situés à la gauche et au dessus de (x,y), ce pixel inclus :

$$\mathrm{II}(x,y) = \sum_{x' \leq x \ , \ y' \leq y} \mathrm{I}(x',y'), \tag{5.4}$$

où $\mathrm{II}(x,y)$ est l'image intégrale et $\mathrm{I}(x,y)$ l'image de départ (voir figure 5.6 (a)). L'image intégrale peut être calculée en une seule passe sur l'image de départ grâce à deux relations de récurrence [VJ01]. Grâce à l'image intégrale, la somme des pixels d'un rectangle de n'importe quelles taille et position peut être calculée à partir de 4 valeurs prises dans l'image intégrale. Pour calculer S, la somme des pixels dans le rectangle $ABCD$ (figure 5.6 (c)), on utilise simplement la relation :

$$S = \mathrm{II}(A) + \mathrm{II}(D) - \mathrm{II}(B) - \mathrm{II}(C). \tag{5.5}$$

Tous les filtres de HAAR, sauf le filtre de prototype (4) (voir figure 5.4), sont constitués de deux rectangles. Grâce à l'image intégrale, on peut calculer l'attribut correspondant à un filtre de HAAR très rapidement. Par exemple, pour un filtre de prototype (1a), il suffit de 8 appels dans une *lookup table* et de 7 additions ou soustractions ! C'est cette rapidité de calcul qui autorise le nombre énorme de filtres calculés.

Pour les filtres orientés à 45°, une seconde image intégrale $\mathrm{II}_R(x,y)$ est introduite dans [LKP03]. Elle est définie comme la somme des pixels d'un rectangle orienté à

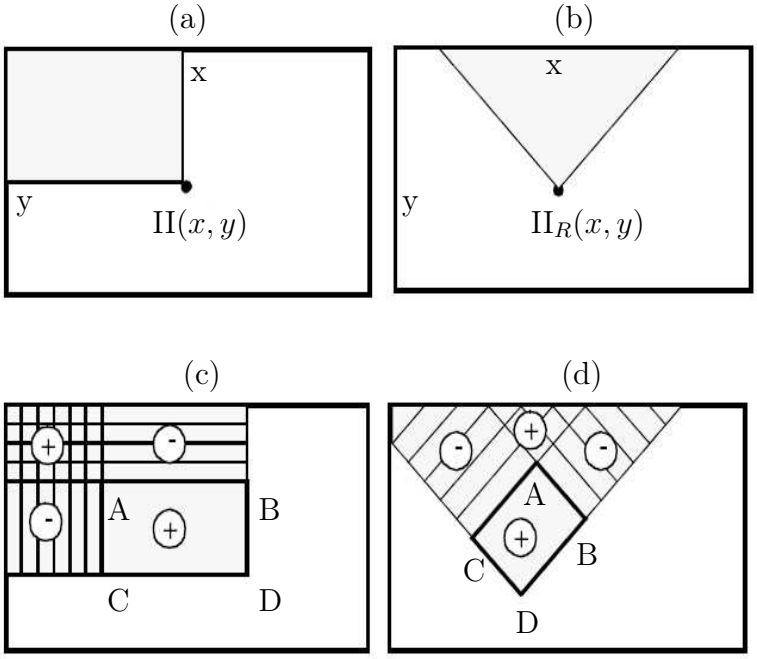


FIG. 5.6 – Utilisation de l'image intégrale pour le calcul de la somme des pixels d'un rectangle. [LKP03]

45° avec le coin inférieur en (x, y) et s'étendant jusqu'au bord supérieur de l'image de départ $\mathrm{I}(x, y)$:

$$\mathrm{II}_R(x, y) = \sum_{y' \leq y \ , \ y' \leq y - |x - x'|} \mathrm{I}(x', y'). \tag{5.6}$$

Elle est illustrée sur la figure 5.6 (b). Le principe de calcul de la somme des pixels d'un rectangle orienté à 45° (figure 5.6 (d)) est identique à celui décrit pour un rectangle non orienté.

5.1.3 Entraînement d'un classeur

Après avoir introduit les attributs utilisés par les classeurs de la cascade, nous décrivons maintenant l'entraînement d'un étage particulier de la cascade. Rappelons qu'un étage de la cascade correspond à un classeur.

Nous venons de voir comment nous générons un grand nombre de filtres de HAAR et comment les attributs d'une image, c'est-à-dire les réponses de l'image à ces filtres, sont calculés. Bien que le calcul des attributs soit extrêmement rapide, il n'est pas imaginable, pour la phase de détection, de les calculer tous. Aussi, l'algorithme d'apprentissage va construire un classeur et par la même occasion, sélectionner les attributs les plus discriminants.

On doit fournir à l'algorithme un ensemble d'apprentissage $\mathcal{LS}$ contenant des exemples d'objets des deux classes avec lesquelles nous travaillons : des images de signaux routiers et des images ne contenant pas de signaux routiers. Nous parlerons d'exemples positifs pour les premières et d'exemples négatifs pour les dernières.

Pour l'apprentissage du classeur, on utilise le *boosting*. C'est une méthode de construction et de combinaison de classeurs. Elle repose sur un résultat théorique fort : partant d'un ensemble de classeurs qui ont une probabilité supérieure au hasard de prédire correctement la classe d'une donnée, on peut construire, en combinant ces classeurs de base, un nouveau classeur dont la probabilité de réussite peut être rendue arbitrairement proche de 1. Pour les classeurs ayant une probabilité supérieure au hasard de prédire correctement la classe d'une donnée, on parle dans la littérature anglophone de *weak classifier*. Nous utiliserons le terme classeur faible qui rappelle que ces classeurs, utilisés séparément, ne présentent pas un comportement satisfaisant.

L'algorithme de *boosting* utilisé est une variante de l'algorithme AdaBoost. Le principe de l'algorithme AdaBoost [FS96] est d'associer des poids aux exemples de l'ensemble d'apprentissage ; le poids d'un exemple indique la difficulté de prédire la classe de celui-ci. AdaBoost concentre ses efforts sur les exemples dont il est difficile de prédire la classe. Un premier classeur faible est construit sur l'ensemble des exemples. Les exemples dont la classe est mal prédite ont leur poids qui augmente, les autres ont leur poids qui diminue. Un second classeur est construit sur les exemples pondérés, et ainsi de suite jusqu'à obtenir un certain nombre de classeurs. Pour prédire la classe d'une donnée, chaque classeur vote pour sa prédiction avec d'autant plus de voix que le classeur est bon. La classe prédite est la classe ayant reçu le plus grand nombre de voix [2].

AdaBoost procède par itération. À chaque itération, un classeur faible est construit. Les classeurs faibles utilisés dans OpenCV sont des arbres de décision dégénérés. Il ne possèdent qu'un nœud de test, on parle de souche. Une souche classe un objet en testant la valeur d'un de ses attributs par rapport à un seuil. L'algorithme de construction des classeurs faibles est tel qu'il sélectionne le filtre de Haar qui sépare au mieux les exemples positifs des exemples négatifs en tenant compte du poids des exemples.

A la fin de l'itération, on obtient un classeur C, prenant comme entrée un vecteur d'attribut $\mathbf{a}$ et donnant en sortie une classe définie par :

$$\mathrm{C}(\mathbf{a}) = \mathrm{sign}\Big(\Big(\sum_{t=1}^{T} \alpha_t \ \mathrm{sign}\ (< \mathbf{h}_t, \mathbf{a} > - \theta_t) \Big) \ - s \ \Big). \tag{5.7}$$

T est le nombre de classeurs faibles utilisés, α_t est le poids du t^{e} classeur faible, $\mathbf{h}_t$ est un vecteur dont tous les éléments sont nuls sauf celui correspondant au filtre de Haar sélectionné pour le t^{e} classeur faible, θ_t est le seuil du t^{e} classeur faible. Le produit scalaire est noté $<,>$ et s est le seuil du classeur C.

[2] La description de l'algorithme AdaBoost qui vient d'être réalisée est tirée de [Pre05, pages 122-123].

Nous savons maintenant comment est construit un classeur qui est, rappelons le, un élément du détecteur, le détecteur total étant une cascade de classeurs. Revenons maintenant sur la façon dont est construite cette cascade.

5.1.4 Construction de la cascade de classeurs

Chaque étage de la cascade, c'est-à-dire chaque nœud de l'arbre de décision ou encore chaque classeur, est entraîné pour détecter presque toutes les fenêtres contenant un signal routier, tout en rejetant une certaine partie des autres fenêtres. Chaque étage est un peu plus sélectif que le précédent, ce qui permet un gain de temps lors de la détection, puisque la majorité des fenêtres sont rejetées dès les premiers étages. Pour chaque étage, la tâche est plus ardue que pour l'étage précédent. En effet, il s'agit de trouver des caractéristiques qui dissocient les exemples négatifs et positifs qui n'ont pu être dissociés par l'étage précédent.

Dans notre cas, chaque étage de la cascade est construit pour éliminer au moins 50 % des fenêtres ne contenant pas de signal routier tout en laissant passer au minimum $99,5$ % des fenêtres en contenant un. Nous utilisons une cascade de 14 étages. Théoriquement, on doit dès lors obtenir un pourcentage de fenêtres ne contenant pas de signal routier acceptées de $0,5^{14} = 6 \cdot 10^{-5}$ et un pourcentage de fenêtres contenant un signal routier acceptées de $0,995^{14} = 93,2$ %.

Pour atteindre les taux désirés, on procède à chaque étage de la même manière. Pour entraîner l'étage courant, l'algorithme de construction de la cascade cherche dans l'ensemble d'apprentissage, des exemples négatifs que les étages précédents ont erronément classé comme positifs. La durée de cette recherche augmente exponentiellement avec le nombre d'étages déjà construits, car il est de plus en plus difficile de trouver des exemples négatifs classés comme positifs à mesure que le nombre d'étages augmente. Une recherche similaire est effectuée pour obtenir des exemples positifs, mais celle-ci est très rapide puisque presque tous les exemples positifs traversent les étages de la cascade.

Quand les exemples positifs et négatifs sont trouvés, l'algorithme AdaBoost effectue une boucle. Après chaque itération, on obtient un nouveau classeur faible (une souche). On évalue alors le classeur total de l'étage C sur les exemples positifs suivant l'équation 5.7. Le seuil s du classeur C est ajusté pour obtenir au moins $99,5$ % des exemples positifs classés comme tels. On évalue ensuite C sur les exemples négatifs. Si plus de 50 % de ceux-ci sont classés comme négatifs, la construction du classeur C est terminée et on passe à la construction de l'étage suivant. Dans le cas contraire, on construit un nouveau classeur faible (nouvelle itération).

Nous venons de décrire la méthode de détection et de localisation que nous allons utiliser, il est temps maintenant d'expliquer en pratique comment nous avons procédé.

5.2 Mise en pratique de la détection et de la localisation

Pour mettre en pratique la méthode que nous venons de décrire, il faut tout d'abord constituer un ensemble d'apprentissage $\mathcal{LS}$. Ce n'est pas une chose aisée. La section 5.2.1 détaille comment nous avons procédé. Une fois le $\mathcal{LS}$ constitué, nous pouvons réaliser des entraînements [3] de cascades de classeurs grâce à l'utilitaire d'OpenCV : `haartraining`. Nous décrivons dans la section 5.2.2 la façon de mesurer les performances des cascades obtenues. Ceci nous permet de discuter dans la section 5.2.3 des performances des différentes cascades suivant les paramètres utilisés pour l'entraînement.

5.2.1 Constitution d'un ensemble d'apprentissage

L'algorithme d'apprentissage va construire une cascade de classeurs fonction de l'ensemble d'apprentissage. On parle d'induction : à partir d'un ensemble fini d'exemples, on essaie d'induire une règle générale. Plus l'ensemble d'apprentissage est représentatif du monde réel, plus on a de chances d'obtenir une cascade ayant un bon pouvoir de prédiction, c'est-à-dire qui ne se contente pas de classer correctement les exemples de l'ensemble d'apprentissage, mais qui est également performante sur de nouveaux exemples.

Généralement, plus l'ensemble est grand, plus il devient représentatif et meilleure sera la cascade de classeurs entraînée. Pour les exemples négatifs, il n'y a pas de problème. Nos séquences vidéo regorgent d'images sans signaux routiers où l'on peut tirer aléatoirement des fenêtres de la taille requise pour l'entraînement. Nous pouvons donc obtenir facilement un grand nombre d'exemples négatifs. Pour les exemples positifs, la tâche est plus ardue.

Nous avons cherché sur internet des bases de données publiques d'images de signaux routiers, mais les seules que nous ayons trouvées, comme [Data] et [Datb], sont inadaptées. Elles contiennent très peu de signaux routiers différents. De plus, les signaux présents dans ces bases de données sont rares dans nos séquences vidéos, voire absents de la signalisation routière belge.

Il nous a donc fallu nous concevoir notre propre base de données. Créer une base de données est un travail des plus fastidieux qui peut être réalisé de plusieurs manières :

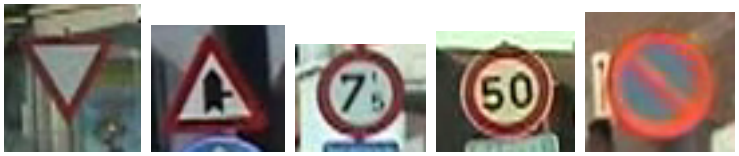
– Prendre des images des schémas disponibles sur internet (par exemple dans [Arr75]). Les images seront-elles vraiment représentatives des signaux routiers que l'on peut rencontrer dans le monde réel ? On doit fournir des images rec-

[3] Nous avons entraîné 57 cascades en faisant varier différents paramètres d'entraînement. Chaque entraînement a duré en moyenne huit heures. Nous avons utilisé des Pentium IV 2.8 GHz. Les entraînements ont été réalisé en partie sur le réseau d'ordinateurs de l'Institut Montefiore.

tangulaires pour l'apprentissage et les signaux sont le plus souvent de forme différente. Que faut-il placer comme fond d'image ?

- Se rendre sur le terrain et prendre des photographies numériques des signaux. Cela nous permettrait d'obtenir des images de qualité pour l'ensemble d'apprentissage, mais cette collecte manuelle serait très longue et nous permettrait difficilement d'obtenir un grand nombre d'images.
- Se servir des séquences vidéo collectées pour y prélever les images de signaux routiers. Cette méthode a l'avantage d'obtenir des images de cas réels sans nécessiter le déplacement devant chaque signal pour en prendre une photographie. Le point négatif est que les images des séquences vidéos sont de qualité limitée pour toutes les raisons citées dans la section sur l'acquisition des images.

Nous avons opté pour la dernière solution, car elle nous semble être le meilleur compromis entre le temps dépensé, le nombre et la qualité des images obtenues. Nous avons construit une base de données d'environ 1988 images de signaux routiers. La taille minimum des images est de 16×16 pixels. Nous avons également collecté une série d'exemples négatifs : 306 images 640×480 pixels ne contenant pas le moindre signal. Nous montrons des exemples positifs contenus dans notre base de données sur la figure 5.7. Des détails sur la base de données constituée sont disponibles en



(a) (b) (c) (d) (e)

FIG. 5.7 – Images de l'ensemble d'apprentissage (taille réelle).

annexe D.

Après avoir constitué une première base de données, on peut augmenter la taille de celle-ci par plusieurs moyens, par exemple :

- Appliquer des transformations aux images disponibles. On peut penser notamment à des rotations suivant différents axes, à un changement d'échelle, à la modification de l'intensité moyenne, à l'ajout de bruit,... Ceci permettrait de générer un nombre important d'images déformées pour simuler la variété des signaux présents dans le monde réel. Plusieurs questions se posent. Les images générées seront-elles vraiment représentatives des signaux réels ? Lorsqu'on effectue une rotation, l'image obtenue est généralement de taille plus grande que l'image de départ. Que doit-on mettre comme fond dans l'image ?
- Construire un détecteur de signaux sommaire à l'aide des images disponibles. Ensuite, appliquer le détecteur sur des séquences vidéo. On marque les objets détectés comme positifs ou négatifs suivant qu'il s'agit ou non d'images de

signaux routiers. On peut ainsi augmenter l'ensemble d'apprentissage, mais cela nécessite de disposer d'un détecteur de signaux.

Nous n'utiliserons pas ces deux solutions pour agrandir notre base de données. La première, principalement, car nous préférons des exemples réels. La deuxième, car la solution n'est pas plus rapide que la méthode que nous avons utilisée pour récolter les premières images de signaux. De plus, il faudrait tout d'abord créer un détecteur. Enfin, nous pensons que la taille de notre base de données est acceptable pour nos tests.

Nous pourrions obtenir une meilleure base de données en utilisant la technique du *bootstrapping* qui est présentée dans la partie « Améliorations et perspectives ».

Maintenant, nous disposons d'un ensemble d'apprentissage. À partir de celui-ci, nous pouvons entraîner une cascade de classeurs. La section suivante décrit la façon dont les performances d'une cascade sont mesurées. Ensuite, nous pourrons discuter dans la section 5.2.3 des performances d'une cascade suivant les paramètres utilisés pour l'entraînement.

5.2.2 Mesure des performances

Quand on a construit une cascade de classeurs, il est intéressant de tester ses performances. Pour ce faire, il faut se constituer un ensemble de test. Il s'agit d'un ensemble d'images dans lesquelles on a repéré les signaux routiers : leurs position et taille sont connues.

On applique la détection sur les images de l'ensemble de test. Comme nous l'avons vu dans la section 5.1, cela consiste à parcourir l'image avec une fenêtre glissante de taille déterminée. En chaque position de l'image, la fenêtre courante est présentée à la cascade et est rejetée ou acceptée. Les signaux routiers d'une taille voisine de celle de la fenêtre glissante sont détectés. Pour pouvoir détecter les signaux aux différentes échelles auxquelles ils apparaissent dans une image, la procédure est répétée pour différentes tailles de la fenêtre glissante. Les tailles de fenêtres utilisées sont données par :

$$(w \cdot sf^i) \times (h \cdot sf^i) \quad , \quad i = 1, 2, \ldots, N \tag{5.8}$$

$w \times h$ est la taille des images avec lesquelles la cascade a été entraînée[4], sf est le facteur d'échelle[5], sf^N est le facteur d'échelle maximal qui peut être appliqué, il est limité par la taille de l'image. Pour les tests, nous utilisons une valeur de $sf = 1,2$.

Pour mesurer les performances, on ne peut se contenter de vérifier si les objets détectés dans l'ensemble de test correspondent exactement en position et en taille aux signaux marqués. On doit donner une certaine tolérance. Nous allons donner les critères utilisés pour déterminer si un signal est détecté.

[4] La discussion relative à la taille des images pour l'entraînement d'une cascade est développée dans la section 5.2.3.

[5] *Scale factor*.

5.2.2.1 Critères de détection

Les paramètres *maxPosDiff* et *maxSizeDiff* sont respectivement les différences de position et de taille autorisées entre les objets détectés dans une image et les signaux préalablement marqués dans celle-ci. Pour des différences inférieures ou égales, les signaux marqués sont considérés comme détectés. Soient d_1 la longueur de la diagonale du rectangle représentant un signal routier marqué, d_2 la longueur de la diagonale d'un rectangle détecté et d_{12} la distance entre les centres des rectangles (voir figure 5.8). Les conditions requises pour que le signal soit détecté sont :

$$\begin{cases} d_{12} < d_1 \cdot maxPosDiff \\ d_2 > d_1 / maxSizeDiff \\ d_2 < d_1 \cdot maxSizeDiff \end{cases} \tag{5.9}$$

Nous utilisons pour nos tests des valeurs de $1,5$ pour *maxSizeDiff* et de $0,15$ pour

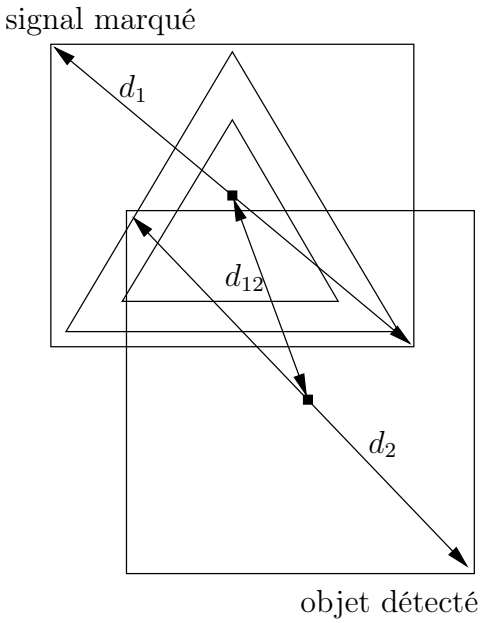


FIG. 5.8 – Conditions pour qu'un signal soit considéré comme détecté.

maxPosDiff.

Généralement, plusieurs détections surviennent à différentes positions et échelles proches de celles du signal routier réel. Les détections multiples sont fusionnées en un seul groupe sous certaines conditions. Deux détections appartiennent au même groupe si (voir figure 5.9)

$$\begin{cases} x_2 \leq x_1 + d \\ x_2 \geq x_1 - d \\ y_2 \leq y_1 + d \\ y_2 \geq y_1 - d \\ w_2 \leq w_1 \cdot 1,2 \\ w_2 \geq w_1 / 1,2 \end{cases} \tag{5.10}$$

où $d = w_1 \cdot 0,2$. Pour la séparation des groupes, un algorithme quadratique décrit dans [CLR90] est utilisé. On associe à chaque groupe un paramètre *neighbors* qui représente le nombre de détections fusionnées dans le groupe.

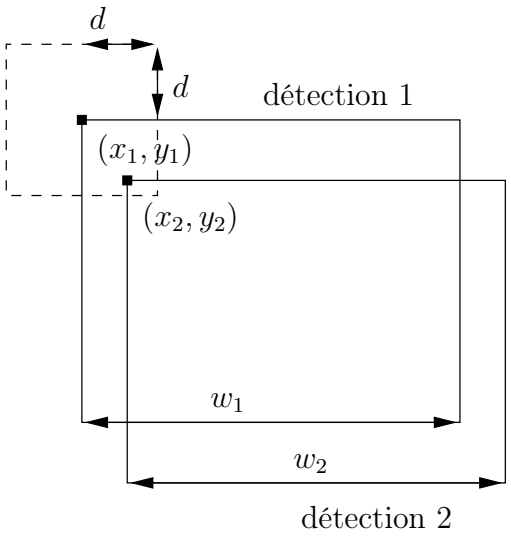


FIG. 5.9 – Conditions de fusion de deux détections.

5.2.2.2 Description des mesures

Pour évaluer les performances de la cascade de classeurs, nous utilisons une série de mesures dont certaines sont définies dans [Pre05, page 32]. Si on sépare l'ensemble des fenêtres testées en deux parties, les fenêtres contenant un signal et les fenêtres n'en contenant pas, autrement dit, les exemples positifs et négatifs, on définit :

- VP : le nombre de vrais positifs : les exemples positifs classés comme positifs.
- VN : le nombre de vrais négatifs : les exemples négatifs classés comme négatifs.
- FP : le nombre de faux positifs : les exemples négatifs classés comme positifs.
- FN : le nombre de faux négatifs : les exemples positifs classés comme négatifs.

On peut définir aussi deux statistiques, la précision et le rappel :

- précision pour les positifs : $\frac{\text{VP}}{\text{VP+FP}}$.
- précision pour les négatifs : $\frac{\text{VN}}{\text{VN+FN}}$.
- rappel pour les positifs : $\frac{\text{VP}}{\text{VP+FN}}$.
- rappel pour les négatifs : $\frac{\text{VN}}{\text{VN+FP}}$.

Intuitivement, la précision mesure la proportion d'exemples vraiment positifs (négatifs) parmi ceux qui sont classés comme positifs (négatifs). Le rappel mesure la proportion d'exemples positifs (négatifs) classés comme tels.

Il est toujours plus pratique de manipuler un seul nombre qui synthétise les autres. On utilise la mesure F, qui est une moyenne harmonique pondérée. Si r est le rappel et p la précision, elle est définie par

$$\mathrm{F}_\alpha = \frac{\alpha+1}{\frac{1}{p}+\frac{\alpha}{r}} = \frac{(\alpha+1)\cdot \mathrm{VP}}{(\alpha+1)\cdot \mathrm{VP} + \alpha \mathrm{FN} + \mathrm{FP}}, \tag{5.11}$$

où le rappel a un poids $\alpha \in [0, +\infty)$ et la précision un poids de 1. Il nous importe d'avoir un bon rappel, c'est à dire de détecter la plupart des signaux. La précision est pour nous moins importante, car nous mettrons en place des moyens pour éliminer les faux positifs. Aussi, nous prenons un α supérieur à 1. Pour $\alpha = 2$, on a

$$\mathrm{F}_2 = \frac{3\mathrm{VP}}{3\mathrm{VP} + 2\mathrm{FN} + \mathrm{FP}}. \tag{5.12}$$

Nous utilisons également les termes suivant :

- TD : le taux de détection : c'est le rappel pour les positifs.
- TFA : le taux de fausses alarmes : $(1 - \text{rappel pour les négatifs}) = \frac{\mathrm{FP}}{\mathrm{VN}+\mathrm{FP}}$.

Nous utilisons un outil supplémentaire pour mesurer les performances d'une cascade de classeurs, la courbe ROC[6]. C'est une courbe du TD en fonction du TFA. L'utilisation des courbes ROC est fréquente pour mesurer les performances d'un détecteur. Nous construisons la courbe ROC d'une cascade de la manière suivante. Nous augmentons progressivement le paramètre *neighbors* requis pour qu'un groupe soit pris en compte dans les TD et TFA. Ainsi, ces taux diminuent peu à peu. Les différentes valeurs obtenues permettent de tracer la courbe ROC.

Un exemple de courbe ROC est montré sur la figure 5.10. La cascade de classeurs idéale est représentée par le point A $(0, 1)$: taux de détection de 1 et taux de fausses alarmes nul. Elle détecte tous les signaux routiers de l'ensemble de test sans fausse détection. Les courbes ROC obtenues n'atteindront pas le point A. Pour quantifier la qualité d'une cascade, on peut mesurer la distance qui sépare sa courbe ROC du point A. Le plus souvent, nous nous cantonnerons cependant à la comparaison des performances entre cascades via leur courbes ROC et nous laisserons de côté l'estimation précise des performances d'une cascade particulière.

Nous venons de décrire les différentes valeurs que nous utilisons pour mesurer les performances d'une cascade de classeurs. Pour obtenir ces valeurs, nous employons l'utilitaire `haarperformance` d'OpenCV. Nous discutons dans la section suivante des performances d'une cascade suivant les paramètres utilisés pour l'entraînement.

[6] *Receiver Operating Characteristics*.

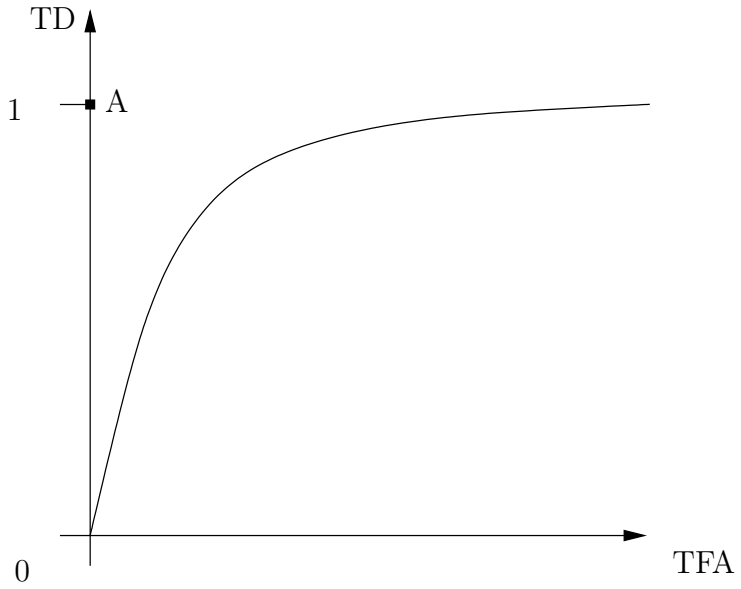


FIG. 5.10 – Exemple de courbe ROC.

5.2.3 Performances d'une cascade selon les paramètres d'entraînement

Bien que la méthode de détection utilisée soit générale, il y a un nombre important de questions à se poser lorsque l'on veut réaliser un entraînement. À ces questions correspondent des paramètres d'entraînement de la cascade de classeurs.

On dénombre notamment les trois paramètres suivants : le nombre d'étages de la cascade, le taux de détection minimum par étage et le taux de fausses alarmes maximum par étage. Comme nous l'avons déjà dit, nous travaillons avec des cascades de 14 étages. Le taux de détection minimum par étage est de $99,5$ % et le taux de fausses alarmes maximum par étage est de 50 %. Théoriquement, ceci nous donne sur l'ensemble de la cascade un taux de fausses alarmes maximum de $0,5^{14} = 0,006$ % et un taux de détection minimum de $0,995^{14} = 93,2$ %. Ces valeurs atteintes lors de la construction de la cascade, sont calculées uniquement sur l'ensemble d'apprentissage. Si celui-ci n'est pas assez représentatif des situations rencontrées dans le monde réel, les valeurs qui viennent d'être citées risquent d'être bien au-dessus des performances réelles de la cascade, ce qui peut être mis en évidence par l'évaluation des performances de la cascade sur un ensemble de test indépendant de l'ensemble d'apprentissage.

Nous l'expliquons dans la section 5.1.4, pour l'entraînement de chaque étage de la cascade, on recherche des exemples positifs et négatifs qui ont traversé tous les étages précédents. Les paramètres *npos* et *nneg* représentent respectivement les nombres d'exemples positifs et négatifs voulus. Nous fixons ces valeurs entre 50 et 2000, selon la taille des ensembles d'apprentissage à notre disposition. Ainsi, à chaque étage,

l'entraînement se déroule sur *npos* exemples positifs et *nneg* exemples négatifs. Les exemples négatifs utilisés diffèrent obligatoirement pour chaque étage.

Nous discutons maintenant de l'influence sur la performance d'une cascade, des paramètres d'entraînement suivants : la taille des images de l'ensemble d'apprentissage, les signaux représentés dans l'ensemble d'apprentissage, le canal couleur des images, la taille de l'échantillon d'apprentissage, le type de classeur faible et enfin l'ensemble de prototypes de filtres de HAAR. Ensuite, nous parlons d'une méthode d'élagage de la cascade qui peut être appliquée après l'entraînement. Enfin, nous tirons certains enseignements.

Dans ce qui suit, sauf explicitement spécifié :

- on travaille avec des images en canal luminance Y, celui-ci est défini par :

$$Y = 0,299\,r + 0,587\,g + 0,114\,b\,; \tag{5.13}$$

- l'ensemble d'apprentissage, tiré de notre base de données (voir annexe D), est constitué de 798 images de signaux triangulaires pour les exemples positifs et de 306 images 640×480 pour les exemples négatifs ;
- l'ensemble de test est constitué de 461 images 640×480 tirées de séquences vidéo différentes de celles utilisées pour la constitution de la base de données présentée en annexe D.

5.2.3.1 Taille des fenêtres pour l'entraînement

Pour l'entraînement, on utilise un ensemble d'images de taille fixe. Avec quelle taille d'image doit-on travailler ? Les signaux routiers apparaissent dans les séquences vidéo dans une gamme de tailles très étendue. Ainsi, lorsque la caméra filme sans zoom, la taille des signaux peut atteindre 100×100. La taille à partir de laquelle les signaux sont reconnaissables à l'œil nu est environ 16×16.

Après un entraînement sur des images de taille donnée, on peut appliquer une cascade de classeurs sur des fenêtres plus grandes, mais pas l'inverse. Lors de l'entraînement, des caractéristiques discriminantes sont trouvées pour séparer les exemples positifs des exemples négatifs. Si on travaille sur des fenêtres de taille plus petite, il est possible que les caractéristiques ne soient plus présentes, car l'information disponible dans une image diminue avec sa taille. D'où l'importance du choix de la taille des exemples lors de l'entraînement.

Nous entraînons des cascades sur des exemples de tailles différentes : 16×16, 24×24, 32×32, 48×48. Pour les exemples positifs, nous utilisons 103 images de signaux triangulaires de notre base de données. Les images sont donc remises à la résolution voulue pour l'entraînement. Les exemples négatifs sont tirés de 66 images de notre base de données.

Les cascades obtenues sont testées sur un ensemble de test composé de 1300 images 640×480. L'ensemble de test est totalement indépendant de l'ensemble d'images

utilisé pour l'apprentissage. Nous obtenons les valeurs suivantes : VP, FN, FP. Nous estimons la valeur de VN par le nombre de fenêtres inspectées NF, qui lui est proche. NF est fonction de la taille des images de test $W \times H$, de la taille initiale à laquelle on fait la détection $w \times h$, du facteur d'échelle sf et de l'espacement horizontal et vertical entre les fenêtres testées, Δx et Δy (en pixels). Si on teste toutes les fenêtres possibles, c'est-à-dire si Δx et Δy sont unitaires, on a sur une image :

$$\sum_{n=0}^{N-1} \Big((W - \lfloor w \cdot sf^n \rfloor + 1) \cdot (H - \lfloor h \cdot sf^n \rfloor + 1) \Big), \tag{5.14}$$

fenêtres testées, avec $N = \min(\lfloor W/w \rfloor, \lfloor H/h \rfloor)$ le nombre d'échelles auxquelles sera effectuée la détection. Ainsi, pour $W \times H = 640 \times 480$, $w \times h = 32 \times 32$, $sf = 1,2$, on aurait $\simeq 2,6 \cdot 10^6$ fenêtres testées sur l'image. Lors de nos tests, Δx et Δy varient suivant plusieurs paramètres, notamment la présence d'objets déjà détectés dans la zone de recherche. Nous avons mesuré le nombre de fenêtres testées pour une image pour différentes valeurs de $w \times h$ (16×16, 24×24, 32×32, 48×48), nous obtenons pour toutes une valeur proche de 700 000 fenêtres testées. À partir des valeurs de VP, FN, FP, VN, nous obtenons les valeurs de TD, de TFA et de la mesure F.

Les résultats des tests des cascades de classeurs sont repris sur le tableau 5.1. Pour chaque cascade, on n'a considéré comme signaux que ceux qui apparaissent avec une taille supérieure ou égale à la taille de la fenêtre pour l'entraînement, ce qui explique les différences entre les VP et FN selon la taille de fenêtre. Les taux de détection obtenus sont éloignés de ceux désirés, mais il nous importe pour le moment de comparer les résultats entre eux. On voit que le meilleur taux de détection est atteint pour des fenêtres 32×32. Le meilleur taux de fausses alarmes est atteint pour des fenêtres 24×24, mais au détriment du taux de détection.

Taille des fenêtres	VP	FN	FP	VN	TD	TFA	Mesure F_2
16×16	22	362	983	9,93E+08	0,057	9,90E-07	0,037
24×24	33	250	220	9,61E+08	0,117	2,29E-07	0,121
32×32	81	174	729	9,28E+08	0,318	7,86E-07	0,184
48×48	14	96	776	8,64E+08	0,127	8,98E-07	0,042

TAB. 5.1 – Comparaison des performances de cascades entraînées avec différentes tailles de fenêtres.

Comme nous l'avons déjà dit (section 5.2.2), nous avons calculé la mesure F en donnant plus de poids au rappel qu'à la précision, car nous diminuerons le nombre de fausses alarmes ultérieurement. Le meilleur score pour la mesure F_2 est obtenu pour des fenêtres 32×32 $(0,184)$, un score un peu moins bon est obtenu pour des fenêtres 24×24 $(0,121)$. Entre ces deux options, notre choix se portera sur les fenêtres 32×32. En plus d'obtenir le meilleur score, la résolution plus grande des fenêtres 32×32 est un atout pour la phase qui succède à la détection, c'est-à-dire la reconnaissance. La reconnaissance nécessite des fenêtres contenant des détails plus fins que pour la

détection. Pour la détection, on doit seulement être capable de dire si oui ou non un signal routier est présent. Pour la reconnaissance, on doit être à même de séparer toutes les classes de signaux routiers. C'est pourquoi, pour la suite, nous choisirons d'entraîner nos cascades de classeurs avec des fenêtres de taille 32×32.

5.2.3.2 Ensemble de signaux détectés

Que doit-on détecter ? Voici la question que nous nous posons ici. On peut entraîner une cascade de classeurs à détecter l'ensemble des signaux disponibles dans notre base de données, mais ce n'est peut-être pas le plus judicieux. Plus l'ensemble des signaux à détecter est grand, plus la variance des images de ces signaux est grande. Cela rend la discrimination entre une image de signal routier et une autre image plus difficile à réaliser pour la cascade de classeurs. C'est pour cela que nous avons réparti notre base de données en 4 catégories :

- signaux triangulaires sur base : exemple voir figure 5.7(b).
- signaux triangulaires sur pointe : exemple voir figure 5.7(a).
- signaux d'interdiction : cercle rouge avec écriture noire sur fond blanc, exemple voir figure 5.7(d).
- signaux relatifs à l'arrêt et au stationnement : cercle rouge avec intérieur rouge et bleu, exemple voir 5.7(e).

Sauf si cela est spécifié, lorsque nous utilisons le terme signaux triangulaires, nous excluons les triangles sur pointes, nous parlons donc des signaux triangulaires sur base.

Pour être capable de détecter tous les signaux en même temps, plusieurs possibilités sont envisageables :

- Entraîner un détecteur sur la base de donnée complète. On obtient la détection la plus rapide, mais la moins performante. La cascade de classeurs a en effet du mal à apprendre un modèle d'objet valable pour tous les types de signaux.
- Entraîner plusieurs détecteurs et les faire fonctionner en parallèle. C'est la solution la plus robuste point de vue détection, mais c'est aussi la solution la plus lente.
- Entraîner un arbre détecteur. Cette solution est introduite dans [LLK03]. L'arbre détecteur est un mélange des deux approches précédentes. Il fusionne les premières étapes réalisées par les détecteurs multiples pour réduire le temps de calcul. Ensuite, il construit des branches spécialisées si cela est bénéfique pour l'efficacité du détecteur. L'arbre détecteur permet un temps de calcul proche du détecteur normal tout en ayant des performances de détection proches de celle des détecteurs en parallèle.

Nous entraînons deux cascades de classeurs. La première sur 103 images de signaux triangulaires, la seconde sur 715 images de signaux majoritairement triangulaires ou circulaires. Les courbes ROC résultant des tests sont montrées sur la figure 5.11.

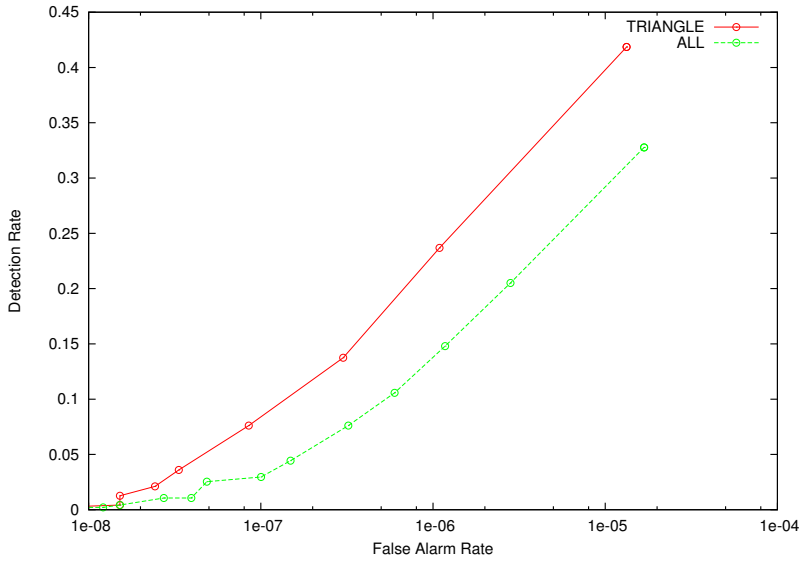


FIG. 5.11 – Courbes ROC de cascades entraînées à détecter des ensembles de signaux différents. Pour la cascade correspondant à la courbe ROC rouge, seuls des signaux triangulaires sont présents dans l'ensemble d'apprentissage. Pour la cascade correspondant à la courbe ROC verte, tous les types de signaux présents dans la base de données ont été inclus dans l'ensemble d'apprentissage.

On peut observer que les performances sont meilleures quand la cascade de classeurs est construite pour détecter uniquement des signaux triangulaires. Nous avons également entraîné des cascades pour qu'elles détectent deux types de signaux : triangulaires sur base et d'interdiction. Il n'est pas évident de déterminer la diminution des performances lorsqu'on entraîne à détecter différentes catégories de signaux. Ces observations pourraient faire l'objet de travaux futurs.

Pour obtenir les meilleurs résultats, nous travaillerons par la suite avec des cascades de classeurs entraînées pour détecter un seul type de signal (triangulaire sur base principalement).

5.2.3.3 Canal couleur

Dans OpenCV, il n'est actuellement pas possible d'utiliser des filtres de HAAR ayant un paramètre canal couleur comme cela est réalisé dans [BZR+05]. Il nous faut donc choisir un canal couleur avec lequel travailler. Certains travaux sur la détection et la reconnaissance de signaux routiers ont été effectués en niveaux de gris. C'est le canal usuel quand plusieurs canaux ne sont pas disponibles. Nous avons

donc naturellement adopté ce canal. De plus, cela rend notre approche applicable à d'autres signaux qu'aux signaux à forte composante rouge dont nous nous occupons.

Cependant, Bahlmann indique dans ses travaux les filtres les plus discriminants dans la cascade. Ceux-ci sont représentés sur la figure 5.12, ils sont placés de gauche à droite par ordre décroissant de pouvoir discriminant. De gauche à droite, les canaux couleurs des filtres sont respectivement : rouge normalisé, rouge, vert, rouge normalisé, niveaux de gris et vert normalisé. On voit que le canal rouge normalisé est celui de plus grande importance. Ceci nous incite à entraîner nos cascades sur ce canal.

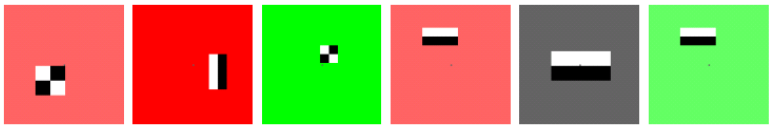


Fig. 5.12 – Filtres de Haar les plus discriminants obtenus dans [BZR+05].

Enfin, nous testons un troisième canal. Le canal de saturation. Celui-ci nous semble potentiellement intéressant, car pour un signal triangulaire, la saturation est faible dans tout la zone blanche et noire du triangle intérieur alors qu'elle est plus élevée sur le bord rouge.

Nous entraînons 3 cascades de classeurs, sur les 103 images de signaux triangulaires utilisées précédemment, respectivement en niveaux de gris, en rouge normalisé et en saturation. Nous utilisons les formules suivantes pour le calcul des canaux à partir des composantes r, g, b :

$$\text{niveaux de gris} : \frac{r+g+b}{3}, \tag{5.15}$$

$$\text{rouge normalisé} : \frac{3r}{r+g+b}, \tag{5.16}$$

$$\text{saturation} : \frac{\max(r,g,b) - \min(r,g,b)}{\max(r,g,b)}. \tag{5.17}$$

Les courbes ROC résultant des tests sont montrées sur la figure 5.13. Nous voyons que les performances de la cascade de classeurs entraînée sur le canal rouge normalisé sont meilleures que celles de la cascade entraînée en niveaux de gris. La cascade entraînée sur le canal saturation présente des performances inférieures à celles des autres cascades. Voici une explication possible : le rouge du bord du signal n'est pas toujours très saturé. Par conséquent, les réponses aux filtres de Haar ne sont pas suffisamment discriminantes. Nous abandonnons donc ce canal couleur.

Dans la suite, nous travaillons principalement avec le canal niveaux de gris [7], ainsi notre approche restera valable pour des signaux autres que les signaux à composante

[7] Nous optons en fait pour le canal luminance Y dans l'optique d'une application future, car les caméras IEEE 1394 que l'on pourrait utiliser pour le projet travaillent généralement dans l'espace YUV.

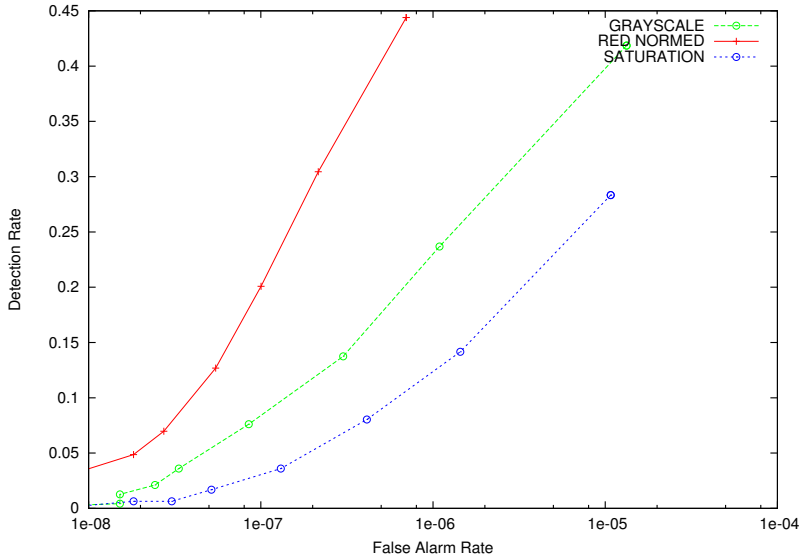


FIG. 5.13 – Courbes ROC de cascades entraînées suivant différents canaux couleurs.

rouge. Par exemple pour les signaux d'obligation qui sont principalement de couleur bleue. Toutefois, nous travaillerons également sur le canal rouge normalisé qui offre de meilleurs résultats dans le cas qui nous occupe.

5.2.3.4 Taille de l'ensemble d'apprentissage $\mathcal{LS}$

Nous comparons les performances d'une cascade de classeurs entraînée sur 798 images de signaux triangulaires, avec celles d'une cascade entraînée sur 103 images de signaux triangulaires [8]. Les courbes ROC résultant des tests sont représentées sur la figure 5.14. Elles montrent que l'augmentation de la taille de l'ensemble d'apprentissage améliore les performances.

Dans [Weh00, chapitre 7], on trouve une introduction à la théorie de l'apprentissage. On y explique que pour un contexte fixé [9], on peut décomposer l'erreur moyenne E_{A} faite par les modèles pouvant être construits par un algorithme A, en une somme de 3 termes :

$$E_{\mathrm{A}} = E_R + E_B + E_V \tag{5.18}$$

Le premier terme est l'erreur résiduelle E_R, aussi appelé bruit. L'erreur résiduelle est l'erreur commise par le meilleur modèle, le modèle de BAYES [Weh00, chapitre 2]. Le second terme est le biais E_B, qui mesure l'erreur entre le modèle moyen construit par l'algorithme A et le modèle de BAYES. Le dernier terme est la variance E_V, qui représente la dépendance du modèle par rapport au $\mathcal{LS}$ [Geu04]. Quand le $\mathcal{LS}$ est

[8] Nous parlons ici des exemples positifs. Le nombre d'exemples négatifs utilisés est adapté au nombre d'exemples positifs.

[9] Étant donnés un problème, un schéma d'échantillonnage et une fonction de calcul d'erreur du modèle.

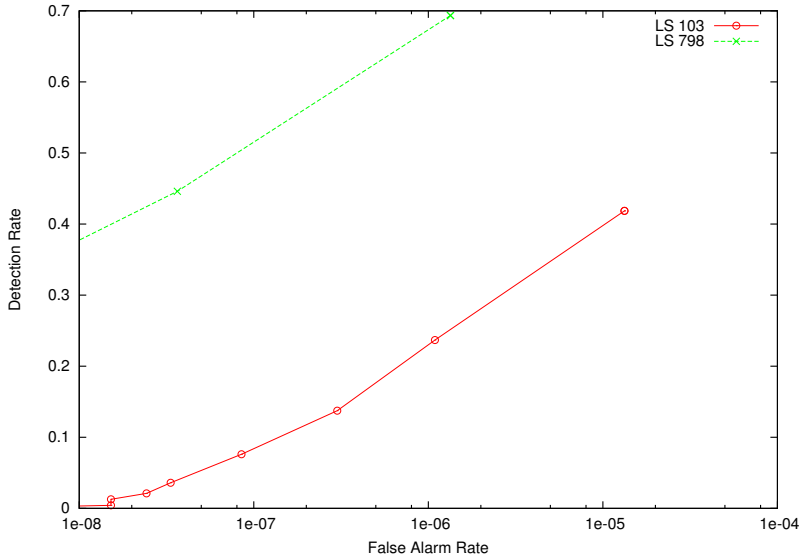


FIG. 5.14 – Courbes ROC de cascades entraînées avec des ensembles d'apprentissage de tailles différentes. Le nombre d'exemples positifs utilisés pour l'entraînement de la cascade représentée par la courbe ROC rouge (verte) est de 103 (798).

petit, la variance du modèle est importante. Cela signifie que le modèle construit est fort dépendant du $\mathcal{LS}$. Quand la taille du $\mathcal{LS}$ augmente, la variance diminue et l'erreur que fait le modèle est alors essentiellement due au biais et à l'erreur résiduelle. L'erreur résiduelle est fonction de la taille de l'espace d'attributs choisi. Généralement, augmenter le nombre d'attributs permet de réduire l'erreur résiduelle. On peut réduire le biais en augmentant l'espace des modèles pouvant être construits, c'est-à-dire en modifiant l'algorithme A. Nous renvoyons le lecteur aux références susmentionnées pour une discussion plus en détails de ces problèmes.

Ce que nous venons d'exposer justifie donc bien le fait qu'un $\mathcal{LS}$ petit mène à un taux de détection faible, ce qui nous pousse à augmenter la taille du $\mathcal{LS}$ autant que faire se peut. Dans la section 5.2.1, nous décrivons les différentes manières d'y arriver.

Une chose est à remarquer sur la figure 5.14. On pourrait être tenté de prolonger la courbe de verte pour des valeurs du TFA plus faibles. Cependant, il faut se rappeler que nous disposons pour les tests d'environ $3 \cdot 10^8$ échantillons (fenêtres testées). Nous ne pouvons donc rien déduire du TD pour un TFA de l'ordre de 10^{-9}. Il faudrait pour cela avoir mesuré une fausse alarme après avoir testé au moins 10^9 échantillons.

5.2.3.5 Ensemble de prototypes et type de classeur faible

Nous traitons ici de deux paramètres : le paramètre *nsplit* qui détermine le type de classeur faible et le paramètre *mode* qui sélectionne l'ensemble de prototypes de filtres de HAAR utilisés.

Par défaut, des souches (arbres de décision à un seul nœud de test) sont utilisées comme classeurs faibles, on a *nsplit* = 1. Le paramètre *nsplit* représente le nombre de nœuds de test dans le classeur faible. Nous utilisons des arbres à 1, 2 ou 3 nœuds. Utiliser un arbre à plusieurs nœuds permet de modéliser des dépendances entre attributs, ce qu'une souche ne peut faire [LKP03].

Pour l'ensemble de prototypes de filtres de HAAR, si *mode* = ALL, tous les prototypes de filtres disponibles sont utilisés. Si *mode* = BASIC, seuls les prototypes (1a), (1b), (2a), (2c) et (4) sont utilisés (voir figure 5.4). Théoriquement, le fait d'utiliser plus de prototypes améliore le pouvoir de représentation des filtres de HAAR. D'un autre côté, l'utilisation d'un nombre réduit de prototypes donne lieu à moins de filtres de HAAR et donc à moins de calculs lors de l'entraînement. Pour la détection, la complexité de calcul est constante quelque soit le paramètre *mode* utilisé.

Les figures 5.15 et 5.16 montrent les courbes ROC des cascades obtenues en faisant varier les paramètres d'entraînement : *mode* et *nsplit*. Selon [LKP03], travailler avec des arbres à 2 nœuds plutôt qu'avec des souches permet d'améliorer les performances. En regardant la figure 5.15, il est difficile de tirer des conclusions sur l'influence de la profondeur de l'arbre. Aussi, nous continuerons à travailler avec les paramètres standards d'OpenCV, c'est-à-dire des souches comme classeurs faibles.

Selon [LM02], l'utilisation de tous les prototypes améliore les performances, en particulier si les objets présentent des structures diagonales (ce qui est le cas pour nos signaux routiers triangulaires). Sur la figure 5.16, qui représente les meilleurs courbes du graphe précédent, ces résultats ne sont pas confirmés.

Nous connaissons la raison pour laquelle nous n'obtenons pas les mêmes résultats que LIENHART. Dans la version d'OpenCV sur laquelle nous travaillons, lorsque *mode* = BASIC, le prototype (4) de la figure 5.4 est utilisé. Dans [LM02], ce n'est par le cas[10]. Or, le but des filtres orientées à 45° est entre autres de remplacer le prototype (4), pour la détection des caractéristiques non horizontale ou verticale.
La figure 5.17 montre les filtres de HAAR du premier étage de cascades entraînées avec *mode* = ALL et *mode* = BASIC. Les filtres de prototype (4) et les filtres orientés à 45° sont tracés en bleu. On remarque que lorsque *mode* = BASIC, les prototypes orientés à 45° n'étant pas disponibles, l'algorithme de construction de la cascade choisit des filtres de prototype (4) pour mettre en évidence les arêtes obliques des signaux triangulaires.
Il apparaît maintenant évident que les résultats obtenus pour *mode* = BASIC seraient, comme le dit LIENHART, beaucoup moins bons si ce mode n'utilisait pas les prototypes (4).
Il est à noter que si seuls des filtres non orientés sont sélectionnés lors d'un entraînement en *mode* ALL, la cascade obtenue pour le même entraînement, mais en *mode* BASIC, sera identique. À partir du moment où un filtre orienté est sélectionné dans le *mode* ALL, les entraînements divergent. Les filtres choisis ensuite ne sont plus les mêmes car les poids des exemples, changent après chaque choix de filtre (voir 5.1.3).

[10] LIENHART n'utilise pas ce prototype, car il ne peut se calculer sur base de deux rectangles comme les autres prototypes.

Sur la figure 5.17, on remarque qu'il y a un filtre de prototype (4) commun aux deux classeurs.

5.2.3.6 Élagage d'une cascade de classeurs

Nous ne parlons pas vraiment ici de paramètre d'entraînement de la cascade, mais plutôt d'un moyen d'améliorer les performances de la cascade entraînée. La structure de la cascade permet d'augmenter facilement le taux de détection lorsque celui-ci, mesuré sur un ensemble de test, est très en dessous des valeurs attendues. Il suffit de supprimer les derniers étages de la cascade, qui sont les plus sélectifs (voir section 5.1.4). Évidemment, si le taux de détection augmente, il en est malheureusement de même pour le taux de fausses alarmes.

Comme la cascade est un arbre de décision dégénéré, on peut parler d'élagage pour la suppression des différents étages. Cet élagage est semblable à celui que l'on peut réaliser sur un simple arbre de décision [Weh00, chapitre 5]. L'élagage peut augmenter l'erreur faite sur l'ensemble d'apprentissage, mais en général, il permet de réduire l'erreur de détection réelle de la cascade de classeurs. L'élagage est une technique de réduction de la variance des modèles, variance dont nous avons déjà parlé. Autrement dit, il sert à diminuer le surapprentissage [11]. Attention, si on élague trop, on risque d'augmenter l'erreur de détection réelle en augmentant le biais. En effet, trop élaguer entraîne du sousapprentissage [12]. Ce compromis à réaliser est souvent appelé compromis biais/variance. On donne des détails à ce sujet dans [Geu04].

L'élagage doit se faire sur un ensemble d'élagage. Nous n'avons pas de résultats d'élagage à présenter, mais l'élagage est une voie à explorer pour améliorer les performances de nos cascades de classeurs.

[11] Le surapprentissage est la spécialisation du modèle sur le $\mathcal{LS}$.

[12] On parle de sousapprentissage si le modèle ne retire pas assez d'informations du $\mathcal{LS}$ pour avoir un bon pouvoir de prédiction.

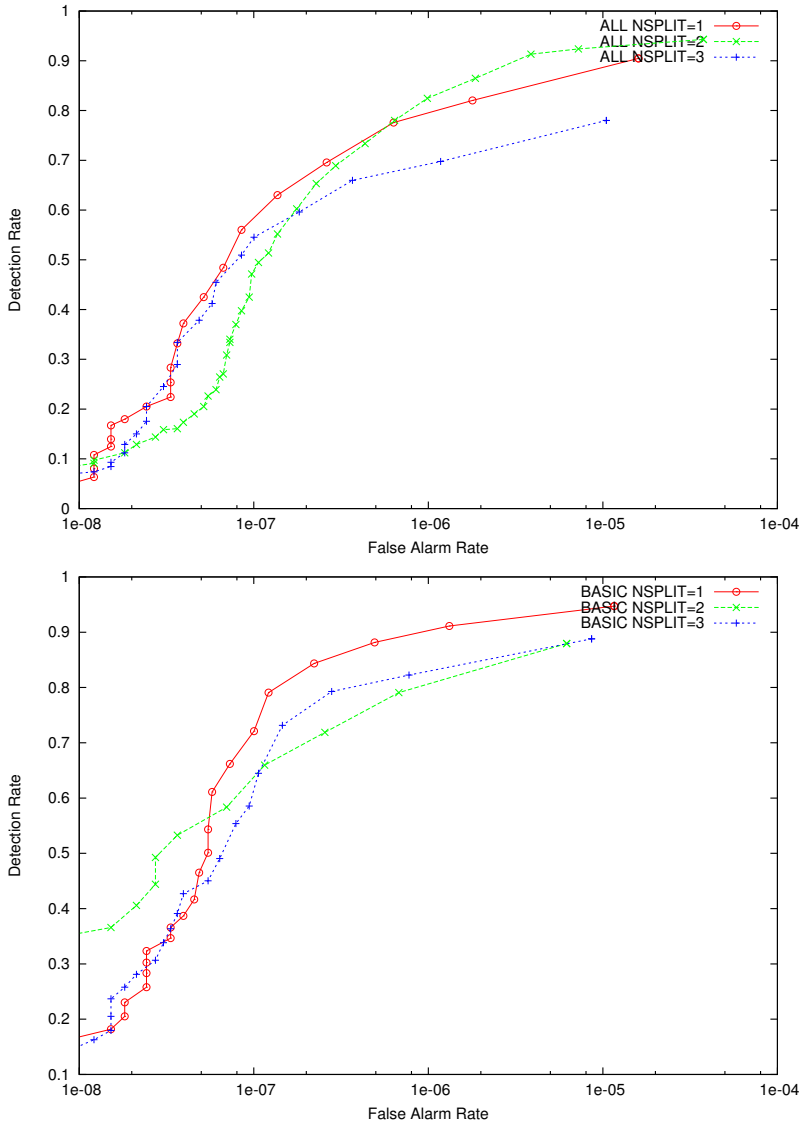


FIG. 5.15 – Courbes ROC de cascades entraînées avec différentes valeurs des paramètres *mode* et *nsplit*. Sur chaque graphe, 3 cascades ont été entraînées avec respectivement 1, 2 et 3 nœuds par classeur faible (*nsplit*). Sur le graphe du dessus, on a utilisé tous les prototypes de filtres de HAAR pour les entraînements alors qu'on n'en a utilisé qu'une partie pour les entraînements des cascades du graphe du dessous.

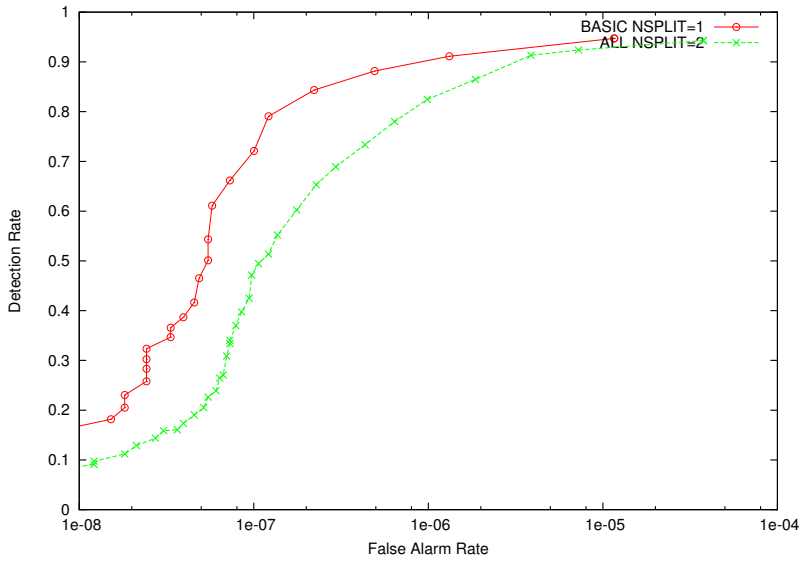
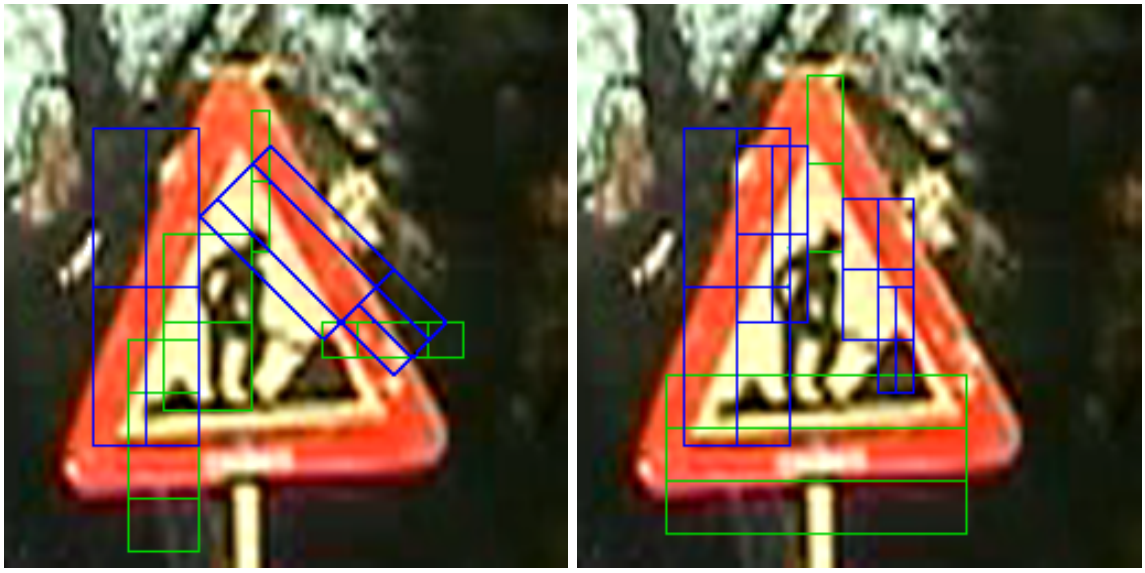


FIG. 5.16 – Courbes ROC de cascades entraînées avec différentes valeurs des paramètres *mode* et *nsplit*. On reprend ici les meilleures courbes des deux graphes de la figures 5.15.



(a) *mode* = ALL (b) *mode* = BASIC

FIG. 5.17 – Représentation, sur une image de signal, de filtres de HAAR du premier étage de cascades entraînées avec deux valeurs du paramètre *mode*. Les filtres représentés en bleu sont des filtres orientés ou des filtres de prototype (4). À droite, les filtres orientés sont indisponibles. Des filtres de prototype (4) sont donc sélectionnés lors de l'entraînement pour mettre en évidence les structures diagonales des signaux routiers.

5.2.3.7 Conclusions

Nous terminons ici l'analyse des paramètres utilisés dans l'entraînement de la cascade de classeurs en tirant quelques enseignements. Nous devons effectuer l'entraînement sur des images 32×32. Pour nos tests, nous choisissons de détecter uniquement des signaux triangulaires, plutôt que de chercher à détecter tous les signaux de notre base de données. Nous utilisons, de manière très générale, le canal luminance Y, tout en rappelant que pour les signaux que nous avons choisi, nous pouvons utiliser le canal rouge normalisé qui donne de meilleurs résultats. La taille de notre ensemble d'apprentissage, 798 images de signaux triangulaires, est raisonnable. Nous avons aussi testé d'autres paramètres comme l'ensemble de prototypes de filtres de HAAR et le type de classeur faible utilisé. Les conclusions au sujet de ces paramètres sont moins évidentes et nécessitent une étude supplémentaire. Il en est de même pour l'élagage dont nous venons de discuter.

Nous avons représenté sur la figure 5.18 les filtres de HAAR utilisés par les classeurs des 5 premiers étages de la cascade entraînée avec les paramètres suivants : taille de fenêtre 32×32, ensemble d'apprentissage composé de 798 signaux triangulaires, canal luminance Y, $mode$ = BASIC, $nsplit = 1$. On voit que les filtres sont concentrés essentiellement dans la zone occupée par le signal triangulaire. La cascade de classeurs entraînée obtient les performances suivantes (voir courbe rouge de la figure 5.16) : un taux de détection supérieur à 90 % pour un taux de fausses alarmes supérieur à 10^{-6}. Ce taux de fausses alarmes indique qu'en moyenne, il y a au moins une fausse détection par image (puisque environ 700 000 fenêtres sont testées par image). Des exemples de détections effectuées sont montrées sur les figure 5.19 et 5.20. La section suivante décrit une méthode pour réduire ce nombre de fausses alarmes.

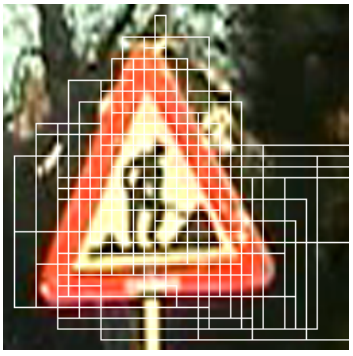


FIG. 5.18 – Représentation, sur une image de signal, des filtres de HAAR sélectionnés dans les 5 premiers étages d'une cascade particulère.

FIG. 5.19 – Exemple de détection : 1 signal détecté, pas de fausse alarme.

FIG. 5.20 – Exemple de détection : 1 signal détecté et 1 fausse alarme.

5.3 Réduction du nombre de fausses alarmes

Grâce à la méthode décrite dans la section précédente, la détection et la localisation des signaux routiers est réalisée de manière satisfaisante. Nous obtenons des taux de détection supérieurs à 90 % sur notre ensemble de test. Nous parlons du taux de détection, mais celui-ci regroupe les phases de détection et de localisation. Nous avons vu aussi que le nombre de fausses alarmes était non négligeable, environ 1 fausse alarme par image testée. Afin de faciliter les étapes qui suivent la détection et la localisation, c'est-à-dire le suivi et la reconnaissance, nous tentons ici de diminuer le nombre de fausses alarmes d'une manière très simple.

La méthode utilisée jusqu'à présent est très générale, elle est applicable à bien d'autres objets qu'aux signaux routiers. Afin de diminuer le nombre de fausses alarmes, nous allons nous servir des informations a priori que nous possédons sur les signaux. Considérons les signaux triangulaires sur base. Nous allons utiliser la distribution spatiale des couleurs. Nous réalisons un test très simple. Pour ce faire, nous décomposons l'image d'un signal triangulaire en 9 zones (voir figure 5.21). Une quantité assez importante de rouge est présente dans les cases contenant les pointes du triangle, ce qui ne devrait pas être le cas pour la plupart des fausses alarmes. Nous allons nous servir de cette idée pour réduire le nombre de fausses alarmes par des contraintes sur la couleur. Nous décidons arbitrairement d'effectuer un test dans les deux cases contenant les pointes inférieures du triangle. Nous définissons un pixel comme rouge si la composante r est maximale et si elle est 10 % plus élevée qu'au moins une des deux autres composantes [13]. Si au moins une des deux cases testées contient moins de 30 % des pixels rouges, l'image est considérée comme une fausse alarme et est rejetée.

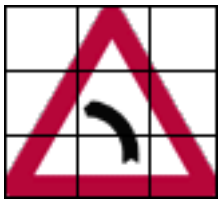
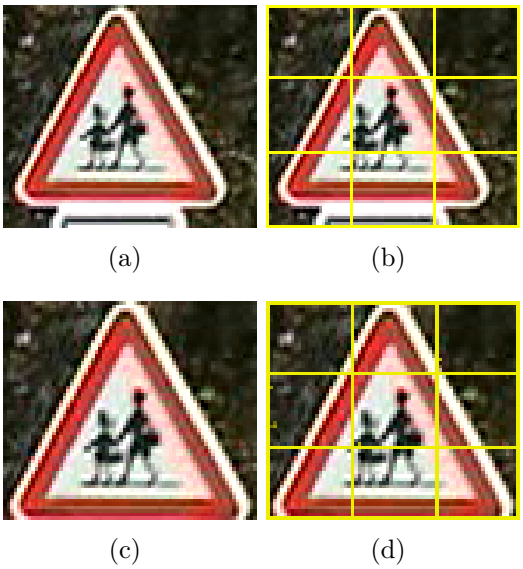


FIG. 5.21 – Découpage de l'image d'un signal triangulaire en 9 zones.

Pour appliquer cette technique, nous avons deux problèmes. Premièrement, les exemples positifs de notre base de données ne sont pas toujours positionnés au centre de l'image. Ensuite, les signaux ne sont pas précisément placés dans les images comme le signal de la figure 5.21. Ils occupent une proportion de l'image qui varie selon les exemples. La figure 5.22(a) montre un exemple de notre base de donnée. Le résultat du découpage en 9 cases, sur la figure 5.22(b), met en évidence la conséquence de ces deux problèmes : le nombre de pixels considérés comme rouges risque d'être insuffisant pour réussir le test couleur.

[13] Nous considérons les images en canaux RGB.



(a) (b)

(c) (d)

FIG. 5.22 – Image de la base de données originale (a) et (b) et image après recentrage sur le signal (c) et (d).

Les fenêtres détectées par la cascade de classeurs présentent les mêmes caractéristiques que les exemples de l'ensemble d'apprentissage. Les signaux ne sont pas toujours centrés et ils occupent une proportion variable de la fenêtre. Nous décidons donc de constituer une nouvelle base de données où chaque signal est centré parfaitement dans l'image et où la taille de l'image correspond à la taille du signal comme sur la figure 5.21. Nous générons cette nouvelle base de données à partir de celle dont nous disposons. Nous travaillons uniquement sur les images de signaux triangulaires. Les triangles de la signalisation routière sont des triangles équilatéraux, ce qui signifie que le rapport de hauteur à la largeur est de : $\sin(60°) \simeq 0,87$. Les triangles équilatéraux de la réalité sont déformés par une transformation projective lorsqu'ils apparaissent dans nos séquences vidéo. Néanmoins, on peut négliger cet effet dans de nombreux cas. Aussi, nous composons notre nouvelle base de données avec des images ayant un rapport hauteur sur la largeur de $0,87$[14]. Nous montrons un exemple sur la figure 5.22. La figure 5.22(c) est la région rectangulaire (avec un rapport hauteur sur largeur de $0,87$) obtenue à partir de l'image 5.22(a). On remarque sur la figure 5.22(d) que le triangle est bien centré pour le découpage de l'images en 9 zones.

Ayant généré ainsi une nouvelle base de données d'images, nous construisons de nouvelles cascades de classeurs. Nous n'entrerons pas dans les détails des performances de ces cascades. Notons simplement que les résultats obtenus sont similaires à ceux obtenus pour les cascades dont nous avons parlé précédemment. Nous ob-

[14] Pour que l'approche reste valable pour des signaux circulaires et octogonaux, et parce que nous ne sommes pas certains de l'avantage tiré d'images rectangulaires, nous générons également une base de données avec des images de dimension carrée. Nous travaillons dans la suite de ce document sur les images rectangulaires sans montrer de résultats obtenus sur les images carrées.

tenons maintenant des détections bien centrées sur les signaux routiers, comme le montre la figure 5.23. Si on compare la figure 5.19 à la figure 5.23, on peut constater la différence entre la détection avec une cascade entraînée sur les images originales et la détection avec une cascade entraînée sur les images recentrées.

FIG. 5.23 – Marquage d'un objet détecté. La cascade de classeurs utilisée a été entraînée sur un ensemble d'apprentissage dans lequel les signaux sont recentrés.

Pour montrer l'avantage que l'on peut tirer lorsque les signaux sont bien centrés dans les fenêtres détectées, nous effectuons le test couleur décrit en début de section sur les images de la figure 5.22. Le tableau 5.2 reprend les résultats obtenus. On voit que lorsque l'on recentre le signal (figure 5.22(c)), la proportion des pixels rouges augmente. Elle passe de 57 % à 67 % pour la case inférieure gauche et de 80 % à 86 % pour la case inférieure droite. Ici, même l'image originale aurait réussi le test couleur, mais cet exemple illustre l'amélioration que l'on peut attendre lorsque les signaux sont mieux positionnés dans les images.

Pixels	Image originale	Signal recentré
Par case	656	496
Rouges C20	376	331
Rouges C22	522	427
Rouges C20 (%)	57	67
Rouges C22 (%)	80	86

TAB. 5.2 – Nombre de pixels rouges dans les cases inférieures gauche (C20) et droite (C22).

Le test couleur peut paraître peu sévère, mais il permet d'éliminer un grand nombre de fausses alarmes présentes dans la végétation et dans les régions à dominante bleue. On pourrait se demander si le test couleur n'élimine pas les signaux dont le rouge est faiblement saturé (par exemple, des signaux routiers vieux ou sales). Le problème ne se pose pas, la cascade de classeurs ne détecte pas ces signaux, car nous ne les avons pas placé dans l'ensemble d'apprentissage. Dans le cadre de la maintenance de la signalisation routière, c'est un avantage. En effet, lorsqu'un signal devient par exemple sale ou vieux, le fait de ne plus le détecter indique qu'il est temps de le laver ou de le remplacer. Pour cela, il faut évidemment qu'il soit déjà présent dans la base de données.

Pour obtenir des valeurs numériques du nombre de fausses alarmes supprimées grâce au test couleur, nous modifions l'utilitaire `haarperformance` et y insérons notre test couleur. Ainsi, nous pouvons obtenir les valeurs des VP, FN, FP, TD. Nous testons plusieurs cascades de classeurs en faisant varier deux paramètres :

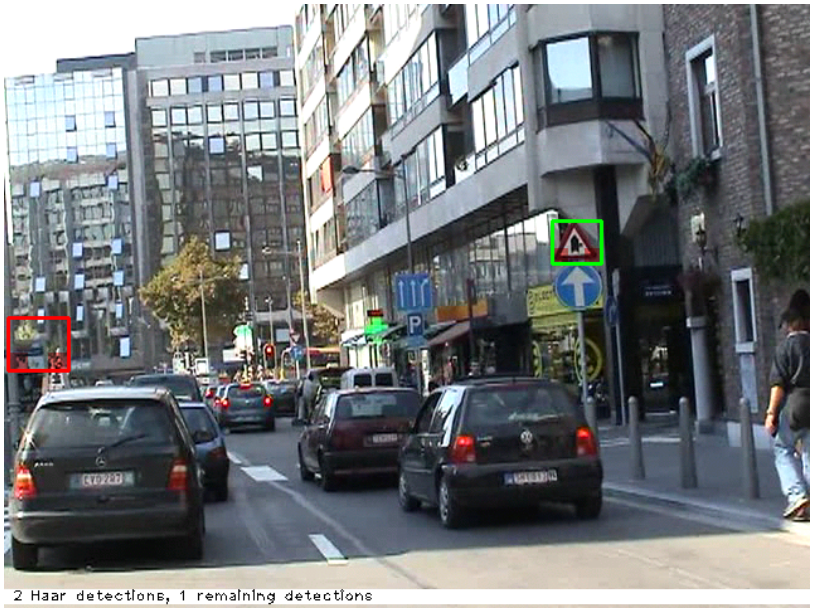
- l'ensemble de prototypes utilisés, à savoir le paramètre *mode* ;
- les images de l'ensemble d'apprentissage, soit les images originales, soit les images recentrées.

L'influence du premier des deux paramètres a été discutée dans la section et 5.2.3.5. Les seules conclusions que l'on tire ici portent sur la réduction du TFA et du TD apportée par le test couleur. Lorsqu'on applique le test couleur, on observe une réduction de 70 à 83 % du TFA, ce qui est satisfaisant. Le test couleur élimine aussi malheureusement certains signaux. C'est ici que l'on voit l'intérêt du recentrage. Sur les cascade de classeurs entraînées avec les images originales de la base de données, la diminution du TD est comprise entre 29 et 44 %. Heureusement, sur les cascade de classeurs entraînées avec les images recentrées, cette diminution du TD est comprise entre 10 et 19 %. Pour le lecteur intéressé, les résultats complets sont repris en annexe E. Nous montrons sur les figures 5.24(a) et 5.24(b) des exemples d'images où le test couleur a éliminé des fausses détections.

La méthode simple présentée dans cette section permet de réduire le nombre de fausses alarmes. Si cela n'était pas suffisant, on pourrait imaginer d'imposer d'autres contraintes en se servant des connaissances a priori sur les signaux routiers. Par exemple, on pourrait effectuer, pour la pointe supérieure du triangle, un test similaire à celui réalisé pour les deux pointes inférieures. On pourrait aussi vérifier que les pixels du triangle intérieur sont essentiellement noir et blanc. Il faudrait par exemple imposer une contrainte sur la saturation [15]. D'autres contraintes sont envisageables, nous nous limiterons cependant dans ce travail à la méthode présentée dans cette section.

Ceci clôture la partie consacrée à la détection et la localisation des signaux routiers. Les prochaines sections seront consacrées au suivi et à la reconnaissance des signaux.

[15] Pour le blanc et le noir, la saturation est faible.



(a)

(b)

FIG. 5.24 – Sur les deux images, une fausse alarme a été éliminée par le test couleur (cadre rouge). La vraie détection réussit le test couleur (cadre vert). Les zones inspectées pour le test couleur sont représentées en filtre rouge.

Chapitre 6

Suivi des signaux

Rappelons que l'objectif ultime de ce travail est de pouvoir construire une base de données de signaux routiers géoréférencés. À chaque signal présent dans la scène doit correspondre une et une seule entrée dans la base de données. Le suivi d'un signal détecté une première fois est donc une opération indispensable pour notre application. De plus, le suivi permet d'extraire plusieurs vues du même signal, ce qui augmente les chances qu'il soit correctement reconnu par le module de reconnaissance (en effectuant par exemple un vote majoritaire).

6.1 Généralités sur le suivi

Le suivi peut être vu comme un problème d'inférence probabiliste [FP03, chapitre 17]. Dans la k^{e} image du flux vidéo, l'objet à poursuivre est caractérisé par un vecteur d'état $\mathbf{x}_k$ qui peut ne pas être observé directement. Le vecteur d'état est une variable aléatoire. Des variables d'état typiques sont la position, la vitesse, l'accélération, la taille de l'objet ou encore la distance parcouru par celui-ci. Grâce à un modèle dynamique de l'état, nous pouvons prédire son évolution dans le temps. Cela permet entre autre de déterminer une région d'intérêt dans laquelle l'objet se trouve probablement. Périodiquement, nous obtenons par des mesures un vecteur d'observations $\mathbf{z}_k$ qui dépend de l'état et grâce auquel nous pouvons corriger l'état prédit. Le vecteur d'observation est une variable aléatoire.

L'inférence probabiliste tente de résoudre essentiellement trois problèmes :

Prédiction : ayant observés $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}$, nous voulons prédire l'état $\mathbf{x}_k$ de la cible dans la k^{e} image. Nous avons besoin pour cela d'une représentation de la densité de probabilité $p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1}) \triangleq p(\mathbf{x}_k \mid \mathbf{z}_0, \ldots, \mathbf{z}_{k-1})$.

Association de données : il se peut que les mesures effectuées sur la k^{e} image apportent plusieurs observations $\mathbf{z}_k$ différentes. Nous pouvons par exemple détecter l'objet actuellement poursuivi à deux endroits différents. Nous avons besoin de $p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1})$ afin de décider quelles mesures correspondent probablement à la cible.

Correction : ayant l'observation $\mathbf{z}_k$ qui correspond à la cible, nous avons besoin de $p(\mathbf{x}_k \mid \mathbf{z}_{0:k})$ afin d'estimer l'état a posteriori de la cible (car la mesure est entachée de bruit).

L'hypothèse d'une chaîne de MARKOV cachée est souvent adoptée pour faciliter le problème du suivi. Dans ce cas, seulement le passé immédiat de la cible importe,

$$p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \tag{6.1}$$

et les mesures ne dépendent que de l'état courant,

$$p(\mathbf{z}_k \mid \mathbf{x}_{0:k}, \mathbf{z}_{0:k-1}) = p(\mathbf{z}_k \mid \mathbf{x}_k). \tag{6.2}$$

Le réseau bayésien correspondant à une chaîne de Markov cachée est représenté à la figure 6.1.

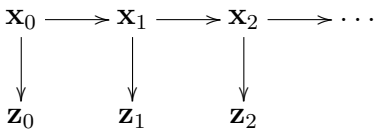


FIG. 6.1 – Chaîne de Markov cachée.

Le problème se résume globalement à trouver une représentation des densités de probabilité qui soit suffisamment précise et qui permette de calculer rapidement et simplement les nouvelles densités qui apparaissent à chaque étape de l'algorithme de suivi. Le cas le plus simple survient lorsque le modèle dynamique ainsi que le modèle de mesure sont linaires et que les bruits qui perturbent ces modèles sont gaussiens. En effet, une variable aléatoire normale reste normale après transformation linéaire. Toutes les densités qui apparaissent au cours du suivi sont alors gaussiennes et les seuls paramètres à manipuler sont des moyennes et des matrices de covariance. Dans ce cas, on peut montrer que le filtre de KALMAN (voir annexe A) est un estimateur récursif optimal. Si les modèles ne sont pas linéaires ou que les distributions ne sont pas normales, d'autres méthodes existent, comme le filtre de KALMAN étendu (voir annexe B) ou les filtres à particules [IB98].

6.2 Modèle dynamique

Dans cette section, nous allons choisir un modèle dynamique permettant de prédire l'évolution de l'état des signaux routiers dans le flux vidéo. Ce modèle doit permettre d'estimer au moins la position future de la cible afin de déterminer une région d'intérêt dans laquelle la cible est probablement située.

La figure 6.2 représente schématiquement la situation générale dans laquelle $\mathbf{s}(t)$ est la position de la cible. Le référentiel $(\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}})$ est celui de la caméra et $(\underline{\mathbf{u}}, \underline{\mathbf{v}})$ est le plan image sur lequel la scène est projetée en perspective, en particulier $\mathbf{p(t)}$ est la

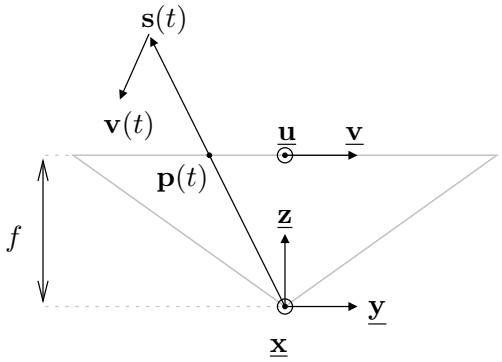


FIG. 6.2 – Modèle sténopé d'une caméra embarquée à bord du véhicule.

position de l'objet sur le plan image. Dans le cas le plus général, le véhicule, la caméra et aussi l'objet peuvent se déplacer. Toute la dynamique du système est ramenée ici à une vitesse équivalente $\mathbf{v}(t)$ de l'objet à poursuivre.

6.2.1 Déplacement à vitesse constante parallèlement à l'axe optique

La situation le plus simple mais néanmoins courante survient lorsque le vecteur vitesse du signal est parallèle à l'axe optique de la caméra. C'est notamment le cas lorsque le véhicule circule sur une voie rectiligne et que la caméra est orientée dans le sens de déplacement. Dès lors, la vitesse du signal dans le référentiel de la caméra est $\mathbf{v}(t) = v_z\,\underline{\mathbf{z}}$ et sa position est donnée par

$$\mathbf{s}(t) = x_0\,\underline{\mathbf{x}} + y_0\,\underline{\mathbf{y}} + (v_z\,t + z_0)\,\underline{\mathbf{z}}. \tag{6.3}$$

En projetant cette position sur le plan image de la caméra on obtient

$$\begin{aligned} \mathbf{p}(t) &= f\,\frac{x(t)}{z(t)}\,\underline{\mathbf{u}} + f\,\frac{y(t)}{z(t)}\,\underline{\mathbf{v}} && (6.4)\\ &= f\,\frac{x_0}{v_z\,t + z_0}\,\underline{\mathbf{u}} + f\,\frac{y_0}{v_z\,t + z_0}\,\underline{\mathbf{v}} && (6.5) \end{aligned}$$

En éliminant le paramètre temps de cette dernière équation, on remarque que la trajectoire locale sur le plan image est une droite de coefficient angulaire y_0/x_0. Nous pouvons donc estimer la direction vers laquelle va se déplacer l'objet en connaissant sa position courante. Si le véhicule se déplaçait en permanence selon ce modèle, l'ensemble des trajectoires serait des droites se croisant au point de fuite $(u, v) = (0, 0)$ (FOE [1]).

[1] FOE : *Focus of Expansion*

L'abscisse $v(t)$ et l'ordonnée $u(t)$ de l'objet dans le plan image sont des fonctions de la forme

$$f(t) = \frac{1}{b\,t + c}. \tag{6.6}$$

Une telle fonction ayant deux degrés de liberté, il est possible de la décrire complètement si l'on en connaît deux points. Cela veut dire que nous pouvons prédire la position de l'objet à un instant t si l'on connaît deux de ses positions antérieures.

$$f_1 \xrightarrow{\Delta t_1} f_2 \xrightarrow{\Delta t_2} f_3$$

La résolution du système

$$\left\{ \begin{array}{lcl} f_1 & = & \frac{1}{b\,t_1 + c} \\ f_2 & = & \frac{1}{b\,t_2 + c} \end{array} \right. \tag{6.7}$$

livrent les paramètres b et c qui, réinjectés dans l'équation 6.6, donnent

$$f(t) = \frac{(t_2 - t_1)\, f_1\, f_2}{(t - t_1)\, f_1 - (t - t_2)\, f_2}. \tag{6.8}$$

En posant $c_{ij} = \Delta t_i / \Delta t_j = (t_{i+1} - t_i)/(t_{j+1} - t_j)$ et en évaluant f à l'instant t_3, on obtient

$$f_3 = \frac{f_1\, f_2}{(1 + c_{21})\, f_1 - c_{21}\, f_2}. \tag{6.9}$$

On retrouve la relation utilisée par Miura et al. [MKS00] pour prédire la position du signal. Cette relation est non linéaire. Nous pourrions l'utiliser avec un filtre de Kalman étendu comme expliqué à l'annexe B.

6.2.2 Déplacement à vitesse constante

Restons dans la situation où le véhicule se déplace localement à vitesse constante et dans une direction donnée. Lorsque le véhicule en dépasse un autre ou lorsque l'axe optique de la caméra n'est pas parallèle à la direction de déplacement, le modèle précédent n'est plus correct. La vitesse de la cible dans le référentiel de la caméra est $\mathbf{v}(t) = v_x\,\underline{\mathbf{x}} + v_y\,\underline{\mathbf{y}} + v_z\,\underline{\mathbf{z}}$ et sa position est donnée par

$$\mathbf{s}(t) = (v_x\,t + x_0)\,\underline{\mathbf{x}} + (v_y\,t + y_0)\,\underline{\mathbf{y}} + (v_z\,t + z_0)\,\underline{\mathbf{z}}. \tag{6.10}$$

En projetant cette position sur le plan image de la caméra on obtient

$$\mathbf{p}(t) = f\,\frac{x(t)}{z(t)}\,\underline{\mathbf{u}} + f\,\frac{y(t)}{z(t)}\,\underline{\mathbf{v}} \tag{6.11}$$

$$= f\,\frac{v_x\,t + x_0}{v_z\,t + z_0}\,\underline{\mathbf{u}} + f\,\frac{v_y\,t + y_0}{v_z\,t + z_0}\,\underline{\mathbf{v}} \tag{6.12}$$

Si l'on fait tendre le temps vers l'infini, on obtient le point de fuite $(u, v) = (f\,v_x/v_z, f\,v_y/v_z)$. Les trajectoires locales sur le plan image sont toujours des droites (champ de mouvement radial). Leur coefficient angulaire vaut

$$\frac{v(t) - f\,v_y/v_z}{u(t) - f\,v_x/v_z} = \frac{y_0 - z_o\,v_y/v_z}{x_0 - z_o\,v_x/v_z}. \tag{6.13}$$

L'abscisse $v(t)$ et l'ordonnée $u(t)$ de l'objet dans le plan image sont des fonctions de la forme

$$f(t) = \frac{a\,t + 1}{b\,t + c}. \tag{6.14}$$

Une telle fonction ayant trois degrés de liberté, il est possible de la décrire complètement si l'on en connaît trois points. Cela veut dire que nous pouvons prédire la position de l'objet à un instant t si l'on connaît trois de ses positions antérieures. La résolution du système

$$\left\{ \begin{array}{lcl} f_1 & = & \frac{a\,t+1}{b\,t_1+c} \\ f_2 & = & \frac{a\,t+1}{b\,t_2+c} \\ f_3 & = & \frac{a\,t+1}{b\,t_2+c} \end{array} \right. \tag{6.15}$$

livrent les paramètres a, b et c qui, réinjectés dans l'équation 6.14, fournissent la relation suivante,

$$f_4 = -\frac{c_{32}\,f_1\,f_2 + (1 + c_{21} + c_{21}\,c_{32})\,f_2\,f_3 - (1 + c_{21} + c_{21}\,c_{32} + c_{32})\,f_1\,f_3}{c_{32}\,f_3 + (1 + c_{21} + c_{21}\,c_{32})\,f_1 - (1 + c_{21} + c_{21}\,c_{32} + c_{32})\,f_2}. \tag{6.16}$$

On obtient, comme dans le cas précédent, une relation non linéaire que l'on pourrait linéariser et utiliser avec un filtre de Kalman étendu ou un filtre à particules [IB98].

6.2.3 Accélération constante

Dans bien des cas, dans le domaine du suivi d'objets, un simple modèle dynamique linéaire tel qu'un modèle à vitesse constante ou à accélération constante donne de très bons résultats. Nous décrivons ici le modèle à accélération constante que nous avons utilisé avec un filtre de Kalman et qui s'est révélé suffisamment efficace dans la plupart des cas.

Pour construire notre modèle, supposons que la projection du centre du signal routier sur le plan image de la caméra soit le point

$$\mathbf{p}(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix} \tag{6.17}$$

qui se déplace à accélération constante. Le mouvement de ce point est régi par l'équation différentielle

$$\frac{\mathrm{d}^2\,\mathbf{p}(t)}{\mathrm{d}\,t^2} = \mathbf{a} \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \ddot{u}(t) = a_u \\ \ddot{v}(t) = a_v \end{array} \right. \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \dot{u}(t) = c_u(t) \\ \dot{c}_u(t) = a_u \\ \dot{v}(t) = c_v(t) \\ \dot{c}_v(t) = a_v \end{array} \right. . \tag{6.18}$$

Les équations étant de la même forme pour l'abscisse $u(t)$ et l'ordonnée $v(t)$, nous ne traiterons qu'un des deux cas en prenant comme notations $x(t)$ pour la position, $c(t)$ pour la vitesse et a pour l'accélération. Le mouvement peut être décrit par l'équation d'état

$$\dot{\mathbf{x}}(t) = \boldsymbol{A}\mathbf{x}(t) \quad \text{avec} \quad \mathbf{x}(t) = \begin{bmatrix} x(t) \\ c(t) \\ a \end{bmatrix} \quad \text{et} \quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.19}$$

En discrétisant cette équation d'état par la méthode d'Euler explicite, c'est-à-dire en approximant la vitesse $c(k\Delta t)$ par

$$c_k \approx \frac{x_{k+1} - x_k}{\Delta t}, \tag{6.20}$$

on obtient l'équation d'état discrète

$$\mathbf{x}_k = \boldsymbol{A}_d\mathbf{x}_{k-1} \quad \text{avec} \quad \mathbf{x}_k = \begin{bmatrix} x_k \\ c_k \\ a \end{bmatrix} \quad \text{et} \quad \boldsymbol{A}_d = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.21}$$

Cette équation d'état à été obtenue en approximant la dérivée de la position par une différence finie. Nous pouvons trouver l'équation d'état exacte. La solution de l'équation 6.19 fait intervenir l'exponentielle matricielle $\exp(\boldsymbol{A}t)$,

$$\mathbf{x}(t) = e^{\boldsymbol{A}t}\mathbf{x}(0) \quad \text{avec} \quad e^{\boldsymbol{A}t} \triangleq \sum_{n=0}^{\infty} \frac{(\boldsymbol{A}t)^n}{n!}. \tag{6.22}$$

En calculant l'état à partir d'un temps τ, on a

$$\begin{aligned} \mathbf{x}(\tau + t) &= e^{\boldsymbol{A}(\tau+t)}\,\mathbf{x}(0) && (6.23)\\ &= e^{\boldsymbol{A}t}\,e^{\boldsymbol{A}\tau}\,\mathbf{x}(0) && (6.24)\\ &= e^{\boldsymbol{A}t}\,\mathbf{x}(\tau). && (6.25) \end{aligned}$$

En prenant $t = \Delta t$, nous obtenons l'équation d'état discrète exacte

$$\mathbf{x}_k = e^{\boldsymbol{A}\Delta t}\,\mathbf{x}_{k-1}. \tag{6.26}$$

L'exponentielle matricielle se calcule facilement car seuls les trois premiers termes sont non nuls,

$$\boldsymbol{A}_d = e^{\boldsymbol{A}\Delta t} = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.27}$$

Remarquons que la discrétisation par la méthode d'Euler revient à ne considérer que les deux premiers termes de l'exponentielle matricielle,

$$\begin{aligned} \mathbf{x}(\tau + \Delta t) &\approx \mathbf{x}(\tau) + \Delta t\,\dot{\mathbf{x}}(\tau) && (6.28)\\ &= \mathbf{x}(\tau) + \Delta t\,\boldsymbol{A}\,\mathbf{x}(\tau) && (6.29)\\ &= (\mathbf{I} + \boldsymbol{A}\Delta t)\,\mathbf{x}(\tau). && (6.30) \end{aligned}$$

6.2.4 Modèle d'état complet

Dans cette section, nous décrivons le modèle d'état complet que nous utilisons pour le suivi des signaux routiers. Pour l'estimation de la position du signal, nous avons choisi un modèle à accélération constante présenté à la section 6.2.3. Comme nous l'avons déjà dit, ce modèle s'est révélé suffisamment efficace dans la plupart des cas. La situation dans laquelle le modèle se comporte le plus mal est celle où le signal se trouve au bord de l'image et que le véhicule se déplace à grande vitesse (plus de 90 km/h). Nous également intégré une autre variable importante dans le vecteur d'état : la taille du signal. Nous noterons s_u la largeur et s_v la hauteur du signal. Comme la taille du signal ne varie pas beaucoup d'une image à l'autre, nous avons choisi le modèle le plus simple : un modèle d'ordre zéro. Le suivi est réalisé à l'aide d'un filtre de Kalman.

L'équation du processus est

$$\underbrace{\mathbf{x}_k}_{8\times1} = \underbrace{\boldsymbol{A}_k}_{8\times8}\underbrace{\mathbf{x}_{k-1}}_{8\times1} + \underbrace{\mathbf{w}_{k-1}}_{8\times1} \tag{6.31}$$

$$\begin{bmatrix} u \\ v \\ c_u \\ c_v \\ a_u \\ a_v \\ s_u \\ s_v \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_k \begin{bmatrix} u \\ v \\ c_u \\ c_v \\ a_u \\ a_v \\ s_u \\ s_v \end{bmatrix}_{k-1} + \mathbf{w}_{k-1} \tag{6.32}$$

avec un bruit de covariance

$$\boldsymbol{Q}_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{a_u} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{a_v} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{s_u} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{s_v} \end{bmatrix}_k . \tag{6.33}$$

Nous avons introduit des composantes aléatoires uniquement sur l'accélération et la taille sans aucune corrélation. Après quelques essais sur différents flux vidéo, nous avons choisi $\sqrt{q_{a_u}} = \sqrt{q_{a_v}} = 100$ pixels/s^2 et $\sqrt{q_{s_u}} = \sqrt{q_{s_v}} = 10$ pixels. Le choix de ces valeurs devrait être étudié de manière plus approfondie. Il semblerait a priori logique de faire dépendre ces valeurs de certains paramètres, comme la vitesse du véhicule par exemple.

L'équation de la mesure est

$$\underbrace{\mathbf{z}_k}_{4\times1} = \underbrace{\boldsymbol{H}_k}_{4\times8}\underbrace{\mathbf{x}_{k-1}}_{8\times1} + \underbrace{\mathbf{v}_k}_{4\times1} \tag{6.34}$$

$$\begin{bmatrix} u \\ v \\ s_u \\ s_v \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ c_u \\ c_v \\ a_u \\ a_v \\ s_u \\ s_v \end{bmatrix}_{k-1} + \mathbf{v}_k \tag{6.35}$$

avec un bruit de covariance

$$\boldsymbol{R}_k = \begin{bmatrix} r_u & 0 & 0 & 0 \\ 0 & r_v & 0 & 0 \\ 0 & 0 & r_{s_u} & 0 \\ 0 & 0 & 0 & r_{s_v} \end{bmatrix}_k . \tag{6.36}$$

Nous avons choisi $\sqrt{r_u} = 2$ pixels, $\sqrt{r_v} = 5$ pixels et $\sqrt{r_{s_u}} = \sqrt{r_{s_v}} = 4$ pixels. Nous avons adopté ces faibles valeurs car notre détecteur est très précis. Le choix d'une valeur plus élevée pour $\sqrt{r_v}$ que pour $\sqrt{r_u}$ n'est pas sans raison. Bien que notre détecteur soit très précis, les irrégularités de la route provoquent des vibrations et des secousses de la caméra. La trajectoire de la cible sur le plan image n'est donc pas parfaitement « lisse » mais oscille autour de la trajectoire qui serait obtenue si la caméra était parfaitement stabilisée. Nous pouvons donc voir cet écart comme un bruit de mesure. Cette valeur plus élevée fait en sorte que le filtre de Kalman accorde moins de confiance à la mesure verticale qu'à la mesure horizontale. De nouveau, le choix de ces valeurs devrait être plus réfléchi. On sens par exemple intuitivement que l'erreur sur la taille est sans doute d'autant plus grande que la taille détectée est grande.

6.3 Association de données

Supposons que nous soyons en train de poursuivre N cibles. Nous prédisons d'abord leur état dans l'image courante et obtenons N vecteurs d'états $\{\hat{\mathbf{x}}^{-i}\}_{i=1}^{N}$. Puis, nous détectons les signaux dans toute l'image et obtenons M observations $\{\mathbf{z}^j\}_{j=1}^{M}$. Il nous faut maintenant décider du sort de chacune de ces observations : c'est l'association de données. Plusieurs algorithmes existent :

1. La méthode du plus proche voisin est la méthode la plus simple et la plus utilisée. L'objectif est de trouver l'association la plus probable entre une cible suivie et une observation. La décision sur l'association est immédiate.

2. Le filtre à association de données probabilistes (PDAF [2]) utilise toutes les observations pour mettre à jour le filtre de Kalman.
3. Le filtre joint à association de données probabilistes (JPDAF [3]) est une extension de PDAF pour le suivi multicible.
4. Le filtre à hypothèses multiples (MHT [4]) retient toutes les hypothèses d'association possible jusqu'à ce que de nouvelles observations permettent de décider quelles hypothèses sont à abandonner ou à conserver.

Encore d'autres algorithmes existent tels le filtre probabiliste à hypothèses Multiples (PMHT [5]) et le filtre à particules pour le suivi multicible. Nous ne détaillerons pas tous ces algorithmes ici, la littérature est abondante sur le sujet (voir [RH01, Cou03] pour une présentation synthétique de ces méthodes).

Nous avons basé notre algorithme sur la méthode du plus proche voisin, facile à implémenter, qui consiste à choisir l'association la plus probable. Pour la cible i, il faut trouver l'observation $\mathbf{z}_k^*$ qui maximise

$$p(\mathbf{z}_k \mid \mathbf{z}_{0:k-1}^i). \tag{6.37}$$

Cette densité de probabilité est simple à calculer avec le filtre de Kalman. En effet, nous connaissons

$$p(\mathbf{x}_k^i \mid \mathbf{z}_{0:k-1}^i) \tag{6.38}$$

qui est une gaussienne de moyenne $\hat{\mathbf{x}}_k^{-i}$ et de covariance $\boldsymbol{P}_k^{-i}$ et nous savons comment les observations sont obtenues à partir des états par l'équation de mesure

$$\mathbf{z}_k = \boldsymbol{H}_k \mathbf{x}_k + \mathbf{v}_k. \tag{6.39}$$

La distribution gaussienne traverse donc un opérateur linéaire et nous avons

$$p(\mathbf{z}_k \mid \mathbf{z}_{0:k-1}^i) = \mathcal{N}(\boldsymbol{H}_k \hat{\mathbf{x}}_k^{-i},\, \boldsymbol{H}_k \boldsymbol{P}_k^{-i} \boldsymbol{H}_k^T + \boldsymbol{R}_k). \tag{6.40}$$

Si toutes les observations sont perturbées par un bruit de même covariance $\boldsymbol{R}_k$, l'observation $\mathbf{z}_k^*$ la plus probable est celle qui est la plus proche de $\boldsymbol{H}_k \hat{\mathbf{x}}_k^{-i}$ selon la distance de Mahalanobis,

$$\mathbf{z}_k^* = \mathbf{z}_k^{\arg\min_{j \in \{1,\dots,M\}} d_{ij}^2} \tag{6.41}$$

avec

$$d_{ij}^2 = (\mathbf{z}_k^j - \boldsymbol{H}_k \hat{\mathbf{x}}_k^{-i})^T \, (\boldsymbol{H}_k \boldsymbol{P}_k^{-i} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \, (\mathbf{z}_k^j - \boldsymbol{H}_k \hat{\mathbf{x}}_k^{-i}). \tag{6.42}$$

En n'utilisant que cette dernière relation, il se peut qu'une observation soit associée à plusieurs cibles (par exemple dans le cas où $N = 2$ et $M = 1$). Dans ce cas, nous conservons l'association dont la distance de Mahalanobis est la plus faible.

[2] *Probabilistic Data Association Filter.*
[3] *Joint Probabilistic Data Asociation Filter.*
[4] *Multiple Hypothesis Tracking.*
[5] *Probabilistic Multiple Hypothesis Tracking.*

Pratiquement, nous construisons une matrice de distances

$$\boldsymbol{D_M} = \begin{bmatrix} d_{11}^2 & \cdots & d_{1M}^2 \\ \vdots & \ddots & \vdots \\ d_{N1}^2 & \cdots & d_{NM}^2 \end{bmatrix}. \tag{6.43}$$

Il faut ensuite sélectionner dans chaque ligne de cette matrice la distance la plus petite mais en même temps il ne peut y avoir qu'un seul élément sélectionné par colonne car une observation ne peut être associée qu'à une seule cible. Le problème revient donc à choisir $n = \min(N, M)$ éléments telle que leur somme soit minimale. C'est un problème d'optimisation combinatoire bien connu qui peut être résolu en un temps polynomial avec l'algorithme hongrois (algorithme de Munkres).

Une suite d'observations et de prédictions (dans le cas d'une observation manquante) associées entre elles constitue une piste. La figure 6.3 illustre la validation, la création et l'abandon des pistes.

Validation d'une piste. Une piste est considérée comme étant effectivement la trajectoire d'un signal routier si la longueur de la piste est plus grande ou égale à un certain seuil α. À partir de ce moment, nous sauvons toutes les sous-images relatives à ce signal (depuis le début de la piste) et continuons de le suivre.

Création de nouvelles pistes. Les observations qui n'ont pas été associées (si $M > N$) et celles qui ont été associées mais dont la distance dépasse un certain seuil (observation trop éloignée de la prédiction) sont considérées comme de nouveaux signaux à suivre, à condition que ces observations ne soient pas trop proches des autres observations déjà associées.

Abandon d'une piste. Si un signal n'a plus été associé à une observation depuis un certain nombre d'images, c'est-à-dire si le nombre d'observations manquantes successives dépasse un seuil β), nous arrêtons de le suivre.

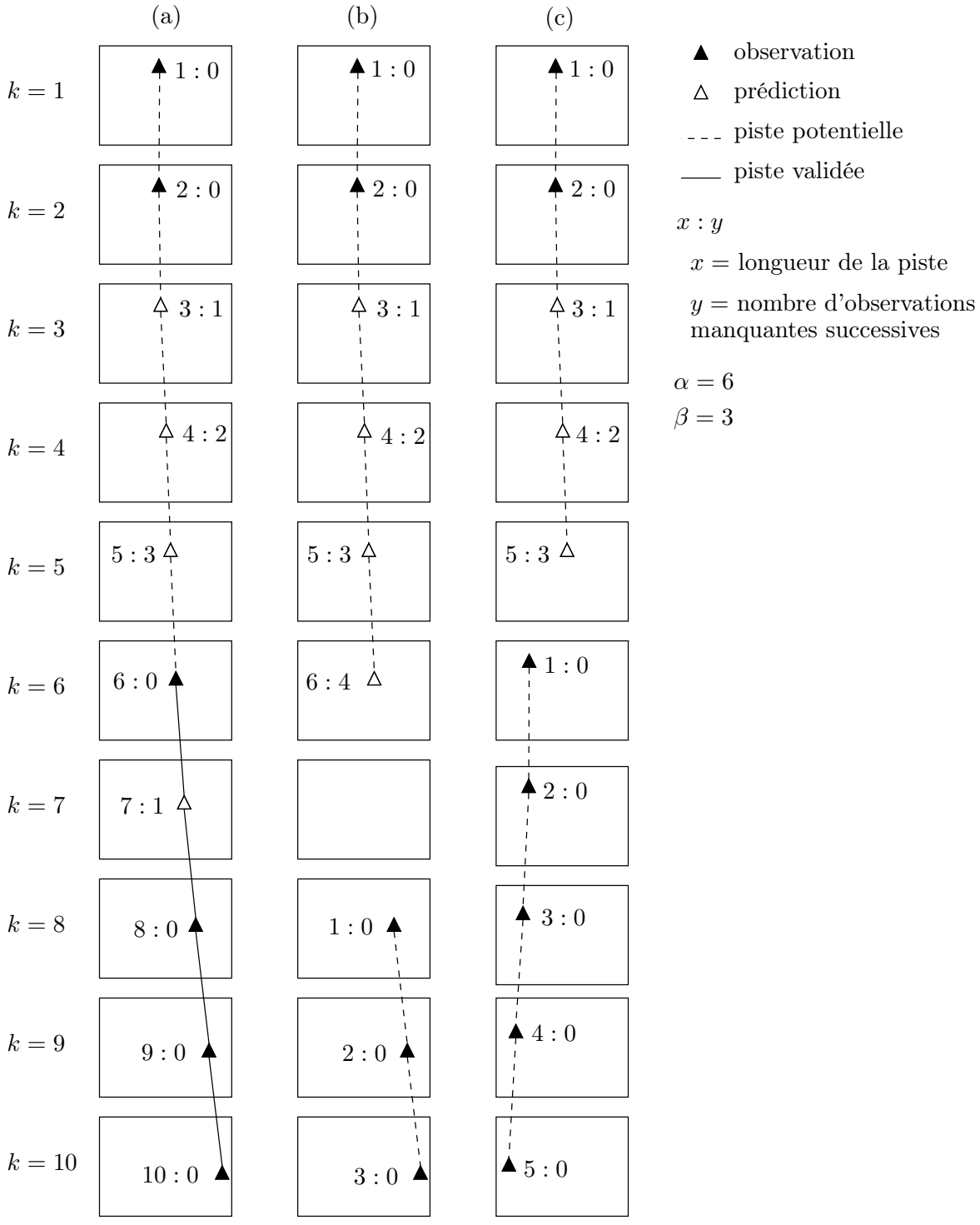


FIG. 6.3 – Validation, création et abandon de pistes. (a) La piste est validée à partir de la 6e image car sa longueur est plus grande ou égale à $\alpha = 6$ et une observation a été associée. (b) La piste est abandonnée à la 6e image car le nombre d'observations manquantes successives est plus grande que $\beta = 3$. Une nouvelle piste est créée à la 8e image. (c) La piste est abandonnée à la 5e image car l'observation dans l'image suivante est trop éloignée de la prédiction. Une nouvelle piste est créée à partir de la 6e image.

Chapitre 7

Reconnaissance des signaux

La détection, la localisation et le suivi ayant été réalisés sur des signaux triangulaires, nous allons faire de même pour la reconnaissance. Nous disposons dans notre base de données de 798 signaux triangulaires dispersés en 16 classes (voir tableau D.1 en annexe). Comme le nombre de signaux par classe doit être assez important pour obtenir des résultats pertinents, nous allons utiliser les 5 classes majoritaires. Chacune d'elle contient au moins 69 images. L'ensemble d'apprentissage ainsi que l'ensemble de test sont constitués de 34 images de chacune des 5 classes. Afin d'estimer l'influence du fond de l'image sur la reconnaissance, nous avons créé de nouveaux ensembles contenant les même images mais dépourvues de leur fond. Vu que nous ne travaillons que sur des signaux triangulaires, l'information est entièrement contenue dans le pictogramme des signaux, nous avons donc également généré des ensembles ne contenant que la partie intérieure des signaux. La figure 7.1 montre une image de chacune des classes que nous avons utilisées lors de nos tests.

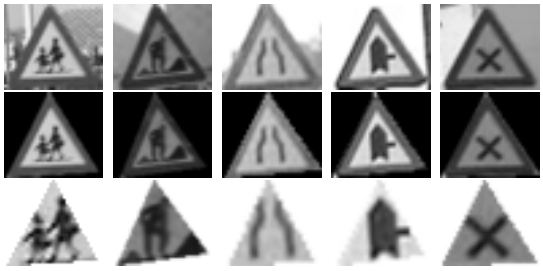


FIG. 7.1 – Classes des signaux utilisées pour la classification. Chacune des 5 colonnes représente respectivement 3 signaux de classe A23, A31, A7_b, B15 et B17. Chacune des lignes correspond respectivement aux images originales redimensionnées, aux images dépourvues de leur fond (seul le triangle subsiste) et aux pictogrammes (seul le triangle intérieur subsiste).

S'il s'avère que la classification des pictogrammes ou des images dépourvues de leur fond donne de meilleurs résultats que sur les images brutes, le module de reconnaissance devra extraire automatiquement des zones triangulaires des images fournies

par le module de détection. Cette tâche peut être réalisée en utilisant par exemple le détecteur optimal de coins de Rangarajan et al. [RSvB88].

Nous avons choisi de tester deux méthodes de classification. La première s'inspire de celle de Bahlmann et al. [BZR+05] exposée dans notre étude bibliographique. Elle se base sur un modèle génératif en utilisant une densité de probabilité normale unimodale pour chaque classe. La classification étant ensuite effectuée en maximisant une fonction de vraisemblance. Les raisons pour lesquelles nous avons choisi cette méthode et la manière dont nous l'avons implémentée sont décrites à la section 7.1.

La seconde méthode repose sur les travaux de Marée [Mar05] qui utilise une extraction de fenêtres aléatoires et les ensembles d'arbres extrêmement aléatoires (*Extra-Trees*) introduits par Geurts dans [GEW06]. La méthode a fait ses preuves sur plusieurs problèmes de classification : objets, visages, chiffres manuscrits, immeubles, ... Nous utilisons cette méthode pour deux raisons. Premièrement, elle n'a pas encore été utilisée, à notre connaissance, pour la classification de signaux routiers, mais la méthode étant générale, elle pourrait très bien s'appliquer à notre problème. Ensuite, une implémentation est disponible dans le logiciel PiXiT [PiX] de la société PEPITe. Nous discutons des résultats obtenus dans la section 7.2.

7.1 Modèle génératif et plus proche voisin

Dans cette partie, nous allons nous inspirer des travaux de Bahlmann et al. [BZR+05] pour classer nos signaux. Pour rappel, cette méthode se base sur un modèle génératif en utilisant une densité de probabilité normale unimodale $p(\mathbf{a} \mid y) = \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$ pour chaque classe y. Le vecteur d'attributs $\mathbf{a} \in \mathcal{R}^{25}$ est obtenu par une analyse discriminante linéaire (LDA [1]) sur les attributs initiaux qui sont les pixels en niveaux de gris. Étant donné une suite d'observations $\{\mathbf{a}^t\}_{t=1}^{t_0}$ relatives au même signal dans t_0 images successives, la classification est effectuée en maximisant une log-vraisemblance pondérée,

$$\hat{y}(o) = \underset{j \in \{1,2,...,M\}}{\arg\max} \sum_{t=1}^{t_0} \pi_t \log p(\mathbf{a}^t(o) \mid y). \tag{7.1}$$

Les π_t sont des poids croissants. Cela permet d'accorder moins de poids à la reconnaissance des premières images qui sont plus petites (on suppose que le véhicule avance vers le signal). Cette méthode a l'avantage d'être intuitive, conceptuellement simple et de donner de bons résultats (6 % d'erreur sur des images isolées d'après [BZR+05]).

Nous avons testé la méthode sur des images isolées donc avec $t_0 = 1$. Nous n'avons malheureusement pas eu le temps de tester la méthode sur des suites d'images fournies par notre détecteur. Cela aurait demandé un plus grand nombre de séquences vidéo de test suivi d'un étiquetage manuel des classes. La deuxième raison est que nous avons

[1] LDA : Linear Discriminant Analysis.

également testé la méthode du plus proche voisin en considérant tous les objets de l'ensemble d'apprentissage.

Contrairement à [BZR+05], nous n'avons pas pratiqué directement une analyse discriminante LDA sur les attributs initiaux. La raison est que nous n'avions pas assez d'exemples dans notre ensemble d'apprentissage, ce qui nous laissait une matrice de covariance singulière. Ce problème est discuté plus en détails à la section 7.1.2. Afin de remédier à ce problème, nous avons procédé à une analyse en composantes principales (PCA [2]) pour réduire la dimension de l'espace des attributs avant d'effectuer LDA. C'est pour cette raison que nous parlerons d'abord de PCA puis de LDA dans la suite de ce travail.

Nous avons également testé deux autres méthodes de classification dans les espaces PCA et PCA+LDA :

- le plus proche voisin en considérant tous les objets de l'ensemble d'apprentissage ;
- le plus proche voisin en considérant la moyenne de chaque classe (la plus proche classe).

Nous avons implémenté les algorithmes dans le langage GNU Octave [Oct].

7.1.1 Analyse en composantes principales (PCA)

Construction d'un espace approprié à la classification. Classer des objets revient explicitement ou non à trouver des hypersurfaces de décision séparant chacune des classes dans un espace de plus ou moins grande dimension selon les attributs caractérisant ces objets. Dans le cas des images, les attributs considérés sont souvent des grandeurs prises à un endroit précis de l'image, par exemple la composante rouge du pixel (x, y). L'information contenue dans l'image est directement liée à sa taille. Dès lors, l'espace des attributs est relativement grand. La vitesse d'un l'algorithme d'apprentissage ainsi que la rapidité avec laquelle va être classé un objet sont d'autant plus élevées que l'espace des attributs est petit. Il est donc souvent intéressant de pouvoir représenter les objets dans un espace réduit sans pour autant perdre trop d'information. C'est ce que font PCA et LDA mais d'une manière différente. Pour procéder à une réduction de dimension, il faut tout d'abord exprimer les objets dans un même espace, ce que nous faisons en redimensionnant les images en 37×32 pixels [3] et en prenant comme attributs les pixels en niveaux de gris, la couleur n'étant ici d'aucune utilité pour la classification (l'information étant entièrement contenue dans le pictogramme).

L'analyse en composantes principales, aussi appelée transformation ou expansion de Karhunen-Loéve, est une technique classique de réduction de dimension. À partir d'un espace d'attributs de dimension D dans lequel les objets sont caractérisés, l'objectif est de trouver un sous-espace de dimension $K \ll D$ représentant au mieux

[2] PCA : Principal Components Analysis.

[3] Ce rapport est équivalent à celui du plus petit rectangle dans lequel est inscrit un triangle équilatéral.

ces objets. Pour cela, PCA construit de nouveaux attributs orthogonaux qui sont des fonctions linéaires des premiers et qui capturent le maximum de variance de l'ensemble des objets. Pour différentes explications sur le sujet on pourra consulter [FP03, section 22.3], [Weh00, section 6.2] ou [Mur00].

Supposons que nous ayons un ensemble de N objets caractérisés par D attributs réels. Cet ensemble est donc représenté par les vecteurs $\{\mathbf{a}^i\}_{i=1}^N$ de $\mathcal{R}^D$. Soient $\mathbf{\Sigma}$ la matrice de covariance de ces vecteurs et $\boldsymbol{\mu}$ leur moyenne. Afin de trouver un nouvel attribut qui explique au mieux la variation des données autour de la moyenne, nous projetons les objets sur un vecteur unitaire $\mathbf{u}$ que nous pouvons interpréter comme un nouvel attribut. La valeur de ce nouvel attribut pour l'objet i est $u(\mathbf{a}^i) = \mathbf{u}^T(\mathbf{a}^i - \boldsymbol{\mu})$. Sa moyenne est nulle et sa variance vaut

$$\begin{aligned}
\text{var}(u) &= \frac{1}{N-1}\sum_{i=1}^{N} u^2(\mathbf{a}^i) && (7.2)\\
&= \frac{1}{N-1}\sum_{i=1}^{N} \mathbf{u}^T(\mathbf{a}^i - \boldsymbol{\mu})(\mathbf{a}^i - \boldsymbol{\mu})^T\mathbf{u} && (7.3)\\
&= \mathbf{u}^T\left\{\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{a}^i - \boldsymbol{\mu})(\mathbf{a}^i - \boldsymbol{\mu})^T\right\}\mathbf{u} && (7.4)\\
&= \mathbf{u}^T\mathbf{\Sigma}\,\mathbf{u} && (7.5)
\end{aligned}$$

Trouver la direction qui capture le maximum de variance revient donc à maximiser la fonction objectif $\mathbf{u}^T\mathbf{\Sigma}\,\mathbf{u}$ sous la contrainte $\mathbf{u}^T\mathbf{u} = 1$. On trouve la solution en annulant le gradient du lagrangien,

$$\boldsymbol{\nabla}_{\mathbf{u}}\mathcal{L}(\mathbf{u},\lambda) = 0 \;\Leftrightarrow\; \boldsymbol{\nabla}_{\mathbf{u}}\left(\mathbf{u}^T\mathbf{\Sigma}\,\mathbf{u} - \lambda(\mathbf{u}^T\mathbf{u} - 1)\right) = 0 \;\Leftrightarrow\; \mathbf{\Sigma}\,\mathbf{u} = \lambda\,\mathbf{u} \qquad (7.6)$$

puis en injectant le résultat dans 7.5,

$$\text{var}(u) = \mathbf{u}^T\mathbf{\Sigma}\,\mathbf{u} = \mathbf{u}^T\lambda\,\mathbf{u} = \lambda. \qquad (7.7)$$

La solution est donc le vecteur propre $\mathbf{v}_1$ de $\mathbf{\Sigma}$ relatif à la plus grande des valeurs propres λ_1 [4]. Après projection des objets dans l'espace de dimension $D-1$ perpendiculaire à ce vecteur propre, on peut montrer que la seconde direction qui capture le maximum de variance est le vecteur propre $\mathbf{v}_2$ relatif à la deuxième plus grande valeur propre, λ_2, et ainsi de suite [Mur00]. La base $\{\mathbf{v}_i\}_{i=1}^K$ est la base orthogonale de dimension K qui préserve le plus de variance. Chaque objet est représenté dans le nouvel espace par $\tilde{\mathbf{a}} = \mathbf{\Phi}^T(\mathbf{a} - \boldsymbol{\mu})$ avec $\mathbf{\Phi} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$. L'objet reconstruit à partir de sa projection est $\mathbf{a}_r = \mathbf{\Phi}\,\tilde{\mathbf{a}} + \boldsymbol{\mu}$. On peut montrer que l'erreur quadratique moyenne de reconstruction qui est la moyenne des distances euclidiennes au carré

[4] La matrice de covariance étant hermitienne, ses vecteurs propres $\{\mathbf{v}_i\}_{i=1}^D$ sont orthogonaux et ses valeurs propres λ_i sont réelles. Elle est diagonalisable, $\mathbf{\Sigma} = \boldsymbol{V}\,\mathbf{\Lambda}\,\boldsymbol{V}^T$ avec $\boldsymbol{V} = (\mathbf{v}_1\,\mathbf{v}_2\,\ldots\,\mathbf{v}_D)$ et $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$. De plus, $\mathbf{\Sigma}$ est réelle, les vecteurs propres sont donc réels. Aussi, $\mathbf{\Sigma}$ est définie semi-positive, ce qui implique que ses valeurs propres sont non négatives.

entre les objets initiaux et les objets reconstruits est égale à la somme des valeurs propres n'ayant pas participé à la projection [Weh00, section 6.2],

$$\epsilon = \sum_{i=K+1}^{D} \lambda_i. \tag{7.8}$$

Dans le nouvel espace, la covariance de l'ensemble des objets vaut

$$\begin{aligned}
\tilde{\boldsymbol{\Sigma}} &= \frac{1}{N-1}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^T \quad \text{avec} \quad \tilde{\boldsymbol{A}} = (\tilde{\mathbf{a}}^1, \tilde{\mathbf{a}}^2, \ldots, \tilde{\mathbf{a}}^N) && (7.9)\\
&= \frac{1}{N-1}(\boldsymbol{\Phi}^T\boldsymbol{A}_c)(\boldsymbol{\Phi}^T\boldsymbol{A}_c)^T \quad \text{avec} \quad \boldsymbol{A}_c = (\mathbf{a}^1 - \boldsymbol{\mu}, \ldots, \mathbf{a}^N - \boldsymbol{\mu}) && (7.10)\\
&= \frac{1}{N-1}\boldsymbol{\Phi}^T\boldsymbol{A}_c\boldsymbol{A}_c^T\boldsymbol{\Phi} && (7.11)\\
&= \boldsymbol{\Phi}^T\boldsymbol{\Sigma}\boldsymbol{\Phi} && (7.12)
\end{aligned}$$

Dans le cas où les objets sont des images et les attributs les pixels, il est d'usage d'appeler les vecteurs propres $\{\mathbf{v}_i\}_{i=1}^{D}$ images propres (*eigenimages*). Le nombre d'images propres K doit être choisi tel que l'erreur moyenne de reconstruction ne soit pas trop grande afin de conserver un maximum d'information. Comme les valeurs de pixels adjacents sont fortement corrélées, très peu d'images propres sont généralement nécessaires à une bonne reconstruction et la réduction de la dimension par rapport à l'espace de départ est alors conséquente. Remarquons que si $N \leq D$, seules $N-1$ images propres et l'image moyenne permettent de reconstruire parfaitement le jeu de données. La matrice de covariance $\boldsymbol{\Sigma}$ est de rang $N-1$ au maximum et seules ses $N-1$ premières valeurs propres peuvent être non nulles. Dans notre cas, $N = 5 \times 34 = 170$ et $D = 37 \times 32 = 1184$. Nous avons retenu les 24 premières images propres qui reconstruisent notre ensemble d'apprentissage avec une erreur moyenne de $8,4$ % pour les images originales, $7,6$ % sur les images dépourvues de leur fond et $7,3$ % sur les pictogrammes.

Les figures 7.2, 7.4 et 7.6 montrent les premières composantes principales et le graphe de dispersion de l'ensemble d'entraînement obtenus respectivement avec les images originales, les images dépourvues de leur fond et les pictogrammes. Les images les mieux et les moins bien reconstruites apparaissent sur les figures 7.3, 7.5 et 7.7. Nous pouvons remarquer sur les graphes de dispersion que dans les trois cas, les classes sont mal séparées si on ne tient compte que des deux premières composantes principales, ce qui laisse présumer que nous n'obtiendrons pas de bons résultats pour la classification en n'utilisant que deux images propres. La classification dans l'espace PCA est discutée à la section 7.1.3 mais les résultats chiffrés peuvent d'ors et déjà être observés sur les figures 7.13, 7.14 et 7.15 à partir de la page 111.

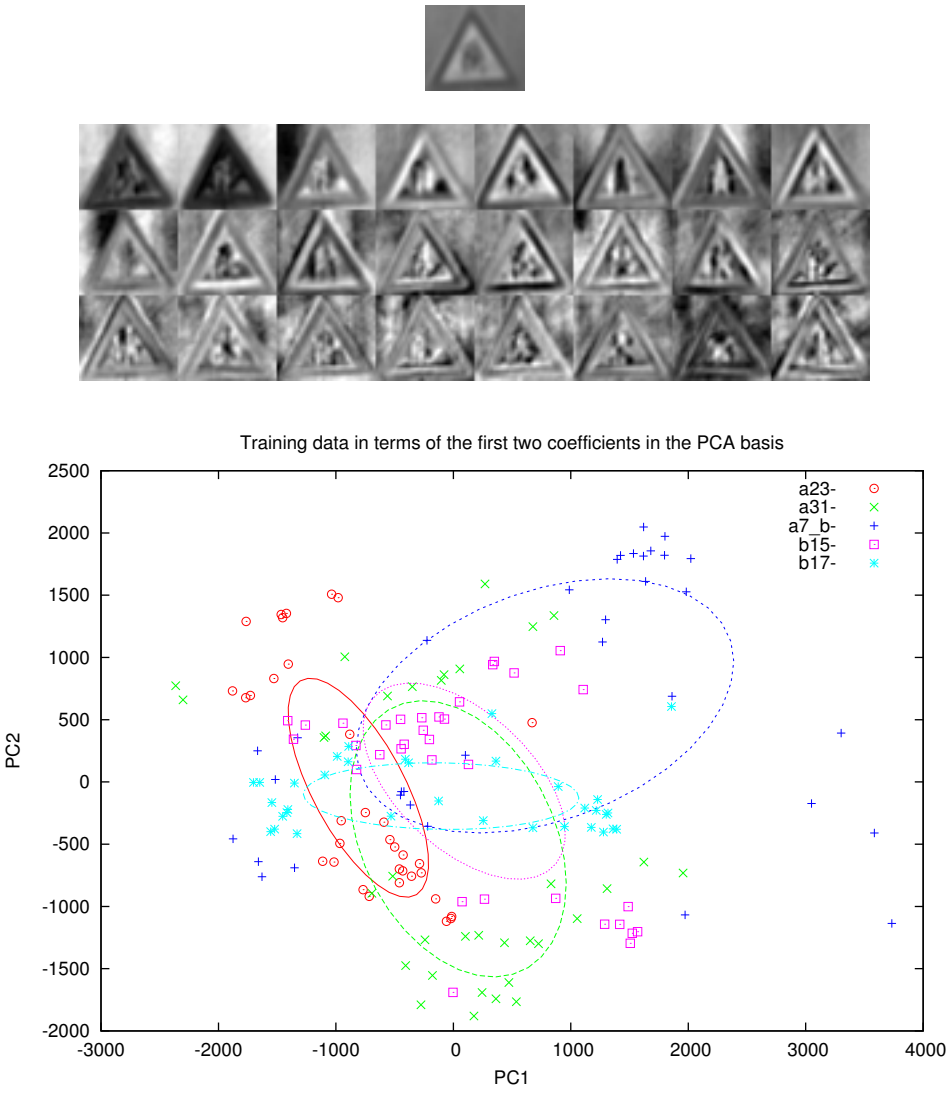


FIG. 7.2 – En haut, l'image moyenne et les 24 premiers signaux propres obtenus à partir des images originales. En dessous, le graphe de dispersion en fonction des deux premières composantes principales et les ellipses de covariance.

FIG. 7.3 – La ligne du haut montre l'image la mieux reconstruite avec les 24 premières composantes principales (l'image de gauche est l'originale). La ligne du bas montre l'image la moins bien reconstruite.

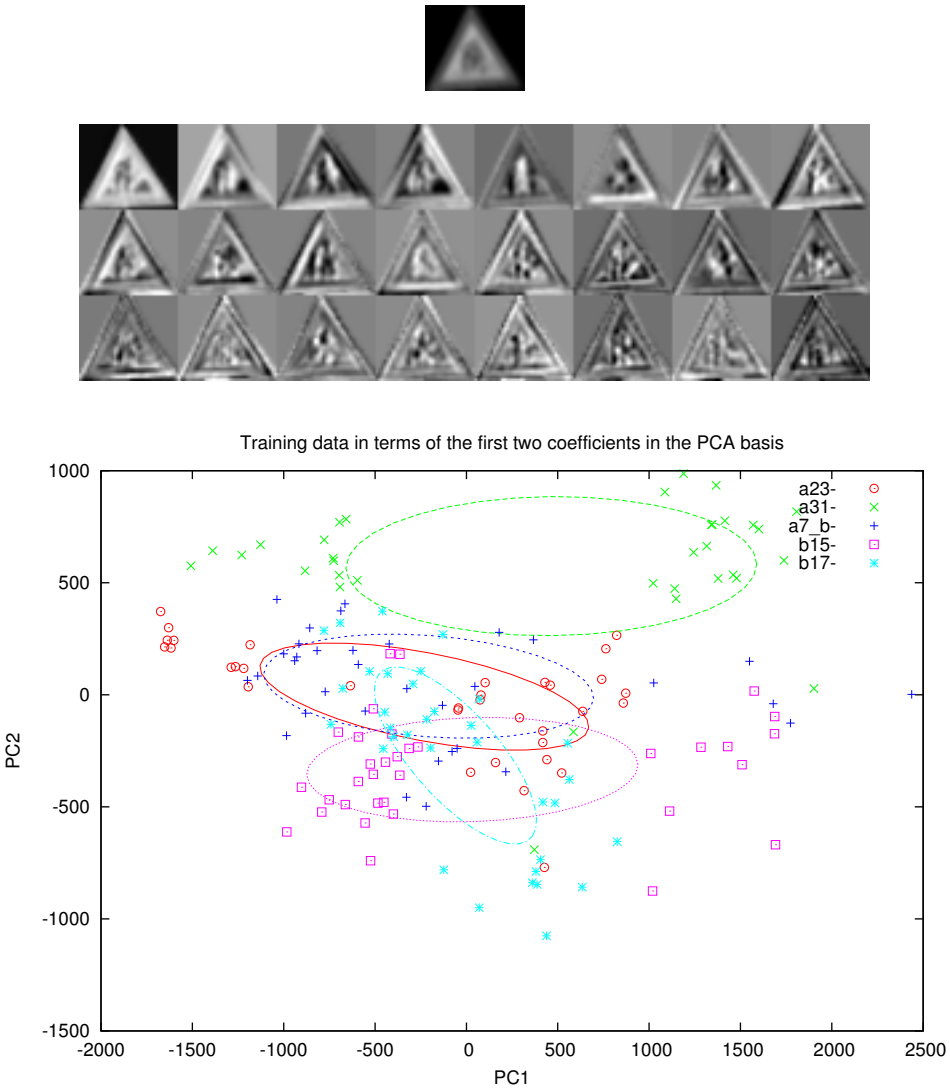


FIG. 7.4 – En haut, l'image moyenne et les 24 premiers signaux propres obtenus à partir des images dépourvues de leur fond. En dessous, le graphe de dispersion en fonction des deux premières composantes principales et les ellipses de covariance.

FIG. 7.5 – La ligne du haut montre l'image dénuée de son fond la mieux reconstruite avec les 24 premières composantes principales (l'image de gauche est l'originale). La ligne du bas montre l'image la moins bien reconstruite.

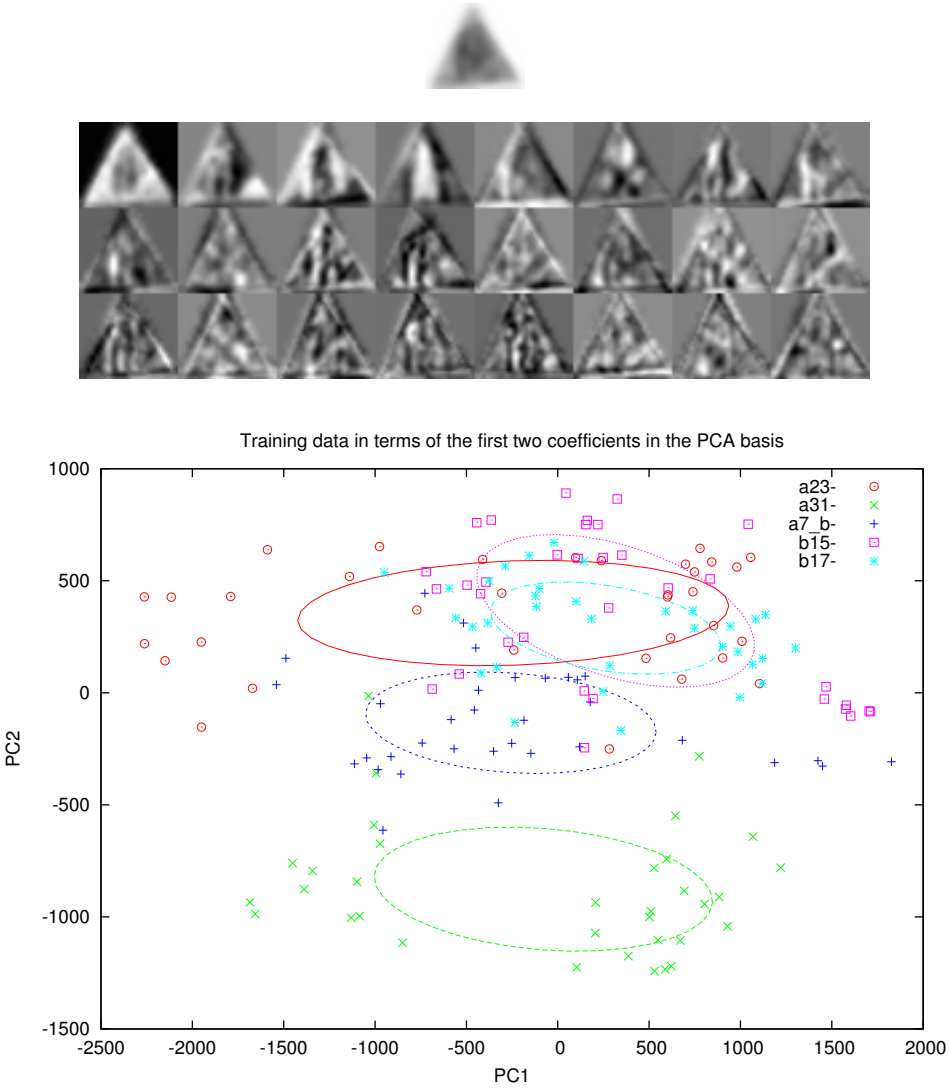


FIG. 7.6 – En haut, l'image moyenne et les 24 premiers signaux propres obtenus à partir des pictogrammes. En dessous, le graphe de dispersion en fonction des deux premières composantes principales et les ellipses de covariance.

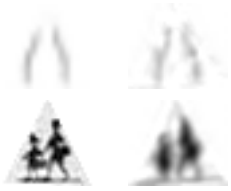


FIG. 7.7 – La ligne du haut montre le pictogramme le mieux reconstruit avec les 24 premières composantes principales (l'image de gauche est l'originale). La ligne du bas montre l'image la moins bien reconstruite.

7.1.2 Analyse discriminante linéaire (LDA)

PCA fournit un sous-espace de dimension donnée qui conserve au mieux la variance d'un ensemble d'objets de grande dimension. Cependant, cela ne garanti pas que ce sous-espace est adapté à la classification. L'analyse discriminante linéaire, aussi connue sous le nom d'analyse discriminante de FISHER (FDA [5]) est largement utilisée dans la reconnaissance de formes. L'objectif de LDA est de trouver un sous-espace qui discrimine au mieux les classes dans le sens où elle maximise le rapport variance inter-classe sur variance intra-classe [6]. Pour différentes explications sur le sujet on pourra consulter [FP03, section 22.3], [DH00, section 3.8] ou [Mur00].

Supposons que nous ayons un ensemble de M classes, la classe j contenant N_j objets chacun caractérisé par D attributs réels. Le nombre total d'objets est N. Si $\boldsymbol{\mu}_j$ est la moyenne de la classe j, la moyenne de la moyenne des classes est

$$\boldsymbol{\mu} = \sum_{j=1}^{M} \frac{N_j}{N} \boldsymbol{\mu}_j \tag{7.13}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}^i. \tag{7.14}$$

On définit la matrice de covariance inter-classe $\boldsymbol{S}_B$ (*between-class scatter matrix*) par

$$\boldsymbol{S}_B = \sum_{j=1}^{M} \frac{N_j}{N} (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T \tag{7.15}$$

et la covariance [7] intra-classe $\boldsymbol{S}_W$ (*within-class scatter matrix*) par la moyenne des covariances $\boldsymbol{\Sigma}_j$ de chaque classe ($\mathbf{a}^{i|j}$ représente l'objet i de la classe j),

$$\boldsymbol{S}_W = \sum_{j=1}^{M} \frac{N_j}{N} \boldsymbol{\Sigma}_j \tag{7.16}$$

$$= \sum_{j=1}^{M} \frac{N_j}{N} \frac{1}{N_j} \sum_{i=1}^{N_j} (\mathbf{a}^{i|j} - \boldsymbol{\mu}_j)(\mathbf{a}^{i|j} - \boldsymbol{\mu}_j)^T \tag{7.17}$$

$$= \frac{1}{N} \sum_{j=1}^{M} \sum_{i=1}^{N_j} (\mathbf{a}^{i|j} - \boldsymbol{\mu}_j)(\mathbf{a}^{i|j} - \boldsymbol{\mu}_j)^T. \tag{7.18}$$

Avec ces définitions, la covariance totale $\boldsymbol{S}_T$ de l'ensemble est égale à la somme des

[5] FDA : FISHER Discriminant Analysis.

[6] Certains auteurs utilisent le terme MDA pour *multiple discriminant analysis* et préfèrent conserver LDA pour le cas où il n'y a que deux classes.

[7] Rigoureusement, nous devrions parler de matrices de dispersion car elles ne sont qu'une estimation de la covariance des distributions sous-jacentes.

covariances intra-classe et inter-classe[8],

$$\begin{aligned}
\boldsymbol{S}_T &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{a}^i-\boldsymbol{\mu})(\mathbf{a}^i-\boldsymbol{\mu})^T && (7.19)\\
&= \frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{N_j}(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j+\boldsymbol{\mu}_j-\boldsymbol{\mu})(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j+\boldsymbol{\mu}_j-\boldsymbol{\mu})^T && (7.20)\\
&= \frac{1}{N}\sum_{j=1}^{M}\left(\sum_{i=1}^{N_j}(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)^T+\sum_{i=1}^{N_j}(\boldsymbol{\mu}_j-\boldsymbol{\mu})(\boldsymbol{\mu}_j-\boldsymbol{\mu})^T\right. && (7.21)\\
&\quad \left.+\underbrace{\sum_{i=1}^{N_j}(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)(\boldsymbol{\mu}_j-\boldsymbol{\mu})^T}_{0}+\underbrace{\sum_{i=1}^{N_j}(\boldsymbol{\mu}_j-\boldsymbol{\mu})(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)^T}_{0}\right) && (7.22)\\
&= \frac{1}{N}\sum_{j=1}^{M}\left(\sum_{i=1}^{N_j}(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)(\mathbf{a}^{i|j}-\boldsymbol{\mu}_j)^T+\sum_{i=1}^{N_j}(\boldsymbol{\mu}_j-\boldsymbol{\mu})(\boldsymbol{\mu}_j-\boldsymbol{\mu})^T\right) && (7.23)\\
&= \boldsymbol{S}_W+\boldsymbol{S}_B. && (7.24)
\end{aligned}$$

Un changement de variable linéaire sur les attributs de l'ensemble des objets $\tilde{\boldsymbol{A}}=\boldsymbol{\Phi}^T\boldsymbol{A}_c$ modifie les covariances inter-classe et intra-classe (voir équation 7.12),

$$\begin{aligned}
\tilde{\boldsymbol{S}}_B &= \boldsymbol{\Phi}^T\boldsymbol{S}_B\boldsymbol{\Phi} && (7.25)\\
\tilde{\boldsymbol{S}}_W &= \boldsymbol{\Phi}^T\boldsymbol{S}_W\boldsymbol{\Phi}. && (7.26)
\end{aligned}$$

Un bon espace de discrimination doit à la fois grouper au mieux les objets appartenant à la même classe et séparer le plus possible les différentes classes. Comme le déterminant d'une matrice est égale au produit de ses valeurs propres et que les valeurs propres d'une matrice de covariance sont égales aux carrés des longueurs des demi-axes de l'hyperellipsoïde de covariance (figure 7.8), une bonne transformation $\boldsymbol{\Phi}^*$ peut être obtenue en choisissant

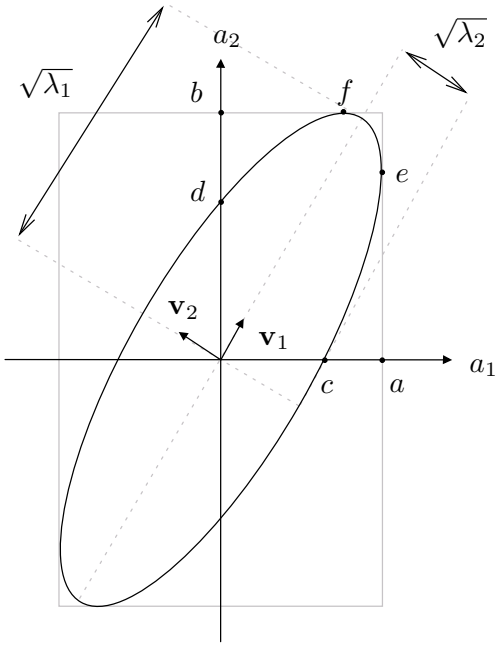
$$\boldsymbol{\Phi}^*=\arg\max_{\boldsymbol{\Phi}}\frac{|\boldsymbol{\Phi}^T\boldsymbol{S}_B\boldsymbol{\Phi}|}{|\boldsymbol{\Phi}^T\boldsymbol{S}_W\boldsymbol{\Phi}|}. \tag{7.27}$$

Les colonnes de la matrice rectangulaire $\boldsymbol{\Phi}^*$ optimale sont les vecteurs propres généralisés du problème ([DH00, section 3.8])

$$\boldsymbol{S}_B\,\mathbf{v}_i=\lambda_i\,\boldsymbol{S}_W\,\mathbf{v}_i. \tag{7.28}$$

Lorsque $\boldsymbol{S}_W$ ou $\boldsymbol{S}_B$ est non singulière, le problème se ramène à un problème aux valeurs propres simple. Cependant, les matrices de covariance sont souvent singulières à

[8] Relation de Huygens.



$$
\begin{aligned}
\boldsymbol{\Sigma} &= \begin{bmatrix} \sigma_1^2 & \rho\,\sigma_1\,\sigma_2 \\ \rho\,\sigma_1\,\sigma_2 & \sigma_2^2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
a &= (\sigma_1,\ 0) \\
b &= (0,\ \sigma_2) \\
c &= (\sigma_1\sqrt{1-\rho^2},\ 0) \\
d &= (0,\ \sigma_2\sqrt{1-\rho^2}) \\
e &= (\sigma_1,\ \rho\,\sigma_2) \\
f &= (\rho\,\sigma_1,\ \sigma_2)
\end{aligned}
$$

FIG. 7.8 – Ellipse de covariance.

cause du faible nombre d'exemples comparé à la dimension de l'espace des attributs [9]. En effet, $\boldsymbol{S}_W$ est la somme de N matrices de rang un, son rang est donc N au maximum. $\boldsymbol{S}_B$ est la somme de M matrices de rang un dont $M-1$ sont indépendantes, son rang est donc $M-1$ au maximum. Dans notre cas, $M=5$, $N=5\times 34=170$ et $D=37\times 32=1184$, $\boldsymbol{S}_W$ et $\boldsymbol{S}_B$ sont donc singulières. Plusieurs techniques existent afin de remédier à ce problème :

- pseudo-inverse LDA [RD98] ;
- LDA régularisée [LPV03] ;
- SVD généralisée [HP04, YJPP04] ;
- LDA à deux étages [BHK97, SW96].

La dernière méthode consiste à utiliser PCA avant LDA afin de réduire la dimension de l'espace des attributs et ainsi de s'assurer que $\boldsymbol{S}_W$ est non singulière. C'est la méthode que nous semble la plus intuitive et que nous avons choisie. De plus, certains travaux [ZCK98, MK01] ont montré que LDA combiné à PCA donnait souvent de meilleurs résultats que LDA seul.

Nous calculons donc les vecteurs propres [10] de $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ où $\boldsymbol{S}_W$ et $\boldsymbol{S}_B$ sont les matrices de covariance de l'ensemble d'apprentissage projeté sur la base PCA $\mathcal{R}^{24}$. Le rang de $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ est au plus $M-1$, seules les $M-1$ premières valeurs propres peuvent être non nulles.

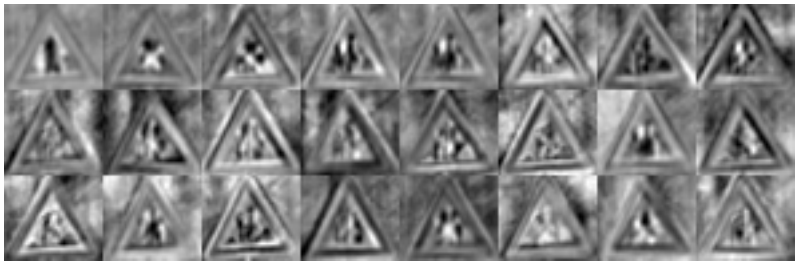
Les figures 7.9, 7.10 et 7.11 montrent les premiers attributs discriminants (reprojetés dans l'espace initial des pixels en niveaux de gris) [11] et le graphe de disper-

[9] Dans la littérature ce problème est souvent appelé *Small Sample Size (SSS) problem*.

[10] Les vecteurs propres ne sont plus nécessairement orthogonaux comme dans PCA car $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ n'est généralement pas symétrique.

[11] Ces attributs sont parfois appelés « images de FISHER ».

sion de l'ensemble d'entraînement obtenus respectivement avec les images originales, les images dépourvues de leur fond et les pictogrammes. On voit clairement sur les graphes de dispersion que les classes sont bien mieux séparées que dans la base PCA de départ. Le taux de classifications correctes devrait être déjà relativement élevé en n'utilisant que les deux premiers facteurs discriminants. La classification dans les espaces PCA et PCA+LDA est discutée à la section suivante.



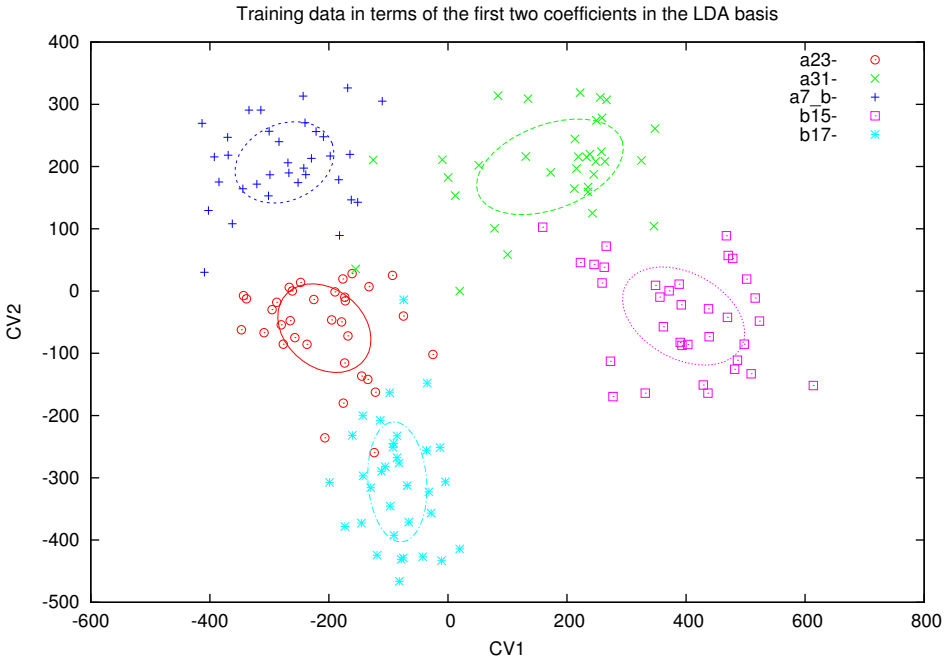


FIG. 7.9 – Images de FISHER obtenues à partir des images originales projetées sur la base PCA et graphe de dispersion en fonction des deux premiers attributs discriminants.

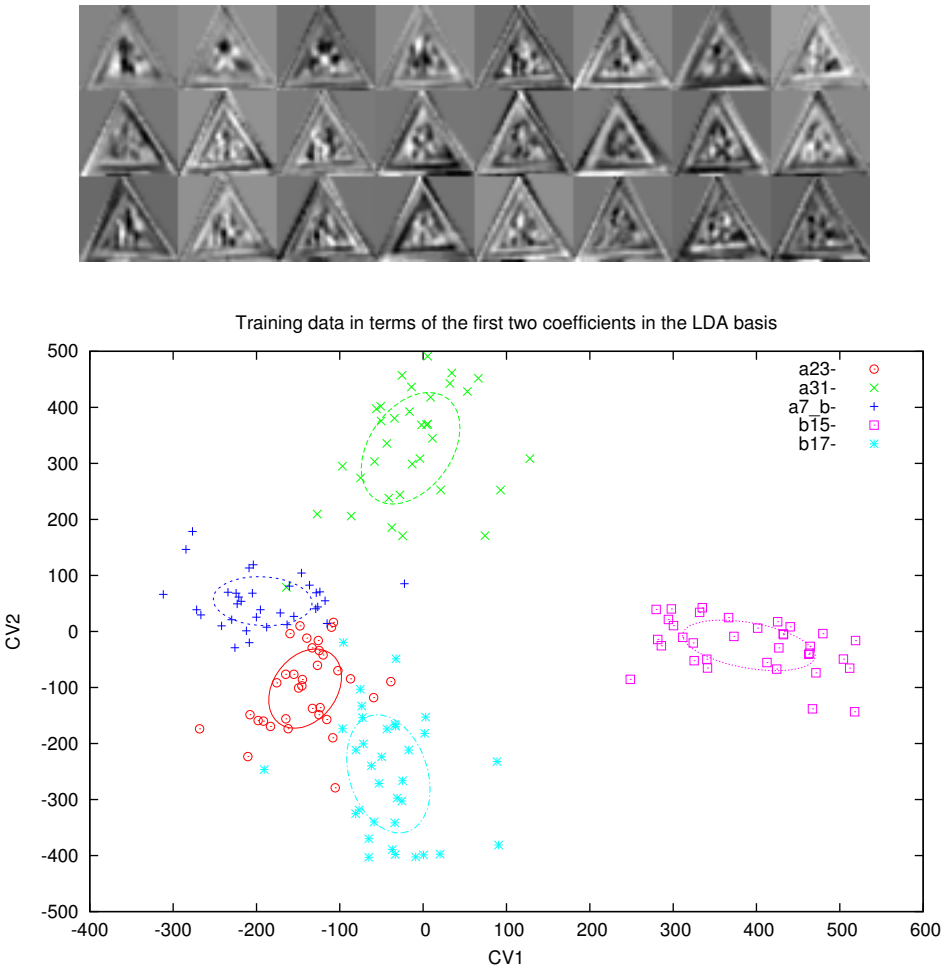


FIG. 7.10 – Images de FISHER obtenues à partir des images dépourvues de leur fond projetées sur la base PCA et graphe de dispersion en fonction des deux premiers attributs discriminants.

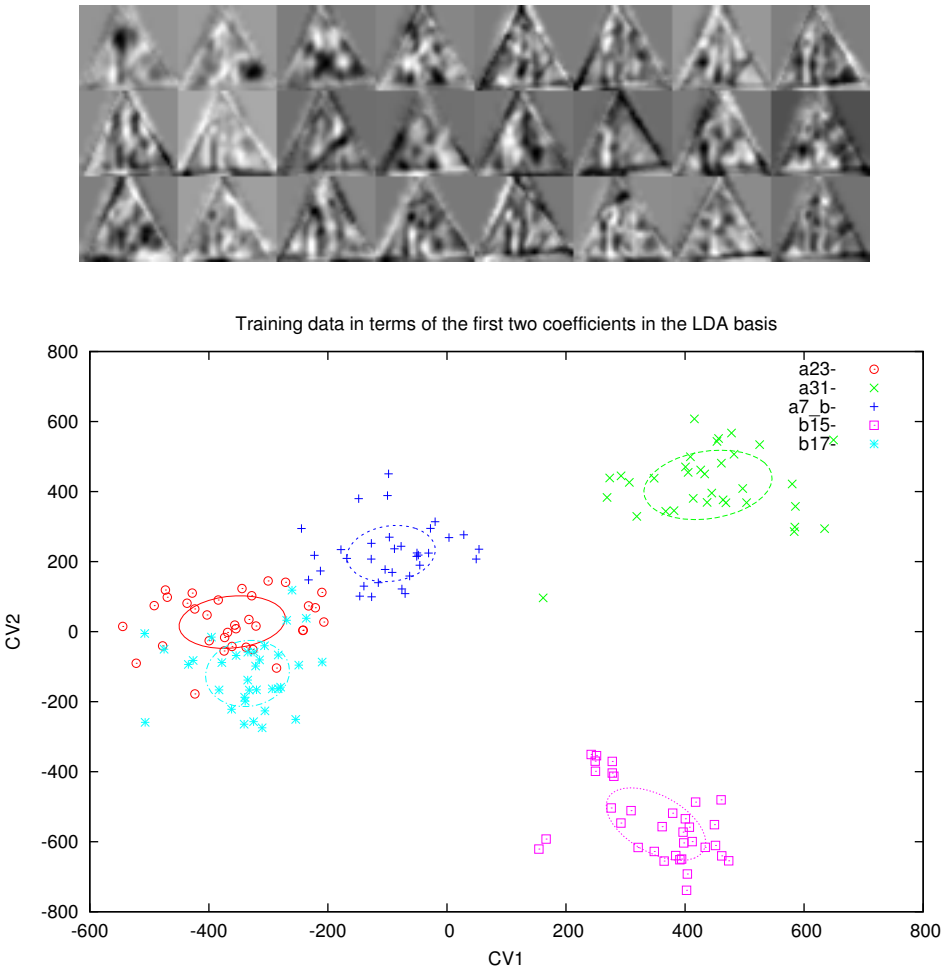


FIG. 7.11 – Images de FISHER obtenues à partir des pictogrammes projetés sur la base PCA et graphe de dispersion en fonction des deux premiers attributs discriminants.

7.1.3 Classification

Nous possédons maintenant l'ensemble d'apprentissage $\mathcal{LS}$ projeté dans les bases PCA et PCA+LDA et les deux transformations $\mathbf{\Phi}_{PCA}^T$ et $\mathbf{\Phi}_{LDA}^T$ permettant de projeter n'importe quel objet dans ces nouveaux espaces et de procéder à une classification. Nous allons tester les performances sur l'ensemble de test $\mathcal{TS}$ composé de 34 images par classe.

Nous définissons la distance quadratique généralisée entre deux vecteurs $\mathbf{x}_1$ et $\mathbf{x}_2$ par

$$d_Q(\mathbf{x}_1,\, \mathbf{x}_2,\, \boldsymbol{A}) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \boldsymbol{A} (\mathbf{x}_1 - \mathbf{x}_2)} \tag{7.29}$$

Nous avons testé trois méthodes de classification :

1. Le plus proche voisin de l'objet o en considérant tous les objets o' de l'ensemble d'apprentissage $\mathcal{LS}$ et en utilisant la distance euclidienne :

$$\begin{aligned}
\hat{y}(o) &= y\left(\underset{o' \in \mathcal{LS}}{\arg\min}\; d_Q\left(\tilde{\mathbf{a}}(o),\, \tilde{\mathbf{a}}(o'),\, \mathbf{I}\right)\right) && (7.30)\\
&= y\left(\underset{o' \in \mathcal{LS}}{\arg\min}\; d_Q\left(\mathbf{\Phi}^T(\mathbf{a}(o) - \boldsymbol{\mu}),\, \mathbf{\Phi}^T(\mathbf{a}(o') - \boldsymbol{\mu}),\, \mathbf{I}\right)\right) && (7.31)\\
&= y\left(\underset{o' \in \mathcal{LS}}{\arg\min}\; d_Q\left(\mathbf{\Phi}^T\mathbf{a}(o),\, \mathbf{\Phi}^T\mathbf{a}(o'),\, \mathbf{I}\right)\right) && (7.32)\\
&= y\left(\underset{o' \in \mathcal{LS}}{\arg\min}\; d_Q\left(\mathbf{a}(o),\, \mathbf{a}(o'),\, \mathbf{\Phi}\mathbf{\Phi}^T\right)\right) && (7.33)
\end{aligned}$$

2. Le plus proche voisin en considérant les moyennes $\boldsymbol{\mu}_j$ des classes et en utilisant la distance euclidienne :

$$\begin{aligned}
\hat{y}(o) &= \underset{j \in \{1,2,\ldots,M\}}{\arg\min}\; d_Q\left(\tilde{\mathbf{a}}(o),\, \tilde{\boldsymbol{\mu}}_j,\, \mathbf{I}\right) && (7.34)\\
&= \underset{j \in \{1,2,\ldots,M\}}{\arg\min}\; d_Q\left(\mathbf{\Phi}^T(\mathbf{a}(o) - \boldsymbol{\mu}),\, \mathbf{\Phi}^T(\boldsymbol{\mu}_j - \boldsymbol{\mu}),\, \mathbf{I}\right) && (7.35)\\
&= \underset{j \in \{1,2,\ldots,M\}}{\arg\min}\; d_Q\left(\mathbf{\Phi}^T\mathbf{a}(o),\, \mathbf{\Phi}^T\boldsymbol{\mu}_j,\, \mathbf{I}\right) && (7.36)\\
&= \underset{j \in \{1,2,\ldots,M\}}{\arg\min}\; d_Q\left(\mathbf{a}(o),\, \boldsymbol{\mu}_j,\, \mathbf{\Phi}\mathbf{\Phi}^T\right) && (7.37)
\end{aligned}$$

3. La classe la plus probable au sens du maximum de vraisemblance [12] :

$$\begin{aligned}
\hat{y}(o) &= \underset{y \in \{1,2,\ldots,M\}}{\arg\max}\; p(\tilde{\mathbf{a}}(o) \mid y) && (7.38)\\
&= \underset{y \in \{1,2,\ldots,M\}}{\arg\max}\left\{\frac{1}{(2\pi)^{K/2}|\tilde{\mathbf{\Sigma}}_y|^{1/2}} \exp\left(-\frac{1}{2}\, d_Q^2(\tilde{\mathbf{a}}(o),\; \tilde{\boldsymbol{\mu}}_y,\; \tilde{\mathbf{\Sigma}}_y^{-1})\right)\right\} && (7.39)\\
&= \underset{y \in \{1,2,\ldots,M\}}{\arg\min}\left\{d_Q^2(\tilde{\mathbf{a}}(o),\; \tilde{\boldsymbol{\mu}}_y,\; \tilde{\mathbf{\Sigma}}_y^{-1}) + \log|\tilde{\mathbf{\Sigma}}_y|\right\} && (7.40)
\end{aligned}$$

[12] Cette approche est ici équivalente à l'approche « maximum a posteriori » car les probabilités a priori $\{p(y)\}_{y=1}^M$ sont égales (nous avons le même nombre d'exemples par classe).

La transformation $\mathbf{\Phi}^T$ est égale à $\mathbf{\Phi}_{PCA}^T$ lors de la classification dans l'espace PCA et $\mathbf{\Phi}_{LDA}^T \mathbf{\Phi}_{PCA}^T$ pour la classification dans l'espace PCA+LDA.

Les équations 7.33 et 7.37 permettent de se rendre compte que la classification pourrait être effectuée dans l'espace initial en utilisant une métrique appropriée mais ne doivent pas être utilisées telles quelles pour l'implémentation. La métrique étant de dimension $D \times D$, le calcul d'une distance demanderait de l'ordre de D^2 opérations. Si $\mathbf{\Phi}$ possède $K \ll D$ colonnes (la dimension du sous-espace), les métriques des équations 7.30 et 7.34 sont des matrices identités de dimensions $K \times K$, le nombre d'opérations est de l'ordre de K. La projection de $\mathbf{a}(o)$ dans les nouveaux espaces demande de l'ordre de $K \times D$ opérations. La seconde et la troisième méthode requiert le calcul que de M distances alors que la première en demande N, ce qui peut devenir excessif si l'ensemble d'apprentissage est trop grand. Plusieurs techniques existent afin de réduire cette complexité. On peut notamment essayer de ne conserver que les objets proches des surfaces de décision comme cela est illustré sur la figure 7.12. Il y a bien sûr encore d'autres techniques [DH00, section 4.5]. Cependant, la rapidité n'est pas notre premier souci car nous n'avons pas de contrainte temps-réel pour la reconnaissance des signaux.

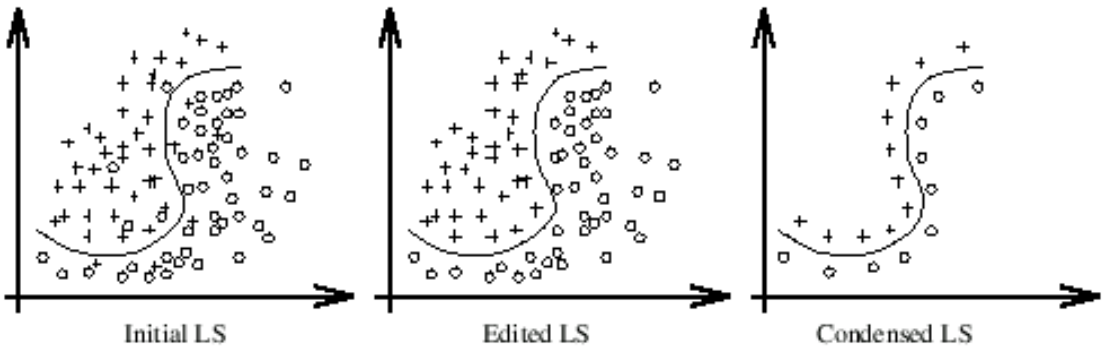
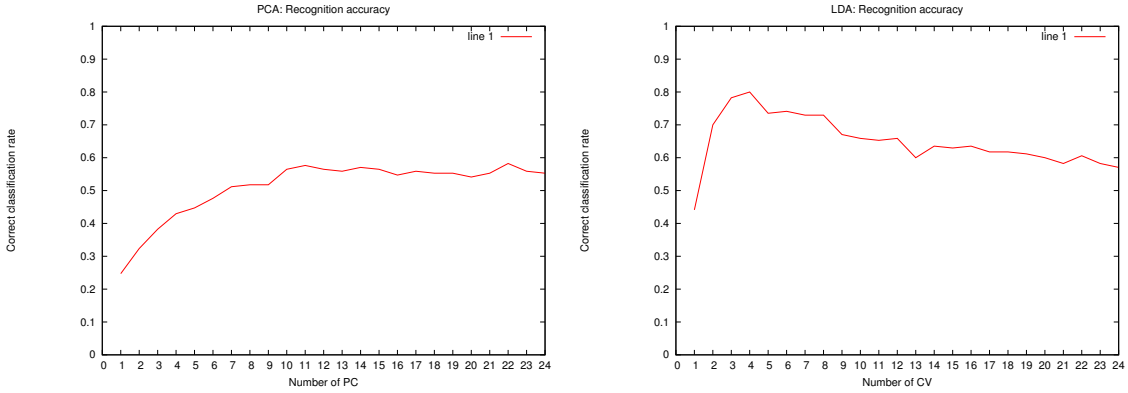


FIG. 7.12 – Plus proche voisin : édition et condensation. [Weh00]

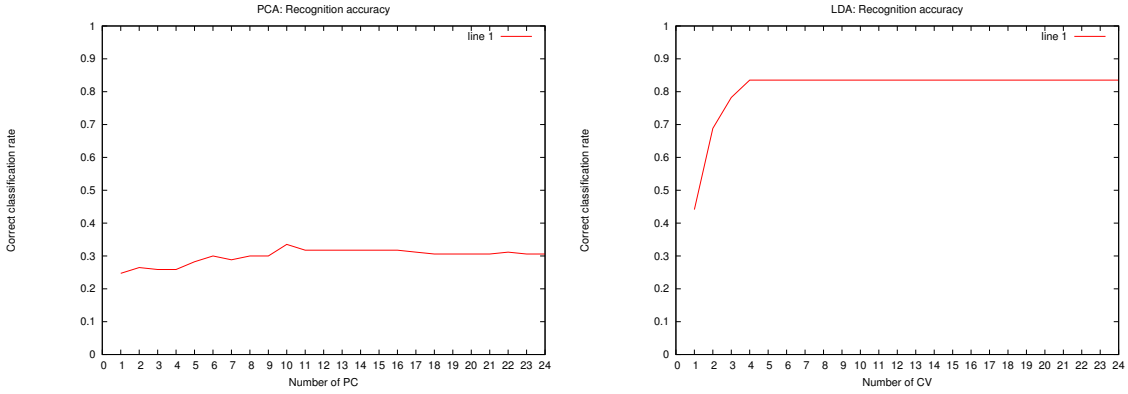
Les performances obtenues sont illustrées sur les figures 7.13, 7.14 et 7.15. Nous pouvons remarquer que :

- La classification dans la base PCA+LDA donne toujours de meilleurs résultats que dans la base PCA si l'on ne considère que les 4 premiers attributs.
- Pour une méthode donnée dans un espace donné, la classification sur les pictogrammes est globalement meilleure que sur les images dépourvues de leur fond et cette dernière est meilleure que sur les images originales.
- Avec les méthodes de la plus proche classe, le taux de classifications correctes est toujours maximal avec $M - 1 = 4$ facteurs discriminants. Cela est intuitivement logique car nous sommes en présence de 5 classes (seules les 4 premières images de FISHER correspondent à des valeurs propres non nulles).

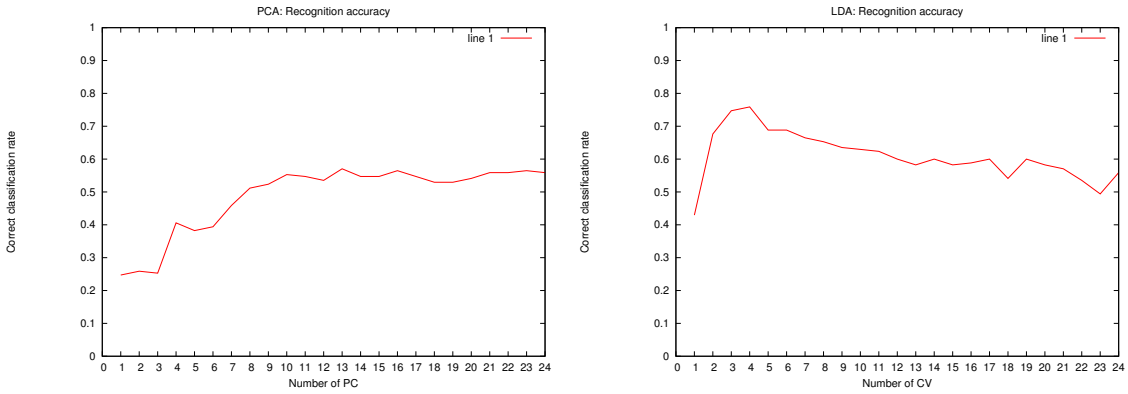
- Le taux de classification dans l'espace PCA+LDA sur les 4 premiers facteurs discriminants en utilisant respectivement la première, la deuxième et la troisième méthode vaut respectivement :
  - 80, 84 et 76 % sur les images originales ,
  - 91, 87 et 84 % sur les images dépourvues de leur fond ,
  - 95, 94 et 92 % sur les pictogrammes.



(a) Classification par la méthode du plus proche voisin en utilisant la distance euclidienne.

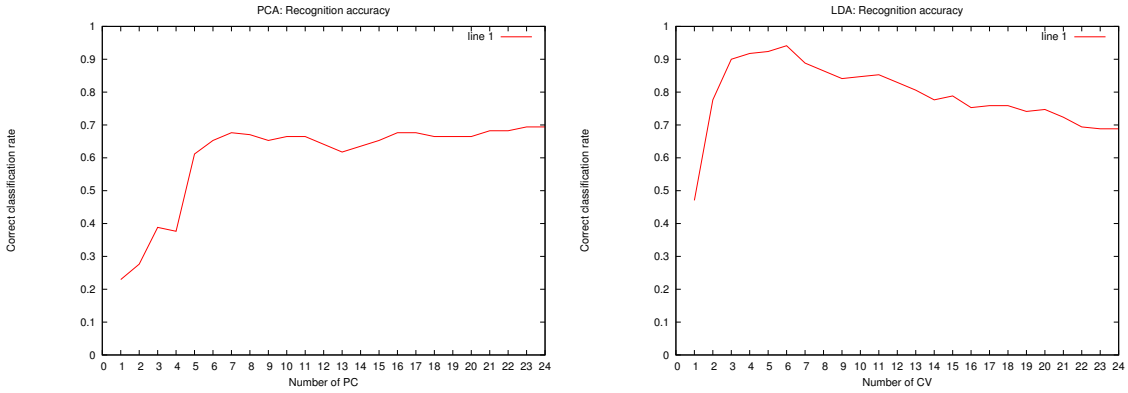


(b) Classification par la méthode de la plus proche classe en utilisant la distance euclidienne.

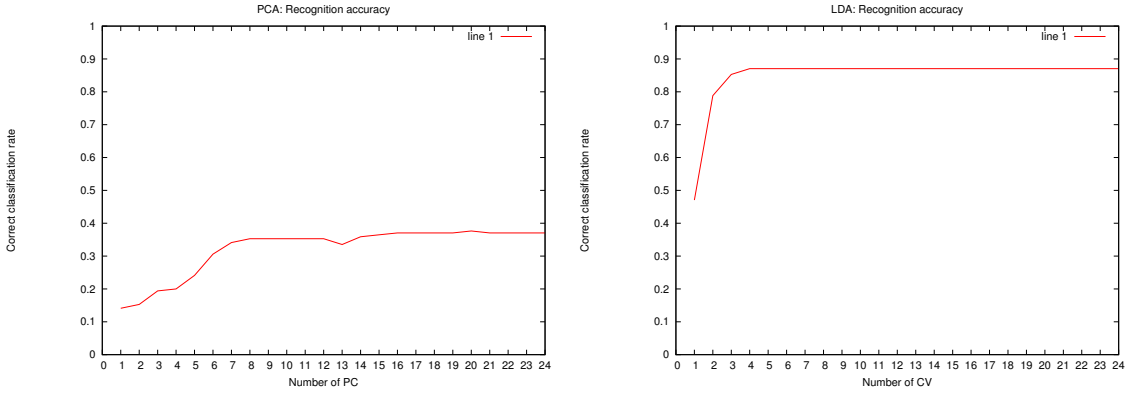


(c) Classification par maximum de vraisemblance.

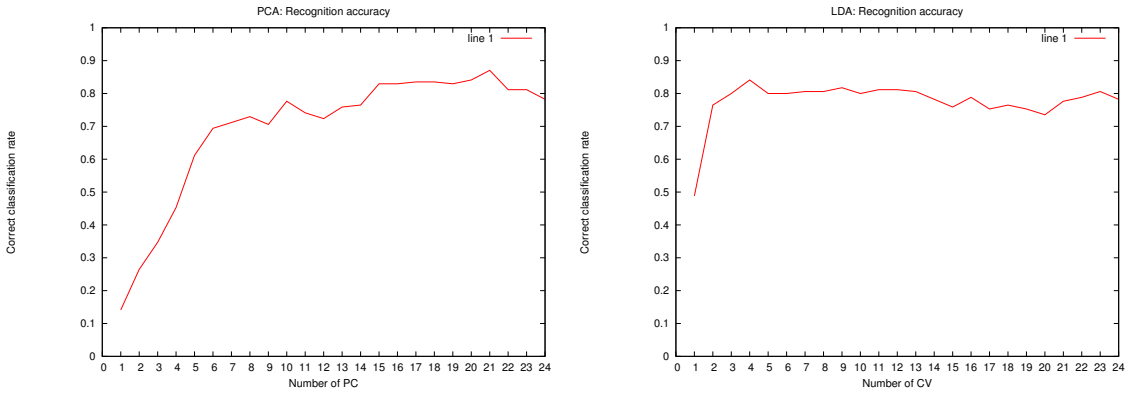
FIG. 7.13 – Taux de reconnaissance sur les images originales. La colonne de gauche est relative à la classification dans l'espace PCA et celle de droite dans l'espace PCA+LDA.



(a) Classification par la méthode du plus proche voisin en utilisant la distance euclidienne.

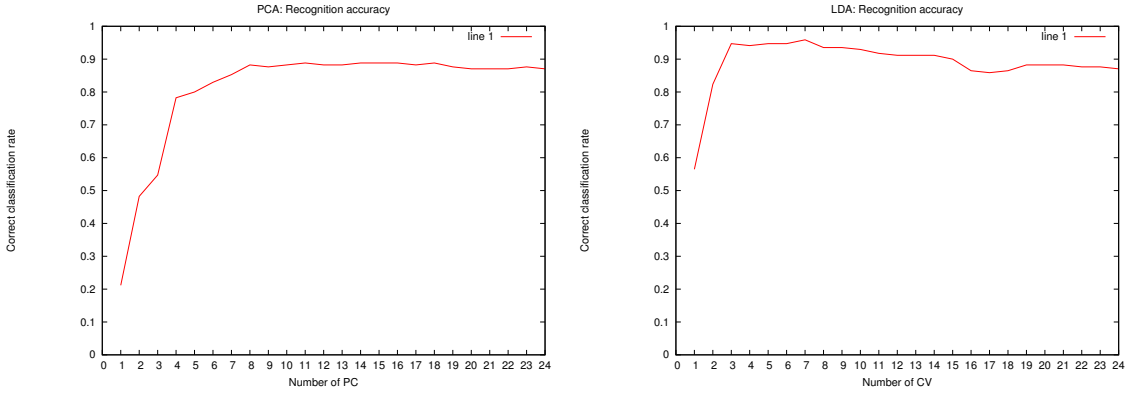


(b) Classification par la méthode de la plus proche classe en utilisant la distance euclidienne.

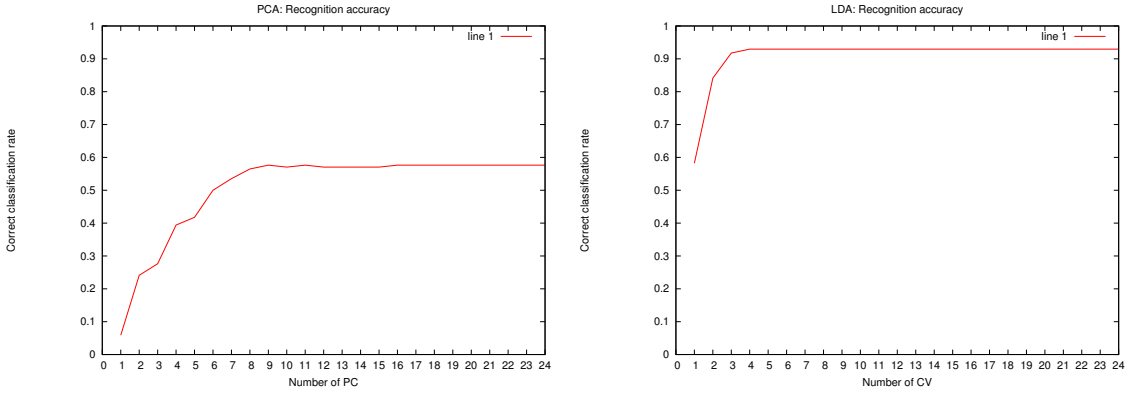


(c) Classification par maximum de vraisemblance.

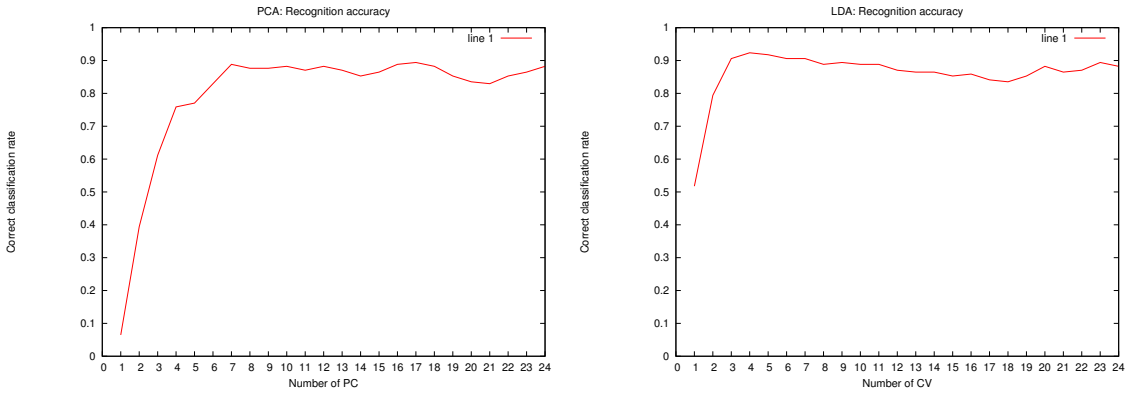
FIG. 7.14 – Taux de reconnaissance sur les images dépourvues de leur fond. La colonne de gauche est relative à la classification dans l'espace PCA et celle de droite dans l'espace PCA+LDA.



(a) Classification par la méthode du plus proche voisin en utilisant la distance euclidienne.



(b) Classification par la méthode de la plus proche classe en utilisant la distance euclidienne.



(c) Classification par maximum de vraisemblance.

FIG. 7.15 – Taux de reconnaissance sur les pictogrammes. La colonne de gauche est relative à la classification dans l'espace PCA et celle de droite dans l'espace PCA+LDA.

7.2 Ensembles d'arbres et extraction aléatoire de fenêtres

Nous évaluons à présent les performances d'une seconde méthode pour la classification de nos signaux. Celle-ci repose sur les travaux de Marée [Mar05] qui utilise une extraction de fenêtres aléatoires et les ensembles d'arbres extrêmement aléatoires (*Extra-Trees*) introduits par Geurts dans [GEW06]. Nous décrivons plus en détails la méthode dans la section suivante. Nous comparons les résultats obtenus avec ceux de la première méthode dans la section 7.2.2.

7.2.1 Entraînement et classification

Le classeur utilisé ici est un ensemble d'*Extra-Trees*. Contrairement à la première méthode de classification utilisée, le classeur travaille directement sur les valeurs des pixels. On extrait un grand nombre (N_w) de fenêtres, à des positions et tailles aléatoires [13]. Le classeur que l'on va construire manipule des fenêtres d'une taille fixe (8×8 pixels dans nos tests). Les fenêtres sont donc remises à cette échelle. On associe à chaque fenêtre la classe de l'image dont elle est issue. Le nombre total de fenêtres extraites est : $N_w \approx 120\,000$. Elles forment l'ensemble d'apprentissage $\mathcal{LS}$. Ainsi, le classeur est construit à partir d'un $\mathcal{LS}$ constitué de N_w objets, décrits chacun par 64 variables (192 variables pour les images RGB) entières et par une classe de sortie. Le classeur est un ensemble d'*Extra-Trees*. Un *Extra-Tree* est un arbre construit avec un choix aléatoire de l'attribut et du seuil à chaque nœud de test. Pour plus de précisions, le lecteur se référera à [GEW06]. Nous travaillons avec un ensembles de $T = 10$ *Extra-Trees*. Chaque arbre est construit à partir de la totalité de l'ensemble d'apprentissage, ce qui n'est pas toujours le cas dans les méthodes d'ensembles, par exemple pour le *Bagging*. Le schéma de l'entraînement du classeur est représenté sur la figure 7.16.

La figure 7.17 décrit le déroulement de la classification d'une image. De la même façon que pour l'entraînement, on extrait de manière aléatoire des fenêtres à partir de l'image à classer. Chaque fenêtre est remise à l'échelle adéquate avant d'être propagée dans chaque arbre. Chaque arbre donne en sortie une classe pour la fenêtre. Il y a donc T votes par fenêtre. On additionne les votes obtenus par chaque fenêtre, classe par classe. L'image est classée suivant la classe majoritaire présente dans l'ensemble des votes.

7.2.2 Résultats

Nous donnons dans cette section les résultats obtenus lors des tests sur nos images de signaux. Pour rappel, nous travaillons avec les classes représentées sur la figure 7.1.

[13] Les signaux sont présents à différentes échelles dans les images de la base de données, d'où l'utilisation d'une taille aléatoire.

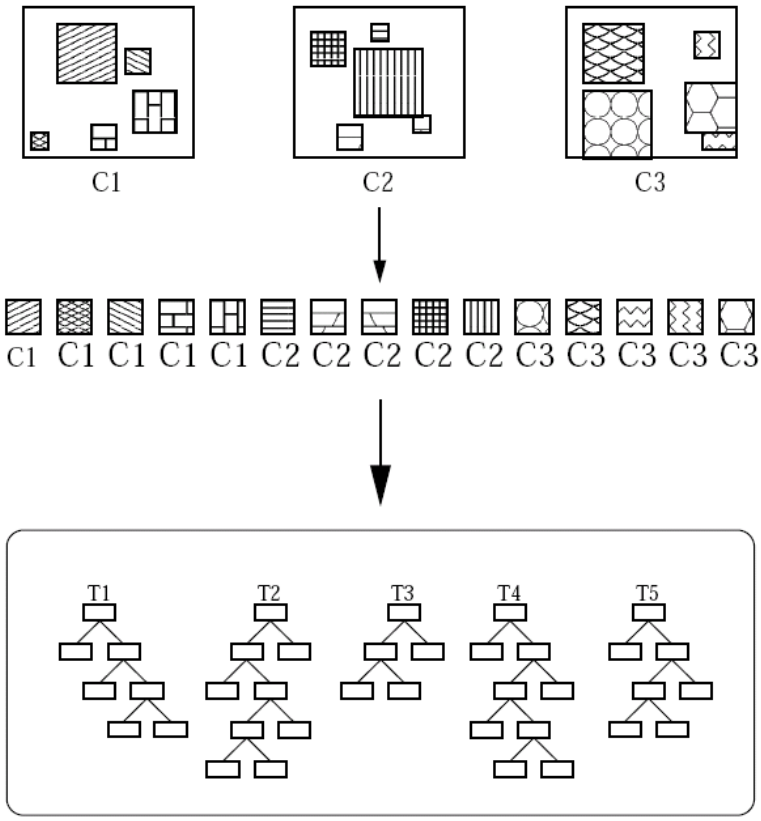


FIG. 7.16 – Entraînement de l'ensemble d'*Extra-Trees*. On extrait des fenêtres, de taille et de position aléatoires, des images de la base de données. On les remet à la même échelle et on construit un ensemble d'arbres de décisions ($T = 5$). [MGPW05b]

Nous entraînons et testons trois types de classeurs selon le type d'images utilisées : images originales, images dépourvues de leur fond et pictogrammes. Nous travaillons sur les images avec leur taille originale [14], contrairement à la première méthode de classification.

Nous travaillons d'abord sur des images en canal Y [15] pour comparer nos résultats à la première méthode. En travaillant sur les mêmes ensembles d'apprentissage et de test que précédemment, nous obtenons les résultats suivants, selon le type d'images utilisées :

- images originales : $44,7$ % de classifications correctes.
- images dépourvues du fond : $72,3$ % de classifications correctes.

[14] Pour cette méthode, nous obtenons des résultats similaires que nous travaillions avec les images de taille originales ou avec des images remises à la taille 37×32 pixels, ceci probablement grâce à l'extraction de fenêtres de taille aléatoire. Aussi, nous n'effectuerons pas de remise des images à une taille unique, par souci de simplicité.

[15] Les images Y sont obtenues à partir des images RGB grâce à la formule $Y = 0,299\,r + 0,587\,g + 0,114\,b$.

– pictogrammes : 91, 8 % de classifications correctes.

Nous remarquons à nouveau le gain obtenu lorsque l'on supprime successivement le fond de l'image et le triangle extérieur. La figure 7.18 représente les résultats sous une manière plus détaillée. On y voit les histogrammes de confusion et de classification. Les histogrammes de confusion permettent de voir pour chaque classe, la confusion qui existe dans le classeur. Autrement dit, on y voit les classes qui sont obtenues (couleur) en fonction des classes désirées (axe horizontal). Sur la figure se trouvent également les histogrammes de classification qui découlent des précédents. On y distingue simplement pour chaque classe, les exemples bien classés en vert des exemples mal classés en rouge. L'observation de ces histogrammes, lors de tests sur de plus grands ensembles d'images, pourraient mettre en exergue des informations intéressantes. Il se pourrait par exemple, que les images d'une classe soient beaucoup moins bien classées que celles de toutes les autres. Un autre cas de figure serait la confusion entre deux classes particulières. L'observation de ces histogrammes de confusion pourrait alors nous permettre de mettre d'envisager des solutions pour améliorer la classification.

Pour mettre en avant les bienfaits de la suppression du fond de l'image et du triangle extérieur, nous présentons des résultats sur les classifications obtenues. Lorsque une image est classée, on a une information sur la confiance que l'on peut donner au classement. En effet, rappelons que le classement s'effectue par un vote majoritaire. Plutôt que de regarder uniquement la classe majoritaire, on peut regarder les scores effectués par chacune des classes. On peut de cette façon savoir la confiance que l'on peut accorder à la classe de sortie. On peut également se rendre compte des classes qui sont confondues.

Sur la figure 7.19, deux signaux sont présents. Les figures 7.19(a) et 7.19(b) correspondent respectivement au signal qui a été classé avec l'indice de confiance maximum et au signal qui a été classé avec l'indice de confiance minimum lors des tests effectués sur les images originales. Les indices de confiance sont respectivement de $84,2$ % et de $22,2$ %. Ils représentent la proportion des votes pour la classe majoritaire. On voit dans le second cas que l'on est proche du classement aléatoire [16]. Pourtant, la qualité des deux images semble similaire. Comme nous le verrons, le faible indice de confiance de l'image de la figure 7.19(b) est dû au fond de l'image. Sur la figure 7.19(a), on a l'impression que le fond de l'image est absent à l'inverse de la figure 7.19(b) où le fond de l'image est très chargé. Sur la figure 7.20, on montre la répartition des votes suivant les classes. La couleur représente la classe et le nombre de votes est porté sur l'axe vertical.

Les figures 7.21(b) et 7.21(d) représentent l'image sans fond et le pictogramme correspondant à l'image de la figure 7.19(b). Les répartitions des votes selon les classes pour ces deux images sont reprises respectivement sur les figures 7.22(b) et 7.22(d). On voit qu'en retirant le fond de l'image et puis le triangle extérieur pour ne gar-

[16] Étant donné que nous travaillons avec 5 classes, un classeur aléatoire accorderait 20 % des votes à chaque classe.

der que l'idéogramme, l'indice de confiance augmente progressivement. Les figures 7.21(a) et 7.21(c) représentent l'image sans fond et le pictogramme correspondant à l'image de la figure 7.19(a). Les répartitions des votes selon les classes pour ces deux images sont reprises respectivement sur les figures 7.22(a) et 7.22(c). Ici, on remarque une diminution de l'indice de confiance lorsque l'on supprime le fond de l'image (par rapport au résultat obtenu en travaillant sur l'image originale). On peut simplement dire que le fond étant déjà « absent » dans l'image originale, la suppression du fond n'apporte aucun avantage. Par contre, on remarque une amélioration de l'indice de confiance lorsque l'on passe de l'image sans fond au pictogramme. Lorsque l'on travaille avec les pictogrammes, les indices de confiances sont respectivement de 55 % et de 57 % (figures 7.22(c) et 7.22(d)) pour les signaux des figures 7.21(c) et 7.21(d). Bien que les indices de confiance ne soient pas proches des 100 %, la classe de sortie est à présent bien dissociée des autres classes et ce pour les deux signaux. En conclusion, la réduction de l'image originale au simple pictogramme améliore nettement la classification, ce qui est logique puisque toute l'information du signal est contenue dans le pictogramme. Nuançons néanmoins nos propos. L'avantage tiré par les pictogrammes sur les images originales est certainement dû en partie au faible $\mathcal{LS}$ dont nous disposons. En principe, avec un $\mathcal{LS}$ vraiment représentatif, cet avantage devrait diminuer. En effet, les fenêtres extraites dans le fond de l'image ne devraient alors plus être discriminantes.

Comparativement à la première méthode de classification, les résultats obtenus ici sont inférieurs, quoique satisfaisants pour les pictogrammes. Ci-dessous, nous présentons d'autres résultats. Certains semblent prometteurs.

Nous essayons maintenant des classifications basées sur les images couleurs. Notre intuition est que toute l'information d'un signal est portée par le pictogramme, qui est en noir sur fond blanc. Néanmoins, nous tentons tout de même de travailler sur les images couleurs. Cela nous permettra de confirmer notre intuition et de nous assurer qu'aucune information n'est perdue lors du passage des images RGB vers les images en canal Y, les images de départ sur lesquelles nous travaillons étant en RGB. Les résultats du tableau 7.1 confirment notre intuition, l'information donnée par la couleur dans nos signaux est nulle. Les classifications réalisées sur base de images en couleurs donnent des taux d'erreurs plus élevés. Dans [Mar05], Marée recommande l'utilisation des canaux HSB [17] lorsque l'on travaille sur des images en couleurs. Ici, les résultats obtenus avec ces canaux sont moins bons qu'avec les canaux RGB. Nous supposons que c'est parce que toute l'information est contenue dans le pictogramme qui est caractérisé par une teinte instable.

Pour une question d'égalité entre les classes, nous avons travaillé jusqu'à présent avec des $\mathcal{LS}$ et $\mathcal{TS}$ équilibrés, contenant 34 images par classe. Nous essayons maintenant des classifications en utilisant toutes les images disponibles dans notre base de données. La composition des $\mathcal{LS}$ et $\mathcal{TS}$ sur lesquels nous allons travailler est donnée sur le tableau 7.2. Nous créons des classeurs en utilisant successivement 3 [18], 4 [19] et

[17] *Hue Saturation Brightness*.
[18] Les 3 classes les plus représentées sont utilisées : A7_b, B15, B17.
[19] Les 4 classes les plus représentées sont utilisées : A31,A7_b, B15, B17.

Canaux couleurs \ Images	Originales	Sans fond	Pictogrammes
Y	44,7	72,3	91,8
RGB	44,8	65,3	88,2
HSB	40,6	33,5	44,7

TAB. 7.1 – Taux de reconnaissance obtenu en fonction des canaux couleurs.

5 classes des $\mathcal{LS}$ et $\mathcal{TS}$ du tableau 7.2. Les résultats sont montrés dans le tableau 7.3. Ils se rapprochent des résultats obtenus par la première méthode et les dépassent même pour la classification basée sur les pictogrammes. On voit que les résultats de cette méthode s'améliorent lorsque l'on augmente la taille de l'ensemble d'apprentissage. Les résultats obtenus sont très satisfaisants : moins de 2 % de taux d'erreur 212 sur images de test.

Classe	$\mathcal{LS}$	$\mathcal{TS}$
A23	54	21
A31	53	16
A7_b	49	20
B15	101	86
B17	167	69

TAB. 7.2 – Composition des $\mathcal{LS}$ et $\mathcal{TS}$ utilisant toutes les images disponibles dans la base de données.

Nombre de classes \ Images	Originales	Sans fond	Pictogrammes
3	77,1	92,6	100
4	74,3	88	100
5	69,8	84	98,1

TAB. 7.3 – Taux de reconnaissance obtenus en fonction du nombre de classes testées. Les $\mathcal{LS}$ et $\mathcal{TS}$ sont ceux du tableau 7.2.

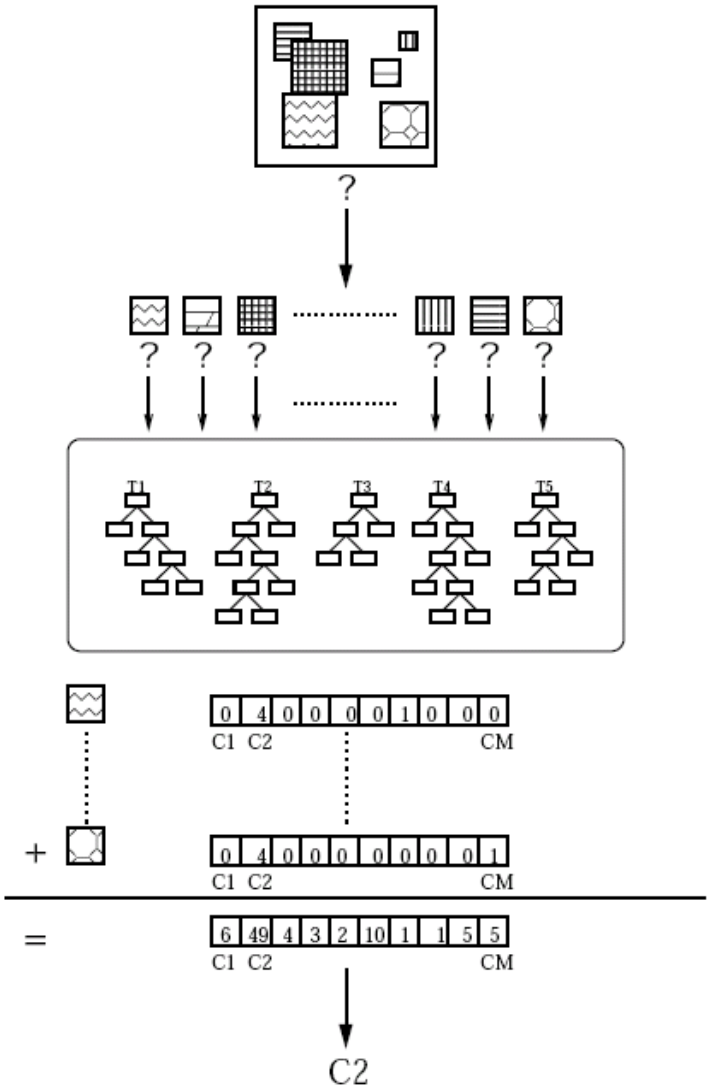
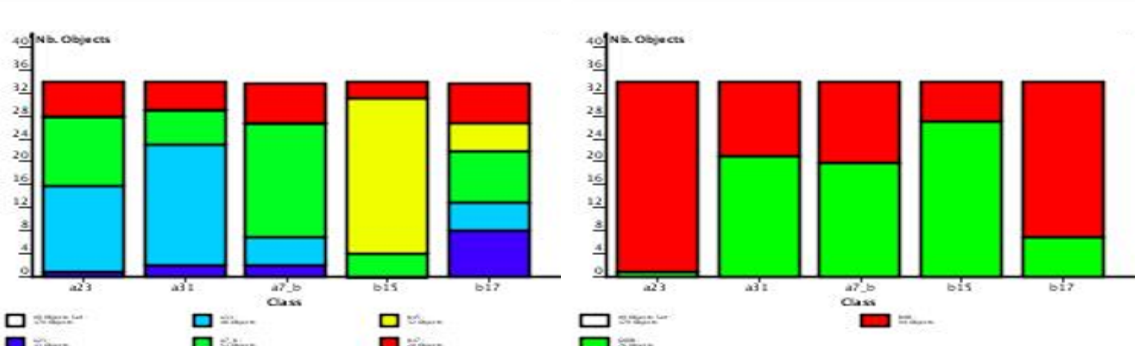
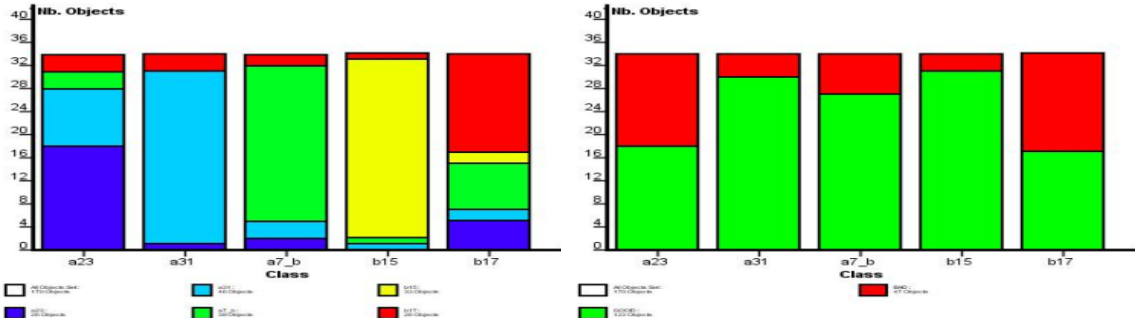


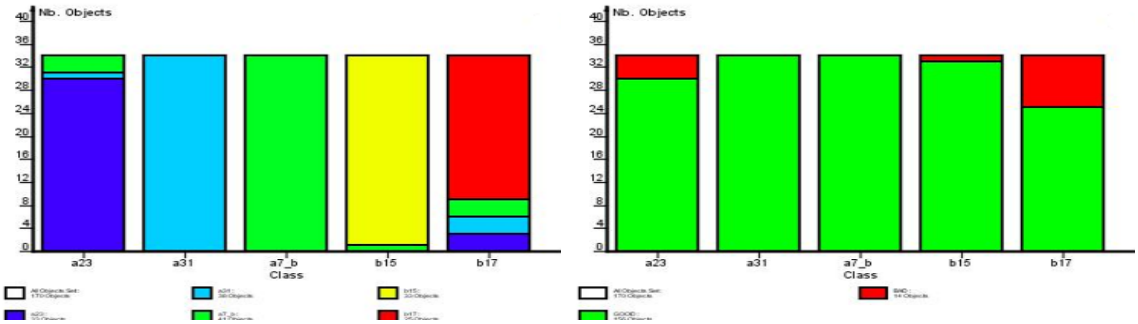
FIG. 7.17 – Classification d'une image par l'ensemble d'*Extra-Trees*. On extrait des fenêtres, de taille et de position aléatoires, de l'image. Les fenêtres sont remises à la même échelle et propagées dans l'ensemble d'arbres ($T = 5$). On additionne les votes et l'image est classée suivant la classe majoritaire. [MGPW05b]



(a) Images originales.



(b) Images dépourvues de fond.

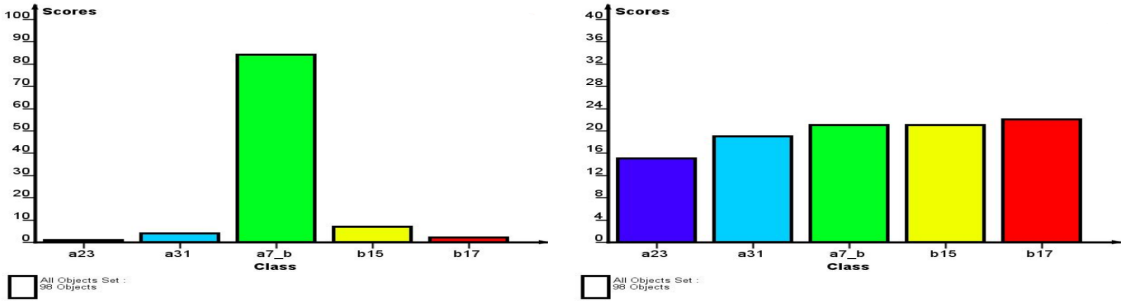


(c) Pictogrammes.

FIG. 7.18 – Résultats sur le $\mathcal{TS}$. Histogrammes de confusion à gauche : chaque couleur représente une classe : A23 en bleu, A31 en bleu clair, A7_b en vert, B15 en jaune et B17 en rouge. Histogrammes de classification à droite : les signaux correctement classés sont en vert et les autres en rouge. Les images sont en canal Yet les $\mathcal{LS}$ et $\mathcal{TS}$ utilisés sont équilibrés.

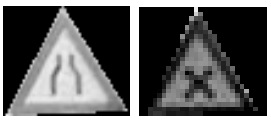
(a) 52x45 (b) 28x24

FIG. 7.19 – Images classées avec le plus grand (a) et le plus petit (b) indice de confiance en travaillant avec les images originales. Les tailles des images réelles sont affichées.

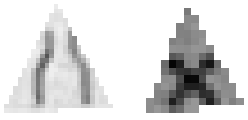


(a) (b)

FIG. 7.20 – Répartition des votes du classeur selon les différentes classes pour les images de la figure 7.19. Les $\mathcal{LS}$ et $\mathcal{TS}$ sont équilibrés. Les couleurs représentent les mêmes classes que sur la partie gauche de la figure 7.18.



(a) 52x45 (b) 28x24



(c) 28x24 (d) 17x14

FIG. 7.21 – Images sans fond (a)(b) et pictogrammes (c)(d) correspondant aux deux images originales de la figure 7.19

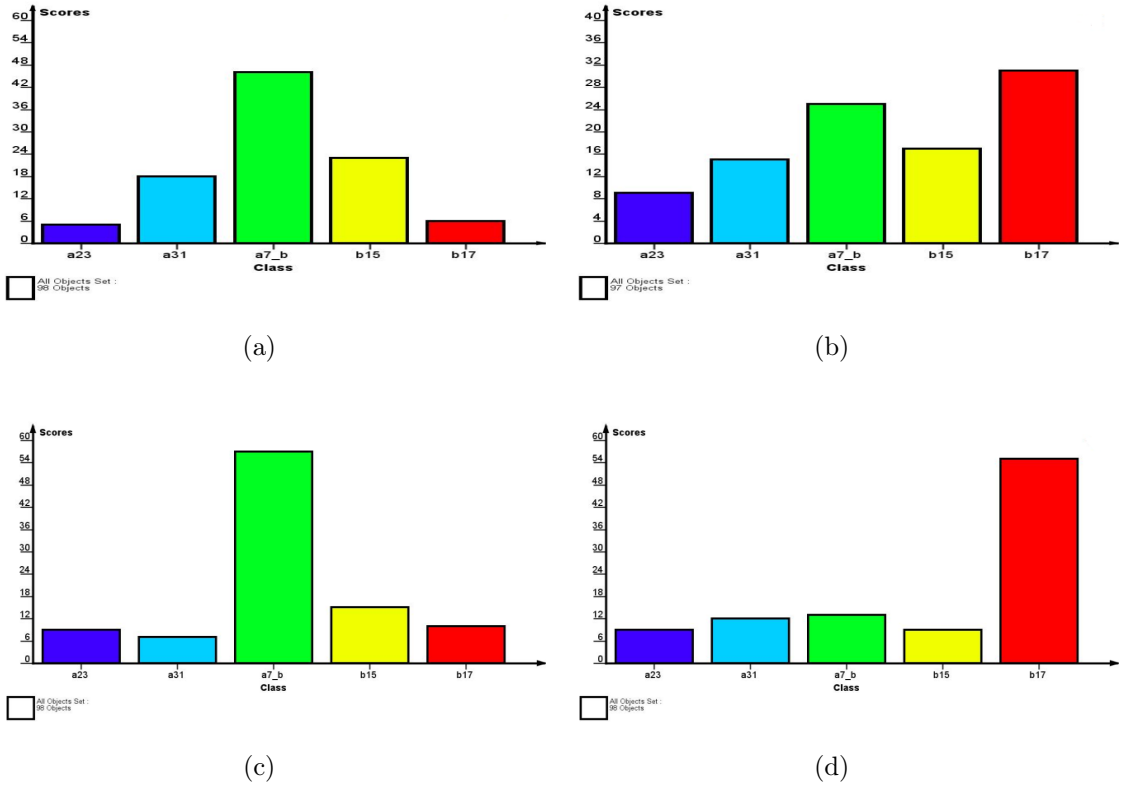


FIG. 7.22 – Répartition des votes du classeur selon les différentes classes pour les images sans fond (a)(b) et pour les pictogrammes (c)(d) des images de la figure 7.21. Les $\mathcal{LS}$ et $\mathcal{TS}$ sont équilibrés. Les couleurs représentent les mêmes classes que sur la partie gauche de la figure 7.18.

Chapitre 8

Améliorations et perspectives

Dans ce chapitre, nous décrivons de multiples améliorations que l'on pourrait apporter à notre travail ainsi que les perspectives offertes par celui-ci.

8.1 Détection et localisation

La méthode de détection et de localisation des signaux routiers que nous avons utilisée, qui combine une cascade de classeurs basés sur des filtres de Haar et une contrainte couleur, fonctionne relativement bien. Les meilleurs performances sont obtenues en milieu rural car la scène contient beaucoup moins de détails et de couleurs rouges qu'en milieu urbain (grands bâtiments, enseignes, publicités, véhicules, etc.).

Les signaux très décolorés comme ceux de la figure 8.1 sont mal détectés par la cascade de classeurs. Et même s'ils le sont, la contrainte couleur a vite fait de les éliminer. Ce défaut peut être un avantage selon l'utilisation que l'on fait du détecteur.

Fig. 8.1 – Signaux décolorés non détectés.

Dans le cas d'un système d'aide à la conduite, l'objectif est de détecter et reconnaître le signal même s'il est détérioré afin d'informer le conducteur de sa présence. Dans notre cas, l'objectif ultime est la création d'un inventaire des signaux présents sur nos routes. Si un signal se trouve dans l'inventaire et que le détecteur ne le détecte plus, cela permettrait de se rendre compte que le signal s'est détérioré et qu'il devrait être remplacé.

Certaines fausses détections pourraient être éliminées en tenant compte de la position du signal. En effet, il est par exemple fort improbable qu'un signal soit

présent dans le bas de l'image. Si nous possédions une base de données reprenant des positions de vrais signaux, nous pourrions par exemple déterminer statistiquement des régions d'intérêts dans lesquelles chercher les signaux. Il nous semble donc utile de sauvegarder la position des détections lors de la construction de l'inventaire. D'un autre côté, nous pourrions aussi imposer des régions d'intérêt. Nous choisirions les régions qui offrent une bonne visibilité, sans nous baser sur des données statistiques.

Un meilleur détecteur pourrait être obtenu en suivant l'approche de Bahlmann et al. [BZR+05] qui utilisent comme paramètre le canal couleur en plus de la position et de l'échelle des prototypes des filtres de Haar. Les prototypes sélectionnés lors de l'entraînement le sont dans les canaux r, g, b, $y = (r + g + b)/3$, $3r/y$, $3g/y$ et $3b/y$ au lieu de l'habituel canal en niveau de gris. Grâce à ce paramètre supplémentaire, le taux de faux négatifs (détections manquées) de leur détecteur passe de $1,6$ % à $1,4$ % et le taux de faux positifs (fausses alarmes) passe de $0,3$ % à $0,03$ %.

Bootstrapping. L'ensemble d'entraînement est évidemment un paramètre crucial qui influence les performances du détecteur. Plus cet ensemble est grand, meilleures sont les performances. Bien qu'il soit assez facile d'obtenir un échantillon représentatif d'exemples positifs (les signaux), il en va tout autrement pour les exemples négatifs. En effet, le nombre d'images que nous pouvons rencontrer ne contenant pas de signaux est beaucoup trop important pour en inclure un échantillon représentatif dans l'ensemble d'entraînement. De plus, tous les exemples négatifs ne sont pas nécessaires pour la construction d'un bon classeur. Supposons que nous ayons un classeur fonctionnel et que nous voulions l'améliorer en le réentraînant avec un nouvel ensemble d'apprentissage. Il est inutile d'agrandir l'ensemble d'apprentissage avec des exemples de tests qui ont été bien classés. Il en va autrement pour les autres exemples. Cette technique est appelée *bootstrapping* et est illustrée sur la figure 8.2, elle a été utilisée notamment par Papageorgiou et al. [POP98] sur un détecteur de piétons à base de filtres de Haar en utilisant l'algorithme donné par Sung [SP95] :

1. Commencer avec un petit nombre d'exemples négatifs dans l'ensemble d'entraînement.
2. Entraîner le classeur avec l'ensemble d'entraînement courant.
3. Tester le classeur sur une suite aléatoire d'images. Collecter tous les exemples négatifs que le système a classé comme positifs et les injecter dans l'ensemble d'apprentissage en tant que nouveaux exemples négatifs.
4. Retourner à l'étape 2.

Le *bootstrapping* est une méthode d'apprentissage actif. L'apprentissage actif est un domaine de recherche qui étudie comment un classeur peut sélectionner intelligemment de futurs exemples d'entraînement afin d'obtenir de meilleurs performances avec moins de données (voir par exemple [CAL94] pour plus d'informations).

Deblurring. Certains signaux peuvent devenir légèrement flous si le véhicule se déplace à grande vitesse, ce qui réduit les chances de le détecter et de déterminer

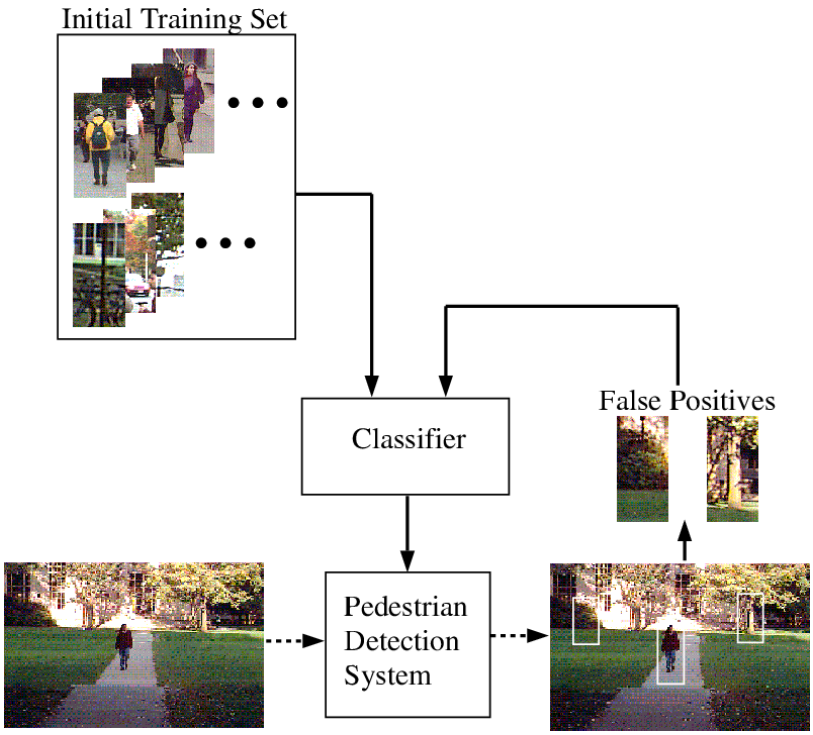


FIG. 8.2 – *Bootstrapping* incrémental pour améliorer les performances d'un classeur [POP98].

son type. Bien que ce problème ne soit pas un des plus importants, nous donnons ici quelques pistes pour le résoudre. Les images floues peuvent être restaurées par des techniques de *deblurring* telle la déconvolution si la fonction d'étalement (PSF [1]) qui a causé le flou est connue. Généralement, comme la PSF n'est pas connue, on essaie de l'estimer à partir de l'image floue, cela est généralement fait via des techniques de déconvolution dites « aveugles ». Il existe aussi des techniques « matérielles » utilisant des capteurs gyroscopiques, des lentilles stabilisées ou des capteurs CMOS spécifiques (voir [BEN04] pour plus d'informations).

Arbre détecteur. Actuellement, la détection ne concerne que certains types de signaux. Par exemple, les signaux d'indication (voir [Arr75]) ne sont pas pris en compte. Nous avons vu que la détection sur l'ensemble des signaux était difficilement envisageable avec une cascade de classeurs unique. L'utilisation en parallèle, de cascades spécialisées sur différents types de signaux, est une solution coûteuse en temps de calcul.

Pour être à même de détecter efficacement l'ensemble des signaux routiers avec une rapidité proche de celle d'une cascade unique, on peut entraîner un arbre détecteur. Cette solution est introduite par LIENHART et al. dans [LLK03]. Lors de la

[1] PSF : Point Spread Function. La PSF est la réponse impulsionnelle du système qui a causé le flou d'où le nom « fonction d'étalement du point ».

construction d'un arbre détecteur, on fusionne les premières étapes de la détection de cascades spécialisées afin de réduire le temps de calcul. Ensuite, on construit des branches spécifiques à chaque type de signal si cela est bénéfique pour l'efficacité du détecteur.

L'arbre détecteur est donc un arbre non dégénéré [2] où chaque nœud est, comme pour une cascade, un classeur entraîné par un algorithme de *boosting*.

8.2 Suivi des signaux

Le suivi des signaux détectés à l'aide d'un filtre de KALMAN permet d'éliminer un grand nombre de fausses détections. En effet, parmi toutes les détections présentes dans une séquence d'images, nous ne conservons que les détections inter-images qui sont assez proches (en terme de distance de MAHALANOBIS) deux à deux et nous considérons après cela qu'une suite de détections correspond effectivement à un signal routier si le suivi persiste pendant un certain temps (pendant un certain nombre d'images). Les fausses détections apparaissant généralement à des endroits aléatoires dans l'image et de manière ponctuelle, beaucoup ne survivent pas à cette contrainte de durée.

Remplacer la contrainte de durée du suivi. Lorsque le véhicule se déplace très lentement ou est à l'arrêt, le nombre de faux suivis augmente si la durée minimale requise (donnée en nombre d'images) pour être considéré comme un signal reste inchangée. Une amélioration consisterait alors à remplacer cette contrainte par une autre, indépendante de la vitesse du véhicule, qui imposerait par exemple que la trajectoire du signal suivi soit plus grande qu'une longueur minimum donnée ou d'une manière plus générale que le vecteur d'état de la première détection de la trajectoire soit assez différent du vecteur d'état de la dernière.

Se servir de caractéristiques propres à chaque signal. La proximité entre deux détections est mesurée à l'aide d'une distance de MAHALANOBIS dans un espace constitué de la position et de la taille des signaux. Mis à part sa taille, nous ne nous servons donc pas de caractéristiques visuelles du signal pour le suivre. En prenant en compte de telles caractéristiques, plusieurs faux suivis pourraient être écartés. Par exemple, nous pourrions calculer les histogrammes des détections successives, les comparer (χ^2, distance de BHATTACHARYYA, etc.), et valider la piste que s'ils ne diffèrent pas trop entre eux.

Meilleur modèle dynamique. Le modèle dynamique que nous avons utilisé est extrêmement simple. La position du signal est modélisée par un mouvement rectiligne uniformément accéléré (modèle d'ordre deux) et sa taille par une constante (modèle

[2] Par rapport à la simple cascade de classeurs, qui est un arbre dégénéré.

d'ordre zéro). Ce modèle s'est révélé amplement suffisant dans la plupart des cas. La situation dans laquelle le modèle se comporte le plus mal est celle où le signal se trouve au bord de l'image et le véhicule se déplace à grande vitesse (plus de 90 km/h). En effet, plus le véhicule s'approche du signal, plus sa vitesse dans le plan image augmente et plus sa taille grandit vite. Cela peut être assez ennuyeux, car c'est dans cette situation que la taille du signal est la plus grande et donc que l'image contient le plus d'information pour la reconnaissance ultérieure. Ce problème pourrait être résolu en utilisant deux modèles dynamiques distincts, un pour les faibles vitesses et un autre pour les vitesse plus élevées. Il ne faut pas non plus perdre de vue que c'est dans cette situation que l'image du signal est la plus floue.

L'utilisation du modèle de Arnoul et al. [AVGM96], avec un filtre de Kalman étendu permettant une localisation 3D du signal, améliorerait sans doute le suivi, mais cette méthode demande de connaître le mouvement de la caméra.

Stabilisation de l'image. La trajectoire réelle d'un signal sur le plan image de la caméra n'est pas parfaitement « lisse ». Les irrégularités de la route provoquent des vibrations et des secousses de la caméra montée sur le véhicule. Ces tremblements ont pour effet d'éloigner la position réelle du signal suivi sur le plan image de la position prédite par le filtre de Kalman. Si cet éloignement est trop important, la détection du signal sur l'image courante n'est pas associée à la précédente et nous perdons ainsi une mesure pour ce signal. Divers systèmes de stabilisation existent. Certaines caméras utilisent des capteurs gyroscopiques pour détecter les secousses qui sont directement compensées par un déplacement de lentilles correctrices (figure 8.3). Il y a typiquement trois étapes majeures dans une stabilisation vidéo :

1. L'estimation du mouvement de la caméra à partir des images initiales.
2. Le filtrage du mouvement qui permet de détecter si c'est un mouvement « parasite » ou non.
3. La compensation qui fournit des images stabilisées.

La phase d'estimation est cruciale. Plusieurs techniques, autant logicielles [SK99] que matérielles, ont été proposées pour améliorer cette étape. Des capteurs mécaniques (gyroscopes, accéléromètres) ont été développés pour une estimation rapide du mouvement. Ils peuvent traiter d'importantes instabilités, mais sont en contrepartie peu précis, encombrants et lourds. D'autres approches ont été développées spécifiquement pour des caméras montées sur un véhicule, notamment par Liang et al. [LTC+04] qui utilisent le marquage des routes tout au long du processus.

Fusion de connaissances. Le principe de fusion de connaissances (présenté dans notre étude bibliographique), proposé par Zhu et al. [ZCR+05] et partiellement utilisé par Bahlmann et al. [BZR+05] dans le cadre de la détection de signaux routiers, est une approche probabiliste élégante permettant d'améliorer considérablement les performances des détecteurs simplement basés sur l'aspect. Notre détecteur s'inspire de ce principe, mais n'exploite pas toutes ses possibilités. Plusieurs améliorations sont à envisager :

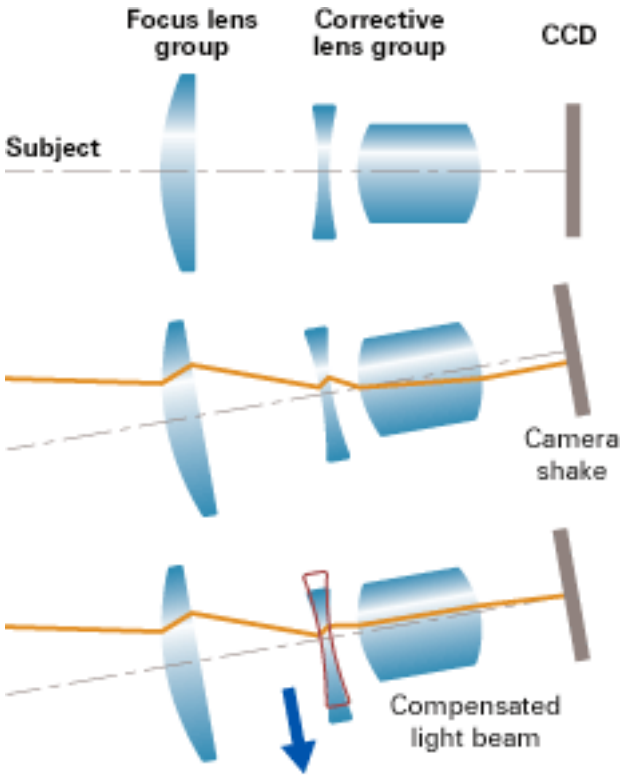


FIG. 8.3 – Système de stabilisation d'image de Canon. Si la lentille frontale descend à cause d'une secousse de la caméra, la lumière émise par l'objet filmé n'est plus parallèle à l'axe optique et l'image se déplace vers le bas. Lorsque ceci arrive, le système optique correctif se déplace et réfracte la lumière de façon à ce que l'image revienne au centre. [Can]

1. Notre détecteur basé sur les filtres de HAAR a été entraîné par *boosting*, nous pouvons donc connaître la probabilité a posteriori qu'une sous-image contienne un signal [FHT98]. Nous pourrions nous servir de cette information pour calculer la probabilité qu'une suite de sous-images corresponde à un signal routier.
2. L'intégration de connaissances a priori sur la géométrie de la scène, en considérant par exemple que la route est plane, permettrait d'éliminer plusieurs fausses alarmes déclenchées par notre détecteur.
3. L'utilisation de connaissances sur les trajectoires des signaux permettrait de rejeter certaines trajectoires absurdes, telle une trajectoire oscillante ou un signal qui rebrousse chemin (lorsque le véhicule se déplace à vitesse constante).

Algorithme d'association de données. Nous avons utilisé la méthode du plus proche voisin pour associer les observations aux signaux suivis. C'est le moyen le plus simple, mais certainement pas le meilleur. L'inconvénient majeur est que la décision d'associer une observation ou non est immédiate et irrévocable. L'utilisation d'une

autre méthode, par exemple un filtre à hypothèses multiples (MHT), améliorerait certainement le taux de détection global.

8.3 Reconnaissance

Les techniques de classification que nous avons testées fonctionnent relativement bien, nous obtenons des taux de classifications correctes dépassant 90 %. Les améliorations possibles sont multiples.

Taille de l'ensemble d'apprentissage. L'ensemble d'apprentissage que nous avons utilisé n'est certainement pas très représentatif. Les 34 images par classe constituant l'ensemble d'apprentissage ne représentent pas toujours des signaux différents. Souvent, un même signal apparaît à plusieurs échelles et sous différents angles de vue. En augmentant la taille du $\mathcal{LS}$, nous avons vu, lors des tests sur la méthode d'ensembles d'arbres, que les résultats s'amélioraient. De nouveaux tests devraient être réalisés avec de plus grands ensembles.

Qualité des images. Les images de notre ensemble d'apprentissage ne sont pas de très bonne qualité, car elles ont été extraites de séquences vidéo désentrelacées et compressées. Des images de meilleure qualité ne peuvent qu'améliorer les résultats.

Normalisation des images. Que ce soit dans la phase d'apprentissage ou dans la phase de test, nous n'avons à aucun moment normalisé les images. La luminosité des images varie fortement, une normalisation de l'histogramme pourrait être envisagée. De plus, les signaux apparaissent sous différents angles de vue. Une normalisation géométrique par transformation homographique permettrait de rectifier les images et ainsi de réduire les variances intra-classes. Enfin, les signaux sont souvent ternes et une binarisation des pictogrammes améliorerait probablement les performances.

Dépendance temporelle. Nous n'avons réalisé nos tests que sur des images isolées. Or, notre programme de détection fournit une suite d'images correspondant au même signal. Un vote majoritaire sur les classifications individuelles, ou mieux, une classification globale obtenue en maximisant une fonction de vraisemblance comme dans [BZR+05] devrait améliorer les performances.

8.4 Usage de la base de données

L'objectif de ce travail était la réalisation d'un outil de détection, de localisation et de reconnaissance de signaux routiers en vue de construire une base de données de signaux géoréférencés à l'aide d'un système GPS. Dans cette section, nous discutons

des applications d'une telle base de données, des informations qu'il serait utile d'inclure, ainsi que de la symbiose qui pourrait s'établir entre cette base de données et les programmes de détection et de reconnaissance.

La base de données de signaux géoréférencés permet d'envisager un éventail d'applications :

- *Maintenance de la signalisation.* Les véhicules équipés du système de détection pourraient s'assurer de l'intégrité et de la bonne visibilité des signaux déjà présents dans la base données .
- *Vérification de la cohérence de la signalisation.* Suite à des travaux de voirie ou de modifications de la signalisation, celle-ci devient parfois incohérente (signaux contradictoires) ou trop abondante. Cela perturbe et met en danger les usagers de la route. Un logiciel utilisant la base de données pourrait débusquer les endroits à risque.
- *Support aux systèmes d'aide à la conduite.* La base de données pourrait épauler les systèmes de reconnaissance de signaux qui équiperont les véhicules grand public du futur. Une base de données complète et à jour serait même en mesure de se substituer à ces systèmes.
- *Perfectionnement de certains logiciels.* L'utilisation de la base de données par les logiciels de calcul d'itinéraires ou par les systèmes de navigation GPS améliorerait leurs performances.
- *Services à divers organismes.* Un accès à la base de données par les forces de l'ordre ou les compagnies d'assurances pourrait être utile en cas d'accident de la route ou de confrontation entre parties.

Suite à ces applications, voici une liste d'informations qu'il serait utile de stocker pour chaque entité dans la base de données :

- La *classe* du signal.
- Une *image du signal* est nécessaire pour un classement manuel ou pour une vérification.
- La *confiance* dans la reconnaissance. Les signaux reconnus avec un faible degré de confiance pourront être examinés par un opérateur humain afin d'être éventuellement reclassés manuellement. Les signaux mal classés pourront être injectés dans l'ensemble d'apprentissage (*Bootstrapping*) afin d'améliorer le système de reconnaissance.
- Les *dernières coordonnées GPS* qui géoréférencent le signal et éventuellement les *premières coordonnées GPS* qui permettront de vérifier que le signal est visible sur une distance suffisamment longue.
- Le *sens* dans lequel le véhicule se déplaçait lors de la détection, car la signalisation n'est pas la même dans les deux sens.

- Les *trajectoires* sur le plan image de la caméra permettront de réaliser des statistiques, voire un apprentissage, afin de construire un modèle qui aidera le programme de détection à valider ou à abandonner certaines pistes lors du suivi.
- Les *dates* auxquelles le signal a été détecté et les dates auxquelles un véhicule est passé devant le signal sans pouvoir le détecter. Si le signal n'est plus détecté à plusieurs reprises, une séquence vidéo pourrait être sauvée et analysée ensuite par un opérateur afin qu'il constate les raisons de cette non-détection (détérioration, mauvaises conditions météorologiques, etc.). La date de la dernière reconnaissance permettrait aussi d'estimer si l'information que l'on possède peut encore être considérée comme valide.

On peut imaginer les avantages d'un retour de la base de données sur le module de reconnaissance. La base de données permettrait, grâce au géoréférencement et aux connaissances a priori dont on dispose, d'orienter la classification. Les signaux présents dans la base de données permettraient de déterminer l'environnement routier dans lequel on se trouve (autoroute, zone urbaine,...). On pourrait ainsi donner une certaine probabilité à priori pour un signal d'appartenir à telle ou telle classe :

- L'accès à une autoroute est marqué par le signal F5. Lorsque l'on circule sur une autoroute, nombre de signaux ont peu de chance d'être détectés, par exemple les signaux indiquant un passage pour piétons.
- Un signal de fin de limitation de vitesse est logiquement précédé du signal de limitation de vitesse correspondant.
- Un signal de limitation de vitesse de 90 km/h ne peut théoriquement pas être détecté dans une zone à l'intérieur de laquelle la vitesse est limitée à 30 km/h.
- ...

D'un autre côté, on peut se servir des connaissances a priori autrement. Au lieu d'orienter la reconnaissance, on peut vérifier, après reconnaissance, si les règles de la signalisation routière sont respectées. Si ce n'est pas le cas, on peut alors demander à un opérateur humain de vérifier la classe du signal et la cohérence de la signalisation.

On pourrait encore imaginer d'autres scénarios. Si, à plusieurs reprise, aucun signal n'est détecté sur le même tronçon de route, une séquence vidéo pourrait être sauvée pour qu'un opérateur humain puisse vérifier qu'il n'y a effectivement pas de signaux sur ce tronçon.

D'autres questions méritent encore réflexion. Serait-il profitable d'améliorer le système de détection pour des signaux déjà détectés auparavant (application d'un traitement spécifique) ? Afin d'améliorer le système de reconnaissance, est-il plus avantageux de réinjecter dans l'ensemble d'apprentissage les signaux qui ont été mal classés, bien classés, classés avec un faible degré de confiance ?

Chapitre 9

Conclusions

Dans ce travail, nous avons mis en place un système capable de détecter, localiser, suivre et reconnaître automatiquement des signaux routiers dans un flux vidéo. Le système est séparé en deux modules indépendants : détection et reconnaissance.

Nous avons d'abord réalisé une étude bibliographique. Nous y avons décrit les méthodes les plus couramment utilisées et celles dont nous nous sommes inspirés pour ce travail.

La détection et la localisation sont accomplies grâce à une cascade de classeurs entraînés sur des filtres de Haar, avec une variante de l'algorithme AdaBoost. Le taux de détection obtenu sur des images isolées est supérieur à 90 % pour environ 1 fausse alarme par image. En imposant une contrainte simple sur la couleur, le nombre de fausses alarmes est réduit d'un facteur 5. L'entraînement de la cascade de classeurs a nécessité la création d'une base de données. Nous avons ainsi constitué une base de données contenant 1988 images de signaux routiers.

Les signaux détectés et localisés sont suivis à l'aide de filtres de Kalman. Un suivi multicible est géré par un algorithme d'association de données qui permet par la même occasion d'éliminer un nombre important de fausses alarmes.

Le programme de détection, implémenté en C sous Linux, est destiné à fonctionner en temps réel. Actuellement non optimisé, il fonctionne à une vitesse de 9 images (640×480) par seconde sur une machine équipée d'un processeur Intel Pentium IV 2.8 GHz. Le système a été testé avec succès sur des séquences vidéo obtenues en environnement urbain et suburbain, dans des conditions météorologiques variables, à des vitesses atteignant 70 km/h.

La reconnaissance des signaux est effectuée entièrement *off-line*. Nous avons testé plusieurs techniques sur des images provenant de 5 classes. La première utilise un modèle génératif dans un espace réduit par une analyse en composantes principales suivie par une analyse discriminante linéaire. La seconde est celle du plus proche voisin dans ce même espace. Nous atteignons des taux de reconnaissance supérieurs à 90 %.

Une troisième méthode utilise directement les valeurs des pixels des images. Il s'agit d'une technique basée sur une extraction aléatoire de fenêtres et sur des ensembles d'*Extra-Trees*. La méthode est implémentée dans le logiciel PiXiT de la société PEPITe. Nous obtenons également des taux de reconnaissance supérieurs à 90 %.

Suite à ce travail exploratoire, de nombreuses perspectives sont envisageables, la plus importante étant sans doute l'apport d'un système GPS qui permettrait de générer une base de données de signaux routiers géoréférencés.

Bibliographie

[Arr75] Arrêté royal belge du 1er décembre 1975. Règlement général sur la police de la circulation routière et de l'usage de la voie publique, 1975.

[AVGM96] P. Arnoul, M. Viala, J.P. Guerin, and M. Mergy. Traffic signs localisation for highways inventory from a video camera on board a moving collection van. In *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, pages 141–146, September 1996.

[BEN04] M. Ben-Ezra and S. K. Nayar. Motion-based motion deblurring. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6) :689–698, June 2004.

[BHK97] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces : Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell*, 19(7) :711–720, 1997.

[BM03] M. Bénallal and J. Meunier. Real-time color segmentation of road signs. In *Canadian Conference on Electrical and Computer Engineering*, volume 3, pages 1823–1826, May 2003.

[Bro98] Eli Brookner. *Tracking and Kalman Filtering Made Easy*. Wiley, 1998.

[BV04] Xavier Baró and Jordi Vitrià. Fast traffic sign detection on greyscale images. In *Catalan Conference on Artificial Intelligence*, 2004.

[BZR+05] C. Bahlmann, Ying Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *IEEE Intelligent Vehicles Symposium*, pages 255–260, June 2005.

[CAL94] David A. Cohn, Les Atlas, and Richard E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2) :201–221, 1994.

[Can] Canon Shift-Method Image Stabilizer System. http://www.canon.com/technology/dv/02.html.

[CG83] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision*, 37 :54–115, 1983.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[Cou03] Christophe Coué. *Modèle bayésien pour l'analyse multimodale d'environnements dynamiques et encombrés : Application à l'assistance à la conduite en milieu urbain*. PhD thesis, Institut National Polytechnique de Grenoble - INPG, December 2003.

[Data] Faculty of Mathematics and Natural Sciences at University of Groningen : Traffic sign image database. http://www.cs.rug.nl/~imaging/databases/traffic_sign_database/traffic_sign.tar.gz.

[Datb] University of Massachusetts Amherst, Computer Science Department, Computer Vision Laboratory at UMASS : Sign Database. http://vis-www.cs.umass.edu/projects/vidi/database.html.

[DH00] R. O. Duda and P. E. Hart. *Pattern Classification*. John Wiley and Sons, 2000.

[dlEAM03] A. de la Escalera, J. M. Armingol, and M. Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image and Vision Computing*, 21(3) :247–258, March 2003.

[dlEM97] A. de la Escalera and L. Moreno. Road traffic sign detection and classification. *IEEE Transactions on Industrial Electronics*, 44(6) :848–859, December 1997.

[Equ] Equbits - accurate, interpretable, automated support vector machine. http://www.equbits.com.

[FCF03] Chiung-Yao Fang, Sei-Wang Chen, and Chiou-Shann Fuh. Road-sign detection and tracking. *IEEE Transactions on Vehicular Technology*, 52(5) :1329–1341, September 2003.

[FFY+04] C. Y. Fang, C. S. Fuh, P. S. Yen, S. Cherng, and S. W. Chen. An automatic road sign recognition system based on a computational model of human recognition processing. *Computer Vision and Image Understanding*, 96(2) :237–268, November 2004.

[FHT98] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting, 1998.

[FP03] David A. Forsyth and Jean Ponce. *Computer Vision : A Modern Approach*. Prentice Hall, 2003.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[Gav98] D. M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *International Conference on Pattern Recognition*, pages Vol I : 439–444, 1998.

[Geu04] Pierre Geurts. Overfitting, bias/variance tradeoff, and ensemble methods. Department of Electrical Engineering and Computer Science, University of Liège, Belgium, 2003-2004. http://www.montefiore.ulg.ac.be/~geurts/Slides/geurts-biasvar-ensemble.ppt.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning Journal*, 2006.

[HP04] Peg Howland and Haesun Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Trans. Pattern Anal. Mach. Intell*, 26(8) :995–1006, 2004.

[IB98] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1) :5–28, 1998.

[Kal60] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society Of Mechanical Engineers : Journal of Basic Engineering*, 82 :35–45, March 1960.

[LKP03] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM-Symposium*, pages 297–304, 2003.

[LLK03] R. Lienhart, L. Liang, and A. Kuranov. A detector tree of boosted classifiers for real-time object detection and tracking. In *IEEE International Conference on Multimedia & Expo*, July 2003.

[LM02] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *2002 International Conference on Image Processing*, volume 1, pages 900–903, September 2002.

[LPV03] Juwei Lu, Kostas N. Plataniotis, and Anastasios N. Venetsanopoulos. Regularized discriminant analysis for the small sample size problem in face recognition. *Pattern Recognition Letters*, 24(16) :3079–3087, 2003.

[LTC+04] Yu-Ming Liang, Hsiao-Rong Tyan, Shyang-Lih Chang, H.-Y.M. Liao, and Sei-Wang Chen. Video stabilization for a camcorder mounted on a moving vehicle. *IEEE Transactions on Vehicular Technology*, 53(6) :1636–1648, November 2004.

[LZ03] Gareth Loy and Alexander Zelinsky. Fast radial symmetry for detecting points of interest. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 25, August 2003.

[Mar05] Raphaël Marée. *Classification automatique d'images par arbres de décision.* PhD thesis, Department of Electrical Engineering and Computer Science, University of Liège, Belgium, February 2005. http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2005/Mar05/.

[MGPW05a] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. Decision trees and random subwindows for object recognition. In *ICML workshop on Machine Learning Techniques for Processing Multimedia Content*, 2005.

[MGPW05b] Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel. Random subwindows for robust image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 34–40, 2005.

[MK01] Aleix M. Martínez and Avinash C. Kak. PCA versus LDA. *IEEE Trans. Pattern Anal. Mach. Intell*, 23(2) :228–233, 2001.

[MKS00] J. Miura, T. Kanda, and Y. Shirai. An active vision system for real-time traffic sign recognition. In *IEEE Intelligent Transportation Systems*, pages 52–57, October 2000.

[Mur00] Fionn Murtagh. Multivariate data analysis with fortran, c and java code. Queen's University Belfast, Astronomical Observatory Strasbourg. Notes de cours, 2000.

[Oct] GNU Octave. http://www.octave.org.

[Ope] OpenCV : Open Source Computer Vision Library. http://www.intel.com/technology/computing/opencv/.

[PiX] PiXiT. http://www.pepite.be/fr/produits/PiXiT/.

[POP98] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision*, pages 555–562, January 1998.

[Pre05] Ph. Preux. Fouille de données. Notes de cours, Université de Lille 3, 228 pages, Juillet 2005.

[RD98] S. Raudys and R. P. W. Duin. Expected classification error of the fisher linear classifier with pseudo inverse covariance matrix. *Pattern Recognition Letters*, 19(5-6) :385–392, April 1998.

[RH01] Christopher Rasmussen and Gregory D. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6) :560–576, 2001.

[RSvB88] K. Rangarajan, M. Shah, and D. van Brackle. Optimal corner detector. In *Second International Conference on Computer Vision*, pages 90–94, December 1988.

[SAANM03] W.G. Shadeed, D.I. Abu-Al-Nadi, and M.J. Mismar. Road traffic sign detection in color images. In *Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems*, volume 2, pages 890–893, December 2003.

[SDS95] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics : A primer, part 1. In *IEEE Computer Graphics and Applications*, volume 15, pages 76–84, May 1995.

[Shi04] M.V. Shirvaikar. Automatic detection and interpretation of road signs. In *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory*, pages 413–416, 2004.

[Shn05] Michael Shneier. Road sign detection and recognition. Submitted to the IEEE Computer Society International Conference on Computer Vision and Pattern Recognition, June 2005.

[Sho02] Hassan Shojania. Real-time traffic sign detection, 2002. Student project for Computer Vision course (ELEC824) in the Department of Electrical and Computer Engineering at the Queen's University.

[SK99] C. Stiller and J. Konrad. Estimating motion in image sequences. *Signal Processing Magazine, IEEE*, 16(4) :70–91, July 1999.

[SP95] Kah Kay Sung and Tomaso Poggio. Example based learning for view-based human face detection. Technical Report CBCL-112, MIT Artificial Intelligence Laboratory, January 24 1995.

[SW96] Daniel L. Swets and Juyang Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell*, 18(8) :831–836, 1996.

[SWS00] Bernhard Schölkopf, Robert C. Williamson, and Alex Smola. SV estimation of a distribution's support, 2000. Neural Information Processing Systems 12. MIT Press.

[TBS04] J. Torresen, J.W. Bakke, and L. Sekanina. Efficient recognition of speed limit signs. In *The 7th International IEEE Conference on Intelligent Transportation Systems*, pages 652–656, October 2004.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[VPH01] L. Vlacic, M. Parent, and F. Harashima. *Intelligent Vehicle Technologies*. Butterworth-Heinemann, 2001.

[VPPS01] Salvatore Vitabile, Giorgio Pollaccia, Giovanni Pilato, and Filippo Sorbello. Road signs recognition using a dynamic pixel aggregation technique in the HSV color space. In *ICIAP*, pages 572–577. IEEE Computer Society, 2001.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR95-041, Department of Computer Science, University of North Carolina - Chapel Hill, 1995.

[Weh00] Louis A. Wehenkel. Applied inductive learning. Notes de cours, Université de Liège, 152 pages, Octobre 2000. PDF version available on line.

[YJPP04] Jieping Ye, Ravi Janardan, Cheong Hee Park, and Haesun Park. An optimization criterion for generalized discriminant analysis on undersampled problems. *IEEE Trans. Pattern Anal. Mach. Intell*, 26(8) :982–994, 2004.

[YLLH03] Hsiu-Ming Yang, Chao-Lin Liu, Kun-Hao Liu, and Shang-Ming Huang. Traffic sign recognition in disturbing environments. In Ning Zhong, Zbigniew W. Ras, Shusaku Tsumoto, and Einoshin Suzuki, editors, *Foundations of Intelligent Systems, 14th International Symposium, ISMIS 2003, Maebashi City, Japan, October 28-31, 2003, Proceedings*, volume 2871 of *Lecture Notes in Computer Science*, pages 252–261. Springer, 2003.

[ZCK98] W. Y. Zhao, R. Chellappa, and A. Krishnaswamy. Discriminant analysis of principal components for face recognition. In *International Conference on Automatic Face and Gesture Recognition*, pages 336–341, 1998.

[ZCR+05] Y. Zhu, D. Comaniciu, V. Ramesh, M. Pellkofer, and T. Koehler. An integrated framework of vision-based vehicle detection with knowledge fusion. In *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, pages 199–204, June 2005.

[ZKS98] Mahmoud M. Zadeh, T. Kasvand, and Ching Y. Suen. Localization and recognition of traffic signs for automated vehicle control systems. In Marten J. de Vries, Pushkin Kachroo, Kaan Ozbay, and Alan C. Chachich, editors, *SPIE Intelligent Transportation Systems*, volume 3207, pages 272–282. SPIE Publishing, January 1998.

Annexe A

Filtre de Kalman

Le filtre de Kalman permet d'estimer récursivement l'état $\mathbf{x} \in \mathcal{R}^n$ d'un processus stochastique régi par une équation stochastique linéaire aux différences

$$\mathbf{x}_k = \boldsymbol{A}_k \mathbf{x}_{k-1} + \boldsymbol{B}_k \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{A.1}$$

avec une mesure $\mathbf{z} \in \mathcal{R}^m$,

$$\mathbf{z}_k = \boldsymbol{H}_k \mathbf{x}_k + \mathbf{v}_k. \tag{A.2}$$

$\boldsymbol{A}_k$ est la matrice de transition d'état, $\mathbf{u}_k$ le vecteur de commande, $\boldsymbol{B}_k$ la matrice de commande et $\boldsymbol{H}_k$ la matrice de mesure. Les processus $\mathbf{w}$ et $\mathbf{v}$ représentent respectivement le bruit du processus à estimer et le bruit de mesure. Ils sont supposés être indépendants l'un de l'autre, blancs, et avec une densité de probabilité normale de moyenne nulle,

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k) \tag{A.3}$$

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_k). \tag{A.4}$$

Le filtrage consiste en deux étapes qui sont décrites ci-dessous de manière très intuitive est représentées sur la figure A.1. Des explications plus rigoureuses sont disponibles par exemple dans [Kal60, WB95, FP03, Bro98].

1. *Prédiction (estimation a priori ou avant la mesure).* La phase de prédiction utilise l'équation aux différences et l'état corrigé précédent $\hat{\mathbf{x}}_{k-1}$ pour prédire l'état courant $\mathbf{x}_k$ du processus. L'incertitude (covariance de l'erreur) sur la prédiction $\boldsymbol{P}_k^-$ est égale à l'incertitude $\boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^T$ provenant de la correction précédente *augmentée* de l'incertitude $\boldsymbol{Q}_k$ engendrée par le bruit du processus.
2. *Correction (estimation a posteriori ou après la mesure).* La phase de correction utilise la mesure $\mathbf{z}_k$ pour corriger l'état prédit $\hat{\mathbf{x}}_k^-$. Le gain de Kalman $\boldsymbol{K}_k$ peut être vu comme un poids dont la valeur reflète la confiance que l'on a dans la mesure $\mathbf{z}_k$ par rapport à la prédiction $\hat{\mathbf{x}}_k^-$. Si ce poids est nul, l'état corrigé $\hat{\mathbf{x}}_k$ est égal à l'état prédit, c'est-à-dire que l'on fait entièrement confiance à la prédiction et non à la mesure. Si le poids est différent de zéro, l'état corrigé

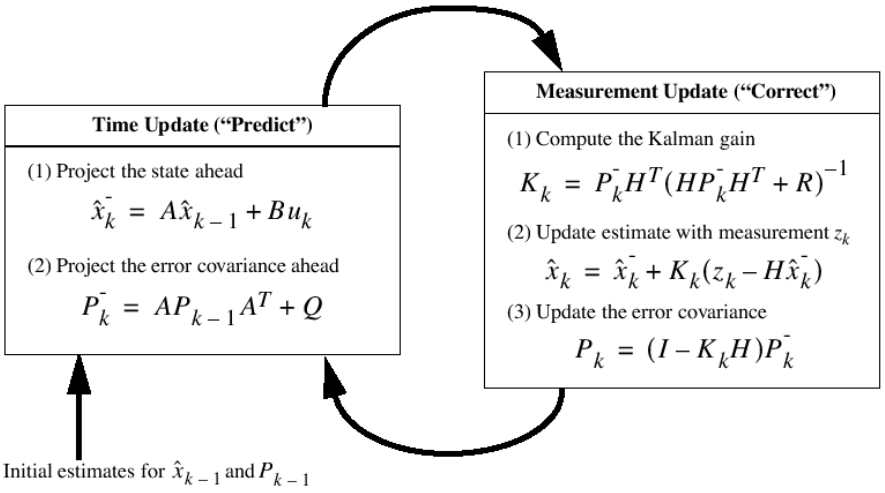


FIG. A.1 – Filtre de KALMAN. [WB95]

est une « moyenne pondérée » de la mesure et de la prédiction. L'incertitude sur la correction $\boldsymbol{P}_k$ est égale à l'incertitude sur la prédiction $\boldsymbol{P}_k^-$ *diminuée* de l'information apportée la mesure.

Annexe B

Filtre de KALMAN étendu

Le filtre de KALMAN étendu permet d'estimer récursivement l'état $\mathbf{x} \in \mathcal{R}^n$ d'un processus stochastique régi par une équation stochastique non linéaire aux différences

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \tag{B.1}$$

avec une mesure $\mathbf{z} \in \mathcal{R}^m$,

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k). \tag{B.2}$$

Comme dans le cas du filtre de KALMAN « standard », les processus $\mathbf{w}$ et $\mathbf{v}$ représentent respectivement le bruit du processus à estimer et le bruit de mesure qui sont à moyenne nulle. Nous pouvons approximer l'état et la mesure par

$$\tilde{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \tag{B.3}$$

et

$$\tilde{\mathbf{z}}_k = \mathbf{h}(\tilde{\mathbf{x}}_k, \mathbf{0}). \tag{B.4}$$

où $\hat{\mathbf{x}}$ est une estimation a posteriori de l'état (état corrigé).

Le problème avec les systèmes non linéaires est que les distributions de probabilité gaussiennes qui traversent ces systèmes ne restent pas gaussiennes. Pour remédier à ce problème, le filtre de KALMAN étendu linéarise l'estimation autour de l'estimation courante en utilisant les dérivées partielles de la fonction du processus f et de la fonction de mesure h. La linéarisation donne

$$\mathbf{x}_k \approx \tilde{\mathbf{x}}_k + \boldsymbol{A}_k(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \boldsymbol{W}_k\mathbf{w}_{k-1} \tag{B.5}$$

et

$$\mathbf{z}_k \approx \tilde{\mathbf{z}}_k + \boldsymbol{H}_k(\mathbf{x}_k - \tilde{\mathbf{x}}_k) + \boldsymbol{V}_k\mathbf{v}_k \tag{B.6}$$

où

– $\boldsymbol{A}_k$ est la matrice jacobienne de $\mathbf{f}$ par rapport à $\mathbf{x}$,

$$\boldsymbol{A}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \ ; \tag{B.7}$$

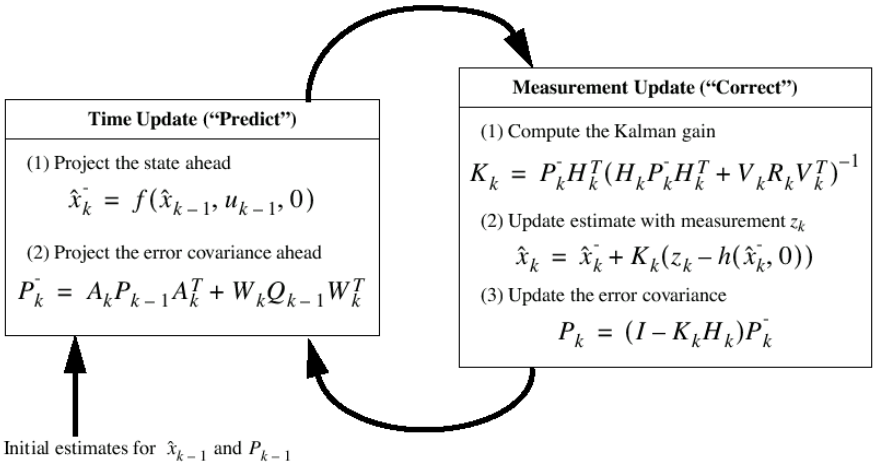


FIG. B.1 – Filtre de KALMAN étendu (EKF). [WB95]

– $\boldsymbol{W}_k$ est la matrice jacobienne de $\mathbf{f}$ par rapport à $\mathbf{w}$,

$$\boldsymbol{W}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{w}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \ ; \tag{B.8}$$

– $\boldsymbol{H}_k$ est la matrice jacobienne de $\mathbf{h}$ par rapport à $\mathbf{x}$,

$$\boldsymbol{H}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}_k, \mathbf{0}) \ ; \tag{B.9}$$

– $\boldsymbol{V}_k$ est la matrice jacobienne de $\mathbf{h}$ par rapport à $\mathbf{v}$,

$$\boldsymbol{V}_k = \frac{\partial \mathbf{h}}{\partial \mathbf{v}}(\tilde{\mathbf{x}}_k, \mathbf{0}). \tag{B.10}$$

Les équations du filtrage sont illustrées sur la figure B.1.

Annexe C

Segmentation des signaux par seuillage sur la couleur rouge

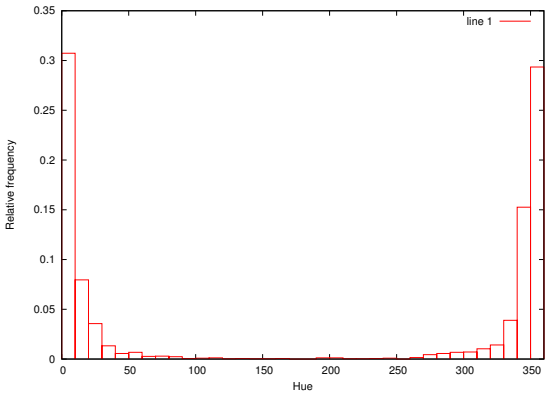
Nos premiers essais de détection de signaux routiers ont consisté en des seuillages sur la couleur rouge de ces signaux. Afin de réaliser un seuillage « intelligent », nous avons recueilli les valeurs de 3394 pixels rouges de signaux provenant d'environ 90 minutes de séquences vidéo. L'histogramme HSV est illustré sur la figure C.1 et le graphe de dispersion dans l'espace teinte-saturation sur la figure C.2.

Une segmentation simple consiste à réaliser deux seuillages successifs. Par exemple, le seuillage

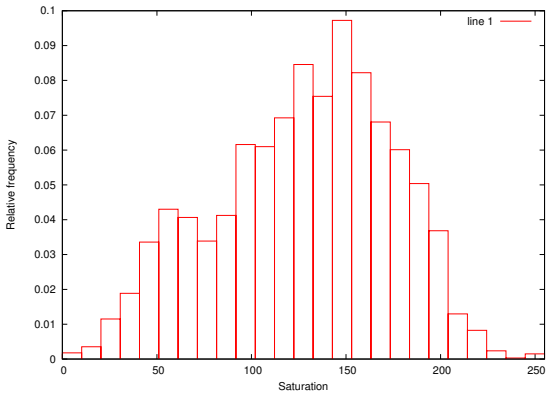
$$I(x,y) = \begin{cases} 255 & \text{si } (h(\text{x,y}) < 40 \text{ OU } h(\text{x,y}) > 320) \text{ ET} \\ & (s(\text{x,y}) < 230 \text{ ET } s(\text{x,y}) > 25) \\ 0 & \text{sinon} \end{cases} \tag{C.1}$$

conserve les pixels compris dans deux boîtes rectangulaires. Un tel seuillage peut donner de très bons résultats comme l'illustre la figure C.3. Malheureusement, les résultats ne sont pas toujours aussi satisfaisants et de nombreux traitements ultérieurs doivent être appliqués pour éliminer les fausses détections. C'est pour cette raison que nous avons abandonné cette technique.

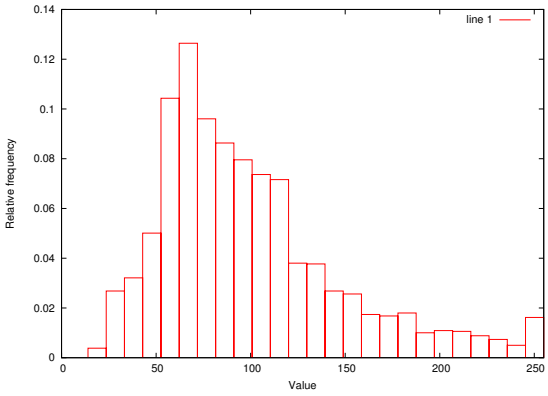
Un seuillage plus performant pourrait être réalisé en estimant le support de la distribution qui, on le suppose, a généré nos données. Plusieurs techniques existent, par exemple la réalisation d'un apprentissage mono-classe avec une machine à vecteurs supports [SWS00].



(a) Teinte



(b) Saturation



(c) Intensité

FIG. C.1 – Histogramme HSV de 3394 pixels rouges.

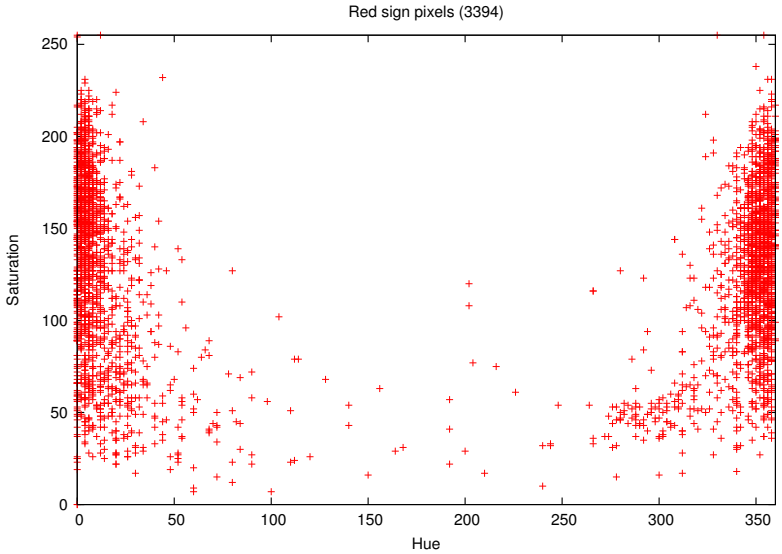
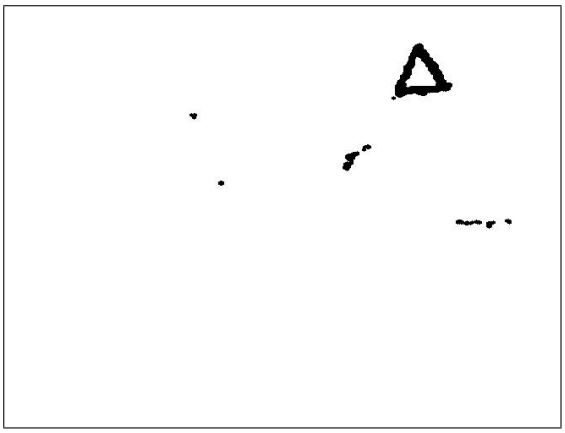
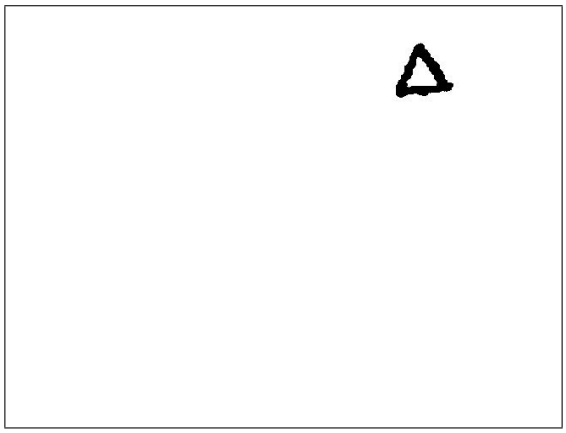


FIG. C.2 – Dispersion de 3394 pixels rouges dans l'espace teinte-saturation.

(a) Image originale.



(b) Seuillage suivi d'une dilatation morphologique.



(c) Contrainte sur la taille des composantes connexes.

FIG. C.3 – Détection de signaux rouges par seuillage simple.

Annexe D

Base de données utilisée pour l'apprentissage et les tests

La base de données de signaux routiers que nous avons construite pour l'apprentissage et les tests est constituée de 1988 images de signaux routiers dispersées en 38 classes. Un résumé est repris dans le tableau D.1. La taille minimum des images est de 16×16 pixels. Les noms des classes correspondent aux noms donnés aux signaux par le *Règlement général sur la police de la circulation routière et de l'usage de la voie publique* [Arr75]. Un même signal peut apparaître à plusieurs reprises, mais dans des conditions différentes (taille, angle de vue, luminosité, etc.). Nous avons également collecté une série d'exemples négatifs : 306 images 640×480 pixels. Les images ont été obtenues à partir de séquences vidéos provenant d'un caméscope DV.

Classe	Nombre d'images	Classe	Nombre d'images
Triangle rouge sur base		**Cercle rouge - fond blanc**	
A14	9	B19	23
A15	4	C11	3
A19	5	C19	7
A1_dl	12	C21_5	10
A1_dr	17	C21_7	35
A21	44	C21_75	98
A23	75	C23	1
A27	4	C25_6	69
A3	20	C3	6
A31	69	C31_r	23
A33	7	C35	37
A7_b	69	C43_30	64
A7_l	7	C43_50	231
A7_r	33	C43_60	1
B15	187	C43_70	94
B17	236	*Sous-total*	702
Sous-total	798	**Cercle rouge - fond bleu**	
Triangle rouge sur pointe		E1	46
B1	239	E3	31
Stop		E5	28
B5	14	E7	66
Sens interdit		*Sous-total*	171
C1	64	*Total*	1988

TAB. D.1 – Résumé des exemples positifs de la base de données utilisée pour l'apprentissage et les tests.

Annexe E

Réduction du taux de fausses alarmes par une contrainte sur la couleur

Notre détecteur consiste en une cascade de classeurs, entraînés sur des filtres Haar, par une variante de l'algorithme AdaBoost. La détection est réalisée sur des images en niveaux de gris. Pour réduire le taux de fausses alarmes TFA, nous imposons une contrainte couleur aux fenêtres détectées [1]. Le test couleur entraîne une réduction du TFA et du taux de détection TD. Le but est de maximiser la chute du TFA et de minimiser la chute du TD. Sont repris ici les résultats obtenus pour différentes cascade de classeurs. On indique la valeur du paramètre *mode* (ensemble de prototype de filtres de Haar utilisé) et le type de recentrage. Lorqu'il y a un recentrage, le type rectangle (carré) indique qu'on travaille avec des images ayant un rapport hauteur sur largeur de $0,87$ $(1,00)$.

Mode	Recentrage	Chute du TD(%)	Chute du TFA(%)
BASIC	Aucun	29	83
BASIC	Rectangle	19	77
ALL	Aucun	44	81
ALL	Rectangle	15	78
ALL	Carré	10	70

Tab. E.1 – Chute du TD et du TFA suite au test couleur, pour différentes cascades de classeurs.

[1] Les images de départ sont en couleurs. À partir de ces images, on génère des images en niveaux de gris pour la détection. Pour la contrainte couleur, on travaille sur les images de départ.