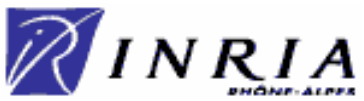

Ecole Nationale Supérieure de
Physique de Strasbourg

Promotion 2001
Mitonneau
Hugues

Guidage par vision de véhicules automatiques

Application sur le Cycab

INRIA
ZIRST – 655 avenue de l'Europe
38334 Saint Ismier Cedex
Tél.: 04.76.61.52.00
Fax: 04.76.61.52.52

Hervé Mathieu
herve.mathieu@inrialpes.fr

Du 5 mars 2001 au 27 juillet 2001

Table des matières

Illustrations

Remerciements :

Je remercie Hervé Mathieu de m'avoir permis d'effectuer mon stage à l'INRIA Rhône-Alpes, ainsi que le service des Moyens Robotique Vision et Réalité Virtuelle, de m'avoir accueilli au sein de leur équipe.

Je remercie le service des moyens informatiques pour avoir mis à ma disposition le matériel utile lors de mon stage.

Introduction

Le stage que j'ai effectué à l'INRIA Rhône Alpes s'est déroulé en deux parties.

La première partie, qui a occupé la plus grosse part du stage, avait pour but de valider les possibilités offertes par une caméra d'un nouveau type : la caméra DVT600. En effet, cette caméra ne se contente pas d'acquérir des images et de les transmettre à une carte d'acquisition. Cette caméra comporte un processeur qui effectue les calculs sur l'image et ne transmet que les résultats via une liaison éthernet.

Afin de valider cette caméra, ses possibilités ont été utilisées pour permettre le suivi de deux véhicules automatiques : les Cycab. Ce suivi était déjà possible avant, mais la détection du véhicule leader était réalisée par un autre moyen : Un laser ou une caméra linéaire. Le but est donc de remplacer les modules existants de détection du véhicule leader par un nouveau module fonctionnant grâce à cette nouvelle caméra.

La seconde partie consistait à étudier les possibilités d'un autre système de guidage par vision pour le Cycab. Ce système consiste à faire suivre au Cycab une trajectoire en comparant les images prises lors de ce trajet avec des images d'une base de données préalablement acquises lors d'un trajet identique.

I. Présentation de l'INRIA

L'INRIA, Institut National de Recherche en Informatique et en Automatique, est un établissement public à caractère scientifique et technologique, placé sous la double tutelle du Ministère de la Recherche et du Ministère de l'Economie, des Finances et de l'Industrie.

L'INRIA Rhône-Alpes (voir Figure 1) mène ses activités en étroite collaboration avec les laboratoires de recherche publics et privés, nationaux et internationaux, et elle entretient des liens privilégiés avec l'institut d'Informatique et Mathématiques Appliquées de Grenoble (IMAG).

L'INRIA Rhône-Alpes entreprend continuellement des actions contribuant à la diffusion des connaissances et du savoir-faire.

- Formation par la recherche : En synergie avec les établissements d'enseignement supérieur locaux, l'INRIA accueille plus de 130 doctorants, élèves-ingénieurs et stagiaires. Ses chercheurs participent à l'enseignement supérieur au sein des universités et grandes écoles de la région Rhône-Alpes (Institut National Polytechnique de Grenoble, université Joseph Fourier, université Pierre Mendès-France, université de Savoie, Ecole Nationale Supérieure de Lyon).
- Information et culture scientifique et technique : Dans le cadre de sa mission de diffusion des connaissances et du savoir-faire, l'INRIA publie et diffuse un grand nombre de résultats émanant des travaux de ses scientifiques. Ces travaux sont disponibles sous forme de rapports - version papier ou électronique - ou sous forme de logiciels accessibles sur le Web.
- Partenariats internationaux : L'INRIA entretient des relations internationales bilatérales ou multilatérales dans le cadre de programmes d'échange, et de recherche en partenariat avec des universités européennes, nord et sud américaines, israeliennes, asiatiques, etc. Les chercheurs de l'INRIA effectuent des séjours de courte ou longue durée dans des laboratoires étrangers et réciproquement, l'INRIA accueille des chercheurs étrangers dans le cadre de bourses d'échanges ou d'invitation de spécialistes. A titre d'exemple, en 1998 ce sont plus de 1400 chercheurs étrangers qui ont participé aux activités de l'INRIA Rhône-Alpes.

Acteur de la recherche au sein d'une région qui a retenu le pôle du numérique comme l'un des deux grands axes de développement technologique, aux côtés des biotechnologies, l'INRIA Rhône-Alpes mène une politique active de transfert vers le monde économique et social et participe activement à la création d'entreprises innovantes. Selon le degré de maturité des projets et les partenaires (industriels, collectivités, etc.), les activités de développement peuvent se décliner sous différentes formes :

- Partenariats
- Actions de développement
- Déploiement d'expérimentations régionales
- Création d'entreprises

Figure 1 : INRIA Rhône-Alpes

Ressources budgétaires de l'INRIA:

- dotation de l'état : 442 MF HT
- ressources propres : 174 MF HT

Ressources humaines :

- 724 titulaires INRIA
- 256 post-Doctorants et Contractuels
- 550 doctorants
- 230 chercheurs et enseignants d'autres organismes
- 430 conseillers, collaborateurs divers et invités

Indicateurs :

- contrats de recettes actifs : plus de 600
- contrats de recettes signés dans l'année : plus de 200
- un peu moins d'une cinquantaine de sociétés sont issues de l'INRIA, depuis Ilog, aujourd'hui cotée au Nasdacq, jusqu'aux toutes dernières, 5 en 1998, 6 en 1999, 11 en 2000.
- 7 brevets initiaux déposés en 1999 : 1 est en pleine propriété INRIA, les autres sont en copropriétés, 5 avec des industriels et 1 avec une université.

II. Le Cycab

Figure 2 : Le Cycab

Dans le cadre de la route automatisée, l'INRIA a imaginé un système de transport original de véhicules en libre-service pour la ville de demain. Ce système de transport public est basé sur une flotte de petits véhicules électriques spécifiquement conçus pour les zones où la circulation automobile doit être fortement restreinte. Pour tester et illustrer ce système, deux prototypes, nommés Cycab (voir Figure 2), ont été réalisés. Les caractéristiques générales du véhicule sont les suivantes :

- Longueur hors tout : 1,90m
- Largeur hors tout : 1,20 m
- Poids total avec batteries : 300 kg
- Motorisation : 4 moteurs électriques de 1 kW
- 4 roues motrices et directrices
- Vitesse théorique maximale : 20 km/h
- Autonomie : 2 heures d'utilisation continue
- Capacité d'accueil : 2 personnes avec bagages

Ces véhicules possèdent un ordinateur embarqué fonctionnant avec un système temps réel : VxWorks. Ainsi ils peuvent être conduit manuellement grâce à un joystick ou de manière automatique grâce aux différents capteurs qu’ils possèdent.

III. Suivi automatique d'un Cycab

III. 1. La caméra DVT600

III. 1. 1. Présentation

La caméra utilisée est livrée par DVT (voir Figure 3). Cette caméra est en fait tout un système comportant une optique noir et blanc, un processeur, plusieurs entrées/sorties analogiques, une liaison éthernet et une liaison série. C'est donc la caméra elle-même qui effectue les calculs, puis qui envoie les résultats sur le réseau.

Figure 3 : Caméra DVT

Les programmes pour la caméra sont développés à l'aide d'un ordinateur sous Windows. Ceux-ci sont sauvegardés dans la mémoire Flash de la caméra. Il est ensuite possible de récupérer les résultats à partir de n'importe quel ordinateur, en se connectant à la caméra par une liaison TCP/IP. Chacun de ces programmes est appelé un 'produit'. Ils sont composés de 'senseurs' qui sont effectués les uns après les autres lors d'une inspection. Il existe plusieurs modes pour lancer une inspection :

- Lancer une inspection à intervalle régulier.
- Lancer une inspection dès que la précédente est finie.
- Lancer une inspection lorsque le signal 'trigger' passe à 1. Dans ce mode, il est également possible de lancer l'inspection via la liaison éthernet.

Ces senseurs sont prédéfinis. Ils réalisent des opérations plus ou moins complexes sur les images (voir Tableau 1). Les résultats de chaque senseur, comme la position d'une transition, peuvent être utilisés par les senseurs suivants. Il est également possible d'écrire des scripts dans un langage proche du C++.

Translation	Ces senseurs ont pour but de rechercher un bord dans une image. Ce bord est recherché soit sur une ligne droite, soit dans une zone. Il est ensuite possible de récupérer la position de ce bord, ainsi que sa direction dans le cas d'une recherche dans une zone.
Rotation	Comme les senseurs de Translation, ces senseurs recherchent un bord. La différence est que ces outils calculent l'angle de ce bord et non pas sa position.
Intensité	Ces senseurs calculent le pourcentage de pixels supérieur à un seuil dans une région. Cette région peut être soit une ligne, soit une zone. La moyenne et la médiane de l'intensité des pixels dans la région est également calculée.

Compteur de Bords	Ces senseurs comptent le nombre de bords se trouvant le long d'une ligne. Un bord est une transition d'un pixel lumineux (au-dessus du seuil) à un pixel sombre (au-dessous du seuil) ou inversement.
Compteur d'Eléments	Ces senseurs comptent le nombre d'éléments le long d'une ligne. Un élément est un ensemble consécutif de pixels, clair ou sombre selon le choix de l'utilisateur, le long de cette ligne.
Mesure Précise	Ces senseurs recherchent un bord ou la distance entre deux bords sur une ligne droite ou sur une zone. La différence avec les senseurs de Translation est que les résultats sont plus précis. En effet, les valeurs sont calculées avec une précision supérieure a celle du pixel.
Mesure de Cercle	Ces senseurs calcul le centre et le rayon d'un cercle. Il est possible de faire ces calculs même si le cercle n'est pas entier. Comme précédemment, les mesures sont effectuées avec une précision supérieure au pixel.
Outils Mathématiques	Ces senseurs font des calculs uniquement avec les résultats des autres senseurs, et non pas à partir de l'image elle-même. Ces senseurs sont : Calcul de distance, Calcul de différence d'angle, Calcul d'intersection, Calcul de point milieu, Calcul de droite passant par deux points, Calcul de perpendiculaire, Calcul de facteur d'échelle, Calcul de changement de coordonnées.
Script	Ces senseurs permettent d'écrire des scripts pour la caméra. Ces scripts peuvent récupérer toutes les valeurs calculées par les autres senseurs. De plus, les valeurs calculées par ces senseurs peuvent, elles aussi, être utilisées par d'autres senseurs. Ils peuvent également : lire et modifier des paramètres d'autres produits comme le temps d'exposition, lire et modifier les registres de la caméra, lire les entrées et modifier les sorties analogiques.
Lecteurs code-barres	Ces senseurs peuvent lire différents type de code-barres : non seulement les code-barres les plus classiques à une dimension, mais aussi des code-barres à deux dimensions.
Reconnaissance d'Ecriture	Ces senseurs peuvent reconnaître et lire du texte après avoir appris les différents caractères.
Outils Tache	Ces senseurs effectuent un seuil sur une zone de l'image, puis, une dilatation, une érosion, une ouverture ou une fermeture. Il est alors possible de compter les taches et de les sélectionner selon divers critères comme, par exemple, la surface, le rayon moyen, ou le périmètre.
Correspondance de Modèle	Les senseurs Correspondance de Modèle sont conçus pour mémoriser automatiquement une portion d'une image et pour rechercher cette image au cours d'inspections successives. De plus, le senseur Correspondance de Modèle calcule les erreurs entre l'image acquise et la portion mémorisée.
Recherche d'Objets	Les senseurs Recherche d'Objets sont des outils de recherche de positionnement. Ils recherchent la position d'objets en se basant sur une détection de contour.

Tableau 1 : Les différents senseurs existants

III. 1. 2. Installation

La caméra était livrée avec un CD d'installation de FrameWork. C'est grâce à ce logiciel, qui fonctionne sous Windows, que l'on peut développer les programmes de la caméra. Pour que ces programmes fonctionnent sur la caméra, la mémoire flash de celle-ci doit contenir un système d'exploitation : Firmeware. Ces deux composants logiciels doivent être compatibles. La caméra était livrée avec la version 2.1 de FrameWork, ainsi que la version 2.1 de Firmeware.

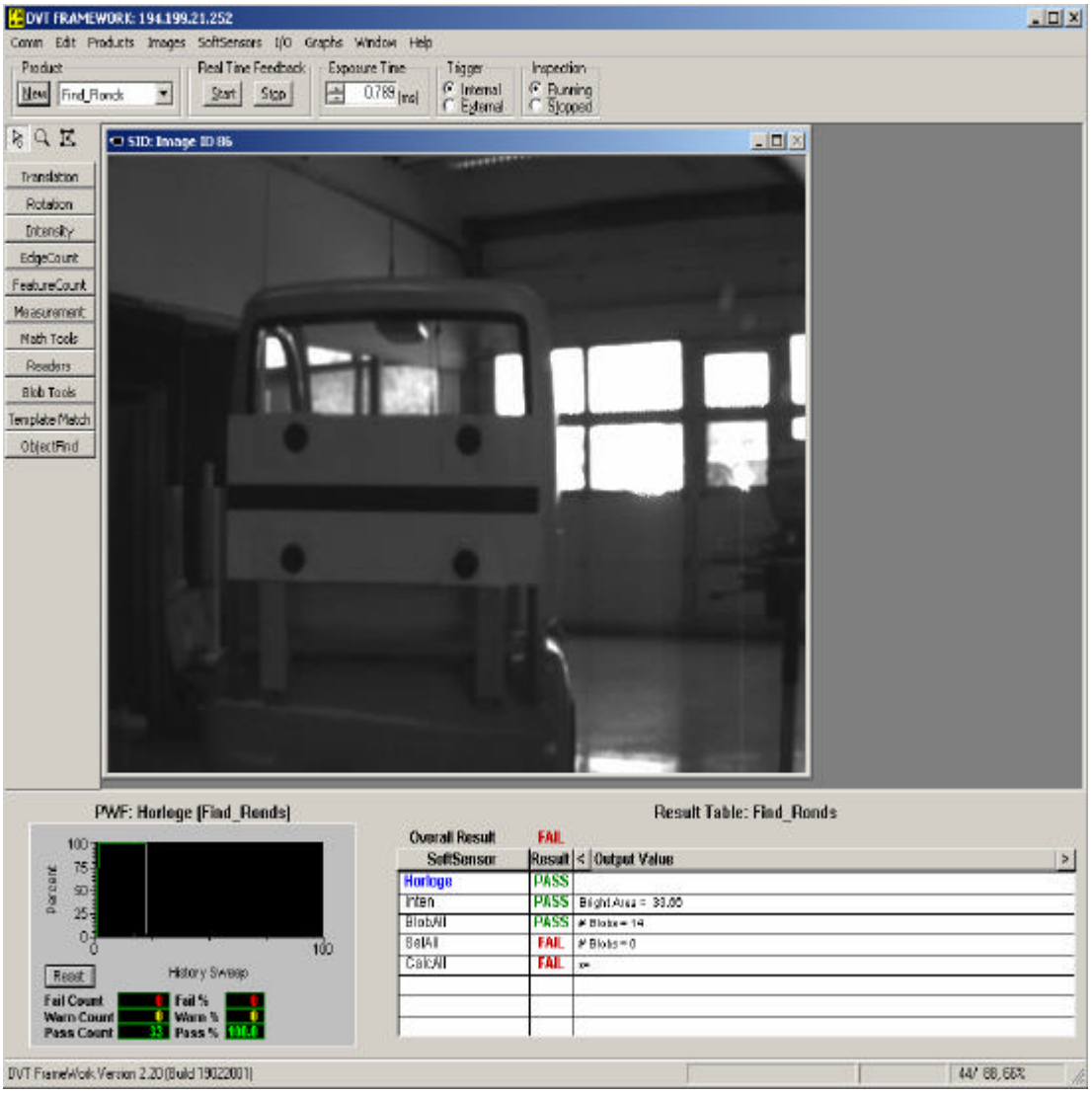

Figure 4 : FrameWork

Afin de pouvoir utiliser le maximum de fonctionnalités, j'ai téléchargé la dernière version de FrameWork : la version 2.2 (voir Figure 4). Pour faire fonctionner le système, il a donc fallu formater la mémoire flash de la caméra avec la version 2.2 de Firmware. Pour cela, il faut connecter la caméra au port série de l'ordinateur en passant par l'adaptateur RS-422/RS-232 (voir Figure 4). Cette première étape a posé des problèmes. Mais après de nombreux essais, il s'est avéré que c'était l'adaptateur (voir Figure 5) qui était en faute. En effet, celui-ci peut être ouvert afin de le configurer. C'est donc en l'ouvrant, que l'on s'est aperçu que cet adaptateur était mal branché. Après la réparation de ce problème, il a été possible d'installer la version 2.2 de Firmeware dans la caméra.

Figure 5 : Adaptateur RS-422/RS-232

Une fois cette opération effectuée, il faut configurer les paramètres TCP/IP de la caméra. Pour cela, il faut, comme précédemment, brancher la caméra sur le port série de l'ordinateur. Il est alors possible de donner à la caméra son adresse IP, un masque de sous réseau et un nom. A partir de ce moment, il n'est plus utile de brancher la caméra sur le port série. Elle peut désormais communiquer avec les ordinateurs branchés sur le réseau par une liaison TCP/IP.

III. 2. Le système développé

III. 2. 1. Calcul de la position du Cycab Leader

L'objectif est de localiser le Cycab leader grâce à la caméra. Des systèmes réalisant ce même travail existaient déjà. Ces systèmes localisant le Cycab leader et renvoyaient la distance entre les deux véhicules, la déviation, ainsi que les angles α et β (voir Figure 6). Ce sont donc ces variables qui sont calculés par la caméra. La caméra est placée à l'avant du Cycab suiveur, et afin de faciliter la localisation du Cycab leader, des motifs (voir Figure 7) ont été dessinés à l'arrière de celui-ci.

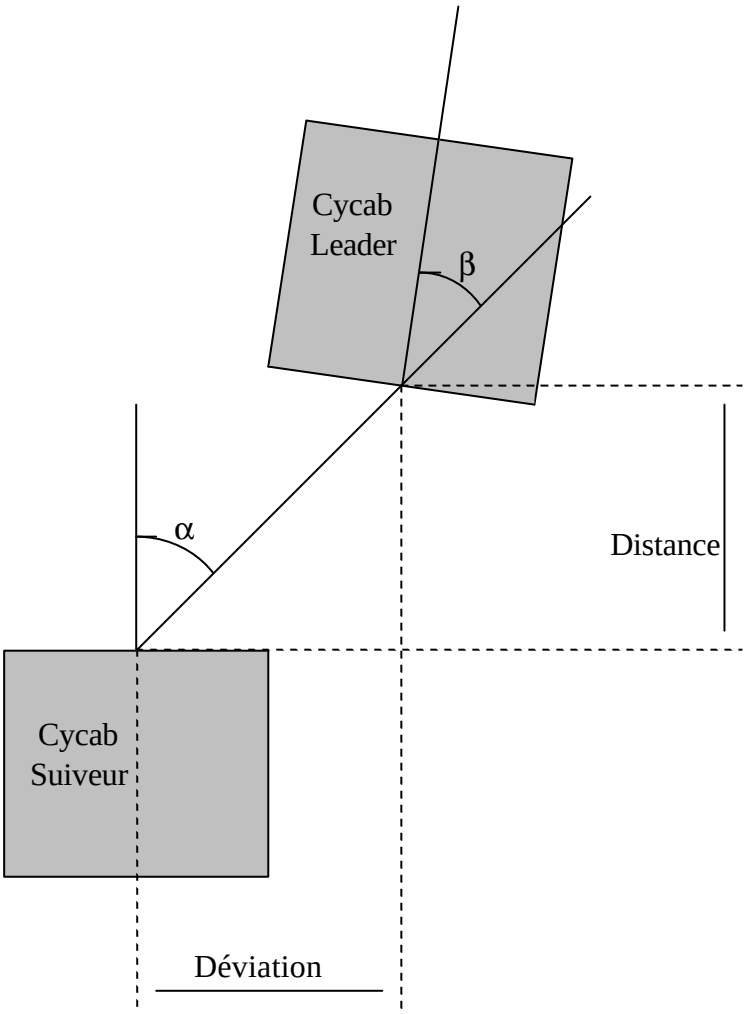

Figure 6 : Paramètres à calculer

Figure 7 : Motifs derrière la voiture

Le calcul effectif de la position du Cycab leader par rapport au Cycab suiveur commence après avoir déterminé les positions des quatre centres des ronds dans l'image.

On calcul alors le nombre de pixels entre les deux ronds de droite. On connaît la distance réelle entre ces deux ronds. En notant A le rond du haut et B le rond du bas, et en utilisant la méthode décrite dans le chapitre suivant, on peut alors calculer la distance séparant la caméra du côté droit du Cycab leader. De même, on calcule la distance entre la caméra et le côté gauche du Cycab leader.

Sur les notations de la Figure 8, O représente le centre de l'avant du Cycab suiveur, c'est à dire là ou est placée la caméra, C représente le centre de l'arière du Cycab leader, G et D représentent le coté gauche et droit du Cycab leader. Ce sont donc les distances OG et OD que l'on vient de calculer. De plus, connaissant la position des ronds dans l' image, on peut déterminer de la même manière les angles α_G et α_D. On connaît alors les positions des points G et D :

$$X_G = OG \cdot \sin(a_G)$$
$$Y_G = OG \cdot \cos(a_G)$$

$$X_D = OD \cdot \sin(a_D)$$
$$Y_D = OD \cdot \cos(a_D)$$

Le point C étant situé au centre de GD, sa position est très simple à retrouver. On obtient de cette façon la valeur de la *Distance* recherchée et la *Déviation* :

$$Déviation = \frac{X_G + X_D}{2}$$

$$Distance = \frac{Y_G + Y_D}{2}$$

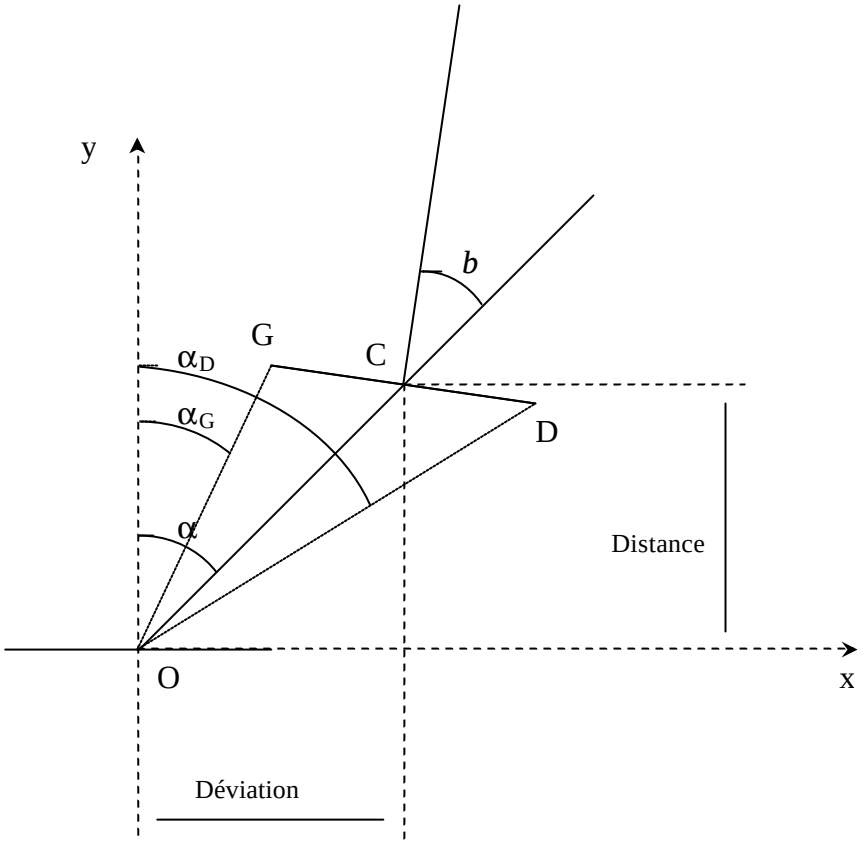

Figure 8 : Notations pour les calculs

La détermination de α es alors possible :

$$a = \tan^{-1}\left(\frac{Déviation}{Distance}\right)$$

Sur le schéma (voir Figure 8), on voit que l'angle a est égal à la somme de l'angle b et de la direction du Cycab leader. On peut alors déterminer b de la façon suivante :

$$b = a + \tan^{-1}\left(\frac{Y_D - Y_G}{X_D - X_G}\right)$$

III. 2. 2. Calcul de la distance d'un objet dans le cas général

Si deux objets A et B sont vus par la caméra avec une différence d'angle de a radians (voir Figure 9), alors la distance séparant les deux points images sur la matrice CCD peut être calculée comme ceci :

$$a = \arctan\left(\frac{d}{f}\right) \approx \frac{d}{f}$$

L'approximation se justifie car les angles en jeux restent relativement faibles. Or la matrice CCD mesure 4,8mm sur 3,6mm. Elle contient 640 pixel sur 480. Chaque pixel est donc espacé de 7,5µm de son voisin. La distance d est donc égale à :

$$d = NombrePixels \cdot 7{,}5mm$$

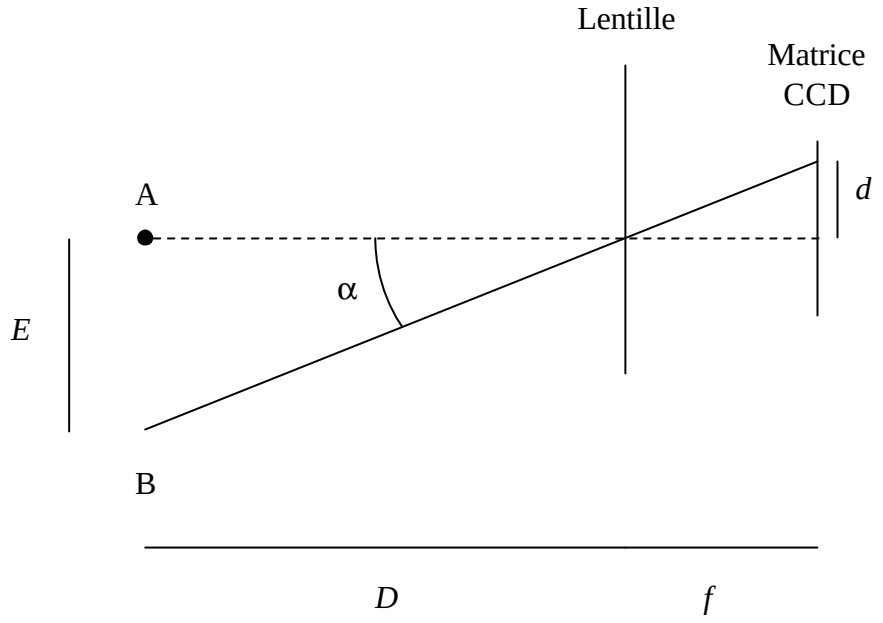

Figure 9 : Optique de la caméra

La distance focale *f* étant égale à 8mm, nous pouvons en déduire la différence d'angle de vue en fonction de la différence de pixel sur l'image :

$$a = \frac{d}{f} = \frac{NombrePixels \cdot 7{,}5mm}{8mm}$$

Un simple coefficient de proportionnalité nous permet donc de calculer cet angle α :

$$a = NombrePixels \cdot Coef$$

$$Coef = \frac{7{,}5mm}{8mm} = 9{,}375 \cdot 10^{-4}$$

De plus, si on connaît l'espacement *E* entre les deux objets A et B, il est possible de connaître la distance *D* qui les sépare de la caméra.

$$\frac{D}{E} = \frac{f}{d} \Rightarrow D = E \cdot \frac{f}{d} = \frac{E}{NombrePixels \cdot Coef}$$

III. 2. 3. Réglage du temps d'exposition

Afin de pouvoir acquérir des images exploitables dans toutes les conditions d'illumination, un réglage automatique du temps d'exposition a été programmé. Ce réglage est effectué dans un produit qui est lancé lors de l'allumage de la caméra. En effet, si ce temps est trop long, le capteur sera complètement ébloui, et si à l'inverse, il est trop court, l'image sera trop sombre pour pouvoir être exploitée correctement. De plus, les conditions d'illumination peuvent varier énormément entre l'extérieur et l'intérieur par exemple. Il n'est donc pas possible de trouver un temps d'exposition moyen qui soit satisfaisant dans toutes les conditions.

Le réglage se fait uniquement avec les informations issues de l'image. Le produit calcule le nombre de pixels surexposés, et le nombre de pixels sous-exposés. L'exposition est alors multipliée ou divisée par un facteur 1,5. Une nouvelle image est alors acquise, et ainsi de suite jusqu'à atteindre une exposition correcte. Le facteur 1,5 permet d'assurer une certaine rapidité dans le réglage tout étant sûr de pouvoir trouver un temps d'exposition satisfaisant.

De plus, pour augmenter encore la vitesse, tous les pixels de l'image ne sont pas pris en compte. Seules une ligne sur dix et une colonne sur dix sont prises en compte. Un centième de l'image est donc observé, ce qui expérimentalement s'est avéré suffisant pour savoir si l'image est surexposée ou sous-exposée.

III. 2. 4. Produit d'initialisation

Il n'a pas été possible de concevoir un produit capable de localiser, sans connaissance a priori sur sa position, le Cycab leader en moins de 100 millisecondes. Deux produits ont donc été développés. Le premier produit, lent, localise le Cycab dans une grande généralité de cas. Le second, plus rapide, traque les motifs dessinés à l'arrière du Cycab leader (voir Figure 7). Pour cette première inspection initiale, le produit procède en cinq étapes :

- Evaluation de la valeur moyenne des pixels
- Seuillage de l'image
- Ouverture sur les éléments connexes noirs
- Filtrage des éléments connexes
- Calcul de la position du Cycab

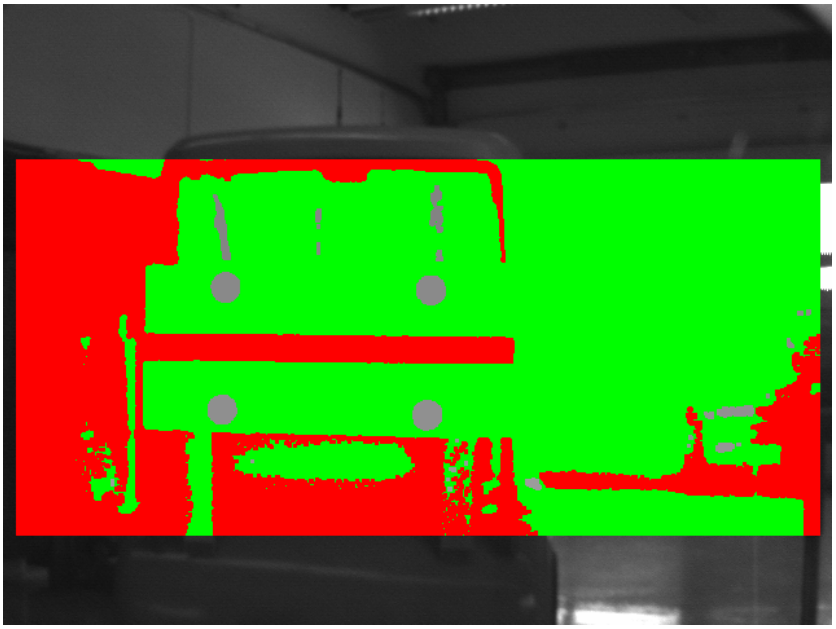

Figure 10 : Seuillage et ouverture

La première étape consistant à évaluer la valeur moyenne des pixels est indispensable pour l'étape suivante. En effet, si le seuil est mal choisi, les motifs ne pourront pas être distingués du reste de l'image. Ce seuil est donc calculé en fonction de la valeur moyenne et non pas en fonction de la valeur maximale et la valeur minimale. En effet, ces valeurs peuvent varier très facilement, et n'ont pas de réelle signification.

La deuxième et troisième étape, le seuillage et l'ouverture, sont réalisée par un seul module fourni dans le logiciel de développement FrameWork. Afin de gagner du temps, ces étapes ne se font pas sur toute l'image, mais seulement sur une zone d'intérêt (voir Figure 10). Cette zone est tout de même suffisamment importante afin de pouvoir détecter la voiture dans une généralité de cas la plus grande possible.

Sur la figure ci-dessus, les couleurs représentent les éléments trouvés :

- En vert : Ce qui est plus clair que le seuil
- En Rouge : Les éléments connexes qui touchent le bord de la zone, donc éliminés d'office
- En gris : Les éléments connexes à l'intérieur de la zone

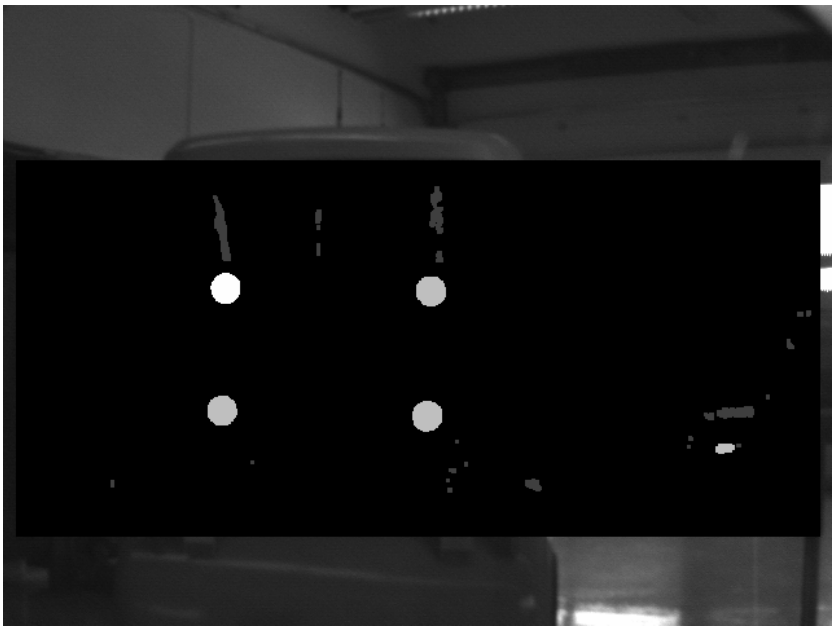

Figure 11 : Sélection des éléments connexes

Un autre module permet de faire un premier filtrage sur les éléments connexes. Ce module sélectionne uniquement dont la surface est supérieure à une certaine limite, fixée ici a 100 pixels, et dont le rapport $4 * p * Surface / Périmètre^2$ est superieur à 0,7 (voir Figure 11). En effet ce rapport est égal à 1 pour un rond et est inférieur pour toutes les autres surfaces. Cette valeur a été choisie afin de détecter à coup sur les motifs. En effet ceux-ci n'apparaissent jamais parfaitement ronds dans l'image pour plusieurs raisons. Premièrement, nous avons à faire à une image numérique, qui a subi un seuillage puis une ouverture, les éléments s'en trouvent forcement déformés. De plus, les deux Cycab ne sont pas forcément alignés. Les motifs apparaissent alors ovales.

Après ce premier filtre, plusieurs éléments connexes indésirables peuvent encore être sélectionnés. Un autre filtre a donc été mis en place afin de ne garder que les éléments correspondant aux motifs dessinés derrière le Cycab leader. Parmi les éléments connexes restants, on ne sélectionne que ceux pour lesquels il existe au moins un autre élément connexe sur la même ligne et un autre sur la même colonne. Il ne reste alors généralement plus que les quatre ronds que l'on voulait détecter. Ce second filtre est réalisé grâce à un script mathématique. Une fois le filtre effectué, si seulement quatre éléments sont sélectionnés, ce même script mathématique peut commencer le calcul de la position de la voiture. Les positions des ronds dans l'image sont sauvegardées dans les registres de la caméra (voir Annexe B) afin de pouvoir être utilisées par le produit de tracking. Une autre information sur ces motifs est également sauvegardée : la valeur de l'intensité du pixel au centre de chaque rond. Cette valeur sera également utilisée dans le prochain produit.

III. 2. 5. Produit de Tracking

Ce second produit suit la position des ronds dans l'image et calcul la position la voiture en fonction de ces positions. Pour cela, le produit procède en trois étapes pour chaque rond, puis calcule la position de la voiture :

- Récupération de l'ancienne position
- Seuillage de la zone autour de cette ancienne position
- Sélection de l'élément connexe rond le plus gros

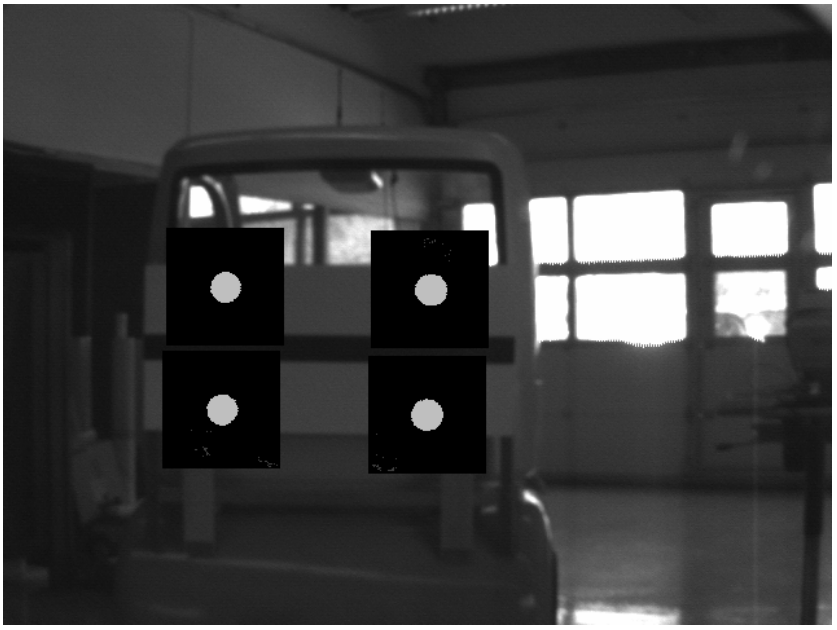

Figure 12 : Tracking des motifs

La position des ronds dans l'image est sauvegardée dans les registres de la caméra (voir Annexe B) soit par le produit précédent, soit par celui-ci. Un script mathématique permet de récupérer ces valeurs afin de pouvoir positionner les zones où seront cherchés les ronds. La valeur de l'intensité du pixel au centre du rond est également récupérée par ce script.

Ensuite, comme précédemment, un module permet d'effectuer un seuil sur la zone sélectionnée. Afin que ce seuil soit le plus efficace possible, il est fixé un peu au-dessus de la valeur récupérée par le script précédent. Ceci permet d'être sûr que les ronds sont bien distingués du reste de l'image. Ceci permet également de faire face à d'éventuels changements de luminosités. Les éléments connexes ainsi détectés sont filtrés de la même façon que pour le produit d'initialisation : seuls les éléments dont la surface est suffisante et dont le rapport $4 * p * Surface / Périmètre^2$ est suffisamment grand sont sélectionnés.

Le calcul de la localisation de la voiture peut commencer. La nouvelle position des ronds est sauvegardée dans les registres, ainsi que la valeur de l'intensité du pixel au centre du rond.

III. 4. La communication avec la caméra

La communication avec la caméra se fait principalement grâce à deux ports TCP/IP : le port 5000 et le port 5001. Le premier sert à dialoguer avec la caméra, alors que le second ne peut servir qu'à récupérer des résultats. Sur ces deux ports, la communication se fait par des envoie de chaînes de caractères en ASCII. Celles ci ne sont pas terminées par un caractère nul, mais par un caractère *'Retour chariot'* suivit d'un caractère *'Nouvelle ligne'*.

Afin de récupérer les résultats que la caméra calcule, plusieurs solutions ont été testées. Ces solutions utilisent toutes un de ces deux ports, ou les deux à la fois. Les résultats récupérés sont les quatre paramètres permettant de localiser la voiture leader (Distance, Déviation, α, β : voir Figure 6) ainsi que une horloge, interne à la caméra, permettant de savoir quand les paramètres ont été calculés, et un indicateur permettant de savoir si la localisation de la voiture a été possible ou non.

III. 4. 1. Première solution : Utilisation du port 5001

Il est possible de configurer la caméra de telle sorte qu'elle envoie des chaînes de caractères sur le port 5001 à la fin de chaque inspection. Il est également possible d'envoyer différentes chaînes de caractères en fonction du résultat des senseurs. C'est donc sur cette solution que je me suis tourné en premier.

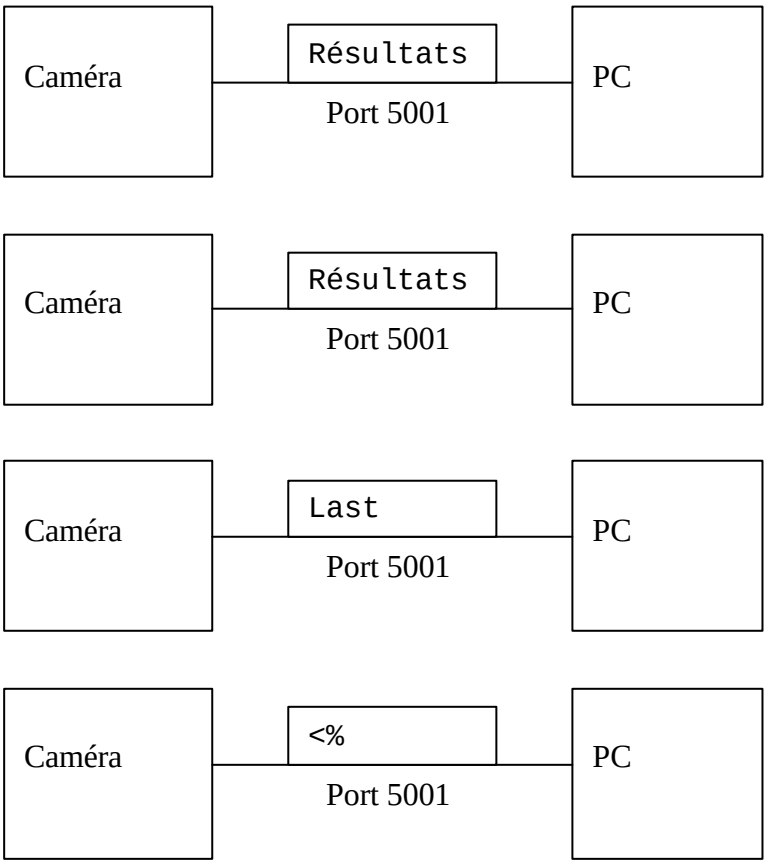

Figure 13 : Echanges sur le port 5001

Après chaque inspection, une chaîne de caractères contenant les résultats est envoyée sur le port 5001. Cette chaîne est de la forme suivante : `"horloge drapeau distance déviation alpha béta %"`. Le drapeau étant à 1 si les calculs ont abouti et à zéro dans le cas contraire. Mais à partir du moment où l'ordinateur s'est connecté à la caméra sur ce port 5001, les messages sont envoyés à chaque inspection, que le programme lise ces résultats ou

non. Si le programme ne lit pas régulièrement les messages, ils seront alors entassés dans un buffer jusqu'à leur lecture. De plus, comme il n'est pas possible de savoir a priori combien de messages ont été envoyés, il a fallu trouver un moyen pour récupérer le dernier message envoyé.

Pour cela, une autre possibilité a été utilisée. Sur la caméra, il est possible de configurer certaines réponses à des messages reçus sur le port 5001. Ainsi, la caméra est configurée de façon à ce qu'elle réponde le message `"<%"` lorsque l'ordinateur envoie le message `"Last"`. Ainsi, avant de lire les résultats, l'ordinateur envoie le message `"Last"`. Ensuite, l'ordinateur lit les résultats jusqu'à trouvé le message `"<%"`. Le dernier résultat calculé par la caméra est alors celui précédent ce message.

Une variante de cette solution a également été testée. Plutôt que de laisser la caméra acquérir des images et calculer des inspections en continue, c'est l'ordinateur qui commande à la caméra de lancer une inspection. Ceci est possible lui envoyant le message `"#YI"` sur le port 5000 (voir Annexe A). Les messages ne sont donc plus envoyés sans contrôle sur le port 5001. Il suffit de lire un message puis de lancer la prochaine inspection (voir Figure 14). De cette façon, les messages ne s'entassent plus dans un buffer s'ils ne sont pas lus régulièrement. Mais la caméra n'effectue plus ses calculs à sa vitesse maximum.

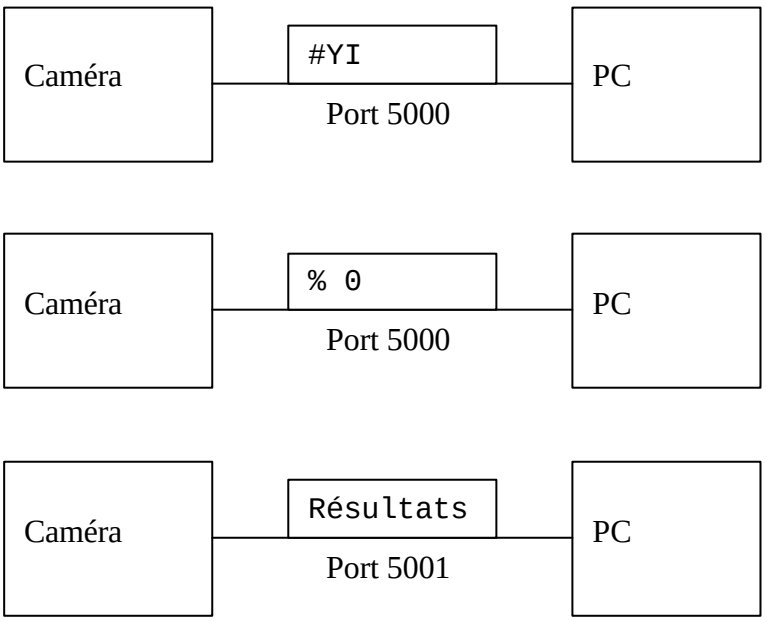

Figure 14 : Echange sur les ports 5000 et 5001

III. 4. 2. Deuxième solution : Utilisation des registres de la caméra

La caméra possède des registres que l'on peut utiliser en lecture ou en écriture depuis la caméra elle-même grâce aux scripts, ou depuis un ordinateur grâce au port 5000. Ces registres sont au nombre de 4096 et contiennent chacun 8 bits. Il est possible de les utiliser pour y enregistrer soit des valeurs entières, en utilisant 1, 2, 4 ou 8 registres, ou bien des nombres réels, en utilisant 4 ou 8 registres.

Les scripts de la caméra enregistrent les résultats dans ces registres. L'ordinateur peut demander à tout moment le contenu de n'importe quel registre. Pour cela, il lui suffit d'envoyer un message du type `"#Rq 201 D"` (voir Annexe A). La caméra renvoie alors la valeur du nombre réel en double précision enregistrée sur les registres 201 à 209. Afin d'être sûr de ne pas avoir de mélange entre des anciennes et des nouvelles valeurs, un système de sémaphore a été mis en place. C'est le registre numéro 200 qui joue ce rôle. Que ce soit la caméra ou l'ordinateur, chacun attend que ce registre ai la valeur 0. Puis ce registre est mis à 1

ou à 2 selon que c'était l'ordinateur ou la caméra qui utilise les registres. Puis, après écritures ou lecture des résultats, ce registre 200 est remis à 0.

Les résultats sont enregistrés en tant qu'entiers en double précision. Chaque valeur utilise donc 8 registres. Ces résultats sont enregistrés dans les registres 201 à 248 (voir Annexe B). La lecture de toutes les valeurs demande alors un nombre important d'échanges sur le réseau (voir Figure 15).

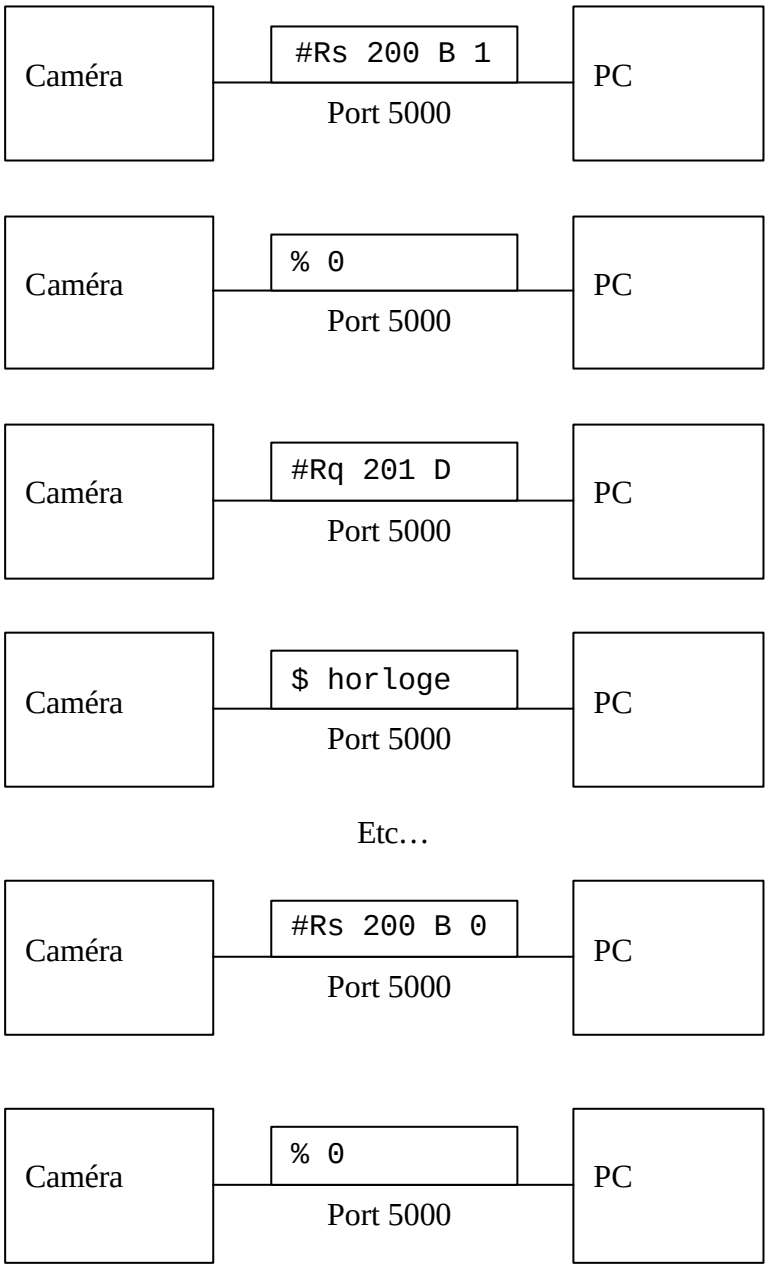

Figure 15 : Lecture des registres

III. 4. 3. Troisième solution : Lecture des résultats du senseur

Il est également possible de lire les résultats de chaque senseur. Pour cela, il faut sélectionner le produit contenant le senseur dont on veut lire les résultats. Une fois cette opération effectuée, il est possible de demander les résultats du senseur désiré grâce à une instruction du type `"#SO 5"` (voir Annexe A).

Le problème est que si l'on demande les résultats alors que la caméra n'a pas encore fini l'inspection, ceux-ci sont nuls. Il faut donc demander les résultats entre deux inspections pour qu'ils aient une signification. Cette solution est donc difficilement envisageable dans le cadre de l'application que l'on veut développer.

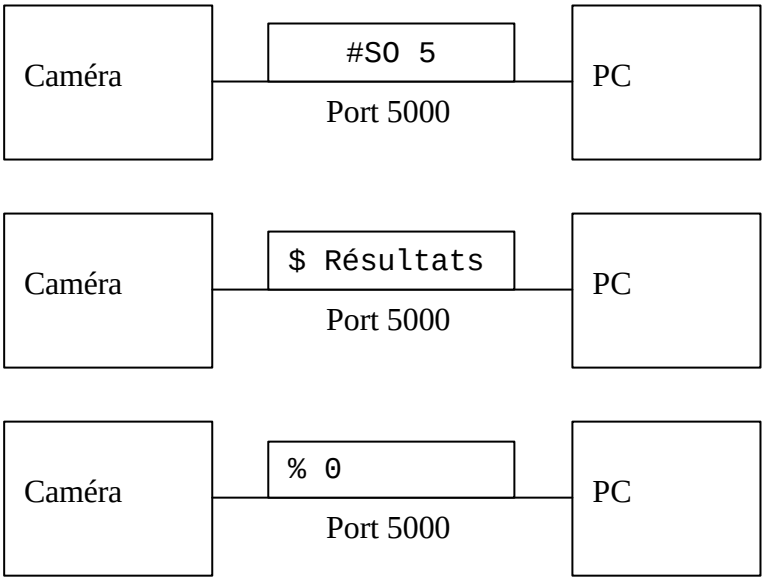

Figure 16 : Récupération des résultats d'un senseur

III. 4. 4. Quatrième solution : Lecture des résultats en tâche de fond

Le problème avec les solutions précédentes, c'est que la communication prend beaucoup de temps. Pour éviter que cette communication perde du temps processeur juste en attendant les réponses de la caméra, une autre solution a été mise en place. La caméra est configurée pour envoyer ses résultats à chaque inspection sur le port 5001. Sur l'ordinateur une boucle récupère ces résultats en tâche de fond.

Lors de la connexion avec la caméra, la boucle est lancée est tâche de fond. Elle récupère les résultats en permanence, et les place dans une zone tampon. Le programme principal peut alors lire ces résultats dans la zone tampon à tout moment. Il faut ensuite arrêter cette boucle lors de la déconnexion de l'ordinateur avec la caméra.

C'est cette solution qui a été retenue car elle permet à l'application principale de ne pas être bloquée pendant la récupération des résultats (voir Figure 17).

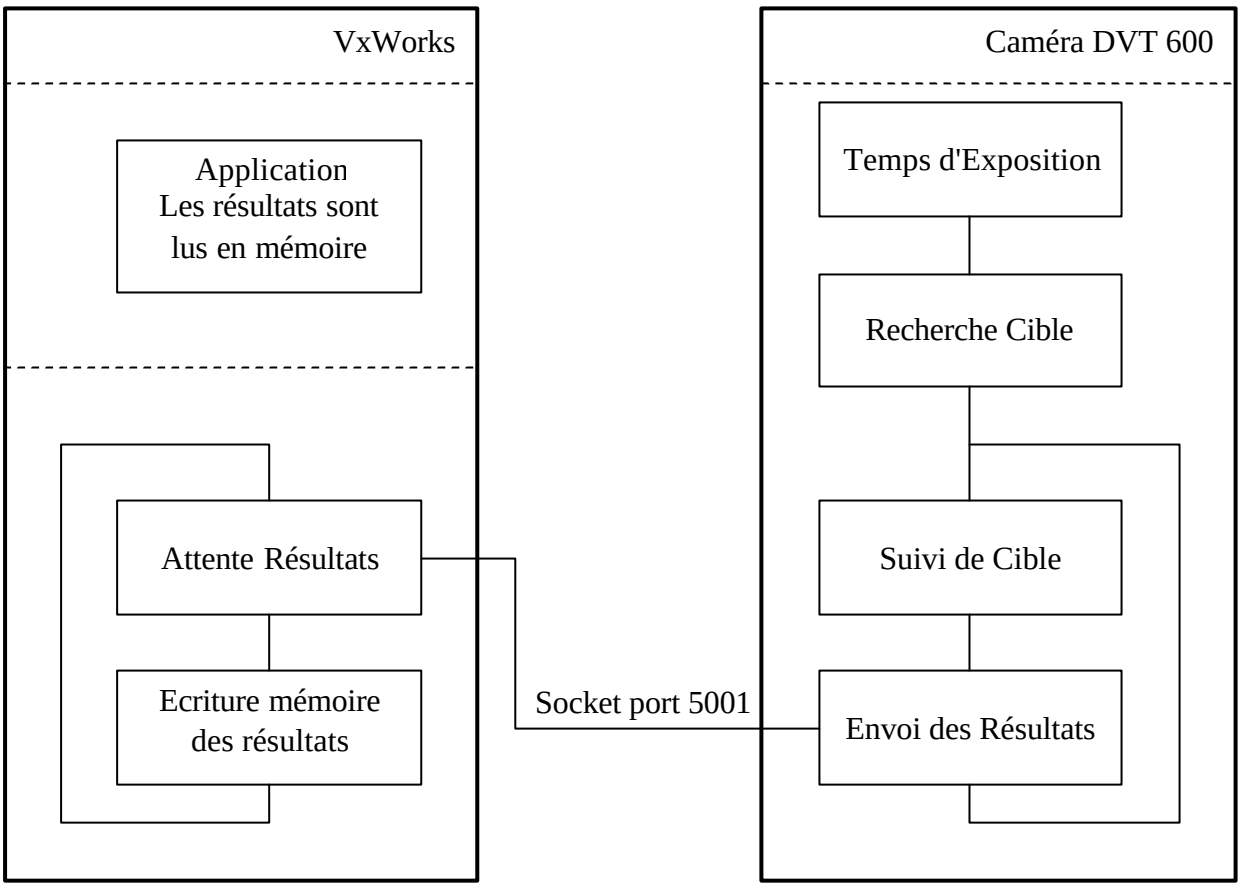

Figure 17 : Automate de tracking

III. 5. Implantation sur le Cycab

III. 5. 1. La librairie dvt

Afin de pouvoir utiliser la caméra, une librairie a été écrite : la librairie dvt. Pour chacune des solutions décrites ci-dessus, une version de la librairie a été écrite. Ces versions sont toutes composées de la même façon. Il n'y a ainsi pas de modifications à apporter au code en fonction de la version utilisée, à l'exception de la version utilisant le lancement des inspections par l'ordinateur (voir page 20). Cette librairie se compose donc des cinq fonctions suivantes : `dvtOpen`, `dvtSetParameter`, `dvtChangeProduct`, `dvtGetresultat` et `dvtClose` plus une fonction `dvtTriggerInspection` pour la version nommée précédemment.

Pour utiliser la caméra, il faut commencer la communication avec celle-ci grâce à la fonction `dvtOpen` en indiquant le numéro IP, ou l'adresse de la caméra. Il faut ensuite régler les paramètres de la caméra : écartement horizontal et vertical des ronds. Il suffit alors de choisir quel produit on veut utiliser, puis de récupérer les résultats quand on veut. Lorsque l'on a plus besoin de la caméra, il faut utiliser la fonction `dvtOpen` afin de clore la connexion avec celle-ci.

Toutes ces fonctions prennent comme argument une variable de type dvtsensor. Cette variable contient les informations sur les liaisons avec la caméra. C'est également dans cette variable que sont enregistrés les résultats. En effet cette structure contient un champ 'valeur' de 10 réels qui sont mis à jours lors de l'utilisation de la fonction `dvtGetresultat`. Le premier réel est l'horloge de la caméra qui permet de savoir si les résultats sont de nouveaux résultats ou si ce sont des résultats qui ont déjà été traités. Le second est un indicateur permettant de savoir si les calculs ont abouti à la localisation de la voiture ou non. Les quatre suivants sont les paramètres Distances, Déviation, α et β. Les autres valeurs ne sont pas utilisées pour le moment.

III. 5. 2. ORCCAD

ORCCAD (Open Robot Controller Computer Aided Design) est un environnement logiciel permettant de concevoir et de mettre en œuvre le contrôle et la commande d'un système robotique complexe. Il permet également la spécification et la validation des missions à réaliser par ce système.

est un environnement de programmation destiné aux systèmes temps réels. Il permet de programmer simplement des automates. Pour cela, il faut créer une ou plusieurs tâche robot (voir Figure 18). Une tâche Robot se constitue de plusieurs modules reliés entre eux. Ces modules sont de trois types :

- Les modules Automates : Il doit y en avoir un par tâche Robot.
- Les modules Ressource Physique : Ces modules permettent de communiquer avec des éléments physiques, comme un laser ou une camera par exemple.
- Les modules Algorithmiques : Ces modules effectuent des calculs sur les variables qui lui sont passées et fournissent les résultats des calculs en sortie.

Pour chacun des ces modules, plusieurs fichiers sont associés. Ces fichiers permettent de décrire le comportement des modules en langage c. Pour les modules Ressource Physique et les modules Algorithmiques, il y a notamment un fichier `init.c` qui est exécuté au début de l'exécution, un fichier `getresultat.c` pour les modules Ressource Physique et `compute.c` pour les modules Algorithmique qui est exécuté régulièrement, et un fichier `end.c` qui est exécuté à la fin de la tâche Robot.

A mon arrivée à l'INRIA, une tâche Robot de suivi existait déjà. Cette tâche Robot comportait un module Ressource Physique correspondant à une caméra linéaire. J'ai donc repris cette tâche Robot, et le module correspondant à la caméra linéaire a été remplacé par

deux modules. Le premier est une Ressource Physique qui récupère les données transmises par la caméra, le second est un module Algorithmique qui permet de transformer les informations transmises par la caméra au même format que celles qui étaient transmises par la caméra linéaire. Ceci a permis de ne pas retoucher le code correspondant aux autres modules.

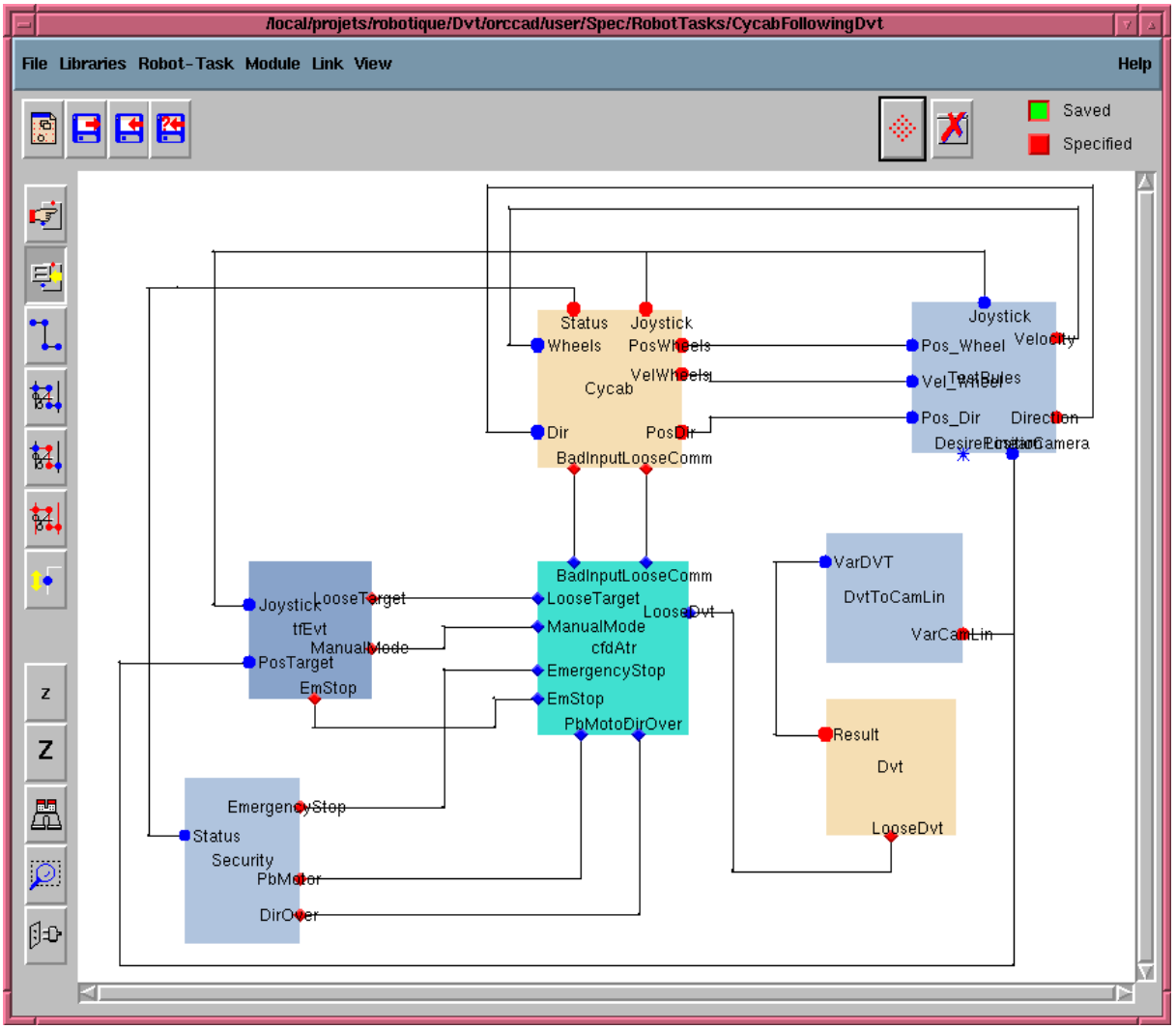

Figure 18 : Tâche Robot

IV. Suivi d'une trajectoire filmée

IV. 1. Introduction

Le seul point commun avec la partie précédente est le Cycab. La manipulation qui suit fonctionne avec une caméra standard reliée par un émetteur vidéo radiofréquence à une station possédant une carte d'acquisition vidéo. La station renvoie ensuite les résultats au Cycab via une liaison éthernet sans fil.

Figure 19 : Système pour le suivi de trajectoire

L'objectif de ce projet est le suivi de trajectoire du Cycab par correction visuelle. Au cours d'une trajectoire de référence, les images prises par la caméra sont enregistrées sur disque. Le but est alors de suivre à nouveau cette trajectoire. Pour cela, à chaque instant, on compare une image prise par la caméra avec l'une des images de référence et on calcul la correction à apporter. Afin d'effectuer ces comparaisons, on suppose que l'erreur reste faible, donc que les images sont présentent de grandes similitudes.

Afin d'effectuer les essais, deux séquences ont étés sauvegardées sur disque. Ces séquences ont été filmées depuis le Cycab. Ce Cycab était alors conduit manuellement en suivant la même trajectoire. Des erreurs sont donc introduites du fait que la conduite était manuelle. Ce sont ces erreurs qu'ils faut estimées. Pour cela, on s'appuie sur la théorie suivante.

IV. 2. Calibrage de la caméra

En coordonnées homogène, il est possible de décrire la projection d'un point 3D dans une image de la façon suivante :

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = \begin{bmatrix} a_u & c & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & R & & T \\ & & & \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

On note K la matrice intrinsèque de la caméra et M la matrice de projection. R est la matrice de rotation de la caméra par rapport au repère de la scène et T est la translation de la caméra par rapport à ce même repère :

$$K = \begin{bmatrix} a_u & c & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad M = K \cdot \begin{bmatrix} & R & & T \\ & & & \end{bmatrix}$$

Le calibrage de la caméra consiste à calculer les paramètres de la matrice intrinsèque K. Cette matrice est de la forme suivante :

$$K = \begin{bmatrix} k_u \cdot f & c & u_0 \\ 0 & k_v \cdot f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- k_u et k_v sont les facteurs d'échelle en pixels/mm
- f est la focale de la caméra
- u_0 et v_0 sont les coordonnées en pixel de l'intersection de l'axe avec la matrice CCD
- c est un coefficient qui est nul lorsque les axes de l'image sont orthogonaux

Afin d'obtenir cette matrice, il faut prendre une image d'une mire possédant un grand nombre d'objets facilement localisable dans l'image et dont on connaît exactement les coordonnées tridimensionnelles (voir Figure 20). A partir de cette image, un programme de calibrage détecte l'emplacement dans l'image du centre des pastilles blanches. Une correspondance entre les coordonnées pixels et les coordonnées tridimensionnelles est alors effectuée. Ceci permet de calculer les paramètres intrinsèques de la caméra.

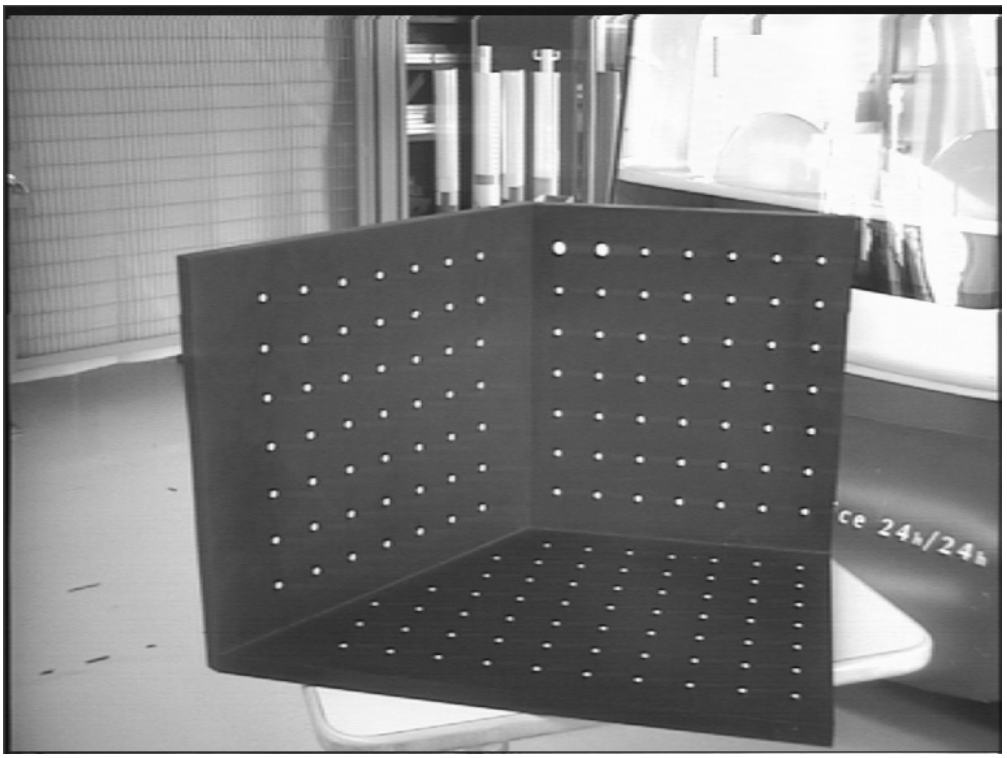

Figure 20 : Calibrage de la caméra

IV. 3. Calcul du mouvement

Si on effectue deux prises de vue de la même scène avec cette caméra depuis deux positions légèrement différentes, il est possible, en retrouvant les points images correspondant, et en utilisant la géométrie épipolaire, de retrouver des informations sur le mouvement effectué entre ces deux prises de vues.

La contrainte épipolaire caractérise le fait que le point P_2 correspondant dans l'image 2 au point P_1 dans l'image 1 se situe forcément sur une droite dans l'image 2 (voir Figure 21). Cette droite est appelée droite épipolaire du point P_1 dans l'image 2. Cette contrainte peut être exprimée sous forme algébrique :

$$p_2^{\,t} \cdot F \cdot p_1 = 0$$

p_1 et p_2 représente les coordonnées homogènes en pixels des points P_1 et P_2. F est appelée la matrice fondamentale.

A partir de ces coordonnées p_1 et p_2 en pixels, on peut retrouver les coordonnées x_1 et x_2 de ces mêmes points dans le repère correspondant au plan image de chaque prise de vue. Pour cela, il faut utiliser la matrice intrinsèque de la caméra :

$$p_1 = K \cdot x_1, p_2 = K \cdot x_2$$

On en déduit alors une nouvelle expression de la contrainte épipolaire. Cette expression fait intervenir la matrice essentielle E :

$$x_2^{t} \cdot K^{t} \cdot F \cdot K \cdot x_1 = 0$$

$$x_2^{t} \cdot E \cdot x_1 = 0$$

Cette matrice essentielle dépend uniquement du mouvement existant entre les deux prises de vue. C'est à partir d'elle qu'il est possible de retrouver la rotation et la translation entre les deux prises de vue. La translation est alors connu à un facteur d'échelle près.

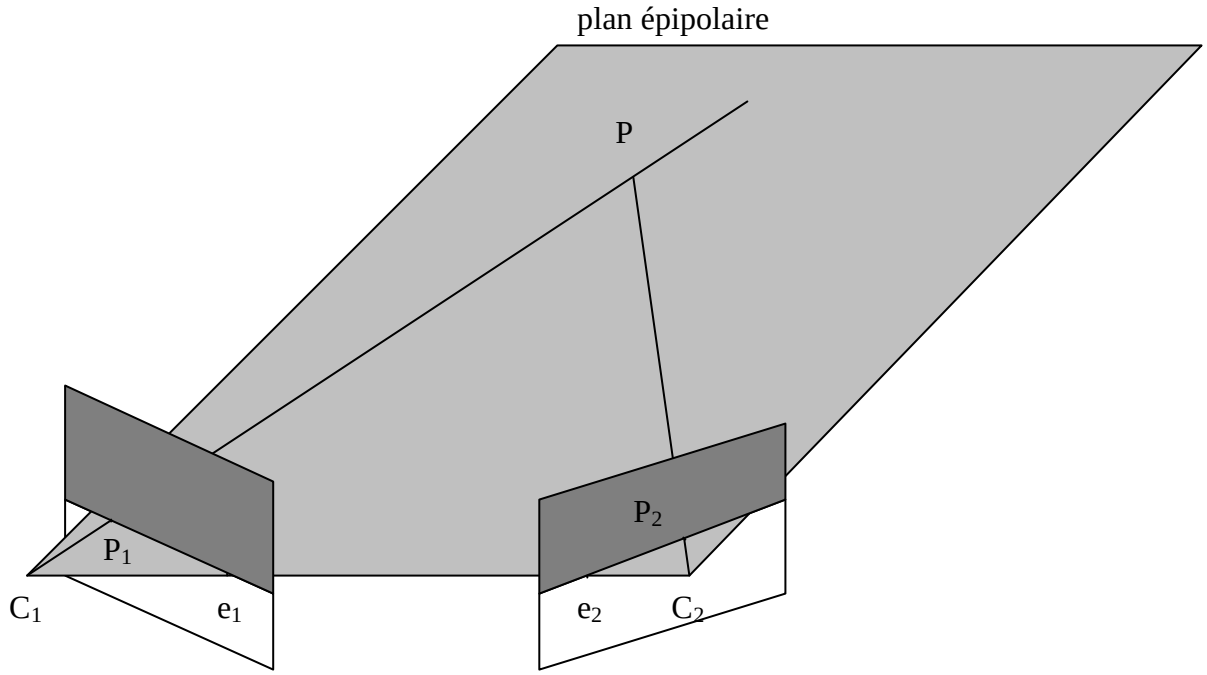

Figure 21 : Contrainte épipolaire

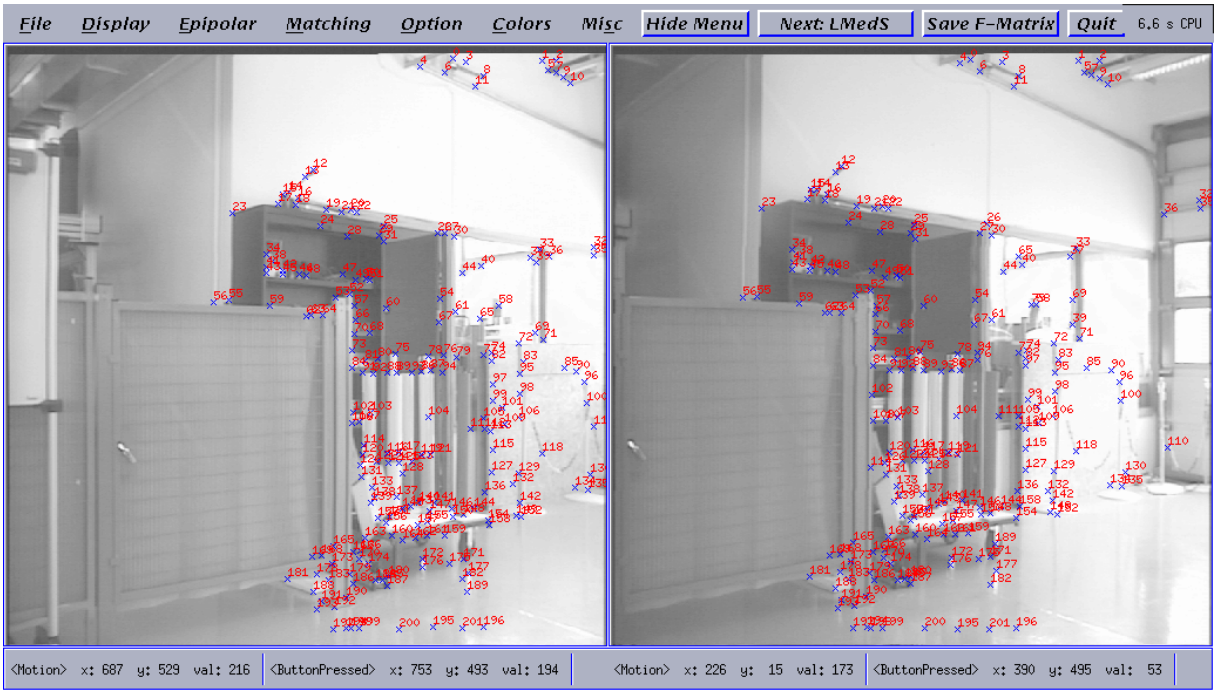

Figure 22 : Points correspondants dans deux images

Mais avant de pouvoir effectuer tous ces calculs, il faut d'abord trouver un grand nombre de points correspondant. Sinon, les calculs ne seront pas possible, ou ils seront beaucoup trop aléatoires. Pour calculer ces points correspondant, on procède en plusieurs étapes. On cherche tout d'abord des points d'intérêt dans chacune des deux images. Ces points sont ceux autour desquels une variation d'intensité relativement importante existe.

On effectue alors un premier appariement entre les points d'intérêt de la première image avec ceux de la deuxième image. Ceci permet de calculer une première estimation de la matrice fondamentale.

On élimine alors tous les appariements qui ne vérifient pas avec suffisamment de précision cette première contrainte épipolaire. Il ne reste alors normalement que des appariements corrects (voir Figure 22). On peut alors avec ces appariements calculer la matrice fondamentale avec une grande précision.

IV. 4. Résultats

Après ces essais, il semble difficile d'utiliser cette technique afin de corriger une trajectoire. En effet, les résultats obtenus sont très instables. Ceux-ci ont pourtant été réalisés dans des conditions très bonnes si on les compare à ce qu'elles seraient dans des conditions réelles. Par exemple, il n'y a pas de changement de luminosité entre les deux séquences, il n'y a pas non plus d'objets qui ont bougés entre les deux séquences.

Conclusion

Le système développé avec la caméra DVT600 fonctionne correctement et a donc permis de valider ses possibilités. Le temps de ce développement ayant pris plus de temps que prévu, je n'ai malheureusement pas eu le temps de beaucoup approfondir la deuxième partie.

Néanmoins, ce stage que j'ai effectué à l'INRIA Rhône-Alpes m'a permis d'utiliser mes connaissances en matière de vision par ordinateur. De plus j'ai pu appréhender deux faces différentes du traitement de l'image. En effet la première partie m'a permis de voir un côté pratique, alors que la seconde partie faisait appel à un côté beaucoup plus théorique.

Bibliographie

[1] *SmartImage Installation & User Guide*, 2nd Edition
DVT Corporation
2000

[2] Hervé Mathieu
La chaîne de l'acquisition d'image (http://www.inria.fr/rrrt/rt-0246.html)
Institut National de Recherche en Informatique et en Automatique (INRIA)
Décembre 2000

[3] Gérard Baille, Philippe Garnier, Hervé Mathieu
Le cycab de l'INRIA Rhône-Alpes (http://www.inria.fr/rrrt/rt-0229.html)
Institut National de Recherche en Informatique et en Automatique (INRIA)
Avril 1999

[4] *FrameWork FirmeWare 1.3 Command List* (CompleteCommandList13.pdf)
DVT Corporation
11 février 1998

[5] *Script Reference Manual*, 3rd Edition (DVTScriptReference.pdf)
DVT Corporation
Janvier 2001

[6] http://www.dvtsensors.com/

[7] http://www.dvtsensors.com/support/DVTDownloads.htm

Annexe A : Principales commandes

Voici les principales commandes que l'on peut utiliser pour communiquer avec la caméra DVT 600. Ces commandes doivent être envoyées à la caméra soit par la liaison série, soit par la liaison éthernet, sur le port 5000. Pour avoir la liste complète de ces commandes, voir [4].

- Information sur les produits sauvés dans la mémoire de la caméra
  Commande :
  ```
  #PQ
  ```
  Réponse :
  ```
  $ produit_sélectionné produit_en_cours_d'inspection
    produit_démarré_à_la_mise_sous_tension
  $ 1 premier_produit
  $ 2 second_produit
  …
  % 0
  ?
  ```

- Sélection d'un produit pour l'inspection
  Commande :
  ```
  #PI numéro_du_produit
  ```
  Réponse :
  ```
  % 0
  ?
  ```

- Sélection d'un produit
  Commande :
  ```
  #PS numéro_du_produit
  ```
  Réponse :
  ```
  % 0
  ?
  ```

- Régler le temps d‘exposition d’un produit (en microsecondes)
  Commande :
  ```
  #PP numéro_du_produit E temps
  ```
  Réponse :
  ```
  % 0
  ?
  ```

- Régler le gain (0-245)
  Commande :
  ```
  #PP numéro_du_produit G gain
  ```
  Réponse :
  ```
  % 0
  ?
  ```

- Demande d'informations sur les senseurs
  Commande :
  ```
  #SQ
  ```
  Réponse :
  ```
  $ nombre_de_senseurs
  $ 1 type nom
  $ 2 type nom
  …
  % 0
  ?
  ```

- Demande des résultats d'un senseur
  Commande :
  ```
  #SO numéro_du_senseur
  ```
  Réponse (Pour un script mathématique):
  ```
  $ Position.X Position.Y Position.ThetaX Position.ThetaY
    Distance Angle Point.X Point.Y
  % 0
  ?
  ```

- Demande de la valeur d'un ou plusieurs registres
  Commande pour un entier sur un octet (0 à 255):
  ```
  #Rq numéro_du_registre B
  ```
  Commande pour un entier sur 2 octets (-32768 à 32767):
  ```
  #Rq numéro_du_registre S
  ```
  Commande pour un réel sur 4 octets :
  ```
  #Rq numéro_du_registre F
  ```
  Commande pour un réel sur 8 octets :
  ```
  #Rq numéro du registre D
  ```
  Réponse :
  ```
  $ valeur
  % 0
  ?
  ```

- Sauvegarde d'une valeur dans un ou plusieurs registres
  Commande pour un entier sur un octet (0 à 255):
  ```
  #Rs numéro_du_registre B valeur
  ```
  Commande pour un entier sur 2 octets (-32768 à 32767):
  ```
  #Rs numéro_du_registre S valeur
  ```
  Commande pour un réel sur 4 octets :
  ```
  #Rs numéro_du_registre F valeur
  ```
  Commande pour un réel sur 8 octets :
  ```
  #Rs numéro_du_registre D valeur
  ```
  Réponse :
  ```
  % 0
  ?
  ```

- Démarrer ou Arrêter les inspections
  Commande pour démarrer:
  ```
  #Y+
  ```
  Commande pour arrêter:
  ```
  #Y-
  ```
  Réponse:
  ```
  % 0
  ?
  ```

- Passer en mode Trigger interne ou externe
  Commande pour le mode interne:
  ```
  #Yp+
  ```
  Commande pour le mode externe:
  ```
  #Yp-
  ```
  Réponse:
  ```
  % 0
  ?
  ```

- Lancer une inspection (si la caméra est en mode trigger externe)
  Commande :
  ```
  #YI
  ```
  Réponse :
  ```
  % 0
  ?
  ```

Annexe B : Utilisation des registres de la caméra

Numéro :	Type :	Description :
100 – 107	Float	Coefficient utile pour calculer les distances et les angles
108 – 115	Float	Espacement vertical entre les ronds
116 – 123	Float	Espacement horizontal entre les ronds
150 – 151	Short	Position en X dans l'image du rond en haut à gauche
152 – 153	Short	Position en Y dans l'image du rond en haut à gauche
154 – 155	Short	Position en X dans l'image du rond en haut à droite
156 – 157	Short	Position en Y dans l'image du rond en haut à droite
158 – 159	Short	Position en X dans l'image du rond en bas à gauche
160 – 161	Short	Position en Y dans l'image du rond en bas à gauche
162 – 163	Short	Position en X dans l'image du rond en bas à droite
164 – 165	Short	Position en Y dans l'image du rond en bas à droite
170 – 171	Short	Intensité du rond en haut à gauche
172 – 173	Short	Intensité du rond en haut à droite
174 – 175	Short	Intensité du rond en bas à gauche
176 – 177	Short	Intensité du rond en bas à droite
200	Byte	Sémaphore pour les registres 201 à 248
201 – 208	Float	Horloge
209 – 216	Float	0 : cible non détectée; 1 : cible détectée
217 – 224	Float	Distance
225 – 232	Float	Déviation
233 – 240	Float	Alpha
241 – 248	Float	Béta

Tableau 2 : Utilisation des registres de la caméra