Architecture de vision multi-GPUs/CPUs temps réel pour l'aide au pilotage d'hélicoptère

ALEXANDRE RIVIERE[1], FRANCK ANGELLA[1], RICHARD ALVAREZ[2]

[1] Laboratoire SI-IS, EADS France / Innovation Works
12 rue Pasteur, BP 76, 92150 Suresnes Cedex, France
[2] ETXSIF, Eurocopter
Aéroport international Marseille Provence, 13725 Marignane Cedex, France

[1]Alexandre.Riviere@eads.net, Franck.Angella@eads.net,
[2]Richard.Alvarez@eurocopter.fr

Résumé - Cet article présente une plateforme matérielle et logicielle de prototypage rapide et d'exécution en temps réel souple de traitements vidéo. Cette plateforme multi-CPU/GPU permet au pilote d'évaluer un panel de traitements en situation de vol. L'architecture fournit aussi les moyens d'adapter ou de développer rapidement une chaine de traitements pouvant être présentée au pilote.

Abstract – This paper describes a co-designed hardware and software platform for fast prototyping of soft real-time video processing. With this multi-CPUs/GPUs platform, a pilot can evaluate many vision-based processing chains in-flight situation. The architecture also provides means and tools for rapid processing chain development that can be submitted to the pilot.

1 Introduction

Les systèmes de vision améliorés pour l'assistance au pilotage intéressent l'industrie aéronautique depuis plusieurs années, que ce soit pour augmenter la sécurité des équipages et des aéronefs ou fournir une image actualisée de l'environnement extérieur [1]. Ces systèmes de vision sont des systèmes relativement complexes et sont aujourd'hui principalement destinés aux pilotes [2]. En conséquence, les pilotes participent à l'élaboration puis à l'évaluation des chaines de vision mises en œuvre. Idéalement, ces évaluations sont réalisées en vol et le pilote doit pouvoir interagir avec le système de vision en situation, en temps réel. L'utilisateur est impliqué dès la phase de recherche. L'enjeu est de pouvoir rapidement développer un éventail de chaines de traitements à évaluer en temps réel, puis, en fonction des résultats, les modifier tout aussi rapidement.

1.1 Architecture matérielle

Pour répondre au cadre applicatif, la plateforme capture deux voies vidéo en entrée pour produire une vidéo en sortie. Bien que peu courante pour ce type de traitements vidéo temps-réel, nous avons choisi une architecture PC hybride CPU/GPU (Figure 1) pour sa forte capacité de traitements bruts et sa relative simplicité de développement. Les GPU sont aussi, par nature, adaptés au traitement d'image. D'autres algorithmes restent plus performants sur CPU. En conséquence, nous avons souhaité étudier la pertinence d'une telle architecture pour une application d'évaluation de traitements vidéo temps réel. Cette plateforme est une station de travail comprenant huit cœurs CPU Intel Xeon, trois GPU Nvidia ainsi qu'une carte de capture vidéo double voie Matrox. Deux GPU sont dédiés aux calculs hybrides GPGPU [3]. Le troisième GPU sert au rendu vidéo de sortie. Sur les huit cœurs CPU disponibles, deux sont dédiés au contrôle des GPU de calculs, quatre aux calculs hybrides. Les deux autres cœurs se chargent du programme principal et d'annexes diverses. Ces attributions simplistes et franches ont pour but d'obtenir des temps de calculs les plus stables possibles dans un environnement non temps réel strict.

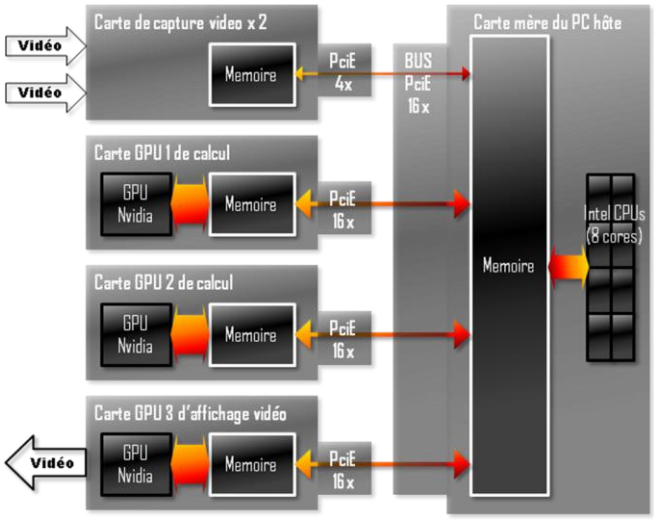

Figure 1 : architecture matérielle

1.2 Architecture logicielle

Nous avons choisi une distribution Linux Suse. Linux à l'avantage d'offrir différents type de solutions d'extension pseudo temps-réel potentiellement utilisables afin d'améliorer les latences, la préemptibilité et le déterminisme du système.

Le composant principal de l'architecture logicielle (Figure 2) est le moteur d'exécution de chaines de traitements - une chaine étant un ensemble de modules de traitements algorithmiques organisés séquentiellement et/ou concurremment. L'exécution de la chaine de traitement est déclenchée dès la capture des deux trames vidéo. Le moteur d'exécution distribue alors la charge de calculs des modules de traitement sur les unités de calculs GPU et CPU. Une fois tous les traitements terminés, le moteur d'exécution présente le résultat au moteur de rendu vidéo puis se place à nouveau en attente pour le cycle suivant.

Une interface graphique permet de concevoir une chaine puis de reconfigurer le moteur d'exécution avec cette nouvelle chaine lors du déroulement du programme. Cette interface a pour but de tester, valider et proposer rapidement à l'évaluateur de nouvelles chaines de traitements.

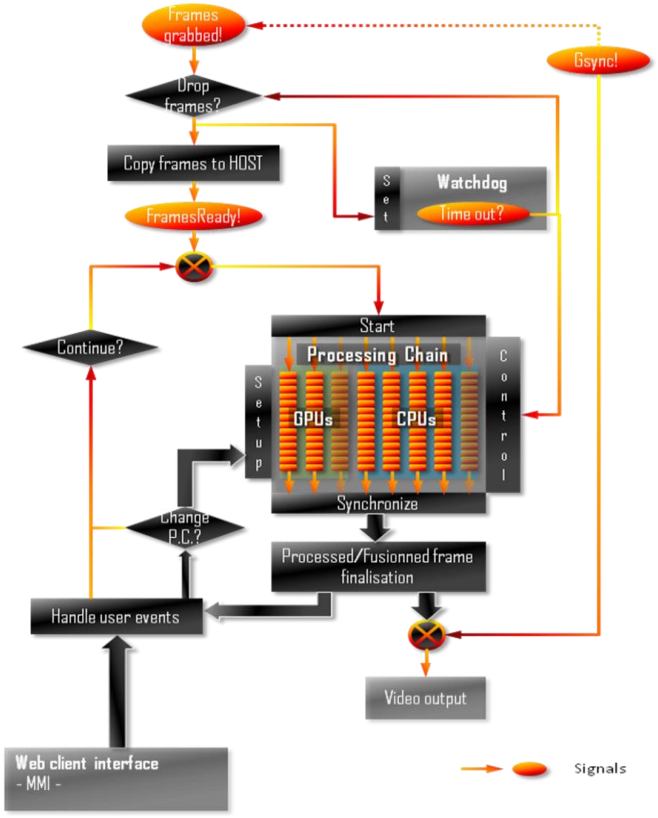

Figure 2 : architecture logicielle

2 Temps réel

Peu courantes pour répondre aux contraintes temps-réel, nous avons choisi d'explorer les capacités d'une architecture PC hybride : multi-CPU/GPU (Figure 1).

Les architectures PC hybride multi-CPU/GPU sont peu utilisées, peu adaptées pour répondre à des contraintes temps-réel. Nous avons cependant choisi d'explorer les capacités d'une telle architecture (Figure 1). A partir des caractéristiques techniques du matériel, il était intéressant de l'utiliser pour remplir des fonctions non usuelles. Les GPU sont, par nature, adaptés au traitement d'image et par extension à la Vision [4]. De plus, leur importante capacité de traitement brute permet de considérer les GPU comme suffisant pour répondre aux exigences temps-réel souple de notre application d'évaluation. Nous avons intégré les défauts inhérents à l'architecture et avons adopté une approche pragmatique.

Le système d'exploitation (OS) utilisé est un Linux standard, sans extensions temps-réel - le matériel n'est pas compatible avec les OS temps-réel. En conséquence, les temps de traitements ne sont pas déterministes et l'application subit les latences dues au système. Nous avons développé un watchdog pour contrôler des temps de traitements dont les rôles sont de maintenir la stabilité du système et de fournir des informations statistiques sur les dépassements de temps. Ces informations servent à affiner l'agencement des chaines ou à dégrader, améliorer automatiquement les performances des traitements adaptatifs.

La limitation majeure des architectures PC hybride est la relative étroitesse de la bande passante de l'interface de communication entre les espaces mémoire CPU et GPU : le bus PCI-express [5]. Par exemple, dans une chaine hybride, les résultats d'un traitement sur GPU sont transmis au traitement suivant sur CPU via ce bus. Dans notre cas les données à transférer sont souvent des trames vidéo de taille importante. Chacun de ces transferts d'image prend un temps non négligeable sur la courte période allouée. Là encore, nous avons adopté un compromis suffisant en réduisant au minimum le nombre de transferts CPU/GPU nécessaire lors de la phase de conception des chaines de traitements. En contrepartie, cela demande un effort supplémentaire au concepteur de chaine qui doit aussi tenir compte de cette contrainte.

La contrainte temps-réel sur notre architecture impose au concepteur de chaine d'optimiser empiriquement la distribution des traitements sur les unités de calculs (UC) et les transferts mémoire. Nous avons considéré cet effort d'optimisation pour atteindre l'objectif de temps comme faible notamment grâce à la forte puissance de calcul brute disponible.

Exemple avec une chaine de correction de défauts optiques et de rehaussement de contraste :

A) Correction des défauts optiques
B) Histogramme
C) Rehaussement de contraste du résultat de A en fonction de B.

Voici une distribution suffisante de cette chaine sur la plateforme hybride (Figure 2):

1) La trame vidéo d'entrée arrive dans l'espace mémoire CPU. Elle est copiée vers un GPU pendant que l'histogramme de l'image est calculé en parallèle sur 4 cœurs CPU. Chaque cœur traite 1/4 de l'image (B).
2) Une fois la trame originale copiée sur le GPU, Le traitement A corrige la déformation due au système d'acquisition optique et produit une nouvelle image corrigée pour C.

3) Sur CPU, dès que les histogrammes partiels sont obtenus, le calcul de leur somme s'effectue sur un des cœurs. Cette somme est ensuite copiée vers le GPU. Sur un autre cœur CPU disponible un module de traitement de routine indépendant (x) s'exécute.
4) Sur GPU, le rehaussement de contraste (C sur GPU) connait déjà le résultat de A (image corrigée) et attend l'histogramme. Dès qu'il le reçoit, le rehaussement de contraste peut commencer.
5) En fin de chaine, l'image finale produite est copiée vers l'espace mémoire CPU.

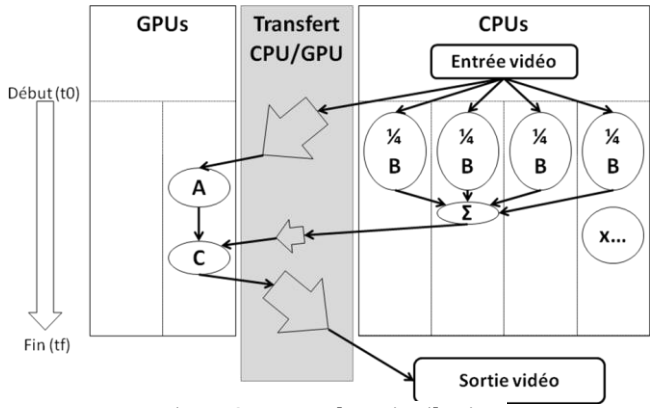

Figure 2 : Exemple : Distribution hybride d'une chaine traitement : correction d'objectif et rehaussement de contraste.

3 Prototypage rapide de chaines

Comme nous l'avons entre-aperçu pour une simple chaine (Figure 2), la construction d'une chaine même suffisamment optimale peut rapidement devenir un exercice de développement informatique complexe. D'autant plus que le nombre de combinaisons de chaines et de distribution des modules de traitement dans la chaine augmente fortement avec l'ajout de nouveaux modules.

L'objectif principal étant de pouvoir prototyper rapidement une chaine de traitement, nous avons développé un moteur d'exécution de chaines de traitement modulaires reconfigurable. Ce moteur utilise six threads CPU parallèles dont deux contrôlent le travail des deux GPU. Chaque thread possède sa propre pile de travaux séquentiels. Techniquement, la re-configurabilité dynamique du moteur est assurée par un tableau de fonctions correspondant à ces six piles de travaux. En début de cycle (i.e. trame(s) vidéo capturée(s)), le moteur active simultanément tous les threads qui exécutent un à un les travaux de leur pile. Une fois leur pile de travaux vide, les threads se bloquent jusqu'au prochain cycle. En fin de cycle, le moteur s'assure que toutes les piles soient vides pour valider le cycle complet et lancer le rendu vidéo final.

A plus haut niveau, pour un utilisateur concepteur de chaine, ce moteur est reconfigurable grâce à une interface graphique (Figure 3). Cette interface est un tableau dont les colonnes représentent les différents threads (parallélisassions). Les lignes représentent les étapes séquentielles des piles de travaux. Le contenu d'une case non vide est appelé une « action ». Une action est définie par son type et une description de cette action que le concepteur peut choisir via l'interface graphique.

Nous avons défini cinq types d'actions :

- *ImgCpy*: Copie d'image.
- *GpbCpy*: Copie de tampon générique.
- *Process*: Traitement d'image.
- *Fusion*: Fusion d'images.
- *Wait*: Synchronisation : attente de la fin d'une ou plusieurs autres actions.

Nous avons ensuite spécifié une interface de description pour chacun de ces types d'action :

Exemples

ImgCpy:

- ID image source.
- Type des données source.
- Décalage (Offset en octet) de début de la copie.
- ID image de destination.
- Type des données de destination.
- Offset de destination.
- Longueur en octet de la copie.

Process:

- ID du *Process*.
- ID image source.
- ID image de destination.
- ID buffer générique source.
- ID buffer générique de destination

L'utilisateur concepteur de chaine enrichit une base de données de descriptions d'actions qu'il peut ensuite associer à une action via l'interface graphique.

Exemples de descriptions d'actions :

ImgCpy 36:
HOST_IMG_SRC1; U16; 0; GPU1_ARRAY0; U16; 0; 1000000

Process 37:
HISTO_Q1; HOST_IMG_SRC1; NULL; NULL; HOST_GPB_0

En reprenant l'exemple de la Figure 2, la Figure 3 montre la configuration de la chaine correspondante dans l'interface graphique.

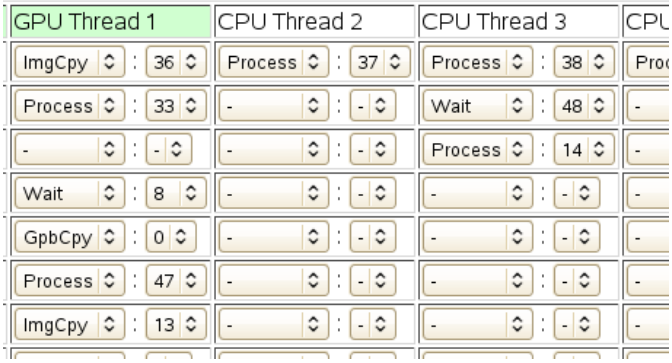

Figure 3 : Exemple : Configuration dans l'interface graphique d'une chaine traitement : correction de défauts optiques et rehaussement de contraste.

L'exemple de la figure 3 dans le détail :

GPU:

1) ImgCpy 36: importe la trame video d'entrée sur le GPU
2) Process 33 : correction d'objectif
4) Wait 8 : attente de la fin de l'étape CPU thread 3 : Process 14 (ie. le calcul de l'histogramme).
5) GpbCopy 0 : importe l'histogramme sur le GPU.
6) Process 47 : rehaussement de contraste.
7) ImgCpy 13 : Exporte l'image résultante pour la sortie vidéo finale.

CPUs :

1) Process 37 à 40 : calcul de 4 histogrammes partiels à partir de la trame video d'entrée.
2) Wait 48 : attente de la fin des Process 37 à 40.
3) Process 14 : somme des histogrammes partiels.

Une fois la chaine construite, l'utilisateur envoie la configuration de chaine à l'application de traitement vidéo en cours. Celle-ci reconfigure le tableau de fonctions de son moteur d'exécutions. La nouvelle chaine de traitements sera pris en compte dès le cycle suivant.

4 Application

La plateforme décrite dans cet article a pu être mise à profit pour concevoir, prototyper, développer et évaluer un grand nombre de chaines de traitements dans un contexte d'aide au pilotage afin de fournir une meilleure appréhension de l'environnement au pilote. La flexibilité de la plateforme a par exemple permis de tester très simplement plusieurs algorithmes de débruitage, de réétalement de dynamique, de détection, etc. et de recueillir les retours opérationnels nécessaires au réglage fin des traitements vidéo et à une réponse efficace aux besoins réels des pilotes. Il est important de noter qu'outre la modularité offerte pour construire rapidement de nouvelles chaines de traitements, la plateforme propose également des canaux d'interaction simples à mettre en œuvre : le pilote peut donc paramétrer en temps réel les algorithmes à l'aide de commandes de haut niveau, relayées dans le poste de pilotage.

5 Conclusion

La capacité de calcul de l'architecture matérielle hybride multi-CPU/GPU et la souplesse de l'architecture logicielle ont fourni un outil répondant aux objectifs d'évaluations en temps-réel et de prototypage rapide de nouvelles chaines.

Cependant, distribuer suffisamment efficacement les traitements sur ce type d'architecture a demandé un effort d'apprentissage important pour l'utilisateur. « Penser parallèle », en terme de développement informatique, n'est déjà pas aisé. « Penser hybride et parallèle » nécessite d'appréhender, en plus, de nouveaux paramètres tels que la pertinence de tel type d'UC pour tel algorithme, les temps de transfert des données, etc. La méthode empirique que nous utilisons montre rapidement ses limites pour résoudre le problème de distribution des tâches de chaines de traitements complexes. Ces architectures offrent de multiples ressources hétérogènes intéressantes et nécessite un système de gestion de ressources intelligent. Par la suite, il est envisageable de concevoir un solver automatique fournissant une solution optimale d'agencement de chaine de traitement.

6 Références

[1] G. D. Hines, Z.-U. Rahman, D. J. Jobson & G. A. Woodell, Real-Time enhancement, Registration and Fusion of a Multi-Sensor Enhanced Vision System, Proc. SPIE vol. 6226, pp. 622609-622609-8, 2006.

[2] M. I. Smith, A. Ball & D. Hooper, Real-Time Image Fusion: a Vision Aid for Helicopter Pilotage, Proc. SPIE vol. 4666, pp. 83-64, February 2002.

[3] General-Purpose Computation on Graphics Hardware (http://gpgpu.org).

[4] GPU for Vision (http://gpu4vision.icg.tugraz.at), Institute for Computer Graphics and Vision, Graz University of Technology.

[5] R. J. Hovland, Latency and Bandwidth Impact on GPU-Systems, Norwegian University of Science and Technology, December 2008.