Train de véhicules à l'aide d'un système mono caméra

Pablo Augusto Negri
Stage du DEA Robotique et Systèmes Intelligents
Université Pierre et Marie Curie
Paris, FRANCE

Responsables: Michel Parent, Tony Noel

Juin 2003

Table des matières

Introduction

Le projet Informatique Mathématique et Automatique pour la Route Automatisée (IMARA) a comme objectif l'amélioration du transport routier, l'augmentation de la sécurité et l'aide à la conduite des transports, entre autres. Pour accomplir ces objectifs, plusieurs techniques de commandes et differents capteurs sont mis en place pour aller jusqu'à une automatisation totale. Le CyCab est un des véhicules électriques de la famille *CyberCars*, destiné à circuler de manière autonome dans les centres-villes, les rues piétonnes, les zones industrielles, les parcs de jeux ou les aéroports.

De nos jours, la gestion d'une flotte de CyCabs est un des projets proposé par le groupe IMARA. L'application envisagée est un système de transport publique libre service. Les véhicules seront à disposition des clients dans des gares spéciales pour qu'ils puissent se déplacer d'un lieu à l'autre. Ces véhicules qui seront conduits par les clients, compteront sur des systèmes de détection des obstacles et sur différentes aides à la conduite, pour un voyage plus sûr. Dans le cas où il n'y a pas d'utilisateur, il faudra automatiser certaines opérations. Il est désirable d'effectuer un train de véhicules vides, à l'aide d'un seul opérateur dans la première voiture. Cette fonction est très utile pour le "pick-up" des véhicules qui ont été abandonnés et pour les ramener jusqu'à un endroit convenable.

L'introduction de plusieurs capteurs dans la voiture permet d'avoir une redondance d'information. Cette redondance rend possible l'élection de l'information la plus pertinente pour la prise de décision et par conséquent, l'augmentation de la securité.

La vision devient un outil majeur pour des futures exploitations commerciales encadrées dans ce type des projets. L'information d'une caméra couleur fournit des données qui n'avaient pas été prises en compte jusqu'à present.

Il y a beaucoup d'années que le projet IMARA travaille pour réaliser projets de "Platooning" [1]. La nouvelle implémentation serait de développer un système de vision à bas coût et sans autre cible que les caractéristiques géometriques du CyCab.

L'objectif du stage concerne la recherche et l'évaluation des algorithmes que permettent une détection en temps réel de la face arrière du CyCab à travers un système monocaméra. Ce rapport de stage est une description théorique-pratique des algorithmes qui ont été mis en place pour cet objectif. La détection permettra le calcul, par triangulation, de la distance inter-véhicules. Cette distance pourra asservir un système de commande pour atteindre l'objectif de la réalisation d'un train de véhicules.

Les deux principaux inconvénients dans cette tâche sont le temps réel et la variabilitée des environnements où se place le CyCab. D'un côté, pour que le train des véhicules puisse rouler à

une vitesse acceptable, l'algorithme doit avoir une réponse dans un temps très court. Une réponse de l'ordre de 100 msec est acceptable. De l'autre côté, le train de véhicules peut avancer tant dans une rue desserte que dans une rue où il y a des voitures garées et tant dans un jour ensoleillé que nuageux. Le train peut aussi aller jusqu'un garage ou un laboratoire. Dans tous ces environnements différents, l'image prise par la caméra peut avoir différentes luminances, des ombres, etc. De la même façon, tous les objets autour le CyCab contribuent à ce que la reconnaissance soit plus difficile.

Une fois défini un modèle géométrique du CyCab, qui sera la cible à trouver dans les images. Les invariants projectifs du modèle vont servir à l'analyse de l'image du véhicule. Ces invariants permettent l'identification de l'objet qui peut être à différentes distances. Nous avons défini le modèle avec six points caractéristiques du CyCab, que nous devrons trouver dans l'analyse des contours. Dû aux différences de luminance, dont nous avons parlé précedemment, la recherche des contours peut nous donner des résultats insuffisants. Dans certains cas, ces six points ne sont pas trouvés. Notre algorithme doit être capable de déterminer la position du CyCab sans la totalité des points du modèle.

La manière de limiter la recherche et de confirmer les résultats est l'évaluation des contraintes: la direction du gradient, la couleur et le suivi des segments. La définition des contraintes évite l'explosion du le nombre de calculs de algorithmes. Les solutions qui ne respectent pas ces contraintes sont rejetées.

Le chapitre 1 est une référence historique et actuelle de l'Institut National de Recherche en Informatique et en Automatique (INRIA) et du groupe IMARA qui m'ont accueilli pour la réalisation de mon stage de DEA.

Le chapitre 2 présente une description des méthodes utilisées pour la détection des contours dans les images d'une caméra vidéo. Ces contours vont être détectés à partir de l'utilisation du filtrage de Sobel. Pour simplifier, ces contours seront considérés comme des segments. Nous obtenons à la fin un vecteur V_h qui est la représentation des segments horizontaux des images.

Le chapitre 3 décrit la méthode pour chercher le modèle du CyCab à partir de l'information du vecteur V_h. L'application des invariants projectifs permet de trouver des relations géométriques entre les points du vecteur V_h. Un ensemble de points du vecteur qui verifient des rélations géometrique formeront un arbre d'interprétation. L'algorithme ajoute progressivement des points du vecteur V_h. Ces nouveaux points seront comparés avec tous les noyaux de l'arbre pour chercher des rélations géometriques. La recherche finit quand un ensemble de points, noyau de l'arbre, qui est formé de plus de cinq points et dont la déviation par rapport au modèle (erreur) du CyCab est très faible. Le noyau résultant est $\mathcal{N}$.

Le chapitre 4 décrit le codage YUV. Avec cet espace de couleur nous utilisons une méthode pour l'identification de la couleur des objets dans l'image. La méthode présentée pour faire l'identification [7] est plus rapide qu'une comparaison directe des valeurs de l'espace de couleur de chaque pixel. Nous allons définir trois classes de couleur d'intérêt: le bleu (couleur du châssis du CyCab), le noir (fenêtre du CyCab) et le claire (autre couleur du châssis). Ces régions nous serviront pour la contrainte de couleur. À partir des données du noyau $\mathcal{N}$, nous vérifions ses valeurs en cherchant une quantitée de couleur correspondante. Si le noyau n'accomplit pas avec cette contrainte, il sera rejeté. Dans ce chapitre, nous allons chercher le châssis du CyCab dans les régions bleues de l'image. Cela va nous servir pour trouver la position du véhicule dans l'axe x.

Dans le chapitre 5 nous allons exprimer la recherche des contours verticaux du CyCab. Cette nouvelle donnée nous sert à compléter la caractérisation du véhicule dans l'image.

Le chapitre 6 présente une méthode pour prédire la prochaine position des points caractéristiques du CyCab. Nous allons utiliser cette prédiction comme une nouvelle contrainte. Si un noyau de

l'arbre d'identification a des points qui ne sont pas dans l'aire de la prédiction, ce noyau est rejeté.

Le chapitre 7 montre l'utilisation d'une enveloppe pour réduire la zone de recherche du véhicule dans l'image. À partir des résultats de l'algorithme de recherche, nous trouvons l'enveloppe qui va encadrer le CyCab. Nous allons appliquer cette enveloppe à la prochaine image de la séquence, et nous allons chercher le CyCab dans les limites de l'enveloppe. Les résultats nous permettrons de trouver une nouvelle enveloppe que nous appliquerons à la prochaine image.

Dans le chapitre 8 nous allons trouver par triangulation la distance D_h où se trouve le CyCab dans le repère de la caméra. Cette distance est donnée au système de commande avec une erreur qui permet à l'algorithm prendre la décision de valider cette donnée.

Finalement, nous appliquerons les algorithmes décris dans une programme écrit en MATLAB et nous allons le tester avec une séquence video. Nous mentionnerons les différents problèmes trouvées dans autres séquences. Nous proposerons des améliorations des algorithmes pour les rendre plus robustes.

Remerciements

Merci à tous les membres de l'équipe IMARA qui ont contribué au bon déroulement de ce stage et à le rendre enrichissant sur des nombreux plans.

Merci à Michel Parent pour m'avoir accueilli au sein de son équipe et pour m'avoir fait confiance pour le développement de ce projet.

Merci à Tony Noel qui m'a appris la méthode de la recherche, mais surtout de m'avoir donné indépendance dans mon travail et de m'avoir apporté ses critiques rigoureux et constructives.

Merci à Chantal Chazelas, les ingénieurs et les stagiaires du projet IMARA qui ont collaborer à faire ce stage très agréable.

Merci à Pablo Lotito.

Merci à Priscila, comme toujours.

Chapitre 1

L'INRIA

1.1 L'histoire

1.1.1 La fondation de l'IRIA

C'est à la fin de l'année 1964 qu'à l'initiative de la Direction Générale de la Recherche Scientifique et Technique (DGRST) qu'un groupe de dix personnalités indépendantes, choisies pour leurs compétences scientifiques ou économiques se réunissait sous la présidence du professeur Lelong pour conduire une reflexion sur l'importance des nouvelles techniques de traitement de l'information. Le mot informatique n'avait pas encore cours.

En février 1966, le groupe adressait au Comité consultatif de la recherche scientifique et technique (CCRST) un rapport mettant en lumière les con-cepts de base de cette nouvelle science, attirant ainsi l'attention sur l'importance qu'elle allait jouer dans tous les domaines d'activités humaines.

Présentées courant 1966 par le CCRST, les propositions contenues dans ce rapport étaient adoptées dans leur principe par un comité interministériel sur la recherche, alors présidé par le Premier ministre Georges Pompidou. Peu de temps après, l'ensemble des mesures connues sous le nom de "Plan Calcul" sont mises en œuvre.

L'Institut de Recherche d'Informatique et d'Automatique (IRIA), créé par le décret 67-722 du **25 août 1967**, constituait l'un des organes principaux d'exécution de ce Plan Calcul. L'étendue de ses missions faisait participer l'IRIA à tous les aspects de l'opération. L'institut un rôle majeur aussi bien pour la recherche que pour la formation des hommes et pour la diffusion de la connaissance scientifique et technique.

La période **1967-1972** constitue une première phase dans la vie de l'IRIA, celle de la naissance, de l'époque des pionniers sous la direction du professeur Michel Laudet, avec l'appui du professeur André Lichnerowicz, président du conseil scientifique. C'est à cette époque que l'IRIA s'installe dans le domaine de Voluceau laissé vacant par l'OTAN.

1.1.2 De l'IRIA à l'INRIA

En février 1972, il est décidé par un conseil interministériel que l'IRIA verrait ses actions amplifiées, notamment dans le domaine de la synthèse et orientation de la recherche française en informatique et dans le domaine de l'assistance technique donnée à la pénétration de l'informatique dans toutes les activités d'intéret national :

- Regroupement de la recherche au sein d'un **Laboratoire de recherche d'informatique et**

d'automatique, le **Laboria**

- Développement des interventions ayant la nature d'une assistance technique à des applications exemplaires de l'informatique dans les différents secteurs d'activités d'intérêt national ;

- Rattachement direct de l'institut des crédits d'irrigation de la recherche en devenant ainsi ordonnateur ;

Parmi les missions auxiliaires de l'IRIA, on trouvait l'animation et la conduite de projets pilotes, le premier étant la réalisation d'un réseau permettant l'interconnexion de plusieurs grands centre de calcul (Projet Cyclades).

Dans son action, l'IRIA se souciait d'accroître le potentiel en chercheurs, mais aussi d'orienter l'implantation des équipes. Il mettait ainsi à la disposition d'équipes de province, des postes budgétaires de chercheurs, notamment à Rennes. Ces équipes portaient le nom d'équipes associées de l'IRIA.

En **1975** était créé "l'Irisa", laboratoire de recherche associé à l'Université de Rennes et au CNRS.

Fin **1979**, l'institut se recentre sur sa mission de recherche et devient par décret du 27 décembre 1979, **l'INRIA (Institut National de Recherche en Informatique et en Automatique**, établissement public à caractère administratif sous la tutelle du ministère de l'industrie. **Jacques-Louis Lions** en devient l président directeur général.

1.1.3 Après 1980 : l'essor de l'INRIA

En **1980**, naissance de l'unité de recherche de Rennes en tant que composante de l'Irisa.
En **1982** est créée l'unité de recherche de Sophia Antipolis.
En **1983**, Alain Bensoussan devient PDG de l'INRIA.
En **1984**, l'unité de recherche de Lorraine est créée conjointement avec l'université de Nancy et le cntre de recherche en informatique de Nancy (Crin) du CNRS. C'est la même année que la société **Simulog** qui consacre son activité dans le domaine de l'ingénierie assistée par ordinateur voit le jour. Il s'agit de la première société de technologie issue de l'INRIA.
En **1985**, l'INRIA devient établissement public à caractère scientifique et technologique (EPST) placé sous la double tutelle du ministère chargé de la recherche et du ministère chargé de l'industrie.
En **1987**, la deuxième filiale de l'INRIA, **Ilog**, chargée de l'industrialisation de produits INRIA en intelligence logicielle, voit le jour.
Création en **1990** de la troisième filiale industrielle de l'INRIA, O2 Technology, dans le domaine des systèmes de gestion de bases de données orientées objet.
En **1992**, l'unité de recherche Rhône-Alpes voit le jour.
En **1996**, Bernard Larrouturou devient président directeur général de l'INRIA.

Le **18 juillet 2000** voit la signature du Contrat Quadriennal de l'INRIA. Il affiche des objectifs très ambitieux concernant le rayonnement de la recherche scientifique de l'institut, l'impact de ses activités de transfert technologique, son ouverture, ses partenariats et sa politique d'accueil. Il marque aussi l'engagement important de l'Etat pour accroître les moyens de l'INRIA dont les effectifs seront fortement augmentés d'ici à 2003 (de 755 à 1180) ainsi que ses moyens de fonctionnement (+10M d'euros en 2001).

1.2 L'INRIA aujourd'hui

Les missions de l'INRIA sont nombreuses et vont de la recherche fondamentale au transfert de technologie vers l'industrie.

Son organisation est décentralisée avec 5 unités de recherche (Rocquencourt, Rennes, Sophia-Antipolis, Lorraine, Rhône-Alpes) fonctionnant en petites équipes autonomes. A titre d'exemple, l'unité de Rocquencourt dans laquelle s'est deroulé le stage comporte 19 équipes de recherche dont 7 en partenariat avec d'autres organismes.

L'INRIA comporte ainsi 87 projets de recherche dont 47 sont communs avec les universités, les grandes écoles et les organismes de recherche. Un de ses rôles consiste aujourd'hui à valoriser les résultats de la recherche et le transfert technologique : 600 contrats Recherche & Développement avec l'industrie et un peu moins d'une cinquantaine de sociétés sont issues de l'INRIA.

L'INRIA en chiffres (données datant de janvier 2003, source : www.inria.fr):
Ressources budgétaires :

- Budget total : 120M d'euros HT
- ressources propres : 1/4

Ressources humaines :

- Titulaires INRIA : 900 (400 chercheurs, 500 ingénieurs et techniciens),
- Post-Doctorants, stagiaires, invités: 750,
- Doctorants : 700
- Chercheurs et enseignants d'autres organismes : 450
- "ingénieurs experts" (sur contrat de recherche): 200.

Indicateurs :

- Contrats de recettes actifs : + de 600
- Contrats de recettes signés en 2000 : + de 300
- Une soixantaine de sociétés issues de l'INRIA, despuis Ilog, aujourd'hui au Nasdaq, jusqu'aux toutes dernières, 6 en 1999, 12 en 2000, 4 en 2001 et 3 en 2002.
- L'INRIA disposait en 2002 de 47 brevets prioritaires et maintenait au total 174 brevets (prioritaires + extensions). 60 licences payantes de logiciels et 60 licences de logiciels étaient actives en janvier 2002. Plus de 143 logiciels sont disponibles en accès gratuit sur le site de l'INRIA ou à travers la diffusion d'un cédérom.

1.3 Le projet IMARA

C'est un projet "horizontal" à l'INRIA. Il est destiné à coordonner et à transférer les efforts de recherche de l'INRIA qui peuvent être appliqués au domaine de la "Route Automatisée".

Plus particulièrement, ce projet a pour but de développer et d'utiliser les résultats d'un certain nombre de projets de l'INRIA dans les domaines suivants :

- Le traitement du signal (filtrage, calculs, traitement de l'image...)

- Le contrôle-commande du véhicule (accelération, freinage, direction)
- Les outils de programmation temps réel distribués
- Les communications
- La modélisation
- Le contrôle et l'optimisation des systèmes de transport.

Les objectifs de ces recherches est l'amélioration du transport routier en terme de sécurité, d'efficacité, de confort et de minimisation des nuisances.

- Augmenter la sécurité du transport
- Minimiser la consommation d'énergie
- Minimiser la pollution et les nuisances
- Offrir un environnement plaisant
- Offrir de nouveaux moyens de transports disponibles pour tout le monde

L'approche technique est centrée sur les aides à la conduite, pouvant aller jusqu'à une automatisation totale.

Le projet IMARA fait partie du consortium français "La Route Automatisée" en coopération avec l'INRETS, le LCPC, l'ENSMP et l'ENST. Un laboratoie d'expérimentation avec des pistes privées a été mis en place à Versailles-Satory dans le cadre de ce consortium (Laboratoire sur les Interactions Véhicule-Infrastructure-Conducteur).

Le projet est impliqué dans le programme européen "*Carsense*" pour le développement des capteurs embarqués dans les véhicules routiers pour la perception de l'environnement. L'INRIA est en particulier chargé du traitement de l'image et de la fusion des capteurs (image, radar, ladar).

Le projet met à la disposition des diverses équipes participantes des moyens importants avec une flotte d'une dizaine de véhicules instrumentés, divers capteurs et des moyens de calculs et de simulation. Une partie du laboratoire de recherche travaille au développement du projet *Cybercars*, un système expérimental basé sur des véhicules urbains totalement automatisés. Il est en cours d'installation sur le site de Rocquencourt.

Cette action fait l'objet d'une collaboration avec différents partenaires industriels comme Yamaha Europe, Frog Navigation Systems et Robosoft. C'est d'ailleurs sur le site de Rocquencourt que les véhicules automatiques sont mis en place et testés.

Chapitre 2

Détection des segments

Nous allons séparer les résultats des filtrages dérivatifs pour obtenir, d'une part les contours horizontaux et d'autre part les contours verticaux. Nous approximons ces contours comme des segments verticaux et horizontaux (voire figures 2.4 et 2.5).

2.1 Conversion de l'image aux niveaux des gris

Pour la détection des segments nous allons travailler sur les niveaux de gris. Il faut convertir la norme couleur de l'image à une norme des niveaux des gris. Soit $I_{rvb}(x, y)$ l'image couleur, donnée par la caméra. La convertion aux niveaux des gris est fait à partir de ces composants de couleur Rouge, Vert et Bleue. Soient $R(x, y)$,$V(x, y)$ et $B(x, y)$ les plans couleurs de l'image $I_{rvb}(x, y)$, la transformation est:

$$I(x, y) = \frac{R(x, y) + V(x, y) + B(x, y)}{3}$$

2.2 Amélioration de l'image

Avec l'objectif d'extraire plus facilement des informations de l'image, une amélioration est mise en œuvre. Nous n'allons pas nous concentrer sur la suppression du bruit, nous sommes intéressé par le renforcement du contraste.

Le problème à résoudre est de réaliser une transition franche entre deux zones de niveaux de gris différents [2], sans modifier les zones homogènes des images.

La fonction Sigmoide s'utilise pour l'ajustement du contrast,

$$f(x) = \frac{1}{1 + e^{-x}}$$

Avant l'adjustement de contraste a une étape d'égalisation d'histogramme. L'égalisation d'histogramme consiste à répartir les fréquences d'apparition des pixels sur la largeur de l'histogramme. De cette façon, si les valeurs de l'histogramme sont très proches les unes des autres, l'égalisation va permettre de fournir une meilleure répartition, afin de rendre les pixels clairs encore plus clairs et les pixels foncés proches du noir.

2.3 Détection des contour

L'objectif de l'opération de détection de contours est la localisation, dans l'image, des changements d'intensité suffisamment importants pour les considérer comme contours [3]. La recherche de

cette transition peut se faire en recherchant les maxima locaux du gradient [2]. Dans le cas 2D des images, le vecteur gradient dans le point *(x,y)* de l'image est défini:

$$\overrightarrow{G(x,y)} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} I_x(x,y) \\ I_y(x,y) \end{bmatrix}$$

Le calcul des dérivées peut être réalisé en faisant la convolution de l'image avec deux masques, une pour la direction x et l'autre pour la direction y. Chaque masque a une sensibilité spéciale pour différentes directions de contour et pour le bruit. On a donc M_x qui est plus sensible aux contours verticaux et M_y qui, au contraire, est plus sensible aux contours horizontaux.

Nous pouvons calculer le module du gradient:

$$\mathbf{G}(x,y) = |M_x * I(x,y)| + |M_y * I(x,y)|$$

et son orientation:

$$\phi(x,y) = tan^{-1}(\frac{M_x * I(x,y)}{M_y * I(x,y)})$$

Les différentes types de masques sont:

- *Calcul direct*

  Il est la forme la plus simple, mais les resultats ont montrént l'inconvénient qu'il est très sensible au bruit.

  $$I_x(y,x) = I(y,x+1) - I(y,x) = (I * M_x)(y,x)$$
  $$I_y(y,x) = I(y+1,x) - I(y,x) = (I * M_y)(y,x)$$

  Celà correspond à une convolution avec les masques:

  $$M_x = [-1 \quad 1] \text{ et } M_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

- *Masque de Prewitt*

  Ce filtrage combine une dérivée et un filtrage. Les masques de Prewitt sont:

  $$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- *Masque de Sobel*

  Ce filtrage combine une dérivée et un filtrage préalable différent que la Masque de Prewitt et il est largement utilisé dans traitemens d'image grâce à son faible erreur. Les masques de Sobel sont:

  $$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

– *Masque dérivée de gaussienne* Le filtrage précédent la dérivation peut être réalisé grâce à un masque gaussien M. Dans ce cas, le calcul de la dérivée en x se fait par:

$$I_x = \frac{\partial(I * M)}{\partial x} = I * \frac{\partial M}{\partial x}$$

Étant donnée la masque gaussien:

$$M(x,y) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2+y^2}{2\sigma^2})}$$

Les pentes des transitions entre deux zones de niveaux des gris différents, vont être détectées avec deux masques dérivatifs de Sobel. Une masque sensible à la direction horizontal des contours et l'autre sensible à la direction vertical.

Pour les valeurs des masques de Sobel, nous allons nous servir des résultats obtenus d'un design optimal des opérateurs qui minimisent l'erreur dans la direction du gradient[4].

Nom	Coefficients du Masque	Error (10^{-4}rad	Nom.Ops.
Sobel	$[1\ 0\ -1] * [1\ 2\ 1]^T/8$	190	2M, 3A.
Sobel opt.	$[1\ 0\ -1] * [3\ 10\ 3]^T/32$	26	2M, 3A.
opt. 5x5	$[3\ 10\ 0\ -10\ -3]/32 * [7\ 63\ 116\ 63\ 7]^T/256$	1.9	4M, 5A.

Bien que la quantité de calculs soit double, nous allons choisir le masque 5x5 optimal à la place des masques de taille 3x3, étant donné ces deux raisons principales:

– la masque 5x5 est moins sensible au bruit que celle de taille 3x3,

– la faible erreur dans la direction du gradient.

2.3.1 Convolution en utilisant la décomposition 1D

Nous allons utiliser une forme de convolution plus spécialisée pour réduire le nombre d'opérations. Une masque 2D séparable est décomposée en deux filtres linéaires.

Soit un masque *M(i,j)*:

$$M(i,j) = M_x(i,0) * M_y(0,j)$$

En utilisant la loi d'association de la convolution, nous séparons la convolution d'une image avec une masque à 2D en deux convolutions avec filtres linéaires.

$$I * M = I * (M_x * M_y) = (I * M_x) * M_y$$

2.3.2 Valeurs des coefficients de la masque

Le masque que nous allons utiliser a les coefficients suivants:

Pour la direction x $M_x = \begin{bmatrix} 21 & 70 & 0 & -70 & -21 \\ 189 & 630 & 0 & -630 & -189 \\ 348 & 1160 & 0 & -1160 & -348 \\ 189 & 630 & 0 & -630 & -189 \\ 21 & 70 & 0 & -70 & -21 \end{bmatrix} = [3\ 10\ 0\ -10\ -3] * [7\ 63\ 116\ 63\ 7]^T$

Pour la direction y

$$M_x = \begin{bmatrix} 21 & 189 & 348 & 189 & 21 \\ 70 & 630 & 1160 & 630 & 70 \\ 0 & 0 & 0 & 0 & 0 \\ -70 & -630 & -1160 & -630 & -70 \\ -21 & -189 & -348 & -189 & -21 \end{bmatrix} = [7\ 63\ 116\ 63\ 7] * [3\ 10\ 0\ -10\ -3]^T$$

2.4 Approximation à la Transformée de Hough

La transformée de Hough est un outil classique de l'analyse d'images qui permet de détecter la présence de courbes ayant une forme paramétrique (droite, conique...). Nous introduisons une approximation à cette transformée.

Par définition, la transformée de Hough utilise la technique appelée principe d'accumulation d'évidence. Elle effectue cette tâche avec un accumulateur, qui dans le cas habituel, est une matrice à deux dimensions. Cette matrice est l'espace $\rho\theta$, trouvé à partir de l'équation:

$$xcos\theta + ysin\theta = \rho \tag{2.1}$$

Où

- x: colonne dans l'image d'un point différent à zero.
- y: ligne dans l'image d'un point différent à zero.
- ρ: distance du point au centre des axes xy.
- θ: angle entre l'axe x et le point.

Chaque élement de la matrice incrément sa valeur en une unité si θ et ρ accomplissent l'équation 2.1.

Si nous sommes intéressés dans les segments des droites horizontaux, ce type de lignes vont incrémenter les éléments de la matrice, où θ est égal à 90. Nous pourrions travailler donc avec les valeurs de ρ pour chercher les segments qui correspond à la partie arrière du CyCab. Mais l'implémentation de la transformée de Hough conventionnelle conduit vers un grand nombre des calculs. Pour chaque point, il faut étudier toutes les valeurs de θ et ρ que accomplissent l'équation 2.1, quand nous sommes interessés que à deux directions. Si nous sommes interessés aux les segments verticaux, nous recupererons les éléments de la matrice avec θ égal à zero ou 180 degrées.

Notre contrainte est toujours le fait de travailler en temps réel, ce qui nous améne à choisir les algorithmes les plus rapides.

Après la convolution avec les masques de Sobel, nous obtenons deux images différentes S_h et S_v. S_h se compose de tous les segments horizontaux. Ces segments sont donc ceux qui auront donné un vote pour les éléments avec θ égal à 90 degrés de la matrice de Hough. De la même façon, S_v consiste en les points qui auront donné un vote pour les éléments avec θ égal à zero degré de la matrice de Hough. Nous implémentons un vecteur accumulateur. Chaque élément du vecteur correspond à une ligne dans le cas de l'image des contours horizontaux, et à des colonnes dans le cas des countours verticaux.

Comme exemple, nous prenons le cas du vecteur des contours horizontaux.

La matrice I_y est le résultat de la convolution de la matrice de l'image I avec la masque optimale 5x5 dans la direction y:

$$I_y = I * M_y \tag{2.2}$$

Les zones considérées comme des contours, vont être celles qui ont un module plus grand qu'un seuil défini. Soit S_h la matrice des contours horizontaux, désormais appelée matrice des segments horizontaux.

$$S_h = |I_y| > seuil \tag{2.3}$$

Voire figure 2.4.

Les valeurs de la matrice S_h peuvent être:

- 1 : pixel consideré comme contour.
- 0 : autres cas.

La $dim(S_h)$ est la même $dim(I) = (L,C)$, où L est le nombre des lignes de l'image et C est son nombre des colonnes.

Soit le vecteur V_h de dimension L. Tel que:

$$V_h(i) = \sum_{j=0}^{C} S_h(i,j) \qquad avec \qquad i = 0, \ldots, L \tag{2.4}$$

Un élément i du vecteur V_h reçoit un vote, chaque fois qu'un balayage au long de la ligne i trouve un élément différent à zéro. Voire figure 2.6.

Nous allons nous servir du vecteur V_h pour trouver les invariants géometriques du CyCab.

Soit la matrice S_v des contours verticaux:

$$S_v = |I * M_x| > seuil \tag{2.5}$$

2.5 Resultats

FIG. 2.1 – *Image couleur de l'arrière CyCab*

FIG. 2.2 – *Image de niveaux de gris*

FIG. 2.3 – *Image après adjustement du contrast*

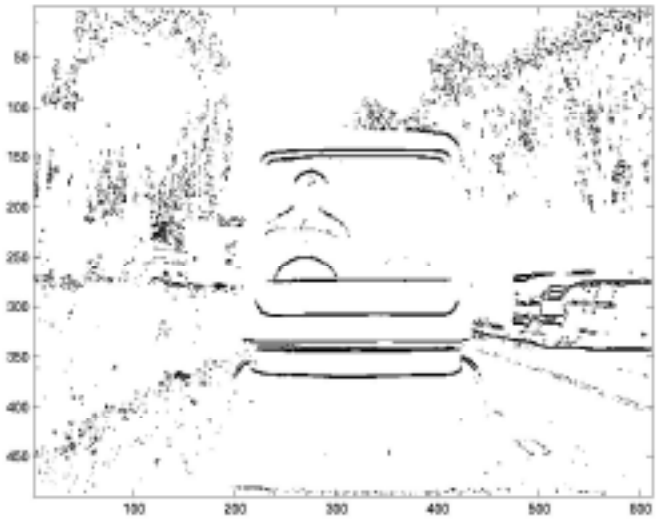


FIG. 2.4 – *Matrice S_h résultat de la convolution de l'image I avec la masque en M_y, équation 2.2, puis équation 2.3*

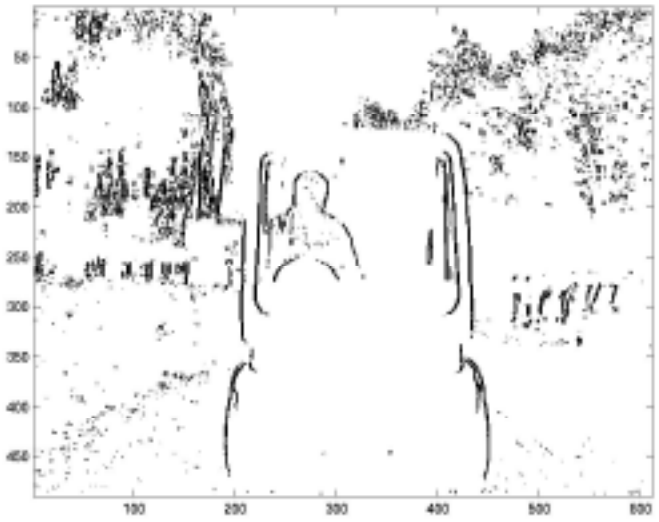


FIG. 2.5 – *Matrice S_v résultat de la convolution avec la masque M_x et seuillage.*

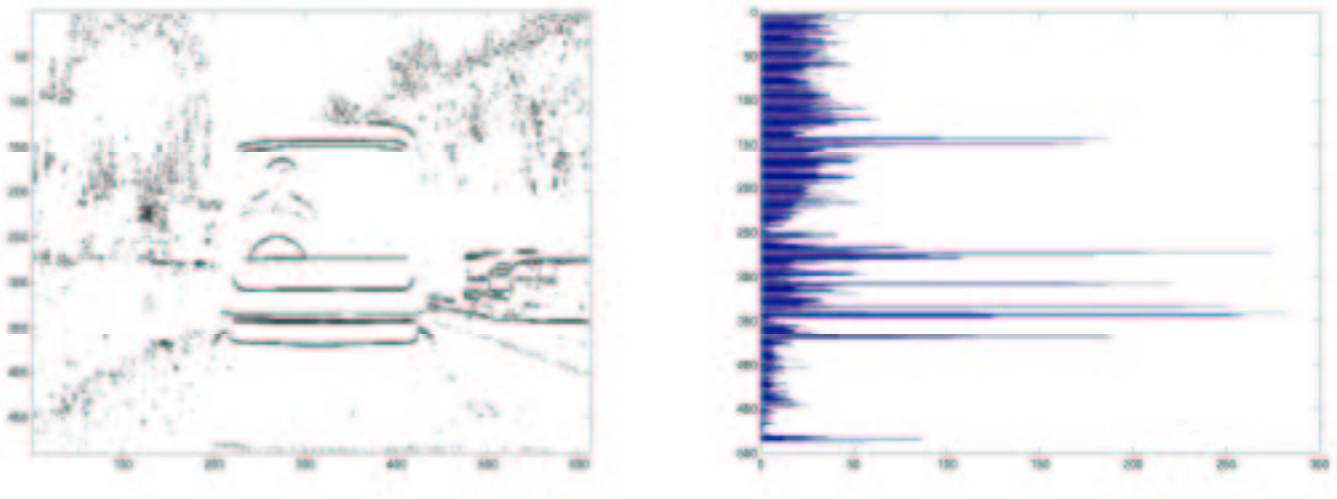


FIG. 2.6 – *Matrice des segments horizontaux et accumulateur de la quantité de points pour chaque ligne. Nous pouvons voir dans la figure de droite, les valeurs accumulées qui sont obtenues à partir de l'équation 2.4*

Chapitre 3

Invariants géométriques

Les capteurs, utilisés dans la robotique mobile , donnent des informations et des mesures sur les propriétés des objets qui se trouvent dans l'environnement proche du robot. Nous obtenons des distances aux objets du monde par rapport au capteur, la localisation des contours par rapport au capteur, etc. Toute cette information n'est pas suffisante pour que le robot sache quels sont les objets mesurés. Il faut qu'il interprète les données des capteurs pour faire des relations entre eux et les objets dans son domaine d'expérience, afin de localiser et de reconnaître ces objets [5].

L'objectif de l'algorithme est de trouver le CyCab dans une image et établir par triangulation la distance par rapport à la caméra. Nous sommes intéressé donc à la détermination des objets à partir de leur forme. Les invariants géométriques utilisent la position des segments, leur orientation, etc., des objets considérés ont une forme invariante, fixe. De la figure 2.1 l'objet à reconnaître est la partie arrière du CyCab, sans ajouter aucune cyble.

Nous définissons un modèle du CyCab en prenant en compte l'architecture et la couleur. Le bon contraste dans certains parties de l'arrière du véhicule, permet une bonne reconnaissance des contours. Ces contours vont être inclus dans le modèle étant donné qu'ils seront présents la majorité du temps, sauf cas d'occlusions spécifiques.

3.1 Définition du problème

Pour résoudre le problème de la reconnaissance de l'objet, nous allons disposer de deux informations:

- Un modèle de l'objet d'intérêt. Ce modèle décrit les proportions de l' objet.
- Un ensemble des données mesurées sur l'image d'intérêt, qui vont permettre, à partir de caractéristiques géométriques, d'obtenir la position de l'objet.

3.2 Paramétrer le modèle du CyCab

3.2.1 Mesures du CyCab

Nous prenons des mesures de la partie arriere du CyCab (toutes les mesures sont en cm).

FIG. 3.1 – *Position des points de mesure du CyCab et de distances.*

A	0	df	92.5
B	9.5	dn	99
C	13	dc	117
D	71	db	123
E	85.5		
F	96.5		
G	150.5		

3.2.2 Invariants projectifs

Soient quatre points: U, V, X, Y, qui vérifient $pos(U) < pos(V) < pos(X) < pos(Y)$.
Nous définissons [6] une relation entre ces points:

$$\tau = \frac{UX}{VX} * \frac{VY}{UY} \quad \rightarrow \quad \tau = \begin{bmatrix} UX & VY \\ VX & UY \end{bmatrix} \tag{3.1}$$

Où UX est la distance entre les points U et X. Même pour VX, VY et UY.

$$\tau = \frac{|U\ X|}{|V\ X|} * \frac{|V\ Y|}{|U\ Y|}$$

Cette relation est indépendante du facteur d'echelle. Pour prouver cette affirmation, nous considèrons que les points sont modifiés avec le même facteur de scale λ. Nous avons donc:

$$\tau' = \frac{UX'}{VX'} * \frac{VY'}{UY'}$$

$$\tau' = \frac{|U'\ X'|}{|V'\ X'|} * \frac{|V'\ Y'|}{|U'\ Y'|}$$

$$\tau' = \frac{|\lambda_U U\ \lambda_X X|}{|\lambda_V V\ ;\lambda_X X|} * \frac{|\lambda_V V\ \lambda_Y Y|}{|\lambda_U U\ ;\lambda_Y Y|}$$

$$\tau' = \frac{\lambda_U \lambda_X}{\lambda_V \lambda_X}\frac{|U\ X|}{|V\ X|} * \frac{\lambda_V \lambda_Y}{\lambda_U \lambda_Y}\frac{|V\ Y|}{|U\ Y|} = \tau$$

Pour trois points:

$$X_U = 0 \quad X_V = 1 \quad X_X = x \quad X_Y = \infty$$

$$Pos = 0, 1, x, \infty$$

Nous avons donc:

$$\tau = \frac{x}{x-1} * \frac{\infty}{\infty} = \frac{UX}{VX} \tag{3.2}$$

3.2.3 Critère de mesure

En prennant les relations 3.1 et 3.2, nous avons les suivants valeurs pour les six points mesurés du CyCab.

Quatre points		Trois points		Couleurs	
τ_{ABEF}	1.0142	τ_{BDE}	5.2414	τ_{noire}	0.1465
τ_{BCDE}	1.0115	τ_{CDE}	5	τ_{claire}	0.0940
τ_{BEFG}	3.6460	τ_{BEF}	7.9090	τ_{blue}	0.4390
τ_{CEFG}	3.5884	τ_{AEF}	8.7728		
τ_{BDEF}	1.5362	τ_{BFG}	2.6111	τ_{vcd}	1.5948
τ_{ABDE}	1.5363	τ_{CFG}	2.5463	τ_{vbd}	1.5040
τ_{ACDE}	1.0380	τ_{CEF}	7.5909		
τ_{CDEF}	1.5269	τ_{BCD}	1.0603		
τ_{BCEF}	1.0061	τ_{ACD}	1.2241		
τ_{ACEF}	1.0204	τ_{DEF}	2.3182		
τ_{ADEF}	1.5582	τ_{ABC}	3.7143		
τ_{BDEG}	2.9552	τ_{ABD}	1.1548		
τ_{CDEG}	2.8909	τ_{ABE}	1.1250		
τ_{ABCD}	3.2173	τ_{ABF}	1.1092		
τ_{BCDF}	1.0177	τ_{CDF}	3.2745		
τ_{ABDF}	1.0408	τ_{BDF}	3.4118		
τ_{ACDF}	1.0592				
τ_{ABCF}	3.3488				
τ_{ABCE}	3.3016				

TAB. 3.1 – *Relations entre les distaces mesurées du CyCab*

3.2.4 Caractérisation de la caméra

Il faut maintenant passer des données réelles à celles que nous rend la caméra video. Avec ces nouvelles données nous refaisons la table 3.2.3.

Pour caractériser la caméra, nous l'utilisons pour prendre des différents séquences du CyCab en mouvement. Des les séquences video, nous obtenons plusieurs images en format .jpg avec lesquelles nous pouvons mesurer la position des six points caractéristiques du CyCab (voire figure 3.1). Nous définissons la distance entre deux points comme le nombre des pixels de séparation. Pour normaliser la mesure des distances, nous faisons un traitement identique sur tous les images. Soit l'image I, nous faisons le traitement suivant avec des fonctions de **MATLAB**, pour obtenir les distances, équalemment mesurées.

```
Ig=rgb2gray(I);
Igf = edge(Ig,'canny');
imagesc(Igf);
```

Avec tous les distances, nous appliquons les relations 3.1 et 3.2 et nous obtenons une table similaire à la table 3.2.3. Nous faisons cet même traitement sur les images des sequences distinctes pour déterminer la moyenne des relations.

Quatre points		Trois points		Couleurs	
τ_{ABEF}	1.0166	τ_{BDE}	4.9348	τ_{noire}	0.1688
τ_{BCDE}	1.0125	τ_{CDE}	4.7010	τ_{claire}	0.1231
τ_{BEFG}	3.2555	τ_{BEF}	6.9639	τ_{blue}	0.3390
τ_{CEFG}	3.2056	τ_{AEF}	7.7056		
τ_{BDEF}	1.5715	τ_{BFG}	2.6439	τ_{vcd}	1.4301
τ_{ABDE}	1.5363	τ_{CFG}	2.5762	τ_{vbd}	1.3408
τ_{ACDE}	1.0285	τ_{CEF}	6.6873		
τ_{CDEF}	1.5600	τ_{BCD}	1.0631		
τ_{BCEF}	1.0072	τ_{ACD}	1.2298		
τ_{ACEF}	1.0239	τ_{DEF}	2.2113		
τ_{ADEF}	1.5978	τ_{ABC}	3.7641		
τ_{BDEG}	2.8511	τ_{ABD}	1.1567		
τ_{CDEG}	2.7864	τ_{ABE}	1.1250		
τ_{ABCD}	3.3145	τ_{ABF}	1.1067		
τ_{BCDF}	1.0199	τ_{CDF}	3.0161		
τ_{ABDF}	1.0456	τ_{BDF}	3.1420		
τ_{ACDF}	1.0664				
τ_{ABCF}	3.4654				
τ_{ABCE}	3.4091				

TAB. 3.2 – *Moyennes des Relations des distances du CyCab à partir des valeurs prises des échantillons*

Avec la valeur des moyennes de la table 3.2, nous calculons la désviation par rapport à la moyenne des valeurs des rélations de toutes les images. Nous définons Δ_{max} comme l'erreur relative en ayant en compte la valeur plus grande des relations de toutes les images par rapport à la moyenne. Nous définissons Δ_{min} comme l'erreur relative en prenant en compte la valeur plus petite des relations de toutes les images par rapport á la moyenne.

À partir de la table 3.3 nous définons une valeur de seuil pour chaque rélation des points. Le seuil est calculé tel que:

$$\Delta_{max} < seuil \quad et \quad \Delta_{min} < seuil$$

Quatre points			Trois points			Couleurs		
Rél.	Δ_{max}	Δ_{min}	Rél.	Δ_{max}	Δ_{min}	Rél.	Δ_{max}	Δ_{min}
τ_{ABEF}	0.006	0.003	τ_{BDE}	0.042	0.029			
τ_{BCDE}	0.010	0.003	τ_{CDE}	0.033	0.022			
τ_{BEFG}	0.093	0.218	τ_{BEF}	0.096	0.210			
τ_{CEFG}	0.091	0.226	τ_{AEF}	0.093	0.206			
τ_{BDEF}	0.116	0.043	τ_{BFG}	0.102	0.037	τ_{vcd}	0.020	0.032
τ_{ABDE}	0.003	0.002	τ_{CFG}	0.080	0.034	τ_{vbd}	0.034	0.074
τ_{ACDE}	0.011	0.004	τ_{CEF}	0.094	0.234			
τ_{CDEF}	0.106	0.042	τ_{BCD}	0.049	0.016			
τ_{BCEF}	0.009	0.002	τ_{ACD}	0.063	0.024			
τ_{ACEF}	0.015	0.004	τ_{DEF}	0.071	0.152			
τ_{ADEF}	0.122	0.045	τ_{ABC}	0.240	0.292			
τ_{BDEG}	0.029	0.021	τ_{ABD}	0.018	0.015			
τ_{CDEG}	0.025	0.021	τ_{ABE}	0.014	0.013			
τ_{ABCD}	0.196	0.314	τ_{ABF}	0.013	0.011			
τ_{BCDF}	0.019	0.005	τ_{CDF}	0.027	0.094			
τ_{ABDF}	0.007	0.005	τ_{BDF}	0.028	0.079			
τ_{ACDF}	0.026	0.008						
τ_{ABCF}	0.201	0.309						
τ_{ABCE}	0.200	0.313						

TAB. 3.3 – *Déviations des valeurs des échantillons par rapport à la valeur de la moyenne des relations*

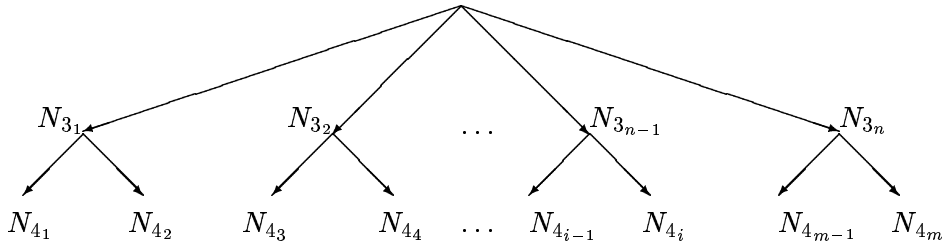
3.3 Algorithme

3.3.1 À la recherche des correspondances

Une fois déterminés les points à utiliser comme invariants géometriques, il faut définir une bonne méthode d'interpretation des données.

Une méthode pour nous guider à travers l'espace de recherche est un "Arbre d'interprétation" [5]. Chaque noyau de l'arbre représente un ensemble des points. Cet ensemble de points accomplit les relations des invariants projectifs du CyCab.

L'arbre est divisé en différents niveaux. Un niveau pour des ensembles à trois points, le niveau suivant pour des ensembles à quatre points et ainsi jusqu'au niveau des ensembles à six points.



L'algorithme commence avec trois points. Il cherche s'il y a des relations géométriques entre eux. Si la relation de ces distances correspondent à celles du CyCab, nous créons une noyau dans

Relation	Seuil	Relation	Seuil	Relation	Seuil
τ_{ABEF}	0.01	τ_{BDE}	0.05		
τ_{BCDE}	0.01	τ_{CDE}	0.05		
τ_{BEFG}	0.22	τ_{BEF}	0.22		
τ_{CEFG}	0.40	τ_{AEF}	0.21		
τ_{BDEF}	0.15	τ_{BFG}	0.15	τ_{vcd}	0.05
τ_{ABDE}	0.01	τ_{CFG}	0.10	τ_{vbd}	0.1
τ_{ACDE}	0.015	τ_{CEF}	0.24		
τ_{CDEF}	0.15	τ_{BCD}	0.05		
τ_{BCEF}	0.01	τ_{ACD}	0.10		
τ_{ACEF}	0.16	τ_{DEF}	0.16		
τ_{ADEF}	0.15	τ_{ABC}	0.30		
τ_{BDEG}	0.03	τ_{ABD}	0.02		
τ_{CDEG}	0.03	τ_{ABE}	0.02		
τ_{ABCD}	0.32	τ_{ABF}	0.02		
τ_{BCDF}	0.02	τ_{CDF}	0.10		
τ_{ABDF}	0.01	τ_{BDF}	0.10		
τ_{ACDF}	0.05				
τ_{ABCF}	0.31				
τ_{ABCE}	0.32				

TAB. 3.4 – *Table des valeurs du seuil des erreurs absolutes par rapport aux moyennes des relations*

l'arbre.

En prenant un autre point, nous vérifions s'il y a des relations entre ce point et le noyau à trois dans l'arbre. S'il y en a, nous créons un noyau dans un niveau inferieur, le niveau des ensembles à quatre points. Avec ce nouveau point, nous tentons aussi des relations de distances à trois points, avec les points anteriéures. Nous pouvons trouver des nouveaux noyaux et les placer dans le niveau à trois points de l'arbre.

Ensuite, en ajoutant des nouveaux points et en essayant des relations entre eux et les noyaux, et entre eux et tous les points existants de l'ensemble, nous remplisons l'arbre des ensembles.

L'algorithme s'arrete s'il trouve un ensemble à six points ou à cinq points avec un erreur totale (voire section 3.4 plus petite qu'un certain seuil, de l'ordre du un au cinq pourcent.

3.3.2 Étude des valeurs

À partir de ce point là, nous commençons à travailler avec tous les donnés que nous avons acquis à partir de l'image.

En prenant le vecteur V_h de la section 2.4, nous pouvons vérifier à partir de la figure 2.6 que les positions des segments du CyCab sont ceux qui ont les valeurs les plus grands. La première décision à prendre, est de choisir avec quels points commencer à travailler. Il n'y a aucun intérêt de prouver avec tous les points du vecteur V_h. Cela est très couteux en temps calcul. Nous voulons aussi que l'algorithme arrive á la bonne solution avec le plus rapide possible. Pour cela et étant donné que le critère pour arrêter la recherche est d'avoir cinq ou six points avec des relations des distances correspondants a ceux du CyCab, nous désirons de prendre ces points au début de l'algorithme.

Avec cet objectif, il faut ordonner les valeurs du vecteur V_h de mayeur à mineur (avec l'algorithme *quicksort*, par exemple) , en gardant la position original des points. Nous obtenons $V_{h,ord}$.

Les trois premiers éléments de la matrice $V_{h,ord}$ sont pris pour commencer à appliquer l'algorithme. Nous prennons la position original des ces points dans le vecteur V_h et nous les gardons dans trois variables x_i.

Nous définissons:

$$x_i = pos(V_{h,ord}(j))_{V_h}, \qquad i = 1, 2, 3 \; et \; j = 0, 1, 2$$

Et

$$x_1 < x_2 < x_3$$

L'ensemble est:

$$\Omega_3 = (x_1, x_2, x_3)$$

Avec ces trois éléments, nous construisons la **"Matrice des Distances"**.

3.3.3 Matrice de distances

Pour un ensemble Ω de n points $(x_0, x_1, \ldots, x_{n-1})$ tells que

$$x_i < x_{i+1}, \; i = 0, \ldots, n-2$$

Nous créons une matrice des distances, *MD*, tele que:

$$MD = \begin{bmatrix} a_{0,1} & a_{1,2} & .. & a_{i,i+1} & .. & a_{n-2,n-1} \\ a_{0,2} & a_{1,3} & .. & a_{i,i+2} & .. & 0 \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & a_{i,n-1} & 0 & 0 \\ .. & .. & .. & 0 & 0 & 0 \\ .. & .. & .. & .. & .. & .. \\ a_{0,n-2} & a_{1,n-1} & 0 & 0 & 0 & 0 \\ a_{0,n-1} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Chaque élément a_{ij} de la matrice *MD* est la distance entre le point i et le point j.

$$a_{ij} = pos(j) - pos(i)$$

Avec cette matrice nous pouvons essayer des relations 3.1 et 3.2 entre tous les points de Ω.

La matrice de distances de l'ensemble Ω_3 est:

$$MD_3 = \begin{bmatrix} a_{0,1} & a_{1,2} \\ a_{0,2} & 0 \end{bmatrix}$$

Nous calculons de 3.2:

$$\widetilde{\tau_x} = \frac{a_{0,2}}{a_{1,2}} \tag{3.3}$$

Pour savoir à quelle ensemble des trois point correspond Ω_3, nous créons un autre ensemble $\mathcal{E}$ des erreurs absolutes, $\mathcal{E} = (e_0, \ldots, e_{15})$, tel que:

$$e_i = \frac{\widetilde{\tau_x} - \mathcal{T}_i}{\mathcal{T}_i}, i = 0, \ldots, 15. \tag{3.4}$$

Et $\mathcal{T}_i$ est un élément de l'ensemble $\mathcal{T}$, qui prend ces valeurs des relations à trois points de la table 3.2.

$$\mathcal{T} = \begin{bmatrix} \tau_{BDE} \\ \tau_{CDE} \\ \tau_{BEF} \\ \tau_{AEF} \\ \tau_{BFG} \\ \tau_{CFG} \\ \tau_{CEF} \\ \tau_{BCD} \\ \tau_{ACD} \\ \tau_{DEF} \\ \tau_{ABC} \\ \tau_{ABD} \\ \tau_{ABE} \\ \tau_{ABF} \\ \tau_{CDF} \\ \tau_{BDF} \end{bmatrix}$$

Soit $e_{mn} = min(\mathcal{E})$ et mn la position de e_{mn} dans le vecteur $\mathcal{T}$.

Si e_{mn} est inférieur qu'un certain seuil d'erreur absolute, nous considèrons l'ensemble Ω_3 du type "mn". Ce seuil est déterminé en prenant la plus grande déviation de chaque relation de la table 3.3.

3.3.4 La direction du gradient comme contrainte

Une façon de limiter la recherche est d'ajouter des contraintes dans les points à prouver.

La première contrainte utilisée est la direction de la pente du niveux de gris, dans une discontinuitée de l'image ou le contour d'une region.

La matrice S_h obtenue de l'equation 2.3 de la section 2.4 nous donne les points considerés comme contour. Si nous multiplions cette matrice, élément par élément, avec la matrice I_y, obtenue de l'équation 2.2, nous trouvons les valeurs des directions des pentes dans les segments horizontaux.

$$S_{dh} = S_h . * I_y$$

La matrice S_{dh} possède trois valeurs dans ces éléments:

- > 0 : point de contour horizontal et de passage d'une region obscure à une région claire,
- < 0 : point de contour horizontal et de passage d'une region claire à une région obscure.
- $= 0$: non consideré comme point de contour.

Pour faire une correspondance entre les valeurs de cette matrice et les valeurs du vecteur V_h, de l'équation 2.4, nous definisons le vecteur V_{dh} de dimension L:

$$V_{dh}(i) = sign(\sum_{j=0}^{C} S_{dh}(i,j)), \qquad i = 0, \ldots, L \tag{3.5}$$

D'accord à l'architecture du CyCab, les six points de référence ont le suivant valeurs dans la direction du changement du niveau de gris:

A	+1 ou -1 (selon le fond)
B	-1
C	+1
D	-1
E	+1
F	-1

TAB. 3.5 – *Direction de peinte de niveau de gris dans les contours horizontaux*

Avec le point A, nous pouvons obtenir différentes valeurs valeur pour la pente, selon le fond. Si le CyCab a le ciel nuagé claire comme fond, la partie du châssis claire est plus obscure que le ciel, la valeur devient -1, si le fond est obscure (arbres, foret,...) la valeur de la peinte devient $+1$.

Avant considerer l'ensemble Ω_3 du type "*mn*", nous devons faire la vérification des points estimées avec la valeur de la direction de la pente.

Exemple,

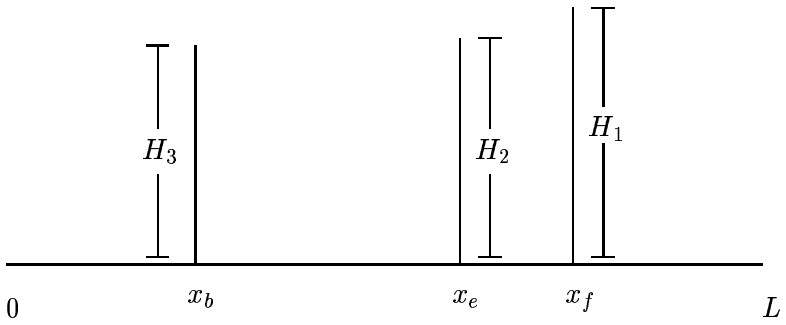
Nous prenons les points de l'ensemble Ω_3 de la section 3.3.2. À partir de ces points, il est contruite la matrice des distances MD_3. Nous calculons $\widetilde{\tau_x}$ à partir de l'equation 3.3. Si nous avons trouvé que:

$$e_2 = min(\mathcal{E}) \;\rightarrow\; \mathcal{T}(2) = \tau_{BEF}$$

C'est l'erreur absolute de valeur minimale de $\mathcal{E}$ calculée à partir de l'équation 3.4. L'indice 2 indique que l'ensemble des points Ω_3, de V_h, a l'erreur absolute minimale par rapport à la relation τ_{BEF}. Nous considérons donc comme possibles les suivants égalités: $x_1 = x_b$, $x_2 = x_e$, $x_3 = x_f$ et $\Omega_3 = (x_b, x_e, x_f)$, $\rightarrow$ ces points doivent accomplir les conditions suivants :

$$V_{dh}(x_b) = -1 \quad AND \quad V_{dh}(x_e) = +1 \quad AND \quad V_{dh}(x_f) = -1$$

Nous avons trouvé donc trois points qui peuvent correspondre à la configuration BEF. Nous ajoutons le premier noyau de l'arbre. Ces points représentent l'estructure suivant:



La hauteur de chaque segment, est donnée par la valeur de V_h dans les positions x_b, x_e et x_f. Alors:

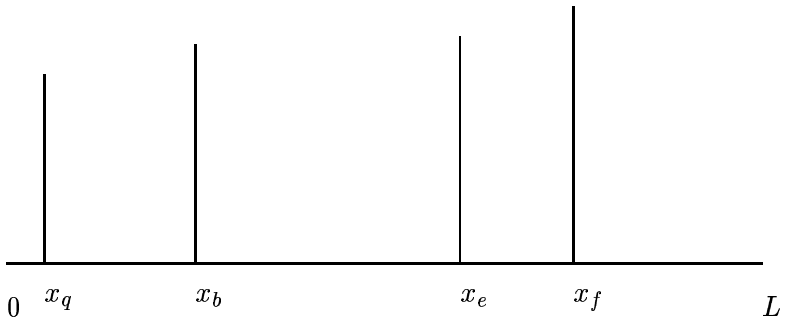
$$H_1 = V_h(x_f) \qquad H_2 = V_h(x_e) \qquad H_3 = V_h(x_b)$$

3.3.5 Nouvel point, nouvelles relations

Si nous ajoutons un nouveau point, $x_q = V_{h,ord}(3)$, le quatrième dans la matrice des points ordonnés $V_{h,ord}$, nous avons un nouvel ensemble $\Omega_4(x_1, x_2, x_3, x_4)$. Sa position definit les relations à quatre a chercher:

– $x_q < x_b$: possible point A,

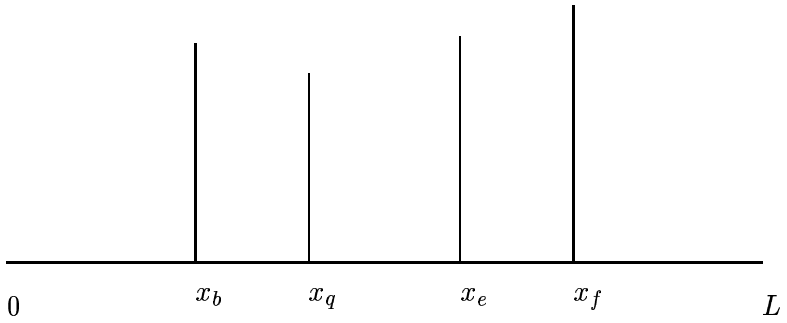
V_{dh} peut être $+1$ ou -1, on tente la relation à quatre points τ_{ABEF}. Si l'erreur absolute est inférieur que la valeur correspondante de la table 3.4, nous ajoutons un nouveau noyau dans l'arbre au niveau de quatre points. Avec ce nouveau noyau N_4, nous essayons les relations à trois points τ_{ABE}, τ_{ABF} et τ_{AEF}, si l'erreur absolute d'une de ces relations est inférieur que les seuils de la table 3.4, nous ajoutons un nouveau noyau dans l'arbre au niveau de trois points, un nouveau noyau N_3.



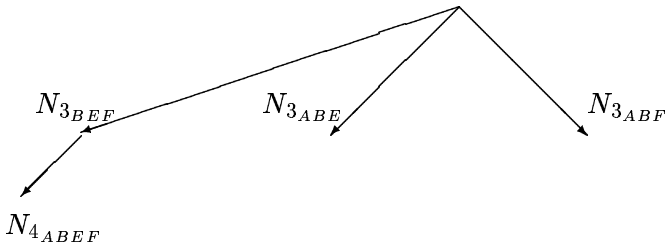
– $x_b < x_q < x_e$: possible point C ou D.

Si $V_{dh}(x_q) = +1 \Rightarrow$ nous essayons la relation à quatre points τ_{BCEF} pour ajouter un nouveau noyau N_4 à l'arbre, et les relations à trois points τ_{BCE}, τ_{BCF} et τ_{CEF} pour ajouter un nouveau noyau N_3 à l'arbre.

Si $V_{dh}(x_q) = -1 \Rightarrow$ nous essayons la relation à quatre points τ_{BDEF} pour ajouter un nouveau noyau N_4 à l'arbre, et les relations à trois points τ_{BDE}, τ_{BDF} et τ_{DEF} pour ajouter un nouveau noyau N_3 à l'arbre.



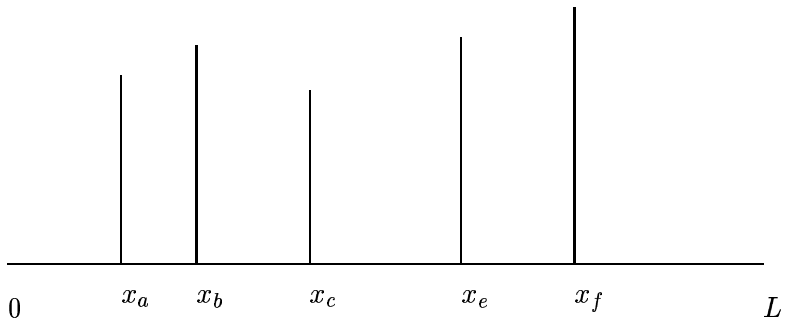
L'arbre d'interprètation peut avoir la structure suivant:



Un nouveau point $x_c = V_{h,ord}(4)$, le cinquième du vecteur $V_{h,ord}$, va conformer un nouveau ensemble $\Omega_5 = (x_1, x_2, x_3, x_4, x_5)$ et va être comparé avec tous les noyau de l'arbre. Sa position va définir la rélation à quatre à essayer avec les noyaus N_3, la rélation à cinq avec le noyau N_4 et les relations à trois avec tous les points de l'ensemble Ω_4.

Nous prenons, par example, la compairaison avec le noyau $N_{4_{ABEF}}$.

- $x_c < x_a$, il n'a pas de correspondance avec ces quatre points.
- $x_a < x_c < x_b$, même cas anterieur.
- $x_b < x_c < x_e$, possible C ou D
  - si $V_{dh}(x_c) = +1 \Rightarrow$ le point peut être le point C. Nous essayons les relations à quatre points: τ_{ACEF}, τ_{BCEF} et τ_{ABCE}. Tous ces rélations doivent avoir un erreur absolute par rapport aux valeurs de la table 3.2, inférieur au seuil correspondant dans la table 3.4 pour ajouter un nouveau noyau N_5 à l'arbre.
  - si $V_{dh}(x_c) = -1 \Rightarrow$ le point peut être le point D. Nous essayons les relations à quatre points: τ_{ADEF}, τ_{BDEF} et τ_{ABDE}. Tous ces relations doivent avoir une erreur absolute par rapport au valeurs de la table 3.2, inférieur au seuil correspondant dans la table 3.4 pour ajouter un nouvel noyau N_5 á l'arbre.



- $x_e < x_c < x_f$, il n'a pas de correspondance avec cette quatre points.
- $x_f < x_c$, même cas antérieur.

L'algorithme s'arrête si une de deux conditions s'accomplient:

1. L'erreur absolute accumulée d'un noyau de niveau cinq ou six, est inférieur qu'une certaine valeur. Nous prenons ce noyau comme vrai et nous faisons des vérificaton des couleurs. Si les constatation des couleurs ne sont pas satisfaites, nous continuons l'altorithme.
2. La "hauteur" des segments, priss du vecteur V_h, soit inférieur qu'un certain seuil.

Nous obtenons $\mathcal{N}$, le noyau résultat de la recherche de l'algorithme. Si $\mathcal{N}$ n'a pas tous ces points définis, nous estimons la position des ceux qui manquent avec les valeurs des autres points du noyau.

3.4 Erreurs

À partir du premier calcul des invariants projectifs, nous obtenons une valeur que peut avoir un erreur de déviation par rapport à la valeur considerée correcte.

1. La première erreur est calculée à partir de l'équation 3.4 pour trois points. Nous obtenons e_3. Nous gardons cette erreur pour la prochaine étape de l'algorithme, si elle est inférieur qu'un certain seuil. Nous obtenons donc un noyau de niveau trois.

2. Dans l'étape de recherche à quatre points, nous faisons un calcule d'une erreur absolute à travers l'équation 3.4, en calculant $\widetilde{\tau_x}$ à partir de la relation 3.1. $\mathcal{T}$ prends maintenent la valeur de la relation de quatre points, d'accord à la position du nouveau point. Nous obtenons e_4. Si cette erreur est inférieur qu'un certain seuil, Er, nous avons un nouveau noyau de niveau quatre. Nous faisons l'adition des "hauteurs" des quatre points, que sont les valeurs dans le vecteur V_h.

$$p_4 = \sum_{i=0}^{4} V_h(N_{4_{x(i)}})$$

   En ayant $v_{h_{max}}$ la valeur maximale du vecteur V_h, nous calculons une expression de l'erreur rélative à cette valeur des poids des points:

$$e_p = \frac{4 * v_{h_{max}} - p_4}{4 * v_{h_{max}}}$$

   Pour calculer l'erreur total de ce noyau, nous prenons e_3, e_4 et e_p et nous calculons e_t. Dans cette formule, nous aurons différents poids pour chaque erreur. L'erreur que nous voulons mettre en valeur est celle ci de la relation à quatre points par rapport à cesses de la relation à trois et des poids. Nous le donnons donc un poids du cinquant pour cent et l'autre cinquant pour cent est distribué entre e_3 et e_p.

$$e_t = \frac{0.5 * e_4 + 0.4 * e_3 + 0.1 * e_p}{0.5 * Er + 0.4 + 0.1} \tag{3.6}$$

   En calculant l'erreur avec cette formule, nous assurons que l'erreur totale est normée:

$$0 \leq e_t \leq 1$$

3. Pour tenter un cinquieme point dans un noyau de quatre, nous faisons sucesivement trois relations à quatre avec le nouveau point et les autres quatre du noyau. Nous obtenons e_{4_1}, e_{4_2} et e_{4_3}. Si ces trois erreurs sont inférieures qu'un seuil spécifique pour chacun d'eux (voire table 3.4), nous avons un nouveau noyau dans le niveau cinq. Nous calculons le nouveau e_p:

$$p_5 = \sum_{i=0}^{5} V_h(N_{5_{x(i)}})$$

$$e_p = \frac{5 * v_{h_{max}} - p_5}{5 * v_{h_{max}}}$$

   Soit $e_{t_{ant}}$ l'erreur totale antérieure du noyau N_4. L'erreur totale du nouveau noyau est:

$$e_t = \frac{0.166 * e_{4_1} + 0.166 * e_{4_2} + 0.166 * e_{4_3} + 0.4 * e_{t_{ant}} + 0.1 * e_p}{0.166 * Er_1 + 0.166 * Er_2 + 0.166 * Er_3 + 0.4 + 0.1} \tag{3.7}$$

Cette erreur est comparée avec un seuil très bas. Si elle est inférieure que ce seuil, nous arrêtons l'algorithme, en prennant cet ensemble des points N_5 comme égal à $\mathcal{N}$. Nous estimons la position du sixième point par rapport à la valeur des autres.

4. La méthode pour tenter un sixiéme point dans un noyau de niveau cinq, est la même que l'anterieure. Nous avons un nouveu p_6 à calculer:

$$p_6 = \sum_{i=0}^{6} V_h(N_{6_{x(i)}})$$

$$e_p = \frac{6 * v_{h_{max}} - p_6}{6 * v_{h_{max}}}$$

L'équation du calcul de l'erreur totale, est fait de la même maniere que l'equation 3.7. Si e_t est inférieure qu'un certain seuil, on prend ce noyau N_6 égal à $\mathcal{N}$.

Chapitre 4

Détection de la couleur

4.1 Transformation de l'espace de la couleur

Au départ les camescopes et autres outils de capture vidéo acquiérent l'image à travers un capteur CCD que convertit la lumiere reçue en un triplet de trois valeurs (trois composantes) pour chaque pixel de l'image. Il y a plusieurs espaces de couleur en utilisation: Hue Saturation Intensity (HSI), YUV et Rouge Vert Bleue (RVB). Le codage couleur RVB (noté RGB, Red-Green-Bue) en anglais) permet, par combinaison, de reconstituer toutes les couleurs. Chaque composante est définie généralement sur huit bits (un octect).

Le RVB des couleurs est le codage utilisé par la télévision ou les écrans d'ordinateur pour reproduire les couleurs.Il est aussi une codage trés utilisé dans le traîtement d'image. Mais, il n'est pas trés bien adapté pour les applications en vision robotique. Avec l'objectif de pouvoir identifier un objet à travers sa couleur, on a besoin d'un classificateur robust contre les variations de luminance. Cela peut être fait avec RGB, mais le volume d'implicance pour cette relation a une forme conique et il ne peut pas être representé pour un simple seuillage.
D'autre part, HSI et YUV ont l'advantage que la chrominance est codée en deux dimensions (H et S pour HSI et UV pour YUV), et la luminance dans la troisième. Donc, une particuliére couleur peut être reprèsentée comme une "colonne" des valeurs.

4.2 Le codage YUV

Historiquement et techniquement, les signaux vidéos analogiques couleurs sont transportés dans un autre mode de représentation que le RGB, nous utilisons un systéme nommé YUV, où Y représente la luminance et U, V les composantes Rouge et Bleu (chrominances). Ces trois informations permettent de restituer à la fin les composantes RGB et apportent l'avantage de permettre une compression facile des couleurs et de fiabiliser le transport du signal vidéo. Mais surtout la partie Y (luminescence) permet une totale compatiblilité avec les anciens équipements noir et blanc, ce qui a permis le déploiement de la diffusion couleur alors que la totalité des foyers étaient alors équipés de téléviseurs noir et blanc.

Ici, nous pouvons voir une des relationes liant Y, U et V à R,G et B qui sont d'habitude utilisées:

- $Y = 0.299 * R + 0.587 * G + 0.114 * B$
- $U = -0.1687 * R - 0.3313 * G + 0.5 * B$
- $V = 0.5 * R - 0.4187 * G - 0.0813 * B$

Soit l'image $I_{rgb}(x, y)$ dans l'espace couleur RGB. En appliquant les dernières relations, nous obtenons l'image $I_{yuv}(x, y)$:

$$I_{rgb}(x, y) \overset{RelationRGB \rightarrow YUV}{\rightharpoonup} I_{yuv}(x, y)$$

4.3 Identification de la couleur

Dans notre application, chaque couleur peut être spécifiée avec un ensemble de six seuils, deux pour chaque dimension de l'espace de couleur. Comme un exemple, pour définir si un pixel est membre d'une classe particulière de couleur, nous pourrions faire la comparaison de la façon suivante:

```
if((Y >= Ylowerthresh)
AND (Y <= Yupperthresh)
AND (U >= Ulowerthresh)
AND (U <= Uupperthresh)
AND (V >= Vlowerthresh)
AND (V <= Vupperthresh))
pixel_color = color_class;
```

Par contre, cette méthode nécessite de six conditions pour determiner l'appartenence de chaque pixel à une classe de couleur. Cela devient très chèr en temps de calcul.

La méthode à utiliser [7] décompose chaque classe de couleur en trois vecteurs. Un vecteur pour chaque dimension de l'espace couleur. Une classification de couleur pour un pixel peut se calculer à travers d'une opération AND avec les trois vecteurs correspondants à la couleur à tester:

```
pixel_class = YClass[Y]
AND UClass[U]
AND VClass[V];
```

Nous n'avons donc que trois opérations AND, qui sont beaucoup moins coûteuses que la comparaison entre entiers.

Comme un exemple, nous numérisons l'espace de couleur YUV en dix niveaux dans chaque dimension. Le "bleu" par exemple peut être représenté par l'assignation des valeurs suivants:

```
YClass[]=[0,0,1,1,1,1,1,1,0,0];
UClass[]=[0,0,0,0,1,1,1,0,0,0];
VClass[]=[0,0,0,0,0,1,1,0,0,0];
```

Pour savoir si un pixel avec les valeurs (2,5,6) est un member de la classe de couleur "bleue", nous évaluons l'expression:

```
pixel_class = YClass[2] AND UClass[5] AND VClass[6];
```

La valeur un ou *true* de la variable *pixel_class* indique que le couleur du pixel est de la classe "bleue".

4.4 Les identifications des multiples couleurs

Une des avantages de la méthode, est la possibilité de déterminer la correspondance d'un pixel en plusieurs classes de couleurs en *simultanéité*. En profitant le parallelisme en l'opération *AND*, on peut déterminer la correspondance de plus d'une classe de couleur.

Comme un exemple, nous supposons que la region de l'espace cette couleur qui identifie le "rouge" est représenté avec:

```
YClass[]=[0,1,1,1,1,1,1,1,0,0];
UClass[]=[0,1,1,1,0,0,0,0,0,0];
VClass[]=[0,0,0,0,0,0,0,1,1,0];
```

Nous pouvons combiner les vecteurs des espaces "bleu" et "rouge" en utilisant la position des éléments:

```
YClass[]=[00,10,11,11,11,11,11,11,00,00];
UClass[]=[00,10,10,10,01,01,01,00,00,00];
VClass[]=[00,00,00,00,00,01,01,10,10,00];
```

Où le premier bit (high-order) de chaque élément représent le "rouge" et le seconde bit est utilisé pour représenter le "bleu". Nous pouvons vérifier si (1,3,7) est d'une des deux classes en évaluant l'expression:

```
pixel_class = YClass[1] AND UClass[3] AND VClass[7];
```

Le résultat est $pixel\_class = 10$, en indicant que la couleur est "rouge", mais n'est pas "bleue".

4.5 Les valeurs des espaces de couleur

Les valeurs des espaces de couleur sont fortement lié à la camera utilisée. Il faut trouver les valeurs des éléments des trois vecteurs YUV pour les classes de couleur que nous sommes intéressé. Pour notre caméra, nous obtenons les suivantes répresentationes pour les classes "bleue", "noire" et "claire".

Nous numérisons les valeurs de chaque dimension de couleur dans vecteurs de 64 éléments.

- Soient les vecteurs *Vby*, *Vbu* et *Vbv* de dimension 64, qui définent la classe de couleur "bleue".

Y	U	V
13	34	24
..	35	25
..	36	26
..	37	27
..	38	28
49	39	

TAB. 4.1 – *Éléments des vecteurs égaux à un pour la classe "bleue" dans un espace de 64 valeurs*

Les valeurs de la table 4.1 répresentent les éléments du vecteur de valeur différent à zero.

Y	U	V
28	30	30
..	31	31
..	32	26
55		

TAB. 4.2 – *Éléments des vecteurs égaux à un pour la classe "claire" dans un espace de 64 valeurs*

- Soient les vecteurs *Vcy*, *Vcu* et *Vcv* de dimension 64, qui définent la classe de couleur "claire". Les valeurs de la table 4.2 répresentent les éléments du vecteur de valeur différent à zero.
- Soient les vecteurs *Vny*, *Vnu* et *Vnv* de dimension 64, qui définent la classe de couleur "noire".

Y	U	V
8	31	29
..	32	30
..	33	31
27		

TAB. 4.3 – *Éléments des vecteurs égaux à un pour la classe "noire" dans un espace de 64 valeurs*

En prenant l'image $I_{yuv}(x, y)$, nous appliquons la méthode de la section 4.3 pour la classe bleue, claire et noire. Trois matrices binaires sont le résultat de l'application de la méthode antérieure:

- *Rb(x,y)*: qui contient des pixels des valeurs:
  - 1: pixel consideré de la classe bleue,
  - 0: autrement.
- *Rc(x,y)*: qui contient des pixels des valeurs:
  - 1: pixel consideré de la classe claire,
  - 0: autrement.
- *Rn(x,y)*: qui contient des pixels des valeurs:
  - 1: pixel consideré de la classe noire,
  - 0: autrement.

En reprenant l'image de la section des résultats:

4.6 Traitement Morphologique

Le traitement morphologique binaire permettra d'avoir des images avec des régions beaucoup mieux délimites, sans bruit et plus complètes. Dans le cas des régions bleues cela va nous servir pour l'etiquetage des ces régions (s'il y a beaucoup de bruit, chaque petit pixel serait une nouvelle région). Dans le cas de la classe noire, il'est important le traitement morphologique pour compléter

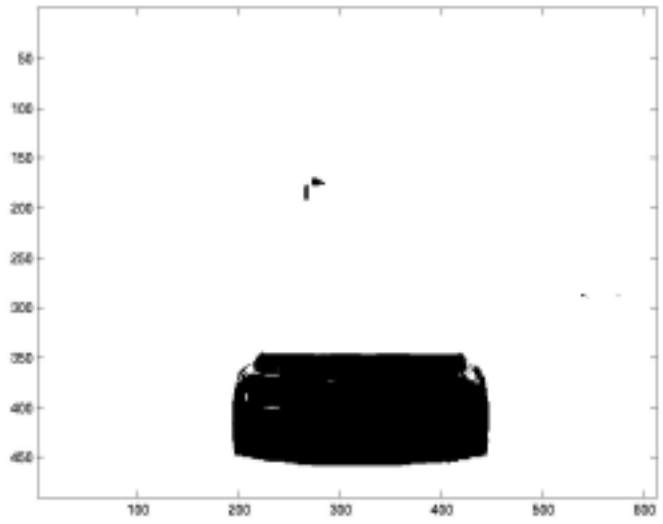


FIG. 4.1 – *Pixels membres de la classe Bleue*

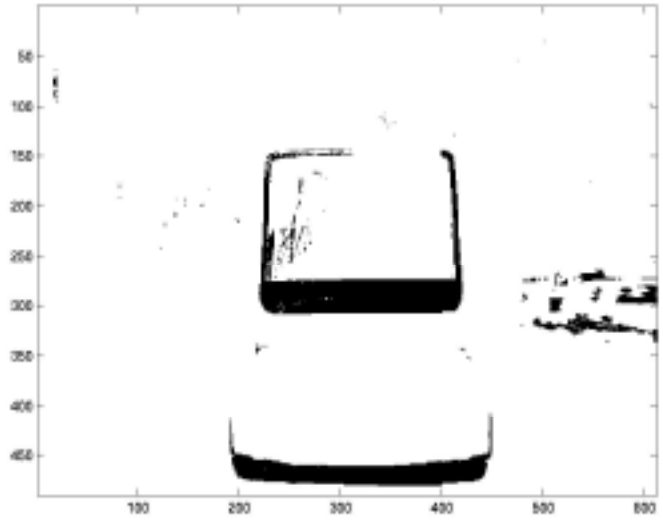


FIG. 4.2 – *Pixels membres de la classe Noire*

la bande noire autour de la fenêtre. Dans le cas de la classe claire, il y a beaucoup de bruit qui peut être éliminé avec cette méthode.

À partir de l'érosion et de la dilatation, nous pouvons définir deux nouvelles opérations: l'overture et la fermeture [2]. L'overture servira à supprimer les fausses alarmes, c'est à dire le bruit. La fermeture supprimera les trous des regions. Ces deux opérations permettent de conserver les proportions des régions.

À l'image bleue, nous appliquons deux overtures et deux fertetures pour améliorer la définition des régions. Le résultat est l'image 4.4.

FIG. 4.3 – *Pixels membres de la classe Claire*

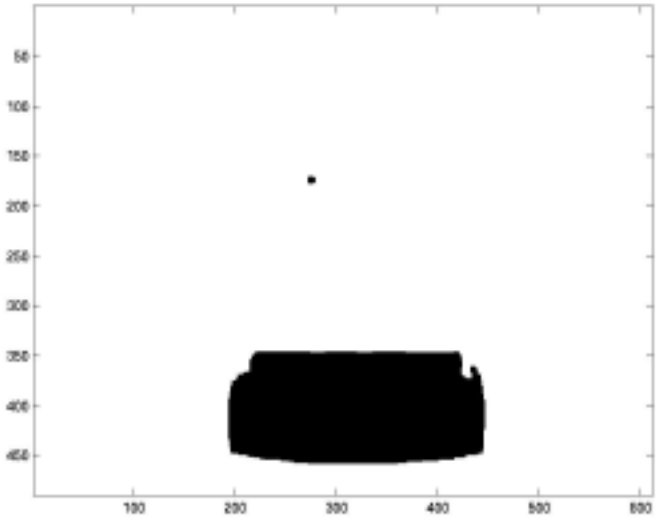


FIG. 4.4 – *Pixels membres de la classe Bleue, après le traitement morphologique*

À l'image des pixels noires, nous appliquons la dilatation pour la suppression des trous. Le résultat est l'image 4.5.

À l'image des pixels claires, nous appliquons l'érosion pour l'élimination du bruit. Le resultat est l'image 4.6.

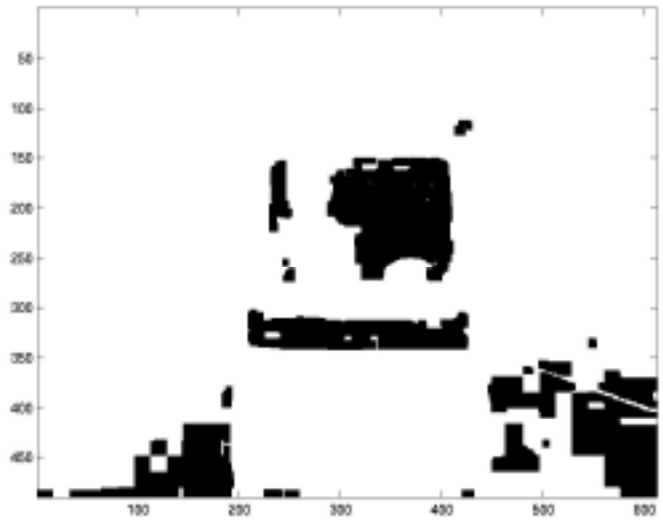


FIG. 4.5 – *Pixels membres de la classe Noire, après le traitement morphologique*

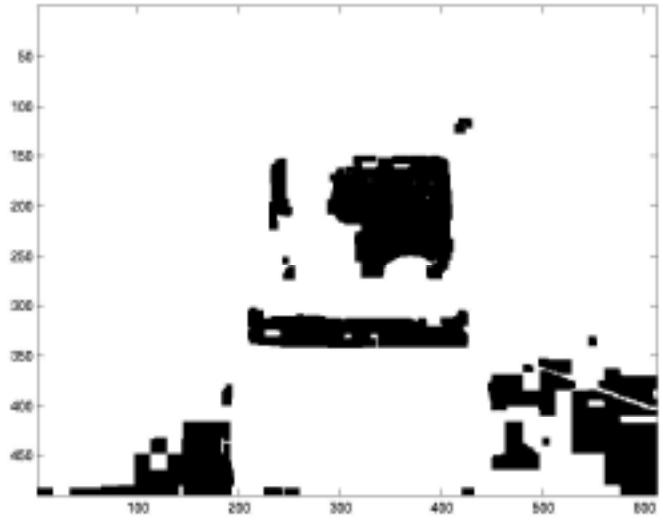


FIG. 4.6 – *Pixels membres de la classe Claire, après le traitement morphologique*

4.7 La contrainte couleur

Dans le cas, où aucune noyau ou ensemble des points trouvés avec l'algorithme des invariantes géometriques a une erreur inférieure qu'un certain seuil, nous devons choisir la solution entre les noyaux de l'arbre. L'élection va être prise à partir des erreurs de chaque noyau et d'une vérification de la couleur.

Pour cette vérification, nous utilisons les matrices obtenues dans la section 4.6. Nous faisons l'histrogramme de quantité de points de chaque ligne pour les trois matrices, en obtenant trois

vecteurs qui sont mises ensembles pour créer la matrice
$sumRegCol$ de taille $(3, C)$. Nous avons donc:

$$sumRegCol(1,i) = \sum_{j=0}^{C} Rb(i,j)$$

$$sumRegCol(2,i) = \sum_{j=0}^{C} Rc(i,j)$$

$$sumRegCol(3,i) = \sum_{j=0}^{C} Rn(i,j)$$

pour $i = 0, ..., L$.

Pour la vérification de la couleur, nous allons faire une mesure de la surface de couleur trouvée en R_b, R_c et R_n, et la comparer avec la surface que devrait avoir selon les dimensions des segments trouvés.

Si nous pensons idéalement la région comme carrée, la surface peut être calculée à partir de:

$$\tilde{s} = dv * dh$$

où dv est la distance vertical, et dh est la distance horizontal.

On peut calculer, aussi, une relation des distances horizontals et verticals en prennant les valeurs des échantillons et en calculant la moyenne:

$$\tau = \frac{dv}{dh}$$

Si nous prenons une des relations à trois adéquate de la table 3.2, nous pourrons calculer de la surface de couleur que nous devrions trouver avec la position des points du noyau que nous sommes en train d'essayer.

En prennant le cas Bleu.

$$\tilde{s}_{bleue} = FG * db \qquad \tau_{bleue} = \frac{FG}{db}$$

La relation à trois qu'on chosit est τ_{BFG}. L'équation de la surface est:

$$\tilde{s}_{bleue} = (\frac{BF}{\tau_{BFG}})^2 * \frac{1}{\tau_{bleue}} \tag{4.1}$$

$\tilde{s}_{bleue}$ indique la surface de la couleur bleue que nous devrions trouver dans l'image, entre la ligne à la position F et la ligne à la position G. Pour savoir que surface nous avons dans la matrice Rb entre les points F et G, nous faisons l'adition des valeurs de la matrice $sumRegCol$ entre la position F et la position G du noyau.

$$\hat{s}_{bleue} = \sum_{i} sumRegCol(1,i) \qquad i = F, \ldots, G$$

et l'erreur:

$$e_{bleue} = |\frac{\tilde{s}_{bleue} - \hat{s}_{bleue}}{\tilde{s}_{bleue}}|$$

De cette façon $\hat{s}_{bleue}$ devra avoir une valeur plus grande d'un 70 pour cent de la valeur de $\tilde{s}_{bleue}$ ($e_{bleue} > 0.70$) pour considérer que la vérification de la couleur bleue s'est accomplie.

Dans le cas de la vérification du noir, les équations sont:

$$\tilde{s}_{noire} = DE * dn \qquad \tau_{noire} = \frac{DE}{dn}$$

et

$$\tilde{s}_{noire} = \frac{DE^2}{\tau_{noire}} \tag{4.2}$$

La surface trouve à partir de Rn est:

$$\hat{s}_{noire} = \sum_i sumRegCol(2,i) \qquad i = D, \ldots, E$$

et l'erreur:

$$e_{noire} = |\frac{\tilde{s}_{noire} - \hat{s}_{noire}}{\tilde{s}_{noire}}|$$

Dans le cas de la vérification de la classe claire, les equations sont:

$$\tilde{s}_{claire} = EF * dc \qquad \tau_{claire} = \frac{EF}{dc}$$

et

$$\tilde{s}_{claire} = \frac{EF^2}{\tau_{claire}} \tag{4.3}$$

La surface trouvée à partir de Rc est:

$$\hat{s}_{claire} = \sum_i sumRegCol(3,i) \qquad i = E, \ldots, F$$

et l'erreur:

$$e_{claire} = |\frac{\tilde{s}_{claire} - \hat{s}_{claire}}{\tilde{s}_{claire}}|$$

4.8 L'identification des régions bleues

Soit la matrice *Rb(x,y)* binaire, des points avec les valeurs:

- 1: point consideré dans la classe de couleur "bleue" avec la méthode de la section 4.3.
- 0: point hors de la classe "bleue".

Avec cette matrice, nous allons identifier les différents régions bleues de l'image.

4.8.1 L'etiquetage en composantes connexes

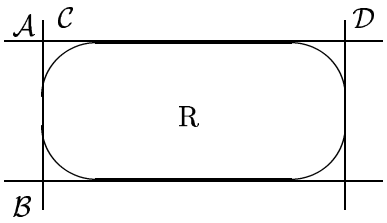
L'étiquetage en composantes connexes est la base de quasiment toutes les chaînes algorithmiques en traitement d'images des qu'il s'agit d'analyser des régions dans une scène [8]. Rosenfeld a révolutionné le domaine en proposant un algorithme qu'à différence avec l'étiquetage par balayages multiples, ne comprenant que deux passes sur l'image.

L'algorithme de Rosenfeld requiert l'utilisation d'une table d'équivalences. Dans le premier balayage de l'image, une étiquette est assignée à chaque point de la matrice *Rb(x,y)* différente à zero. Une nouvelle étiquette est assignée si ce point n'a pas un voisin déjà étiqueté. Si le voisinage du point courant comporte un ou plusieurs points déjà étiquetés, la valeur minimale de ces étiquettes est choisie et les conflits sont stockés dans la table d'équivalences. Les équivalences sont résolues à la fin du premier balayage, et c'est dans le second balayage où nous attribuons à chaque pixel son étiquette définitive.

Nous utilisons l'algorithme de Rosenfeld pour étiqueter l'image, en obtenant une matrice des propietés des régions bleues, *Mpr*. Cette matrice a les suivants composants pour chaque région i:

$$Mpr(i,:) \Longrightarrow \boxed{\mathcal{A}_i \mid \mathcal{B}_i \mid \mathcal{C}_i \mid \mathcal{D}_i \mid Poid_i}$$

Où:



$\mathcal{A}_i$,$\mathcal{B}_i$,$\mathcal{C}_i$ et $\mathcal{D}_i$ sont les extrêmes de la boîte qui contient la région R. L'élément $Poid_i$ indique la quantite des points de la région.

4.8.2 La recherche du chassis

Nous allons chercher dans la matrice Rb la région qui a dans sa taille, une relation géometrique avec les points du noyau $\mathcal{N}$, résultat de l'algorithme de la recherche.

Nous prenons la relation τ_{BEFG}. D'accord à l'équation 3.1, la rélation antérieure s'écrit:

$$\tau_{BEFG} = \frac{BF}{EF} * \frac{EG}{BG}$$

Nous comptons avec les valeur B et E du noyau $\mathcal{N}$. Pour les valeurs F et G nous prenons les propietées des régions dans la matrice M_{pr}:

$$F = \mathcal{A}_i \quad et \quad G = \mathcal{B}_i$$

Nous avons donc:

$$\tilde{\tau}_{BEFG_i} = \frac{B\mathcal{A}_i}{E\mathcal{A}_i} * \frac{E\mathcal{B}_i}{B\mathcal{B}_i}$$

L'erreur absolute est:

$$e_i = |\frac{\tilde{\tau}_{BEFG_i} - \tau_{BEFG}}{\tau_{BEFG}}|$$

Si cet erreur est inférieure qu'un certain seuil, nous considèrons cette région, comme la partie bleue du châssis du CyCab.

Trouver cette région est aussi une autre vérification des points du noyau $\mathcal{N}$. Si nous ne trouvons pas une région, ce noyau est négligé.

Chapitre 5

La recherche des segments verticaux

Nous allons chercher les segments verticaux pour compléter les données du modèle du CyCab dans l'image. L'utilisation de ces segments peut être encadrée dans la détection d'un virage. Si les segments avaient au départ une séparation l, quand le véhicule commence un virage, cette séparation va diminuer un Δl. Nous pourrions donc détecter un virage quand le Δl accumulé dans quelques images prend une valeur importante. Le signe du Δl pourrait nous donner aussi la direction du virage.

5.1 Algorithme

Nous allons reprend:

- la matrice S_v résultat de l'équation 2.5,
- la matrice Rn de la section 4.5 après le traitement morphologique,
- la matrice M_{pr} qui a les propretés de la région trouvé dans la section 4.8.2,
- le noyau résultat $\mathcal{N}$.

Nous prenons les extrêmes $\mathcal{C}$ et $\mathcal{D}$ de la matrice M_{pr} et les positions des points B et D du noyau $\mathcal{N}$. Avec ces valeurs nous construisons une enveloppe. Nous obtenons des nouvelles matrices S_{v2} et Rn_2 avec les valeurs des S_v et Rn dedans l'enveloppe. Voire figure 5.1.

Avec ces matrices nous faisons l' histogramme des pixels des colonnes. Nous obtenons les vecteurs $sumSegVert$ et $sumRegVert$. Voire figure 5.2.

Avec ces vecteurs nous calculons deux matrices des distances de la même façon que dans la section 3.3.3. Les matrices obtenuées sont M_{dsv} pour le vecteur $sumSegVert$ et M_{drv} pour le vecteur $sumRegVert$.

5.2 Rélation

Soient les matrices des distances pour n points du type:

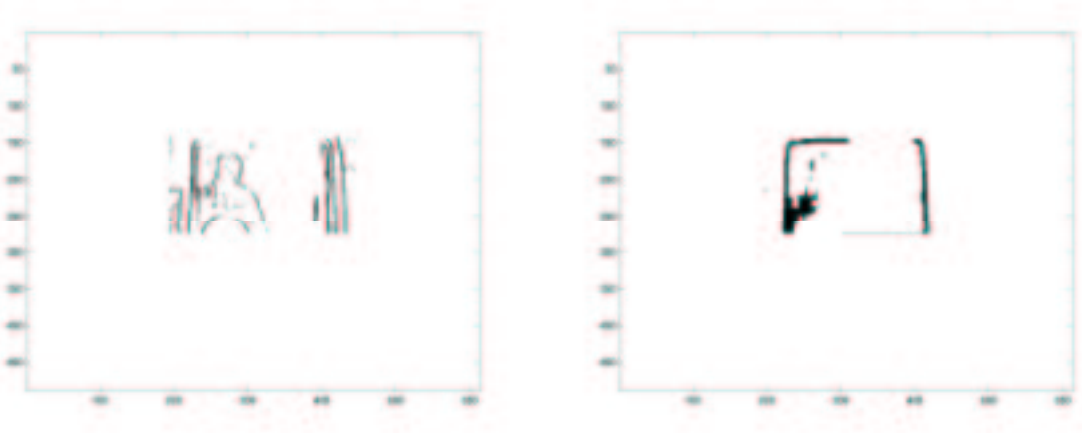


FIG. 5.1 – *Matrices S_{v2} et Rn_2*

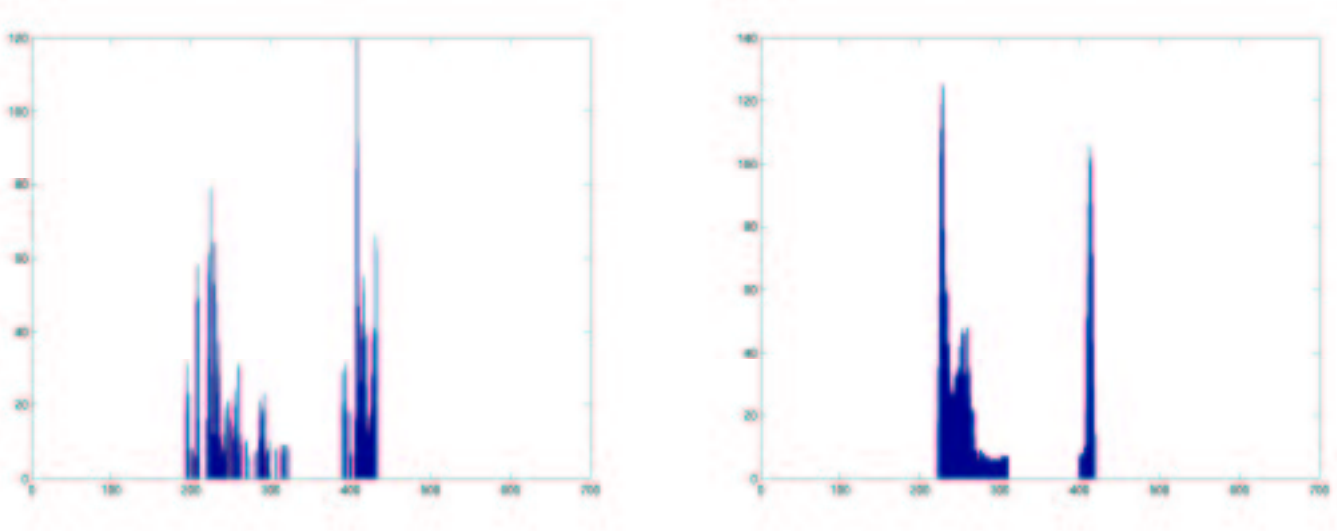


FIG. 5.2 – *Vecteurs $sumVertSeg$ et $sumRegVert$*

$$MD = \begin{bmatrix} b_{0,1} & b_{1,2} & .. & b_{i,i+1} & .. & b_{n-2,n-1} \\ b_{0,2} & b_{1,3} & .. & b_{i,i+2} & .. & 0 \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & b_{i,n-1} & 0 & 0 \\ .. & .. & .. & 0 & 0 & 0 \\ .. & .. & .. & .. & .. & .. \\ b_{0,n-2} & b_{1,n-1} & 0 & 0 & 0 & 0 \\ b_{0,n-1} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La rélation à trouver va être:

$$\tau_v = \frac{b_{i,j}}{df} \tag{5.1}$$

Où $b_{i,j}$ est une des valeurs de les matrices des distances et df est la hauteur de la fenêtre, distance entre les points B et D du noyau résultant $\mathcal{N}$. La rélation τ_v devra avoir une valeur inférieur au

valeur τ_{vbd} de la table 3.2. L'utilisation de l'équation 5.1 nous permet de rélationar les contours verticaux et les contours horizontaux. L'election des points verticaux sera réduit à ceux qui ont une rélation avec le modèle du CyCab trouvé. L'utilisation de cette algorithme sera limitée pour la recherche du véhicule dans le cas du suivi longitudinal. Le cas de suivi lateral n'a pas été étudiée dans le présent stage.

Pour la distance df, les vecteurs $sumVertSeg$ et $sumRegVert$ nous obtenons deux matrices de résultats. Les matrices sont M_{rv}^{seg} et M_{rh}^{reg}. Chaque ligne i de ces matrices a les suivant éléments:

$$M_{pr}^{seg}(i,:) \Longrightarrow \boxed{pg_i \mid pd_i \mid distance_i \mid er_i}$$

$$M_{pr}^{reg}(i,:) \Longrightarrow \boxed{pg_i \mid pd_i \mid distance_i \mid er_i}$$

Où:

- pg_i est la position du vecteur gauchier.
- pd_i est la position du vecteur droite.
- $distance_i$ distance entre pg_i et pd_i.
- er_i erreur de la rélation

$$er_i = |\frac{\tau_{v_i} - \tau_{vbd}}{\tau_{vbd}}| < 0.1$$

5.3 Croiser les résultats

À partir des matrices M_{rv}^{seg} et M_{rh}^{reg}, nous allons choisir le résultat de l'algorithme. Nous allons comparer la positions des couples des points de la matrice M_{rh}^{reg} par rapport aux couples des points de la matrice M_{rv}^{seg}.

Nous définions e^g et e^d telles que:

$$e_i^g = |\frac{M_{rv}^{seg}(i,0) - M_{rv}^{reg}(j,0)}{M_{rv}^{seg}(i,0)}|$$

$$e_i^d = |\frac{M_{rv}^{seg}(i,1) - M_{rv}^{reg}(j,1)}{M_{rv}^{seg}(i,1)}|$$

$$i = 0, \ldots, nroLignM_{rv}^{seg} \qquad et \qquad j = 0, \ldots, nroLignM_{rv}^{reg}$$

La décision pour le résultat final va être prise entre les valeurs des couples des matrices M_{rv}^{seg} et M_{rh}^{reg} qui ont les deux erreurs e^g et e^d inférieurs à 0.05. Dans le cas que nous ayons plusieurs couples, nous allons choisir celle qui a l'hauteur plus importante. Cette hauteur est calculée des les segments $sumSegVert$ et $sumRegVert$.

Le résultat est le vecteur $vectSegVert$ qui a les éléments suivantes:

$$vectSegVert \Longrightarrow \boxed{pg \mid pd \mid er}$$

Chapitre 6

Le suivi des segments

Le suivi d'une primitive d'image, dans notre cas des segments de droite, à travers une séquence d'images monoculaire va être une autre contrainte. Cette nouvelle contrainte va nous servir pour l'aceptation du résultat du noyau $\mathcal{N}$.

Le problème de l'appariement de deux segments est résume par la suite:

1. Prédiction de la position du segment
2. Appariement avec le segment correspondant du noyau résultat $\mathcal{N}$.

6.1 Les étapes du suivi

Le suivi des segments de droite dans une séquence d'images est réalisé par un algorithme comportant les suivantes phases distinctes:

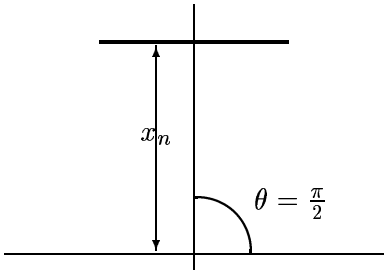
- Assignation d'un modèle cinématique à chaque ensemble des paramètres de la représentation du segment de droite.
- Utilisation du modèle pour prédire la valeur des paramètres dans la prochaine image ainsi que l'incertitude associée.
- Détermination de l'air de recherche autour la position prédite.
- Trouver l'appariement des prédictions dans le noyau $\mathcal{N}$.
- Losrqu'un appariement est trouvé, utiliser le noyau $\mathcal{N}$ pour mettre à jour le modéle cinématique.

Ainsi nous faisons correspondre à chacun des segments du noyau $\mathcal{N}$ de la première image une autre noyau résultat des images suivantes qui permet de représenter la trajectoire cinématique des segments du CyCab au cours du temps.

6.2 Représentation du segment de droite

Pour suivre les segments nous devons d'abord les représenter par des paramètres.

Nous travaillons avec segments horizontaux, ceux qui nous rendrent toujours une orientation θ égal à 90 dégres. Notre algorithme ne nous rend pas la longueur du segment, mais, nous avons la position du segment dans l'axe vertical.



6.3 L'appartiement des segments

L'appariement des egments consiste à calculer la distance absolute entre le segment de la prédiction et ce qui a été trouvé par l'algorithme. En considérant x_p la position du segment de la prédiction et en considérant de même x_n la position du segment correspondant du noyau résultat, x_n doit vérifier:

$$d = \frac{|x_n - x_p|}{L} < \epsilon$$

6.4 La méthode du filtre scalaire

Nous pouvons maintenant définir, à partir de la représentation, le filtre qui va permettre de suivre les paramètres du segment à travers le temps. Nous devons implanter un filtre scalaire pour la composante du segment. Nous choisissons un modèle à accéleration constante. Le vecteur d'état a la forme suivante:

$$X_t^{x_n} = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix}$$

Les matrices du modèle pour la coordonné est:

$$\phi = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

ϕ est la matrice de dynamique. Le modèle d'évolution correspondante s'écrira $X_{t+1} = \phi * X_t$.
ϕ se calcule en intégrant la relation fondamentale de la dynamique:

$$x_{t+1} = x_t + \delta t.\dot{x}_t + \frac{1}{2}.(\delta t)^2.\ddot{x} \tag{6.1}$$

$$\dot{x}_{t+1} = \dot{x} + \delta t.\ddot{x}_t$$

$$\ddot{x}_{t+1} = \ddot{x}_t$$

À partir de l'équation 6.1, nous calculons la position éstimée des segments du CyCab dans la prochaine image. Pour pouvoir implémenter cette équation nous avons besoin des trois dernières positions des segments dans les trois images antérieures. La vitesse est définie:

$$\dot{x}_t = x_t - x_{t_1}$$

L'accéleration est définie:

$$\ddot{x}_t = x_t - 2 * x_{t-2} + x_{t-3}$$

Nous trouvons donc à partir de 6.1 $x_{t+1} = x_p$.

6.5 L'application de la contrainte

De la section 6.3 nous calculons l'erreur absolute de toutes les positions des segments du noyau $\mathcal{N}$. Pour chaque point l'erreur doit être plus petite qu'un seuil, c'est seuil est définit comme du 5 %:

$$e_{ss}^i = \frac{|x_i - x_p^i|}{L} < 0.05 \qquad i = 1, \ldots, 6$$

Si au moins, une erreur e_{ss}^i est plus grande que le seuil du 5 %, le noyau est negligé.

Chapitre 7

L'enveloppe du CyCab

Étant donné que la quantité des opérations éffectuées par l'ordinateur dépend fortément de la taille de l'image, est désirable diminuer le champs de recherche à une image plus petite. Nous pouvons faire cela de deux manières:

- En diminuant la taille de l'image complète par un facteur d'escale k.
- Trouver une enveloppe qui contient le CyCab.

La première option a le problème de qu'on perd d'information pour la recherche des invariants projectifs. Cela veut dire que si nous diminuons la taille, les points d'interêt seront plus proches entre eux. Il y a toujours une relation géométrique entre les points, mais les test efectués ont donné des erreurs plus grands que les seuils de la table 3.4. On travaille donc avec l'enveloppe.

7.1 Trouver l'enveloppe

Soit le vecteur:

$$env = \begin{bmatrix} y_1 & y_2 & x_1 & x_2 \end{bmatrix}$$

Pour trouver l'enveloppe nous nous servons des résultats obtenues dans les sections précedentes. L'enveloppe va être un dix pour cent plus grande que l'image du CyCab. les points du vecteur env sont calculés dans la suite:

- y_1 égal à la position du point A du noyau $\mathcal{N}$, plus un 10%.
- y_2 égal à la position du point G estimé avec les valeurs des points du noyau $\mathcal{N}$, plus un 10%.
- x_1 égal á la propietée $\mathcal{C}_i$ de la ligne i de la matrice M_{pr}, plus un 10%. La région étiquetée comme i est la région trouvée dans la section 4.8.2 que correspond au châssis du CyCab.
- x_2 égal á la propietée $\mathcal{D}_i$ de la ligne i de la matrice M_{pr}, plus un 10%. La région étiquetée comme i est la région trouvée dans la section 4.8.2 que correspond au châssis du CyCab.

7.2 Utilisation de l'enveloppe

L'enveloppe sera utilisée pour réduire le champs de recherche du CyCab dedans l'image que rend la caméra.

Nous allons chercher dans la première image la position du CyCab. Avec les données trouvées par les algorithmes de recherche, nous pouvons définir les bords de l'enveloppe. Nous appliquons cet enveloppe dans la prochaine image, et nous ne prendons que la image dedans l'enveloppe.

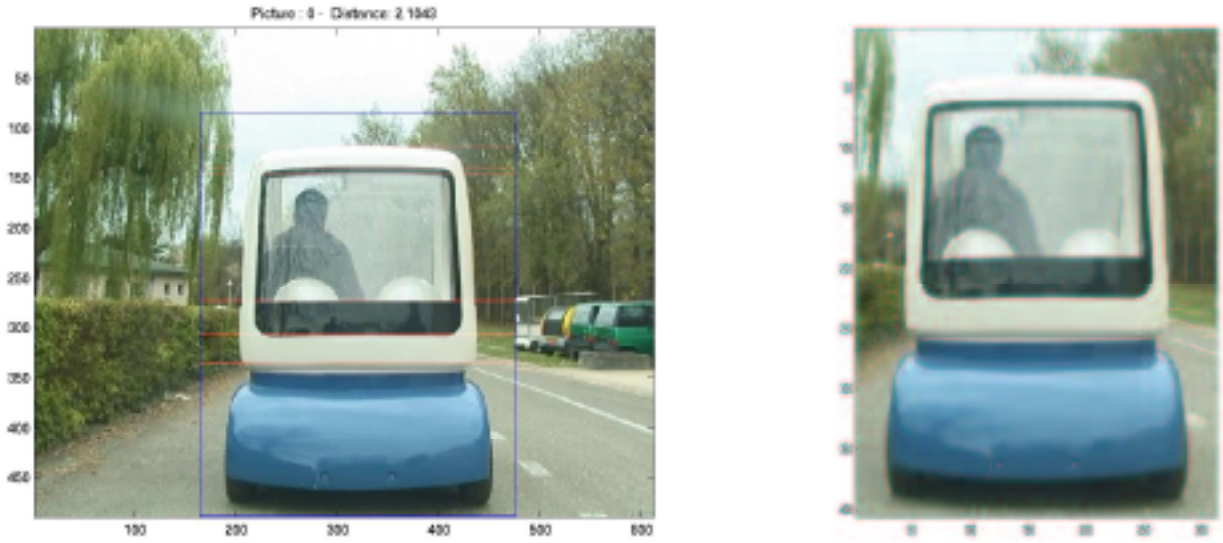


FIG. 7.1 – *À gauche, la première image du CyCab, où nous cherchons la position du véhicule. En bleue nous avons l'enveloppe, et en rouge les positions des points du noyau $\mathcal{N}$. À droite, nous avons la deuxiéme image d'où nous avons appliqué l'enveloppe antérieure pour extraire seulement ce qui est dedans ces limites.*

Nous travaillons avec cette nouvelle image. La même n'a pas perdu d'information pour un changement d'échelle. Nous pouvons trouver de bonnes résultats pour des images du CyCab qui sont à une distance d'environt dix mètres de la caméra.

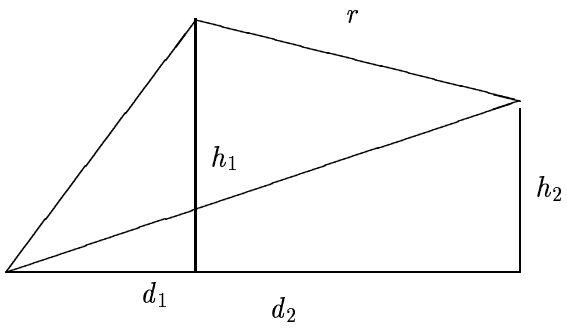
Chapitre 8

Calcul de la distance

Ayant le noyau résultat $\mathcal{N}$, la position des ces points va nous donner la distance où se trouve le CyCab par rapport à la caméra. Le calcule va être fait par triangulation.

Nous prenons deux images et nous mesurons physiquement les distances d_1 et d_2 en metres. Les distances h_1 et h_2 sont les hauteurs de la fenêtre, pris avec la même méthode de la section 3.2.4.

Soient deux points $x_1 = (d_1, h_1)$, $x_2 = (d_2, h_2)$,



Nous cherchons l'équation de la droite r:

$$y = b.x + c$$

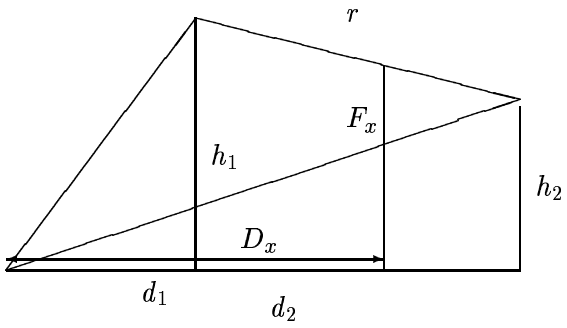
$$\begin{cases} h_1 = b.d_1 + c \\ h_2 = b.d_2 + c \end{cases}$$

$$b = \frac{h_1 - h_2}{d_1 - d_2} \qquad c = \frac{h_2.d_1 - d_2.h_1}{d_1 - d_2}$$

Nous avons donc l'équation de la droite:

$$y = \frac{h_1 - h_2}{d_1 - d_2}.x + \frac{h_2.d_1 - d_2.h_1}{d_1 - d_2} \tag{8.1}$$

La donnée est la hauteur de la fenêtre du CyCab, F_x. Nous calculons à partir de 8.1 la distance à laquelle se trouve le véhicule.



$$F_x = \frac{h_1 - h_2}{d_1 - d_2}.D_x + \frac{h_2.d_1 - d_2.h_1}{d_1 - d_2}$$

b

$$\Rightarrow \qquad D_x = \frac{d_1 - d_2}{h_1 - h_2}(F_x - \frac{h_2.d_1 - d_2.h_1}{d_1 - d_2}) \tag{8.2}$$

Les hauteurs h_1 et h_2 sont celles de la fenêtre entre les points C et D des points d'intérêt (voire figure 3.1). Les h_1 et h_2 vont prendre ses valeurs de quantité de pixels entre ces deux points.

Pour une image du CyCab mesurée à deux mètres de la caméra:

$$h_1 = 128 \qquad et \qquad d_1 = 2$$

Pour une image du CyCab mesurée à huit métres de la caméra:

$$h_2 = 41 \qquad et \qquad d_2 = 8$$

Des valeurs du noyau résultat $\mathcal{N}$ nous pouvons calculer la distance en trouvant V_x comme:

$$V_x = \mathcal{N}(4) - = \mathcal{N}(3)$$

et à partir de l'équation 8.2, nous trouvons la distance objetif de l'algoritme D_x.

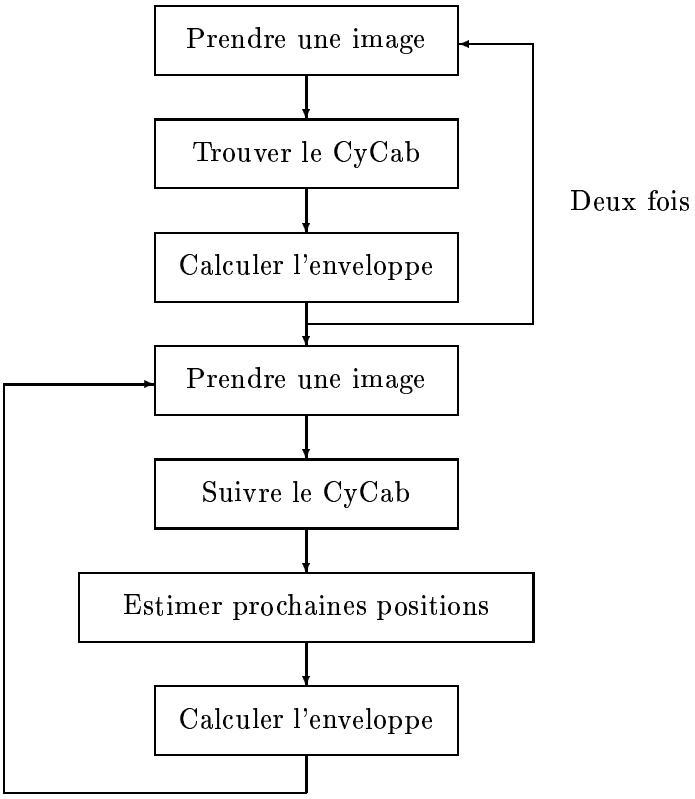
Chapitre 9

Implémentation, problèmes et propositions

9.1 Mise en œvre

La mise en œvre du projet a été fait avec le logiciel MATLAB. Les descriptions des fonctions utilisées sont présentées dans l'annexe A.

Nous utilisons une séquence vidéo à partir une position fixe et d'un CyCab qui s'éloigne à un faible vitesse. La vidéo est de format AVI avec 25 frames par second. De ces frames, nous prenons une cadence de cinq frames par second. Cette séquence vidéo va fournir nos algorithmes et ils vont calculer la distance du véhicule par rapport à la caméra.



Au départ, nous sommes dans l'étape de *RechercheduCyCab*. Nous utilisons une reconnaissance complète du modèle du véhicule. Nous trouvons la position des points horizontaux caractéristiques, les segments verticaux de la fenêtre et la région bleue du châssis. Avec ces données nous calculons l'enveloppe du CyCab. Dans la prochaine image de la séquence, nous répétons le calcul complet pour chercher tous les données du CyCab dans cette enveloppe. Voire figure 9.1.

L'opération de calcul complet des données est réalisée avec trois images pour assurer que nous avons trouvé le CyCab dans plusieurs images. À partir de cela, nous sommes dans l'étape de *SuividuCyCab*. Nous calculons les estimations des prochaines positions des segments avec les trois données antérieures. Nous cherchons maintenant seulement la position des points horizontaux. L'algorithme est beaucoup plus rapide étant donné que nous ne faisons pas un balayage sur toute l'image pour chercher les régions bleues ni cherchons les segments verticaux.

9.2 Résultats

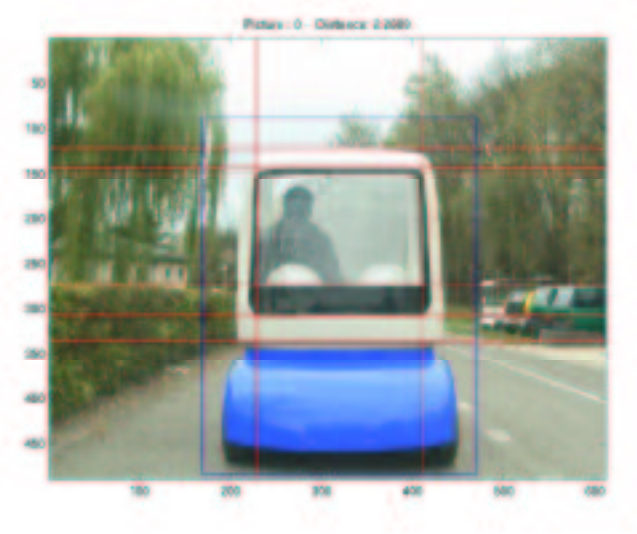


FIG. 9.1 – *Détection complète du CyCab dans la séquence.*

9.3 Problèmes

Les problèmes suivantes ont été détectées au cours des tests de l'algorithme :

- Quand l'algorithme ne trouve pas un ensemble de points avec une erreur plus basse que les seuils définis, il continue la recherche jusqu'à la fin avec tous les points qui ont une valeur différente à zéro dans le vecteur V_h. Cela prend beaucoup de temps de calcul étant donné que l'ajout d'un nouveau point signifie la recherche des relations avec tous les points de l'arbre d'interprétation.

- Le CyCab doit être à une distance pas trop grande, au moment de la détection. Après la détection, il peut s'éloigner jusqu'à une distance d'environ huit mètres.

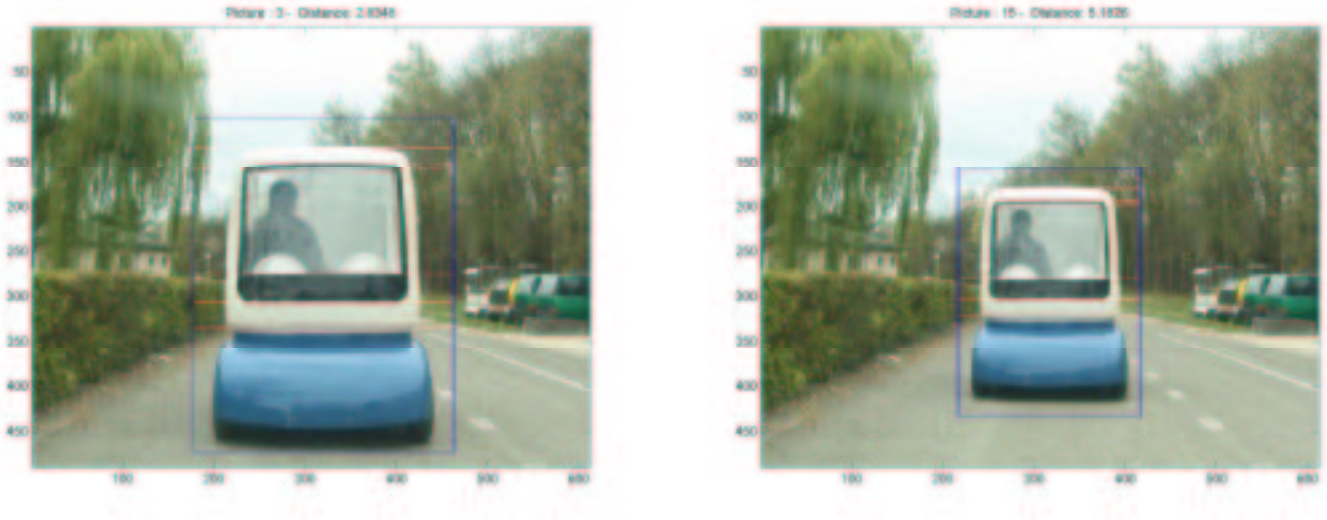


FIG. 9.2 – *Détection du CyCab. Étape de suivi.*

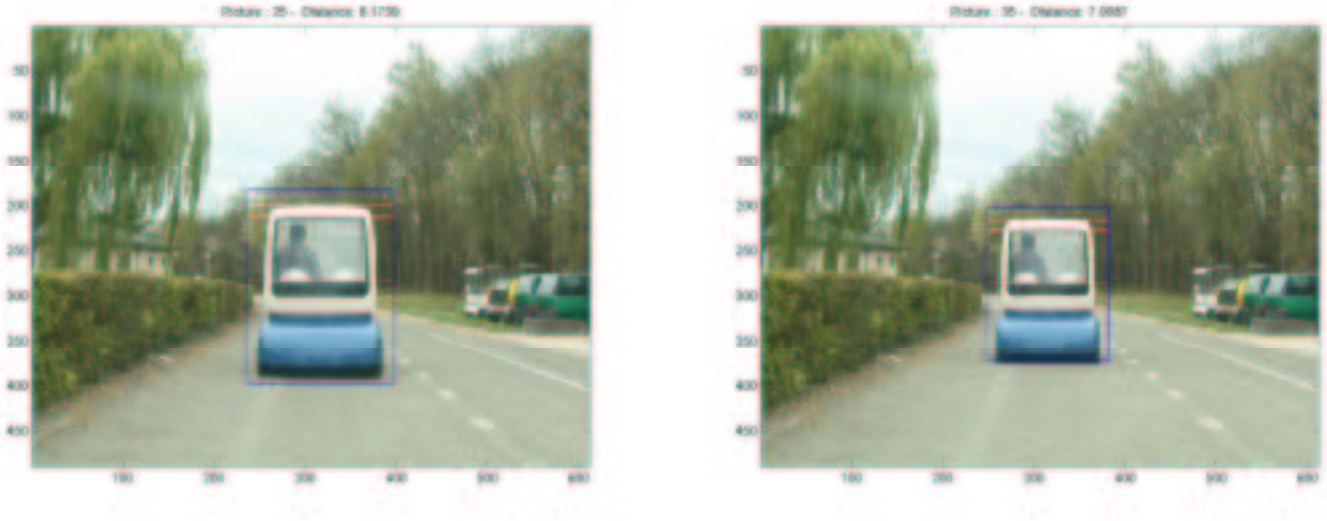


FIG. 9.3 – *Détection du CyCab. Étape de suivi.*

- Dans le cas d'une faible luminosité, la reconnaissance de couleur peut faillir dans zones du châssis, en rendant des régions bleues plus petites, qui peuvent être près de la limite d'acceptation de l'algorithme. Voire figure 9.5.
- Dans le cas d'une perte du véhicule dans une image, jusqu'au présent, l'algorithme n'a pas été capable de le retrouver dans la prochaine image.

9.4 Propositions

Nous pouvons nous apercevoir, de la figure 2.4, que les contours horizontaux du CyCab ont une positions les uns au dessous des autres.

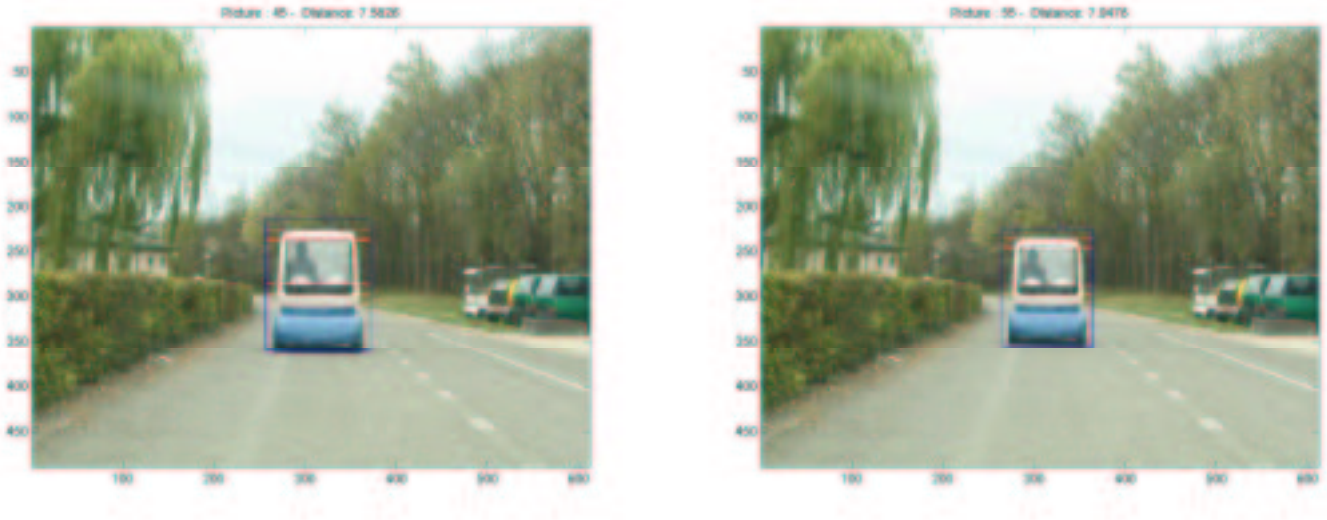


FIG. 9.4 – *Détection du CyCab. Étape de suivi.*

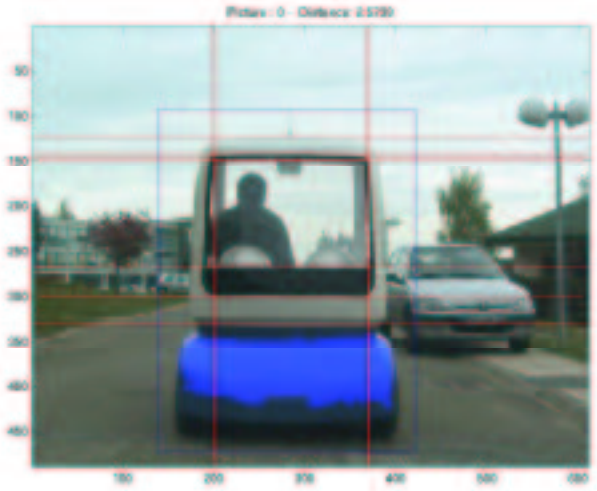
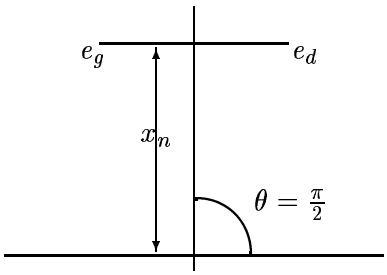


FIG. 9.5 – *CyCab détecté dans une imagen plus obsure et bruitée, les voitures à coté et les lignes des bâtiments*

Cette caractéristique de notre modèle n'est pas negligeable et peut devenir un nouvel ensemble de paramètres pour décrire chaque élément du vecteur V_h.

Une solution pourrait être d'étiqueter la matrice des segments et de trouver ces propriétés. Les nouveaux paramètres pourront être la position des extrêmes de chaque segment. Cela nous donnera une nouvelle matrice des propriétés des segments. Nous pouvons utiliser cette matrice comme une

nouvelle contrainte qui nous permettra d'obtenir des résultats plus robustes.



Où x_n est la hauteur où se trouve le segment, e_g et e_d sont les extrêmes gauche et droit respectivement du segment. θ va être toujours $\frac{\pi}{2}$ étant donné que les segment sont horizontaux.

Chapitre 10

Conclusions

La théorie des invariants projectif nous a servi pour déterminer la position de la face du CyCab, dans une image. Un modèle du véhicule a été défini avec cet objectif. La méthode de recherche s'est centrée sur les segments de droites et les distances de séparation entre eux.

Nous avons constaté l'importance du choix et de l'incorporation des contraintes. Elles sont implémentées pour incrémenter la vitesse de l'algorithme et pour vérifier les résultats obtenus.

Le codage couleur YUV a été utilisé pour la reconnaissance des objets et a donné des informations fondamentales pour l'identification du véhicule.

L'utilisation d'une enveloppe nous a permis d'éviter le bruit des objets autour le CyCab et d'incrémenter la vitesse du calcul.

Nous avons suivi le CyCab à travers d'une séquence d'images jusqu'à une distance d'environ huit mètres. Le program a été fait avec MATLAB, ce que n'est pas vraiment temps réel, mais qui est utile pour les tests.

Dans une autre séquence, le manque de luminance est un problème que l'algorithme n'a pas pu éviter. Il nous a rendu des résultats peu satisfaisants, avec la perte du CyCab à travers les images et des erreurs très grandes. Cette contrainte a rendu difficile la détection des contours et la reconnaissance des couleurs.

Nous avons proposé une amélioration, à travers l'incorporation des nouveaux paramètres aux segments qui pourront servir pour une reconnaissance plus robuste.

Personnellement, ce stage m'a permis d'approfondir mes connaissances dans le champ du traitement d'image. De plus, j'ai joui d' heures de travail à côté des ingénieurs et d'autres stagiaires du projet IMARA. Également, l'INRIA est un laboratoire où nous pouvons trouver tout type d'information dans un centre de documentation très complet et où il y a des spécialistes sur des thèmes différents qui nous pouvons consulter.

J'espère que mon travail sera un bon point de départ pour une prochaine recherche destinée à implémenter un système de vision pour le "Platooning" ou d'intégrer ce capteur comme complément des informations d'autres capteurs.

Annexe A

Fonctions MATLAB

adjcontrast

Fonction pour faire l'adjustement de contrast. Author: Peter Kovesi Department of Computer Science & Software Engineering The University of Western Australia pk@cs.uwa.edu.au www.cs.uwa.edu.au/ pk July 2001

algorithmeDistances

Cette fonction cherche des relation des distances en prenant de à trois points.

algorithmeDistancesVertical

Fonction qui travaille avec histogrammes de données horizontaux. Elles peuvent être de regions ou de contours.

appliquerSobel

Application des masques 5x5 de Sobel.

assignationSegments

Fonction de conversion entre la matrice des distances et la matrice de position des segments. Elle prendre en compte que les résultats de les points B et C peuvent être très proches. La fonction deside entre ces points par la direction du gradient.

calculeCentroid

Fonction qui calcule le centroide d'une region.

calculerDistanceCyCab

Calcul de la distance par triangulation.

chercherInvariants

Fonction qui gère la recherche des invariants projectifs du modele du CyCab à partir des segments horizontaux.

chercherRegion

Fonction qui cherche la region bleue qui accomplit les relations geometriques de l'ensemble des points trouvé.

chercherRelationACinq

Fonction qui cherche des rélations entre un nouvelle point est les noyaux à quatre points de l'arbre d'interpretation.

chercherRelationAQuatre

Fonction qui cherche des rélations entre un nouvelle point est les noyaux à trois points de l'arbre d'interpretation.

chercherRelationASix

Fonction qui cherche des rélations entre un nouvelle point est les noyaux à cinq points de l'arbre d'interpretation.

comparerGrpSegAvcEst

Fonction qui fait la comparaison des points d'un noyau avec l'estimation.

comparerSegReg

Fonction qui fait la recherche de la position horizontaux en croisant les resultats des contours est des regions noires.

completerVecPosSeg

Fonction qui complète les points manquantes du noyau résultant $\mathcal{N}$.

comprobationCouleur

Fonction qui va faire une comprobation avec les regions de couleur trouves et la position des segments. Nous allons calculer une superficie par rapport aux segments trouves, avec des invariants du CyCab et nous allons faire la comparaison de cette superficie avec la region du couleur.

creerMatriceDistance

Fonction qui calcule la matrice des distances à partir d'un vecteur.

designCyCab

Fonction qui sert pour montrer les résultats à travers un moyen graphique.

estimationSegment

Fonction qui estime la position d'un point du noyau, qui n'a pas été trouvé par l'algorithme en utilisant les données des autres points du noyau. Complement de la fonction *completerVecPosSeg*.

estimerEvolutionsegment

Fonction qui prend les resultats des positions des segments des images anterieures et fait une estimation de la position des segments dans la prochaine image.

etiquetageRegions

Fonction qui fait l'etiquetage des régions bleues.

isolerRegions

Fonction qui convert l'image de l'estpace couleur RGB à l'espace YUV. Elle fait la recherche des regions bleues, noires et claires. Ella applique aussi le traitement morphologique.

rgb2yuv

Fonction qui fait la convertion de l'espace RGB à l'espace couleur YUV.

selectionnerResAvcEst

Fonction qui fait la recherche du noyau résultant $\mathcal{N}$, à partir des noyaux de l'arbre d'interpretation et en faisant la comparaison des ses résultats avec l'estimation.

selectionerResultatAvecCouleur

Fonction qui fait la recherche du noyau résultant $\mathcal{N}$, à partir des noyaux de l'arbre d'interpretation et en faisant la comparaison de la quantitée de couleur correspondant pour chaque valeur du noyau.

suivreCyCab

Fonction qui, une fois définis l'estimation de la position des segements dans la prochaine image, fait la recherche du CyCab dans le noyau, en implementant l'estimation comme contrainte.

testSerie

Batch qui organize le test, en prenant les images des fichiers de format jpeg.

trouverCyCab

Fontion qui fait une reconnaissance complète du modèle du véhicule. Elle trouve la position des points horizontaux caractéristiques, les segments verticaux de la fenêtre et la région bleue du châssis.

trouverEnveloppeCyCab

Fonction qui cherche la position de l'enveloppe à partir des résultats du noyau résultant $\mathcal{N}$ et la région bleue du châssis.

trouverMaxLocaux

Fonction qui cherche les maximums locaux dans un vecteur.

trouverRegionBleueYUV

Fonction qui utilise la méthode de la section 4.3 pour trouver les régions bleues.

trouverRegionClaire

Fonction qui utilise la méthode de la section 4.3 pour trouver les régions claires.

trouverRegionNoire

Fonction qui utilise la méthode de la section 4.3 pour trouver les régions noires.

trouverSegments

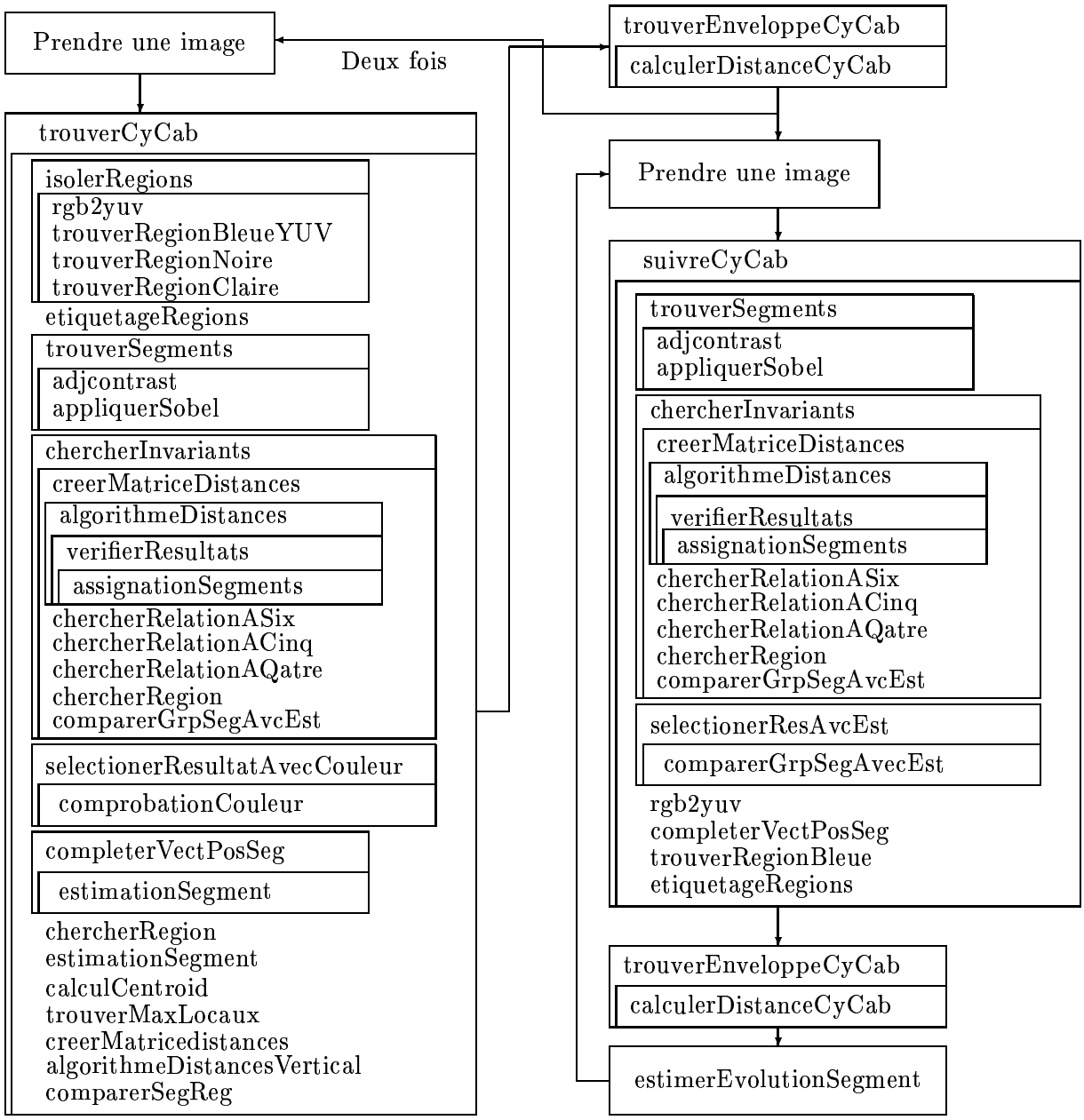
Fonction qui prend l'image RGB et fait la convertion au format YUV, applique les masque de Sobel et trouve par seuillage les matrices des segments verticaux et horizontaux.

verifierResultats

Fonction qui verifie la direction du gradient pour l'ensemble à trois points trouvé avec la fonction *algorithmeDistances*.

Diagramme des fonctions

Le diagramme suivant montre les rélations entre les fonctions.



Bibliographie

[1] P. Daviet, S. Abdou, M. Parent, *Platooning for Vehicles and Automatic Parking by Scheduling Robotic Actions*, 1996, RR INRIA.

[2] C. Achard, *Cours de Traitement d'Images.*

[3] V. F. Leavers, *Shape Detection in Computer Vision Using the Hough Transform*, 1992, Springer-Verlag.

[4] H. Haubecker, P. Geibler, *Handbook of Computer Vision and Applications - Volume 2 - Signal Processing and Pattern Recognition.*

[5] W. Eric L.Grimson, *Object Recognition by Computer, The role of Geometric Constraints*, 1990, MIT Press.

[6] C. Rohtwell, *Object recognition through invariant indexing*, Oxford science publications.

[7] J. Bruce, T. Balch, M. Veloso, *Fast and Inexpensive Color Image Segmentation for Interactive Robots*, School of Computer Science, Carnegie Mellon University.

[8] L. Lacassagne, *Motion detection and multitarget trackin in real time*, Thesis.

[9] B. Giai-Checha, R. Deriche, O. Faugeras, *Suivi de segments dans une séquence d'images monoculaire*, INRIA 1993.