Parallélisation du suivi de cible pour la robotique mobile[1]

Fabrice Chantemargue et Heinz Hügli

CERN, ECP Division
Bâtiment 14, Bureau 3-022
CH-1211 Genève 23
chante@sunrans.cern.ch
et
Institut de microtechnique
Université de Neuchâtel
Rue de Tivoli 28
CH-2003 Neuchâtel
hugli@imt.unine.ch

RÉSUMÉ

Cet article propose la parallélisation d'une technique de suivi de cibles dans le cadre d'une application de vision pour la robotique mobile, selon les deux aspects suivants: gain en temps d'exécution et amélioration de la technique. Après avoir décrit l'algorithme séquentiel, nous proposons sa parallélisation. Des résultats expérimentaux obtenus sur une architecture à deux transputers sont présentés. Des extrapolations sont tirées de ces résultats afin d'en déduire un nombre maximal de transputers utiles pour la parallélisaton de cet algorithme, en vue de l'implantation sur une architecture plus riche en nombre de processeurs.

ABSTRACT

In this paper, we present the parallelisation of a target tracking technique used for vision in mobile robotics. Two aspects are considered: improvement of computing performance and improvement of tracking robustness. Following a resume of the sequential algorithm, we present the parallelisation of its two main components, namely blob coloring and blob tracking. Finally, we present the practical results obtained with a system of two transputers, and extrapolate in order to find the potential and practical limits of parallelism in a specific architecture with many processors.

1 Ce travail a été effectué à l'Institut de microtechnique de l'Université de Neuchâtel dans le cadre du projet 4023-027037 du Programme National de Recherche 23 de la Confédération suisse

I - INTRODUCTION

Ce travail s'inscrit dans le thème général de la robotique mobile, et est partie intégrante de l'étude d'une approche comportementale d'un robot mobile autonome disposant de capteurs de vision et de proximité. Plus particulièrement, ce travail traite de la parallélisation d'une technique de suivi de cibles dans une séquence d'images, technique revêtant une grande importance en traitement d'images, et par conséquent fréquemment utilisée.

Le cadre expérimental de notre application est un environnement réel non modélisé dans lequel se trouvent des balises réfléchissantes sur des sites d'intérêt. Une des tâches devant être accomplies par le système de vision du robot comprenant une source de lumière et une caméra vidéo, consiste à détecter les images des balises (cibles ou amers), puis à les identifier et les suivre dans la séquence d'images : cette tâche est appelée, de manière générale, suivi de cibles. Cette tâche de vision peut ensuite être utilisée pour la réalisation des comportements du robot (tâches de niveau supérieur). Par exemple, un comportement «aller vers» consistera à faire se déplacer le robot vers une des balises selon une stratégie définie [FAC-92]. En robotique mobile, les applications potentielles sont nombreuses: rangement automatique d'objets, distribution automatique de courrier, ...

La technique de suivi de cibles a déjà fait l'objet de nombreuses études et a été

Parallélisation du suivi de cible pour la robotique mobile

implantée sur un ordinateur séquentiel précédemment [HUG-92], [FAC-92]. De cette étude, il est ressorti une fréquence de traitement prohibitivement basse. Ainsi, le but de l'étude de l'implantation parallèle sur un réseau de transputers est d'une part, d'améliorer les performances de calcul (abaissement du temps de traitement) et d'autre part, d'améliorer éventuellement la technique.

Cet article est structuré de la manière suivante. Après un rappel du contexte général du parallélisme en traitement d'images et une présentation de l'architecture matérielle pour notre application, nous réservons un paragraphe aux fondements de la technique de suivi de cibles (algorithme séquentiel) avant de traiter, dans les deux parties suivantes, deux implantations parallèles, de finalité identique, mais de sémantique différente. Des résultats expérimentaux permettent de conclure sur cette étude.

D'une manière tout à fait générale, on peut ajouter que le problème posé est un exemple représentatif qui illustre les tâches que doit résoudre l'ingénieur pour paralléliser un algorithme.

II - PARALLÉLISME EN TRAITEMENT D'IMAGES

Les parallélisations de nombreux algorithmes de traitement d'images ont déjà été étudiées dans le contexte des réseaux de transputers [CHA-91], [AKI-92], [CHAS-92]. De manière générale, on peut regrouper ces algorithmes en trois classes:

- algorithmes de type pixel ou bloc sans recouvrement (niveau iconique),
- algorithmes de type bloc avec recouvrement (niveau iconique),
- algorithmes globaux (niveaux intermédiaire et/ou haut).

Les algorithmes de type pixel ou bloc sans recouvrement se composent de trois étapes:

- découpage des données à traiter en paquets,
- traitement des paquets (envoi paquet par paquet au réseau de transputers et traitement d'un paquet par un transputer),
- rangement des paquets traités par les transputers, puis tri des paquets pour restituer une structure de données en sortie cohérente.

Souvent, chaque processeur applique le même traitement sur des données différentes. On peut citer comme exemple le seuillage.

Dans les algorithmes de type bloc avec recouvrement, un pixel est traité en fonction de son voisinage: opérations de filtrage, convolution, morphologie mathématique. Pour cette sorte d'algorithme, les processus de traitement ne sont plus indépendants: en effet, durant le traitement, des échanges de données entre les processus sont nécessaires.

Pour les algorithmes globaux (segmentation, coloriage, ...), il n'existe pas de modèle général pour paralléliser.

La parallélisation dépend des structures de données à traiter, des structures de données de sortie, de la nature du traitement.

On ne parallélise pas par plaisir, mais dans un but précis: cela peut être pour abaisser les temps de calcul, pour augmenter la robustesse, pour avoir une meilleure tolérance aux pannes, ... Divers critères, plus ou moins subjectifs, plus ou moins qualitatifs et/ou quantitatifs, peuvent être trouvés dans la littérature, quant à la caractérisation d'une parallélisation, suivant l'un ou l'autre des buts recherchés. Cependant, en ce qui concerne le but le plus couramment poursuivi, c'est-à-dire l'amélioration des performances de calcul d'un algorithme sur une machine parallèle dédiée, deux critères objectifs sont universellement reconnus et utilisés, à savoir l'accélération A et l'efficacité E, définis ci-après:

$$A = \frac{\text{Temps seq}}{\text{Temps par}} \qquad E = \frac{A}{P}$$

P représentant le nombre de processeurs de la machine parallèle. Le temps séquentiel (Temps seq) doit être le meilleur temps obtenu pour l'exécution d'un algorithme donné avec un seul processeur. Le temps parallèle (Temps par) est le temps obtenu pour l'exécution du même algorithme avec P processeurs.

Le cas idéal pour une parallélisation, du point de vue des performances, est obtenu lorsque l'accélération vaut P (efficacité de 100%). En réalité, cette valeur n'est jamais atteinte à cause:

- du coût de chargement des structures de données à traiter, ainsi que du coût de rapatriement des structures de données résultats;
- du coût des communications induites par le traitement, lui-même engendré par la parallélisation. Par exemple, en ce qui concerne le transputer, son nombre de liens de communication limité à quatre impose la prise en compte de la notion de localité [QUI-89]. Ainsi, il est nécessaire de faire transiter les structures de données par des processeurs intermédiaires. Cela se traduit par des interruptions d'exécution de processus (le temps de traiter le message).

Ainsi, lors de l'étude de la parallélisation d'un algorithme, l'aspect communications entre les processeurs ne doit pas être négligé. Ceci peut souvent remettre en cause une décomposition algorithmique trop fine, au sens du grain, lors de la réalisation du graphe de dépendance de tâches.

III - ARCHITECTURE MATÉRIELLE

La figure 1 illustre l'architecture matérielle à notre disposition. Un robot mobile, fourni par la société Nomadic [NOM-92] et possédant des détecteurs de collision ainsi que des capteurs infrarouges, a été complété par plusieurs systèmes de vision. Ses déplacements et ses capteurs sont contrôlés par une station de travail Sun, au moyen d'une communication par radiofréquence [HUG-93]. Pour l'application décrite dans cet article, nous nous intéressons au système

Parallélisation du suivi de cible pour la robotique mobile

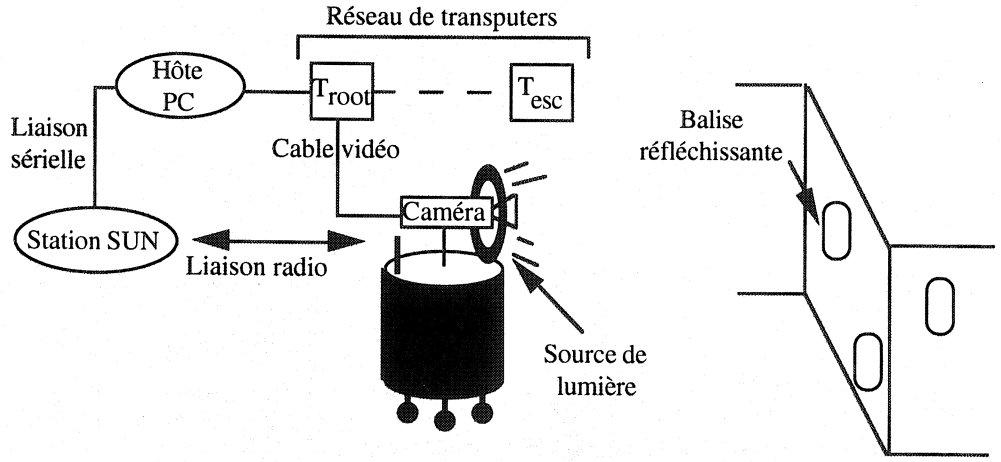


Figure 1: Architecture matérielle.

de vision combinant caméra et source de lumière et dont l'image est analysée par le réseau de transputers

Un PC joue le rôle d'ordinateur hôte et comprend actuellement deux cartes mères, support physique du réseau de transputers: une carte mère (type TBX10D), pouvant recevoir jusqu'à dix modules transputers TRAM (TRAnsputer Module) de taille 1 et une carte mère (type TBX05) pouvant accueillir jusqu'à cinq modules TRAM de taille 1. La carte TBX10D comprend un module TRAM de taille 8 dédié à l'acquisition d'images (pilotage de la caméra du couple de capteurs caméra/lumière fluorescente). Ce module se compose d'un transputer T800 possédant 2M octets de VRAM (mémoire vidéo) et 4M octets de RAM. Par définition, ce transputer désigne le transputer maître (Troot) du réseau. Les autres modules TRAM (sur les cartes TBX10D et TBX05) sont de taille 1 et comprennent un T800 avec 4M octets (ou 1M octets) de RAM. Tous les transputers T800 des modules TRAM de taille 1 jouent le rôle de transputers esclaves (Tesc); ils sont donc contrôlés par le transputer maître (Troot). De plus amples détails concernant ces cartes peuvent être trouvés dans les rapports techniques [TBX10-91], [TBX05-92], [IP1-92].

Les «comportements» du robot (tâches de haut niveau, introduites au paragraphe 1) sont logiquement implantés sur la station de travail SUN. Ainsi, une communication entre cette station et l'ordinateur hôte s'avère nécessaire, afin que les résultats des traitements effectués sur le réseau de transputers soient connus du «comportement». Compte tenu d'une faible vitesse de déplacement du robot (contraintes mécaniques) et d'un volume d'information à transmettre peu important (quelques dizaines d'octets au maximum), il a été décidé que cette communication s'effectue au moyen d'une liaison sérielle de type RS 232.

IV - TECHNIQUE DE SUIVI DE CIBLES

L'intérêt d'utiliser une source de lumière et une caméra conjointement à des balises réfléchissantes est que l'on simplifie considérablement le traitement des images. En effet, la détection des cibles est facilitée et peut se faire à partir d'une séquence d'images binarisées: ainsi tous les pixels éclairés d'une image appartiennent à des cibles.

Plusieurs mesures ont été prises pour faire que les pixels éclairés dans une image correspondent bien à la projection de balises, et non à la projection d'objets parasites réfléchissants se trouvant dans la scène. La première mesure, appliquée en amont de tout traitement, tire avantage de la propriété du matériau constituant les balises de réfléchir de manière préférentielle la lumière en provenance de la source vers la caméra qui lui est proche. Un seuillage sélectif appliqué sur l'image permet ainsi d'éliminer de nombreuses cibles parasites. Une deuxième mesure est appliquée en aval du traitement, c'est-à-dire une fois que l'on a détecté dans l'image toutes les cibles présentes. Elle stipule qu'une cible soit acceptée comme balise à la seule condition de satisfaire un double critère de forme et de taille. Concernant la forme, celle des balises étant circulaire, le critère permet d'éliminer grand nombre de cibles parasites qui correspondent à des images de surfaces réfléchissantes diverses (pied de table métallique, arête d'une armoire métallique, ...) et ont généralement une forme plus allongée. Concernant la taille, le critère se base sur une taille acceptable exprimée par un nombre de pixels compris entre un minimum et maximum: il rejette les cibles de taille sensiblement différente des balises. La conjugaison de toutes ces mesures permet de minimiser le nombre de détections parasites, sans pour autant accroître la charge algorithmique.

Notons encore qu'en cas de non détection de cibles dans l'image, un ordre de rotation est transmis au robot, afin que la caméra puisse acquérir l'image d'un autre secteur de l'environnement.

D'un point de vue algorithmique, la technique de suivi de cibles peut être décomposée en deux tâches exécutées séquentiellement [FAC-92]:

- une tâche d'initialisation appelée encore étiquetage de composantes connexes ou coloriage: à chaque cible de l'image repérée par son centre de masse, est attribuée une étiquette différente.

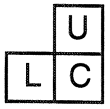
- une tâche de suivi de cibles dans la séquence d'images: le principe se base sur une méthode de prédiction du lieu de présence d'une cible à partir de son histoire dans la séquence d'images.

IV 1. Phase de Coloriage

L'algorithme de coloriage retenu est l'algorithme de coloriage de bulles proposé dans [BAL-82]. Etant donnée une image binaire contenant des régions connexes (en 4-connexité) composées de pixels blancs (niveau_cible) sur un fond noir (niveau_fond), le but est d'étiqueter chaque région différemment. Pour cela, il

Parallélisation du suivi de cible pour la robotique mobile

est nécessaire de scruter l'image entière avec un masque comme défini ci-après:



où C représente le point central, U le point voisin de dessus, L celui de gauche.

L'algorithme de coloriage est décrit ci-après :

k = 1 /* initialisation du numero d'etiquette pour la table color */

l = 1 /* initialisation du numero d'etiquette pour la table color_equi*/

Pour tous les pixels de l'image :

```
Si f(Xc) = niveau_fond
  alors rien
Sinon
  Si ( f(Xu) = niveau_cible et f(XL) = niveau_fond)
    color(Xc) = color(Xu)
  Si ( f(XL) = niveau_cible et f(Xu) = niveau_fond )
    color(Xc) = color(XL)
  Si ( f(XL) = niveau_cible et f(Xu)= niveau_cible)
    color(Xc) = color(XL)
    /* color(XL) est équivalente à color(Xu) */
    equivalence (color(XL), color(Xu))
  Si ( f(XL) = niveau_fond et f(Xu) = niveau_fond)
    color(Xc) = k
    color_equi[k] = l
    k = k + 1
    l = l + 1
  Fin Si
Fin Sinon
Fin pour
```

où:

f(x): image binaire à étiqueter,
color(x): image des étiquettes,
color_equi[k]: table d'équivalence des couleurs,
f(Xc): niveau binaire du point X coincidant avec le point C du masque,
f(Xu), f(XL): idem avec les points U et L du masque

La table d'équivalence des couleurs est obtenue avec la procédure équivalence qui est définie comme suit:

```
/* construction de la table d'équivalence */
Si color_equi[color1] ≠ color_equi[color2]
  Si color_equi[color1] > color_equi[color2]
    clarge = color_equi[color1]
    csmall = color_equi[color2]
  sinon
    clarge = color_equi[color2]
    csmall = color_equi[color1]
  Pour les points n de la table (1 à k)
    Si color_equi[n] = clarge
      color_equi[n] = csmall
    Si color_equi[n] > clarge
      color_equi[n] = color_equi[n]-1
    Fin Si
  Fin Pour
  l = l-1
Fin Si
```

Après un balayage complet de l'image, on obtient un étiquetage partiel de l'image (une même région peut posséder plusieurs étiquettes différentes) et une table d'équivalence de couleurs (color_equi).

Un deuxième balayage de l'image est alors nécessaire afin d'obtenir un étiquetage complet de l'image, c'est-à-dire d'assurer qu'à chaque ensemble connexe de pixels ne soit attribuée qu'une seule étiquette. Ceci est réalisé grâce à la table d'équivalence de couleurs.

Lors des deux balayages successifs de l'image, les attributs caractéristiques (nombre de points, ...) de chaque cible, nécessaires au calcul des centres de masse, sont déterminés. Par conséquent, à la fin de la phase de coloriage, on dispose du nombre de cibles présentes dans l'image ainsi que de leurs barycentres.

Parallélisation du suivi de cible pour la robotique mobile

IV 2. Phase de suivi

La phase de suivi de cibles est basée sur une méthode de prédiction du lieu de présence d'une cible à partir de son histoire dans la séquence d'images, en partant de l'hypothèse selon laquelle les déplacements des cibles sont soumis à une accélération nulle d'une image à l'autre dans la séquence [FAC_92]. Ainsi, dès que l'on possède tous les barycentres des cibles présentes dans l'image à l'issue de la phase de coloriage, on prédit les positions de ces barycentres dans l'image suivante, à partir des positions précédentes tel qu'indiqué ci-après :

$$\hat{P}^{n+1} = \begin{pmatrix} \hat{X}^{n+1} \\ \hat{Y}^{n+1} \end{pmatrix} = \begin{pmatrix} \frac{5}{2}X^{n} - 2X^{n-1} + \frac{1}{2}X^{n-2} \\ \frac{5}{2}Y^{n} - 2Y^{n-1} + \frac{1}{2}Y^{n-2} \end{pmatrix}$$

avec $\begin{pmatrix} \hat{X}^{n+1} \\ \hat{Y}^{n+1} \end{pmatrix}$ les coordonnées prédites dans l'image $n+1$

et $\begin{pmatrix} X^{n} \\ Y^{n} \end{pmatrix}, \begin{pmatrix} X^{n-1} \\ Y^{n-1} \end{pmatrix}, \begin{pmatrix} X^{n-2} \\ Y^{n-2} \end{pmatrix}$ les coordonnées calculées dans les images n, n - 1, n - 2

Le principe de la phase de suivi consiste à centrer des fenêtres de recherche de taille variable et de forme rectangulaire sur les positions prédites des barycentres des cibles. La taille d'une fenêtre est bien sûr fonction de l'évolution de la taille d'une cible au cours de la séquence. Par définition, on considère que tous les points éclairés à l'intérieur d'une fenêtre de recherche appartiennent à une même cible. Le calcul des coordonnées du barycentre d'une cible est par conséquent très rapide, et le suivi des cibles dans la séquence est ainsi assuré.

V - PARALLÉLISATION

L'application de suivi de cibles est composée de deux tâches séquentielles: une tâche de coloriage et une tâche de suivi. Par conséquent, le parallélisme n'intervient qu'à l'intérieur de chacune des tâches. La technique de suivi de cibles appartient à la classe des traitements globaux: compte tenu de ses caractéristiques, cette technique se prête bien à un parallélisme suivant le modèle ferme de processeurs.

V 1. Phase de Coloriage

La parallélisation de la phase de coloriage se compose de trois étapes sérielles, dont le principe est exposé ci-après:

- étape 1 : le maître découpe l'image en N secteurs de manière optimale, puis répartit les N secteurs sur les N processeurs esclaves.
- étape 2 : sur un processeur esclave, chaque secteur est perçu comme une image à part entière. Ainsi, à chaque secteur est appliqué l'algorithme de coloriage tel que décrit dans le paragraphe 4.
- étape 3 : le maître reçoit les résultats des traitements accomplis par les N processeurs afin de construire un résultat cohérent sur l'image: pour notre application, il s'agit de l'estimation des centres de masse de chaque cible de l'image à partir de ses attributs caractéristiques.

Le gain apporté par le parallélisme se situe dans l'étape 2. En effet, les processeurs esclaves traitent les secteurs en parallèle. Cependant, le fait d'avoir distribué l'image à traiter sur le réseau de processeurs induit forcément des communications inter-processeurs ainsi qu'un surplus de traitement.

On peut représenter le déroulement de l'algorithme parallèle sous la forme suivante :

```
Faire séquentiellement
  /* étape 1 */
    découpage de l'image en secteurs, puis distribution.
  /* étape 2 */
    Faire en parallèle
      Traitement des secteurs :
        réception d'un secteur.
        coloriage dans le secteur.
        émission du résultat du traitement.
    Fin faire en parallèle
  /*étape 3 */
    regroupement et traitement final.
Fin faire séquentiellement
```

Pour que le temps de traitement d'une image sur une machine multiprocesseurs converge vers le temps de traitement d'un secteur (objectif fixé), il faut d'une part, que les étapes 1 et 3 soient exécutées le plus brièvement possible et d'autre part, que le volume de communications inter-processeurs soit le plus compact possible, sans pour autant accroître de manière significative le traitement nécessaire à la gestion de ces communications.

V *1.1. Découpage de l'image en secteurs (étape 1)*

Le choix dans le découpage de l'image en secteurs est important car il influe directement sur le volume des communications dans le sens processeurs esclaves - processeur maître et également sur la nature de l'algorithme de l'étape 3. Ainsi, il est judicieux de découper l'image en secteurs S0, S1 ..., SN-1 se chevauchant d'une colonne, appelée

colonne frontière (voir figure 2): cela permet notamment de prendre en compte de manière implicite l'adjacence de pixels appartenant à une cible s'étalant sur deux secteurs.

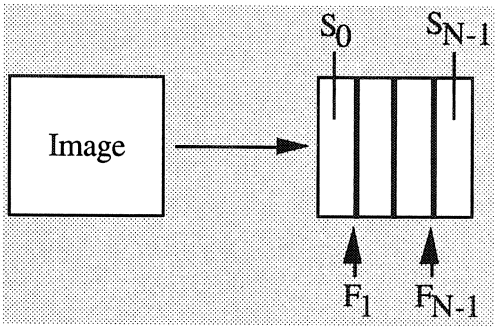


Figure 2: Découpage de l'image en secteurs.

Une colonne Fi à l'intersection de deux secteurs adjacents, sera traitée par les deux tâches relatives aux deux secteurs Si-1 et Si. Ainsi un pixel d'une colonne frontière appartenant à une cible aura une étiquette attribuée par une première tâche et une autre étiquette attribuée par une seconde tâche.

V *1.2. Traitement des secteurs (étape 2)*

La tâche Ti relative à un secteur Si effectue le coloriage sur le secteur (détection de toutes les cibles présentes) et elle caractérise chacune des cibles par des attributs propres à l'application. Le coloriage s'applique à l'ensemble du secteur, alors que le calcul des attributs, par convention, est réalisé en excluant la colonne frontière de droite Fi+1 . En effet, la tâche Ti+1 relative au secteur suivant Si+1 prendra en compte cette information de manière implicite (lors du traitement de sa colonne frontière de gauche).

Ainsi, la tâche Ti fournit en sortie les paramètres suivants[1]:

- le nombre de cibles dans le secteur Si, ainsi qu'une liste d'étiquettes des cibles.
- une liste d'attributs pour chaque cible (les indices des sommes portent sur tous les points de la cible):

  X_Σ, Y_Σ, N_Σ, étiquette

  (X_Σ et Y_Σ représentent les sommes des abscisses respectivement des ordonnées d'une cible et N_Σ son nombre de points).
- une liste appelée Ldi: elle contient les étiquettes des cibles sur la colonne frontière Fi+1 (colonne frontière droite du secteur Si).
- une liste appelée Lgi: elle contient les étiquettes des cibles sur la colonne frontière Fi (colonne frontière gauche du secteur Si).

Le remplissage des listes Lgi et Ldi peut se faire soit en parcourant la colonne frontière de haut en bas ou de bas en haut, mais il importe que le sens retenu soit le même pour les deux listes, autrement la notion d'équivalence entre les régions disparaît.

1 La tâche T0 relative au secteur S0 ne s'occupe pas de la liste Lg0. De manière analogue, la tâche TN-1 relative au secteur SN-1 ne s'occupe pas de la liste LdN-1.

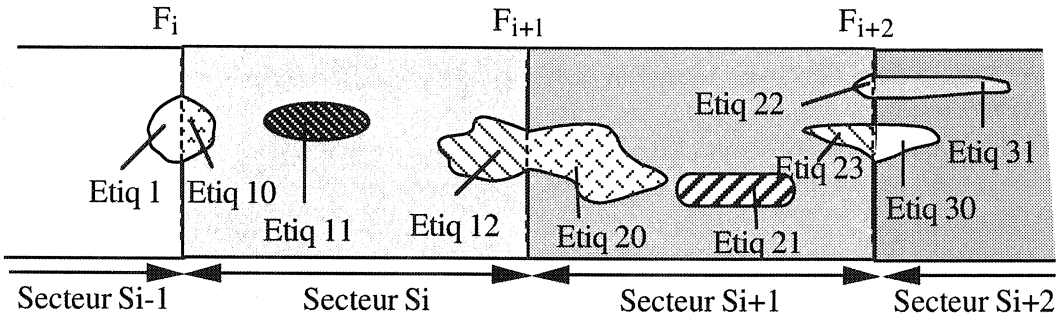


Figure 3: Cas typique d'image contenant des cibles.

La figure 3 illustre un cas classique de cibles à cheval sur plusieurs secteurs.

Pour chaque secteur, une étiquette de base différente est fixée, ceci afin d'éviter les superpositions d'étiquettes entre secteurs. Les figures 4, 5 et 6 présentent respectivement le remplissage des listes Lgi, Ldi, Lgi+1, Ldi+1, Lgi+2 et Ldi+2 pour les secteurs Si, Si+1 et Si+2, dans le cadre de l'exemple de la figure 3 (X signifie indifférent).

L_{gi}	L_{di}
10	12

Figure 4: Remplissage des listes Lgi et Ldi pour le secteur Si.

L_{gi+1}	L_{di+1}
20	22
	23

Figure 5: Remplissage des listes Lgi+1 et Ldi+1 pour le secteur Si+1.

L_{gi+2}	L_{di+2}
31	X
30	X

Figure 6: Remplissage des listes Lgi+2 et Ldi+2 pour le secteur Si+2.

V *1.3 Regroupement et traitement final (étape 3)*

Le but de cette étape est de fusionner les cibles à cheval sur les colonnes frontières. Pour cela, il faut parcourir les listes de chaque colonne frontière: ceci fournit une table d'équivalence de couleurs à la frontière de deux secteurs, de manière analogue à celle que l'on construit à l'intérieur d'un secteur. La scrutation des listes des colonnes frontières est effectuée selon un ordre strict, dicté par les indices des frontières Fi, c'est-à-dire en commençant par la frontière F0 et en terminant par la frontière FN-1 (N est le nombre de secteurs).

La figure 7 illustre la manière de traiter la

Parallélisation du suivi de cible pour la robotique mobile

colonne frontière Fi+1, à partir de l'exemple de la figure 3. Cette colonne ne peut être analysée qu'après réception des résultats issus des traitements des secteurs Si et Si+1. La colonne Fi+1 est caractérisée par les listes Ldi et Lgi+1 de la figure 7.

L_{di}	L_{gi+1}
12	20

Colonne F_{i+1}.

L_{di+1}	L_{gi+2}
22	31
23	30

Colonne F_{i+2}.

Figure 7: Gestion des colonnes frontières

Ainsi, on en déduit aisément, après balayage des listes Ldi et Lgi+1 que dans l'exemple donné, les étiquettes 12 et 20 sont équivalentes. En d'autres termes, cela signifie que les régions étiquetées 12 et 20 ne forment qu'une seule et même région que l'on renomme par convention par l'étiquette de valeur la plus faible (c'est-à-dire étiquette 12). Dans le cadre de notre application, la relation «étiquette 12 est équivalente à étiquette 20» se traduit simplement par la mise à jour des attributs de la manière suivante (les sommes portent sur tous les points de la cible):

$X_\Sigma 12 = X_\Sigma 12 + X_\Sigma 20$

$Y_\Sigma 12 = Y_\Sigma 12 + Y_\Sigma 20$

$N_\Sigma 12 = N_\Sigma 12 + N_\Sigma 20$

On n'a donc pas besoin d'étiqueter complètement l'image: on peut disposer des attributs nécessaires au calcul des centres de masse des cibles avec seulement un étiquetage partiel de l'image.

La figure 7 illustre aussi la façon de traiter la colonne frontière Fi+2, à partir de l'exemple de la figure 3. Cette colonne ne peut être analysée qu'après réception des résultats issus des traitements des secteurs Si+1 et Si+2. La colonne Fi+2 est caractérisée par les listes Ldi +1 et Lgi+2 présentées ci-après:

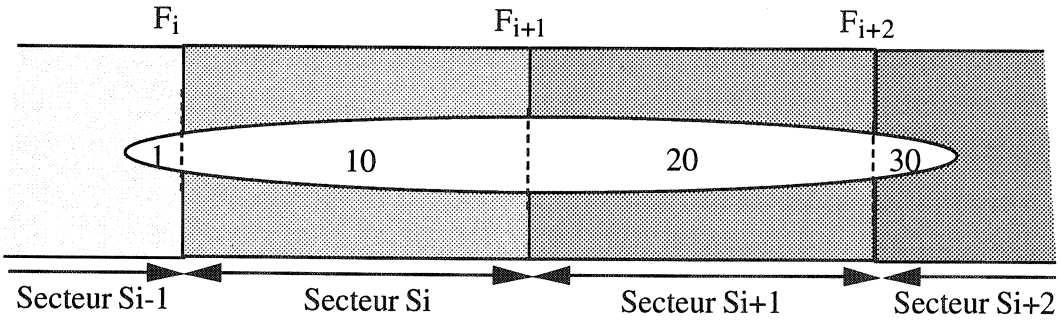


Figure 8: Cible s'étalant sur plusieurs secteurs et étiquettes attribuées dans chaque secteur.

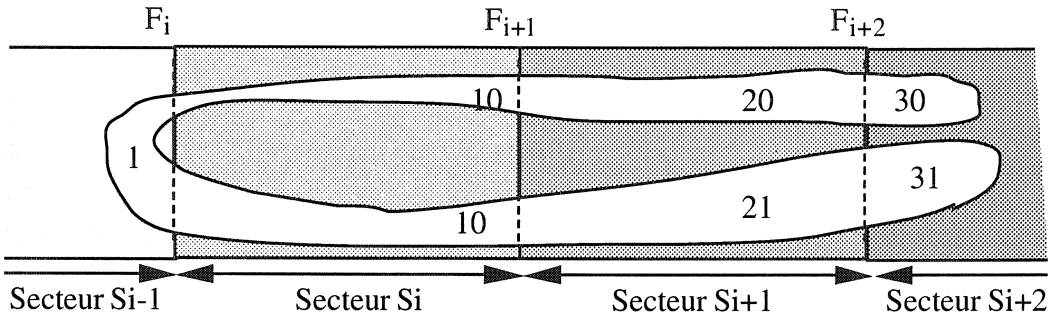


Figure 9: Cible bruitée s'étalant sur plusieurs secteurs.

Après balayage des listes Ldi+1 et Lgi+2, on peut en déduire que les étiquettes 22 et 31 sont équivalentes, ainsi que 23 et 30. La mise à jour des attributs des nouvelles régions se fait comme indiqué précédemment pour la gestion de la colonne frontière Fi+1.

Dans le principe de gestion d'une colonne frontière qui est exposé ci-dessus, des précautions supplémentaires sont néanmoins à prendre pour des cas de cible s'étalant sur plusieurs secteurs. La figure 8 présente un exemple typique.

On rappelle que la scrutation des listes des colonnes frontières est effectuée selon un ordre strict, dicté par les indices des frontières Fi, c'est-à-dire en commençant par la frontière F0 et en terminant par la frontière FN-1 (N étant le nombre de secteurs). Ainsi, l'analyse des colonnes frontières indiquera que les régions 1 et 10, 10 et 20, 20 et 30 sont équivalentes. Le nombre de cibles présentes dans l'image (une seule) peut être aisément connu (nombre par secteur moins le nombre d'équivalence). Par contre, on ne possédera des attributs de région corrects que si l'on remplace, en fin d'étude de la colonne frontière Fi, la région 10 présente dans la liste Ldi de la colonne frontière Fi+1, par la région 1. On procède de manière analogue pour toutes les colonnes. La propagation des attributs est ainsi assurée.

La figure 9 présente un cas moins trivial de cible bruitée s'étalant sur plusieurs secteurs.

En procédant de la même manière qu'indiquée précédemment, les listes Ld et Lg relatives aux différents secteurs sont remplies comme indiqué dans la figure 10.

L_{di-1}	L_{gi}
1	10

L_{di}	L_{gi+1}
10	20
10	21

L_{di+1}	L_{gi+2}
20	30
21	31

Figure 10: Gestion des colonnes frontières

Parallélisation du suivi de cible pour la robotique mobile

L'étude de la colonne frontière Fi indique que les régions 1 et 10 sont équivalentes. Après mise à jour des attributs de la nouvelle région 1 (union de 1 et 10), il est nécessaire de remplacer, en fin d'étude de la colonne frontière Fi, la région 10 présente dans la liste Ldi de la colonne frontière Fi+1, par la région 1. Ainsi, l'étude de la colonne frontière Fi+1 indiquera que les régions 1 et 20 puis 1 et 21 ne font qu'une seule région. La procédure est maintenant classique: mise à jour des attributs de la région 1, et remplacement des régions 20 et 21 présentes dans la liste Ldi+1 par la région 1. Enfin, l'étude de la colonne frontière Fi+2 permettra, selon la même méthode de fusionner la région 1 avec les régions 30 et 31 et de posséder des attributs corrects pour la région finale.

V 2. Phase de suivi

La parallélisation de la phase de suivi est immédiate, compte tenu de l'indépendance au niveau des données à traiter. Le principe consiste à fournir à chaque processeur une fenêtre ou un ensemble de fenêtres, afin que celui-ci s'en occupe. La figure 11 illustre cette application dans le cas d'une architecture à deux transputers pour une image contenant six cibles après la phase de coloriage, donc six fenêtres.

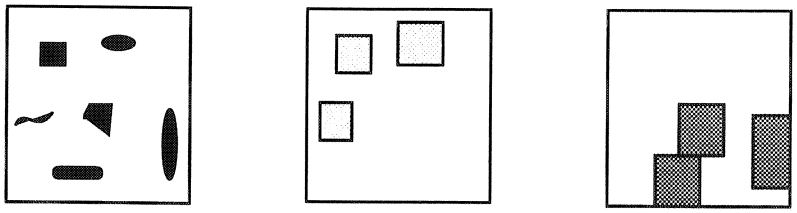


Figure 11: Parallélisation de la phase de suivi sur deux transputers.

V 3. Résultats expérimentaux

Les figures 12 et 13 présentent les gains en temps apportés par la parallélisation. Ces mesures ont été obtenues pour des programmes écrits en langage C parallèle de chez C3L, sur une architecture composée de deux transputers.

La figure 12 présente le gain en temps de calcul apporté par la parallélisation dans le cas d'une image de taille 440x460 pixels contenant quatre cibles, ainsi que les accélérations et efficacités de parallélisation.

Les performances obtenues par la parallélisation sont très satisfaisantes (efficacité supérieure à 80 %). Cependant, pour diminuer les temps de traitement, il est possible de sous-échantillonner l'image afin de réduire la quantité de données à traiter. La figure 13 présente les mesures obtenues dans les mêmes conditions que pour la figure 12, mais en faisant varier la résolution[2].

2 Une résolution de 1:4 signifie que l'on ne prend qu'un point image sur 4 pour chaque dimension (réduction d'information dans un rapport 16)

	Coloriage		Suivi	
	T_1	T_2	T_1	T_2
Temps T=	3198 ms	1814 ms	149 ms	88 ms
Accélération A=	1.76		1.69	
Efficacité E=	88 %		84.5 %	

Figure 12: Gain apporté par la parallélisation (T1: un seul transputer, T2: deux transputers)

	Coloriage		Suivi	
Résolution	T_1	T_2	T_1	T_2
1:1	3490 ms	2561 ms	170 ms	154 ms
1:2	912 ms	659 ms	45 ms	43 ms
1:3	433 ms	299 ms	22 ms	22 ms
1:4	261 ms	174 ms	14 ms	14 ms
1:5	181 ms	115 ms	10 ms	10 ms
1:6	139 ms	82 ms	8 ms	8 ms
1:7	112 ms	61 ms	7 ms	8 ms
1:8	95 ms	49 ms	6 ms	6 ms
1:9	83 ms	39 ms	6 ms	6 ms
1:10	74 ms	34 ms	5 ms	6 ms

Figure 13 : Résolution variable (T1: un seul transputer, T2: deux transputers).

Parallélisation du suivi de cible pour la robotique mobile

A la vue de ces tableaux, on peut tirer la conclusion suivante: l'introduction de la possibilité de faire varier la résolution a pour conséquence d'alourdir les traitements: ceci explique pourquoi, par exemple, le temps de traitement d'une image complète, sur un seul transputer, est plus faible si le programme ne permettant pas la prise en compte de la résolution est exécuté (3198 ms contre 3490 ms pour le coloriage). Il existe deux raisons principales à cela. D'une part, le code programme est quelque peu augmenté (plus d'opérations élémentaires à effectuer pour réaliser le même traitement). D'autre part, les temps de transfert des données (image à traiter) de la mémoire vidéo VRAM du transputer maître dans sa mémoire standard RAM deviennent prohibitifs. En effet, on n'a plus la possibilité de transférer les données bloc par bloc (DMA), puisque la prise en compte de la résolution oblige le transfert point par point.

V 4. Extrapolation des résultats

Des extrapolations de ces résultats ont été réalisées avec pour but la détermination d'un nombre maximum utile de transputers pour la parallélisation du suivi de cibles. La topologie retenue pour réaliser ces estimations est une topologie en arbre, telle que celle présentée sur la figure 14: il s'agit d'une topologie réalisable compte tenu de notre architecture matérielle.

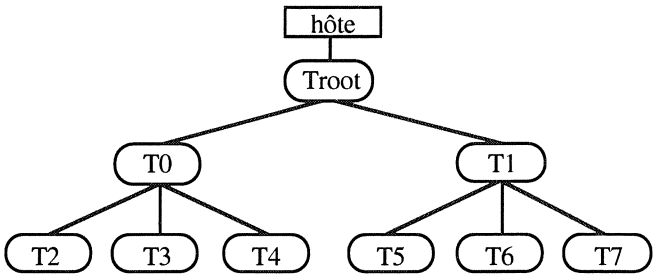


Figure 14 : Topologie en arbre utilisée

Soient:

- *n* le nombre de processeurs esclaves utiles pour le traitement désiré (coloriage ou suivi),
- *k* le nombre de passes à effectuer pour charger les données sur les processeurs esclaves,
- *Tc* le temps moyen pour communiquer un octet sur un lien,
- *TVR-R* le temps nécessaire au transfert d'un octet de la VRAM dans la RAM,
- *T* le rapport entre le temps pour effectuer le traitement désiré sur une portion d'image et le nombre de points (octets) de cette portion.

Dans [CHA-93], on a établi l'inéquation suivante:

$$n < \frac{T}{T_{VR\text{-}R}} - k\,\frac{T_c}{T_{VR\text{-}R}}$$

Suivant la parité du nombre de processeurs esclaves, k peut prendre deux valeurs:

$$k = \frac{n}{2},$$

si il y a un nombre pair de processeurs esclaves

Parallélisation du suivi de cible pour la robotique mobile

$$k = \frac{n+1}{2},$$

si il y a un nombre impair de processeurs esclaves

A partir des mesures expérimentales présentées dans les tableaux des figures 12 et 13, et des relations introduites ci-dessus, on peut en déduire le nombre maximal de processeurs esclaves pour les phases de coloriage et de suivi selon les deux cas suivants:

Transfert		Coloriage		Suivi	
T_c	T_{VR-R}	$T_{coloriage}$	nmax	T_{suivi}	nmax
0.7 µs/octet	0.5 µs/octet	15.9 µs/octet	18	6.4 µs/octet	7

- non possibilité de faire varier la résolution

Transfert		Coloriage		Suivi	
T_c	T_{VR-R}	$T_{coloriage}$	nmax	T_{suivi}	nmax
0.7 µs/octet	6 µs/octet	17.9 µs/octet	2	6.8 µs/octet	1

- possibilité de faire varier la résolution

Ainsi, on montre que le nombre de transputers esclaves (transputer maître non inclus) ne doit pas être supérieur à 18 pour la parallélisation de la tâche de coloriage et à 7 pour la phase de suivi, dans le cas où l'on ne fait pas varier la résolution. Si l'on veut garder la possibilité de faire varier la résolution, alors ce nombre passe à 2 pour le coloriage et à 1 pour le suivi. Au-delà de ces limites, les temps de communications deviennent prépondérants devant les temps de traitement: la parallélisation n'apporte alors plus de gain en temps de traitement, l'addition de nouveaux transputers devenant inutile.

Finalement, ces estimations permettent de tirer les conclusions suivantes.

Si le traitement est tel qu'il nécessite une image à pleine résolution (pas de perte de détails, etc.), alors la parallélisation de l'algorithme de suivi de cibles est efficace. Ainsi, on peut évaluer que le temps nécessaire au coloriage pour une image de taille 440x460 pixels sera de 340 ms avec 18 transputers, contre 3198 ms en séquentiel. En ce qui concerne la partie suivi, le temps sera de 69 ms avec 7 transputers, contre 232 ms en séquentiel, pour une image de taille 440x460 contenant 7 cibles.

Si on n'a pas les possibilités de posséder un nombre de transputers suffisant pour satisfaire les exigences temporelles de l'application, ou si malgré tout le potentiel de transputers à notre disposition les exigences de l'application ne sont toujours pas satisfaites, la seule solution «a priori sensée» qui reste semble être de diminuer la quantité d'informations à traiter. Il faut donc faire varier la résolution de l'image.

Or, il vient d'être montré que vouloir encore abaisser les temps de traitement séquentiels obtenus avec la possibilité de faire varier la résolution, par la parallélisation est illusoire, du moins sur une architecture matérielle telle que celle employée. En effet, des temps de transferts prohibitifs entre la VRAM et la RAM en sont la cause. Pour remédier à ce problème, il suffirait de pouvoir disposer d'une architecture à mémoire partagée, contre une à mémoire distribuée.

VI - AMÉLIORATION DE LA TECHNIQUE

VI 1. Présentation de l'approche

La limitation de la technique de suivi de cibles, d'un point de vue sémantique, vient du fait que la phase de suivi se base sur le nombre de cibles détectées dans l'image par la phase de coloriage. Ce problème est commun aussi bien à l'implantation sérielle qu'à l'implantation parallèle. En effet, si à un moment donné l'environnement est modifié, c'est-à-dire si des balises nouvelles apparaissent dans le champ de vision de la caméra, elles ne seront jamais prises en compte automatiquement par la tâche de suivi. Une intervention humaine est alors nécessaire pour ré-initialiser l'algorithme, c'est-à-dire exécuter à nouveau la phase de coloriage. Ceci est un problème qui doit absolument être résolu pour le projet qui nous concerne. Pour prendre en compte la présence de nouvelles cibles automatiquement, l'idée est de faire se dérouler la tâche de coloriage de manière concurrente à la tâche de suivi. Ainsi, sur un groupe de processeurs est implantée la technique de suivi telle que décrite dans le paragraphe 5, et sur un autre groupe est exécutée la tâche de coloriage (voir la figure 15 ci-après).

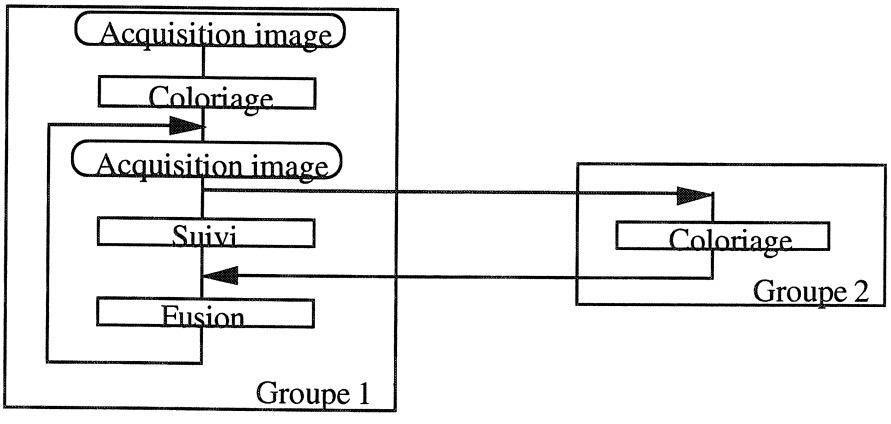


Figure 15: Coloriage et suivi en parallèle

La figure 16 présente une topologie adaptée à l'implantation de cette application. Sur chaque groupe de transputers, qui dans le cas limite peut se composer d'un seul transputer, est définie une topologie en arbre. Le modèle ferme de processeurs s'applique sur chaque groupe où est désigné un processeur maître (Troot). De plus, un des transputers maîtres d'un groupe doit être maître du réseau afin d'assurer la gestion et la synchronisation de l'ensemble du réseau.

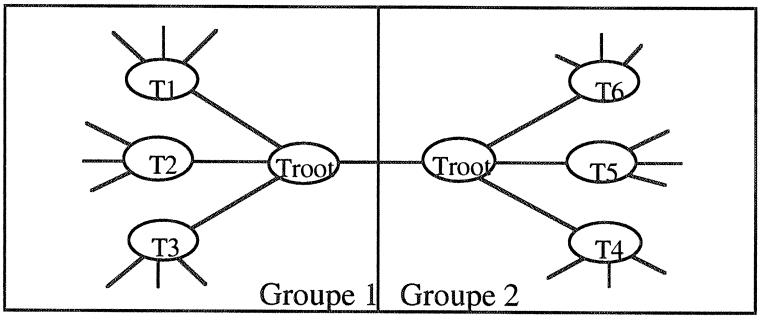


Figure 16 : Topologie adaptée à cette implantation

Le principe de synchronisation du réseau est décrit ci-après. Soit Troot1 le transputer maître à la fois du groupe1 et du réseau. Soit Troot2 le transputer maître du groupe 2 et esclave du transputer Troot1. Troot1 s'occupe de faire exécuter par son groupe le coloriage d'initialisation sur la première image, puis fait exécuter le suivi sur la séquence d'images. Durant le suivi, il interroge régulièrement Troot2 pour savoir si le groupe 2 est actif ou non. Deux cas peuvent se présenter:

• Soit Troot2 est inoccupé et Troot1 lui envoie alors une image afin que Troot2 la fasse colorier par son groupe. Hormis pendant la phase d'initialisation, si Troot2 est inoccupé, cela signifie qu'il a terminé son travail (coloriage). Ainsi, il peut donc communiquer à Troot1 les résultats de son coloriage, c'est-à-dire le nombre de cibles détectées dans l'image, ainsi que leurs attributs.

• Soit Troot2 est occupé (à gérer le coloriage sur son groupe), alors Troot1 continue à exécuter le suivi.

Le module *Fusion*, introduit sur la figure 15, a pour but de fusionner les résultats fournis par le transputer Troot2 au transputer Troot1. En d'autres termes, il s'agit pour le transputer maître Troot1 de remettre à jour de manière cohérente la structure des données relative aux cibles à suivre dans la séquence d'images. La stratégie de fusion des données est décrite ci-après:

• Soit *C1* la structure de données relative aux cibles détectées par le groupe 2 et qui sont encore suivies par le groupe 1.

• Soit *C2* la structure de données relative aux cibles nouvelles, c'est-à-dire celles qui sont détectées par le groupe 2 et qui n'étaient pas suivies auparavant par le groupe 1.

• Soit *C3* la structure de données relative aux cibles non détectées par le groupe 2 et pourtant toujours suivies par le groupe 1. Ce cas peut en effet se présenter

fréquemment: pour détecter une cible par coloriage, il faut que celle-ci possède une taille suffisante et une forme particulière; par contre, pour le suivi il suffit que quelques points demeurent présents dans la fenêtre de prédiction.

La structure de données globale C, relative aux cibles à suivre dans l'image suivante, est remise à jour par une simple union : $C = C1 + C2 + C3$.

Il est important de remarquer que lorsque l'on obtient les résultats du coloriage réalisé par le groupe 2, plusieurs images dans la séquence ont pendant ce temps été traitées par le groupe 1 (phase de suivi). Ainsi, pour des cibles nouvelles détectées dans le champ de la caméra, nous avons introduit la possibilité de prendre en compte ce décalage temporel afin d'estimer au plus juste la position de la fenêtre de recherche. Le principe retenu est le suivant: il s'agit d'estimer la position de la fenêtre de recherche de la cible nouvelle dans l'image suivante, à partir des déplacements des cibles présentes dans son voisinage et toujours suivies.

VI 2. Résultats

Cette version de suivi de cibles a été implantée sur un réseau composé de deux transputers. Ainsi, chaque groupe de transputers ne se compose en fait que d'un seul transputer. L'extension de la parallélisation à un réseau comprenant plus de transputers peut aisément se faire conformément à ce qui a été décrit dans le paragraphe 5.

Expérimentalement, le fait de prendre en compte le décalage temporel pour l'estimation de la position des fenêtres de recherche des cibles nouvelles ne semble pas être d'un grand intérêt. En effet, compte tenu des fréquences d'exécution des phases de suivi et de coloriage (10 Hz environ pour le suivi contre 2 Hz environ pour le coloriage en travaillant à une résolution de 1:4), le système fonctionne très bien sans prise en compte du décalage temporel. Ainsi, avec une résolution de 1:4, on a mesuré expérimentalement une fréquence de traitement du système de l'ordre de 2 Hz. Certes cela impose des déplacements moins rapides de la caméra (ou des cibles relativement à la caméra), puisque la fréquence de traitement du système est considérablement diminuée par rapport à la première implantation (paragraphe 5).

Cependant maintenant, la prise en compte de cibles nouvelles dans le champ de la caméra se fait automatiquement. De plus, la robustesse de l'algorithme est considérablement accrue. En effet, prenons le cas de deux balises dont les images sont voisines. Ces deux cibles seront détectées et suivies. A un certain moment, suivant le déplacement du robot, imaginons que ces images deviennent confondues. Dans ce cas, le suivi ne s'appliquera plus que sur une seule cible (considérée comme l'union des deux). Quand la position du robot sera telle que les images des balises sont à nouveau disjointes, cela sera pris en compte automatiquement par l'algorithme.

VII - CONCLUSION

Dans cet article nous avons présenté un travail typique de parallélisation en développant les différentes étapes de la réalisation d'une application de suivi de cible en robotique mobile. Il met en évidence le potentiel que peut apporter le parallélisme. Si l'utilisation d'une architecture parallèle pour obtenir un gain en temps d'exécution est somme toute assez banale, celle pour l'amélioration de la sémantique et/ou la robustesse d'un algorithme est quelque peu délaissée, souvent à tort. En effet, dans cet article nous avons abordé ces deux avantages du parallélisme. Tout d'abord, nous avons proposé la parallélisation d'une technique de suivi de cibles, permettant des fréquences de traitement plus grandes et ensuite amélioré cette technique grâce au potentiel architectural à notre disposition. Ainsi, nous disposons d'un système de vision artificielle pour la robotique mobile qui est capable de détecter automatiquement des balises placées sur des sites d'intérêt. A partir de là, de nombreuses applications peuvent être envisagées.

RÉFÉRENCES

[AKI-92] M. Akil, E. Dujardin, «Parallélisation des transformations morphologiques des images sur réseau de transputer», La Lettre du Transputer, nº 13, p. 7-19, Mars 1992.

[BAL-82] D.H. Ballard, C.M. Brown, «Computer Vision», Prentice Hall Inc., Englewood Cliffs, N.J. 07632, p. 151-152, 1982.

[CHA-91] F. Chantemargue, M. Popovic, R. Canals, P. Bonton «Parallelization of the merging step of the region segmentation method», The 7th Scandinavian Conference on Image Analysis, Aalborg, Denmark, p. 933-940, August 13-16 1991.

[CHA-93] F. Chantemargue, «Suivi de cibles : Implantation sur un réseau de transputers», rapport IMT, Université de Neuchâtel, IMT 345 HU07/93, Juillet 1993.

[CHAS-92] J.M. Chassery, G. Mando, D. Adelh, J.L. Roch, «Implantation des algorithmes bas niveau de traitement d'images sur un réseau de transputers», La Lettre du Transputer, nº 13, p. 55-69, Mars 1992.

[FAC-92] C. Facchinetti & H. Hügli, «Two vision-based behaviours for autonomous mobile robots», Proceedings of the First Swiss Symposium on Pattern Recognition and Computer Vision, Ed. J. Bigün & J.M.H. du Buf, EPF Lausanne, January 1992.

[HUG-92] H. Hügli, G. Maître, F. Tièche, C. Facchinetti «Vision-based behaviours for robot navigation», Proceedings of the Fourth Annual SGAICO Meeting, Neuchâtel, Septembre 1992

[HUG-93] H. Hügli, F. Tièche, F. Chantemargue, G. Maître, «Architecture of an experimental vision-based robot navigation system», Proc. of Swiss Vision'93, Sept.14, 1993.

[IP1-92] «IP1 User Manual», National Engineering Laboratory, 1992.

[NOM-92] «Nomadic robot programming manual», Nomadic Technologies Inc., Standford, 1992

[QUI-89] P. Quinton, Y. Robert, «Algorithmes et architectures systoliques», Edition Masson, E.R.I., 1989.

[TBX05-92] «TBX05 Transputer Module Motherboard Users Guide», A Mark Ware TM Product, 1992.

[TBX10-91] «TBX10 Transputer Module Motherboard Users Guide», A Mark Ware TM Product, 1991.