Programmation d'un Système de Vision Temps Réel Embarqué sur un Robot Quadrupède Mobile et Autonome

Patrick Bonnin (1,2), Didier Solheid (1,2), Vincent Hugel (1), Pierre Blazevic (1),
Laboratoire de Robotique de Paris, 10 - 12 Avenue de l'Europe, 78140 Vélizy
IUT de Villetaneuse, Dépt GEII, Av JB Clément, 93430 Villetaneuse
{bonnin, hugel, pierre @robot.uvsq.fr}

Résumé :

Nous présentons la programmation d'un Système de Vision Temps Réel Embarqué sur un robot quadrupède mobile et autonome, conçu et réalisé par SONY dans le contexte de la RoboCup (compétition de robots jouant au football en équipe).

Nous analysons les problèmes rencontrés par le système de vision temps réel : l'éclairage, la puissance de calcul disponible, ainsi que le contexte temps réel dans le cas d'un robot à pattes.

Nous présentons la place du système de vision parmi l'ensemble des développements algorithmiques embarqués. Puis nous nous focalisons sur le système de vision temps réel embarqué : son rôle, l'évaluation de sa qualité dans le contexte temps réel (que nous définissons) et de la RoboCup, les grandes lignes de sa conception, l'interface conviviale développée pour son réglage mais également pour un cadre plus général d'analyse d'images couleur, ainsi que le détail des algorithmes embarqués et concluons sur les résultats obtenus.

Abstract :

We present a software implementation of a real time vision system which is inboard a mobile and autonomous four legged robot, designed and realized by SONY. This implementation was realized for the RoboCup context (competition between robot teams playing soccer).

The encountered problems are analyzed : the lighting conditions, the available inboard computing power, the real time context in the case of legged robots.

We present the place of the vision system amoung the inboard software developments. Then we focus our attention on it : its rule, its evaluation in a real time context (that we define) for the RoboCup competition, the main outlines of its design, the easy to use interface specially developped for its setting and color analysis, the detail of the inboard algorithms and we conclude upon the obtained results.

1 Introduction :

La Vision et l'Interprétation de scènes en « Temps Réel » sont une tâche primordiale à réaliser et à implanter dans tout système robotique mobile et autonome. Un contexte applicatif comme celui de la RoboCup [1] permet de se confronter aux difficultés réelles. En effet, il n'est pas simple de faire jouer des robots en équipe au football. Les tâches élémentaires à réaliser par les robots sont entre autres : aller à la balle, se localiser sur le terrain, localiser les buts, aller au but adverse avec la balle, tirer au but adverse, dégager de son propre but, éviter les adversaires et partenaires etc ... Les joueurs et / ou amateurs apprécieront ! Le rôle du système de vision est primordial pour réaliser ces tâches. Notons qu'elles imposent d'importantes contraintes temporelles.

Nous présentons, dans le cadre de cette communication, la programmation d'un tel système. Après avoir décrit le contexte (§ 2), nous analyserons les problèmes rencontrés (§ 3), puis détaillerons la programmation du Système de Vision temps réel embarqué (§ 4), avant de présenter les résultats (§ 5) et de conclure sur l'intérêt d'une telle expérience (§ 6).

2 Contexte :

2.1 Le Robot :

Le robot (cf figure n°1 dernière page de l'article), sur lequel nous avons implanté le système de vision temps réel, est un robot mobile et autonome de petite taille : 25 cm de haut sur 25 cm de long et 10 cm de large, ressemblant à un animal familier de petite taille tel qu'un chat ou un petit chien. C'est bien évidemment un robot à 4 pattes, chacune étant dotée de 3 moteurs, avec une tête comportant 3 degrés de liberté commandés par 3 moteurs, et d'une queue mobile dotée d'un moteur. La vitesse de déplacement que nous obtenons est à l'heure actuelle d'environ 5 cm par seconde. Il a été conçu et réalisé par SONY [2], et mis à notre disposition dans le cadre de la RoboCup.

La partie matérielle informatique est bâtie autour d'une architecture MIPS série R 4000 [2] et de circuits ASICs pour contrôler les périphériques. La tête comporte une micro-caméra avec un objectif intégré, le tout de taille 23x16x4 mm avec une résolution de 362 x 492 pixels. Un système matériel de détection des couleurs permet de détecter jusqu'à 8 couleurs préalablement définies dans l'espace des couleurs YUV en un temps de 50 ms.

La partie logicielle informatique est constituée d'un Système d'Exploitation Temps Réel Orienté Objet, APERIOS développé par SONY. Les divers objets (cf § 3.3) développés pour la programmation du robot peuvent dialoguer entre eux par plusieurs types de communications par échange de message [2]. Ils peuvent avoir des niveaux de priorité différents.

Alimenté par deux batteries, l'autonomie du robot est d'environ 15 à 20 minutes.

Le robot est doté d'un gyromètre et d'un accéléromètre, tous deux trois axes. L'accéléromètre permet de détecter la chute du robot et d'enclencher la procédure de relevage.

La caméra est le seul capteur permettant de percevoir l'environnement pour élaborer la stratégie de jeu.

2.2 La « RoboCup » :

Le but est de faire jouer les robots au football, par équipe de 3 joueurs. Les divers éléments de la scène : la balle (orange), les buts (jaune et bleu clair), les joueurs (rouge et bleu foncé), le terrain (vert avec ses bords blancs) sont discernables par leurs couleurs respectives. Enfin, six balises, situées en hauteur, de couleur rose et bleu clair, jaune ou vert foncé sont placées de chaque côté du terrain sur les lignes des buts et sur la ligne médiane. Elles permettent aux joueurs de se localiser sur le terrain.

Les 3 robots sont totalement autonomes, ne communiquent pas entre eux, ni avec une éventuelle base. Le contrôle du jeu est donc totalement local, basé sur la seule « Vision » du robot.

3. Analyse des Problèmes

Trois principaux problèmes apparaissent au niveau de la conception et de la réalisation du système de vision :

- les conditions d'éclairage,
- la puissance de calcul souhaitable, comparée à la puissance de calcul disponible, embarquée dans un système mobile et autonome de petite taille,
- son intégration dans un système temps réel, dans un contexte de robotique à pattes.

3.1 L'Eclairage :

Les conditions d'éclairage, bien que maîtrisées car les compétitions ont lieu en intérieur sur une scène de spectacle, sont loin d'être idéales : projecteurs trop puissants qui génèrent des reflets sur les objets, ainsi que des ombres portées etc... Bien que l'éclairage soit garanti homogène : température de couleur de 4500K, et illumination de 580 Lux, les aspects des divers éléments de la scène (c'est à dire leurs plages de valeurs respectives dans les plans Y, U et V de l'image couleur) sont différents en fonction de leur situation sur le terrain (pour les objets mobiles : balle et joueurs bien évidemment) et du point d'observation c'est-à-dire de la position du robot sur le terrain. Il faut préciser que cet éclairage « théoriquement » homogène est obtenu à l'aide de 4 projecteurs de 1200 W en incidence oblique (éclairage du terrain de compétition). Paradoxalement, un seul projecteur de même puissance placé en « douche » approximativement au dessus du milieu du terrain génère certes un éclairage moins homogène selon le luxmètre (éclairage du terrain d'entraînement), mais les aspects des divers objets de la scène dans les images couleurs sont beaucoup moins dépendant de leur positionnement et du lieu d'observation.

Ce problème d'éclairage est d'autant plus délicat que dans le contexte de la compétition, nous devons tirer le meilleur parti des ressources matérielles du robot, donc utiliser le matériel spécifique de détection des couleurs, qui procède par un «multi-seuillage» (cf § 4. 4), d'où le développement d'une interface conviviale et ergonomique de réglage (cf § 4. 3 et figure n°2 en dernière page de l'article).

3.2 La Puissance de Calcul Disponible :

Notre première idée, pour effectuer la segmentation d'images couleurs (sous forme vectorielle 3 plans Y, U et V) a été d'évaluer la possibilité de réutiliser les algorithmes de Segmentation Multi-Spectrale Coopérative [3], développés en mode de programmation « Data Parallel » [4] sur Connexion Machine CM5 à 32 noeuds [5], hors contexte temps réel. Ces algorithmes constituent une adaptation au domaine du Multi-Spectral des algorithmes de Segmentation Coopérative Contour / Région [6] développés en imagerie classique. Ils proposent un nouveau niveau de fusion de données, intermédiaire entre la fusion à bas niveau (ou fusion pixel) et la fusion à niveau intermédiaire (fusion d'attributs de primitives issues de l'image) [7], car la fusion des données est assurée tout au long de la phase de segmentation. Testés sur des images du contexte de la RoboCup, il s'avère que ces algorithmes produisent des résultats de très bonne qualité « visuelle », nous reviendrons sur notre méthode « roboticienne » d'évaluation du système de vision (cf § 4 . 1).

Mais une étude des architectures parallèles spécifiques aux Systèmes de Vision [8], nous dissuade d'utiliser de tels algorithmes, beaucoup trop « gourmands » en puissance de calcul dans une optique d'implantation en temps réel, compte tenu de la puissance de calcul embarquée (cf § 2.1). En effet, compte tenu de leur volume relativement important (la Connection Machine CM5 occupe environ $1m^3$) , aucune des architectures parallèles spécifiques ne pourrait être embarquée sur notre robot quadrupède de la taille d'un petit animal familier , chef d'œuvre de miniaturisation!

En revanche, le système matériel de détection des couleurs du robot réalise une fusion de données très simple, de type « fusion pixel » à la cadence vidéo. Dans les algorithmes de type « fusion pixel », une « décision d'appartenance du pixel à une classe » est prise en fonction de la valeur du pixel dans les diverses bandes spectrales. Ces algorithmes sont généralement très efficaces compte tenu de leur rapidité [9]. Dans notre cas, le pixel est reconnu comme appartenant ou n'appartenant pas à une des 8 couleurs prédéfinies en fonction de sa valeur dans les 3 plans Y, U et V (obtenus par combinaison linéaire de R, V et B).

3. 3 L'Intégration dans le Système Temps Réel du Robot Quadrupède :

Pour des raisons d'efficacité, nous avons divisé l'ensemble des traitements embarqués sur le robot en trois objets (au sens informatique : le Système Temps Réel APERIOS développé par SONY est orienté objet) que nous appelons :

- Locomotion,
- Vision,
- et Relevage (cf figure n° 3).

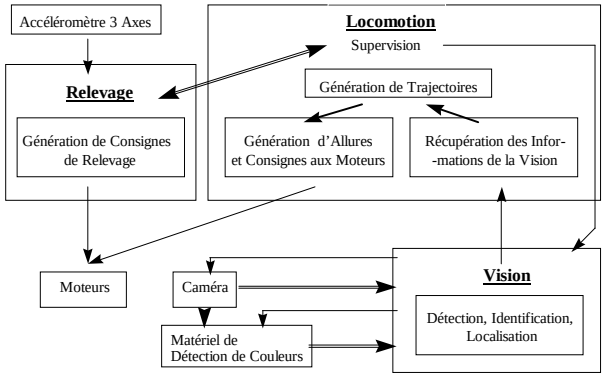

Figure n°3 : Organisation des Développements Logiciels : Les 3 Objets et leurs Interactions.

L'objet « Locomotion » représente la tâche centrale de supervision et réalise trois fonctions dont deux de bas niveau :

- la génération d'allures de marche et l'envoi des consignes correspondantes aux moteurs des axes,
- la récupération des informations de l'objet Vision (ie. du Système de Vision),
- et la troisième de plus haut niveau : l'élaboration de stratégie de déplacement sur le terrain de jeu, compte tenu du positionnement de la balle, des buts et des autres joueurs.

L'objet « Vision », thème du paragraphe suivant fournit des informations de « haut niveau » : qualitatives : la présence des divers éléments de la scène, ainsi que quantitatives leur direction angulaire et une estimation « grossière » de leur distance. Tous ces paramètres sont estimés par rapport à la caméra située dans la tête du robot. L'objet «Relevage » est nécessaire pour que le robot se relève lorsqu'il tombe, ce qui est relativement fréquent en cours de match (collisions avec les autres joueurs ou dérapages). Il préempte la tâche de supervision lorsque la chute est détectée par un accéléromètre 3 axes embarqué. Pour ne pas perturber le déroulement de la marche, l'objet Vision s'exécute avec une priorité inférieure à l'objet Locomotion. Dès que l'objet Vision a envoyé ses données, l'objet Locomotion les récupère en parallèle avec la génération d'allures et l'envoi de consignes aux moteurs.

De manière concrète, l'intégration dans un système Temps Réel, où l'objet « vision » n'est pas prioritaire introduit des aléas d'une part dans les instants d'acquisition des images : l'intervalle de temps entre l'acquisition de deux images consécutives de la séquence n'est pas régulier, d'autre part dans le temps de latence (temps entre les instants d'acquisition d'une image et de la production des informations) incluant le traitement de l'image. De manière générale, le temps de traitement d'une image n'est pas constant non plus, dès que l'on effectue des traitements qui ne sont plus dits de bas niveau, qui dépendent des données, c'est-à-dire du contenu de l'image.

En plus des problèmes d'intégration dans le système temps réel, le contexte de la robotique à pattes est particulièrement difficile pour le système de vision car la marche introduit un mouvement très saccadé des projections dans l'image des objets de la scène. Recaler les images serait bien trop long !

Dans cette première version, réalisée dans « la hâte » pour la première édition de la compétition, le système de vision est « autonome » par rapport aux autres objets : « il voit », c'est-à-dire qu'il traite l'image et en extrait toutes les informations disponibles, qu'il envoie à l'objet de « Supervision ». Dans la seconde version, qui sera mis en œuvre pour la seconde édition de la compétition, à Stockholm en août 1999, l'objet Vision « percevra » son environnement, c'est-à-dire qu'il sera relié à la partie « intelligente » de l'objet de « Supervision ».

4. Le Système de Vision Temps Réel Embarqué :

4.1 Rôle et notre Définition du « Temps Réel » :

Le rôle du système de vision est de **détecter**, **d'identifier** puis de **localiser** les divers éléments (balle, balises, buts, joueurs) constituant la scène, pendant le jeu. « Détecter » signifie extraire les composantes connexes de l'image couleur, projection dans l'image de ces éléments. « Identifier » signifie trouver la ou les composantes connexes représentant un élément donné, toujours au niveau de l'image. « Localiser » signifie déterminer les angles (site et azimut) par rapport à la tête du robot, ainsi qu'une estimation grossière de la distance entre l'élément de la scène et le robot, donc dans la scène 3D.

Dans ce contexte, la qualité de la segmentation est jugée selon des critères de « roboticiens » : obtenir le plus rapidement possible une localisation 3D relativement correcte et exploitable, sans forcément avoir une extraction dans l'image complète des éléments. Par exemple, il est impossible d'extraire la balle en sa totalité, sans extraire d'autres éléments parasites, à cause du reflet en son sommet, et de l'ombre portée en dessous. En revanche elle est correctement localisée (angle et estimation grossière sur la distance).

Notre définition du Temps Réel n'est pas de traiter les images à la cadence vidéo. Cette cadence est bien évidemment la plus importante que l'on puisse atteindre. La cadence de traitement ne dépend pas uniquement du système de Vision, mais également des autres algorithmes de commande du robot qui partagent les mêmes ressources matérielles, avec une priorité plus importante ! Notre définition du Temps Réel est l'intégration dans un système temps réel tel qu'APERIOS avec des cadences de traitement suffisantes, de manière à ce que le robot puisse effectuer correctement sa tâche (par exemple ici de suivre la balle) à une vitesse qui n'est pas limitée par le système de vision, qui est généralement le plus « gros consommateur » de ressources matérielles.

4.2 Les Grandes Lignes de sa Conception et Réalisation :

Concernant les problèmes d'éclairage, et le fait que le système matériel de détection des couleurs nécessite le réglage de 128 valeurs de seuil pour chacune des 8 couleurs à détecter (nous utilisons effectivement les 8 couleurs pour identifier tous les éléments de la scène) nous avons développé une interface conviviale d'analyse (cf figure n° 2) qui génère directement le fichier de seuils, à partir d'une estimation réalisée sur diverses zones de diverses images sélectionnées par un opérateur.

Concernant les algorithmes embarqués proprement dits, qui sont, comme tout algorithme de traitement d'image, gourmands en temps de calcul, nous avons cherché à les optimiser à deux niveaux différents :

- leur choix et / ou leur conception,
- et leur implantation.

Au niveau de leur conception, nous tentons de diminuer le nombre de balayages de l'image en regroupant les traitements locaux (en chaque pixel) qui peuvent être regroupés. Une implantation efficace temporellement remet généralement en cause le découpage « historique » en étapes d'une chaîne de traitement d'images [6]. Nous choisissons des structures de données simples et d'accès rapide. Pour cela nous avons banni les listes, à accès séquentiel. Nous privilégions si possible, pour une même fonctionnalité, un algorithme de traitement d'image à niveau intermédiaire (filtre sur attribut) plutôt qu'à bas niveau (filtre sur l'image).

Au niveau de l'implantation, les balayages de l'image sont effectués linéairement comme en modèle de programmation par flot de données. En fait, nous concevons les algorithmes « bas niveau » implantés sur le robot de manière à pouvoir les implanter sans trop de modifications sur architecture à FPGA :

- soit dans cadre de la RoboCup, mais dans la catégorie des « Middle Size », c'est-à-dire moins de 50 cm, car dans cette catégorie, l'équipe conçoit le robot sur tous les plans : mécanique, informatique matérielle et logicielle.
- soit pour des applications moins spécifiques de robotique mobile et autonome, dans le cas où la couleur est déterminante pour la perception de l'environnement.

Même s'il est plus coûteux en place, et nécessite des tests de débordement, nous privilégions une gestion statique de la mémoire en évitant les allocations et désallocations successives (car le même algorithme est appliqué sur toutes les images de la séquence), pour deux raisons : la rapidité, et éviter le problème de saturation dans le cas où le système d'exploitation ne gérerait pas correctement la désallocation.

4.3 Interface de Réglage :

Cette partie du travail est totalement différente des autres. Elle constitue néanmoins une recherche en tant que telle sur la conception et la réalisation d'outils nécessaires à l'expérimentation, notamment sur le plan de l'ergonomie. Ces outils sont indispensables certes dans un contexte de compétition entre différentes équipes, mais également dans tout contexte de manipulation sur site, hors laboratoire...

L'interface réalisée à ce propos (cf figure n°2), en Builder C++ permet :

- la visualisation d'un nombre important d'images (en général une bonne dizaine d'images sont nécessaires pour assurer des réglages corrects),
- la sélection dans les images de diverses zones représentant les objets de la scène à des emplacements différents sous plusieurs points de vue,
- la vérification de la cohérence des seuils des différentes couleurs de référence (1 pixel peut appartenir à au plus une couleur de référence),
- la visualisation des résultats des traitements obtenus en simulation : l'identification des différentes composantes connexes,
- la génération automatique d'un fichier de seuils (128 valeurs de seuil pour chacune des 8 couleurs de référence !) qui sera téléchargé sur le robot.

Pour l'avoir vécu, c'est à « quatre pattes » sur le terrain avec l'ordinateur portable et le robot, sous les yeux du public attendant le prochain match, que l'on s'aperçoit de l'intérêt d'une recherche sur l'ergonomie des outils de réglage !

4. 4 Algorithmes Embarqués sur le Robot :

Les algorithmes embarqués de vision sont organisés en 3 étapes, correspondant aux 3 tâches à remplir par le système de vision : - la détection, - l'identification, - la localisation.

L'étape de détection est constituée des algorithmes :

1. de détection des couleurs, utilisant le matériel spécifique, produisant , au bout d'un délai de 50 ms, une image de profondeur un octet où chaque bit représente la détection ou non de l'une des 8 couleurs prédéfinies,
2. d' « ouverture » simultanée sur chacun des 8 bits de l'image résultant de la détection des couleurs, par un élément structurant 3x3 centré 8-connexe, de manière à éliminer les détections parasites dues aux réflexions, compte tenu d'un éclairage trop puissant,
3. d'Extraction des Composantes Connexes selon l'algorithme bien connu [10], mais que nous avons implanté de manière à obtenir tous les attributs des composantes connexes (couleur, surface, centre de gravité, boite englobante) en un seul passage sur l'image, le réétiquetage des pixels de l'image, tenant compte des équivalences de l'image des étiquettes ne nous intéressant pas (*).
4. de suppression des composantes connexes trop petites (filtrage sur l'attribut surface),
5. de fusion de composantes connexes « proches » et de même couleur en ensembles de composantes connexes (non connexes, bien évidemment !), pour regrouper les morceaux extraits séparément d'un même objet, compte tenu des conditions d'éclairage difficiles.

L'étape 1 effectue la fusion des informations contenues dans les 3 canaux Y, U et V. Cette fusion est du type « fusion pixel » [7]. Un tel traitement pourrait être utilisé avec des canaux différents dans le cadre d'une Segmentation Multi-Spectrale. Pour chaque couleur à détecter, le canal Y (codé sur un octet) est discrétisé sur 32 niveaux. Pour chaque niveau, la couleur est représentée par un parallélépipède dans l'espace YUV compris entre les plans Umin, Umax, Vmin, Vmax où Umin, Umax, Vmin, Vmax sont les valeurs extrémales respectivement pour les canaux U et V.

Les étapes 1, 2 et 3 sont de bas niveau, appliquées sur tous les pixels de l'image, les étapes 4 et 5 sont de niveau intermédiaire, appliquées sur les attributs des primitives composantes connexes.

Remarquons que l'étape 2 pourrait très bien être supprimée sans entraîner de modifications significatives sur les résultats. Même si l'ouverture, nécessitant deux balayages de l'image, peut paraître longue, elle permet d'éliminer de nombreux pixels parasites qui génèrent un grand nombre de composantes connexes très petites éliminées ensuite par l'étape 4. Elle évite ainsi, en chaque pixel éliminé, un calcul lourd de remise à jour des paramètres des composantes connexes, d'où un gain de temps important, et réduit le nombre de composantes connexes traitées, d'où un gain mémoire également important. Paradoxalement, cette remarque ne remet pas en cause le fait que nous privilégions pour une même fonctionnalité des algorithmes du plus haut niveau possible (cf § 4.2), car ici le but est d'accélérer l'étape la plus longue, c'est-à-dire l'étape 3.
(*) Pour une visualisation des résultats, le réétiquetage est effectué sur le simulateur (implanté sur PC) des traitements embarqués.

Concernant l'étape 3, en chaque pixel appartenant à l'une des 8 couleurs prédéfinies, sont relevés ou mis à jour les paramètres suivants : la surface de la composante connexe, les sommes des valeurs des positions du pixel selon x et y (pour le calcul du centre de gravité), les paramètres de la boîte englobante (Xmin, Xmax, Ymin, Ymax). Lors de la fusion entre 2 composantes, nécessaire à la fin de l'unique balayage de l'image pour tenir compte des équivalences entre étiquettes, les remises à jours des paramètres sont effectuées directement : les surfaces et les sommes sont simplement additionnées et les extréma sont obtenus par combinaison d'extréma : maximum des maxima et minimum des minima.

La tâche d'identification est constituée de filtres sur attributs des ensembles de composantes connexes, de manière à déterminer les divers éléments de la scène : balle, buts, balises, joueurs. C'est une étape de niveau « intermédiaire », donc rapide, qui ne pose pas de problème d'implantation.

La balle est détectée en premier, c'est l'ensemble de composantes connexes de couleur orange de surface la plus importante. Les coordonnées du centre de gravité donnent la direction, et la surface une estimation grossière de la distance. Nous n'avons pas visualisé d'image où la balle était coupée en plusieurs ensembles de composantes connexes.

Les balises sont détectées ensuite. La projection dans l'image d'une balise comporte un ensemble de composantes connexes roses et un ensemble de composantes connexes soit bleues claires, soit jaunes, soit vertes foncées situé soit au dessus, soit au dessous de l'ensemble de composantes connexes roses. Chacune des 6 combinaisons (rose / vert, rose / jaune, rose / bleu clair, vert / rose, jaune / rose, bleu clair / rose) permet d'identifier une balise, donc une position connue sur le terrain par rapport à laquelle le robot va se repérer.

Ensuite sont détectés les buts. Il faut remarquer que ceux-ci sont rarement visibles en totalité compte tenu du nombre de joueurs sur le terrain. En conséquence, les buts sont les ensembles de composantes connexes jaunes et bleues claires détectées, non identifiées comme balise.

La tâche de localisation, qui consiste à transformer géométriquement les coordonnées en pixel des éléments reconnus, en angle (site et azimut) par rapport à l'axe optique de la caméra, connaissant les paramètres de calibration, ainsi que leurs surfaces en distances, par tabulation, est aussi rapide, et ne pose pas de problème d'implantation.

Si pour la balle et les balises l'estimation de la distance est obtenue par tabulation sur la surface de l'élément identifié dans l'image, pour les buts cette estimation est obtenue sur la hauteur (composante en y) de la boîte englobante. En effet, comme un but est rarement visible en totalité la surface n'est pas un attribut fiable. En revanche, sur les parties visibles, il est généralement possible de distinguer la totalité de la hauteur, ce qui explique pourquoi nous réalisons la tabulation sur la hauteur de la boîte englobante.

Compte tenu de la variation de l'aspect d'un joueur en fonction de l'angle sous lequel il est observé, la distance aux divers joueurs n'est pas estimée. En revanche, pour éviter la collision du robot avec un autre joueur, une alerte est générée si la surface dans l'image du joueur est trop importante.

5. Présentation des Résultats :

Nous avons été les premiers impressionnés par la cadence des traitements embarqués, obtenue par notre robot compte tenu de sa taille. Nous nous étions fixé un minimum de 5 images par seconde à traiter, pour pouvoir contrôler correctement le déplacement du robot par la vision. Deux « benchmarks » différents nous ont donné un résultat identique de 15 images traitées par seconde pendant la marche et la génération de trajectoires simples, c'est-à-dire la cadence d'acquisition de la caméra. Le suivi par la tête d'une balle lancée sur le terrain est parfaitement fluide, que le robot soit arrêté ou en mouvement (rotation sur lui-même ou marche classique).

Il est difficile de juger la qualité des résultats du système de vision autrement que par des critères de comportement du robot. En effet, il est impossible de sortir une image résultat du robot par la liaison série sans le perturber dans son mouvement. Les seuls résultats en image sont issus du simulateur sur PC (cf figure n° 4).

D'après le comportement du robot : il va au but adverse avec la balle, et il tente de dégager la balle lorsqu'il voit également son propre but ; la balle ainsi que les buts sont correctement identifiés et localisés. L'intégration des autres éléments dans la stratégie de jeu est en cours.

L'interface nous a permis de nous adapter rapidement aux conditions d'éclairage qui ont variées au cous de la compétition : panne ou déplacement de projecteurs ... En moins d'un quart d'heure, dans des conditions difficiles en présence de public, il nous était possible d'obtenir des valeurs de seuil correctes.

6. Conclusion :

Bien que le principe du jeu semble relativement simple, il n'est pas facile de faire jouer des robots à roues (small ou middle size), ou à pattes au football. Certes, la qualité de jeu s'améliore d'années en années, mais il reste néanmoins des progrès à faire : la RoboCup a de bonnes années devant elle !

Nous tenons à souligner l'intérêt d'une manipulation réelle qui nous a permis de nous confronter à de nombreux problèmes concrets, du système de vision, mais également d'intégration des différents objets locomotion / stratégie, vision et relevage, pour fonctionner ensemble et assurer au robot mobile et autonome un comportement satisfaisant. Il est rare d'avoir la chance de pouvoir travailler sur une application globale comme celle-ci : du système de vision jusqu'à la commande du robot. Nous soulignons également l'intérêt d'un contexte de compétition qui créé une saine émulation au sein même des

équipes, ainsi qu'un bon contact entre équipes lors de la compétition.

Concernant le système de vision, il est important de définir clairement son rôle, afin de choisir parmi les nombreux algorithmes de traitement d'image proposés dans la littérature ceux qui remplissent correctement la tâche en un temps minimal. Les solutions les plus simples sont généralement les meilleures !

Concernant notre recherche, plus fondamentale, relative à la création d'algorithmes de segmentation multispectrale, et de segmentation en mouvement, le retour de cette expérience sera très bénéfique dans le sens où les problèmes réels précédemment examinés seront bien évidemment pris en compte à des degrés divers. En effet, par exemple, concernant la puissance de calcul embarquée, il sera possible de développer toute une famille d'algorithmes plus ou moins « gourmands » en puissance de calcul, à utiliser en fonction de la cible (hardware spécifique, ou processeur central) disponible sur le robot. De plus les algorithmes utilisés par l'interface permettant de déterminer les seuils nécessaires au matériel spécifique de détection des couleurs sont employés dans un contexte hors temps réel. L'automatisation de la procédure manuelle de recherche de zone de l'image correspondant aux divers éléments pourra être réalisée par une segmentation couleur vectorielle (ou multispectrale).

Une autre voie de recherche prometteuse est d'inclure de « l' Intelligence » dans l'objet Vision. Cette nouvelle stratégie, en cours de conception et de réalisation sera testée à Stockholm en Août prochain.

7. Références :

[1] M. Asada ,
Proceedings of the second RoboCup Workshop, RoboCup-98 : Robot Soccer World Cup II,
Paris Cité des Sciences et de l'Industrie Juillet 98

[2] M. Fujita, S. Zrehen, H.Kitano,
A Quadruped Robot for RoboCup Legged Robot Challenge,
Proceedings of the second RoboCup Workshop, Paris Juillet 98

[3] P.Bonnin, B.Hoeltzener, E.Pissaloux,
A new way of image data fusion : the multi-spectral cooperative image segmentation,
IEEE Int Conf on Image Processing ICIP 95 Washington

[4] P.Bonnin, B.Hoeltzener, E.Pissaloux,
A Data Parallel implementation of an edge point chaining : Towards a new principle of Edge Linking ,
IEEE Int Conf on Image Processing ICIP 96 Lausanne.

[5] Thinking Machine Corporation,
C* Programming Guide, Cambridge 1993

[6] P. Bonnin,
Méthode systématique de conception et réalisation d'applications en Vision par Ordinateur,
Thèse Paris VII, déc 91

[7] M. Mangolini,
Apport de la Fusion d'images satellitaires multicapteur au niveau pixel en télédétection et photo-interprétation,
Thèse de l'Université de Nice Sophia Antipolis 1995

[8] E.Pissaloux, P.Bonnin,
On the evolution of parallel computers dedicated to image processing through examples of french computers, Digital Signal Processing, Academic Press, vol 7, n°1, January 1997

[9] L. Kuntz-Sliwa,
Optimisation d'une Configuration Multicapteurs donnée - Fusion Pixel,
Thèse de l'Institut National Polytechnique de Toulouse, Février 1996.

[10] A. Rosenfeld, JL Pfalz,
Sequential operations in digital picture processing,
Journal of ACM, vol 13, n°4 1966

8. Remerciements :

Toute l'équipe des « Titis Parisiens » du Laboratoire de Robotique de Parie remercie l'équipe du Laboratoire D21 de SONY pour leur confiance (la sélection à la Coupe SONY), pour la mise à disposition des robots, pour le support technique fourni, et pour l'ambiance chaleureuse de la semaine passée à la Cité des Sciences de La Vilette en juillet dernier.

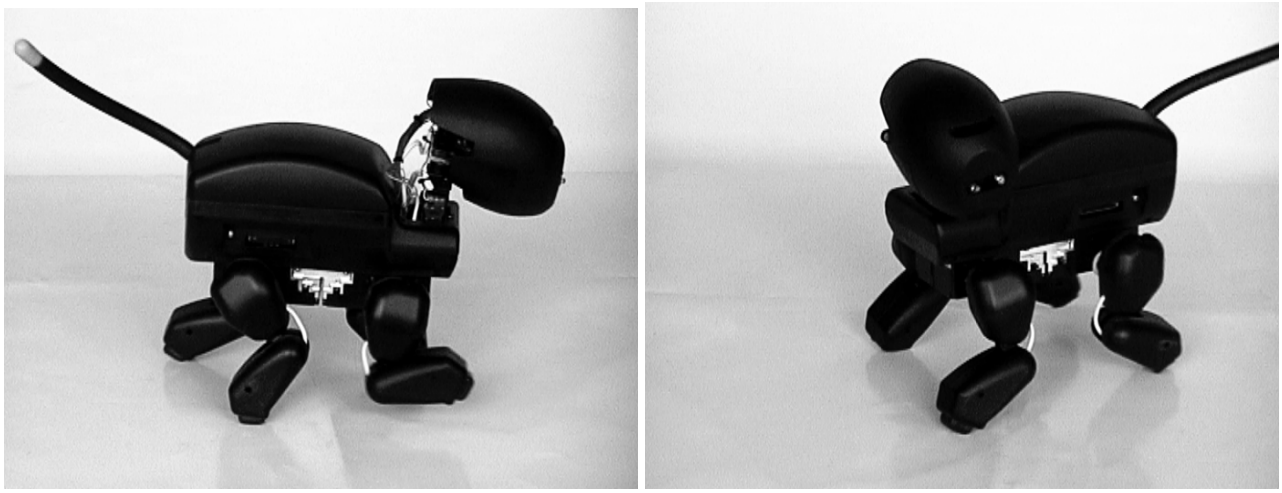

Figure n° 1 : Robot en Marche Rectiligne et Rotation sur lui même

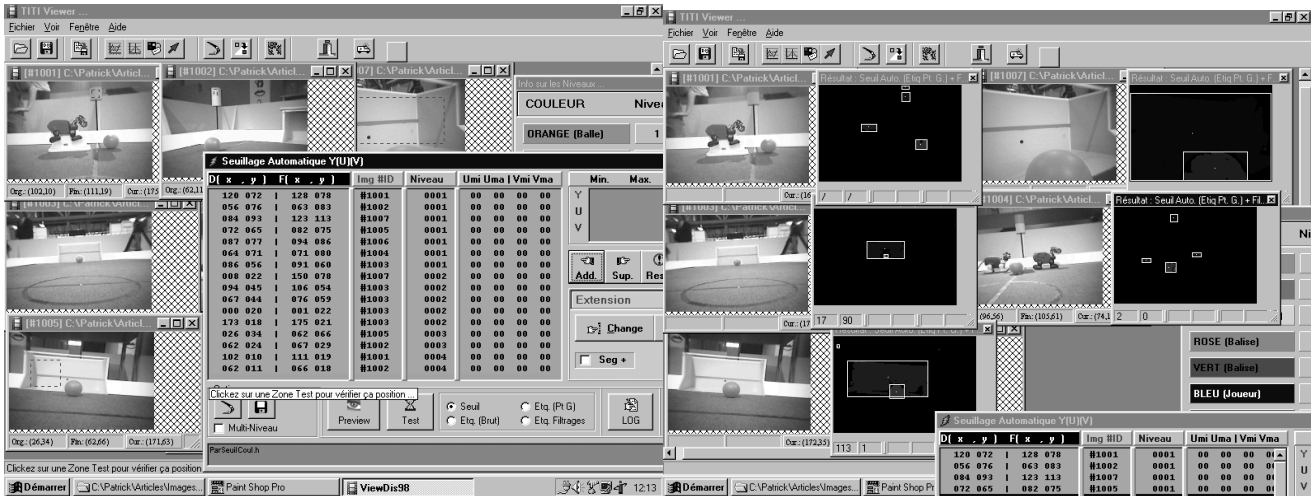

Figure n° 2 : L'interface Conviviale de Réglages : Sélection des zones de test, et visualisation des résultats

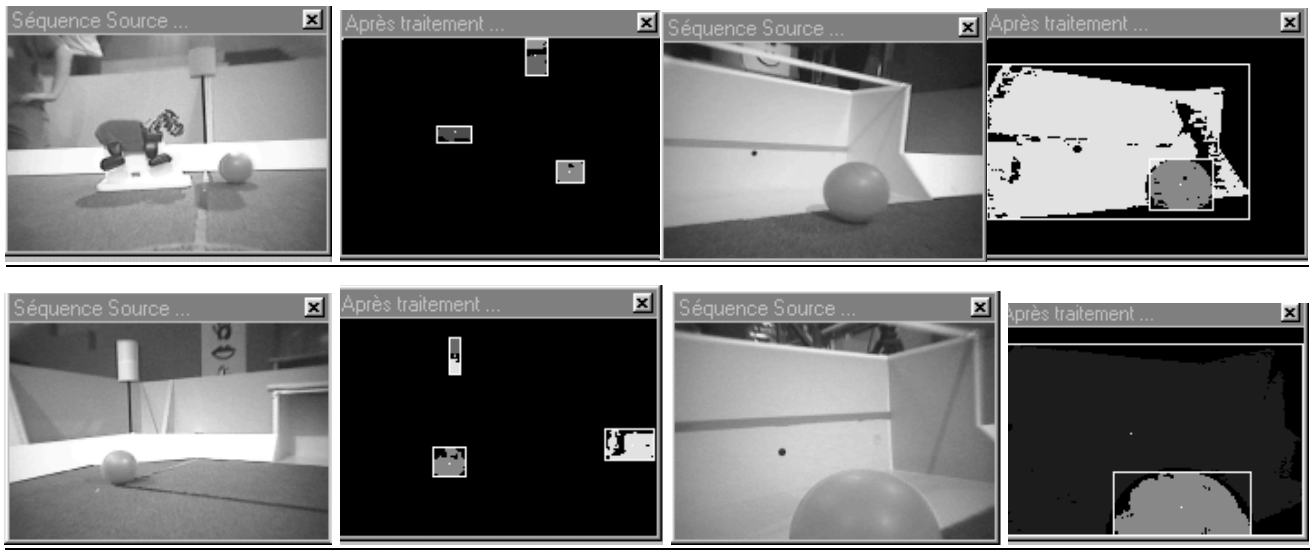

Figure n° 4 : Quelques Images vues du Robot, et leurs Résultats de Traitement sur Simulateur