Implémentation parallèle d'un système de focalisation de l'attention

Mathias QUOY, Olivier GALLET, Philippe GAUSSIER

ETIS, Université de Cergy-Pontoise, ENSEA, URA CNRS 2235
ENSEA 6, av du Ponceau 95014 Cergy-Pontoise, France
email : quoy@ensea.fr

RÉSUMÉ

Cet article présente l'implémentation parallèle d'un algorithme permettant de déterminer une séquence de points de focalisation et de simuler le "pop-out" (une figure présentant des caractéristiques visuelles différentes "saute aux yeux" de celui qui la regarde). L'algorithme développé se base sur une architecture neuronale. Il se sert de la bibliothèque de parallélisation PVM utilisant un ensemble de stations de travail et non une machine dédiée. Le temps de calcul est diminué d'un facteur 4,5. La même architecture est utilisée pour les différentes étapes de l'algorithme. Le programme intègre d'autres fonctions parallélisées (filtrage de Gabor) et non parallélisées (flot optique).

ABSTRACT

This article is presenting the parallel implementation of a focalization points detector. It enables to simulate the "pop-out" mechanism. The algorithm is based on a neural architecture. It is using the PVM library. This library provides parallelization procedures using a pool of workstations (and not a specific computer). The computation time is reduced by 4.5. The same architecture is used for each step of the algorithm. The whole developped package includes other parallelized and not parallelized functions (Gabor filtering and optical flow respectively).

1 Introduction

D'un point de vue général, la conception de systèmes de vision intégrés est toujours délicate. Ils doivent notamment faire des traitements sur des données de très grande taille (contrainte de temps de calcul et d'espace mémoire). Au niveau du cerveau, cette très grande masse d'informations est d'abord traitée de manière massivement parallèle (vision préattentive). Mais même ce type de système massivement parallèle est vite soumis à une explosion combinatoire [1]. Un moyen de réduire la quantité d'information à traiter est de sélectionner certaines parties de l'images. C'est ce que semble permettre l'attention visuelle. L'intérêt de cette "focalisation de l'attention" est aussi de pouvoir recaler une forme dans le centre du champ de vision pour sa reconnaissance. Cela fournit un mécanisme simple pour obtenir l'invariance par rapport à la position. Les différentes zones de focalisation sont explorées séquentiellement. Nous utilisons ce mécanisme pour la navigation de robots mobiles [2]. Nos robots doivent se déplacer dans une pièce : ils détectent des obstacles et se repèrent uniquement grâce aux informations fournies par la caméra CCD dont ils sont dotés. Nous définissons la partie "intéressante" d'une image comme étant l'endroit comportant le moins de caractéristiques partagées par les autres endroits. Cette définition s'appuie sur les observations de psychophysiciens qui ont étudié le phénomène de "pop-out" ("qui saute aux yeux"). Sans entrer dans les détails du phénomène [3, 4], ce mécanisme permet de reconnaître de manière instantanée (sans parcours séquentiel de tous les objets d'une image) celui (ou ceux) qui sont les plus dissemblables des autres (Fig. 1). Cette propriété nous intéresse particulièrement pour que nos robots puissent se diriger. En effet, ces parties "qui ont sauté aux yeux" peuvent leur servir de point de repère (amer). En se basant sur au moins 3 de ces points de repères, un robot peut connaître sa position. L'ensemble ordonné des points de focalisation fourni la séquence de focalisation du système visuel du robot. L'exploration de la scène se fait alors par le parcours de cette séquence de focalisation (c'est l'équivalent des saccades oculaires). Ce système doit être très rapide pour pouvoir fonctionner en-ligne. Une version séquentielle simplifiée tourne actuellement sur nos robots [5]. L'algorithme développé se fonde sur des réseaux de neurones artificiels. Ainsi, les aspects d'intégration (bottom-up) et de rétroaction (top-down) sont faciles à mettre en oeuvre. La rétroaction permet en outre d'ajouter un mécanisme attentionnel guidant la recherche sur des points intéressants différents suivant les taches à effectuer. De plus, ce formalisme nous permet aussi de développer naturellement des algorithmes parallélisables. La parallélisation au niveau du neurone individuel n'est pas rentable. C'est donc la parallélisation de groupes de neurones qui est réalisée ici.

2 Modèle

La réalisation de la séquence de focalisation utilise trois grandes étapes : extraction des contours orientés, détection des points de courbure maximale (points caractéristiques (le plus souvent des coins)) et appariement de ces points [6] (Fig. 2). Chacune de ces étapes génère plusieurs cartes dont la taille correspond à l'image de départ. Chaque étape est elle-même découpée en trois séquences (identiques pour les trois étapes) : diffusion, compétition à l'intérieur de chaque carte et compétition entre les cartes. Ce sont les deux premières étapes qui sont parallélisées (Fig. 3). Les images correspondants

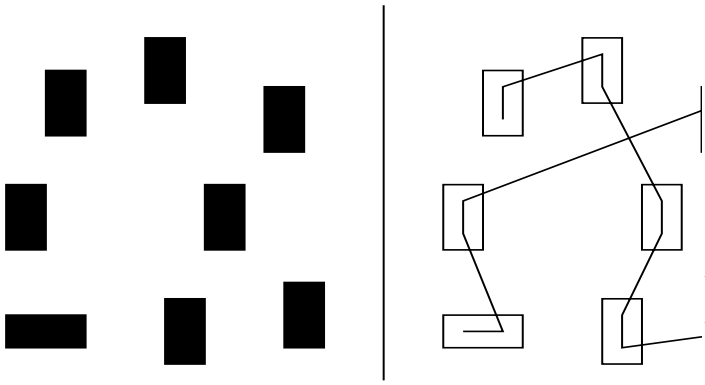

FIG. 1 — L'image de gauche montre un exemple de "pop-out" sur le rectangle horizontal. Sur l'image de gauche, les points de focalisation successifs sont reliés par des traits. Les deux premiers points de focalisation se trouvent sur le rectangle horizontal.

aux différentes cartes sont découpées en plusieurs morceaux (imagettes). Chacune de ces imagettes est traitée en parallèle. Les imagettes résultats sont ensuite regroupées pour reformer une image complète utilisée lors de la compétition entre les cartes. L'étape de création des cartes de caractéristiques est aussi parallélisée au niveau du regroupement des cartes de points de points de courbures appariés (ce n'est donc pas le découpage en imagettes qui est utilisé ici). Le groupe de neurones dont le traitement est parallélisé correspond à la taille des données d'une imagette plus la taille des filtres utilisés pour la diffusion et la compétition. L'étape de compétition est la plus limitative pour la parallélisation, car si la compétition s'effectue sur un large domaine, elle oblige à conserver un ensemble de neurones plus important.

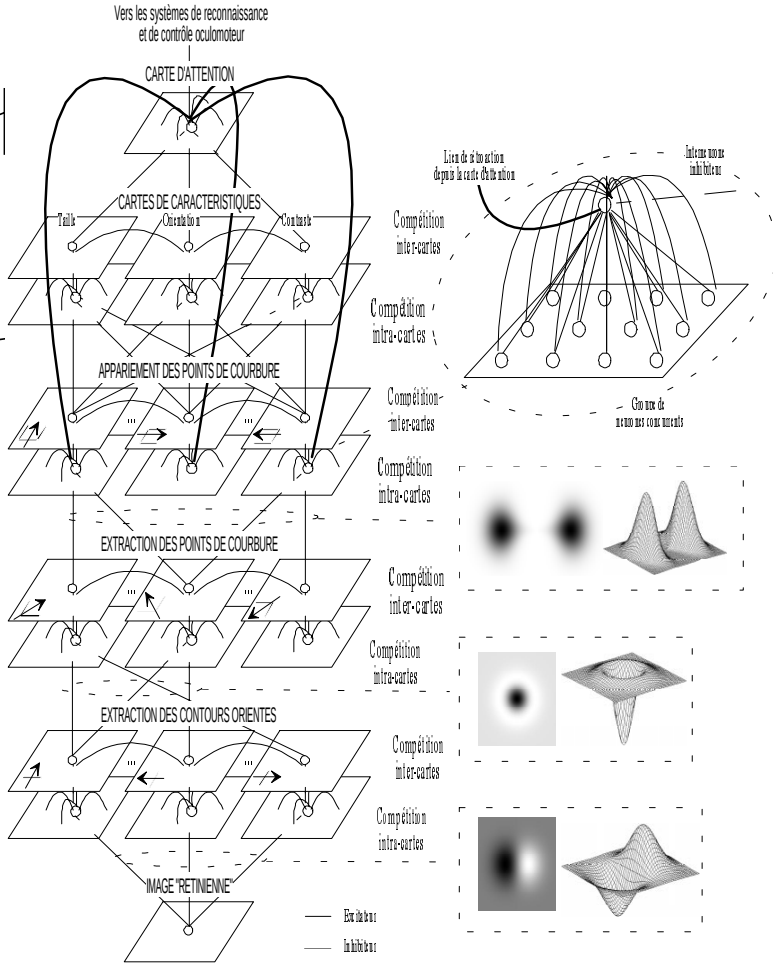

FIG. 2 — Schéma général du système de focalisation de l'attention : extraction des contours orientés, détection des points de courbure maximale et appariement de ces points. Il y a ensuite regroupement dans des cartes de caractéristiques. La carte maîtresse fournit le point de focalisation. Le point suivant est trouvé en inhibant au niveau des cartes de caractéristique celles qui ont généré les points précédents. Les dessins encadrés en pointillés correspondent aux différents filtres utilisés à chaque étape.

3 Implémentation

La programmation a été réalisée en langage C en utilisant la bibliothèque de parallélisation fournie par PVM (Parallel Virtual Machine) [7]. L'interface graphique utilise les fonctions de OpenView (fonctionnement dans l'environnement Solaris 4.2 sur Sun et sur PC). Les routines PVM permettent de regrouper des stations de travail (même avec des systèmes d'exploitation différents) dans une même "machine virtuelle" comprenant autant de processeurs que de stations. Chaque programme lancé sur un processeur est appelé une tâche.

La parallélisation de l'algorithme se fait au niveau du découpage des données de chaque carte en imagettes (Fig. 3). Une tache maître est chargée de découper les images et de répartir les imagettes sur les différentes taches (taches esclave). Dans notre cas, chaque processeur reçoit une tache esclave. Une tache de contrôle est chargée de suivre l'évolution des opérations. Le nom des processeurs et la durée du traitement d'une imagette sont mis-à-jour en temps réel pour que l'utilisateur ait un retour visuel du déroulement du programme. Le découpage d'une image (et donc une partie de l'efficacité de la parallélisation) est déterminé par la taille des masques de diffusion et de compétition. Ainsi la taille minimale de l'imagette doit être au moins égale à deux fois la taille du plus grand masque. Dans tous les tests effectués, le nombre d'imagettes (N) est supérieur au nombre de processeurs (P) (dans la plupart des simulations, $N = 16$ et $P = 7$). Chaque esclave renvoie ensuite à la tache maître une imagette résultat correspondant aux deux premières séquences définies plus haut : diffusion et compétition à l'intérieur de la carte. La tache maître regroupe ces imagettes pour former les nouvelles cartes de caractéristiques. S'il reste des imagettes à traiter, la première de la liste est renvoyée à la tache esclave venant de finir un calcul. Une fois toutes les imagettes envoyées et tous les résultats reçus, la tache maître effectue l'étape de compétition entre les nouvelles cartes. On a donc une succession de trois phases de parallélisation (une pour chaque étape) suivies de 3 phases de regroupement des données. Ce regroupement des données est alors un goulot d'étranglement pour les performances globales du système, notamment à cause du temps d'attente pour

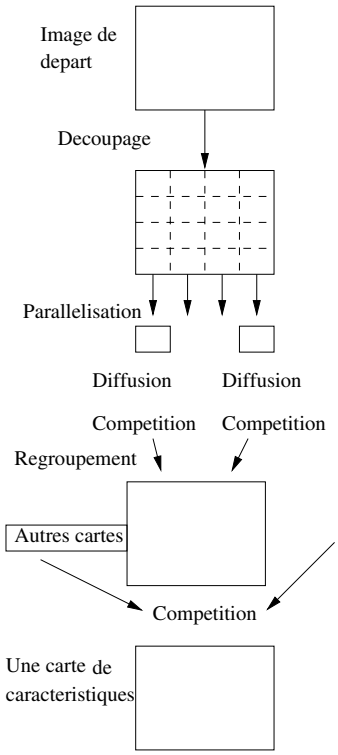

FIG. 3 — Schéma de parallélisation et de regroupement en vue de l'obtention d'une carte de caractéristiques. Chaque carte est obtenue par une compétition entre des cartes différentes.

le calcul de la dernière imagette à recevoir. En effet, les taches s'effectuant sur des processeurs différents ne s'exécutent pas à la même vitesse. Dans le pire des cas, il est alors tout à fait possible d'envoyer la dernière imagette à traiter à la tache s'exécutant sur le processeur le moins rapide. Le temps d'attente sera alors maximum. Pour éviter ce phénomène, on lance aussi ce dernier motif sur un processeur plus rapide dès qu'il a terminé le traitement de l'imagette précédente. Ce processus est généralisé et appliqué aux P dernières imagettes. L'algorithme devient alors : répartition des N imagettes aux P taches (lorsqu'une tache a fini un calcul, elle reçoit une nouvelle imagette). Une fois que $N - P$ imagettes résultat ont été reçues, la vitesse de chaque processeur est estimée par le nombre d'imagettes qu'il a traité. Cela fournit un classement des processeurs en fonction du nombre d'imagettes traitables parmi les P renvoyables (coefficient de renvoi). Ainsi, s'il y a 4 processeurs A, B, C et D, que C a traité 50% des N-P premières imagettes, B et D 20% et A 10%, C sera affecté du coefficient de renvoi 2 (il peut traiter 2 imagettes renvoyées), B et D de 1, et A de 0 (il ne recevra aucune imagette relancée). Il reste donc maintenant à la tache maître P imagettes à recevoir. Dès qu'une imagette est reçue, si le coefficient de renvoi du processeur qui a fait le calcul est strictement positif, l'imagette calculée par le processeur le plus lent lui est renvoyée. Son coefficient de renvoi est diminué de 1 et l'imagette renvoyée est placée dans la liste des imagettes renvoyées. Cette dernière manipulation permet à la tache maître d'envoyer au prochain processeur non pas la même imagette que précédemment (celle traitée par le processeur le moins rapide), mais celle traitée par le deuxième processeur le moins rapide. Cette procédure est répétée jusqu'à ce que toutes les imagettes soient reçues. Le classement est remis à jour à chaque étape permettant ainsi une adaptation dynamique de l'algorithme aux changements de l'environnement de travail (taux d'occupation du réseau, machine utilisée soudainement par un autre utilisateur, panne d'une machine ...). Il faut occuper au maximum le temps de calcul des processeurs aussi bien avec la tache maître qu'avec les imagettes relancées sur les taches esclaves. Le système est donc robuste à la disparition d'un processeur comportant une tache esclave. Ce n'est plus vrai s'il s'agit de la tache maître ou de la tache de contrôle. Une manière d'y remédier serait de relancer automatiquement ces taches si elles n'envoyaient aucun message pendant un certain temps. Un autre procédé pour accélérer l'algorithme est de diminuer le temps séparant les phases parallélisées. On a vu que ce temps était dû à la compétition entre les cartes résultat de l'étape précédente. Ainsi, en envoyant systématiquement la tache maître sur la machine la plus rapide (suivant le critère défini ci-dessus), on diminue ce temps de calcul.

4 Résultats

Le réseau de stations de travail comporte au maximum 7 stations de travail SUN (Sparc 5). L'image à traiter est de taille 256x256 en 16 niveaux de gris. Les tableaux suivants présentent le temps de calcul (en secondes) de chacune des étapes et total en fonction du découpage de l'image de départ ; plus le découpage est grand, plus l'imagette est petite. Ce temps de calcul est une moyenne sur 10 essais de l'algorithme. Le temps de calcul moyen du même algorithme en séquentiel sur un Pentium Pro200 avec 16M de RAM est de 15 minutes (900 secondes). La version parallélisée va donc en moyenne 4,5 fois plus vite.

La troisième étape montrée dans les tableaux (points de focalisation) ne dépend pas du découpage comme déjà indiqué plus haut dans le paragraphe Modèle. Il est à noter que le temps de calcul ne diminue pas avec un découpage plus grand. Ceci est dû à l'augmentation du nombre des échanges entre la tache maître et les esclaves (plus d'imagettes), et aussi à la diminution de la taille utile d'une image : les masques de diffusion et de compétition sont toujours de même taille, mais la taille de l'image à traiter diminue. Nous avons aussi constaté que le nombre d'utilisateurs du réseau influe sur les performances de la parallélisation. Il est aussi plus souhaitable d'avoir un réseau homogène de peu de machines plutôt qu'un réseau avec une ou deux machines très rapides et beaucoup de machines moins performantes. Ces dernières effectuent peu de calculs (au mieux une imagette) et tout le reste du traitement est fait par les machines rapides, ramenant les performances globales à celles d'un réseau possédant deux machines.

7 stations	Découpage			
	9	16	36	64
contours	10 s	10 s	10 s	10 s
points caractéristiques	115 s	105 s	130 s	200 s
points de focalisation	80 s	80 s	80 s	80 s
total	205 s	195 s	220 s	290 s

6 stations	Découpage			
	9	16	36	64
contours	10 s	10 s	10 s	10 s
points caractéristiques	105 s	100 s	145 s	200 s
points de focalisation	80 s	80 s	80 s	80 s
total	195 s	190 s	235 s	290 s

5 Version complète

Nous n'avons parlé jusqu'à présent que du système de focalisation de l'attention fondé sur l'extraction de points caractéristiques. Le programme développé permet de prendre en compte d'autres modalités du système visuel. D'abord la détection du mouvement dans les zones périphériques (ou non) de l'image à travers le calcul du flot optique. L'endroit où le flot est maximal fourni un point de focalisation qui est prioritaire dans la séquence de focalisation. Ainsi, par exemple, la focalisation se fera en priorité sur un objet faisant irruption soudaine dans le champ visuel. Une version séquentielle du calcul du flot optique est déjà intégrée dans le programme. Ensuite, Il est prévu d'intégrer l'analyse des textures. Un banc de filtres de Gabor permet déjà d'avoir différentes cartes filtrées de l'image. Il reste à appliquer les principes précédents de diffusion et de compétition pour extraire de l'image les zones répondant au mieux aux différents filtres. Ceci sera réalisé prochainement.

6 Conclusion

L'architecture neuronale développée pour la détermination d'une séquence de focalisation a permis une parallélisation naturelle de cet algorithme. Ce programme parallèle s'intègre dans un système plus complexe qui traitera d'autres caractéristiques visuelles. L'approche neuronale, qui permet un codage de l'information analogique dans toute l'architecture, permet aussi de déterminer des rétroaction "top-down" : il suffit de favoriser certaines cartes de caractéristiques pour permettre la focalisation sur des zones où elles sont présentes. Ceci introduit un biais sur la focalisation permettant de guider sélectivement l'attention (pour la recherche d'une caractéristique précise par exemple).

Nous tenons à remercier Laurence Hafemeister et Fabien Tran pour leur collaboration. Ce projet a été financé par le contrat F61708-96-W0197 avec l'Air Force Office of Scientific Research.

Références

[1] Tsosos J.K. A 'Complexity level' analysis of immediate vision, Int Journ of Computer Vision, (1988), 303-320.

[2] Gaussier P., Joulain C., Zrehen S., Banquet J.P., Revel A. Navigation visuelle dans un environnement ouvert : reconnaissance de vues panoramiques, Gretsi, (1997).

[3] Treisman A., Gelade D. A feature integration theory of attention, Cogn. Psych., 16, (1980), 97-136.

[4] Duncan J., Humphreys G.W. Visual search and stimulus similarity, Psych. Rev., 96, (1989), 433-458.

[5] Gaussier P., Cocquerez J.P. Utilisation des réseaux de neurones pour la reconnaissance de scènes complexes : simulation d'un système visuel comprenant plusieurs aires corticales, Trait. Signal, Vol 8, num 6, (1994), 441-466.

[6] Gallet O., Gaussier P., Cocquerez J.P. Focalisation de l'attention visuelle : un modèle parallèle d'amplification de singularité, NSI 96, Marly-le-Roi, 6-9mai, (1996), 237-240.

[7] Geist G.A., Beguelin A., Dongarra J.J., Jiang W., Manchek R., Sunderam V.S. PVM : a users' guide and tutorial for networked parallel computing, MIT Press, (1994)