$2^{\text{ième}}$ **année Magistère d'informatique**

PLATE-FORME EXPÉRIMENTALE MULTI-CAMÉRAS

Auteur :
Nicolas BOREL

Tuteur :
Lionel REVERET

Le 23 septembre 2002

Remerciements

Je tiens à remercier en premier lieu toute l'équipe MOVI de l'INRIA Rhône-Alpes, en particulier Radu Horaud,son directeur, de m'avoir proposé ce stage au sein de l'INRIA et par la même occasion de m'avoir permis de découvrir le monde de la recherche.

Je remercie Lionel Reveret et Edmond Boyer pour m'avoir accueilli chaleureusement et m'avoir suivi avec attention tout au long de ce stage.

Je remercie également David Bourguignon et Marc Lapierre pour leurs aides précieuses. Leurs coopérations ont été très utiles et m'ont beaucoup aidé à résoudre les nombreux problèmes techniques du stage.

L'accueil de l'équipe MOVI et de l'équipe du Service Robotique fut très agréable et ce fut un réel plaisir de participer à un tel projet.

Je remercie aussi tous les stagiaires MOVI, IMAGIS et du SERVICE ROBOTIQUE, pour les moments de détente durant les pauses.

Table des matières

Remerciement **3**

1 Introduction **11**
1.1 Lieu de stage . . . 11
1.1.1 INRIA . . . 12
1.1.2 INRIA Rhône-Alpes . . . 12
1.1.3 Les projets MOVI et IMAGIS . . . 12
1.2 Sujet de stage . . . 14
1.3 Buts du stage . . . 15

2 La technologie Firewire **17**
2.1 Présentation de la technologie Firewire . . . 17
2.2 Avantages et inconvénients du Firewire . . . 21

3 Evaluation de la situation initiale **23**
3.1 Situation par rapport à l'existant . . . 23
3.2 Le matériel et les logiciels utilisés . . . 23
3.3 Évaluation de la situation initiale . . . 26
3.3.1 Pourquoi coriander ne suffit pas ? . . . 27
3.3.2 Pourquoi la lib DC1394 ne suffit pas ? . . . 27
3.3.3 La solution que je propose . . . 27
3.3.4 Architecture logicielle . . . 28
3.3.5 Pourquoi le DMA . . . 28
3.3.6 La question de l'affichage . . . 28

4 Travail effectué et résultats obtenus **31**
4.1 Recherche de documentation . . . 31
4.2 Installation du système . . . 31
4.3 Conception et programation . . . 32
4.3.1 Obtention d'information . . . 33
4.3.2 Initialisation du système . . . 34
4.3.3 Acquisition et capture d'image . . . 35
4.3.4 Évolution future . . . 38

5 Les résultats obtenus **39**
5.1 Choix de programmation . . . 39
5.1.1 Méthode de mesure . . . 39
5.1.2 Les résultats et leurs analyses . . . 40
5.1.3 tests sans les threads . . . 41
5.1.4 tests avec les thread . . . 42
5.2 Synchronisation local . . . 46
5.2.1 Méthode de mesure . . . 47
5.2.2 Les résultats et leurs analyses . . . 47
5.3 Synchronisation réseau . . . 51
5.4 synchronisation avancé . . . 52
5.4.1 synchronisation de 6 caméras . . . 52
5.4.2 synchronisation de 5 caméras . . . 53
5.4.3 Synchronisation de 6 caméras dans une nouvelle configuartion . . . 54
5.4.4 Synchronisation avec plus de 6 caméras . . . 55

6 Modèles de couleurs et calibration de caméra **57**
6.1 Les modèles de couleurs . . . 57
6.1.1 L'espace RGB . . . 57
6.1.2 L'espace YUV . . . 58
6.1.3 Utilisation des différents modèles de couleurs . . . 58
6.2 calibration de caméra . . . 59
6.2.1 Modélisation d'une caméra . . . 59
6.2.2 Méthode de calibration . . . 60
6.2.3 Protocole de calibration . . . 61
6.2.4 Amélioration apporté au protocole de calibration . . . 63
6.2.5 Limites de cette méthode . . . 63
6.2.6 Procédure de calibration . . . 64

7 Évolutions future et bugs connus **65**
7.1 Évolutions future . . . 65
7.1.1 Initialisation du système de capture . . . 65
7.1.2 Affichage des caméras . . . 66
7.1.3 Acquisition et synchronisation de plusieurs caméras . . . 66
7.2 Bugs connus . . . 67
7.2.1 Initialisation du système de capture . . . 67
7.2.2 Fermeture du système de capture . . . 68
7.2.3 Affichage des caméras . . . 68
7.2.4 Synchronisation . . . 68
7.2.5 Sauvegarde . . . 68
7.2.6 Les différentes bibliothèque annexes . . . 68

Conclusion **71**

ANNEXES **75**

A Fichier de configuration **75**

B Mode d'emploi de l'installation **83**

C Liste et description des exemples **87**

D Documentation de la librairie util_1394 **91**

Table des figures

1.1 *Le projet MOVI* 12
1.2 *Le projet IMAGIS* 14

2.1 *le logo FireWire* 17
2.2 *schéma technique* 18
2.3 *Exemple de topologie de réseau FireWire* 20
2.4 *Les connecteurs des câbles* 21

3.1 *Caméras Dragonfly et l'unité de synchronisation* 24
3.2 *Caméra SONY DFW-VL500* 24
3.3 *Système d'acquisition* 25
3.4 *schéma de l'organisation software* 25

4.1 *La lib util_1394 dans le système Linux.* 32
4.2 *Organisation de la lib util_1394* 33
4.3 *Organisation du compteur Firewire.* 37

5.1 *Système de capture avec 1 caméra et 1 PC* 40
5.2 *Système de capture avec 2 caméras et 1 PC* 43
5.3 *Installation de test de la synchronisation* 48
5.4 *Résultat de la synchronisation (1 point sur la grille)* 49
5.5 *Résultat de la synchronisation (plusieurs points sur la grille)* 50
5.6 *Système de capture avec 4 caméras et 2 PC* 52
5.7 *Système de capture avec 6 caméras et 3 PC* 53
5.8 *Système de capture avec 5 caméras et 3 PC* 54
5.9 *Système de capture avec 6 caméras et 4 PC* 55

6.1 *Le cube RGB* 57
6.2 *Schéma d'une caméra avec la distance focale et la projection de l'axe optique* 60
6.3 *Exemple de passage du repère 1 au repère 2* 61
6.4 *La mire de calibration* 61
6.5 *Détection des coins du damiers sur la mire de calibration* 62
6.6 *Schéma de la mire avec l'origine* 63

Chapitre 1

Introduction

Le domaine de la vision par ordinateur est actuellement en pleine expansion. L'augmentation de la puissance des ordinateurs et la baisse des prix des caméras numérique permet maintenant de pouvoir faire de l'acquisition sur un simple PC et non plus sur une station dédié.

La reconstruction 3D consiste à déterminer la position dans l'espace des objets à partir d'images de la scène. A partir d'un minimum de 2 images, nous pouvons calculé les coordonnées dans l'espace d'un point visible dans les 2 images. Pour pouvoir faire ce calcul, il faut bien entendu que les 2 images aient été prises en même temps. Donc il faut que notre bibliothèque puisse capturer des images synchronisées sur plusieurs caméras en même temps. Plus on possède d'image, plus le calcul des coordonnées dans l'espace sera précis et robuste. De plus filmé une scène complexe sous plusieurs angles différents (plus de 2) permet de gérer le problème des occultations (un point de la scène est visible depuis une caméra mais pas depuis une autre). C'est dans cette optique que s'inscrit ce stage, il doit permettre de donner une infrastructure aussi bien software que hardware aux autres ou futurs projets qui traitent de la vision par ordinateur, de la reconstruction 3D et de la capture de mouvement. Ce stage doit évaluer les posibilité de l'acquisition sur PC, ainsi que les possibilité de synchronisation des caméras Firewire à une vitesse de 30 images par seconde.

1.1 Lieu de stage

Mon stage s'est déroulé à l'INRIA Rhône-Alpes au sein de l'équipe de recherche MOVI en coopération avec l'équipe IMAGIS. Je vais tout d'abord présenté l'INRIA. Puis je m'intéresserai à l'unité de recherche INRIA Rhône-Alpes et enfin aux projets de recherche MOVI et IMAGIS.

1.1.1 INRIA

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

1.1.2 INRIA Rhône-Alpes

Créé en décembre 1992, l'INRIA Rhône-Alpes est la plus récente des cinq unités de recherche de l'INRIA. Elle a son siège à Montbonnot, dans un bâtiment qui accueille aujourd'hui plusieurs projets de recherche répartis en quatre pôles qui sont : maîtriser les systèmes et réseaux informatiques, aider à la conception et à la création, percevoir,simuler et agir et enfin modéliser les phénomènes complexes.

C'est sous la responsabilité de Radu HORAUD, Edmond BOYER et de Lionel REVERET que s'est déroulé mon stage. Radu HORAUD est directeur de recherche du projet MOVI, Edmond BOYER est enseignent-chercheur au sein de l'équipe MOVI et Lionel REVERET est chargé de recherche au sein de l'équipe IMAGIS.

1.1.3 Les projets MOVI et IMAGIS

Le projet MOVI

Le projet MOVI (Modélisation pour la Vision) s'articule autour d'un thème majeur. Il s'agit de modéliser des objets géométriques avec pour objectifs leur reconnaissance, leur localisation et leur interprétation en utilisant des images ou des séquences vidéo. L'équipe de MOVI est composée de 6 chercheurs permanents et d'une dizaine de thésards

FIG. 1.1 – *Le projet MOVI*

et post doctorats. En période de fin d'année scolaire une dizaine de stagiaires viennent généralement compléter les rangs de l'équipe.

Présentation du laboratoire :

Le projet MOVI vise à apporter des solutions générales et fondamentales permettant d'identifier, localiser, mesurer des formes vues à travers une ou plusieurs caméras, et de valider ces solutions à travers des applications diverses allant de la robotique jusqu'à la consultation de bases d'images. Les outils de base développés au sein du projet sont la géométrie de la perception tridimensionnelle, les invariants associés à cette géométrie, la mise en correspondance image - image ou image - modèle. MOVI est une équipe de recherche qui fait partie du laboratoire GRAVIR-IMAG . Ce dernier regroupe cinq équipes travaillant dans le domaine de la robotique, de l'imagerie et de la vision par ordinateur et dont les instances de tutelle sont le CNRS, l'INRIA Rhône-Alpes, l'INPG et l'UJF.

Axes de recherche :

- *La géométrie des images multiples* : Il s'agit de la modélisation des phénomènes géométriques liant la perception de l'espace dans plusieurs vues. Un domaine d'application est l'étalonnage des caméras maintenant bien maîtrisé, ou l'autocalibration permettant de travailler sans mire.
- *Le couplage perception-action* : L'objet ici est l'intégration des différents aspects perception-action et l'étude des nouveaux problèmes posés comme la commande référencée par la vision dans le cadre de capteurs mal ou pas étalonnés.
- *La mise en correspondance* : Ce problème basique est exploré selon deux directions : entre images afin d'assurer les mises en correspondance pour des paires stéréoscopiques, et entre modèles et images.
- *L'indexation d'ensembles d'images par le contenu* : L'objectif est d'associer à des images des descripteurs permettant de les retrouver facilement, même à partir de fragments. Le domaine d'application privilégié est la recherche dans des bases d'images ou dans des vidéos.

Le projet IMAGIS

Le projet iMAGIS travaille dans le domaine de l'informatique graphique et de la synthèse d'images. iMAGIS développe des outils permettant de concevoir, puis d'utiliser dans le cadre d'applications de taille significative, et en particulier en vue de simulation, des maquettes numériques 3D. Ces maquettes peuvent être purement géométriques ou posséder également des propriétés « physiques » (photométriques ou mécaniques par exemple). Les applications visées se situent dans des domaines très divers (construction automobile ou aéronautique, urbanisme, éclairagisme, bâtiment, téléphonie mobile, chimie, chirurgie assistée, agronomie, environnement, audio-visuel, etc.). Dans bien des cas, il s'agit de concevoir les techniques (modélisation et algorithmes graphiques) sur lesquelles reposent les systèmes « de réalité virtuelle » (ou « de réalité augmentée ») qui commencent à voir le jour. Le défi à relever est de leur fournir la puissance nécessaire « temps réel » qui les caractérise.

iMAGIS est un projet commun avec le CNRS, l'INPG et l'Université Joseph Fourier.

FIG. 1.2 – *Le projet IMAGIS*

Axes de recherche :

- Visualisation d'environnements complexes.
- Rendu réaliste, simulation de l'éclairage (utilisation de modèles complexes de réflectance, éclairage global, techniques de radiosité hiérarchiques, environnements dynamiques, etc.).
- Animation, modélisation d'objets déformables et de leur comportement.
- Simulation d'éboulements, études de « risques ».
- Synthèse d'images réalistes.
- Plate-forme pour l'animation et la réalité virtuelle.
- Algorithmique de la visibilité, structures de données efficaces pour le rendu de scènes très complexes (prise en compte de la cohérence, cas dynamique, etc.).
- Interactivité, réalité augmentée (manipulation et partage d'objets virtuels, intégration image-son, etc.).

1.2 Sujet de stage

Le sujet de stage a été proposé par Radu Horaud, le directeur du projet MOVI. Ce sujet était initialement un sujet de DEA. Tout d'abord nous donnerons l'intitulé exacte du sujet du stage, puis nous expliquerons les différents buts que ce stage cherche à atteindre.

Intitulé du sujet :

Plate-forme expérimentale multi-caméras

La reconstruction en vision par ordinateur de scènes et de mouvements tridimensionnels nécessite de mettre en place un réseau de plusieurs caméras parfaitement synchronisées. Le but de ce stage consistera à mettre en œuvre une telle plate-forme de capture à base de plusieurs caméras numériques de type IEEE 1394 (FireWire). Après installation des caméras, il sera nécessaire de valider expérimentalement les possibilités et la qualité de synchronisation. Dans le cadre d'une architecture existante d'application interactive pour la réalité virtuelle, une intégration logicielle devra être développée entre la bibliothèque de capture et des algorithmes de calibrage permettant de déterminer les positions relatives des caméras. Une application de suivi 3D d'un marqueur permettra de valider l'ensemble de la plate-forme.

1.3 Buts du stage

Le stage cherche à répondre a plusieurs buts différents, qui seront traités lors des différentes phases composant le projet.

Voici les différentes phases qui composent le stage :

- 1) Mettre en oeuvre physiquement une plate forme de capture constituée de 4 caméras numériques, comme chaque carte d'acquisition ne comporte que 2 entré, nous installerons chaque carte dans un PC différent.
- 2) Les différents constructeurs de caméra Firewire fournissent des bibliothèque d'utilisation de leurs caméras sous environnement Windows. Nous avons décidé que notre plate forme de capture serait sous environnement Linux, il faut donc écrire une bibliothèque pour utiliser les cartes et les caméras FireWire. Pour pouvoir réaliser l'étape précédente, il faut écrire ou compléter les drivers existant sous Linux pour la gestion des cartes FireWire. Notre plate-forme de capture doit être sous Linux pour s'intégrer avec des applications de réalité virtuel (VR Juggler) de l'équipe APACHE de l'INRIA.
- 3) Valider cette plate-forme expérimentale utilisant notre bibliothèque de capture.
- 4) Pouvoir réutiliser cette bibliothèque dans d'autres applications.

En conclusion, l'objectif de ce stage est de créer une bibliothèque gérant une plate-forme constituée par 4 caméras numériques. Cette bibliothèque aura les fonctions suivantes :

- Initialiser les cartes firewire.
- Initialiser les caméras, régler les paramètres internes (balance de blanc, luminance, zoom, taille de l'image, etc...).
- Obtenir un affichage simultané en temp réel sur un même écran de PC des 2 caméras connectées.
- Possibilité de sauvegarder l'ensemble des images obtenues par les deux caméras présente sur un même PC.
- Capturer le flux vidéo des 2 caméras en mémoire et guarantir la synchronisation à la cadence maximal (15 ou 30 fps).
- Gestion du rapatriment mémoire vers disque dur et mesure de la perte de cadence d'acquisition.
- Synchroniser 2 système (PC + 2 caméras) grace à un dispositif matériel fournis par le constructeur et mesurer ainsi sa capacité à synchroniser 4 caméras.
- **les différentes caméras doivent être parfaitement synchronisées entre elles, pour faire l'acquisition des images en même temps. Ceci pour deux caméras sur un même PC comme pour des caméras reliées par un réseau Firewire**.
- Calibrer les caméras (connaitre les paramètres intrinsèque et extrinsèque de chaques caméras).

Chapitre 2

La technologie Firewire

Je donnerai dans ce chapitre un petit aperçu de la technologie Firewire, son mode de fonctionnement, ses avantages et ses inconvénients.

2.1 Présentation de la technologie Firewire

Historique La technologie 1394 a commencé à être développée en 1986 par les ingénieurs d'Apple qui ont choisi le nom "FireWire" en raison de son important débit. La première spécification de FireWire est sortie en 1987, elle a été adoptée en 1995 par le consortium IEEE (Institute of Electrical and Electronics Engineers) en tant que standard identifié par le nombre 1394 (figure 2.1, page 17).

FIG. 2.1 – *le logo FireWire*

On pouvait déjà obtenir des débits de 100mb/s en 1995, puis à partir de 1996 on a pu obtenir des débits de 200 et 400mb/s. En 2000 la norme 1394 a été mise à jour sous le nom de 1394a, et sera de nouveau mise à jour sous le nom de 1394b. Cette version de la norme permettra d'atteindre des débits allant jusqu'à 800, 1600 et 3200 Mb/s.
Comparaison avec d'autres technologies :
SCSI : 320 Mb/s au maximum (vitesse théorique atteinte avec les derniers modèles)
USB (version 1.0) : 12 Mb/s au maximum
USB (version 2.0) : 480 Mb/s au maximum
FireWire (spécification 1996) : 400 Mb/s au maximum
FireWire (spécification 2001) : 3200 Mb/s au maximum

Cette norme existe sous plusieurs dénominations différentes : FireWire pour APPLE, iLink sur les appareils SONY grand public par exemple.

Depuis Juin 2002, tout les appareils utilisant cette norme, peuvent utilisé le nom Firewire, qui devient ainsi le terme générique pour désigner cette technologie.

L'USB est dévellopé conjointement par Microsoft et par Intel.

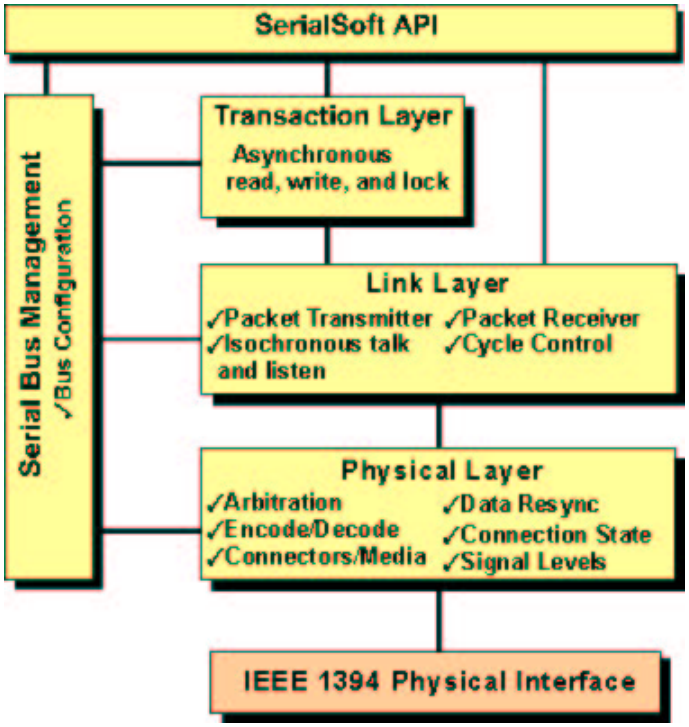

FIG. 2.2 – *schéma technique*

Description du point de vue réseau : La technologie 1394 est avant tout une norme réseau à l'instar d'Ethernet ou de TPC/IP. En s'appuyant sur le modèle OSI, 1394 s'étend sur 3 niveaux : le niveau physique, le niveau liaison et le niveau réseau (figure 2.2, page 18).

Niveau physique : couche PHY : La couche PHY est chargée de gérer le signal de mise sous tension à distance, la reconnaissance du signal de sélection de l'appareil, le signal d'initialisation du bus et la réception/émission des données. La vitesse de transmission du bus 1394 dépend de l'électronique de la couche PHY. Généralement cette couche supporte des débits de 100, 200 et 400Mb/s

Pour se rendre mieux compte de ce débit, prenons un exemple de vidéo conférence. Six standards ont été choisis pour le 1394, basés sur différentes considérations : la taille des images, le débit et la taille des paquets dépendant du mode de l'image. Pour un standard

de haute qualité, on transmet 30 images par seconde (l'œil humain n'en perçoit que 25). Chaque image a une résolution de 640x480 pixels et chaque pixel est codé sur 24 bits (ce qui représente à peu près une palette de 16 millions de couleurs). Cette qualité exige un taux de transfert de 240Mb/s. On comprend donc aisément que ce bus est de plus en plus utilisé pour le matériel vidéo vu la possibilité d'atteindre une très haute qualité.

Niveau liaison Ce niveau s'occupe de gérer l'envoi et la réception de paquets de données. Elle définit deux modes de communication : asynchrone et isochrone.

Le mode de transmission asynchrone *Le mode de transmission asynchrone* garantit la bonne réception des données car la cible envoie, dès la réception d'un paquet, un signal de reconnaissance immédiatement à l'envoyeur. Ce temps de latence ne peut pas être quantifié car il dépend du taux d'utilisation du bus 1394 par d'autres transmissions pour d'autres appareils communicant entre eux. Ce paquet de données peut être envoyé à une adresse d'un appareil connecté au réseau ou à toutes les adresses. Par contre, on ne peut envoyer un paquet à une branche (sous-ensemble) du réseau.

Le mode de transmission isochrone *Le mode de transmission isochrone* est différent. Il réserve, pour la transmission, un espace-temps de dimension particulière et cyclique, toutes les 125μs (soit 8000 cycles isochrones par seconde). Depuis un appareil, un espace-temps isochrone est garanti. Les communications isochrones sont prioritaires aux asynchrones de sorte que la bande passante pour les communications isochrones est assurée. Ainsi, la communication isochrone entre deux appareils ou plus est assimilable à un canal. Une fois qu'un canal a été établi, l'appareil demandeur est garanti d'avoir l'espace-temps demandé à chaque cycle.Non seulement un appareil peut émettre sur son canal mais ce canal peut être reçu par plusieurs appareils. Plusieurs canaux peuvent être réservés pour un seul appareil et des canaux supplémentaires peuvent lui être ajoutés tant que la capacité d'émission isochrone de l'appareil n'est pas atteinte et que la bande passante isochrone du bus 1394 est disponible. C'est ce mode de transmission que l'on choisit pour le transport de données vidéo ou toutes autres données qui ont besoin d'avoir une transmission garantie en "temps réel".

Niveau réseau : Ce niveau s'occupe de gérer la transmission des différentes données montantes et descendantes à travers le réseau d'appareil FireWire que l'on peut avoir.

Parallèlement à ces trois couches, il subsiste une couche de gestion de bus série. Cette couche s'étend sur les trois niveaux précédents et s'occupe de gérer le bus et de le configurer.

Topologie du réseau Le bus 1394 accepte plusieurs sortes de topologie pour son réseau : en chaîne, en arbre, en étoile ou une combinaison de toutes ces topologies (figure 2.3, page 20). La seule restriction est qu'il ne faut pas avoir plus de 16 cables entre deux appareils. On peut connecter à un tel réseau 63 appareils.

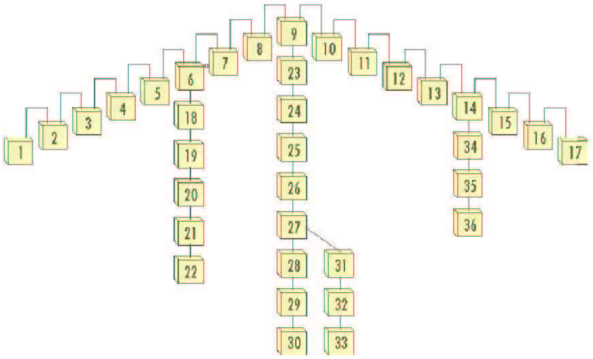

FIG. 2.3 – *Exemple de topologie de réseau FireWire*

La transmission L'IEEE 1394 utilise une technologie de transmission de données par paquets qui est organisée comme si c'était un espace mémoire interconnecté avec plusieurs appareils. La topologie d'un réseau I/O 1394 est constituée de deux couches : une couche physique (communément appelée PHY) et une couche de lien. Il y a également deux couches logiciel : une couche de transaction et une couche de gestion du bus série. Ces couches logiciel peuvent être directement intégrées en "hard". La couche PHY doit gérer le signal de mise sous tension à distance, la reconnaissance du signal de sélection de l'appareil, le signal d'initialisation du bus et la réception/émission des données. La couche de lien formate les données en paquets pour la transmission via le câble 1394 et supporte les modes de communication asynchrone et isochrone.

Description du point de vue bus série : Le bus 1394 en plus d'être une norme de réseau est avant tout considéré comme un bus série rapide. Avec l'arrivée de nouveaux composants, grands consommateurs de bande passante, tel que le DVD ou la vidéo numérique, le besoin de ce type de bus s'est trouvé précipité. Mais également le besoin d'un standard.

Pour pouvoir devenir un standard, le bus 1394 se devait d'être innovant et performant. Il supporte de nombreuses technologies dorénavant indispensables.

Le câble Deux sortes de câble sont actuellement disponibles.

Le câble original est constitué de 6 fils. Deux paires torsadées sont réservées au transport des données. Une autre paire torsadée, optionnelle, est réservée à l'alimentation.

Le nouveau standard est un câble à 4 fils, dédié à la vidéo et aux applications nécessitant une bande passante encore plus importante. Il est constitué de deux paires torsadées transportant les données. La paire torsadée pour l'alimentation a été supprimée sans doute pour diminuer les perturbations électromagnétiques, ainsi on peut monter plus haut en fréquence, sans être limité par le bruit.

Les périphériques sont reliés entre eux par un câble souple (3 paires torsadées) d'une longueur maximale de 4,5 m.

Le bus FireWire peut fournir l'énergie pour les appareils connectés : 24 volts, 15 Watts de puissance. Les appareils les plus gourmands en électricité doivent apporter leur propre alimentation.

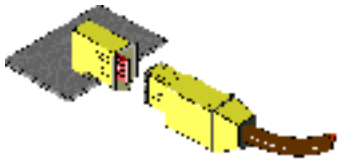

FIG. 2.4 – *Les connecteurs des câbles*

Les connecteurs des câbles Les connecteurs du bus 1394 sont dérivés des connecteurs de la Game Boy (tm) de Nintendo. Petit et flexible, ils sont adaptés à une utilisation intensive et sont durables. Leur forme les rend inoffensifs vis à vis des enfants de tous âges. Il est important de préciser qu'il ne nécessite pas de terminateur (ou bouchon) en bout de câble comme avec le SCSI (figure 2.4, page 21).

Hot Plug'n Play On peut connecter et déconnecter à volonté un appareil lorsque le bus 1394 est sous tension. L'appareil en question est automatiquement initialisé. Un "id" lui est attribué en guise d'adresse et ainsi il est prêt à être utilisé. C'est probablement la caractéristique la plus intéressante car elle ne nécessite pas de redémarrer la machine lorsque l'on rajoute ou enlève un appareil. De plus, ce bus permet le dialogue des appareils entre eux. Il n'y a pas de maître !

Architecture mixte Le bus 1394 peut mélanger plusieurs débits différents sans aucun problème. Ainsi la nouvelle norme 1394b qui permettra des débits allant jusqu'à 3.2Gb/s sera compatible avec des périphériques à 100Mb/s sans pour autant pénaliser les autres périphériques plus rapides.

haut débit 1394 est probablement le bus série le plus rapide actuellement, il dépasse même la technologie SCSI qui est réputée pour ses débits rapides. Ce bus est probablement destiné à remplacer le SCSI ainsi que d'autres sortes de bus dans les années à venir.

non propriétaire La technologie n'est la propriété d'aucune société ou organisme, ce qui signifie que n'importe qui peut commercialiser un appareil 1394 sans verser de royalties. Ainsi cette technologie est destinée à se diffuser rapidement.

2.2 Avantages et inconvénients du Firewire

Juqu'à assez récemment le Firewire était la norme permettant la vitesse la plus grande possible. Il n'a été dépassé que très récement par l'USB, dépassé de facon temporaire

puisque une nouvelle version du Firewire devrait sortir prochainement.

Le Firewire est beaucoup plus stable même sous un environnement Linux que l'USB.

Le Firewire a été dévellopé initialement pour les caméras numérique, ses évolutions successives lui on permit de gérer de plus en plus de périphérique différents, alors que l'USB a été créé dès le départ pour géré le maximum de périphériques numérique différents (caméra numérique, souris scanner, etc).

Le Firewire est une norme créé par Apple et géré par un organisme international l'IEEE, qui est chargé de mettre dans le domaine public les spécification du Firewire. alors que l'USB est une norme dévellopé par deux entreprise (Microsoft et Intel) qui ne rendent pas public toutes les spécifications de l'USB.

Chapitre 3

Evaluation de la situation initiale

3.1 Situation par rapport à l'existant

Programmes fonctionnant : La technologie 1394 étant relativement récente dans le monde de Linux, il n'existe pas encore de programmes complètements fonctionnels. Néanmoins plusieurs projets ont le mérite d'exister et de progresser régulièrement.

Coriander est un programme open source permettant de régler et visualiser les images que prend une caméra numérique ainsi que de sauvegarder des captures sur le disque dur. Le projet a débuté en janvier 2001 et en est toujours au stade expérimental.

XVision permet d'afficher à l'écran une vidéo où le flux que reçoit une caméra numérique. Néanmoins il ne semble pas fonctionner correctement avec certaines caméra et semble trop dépendant d'un matériel spécifique.

Programmes nécessitant des modifications : Il existe de nombreux logiciels d'édition et de traitements vidéo sur Linux mais ils fonctionnent tous en utilisant l'interface standard " **Vidéo4Linux** ". Ce standard permet d'utiliser et de traiter des vidéos quel que soit le type de caméras analogiques. A l'heure actuelle les caméras numériques que j'utilise ne supportent pas ce type d'interface. En conséquence si l'on souhaite utiliser tout de même un de ces logiciels, il faudra procéder à de nombreuses modifications. Il n'a pas été jugé intéressant de faire ces modifications en raison de la complexité de la tâche.

Broadcast 2000 est un logiciel d'édition de vidéo utilisant l'interface Vidéo4Linux.

3.2 Le matériel et les logiciels utilisés

matériel disponible : Il s'agit ici de développer une application reposant sur du matériel accessible, c'est à dire peu cher et facile à obtenir. Je disposais d'une machine PC dont le processeur est cadencé à 1 Ghz. L'implantation se fait sous un système d'exploitation LINUX. J'utiliserais une carte interface 1394 de marque Orange Micro.

J'utiliserais plusieurs périphérique différente :

- 4 caméras noir et blanc DRAGONFLY de PointGrey Research (figure 3.1, page 24)
- 1 caméra couleur webcam de PYRO
- 1 camera couleur, la DFW-VL500 de SONY (c'est ce type de caméra qui doit être utilisé au final) (figure 3.2, page 24)
- Ainsi qu'un boitier de synchronisation de PointGrey Research, qui permet de synchroniser 2 carte Firewire installé sur 2 PC différent (figure 3.1, page 24).

FIG. 3.1 – *Caméras Dragonfly et l'unité de synchronisation*

FIG. 3.2 – *Caméra SONY DFW-VL500*

Logiciels : J'ai débuté le stage sans vraiment posséder d'outils logiciels en dehors des langages de programmations habituels. Mon premier travail a donc été de me documenter sur la technologie pendant quelques jours. J'ai ainsi pris connaissance d'outils existants correspondant à mes besoins.

Ces outils sont des bibliothèques écrites en C et quelques programmes le tout fonctionnant sous Linux. Ces outils sont organisés au sein du système Linux suivant une

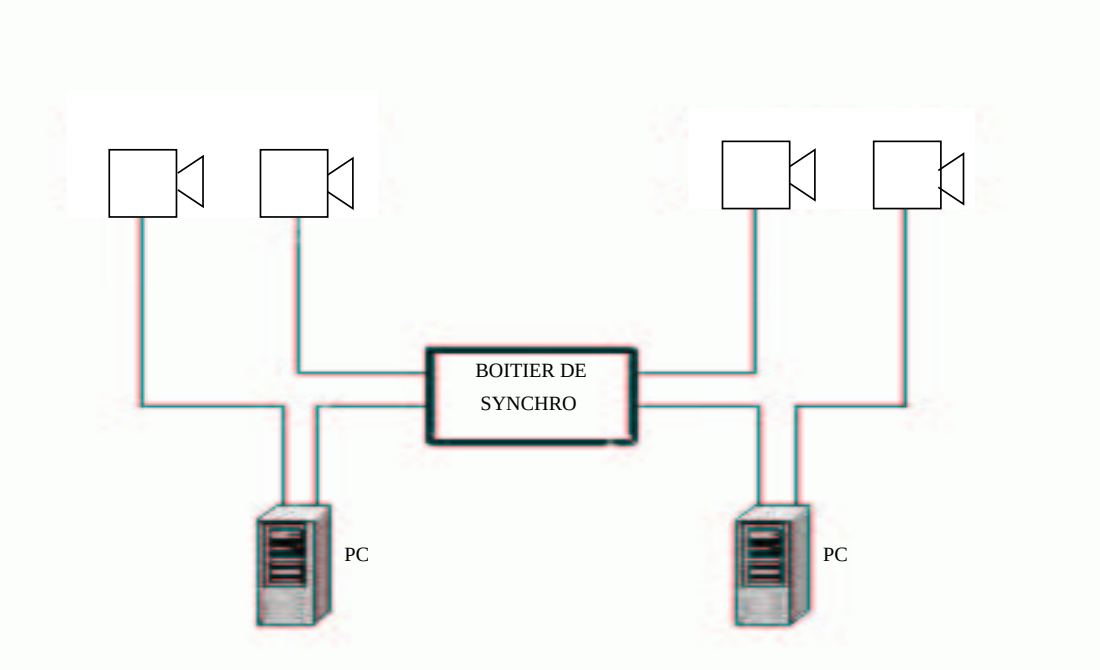

FIG. 3.3 – *Système d'acquisition*

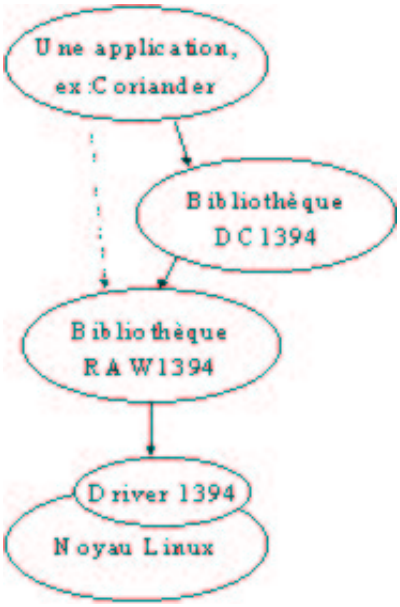

FIG. 3.4 – *schéma de l'organisation software*

arborescence particulière. Les flèches du schéma (figure 3.4, page 25) indiquent le sens des dépendances (une flèche pleine montre une forte dépendance, une flèches en pointillé indique une faible dépendance).

Noyau Linux et driver du bus 1394 : Le noyau de Linux correspond à la base du système d'exploitation. C'est un programme qui fait le lien entre le matériel et le logiciel. Le support du bus 1394 se fait au niveau du noyau de Linux. On peut choisir d'utiliser le bus 1394 sous forme de module ou de l'implémenter directement dans le noyau (comme on le montre le schéma). C'est généralement sous la forme de module qui est choisie car elle permet une plus grande flexibilité. Un module est un composant logiciel qui peut être chargé ou déchargé en mémoire sans avoir besoin de redémarrer la machine. Dans ce cas précis c'est intéressant étant donné l'état expérimental du bus 1394 sous Linux. Le noyau Linux est la pièce maitresse de toutes distribution Linux. Le driver du bus 1394 est maintenant compris dans tous les noyaux que l'on peut trouver sur les distributions les plus récentes. Le drivers 1394 ainsi que le noyau du système Linux est Open Source.

Bibliothèque RAW1394 : Pour pouvoir communiquer avec des appareils 1394 il faut passer par l'utilisation de la bibliothèque RAW1394. Elle s'occupe d'envoyer et de recevoir les données vers les périphériques, de gérer le bus, etc. Je ne me suis pas attardé sur le fonctionnement de cette bibliothèque qui est de trés bas niveau et trés compliquée. Cette bibliothèque commence à être présente dans les distributions, de linux sinon ses sources sont distribué sur Internet. Cette bibliothèque est Open Source.

Bibliothèque DC1394 : Cette bibliothèque est spécialisée dans la gestion de caméras numériques sur bus 1394 fournissant des vidéos non compressées. Elle possède un vaste ensemble de fonctions permettant d'initialiser les caméras, de régler certains paramètres tel que le focus ou le zoom, de faire des captures d'images. Mais dans l'ensemble son fonctionnement est relativement complexe, elle comprend environs 120 fonctions. Cette bibliothèque est Open Source et elle aussi commence a être presente dans les dernières distributions sinon ses sources sont distribué sur Internet.

Coriander Coriander est un projet Open Source (trouvé sur Internet), qui est à mon avis, le plus abouti, de tous les projets qui cherchent à utilisé des caméras Firewire. C'est ce logiciel qui nous a permit de tester nos caméras Firewire.

3.3 Évaluation de la situation initiale

Nous venons de voir que certains outils existaient sous Linux. Pourquoi avons nous eut besoin d'écrire cette bibliothèque, au lieu de réutilisé les logiciels qui existait déja. Ces outils ne nous conviennent pas car soit ils leurs manquent certaines fonctionnalitées, soit à cause de leurs dépendances à un matériel trop spécifique.

3.3.1 Pourquoi coriander ne suffit pas ?

Le logiciel coriander, est trés bien pour vérifier si des caméras marche ou non, mais il présente quelques défauts, et des fonctionalités sont manquantes.

- Impossibilité de faire marcher 2 caméra en même temps.
- Impossibilité de sauver 30 images/secondes au mieux 5 images par secondes pour une seul caméra.
- Impossibilité d'apparier temporellement les images venant de 2 caméra différentes (sur un même PC ou sur un réseau de PC).
- Impossibilité de calibrer les caméras.

3.3.2 Pourquoi la lib DC1394 ne suffit pas ?

DC1394 est une bibliothèque bas niveau de gestion de matériel et à ce titre est assez difficile à comprendre et à utiliser. De plus, cette bibliothèque ne comprend pas toutes les fonctionnalitées que nous voudrions que nos caméras Firewire puisse supporté (nous donnons une descriptions des fonctionnalité supplémentaire apporté par notre bibliothèque dans le paragraphe suivant).

3.3.3 La solution que je propose

Je propose d'écrire une bibliothèque nommé Util_1394, qui sera une surcouche des librairie DC1394 et RAW1394, et qui permettra de récupérer, afficher, sauver des images en provenance de caméras firewire. De plus cette bibliothèque aura des fonctionalitées supplémentaires :

- Sauvegarde des images récupérer sur chaque caméras en plusieurs mode différent (différents format graphique géré BMP, PPM, ...).
- Gestion multi-caméras (récupération, affichage et sauvegarde des images peuvent être réalisé en même temps sur plusieurs caméras).
- Chaque paramètres de chaque type de caméras est réglé par le biais d'un fichier de configuration (un fichier de configuration par type de caméra).
- On peut afficher en temps réel les images récupérées sur 1 ou plusieurs caméras sur le même écran de PC.
- On peut calibrer les caméras (connaitre les paramètres extrinsèque et les paramètres intrinsèque propre à chaques caméras).
- On peut apparier temporellement les images de 2 caméras différentes (sur un même PC).
- On peut apparier temporellement les images de 4 caméras différentes (brancher 2 par 2 sur 2 PC différents et relié entre elle par le boitier de synchronisation fournit par PointGrey le constructeur des caméras).
- On peut modifier en temps réels les paramètres propres à chaque caméras.
- Utilisation intensive du mode DMA (**D**irect **M**emory **A**cess) dans la décupération des images, pour alléger au maximum la chage du processeur central.

La solution que je propose s'articule autour de plusieurs choix de programmation différent (utilisation des threads et du mode DMA) et d'une architecture logicielle particulière (dut à l'utilisation intensive des threads).

3.3.4 Architecture logicielle

Comme notre bibliothèque, devait pouvoir gérer plusieur caméras en même temps, nous avons décidé d'utiliser des threads pour la récupération, l'affichage, et la sauvegarde des images provenant des différentes caméras. L'utilisation des threads permet une certaine modularité à la bibliothéque, ainsi la même version peut être utilisé pour gérer 1 ou plusieurs caméras. L'utilisation des threads permet d'utiliser au maximum les ressources processeur mis à ma disposition. Elle permet aussi de pouvoir faire marcher 2 caméras en parallèle, mais avec des vitesses d'acquisition différentes.

3.3.5 Pourquoi le DMA

L'utilisation du DMA a permit d'augmenter de facon trés significative la vitesse d'acqisition des images auprès des différentes caméras. On a les tableaux des résultats complets dans le chapitre résultats obtenus.

On a put constater que en utilisant le DMA, nous pouvions stocké 2 fois plus d'images que si on ne l'utilisait pas, dams le même laps de temps (dans le cas ou on utilisait pas les threads).

Le mode DMA (en anglais direct memory access) permet d'effectuer des transferts d'informations sans passer par l'utilisation du processeur. Ce type de mode permet dans ce cas précis de faire des captures d'images sans passer par le processeur. Le facteur de rapidité accrue est d'à peu près 150.

L'implémentation de ce mode a posé quelques problèmes au niveau de la configuration de Linux. Ce mode nécessite des options particulières dans le noyau qui n'ont pas été mentionnées dans la documentation que j'ai pu trouver. Par conséquent il ne fonctionnait pas sur mon ordinateur et j'ai mis beaucoup de temps avant de comprendre pourquoi. J'ai programmé l'intégralité de la bibliothèque sans ce mode pour ne pas perdre de temps et c'est vers la fin que j'ai trouvé la solution. J'ai donc procédé à plusieurs modifications pour intégrer le mode DMA tout en préservant l'ancien mode. La bibliothèque peut donc utiliser les deux modes.

3.3.6 La question de l'affichage

Pour afficher les images récupérées sur chaques caméras, plusieurs méthodes différentes on été essayé, je vais en donner la liste, ainsi qu'un bilan :

- **SDL** : (Simple Directmedia Layer), est une bibliothèque multimédia multi plateforme créé pour donner un accès rapide aux périphériques graphiques et aux périphériques sonores. Elle est utilisé par les players de vidéo, les émulateurs, et les

jeux vidéo sous Linux. SDL existe sous différente plate-forme : Linux, Win32, BeOS, MacOS, Solaris, IRIX, et FreeBSD. C'est la méthode qui donne les meilleurs résultats, le seul problème, mineur, c'est que l'on ne peut pas l'arreter brutalement. Si on affiche plusieurs caméras, elle s'afficheront dans la même fenêtre.

- **X11** : On utilise les fonctions d'affichage de XFree86 4.2 pour afficher les images, c'est une méthode qui demande un processeur puissant car on doit effectuer au minimum 1 conversion. Si on affiche plusieurs caméras, elles s'afficheront chacunes dans une feneêtre différente.
- **GLX** : C'est le mode OpenGL sous X. Il affichera chaque caméras dans la même fenêtre, en redimensionnant la fenêtre.
- **XV** : C'est une nouvelle fonctionalité de XFree84 version 4.2 qui permet d'afficher des images YUV sans conversion. C'est cette fonctionalité qui permet de voir des DVDs sous Linux. n ne peut visualiser qu'une seul caméra, si on en visualise deux un clipping trés important fait son apparition.
- **GDK** : Cette méthode n'a pas complètement été implenté.

Chapitre 4

Travail effectué et résultats obtenus

Le stage c'est déroulé en plusieurs étapes qui ont chacunes données lieu a un travail spécifique. On peut découper le travail en plusieur partie :

- Recherche de Documentation.
- Installation du système.
- Conception et programmation

4.1 Recherche de documentation

Le premier travail qui a été fait a été de trouver de la documentation. On recherchait de la documentation de plusieurs sortes :

- Les spécifications techniques du FireWire par le consortium IEEE.
- La documentation technique de chaque type de camera que nous utilisons. Pour connaître les modes d'acquisitions acceptés (quel format de sortie, noir et blanc ou couleur, taille de l'image), les options éventuellement disponibles sur la camera (zoom, auto-focus, etc ..)
- La documentation sur les différentes carte FireWire utilisées.
- La documentation sur les biliothèques DC1394 et RAW1394.

4.2 Installation du système

Il a ensuite fallu installer un système d'acquisition sur un ordinateur compatible PC, avec un système d'exploitation LINUX. Le système d'acquisition était composé d'une carte FireWire et de 2 cameras FireWire reliées à la carte FireWire. L'installation matérielle n'a pas posé de problème particulier, mais l'installation logicielle a posé de nombreux problèmes :

- Incompatibilité entre la carte et certaine distribution de LINUX (le système marche sous une distribution MANDRAKE 8.* mais bloque sur une distribution REDHAT 7.*)

- Selon la version du kernel utilisé par LINUX, le mode d'installation logicielle diffère (les nouveaux noyaux de la série 2.4.10 et supérieur, ont introduit un système de fichier pour les périphériques (DEVFS) qui reconnait automatiquement la carte et les caméras, alors qu'avec l'ancien système, il fallait les déclarer manuellement (création du fichier périphérique dans le répertoire DEV et association d'un drivers avec ce fichier systéme) ce qui pouvait entrainer des incompatibilités, voir le blocage du sytème)
- Les bibliothèques RAW1394 et DC1394 étant en phase de développement leurs spécifications ont changé plusieurs fois au cours du stage, ce qui a entrainé une mise à jour de la bibliothèque util_1394.

4.3 Conception et programation

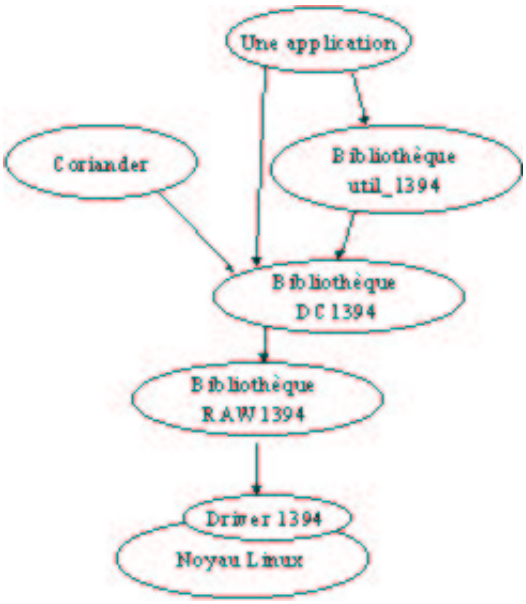

FIG. 4.1 – *La lib util_1394 dans le système Linux.*

La bibliothèque util_1394 Util_1394 est une surcouche de DC1394. Elle permet de faciliter et d'automatiser certaines tâches redondantes pour le programmeur. En reprenant le schéma (figure 3.4, page 25) et en le modifiant pour y ajouter util_1394 on voit qu'elle se situe au même niveau que le logiciel Coriander. C'est à dire qu'elle utilise directement les fonctions de DC1394. Par contre une application utilisant util_1394 sera disposée à un niveau supérieur. Plus une application est à un niveau élevé plus elle est écrite dans un langage simple à appréhender.

Conception de la bibliothèque Avant de se lancer dans la programmation, quel que soit le projet envisagé, une étape d'analyse et de conception est nécessaire. Elle permet de

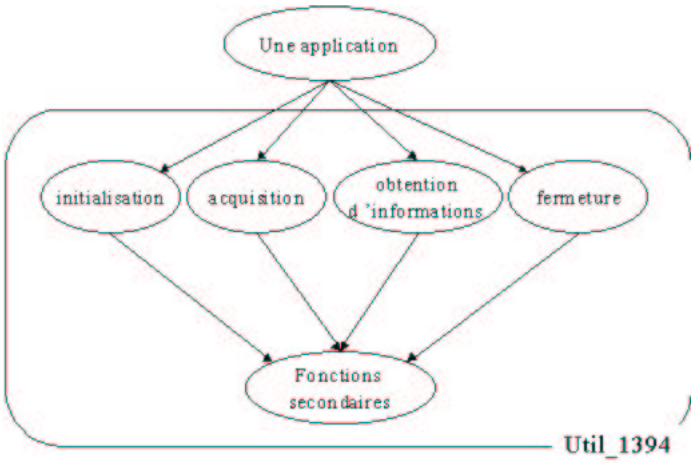

FIG. 4.2 – *Organisation de la lib util_ 1394*

bien cerner les objectifs et d'éviter beaucoup d'erreurs et donc de gagner du temps. Pour ce type de projet il n'a pas été nécessaire d'employer une méthode robuste tel que Merise ou UML étant donné que la réalisation de la bibliothèque n'était pas assez complexe.

Découpage en plusieurs parties :

Le découpage de la bibliothèque s'est fait en suivant l'ordre chronologique des actions à effectuer. Lorsque l'on veut utiliser les caméras,, on commence d'abord par initialiser les caméras ainsi que les variables qui recevront les images. Ensuite on fait l'acquisition d'une ou plusieurs images, puis on stoppe le programme en n'oubliant pas de libérer la mémoire utilisée. Il y a 2 modes d'acquisition différents, prendre une photo, ou prendre une vidéo. On peut regrouper ces opérations sous trois grandes parties qui sont initialisation, acquisition et fermeture du programme.

En plus de ces trois parties il est intéressant de pouvoir obtenir des informations techniques sur une caméra. De telles informations peuvent indiquer quel est le nom du vendeur et du modèle de la caméra, quel est son numéro de série mais aussi quelle résolution elle supporte et quel est son débit. Ce type de fonctions est réuni dans une partie intitulée obtention d'informations.

L'ensemble des fonctions de ces quatre parties utilise les bibliothèques mises à ma disposition mais ce n'est pas suffisant. Les fonctions " cachées" à l'utilisateur sont regroupées dans la partie intitulée " fonctions secondaires ".

Les dépendances des bibliothèques ne sont pas représentées sur le schéma mais il faut savoir que toutes les parties utilisent la bibliothèque DC1394.

4.3.1 Obtention d'information

Notre bibliothèque permet de retrouver toutes les informations possibles sur le systèmes que nous utilisons :

- Nombre de carte Firewire installé sur le système..
- Nombre de caméra branché sur chaque carte.
- Connaisance de tous les modes graphiques supporté pour une caméra donné.
- Connaissance de tous les options supporté (balance des blanc, zoom, ...) pour une caméra donné.
- Connaissance de tous les paramètres de la caméras (nom, nom du vendeur , etc...).

D'après les spécifications de IEEE, les caméras firewire peuvent avoir 19 options différentes, dont la liste à été définis par l'IEEE.

- *Brightness* : gère la brillance de l'image.
- *Exposure*
- *Sharpness* : gère le contraste de l'image.
- *White balance* : gère la balance des blancs pour des caméras couleur.
- *Hue* : gère la teinte de l'image.
- *Saturation* : quantité de couleur.
- *Gamma* : gestion du gamma.
- *Shutter* : gère la durée d'exposition de la prise d'une image.
- *Gain* : facteur de clarté.
- *Iris* : taille de l'iris de la caméra
- *Focus* : gère le focus
- *Temperature* : gère la température des couleurs.
- *Trigger* : permet d'utiliser un GBF externe pour fournir la vitesse d'aquisition.
- *Zoom*
- *Pan* : pour les caméra motorisé sert pour les panoramiques horizontaux.
- *Tilt* : pour les caméra motorisé sert pour les panoramique verticaux
- *Optical filter* : utilisé pour le Partial scan
- *Capture size* : utilisé pour le Partial scan
- *Capture quality* : utilisé pour le Partial scan

Chaque type de caméra, est géré par un fichier de configuration. Le fichier de configuration est personnalisé pour chaque type de caméra, les options indisponible sont marqué dans le fichier.

4.3.2 Initialisation du système

La division par partie est trop générale pour suffire à la réalisation. On peut identifier pour chaque partie plusieurs scénarios différents. Un scénario est un cas d'utilisation. Par exemple pour l'initialisation, l'utilisateur peut vouloir avoir une initialisation manuelle ou automatique. Ces deux manières d'initialiser sont des scénarios différents d'une même partie.

J'ai observé trois scénarios pour l'initialisation :

- Initialisation manuelle.
- Initialisation par paramètres.
- Initialisation par fichiers de configuration.

- *L'initialisation manuelle* consiste à demander à l'utilisateur pour chaque caméra ce qu'il souhaite utiliser comme configuration. Ce scénario est surtout utile lorsque l'on essayé une nouvelle caméra (maintenant ce scénario est obsolète par l'utilisation de fichier de configuration personnalisable).
- *L'initialisation par paramètres* permet de stocker directement dans le programme la configuration des caméras. C'est utile lorsque l'on est sûr d'utiliser la même caméra à chaque fois.
- *L'initialisation par fichier de configuration* est le scénario le plus utile. Il permet de stocker dans un fichier texte la configuration d'une caméra. A raison d'un fichier texte par type de caméra, le programme consulte ces fichiers et initialise les caméras en fonction de leurs contenus. Pour faire correspondre à une caméra un fichier texte de configuration, le nom d'un fichier de configuration est composé du nom du vendeur et du modèle de la caméra. En conséquence deux caméras de même modèle utiliseront le même fichier de configuration. Lorsqu'un fichier de configuration n'existe pas il est créé automatiquement avec une configuration, qui est sur de marcher. Le fichier de configuartion prend en compte toutes les caractéristiques des caméras, toutes celles définies dans les spécifications de l'IEEE. Si une caractéristique n'est pas présente sur une caméra, elle ne sera pas prise en compte lors de la lecture d'un fichier de configuration

4.3.3 Acquisition et capture d'image

En premier lieu, il faut différencié l'acquisition de la capture. L'acquisition est le procédé qui transfert l'image de la caméra à la mémoire de l'ordinateur. Au contraire la capture est le procédé qui transfert l'image de la caméra, à un support physique tel qu'un disque dur. La sauvegarde peut dans se cas très précis être considéré comme l'étape suplémentaire à appliquer à l'acquisition pour avoir la capture.

Pour gagner le maximum de temp, lors de la capture, la sauvegarde des fichier peut se faire en mode brute, ou en mode convertie. Dans le mode brute on stocke l'image sans la convertir, elle sera convertie lors d'un post-traitement effectué à la fin de l'acquisition.

Comme cette bibliothèque peut être utilisé, aussi bien pour prendre des photos, que des vidéos, 2 mode de capture-acquisitions-sauvegarde différents ont été créé.

A : Mode photo

Il a été, selon l'ordre chromologique, le premier créé, il permet de prendre une photos sur une caméra, ou sur toutes les caméras branchés sur un meme PC. Ce mode permet comme son nom l'indique de prendre une photos à un moment donné et de la sauver imédiatement sur le disque dur dans un fichier séparé pour chaques photos. Ce mode

permet aussi de prendre des ertstaz de vidéo. Mais les vidéo prise comme ceci ont beaucoup de défaut, il leur manque un certains nombres d'images, et les images la composant ne sont pas prise à intervalle régulier.

B : Mode vidéo

Ce mode à été créé pour pouvoir sauvegarder toutes les images qui sont acquise par une ou plusieurs caméra ou le couple fréquence d'acquisition et taille de l'image récupéré est le plus grand. A la fréquence d'acquisition, la plus élevé (30 images par secondes) dont nous disposons, nous n'avons accès qu'a un certain nombre de mode graphique. Ainsi en mode 640*480 niveaux de gris, chaque image pèse 307200 octets (640*480 = 307200) c'est le mode, qui donne les images les plus lourde, et qui accepte de marcher à 30 images par seconde. Ainsi pour 2 caméra on devrait stocké à chaque seconde près de 18432000 octets (640*480*30*2) soit 18.4 Mo. Pour pouvoir le réaliser, plusieur méthode ont été testé, chacune avec ses avantages et ses défauts.

- **Stockage séquentielle :** ont stocke chaque images récupérer dans un fichier différents. Cette méthode ne donne pas de trés bon résultats à cause du temps perdu pour la création et la fermeture du fichier. Cette méthode ne permet pas de dépassé une vitesse de 7.5 images secondes sur chaque caméras.

- **Stockage dans la RAM :** C'est une méthode qui permet toujours de sauvegarder toutes les images issus des 2 caméras et ceux quelque soit la vitesse employé, son seul défaut est qu'on est limité à 5 seconde de vidéo pour 2 caméras. Un traitement ultérieur permet d'extraire les différentes images du buffer mémoire et de les stocker après une éventuelle conversion sur le disque dur.

- **Stockage fichier caméra :**. En faite on crée un fichier temporaire pour chaque caméra, et ont stocke temporairement les images dans ce fichier. A la fin de la capture, un traitement permet déxtraire les images de chaque fichier temporaire. Ce mode permet en connectant 2 caméras, de sauver entre 29 et 30 images par seconde, selon l'occupation du système. en faite dans ce mode la limite est la taille maximal, que peut avoir les fichiers temporaires (en ext2 la taille maximal est de 2 Go).

- **Stockage dans 1 fichier unique :** On reprend le meme principe qu'avant à part que les images sont stocké dans un seul fichier temporaire. Ceci permet de perdre moins de temps lorsqu'on accède au fichier temporaire, mais le facteur de la taille devient un problème plus pressant. Pour différenciée les images, on les enregistre avec leur numéro de caméra. Ce mode permet parfaitement de sauver 30 images par seconde sur 2 caméras, mais il peut manquer des images selon l'occupation du systèmes.

Dans le mode vidéo, l'acquisition se fait en parallèle de la sauvegarde, ainsi si pour une raison quelconque, la sauvegarde ne parvenait pas à se faire dans les temps, l'acquisition se ferait malgrés tous et ceux quelque soit l'état de la sauvegarde.

Une fois que l'on peut sauver des vidéo, le nouveau problème qui se pose est de synchroniser les différentes images que nous capturons. Le problème est de trouver l'image qui a été prise en meme temps sur une autre caméra.

C : Implémentation du compteur

Ce problème est en faite double, il faut d'abord trouver une horloge qui est commune au deux caméra, et retrouver grace au temps pris sur cette caméra, l'image correspondante prise sur une autre caméra. Heureusement les cartes firewire possède **une horloge indépendante du système**, c'est elle que l'on va utilisé. Comme cette horloge est indépendante du système, elle ne peut pas être sujette aux problème lié aux taux d'occupation du processeur, ou aux changemant de fréquence de celui ci dans le cas d'une utilisation sur un ordinateur portable.

L'implémentation et l'utilisation de ce compteur est le coeur de notre système de synchronisation, sans lui aucune synchronisation meme sur un système en local ne serait possible. L'horloge des cartes Firewire est codé sur 32 bits, et décomposé ainsi : 8 bits pour les secondes (0 à 128), 13 bits pour les cycles (0 à 8191) et 12 bits pour les offset (0 à 4095) (figure 4.3.3, page 37).

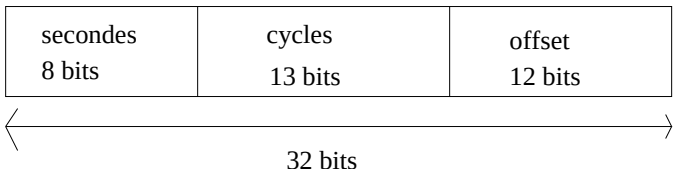

FIG. 4.3 – *Organisation du compteur Firewire.*

La définition du compteur a été trouvé dans dans la documentation technique du FireWire ([3] et [4]).

De plus comme on stocke les images dans l'ordre chronologique ou elles nous arrive, et que on arrête la recherche de l'image complémentaire dès qu'on en trouve une qui convient et que cette image est retiré de toute future recherche, on peut donc apparier plus de 128 seconde de vidéo.

Lorsqu'on sauvera une image quelque soit la méthode utilisé, on sauvegardera en plus du buffer image récupéré à la caméra, le numéro de la caméra et la valeur de l'horloge au moment de la capture.

Pour prévenir les images manquantes, quelqu'en soit la raison, on stocke les image apparier avec le même numéro, et en plus leur numéro d'image. Si il manque une images

ou qu'on ne peut pas les apparier, le numéro sera laisser vacant.

D : Synchronisation sur un seul PC

C'est le cas le plus simple, les deux caméras, dont on veut synchroniser les images sont connecter à la meme carte d'acquisition. Après la capture, on essaye de voir si chaque image prise sur la caméra 1 à son complémentaire sur la caméra 2, si on en trouve pas on passe à l'image suivante, comme on stocke les image apparier avec le meme numéro, si il manque une images ou qu'on ne peut pas les apparier, le numéro sera laisser vacant.

Ainsi au final, on aura sur le disque dur, que les images qui ont put être apparier.

E : Synchronisation sur un réseau de PC

Maintenant le problème devient plus complexe, puisqu'on doit synchroniser 4 caméra, branché 2 par 2, sur 2 PC différent. Heuresement, le fabricant des caméras nous fournissait un boitier de synchronisation, qui permet de synchroniser l'acquisition faite sur 2 PC différents. Le boitier de synchronisation permet de synchroniser la prise d'image. En faite il ne restait plus qu'a apparier les images 2 par deux sur chaque PC et de stocké que les deux paire complémentaire. Pour ce faire, nous avons dévelloper un protocole trés simple de communication TCP/IP pour pouvoir faire communiquer les 2 PC entre eux.

4.3.4 Évolution future

Nous donnerons ici ce qui n'a pas été réalisé et les apport qui pourrait être fait pour amélioré cette bibliothèque.

- Nous ne gérons pour l'instant qu'un seul mode graphique parmis les 7 que prévoit la norme Firewire, les autres mode introduissent des tailles d'images supérieur à celle que nous utilisons maintenant, en contrepartie la vitesse d'acquisition baisse en conséquence.
- Lors de la synchronisation réseau, la bibliothéque ne gère que 2 système relié entre eux. Physiquement on peut relier autant de système que l'on veut, tou en restant dans les normes du Firewire. Mais logiciellement, motre bibliothèque ne peut géré pour le moment que la communication entre 1 serveur et un client. On pourrait augmenter le nombre de client pouvant se connecté, ainsi on pourrait géré plus de caméras synchroniser en réseau.
- La calibration n'est pas très fonctionnelle, elle donne encore trop souvent des résultats trop aberrants.
- Une interface graphique, pour utiliser le système d'acquisition est encore en cours de réalisation.

Chapitre 5

Les résultats obtenus

Dans ce chapître, nous donnerons les résultats que nous avons obtenus jusque içi. Nous expliquerons le protocole expérimental qui nous a permis de les obtenir. Nous analyserons ces résultats pour montrer quels sont les points à améliorer, voir modifier.

5.1 Choix de programmation

Dans cette section, nous montrerons en quoi certains choix de programmation comme l'utilisation intensive du DMA, des threads , et du stockages sur la RAM nous ont permis d'augmenter de manière significative le nombre d'images que nous pouvons capturer.
Nous commencerons par donner la méthode utilisée pour mesurer le nombre d'images capturées, puis nous analyserons les résultats obtenus.

5.1.1 Méthode de mesure

Protocole expérimentale Nous devons mesurer combien d'images sont capturées en un temp fini qui est fixé à 4 secondes. Toutes les mesures ont été prises avec la même camera. Comme nous ne possédions pas 2 cameras couleur identiques, les différents tests ont été effectués avec des caméras noir et blanc. Les cameras utilisées ont fonctionné en envoyant des images en 640*480 et le niveau de gris codé sur 1 octet. Donc la taille d'une image était de 640*480 = 307200 octet. Les caméras étaient réglées pour capturer 3, 7, 15 ou 30 images par seconde, ce qui fait un débit de 9216000 octets par seconde pour une caméra dans le cas de la vitesse maximal soit 9.2 Mo par caméra pour 1 seconde.

Pour être sûr de la validité de nos résultats, nous avons fait plusieur fois les mêmes tests.

Nous avons effectué plusieurs mesures pour tester notre bibliothèque, des mesures avec une seule ou deux caméras, avec ou sans sauvegarde sur le disque dur, avec ou sans conversions, avec ou sans utilisation du mode DMA, et enfin avec ou sans utilisation des threads.

5.1.2 Les résultats et leurs analyses

Définissons tout d'abord quelques constantes :

- Pour une caméra, nous voulons capturer au mieux 120 images en 4 secondes (à 30 fps).
- Pour deux caméras, nous voulons capturer 240 images en 4 secondes (à 30fps).
- Chaque image stockée en brut sur le disque dur pèse 307200 octets soit 0.3 Mo.
- Chaque image stockée dans le format PPM sur le disque dur pèse 921615 octet soit 0.9 Mo (640*480*3=921600 + le header du fichier).

Nous avons plusieurs façons différentes de sauvegarder les images que nous avons acquise :

- 1) Sauvegarde séquentielle des images au format brut (RAW) ou au format PPM. Lorsque nous sauvegardons en PPM, nous devons passer obligatoirement par une étape de conversion, qui on le verra consomme beaucoup de CPU.
- 2) Sauvegarde dans un fichier différent pour chaque image (appelé méthode séquence) ou dans un seul fichier (on sauvegarde les différentes images les unes derrière les autres) (appelé méthode fichier unique).
- 3) Sauvegarde dans un buffer dans la RAM, les images étant transférées sur le disque dur, après un post-traitemant (appellé méthode mémoire).

Voici les différents résultats obtenus suivant le test réalisé, il y a deux grande séries de test suivant que nous avons utilisé la version de la bibliothèque avec ou sans threads.

Pour garder, une certaine cohérence entre les différents résultats des tests, nous donnerons un résultats moyen : le nombre d'images acquise par seconde sur les 4 secondes pour une caméra.

Dans chaque tableau, nous donnons le nombre d'images récupérées (le nombre d'image que la caméra nous envoit), et le nombre d'images sauvées (nombre d'images qui sont sauvées sur le disque dur).

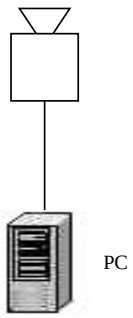

FIG. 5.1 – *Système de capture avec 1 caméra et 1 PC*

5.1.3 tests sans les threads

Pour cette méthode nous acquérons les images et nous les sauvons de facon séquentielle.

Nous donnons le nombre d'images pour une caméra, que nous pouvons acquérir en 4 seconde, puis nous donnons le nombre d'images que nous pouvons aquérir en 1 seconde (figure 5.1, page 40).

Le nombre d'images acquise est la moyenne pour 10 tests consécutifs.

non DMA, non thread, RAW, 1 caméra, 4 secondes

vitesse caméra	nombre d'images pour 4 sec	nombre d'images par seconde
3	13	3.25
7	29	7.25
15	32	8.00
30	34	8.50

non DMA, non thread, PPM, 1 caméra, 4 secondes

vitesse caméra	nombre d'images pour 4 sec	nombre d'images par seconde
3	14	3.50
7	28	7.00
15	32	8.00
30	34	8.50

DMA, non thread, RAW, 1 caméra, 4 secondes

vitesse caméra	nombre d'images pour 4 sec	nombre d'images par seconde
3	14	3.50
7	29	7.25
15	52	13
30	51	12.75

DMA, non thread, PPM, 1 caméra, 4 secondes

vitesse caméra	nombre d'images pour 4 sec	nombre d'images par seconde
3	14	3.50
7	30	7.50
15	42	10.50
30	44	11.00

Analyse des résultats En utilisant ce mode, nous ne pouvons guarantir que le nombre d'images aquises, sera égal à la vitesse de la caméra qu'a la plus faible vitesse (3 et 7 fps).

Ce test permet de voir assez facilement l'apport du mode DMA, dans la capture des images en provenance de la caméra. Le mode DMA permet de pouvoir sauver beaucoup plus d'images à la seconde, puisqu'on ne surchage pas le CPU inutilement.

Ce test montre aussi, le gain non négligeable que l'on obtient si on sauve les images en brute (pour les convertir dans un post-traitement), plutot que de les convertir à la volée.

En utilisant ce mode nous ne pouvons jamais sauver plus de 12 images par seconde et ce pour une unique caméra. C'est beaucoup trop faible, puisque d'après les spécifications du stage, nous devons sauver 30 images par seconde pour chacune des 2 caméras connectées. Pour pouvoir le faire nous avons utiliser une architecture logiciel utilisant massivement des threads pour pouvoir acquérir et sauver le maximum d'images pour les 2 caméras connectées.

5.1.4 tests avec les thread

Au début, nous avions commencé à tester l'utilité des threads avec une caméra, mais nous nous sommes aperçu que dès que nous utilisions le mode DMA, nous capturions autant d'images que la caméra pouvait nous en envoyer.

vitesse caméra	**nombre d'images récupérées**	**nombre d'images sauvées**	**nombre d'images par sec**
3	17	14	3.5
7	33	30	7.5
15	65	60	15
30	123	120	30

Comme, notre système devait gérer au final 2 caméras, nous détaillerons içi les résultats expérimentaux obtenus avec 2 caméras (figure 5.2, page 43). Les résultats obtenus avec 2 caméras sont plus significatifs, que ceux obtenus avec une seul caméra.

Les tableaux donneront le nombre d'images récupérées pour chaque caméras et le nombre d'images réellement sauvées.

Certaines cases du tableau, contiennent deux valeurs, il s'agit du nombre d'images pour chaque caméra.

La dernière collone donne la vitesse d'acquisition effective (mesurée expérimentalement), si le chiffre est en vert, c'est qu'il correspond à la vitesse théorique (première collone du tableau), si il est en rouge il ne correspond pas à la vitese théorique.

A : Test de la sauvegarde mémoire

Cette méthode de sauvegarde, sauve les images dans un buffer situé dans la RAM. Cette série de test permet de valider si l'acquisition se fait bien à la vitesse requise,

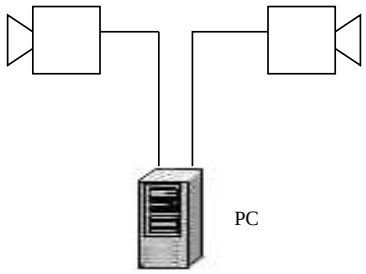

FIG. 5.2 – *Système de capture avec 2 caméras et 1 PC*

puisque le chargement d'une image en RAM est presque instantané.

non DMA, thread, 4 secondes, 2 camera, méthode mémoire

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	9 + 8	7 + 7	17	14	1.75
7	16 + 17	15 + 16	33	31	3.875
15	37 + 33	35 + 31	70	66	8.25
30	87 + 76	72 + 68	163	140	17.5

DMA, thread, 4 secondes, 2 camera, méthode mémoire

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	15 + 15	14 + 14	30	28	3.5
7	32 + 31	31 + 31	63	62	7.75
15	62 + 64	60 + 60	126	120	15
30	122 + 126	120 + 121	248	241	30.125

Analyse des résultats Une première analyse montre, que meme en utilisant des threads, l'utilisation du mode DMA, apporte un gain non négligeable. On peut voir que même en sauvant les images dans la RAM on ne peut atteindre la vitese théorique recherché que en utilisant le mode DMA. Donc on peut conclure que l'utilisation du mode DMA est obligatoire si on veut acquérir des images en temps réel.

B : Test de la sauvegarde fichier unique

Dans ce mode sauvegarde, on sauve toute les images dans un seul fichier temporaire situé sur le disque dur quelque soit le nombre de caméra. Ce mode a été créé pour perdre le moins de temps en accès disque puisque nous n'ouvrons qu'un seul fichier.

non DMA, thread, 4 secondes, 2 camera, méthode fichier unique

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	9 + 5	7 + 8	18	15	1.875
7	19 + 17	15 + 15	36	30	3.75
15	43 + 37	27 + 25	80	52	6.5
30	91 + 86	80 + 82	178	162	20.25

DMA, thread, 4 secondes, 2 camera, méthode fichier unique

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	15 + 19	14 + 15	34	29	3.625
7	31 + 34	30 + 30	65	60	7.5
15	61 + 65	60 + 61	126	121	15.125
30	122 + 125	122 + 123	247	245	30.625

Analyse des résultats Comme dans le mode de sauvegarde précédent, nous voyons que seul l'utilisation du mode DMA permet d'avoir une sauvegarde en temps réel. Pour gagner du temps toutes les imgages ont été stocké en brute, pour éviter de perdre du temps lors de leurs conversions (elles sont convertient lors d'un post-traitement).

C : Test de la sauvegarde fichier caméra

Ce mode de sauvegarde, sauve les images dans plusieurs fichiers temporaire, un fichier temporaire par caméras utilisées. Ce mode a été créé pour pouvoir limité le problème de la taille maximale que peut avoir un fichier. De plus se mode sépare physiquement les images issus de plusieurs caméras différentes, ce qui est pratique, si seuls les images d'une caméra nous intéresse, ou si on veut les apparier.

non DMA, thread, 4 secondes, 2 camera, méthode fichier caméra

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	9 + 5	7 + 8	18	15	1.875
7	20 + 16	15 + 15	36	30	3.75
15	42 + 38	28 + 25	80	53	6.625
30	96 + 86	83 + 83	182	166	20.75

DMA, thread, 4 secondes, 2 camera, méthode fichier caméra

vitesse caméras	nombre d'images récupérés	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	15 + 19	14 + 15	34	29	3.625
7	31 + 34	30 + 30	65	60	7.5
15	61 + 65	59 + 61	126	120	15
30	122 + 125	121 + 122	247	243	30.375

Analyse des résultats Ce mode de sauvegarde donne des résultats trés similaire au mode 1 fichier, et il confirme que l'utilisation du mode DMA est obligatoire si on veut sauver les images en temps réel.

D : Test de la sauvegarde séquentiel

non DMA, thread, 4 secondes, 2 camera, méthode séquentiel

Ce mode de sauvegarde sauve chaque image dans un fichier différent, chaque image sera nommé avec son numéro d'acquisition et son numéro de caméra. C'est le mode qui se rapproche le plus du mode de sauvegarde sans threads. Comme il fait des sauvegarde du meme type (une image pour un fichier) on peut facilement les comparer.

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	11 + 12	8 + 9	23	17	2.125
7	27 + 18	20 + 16	45	36	4.5
15	62 + 60	45 + 39	120	84	10.5
30	92 + 91	79 + 85	183	164	20.5

DMA, thread, 4 secondes, 2 camera, méthode séquentiel

vitesse caméras	nombre d'images récupérées	nombre d'images sauvées	total images récupérées	total images sauvées	nombre d'images par sec
3	15 + 19	14 + 15	34	29	3.625
7	31 + 34	30 + 30	65	60	7.5
15	61 + 65	60 + 61	126	121	15.125
30	122 + 125	120 + 120	247	240	30

Analyse des résultats Cette série de test confirme que l'utilisation du mode DMA est obligatoire si on veut avoir une acquisition sauvegarde en temps réel. Dans ce mode de capture, comme dans tous les autres on a réussi a obtenir les vitesses théorique.

Dans un système de capture, nous sommes limité par la vitesse de sauvegarde. Un disque dur, même récent est lent par rapport à de la RAM. C'est pour cette raison, qu'il y a plusieurs mode de sauvegarde différents. Avec une sauvegarde en RAM, nous sommes sûr que toutes les images seront stockées, tandis qu'avec un disque dur, il se peut que certaines images manque. De plus la création et la fermeture d'un fichier sont des opérations lentes, c'est pour cela que nous essayons des mode de sauvegarde avec un nombre de fichier temporaire différent. La version utilisant un seul fichier temporaire est celle qui consomme le moins de ressources système.

E : Comparaisson entre le mode utilisant des threads et le mode ne l'utilisant pas

Dans le mode n'utilisant pas les threads, on a réussi au maximum à sauver 13 images par seconde pour 1 seul caméra, mais dans le mode utilisant des threads nous avons toujours réussi (en utilisant le mode DMA) a sauver autant d'image que nous en acquerions (meme à la vitesse de 15 ou 30 fps) et ceux pour 2 caméras en parallèle.

Donc si on veut sauvé des images en temps réel, il faut utiliser **le mode threads et le mode DMA en parallèle**.

5.2 Synchronisation local

Après avoir prouvé que nous pouvions sauver toutes les images des 2 caméras que nous acquérions, quelque soit la vitesse d'acquisitions des caméras, nous montrerons comment nous avons prouvé expérimentalement la synchronisation entre les différentes caméras.

5.2.1 Méthode de mesure

Protocole expérimentale Nous avons filmé un ordinateur portable sur lequel défilait un compteur. Nous avons choisit d'utilisé un ordinateur portable pour que l'affichage du compteur soit bien séparé de la partie capture. Ainsi en utilisant un autre ordinateur, nous empéchons que l'affichage graphique est la moindre influence sur la capture, ce sont 2 entités bien séparées.

L'ordinateur portable affichait un compteur et une grille (figure 5.3, page 48) , c'est nous qui définissions la fréquence de rafraichissement du compteur et de la grille (15 ou 20 ms lors de nos tests). L'élément limitant de cette installation est la fréquence de rafraichissement de l'écran qui doit être régler au maximum pour pouvoir garantir les résultats (réglé à 100 Hz).

Détaillons le mode de fonctionnement du compteur et de la grille. La grille est composé d'un nombre fixe d'intersections, qui sont parcourues par un point. A chaque rafraichissement, le point est affiché à l'intersection suivante (en s'effaçant de l'intersection précédente). Le compteur permet de vérifier la position du point sur la grille, le premier chiffre correspond au nombre de grille dont le point est passé sur toutes les intersections. Le second chiffre donne le numéro de l'intersection où se trouve le point. La lecture du compteur doit permettre de définir la position du point sur la grille et inversement. Pour chaque image, nous avons vérifié si le point se trouvait bien là ou le deuxième chiffre du compteur le situait. Pour voir si nous avions bien des images synchronisé, nous avons observé chaque paire d'images pour vérifier si le compteur donnait bien le même chiffre et si le point se situait bien à la même intersection sur les deux images (figure 5.4, page 49). Au taux de rafraichissement les plus élevés (15 ms), l'ordinateur portable n'était plus assez rapide pour afficher un points à chaques intersections, de temps en temps il en sautait une (figure 5.5, page 50).

Donc un point restera affiché à l'écran, durant le temps d'un rafraichissement. Mais la caméra ne prend pas une image de façon instantanée, il lui faut un temps de pose de quelques millisecondes, ce paramètre de la caméra peut être réglé par l'option shutter du fichier de configuration. Comme le temps de pose d'une prise de vue n'est pas instantané, il est possible d'avoir 2 point à l'écran situé à 2 intersections différentes, de même il est courant de voir plusieurs chiffres dans plusieurs teintes de gris supperposées pour le compteur (figure 5.5, page 50).

Nous avons testé la synchronisation avec tous les modes de sauvegarde implémentés, mais nous ne détaillerons les résultats que pour un seul mode en particulier (le mode mémoire).

5.2.2 Les résultats et leurs analyses

Dans tous les modes testés, nous avons obtenu plus ou moins le même nombre de paires d'images synchronisées selon l'occupation du système.

FIG. 5.3 – *Installation de test de la synchronisation*

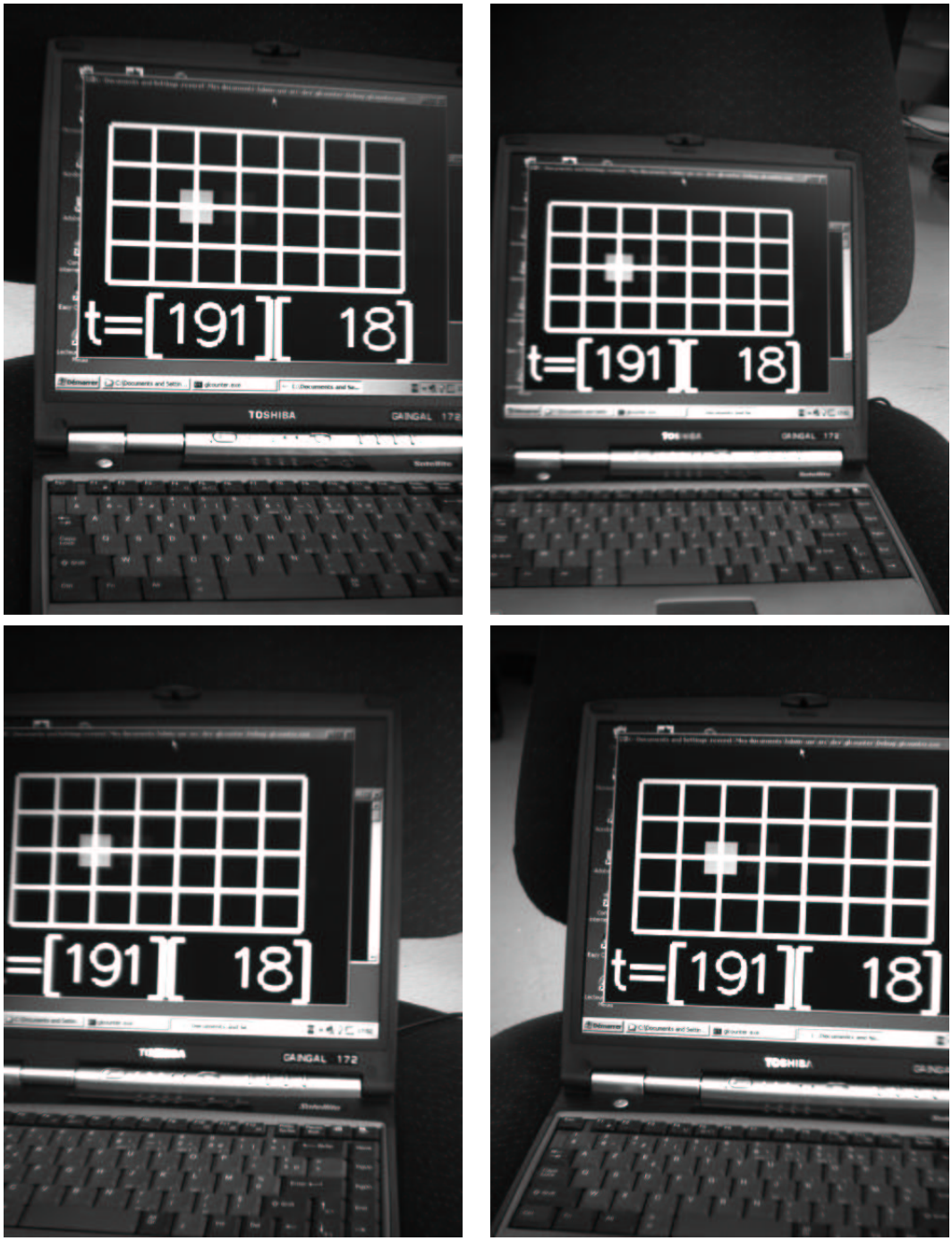

FIG. 5.4 – *Résultat de la synchronisation (1 point sur la grille)*

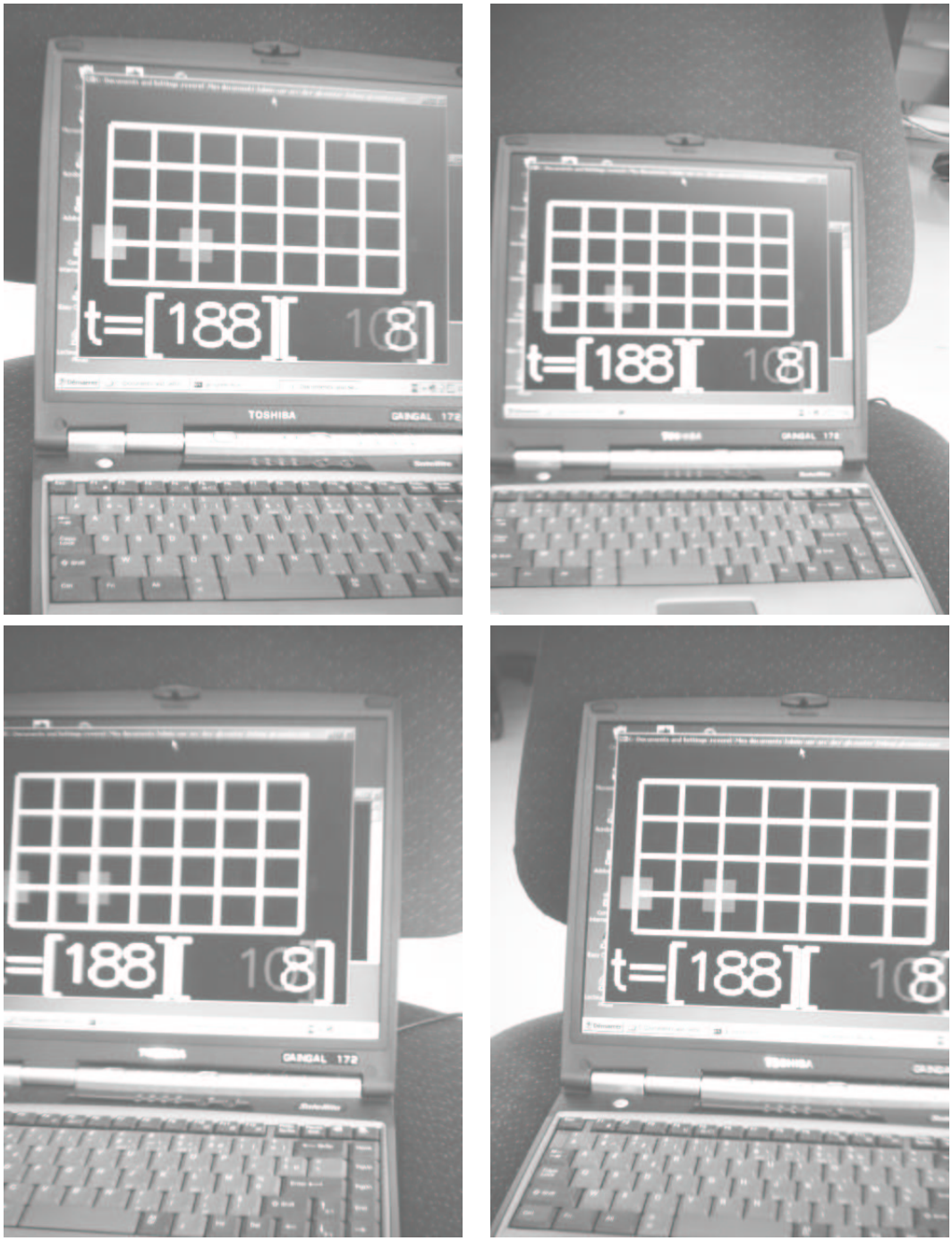

FIG. 5.5 – *Résultat de la synchronisation (plusieurs points sur la grille)*

La dernière colonne donne le nombre d'images appariées pour chaque caméra. Si on veut que notre système soit performant il faut que ce chiffre soit le plus proche possible du nombre d'images sauvées (nombre théorique d'images appariées = vitesse caméra * nb de secondes).

vitesse caméras	**nombre d'images récupérées**	**nombre d'images sauvées**	**total images récupérées**	**total images sauvées**	**paires d'images appariées**	**nombre d'images par sec**
3	15 + 19	14 + 15	34	29	14	3.5
7	31 + 34	30 + 30	65	60	30	7.5
15	61 + 65	60 + 61	126	121	59	14.75
30	122 + 125	120 + 120	247	240	120	30

D'après ces résultats nous observons que nous avons près de 95% des images qui peuvent être apparier. On peut expliquer le plus grand nombre d'images acquises, sur celles sauvées, par le faite qu'on lance d'abord l'acquisition puis ensuite la capture, durant l'intervalle de temps entre leur deux mises en route le système à le temps d'acquérir un certain nombre d'image.

Pour prouver un peu plus l'efficacité de notre système, nous l'avons testé sur une prise de vue d'une minute. Nous avons obtenu les résultats suivant : 1807 images sauvées sur la première caméra, 1804 sur la seconde et au final nous obtenions 1800 images appariées (en 1 minute nous devions sauver 30*60 = 1800 image). ce test nous a permis de voir que nous pouvions sauvegarder 1 minute de flux vidéo récupéré sur 2 caméras distinctes et parfaitement synchronisées entre elles.

5.3 Synchronisation réseau

Il s'agit de faire les même test que pour la synchronisation local, mais en connectant 2 système d'acquisition entre eux par le boitier de synchronisation fournit par PointGrey.

Il s'agissait de vérifier si nous obtenions les mêmes résultats sur 2 PC en réseau que sur 1 PC en local

vitesse caméras	**total images récupérées PC 1**	**total images sauvées PC 1**	**total images récupérées PC 2**	**total images sauvées PC 2**	**quatuors d'images appariées**	**nombre d'images par sec**
3	34	30	35	30	30	3.75
7	65	60	65	60	59	7.375
15	129	123	126	121	121	15.125
30	243	241	247	240	239	29.875

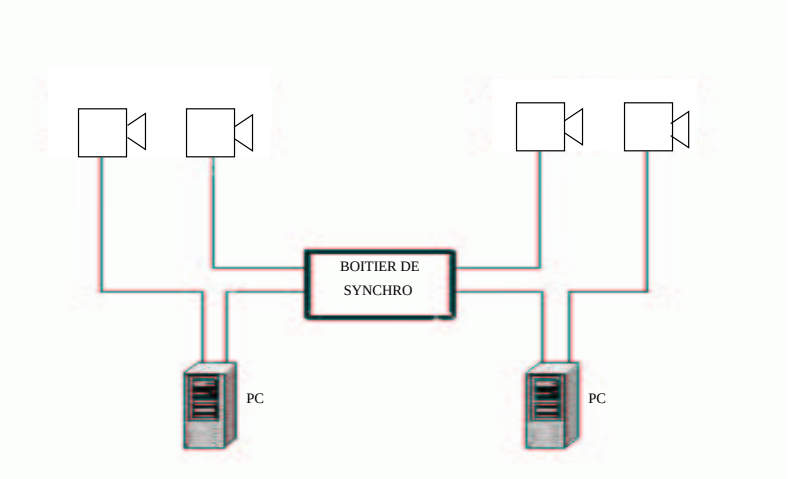

FIG. 5.6 – *Système de capture avec 4 caméras et 2 PC*

Analyse des résultats Comme dans le cas de la synchronisation en local, nous parvenons parfaitement à synchroniser deux système d'acquisition en réseau, nous obtenons un taux d'appariment presque aussi bon 94% chiffre un peu moins bon à cause de l'impréçission de la synchronisation des deux compteurs entre eux.

5.4 synchronisation avancé

5.4.1 synchronisation de 6 caméras

Dans cette section nous nous interreserons brièvement à la synchronisation de plus de 4 caméras. Durant ce stage, nous avons put testé la synchronisation de 6 caméras, au maximal, relié 2 par 2 à 3 PC (figure 5.7, page 53).

Dans cette configuration, nous n'avons pas obtenus de résultat très significatif, à cause de l'instabilité de cette configuration. En effet, un réseau entre les boitiers de synchronisation, entraine une instabilité des 3 systèmes de capture, il était très fréquent, que les boitiers n'arrivent pas à ce synchroniser entre eux, ce qui entrainait que sur le système 2 chacune des 2 caméras était synchronisée à un boitier différents. Dans ce cas de figure, nous n'obtenions que très peu d'image aparier. La caméra qui nous possaient le plus de prolème est celle qui est relié à un PC par un réseau Firewire passant par 2 boitier de synchronisation. Cette erreur est apparut dans environs 40% des tests que nous avons effectués dans cette configuration. Lorsque cette erreur était détecté, nous obtenions un taux de 12% de synchronisation (12 % de la totalité des images récupérées sur un système pouvait être synchroniser avec les deux autres système). Une autre erreur fréquente, était que sur le système 2 (toujours lui), une des 2 caméras ne capture qu'une fraction des images envoyées, et quelque soit la puissance du PC employé.

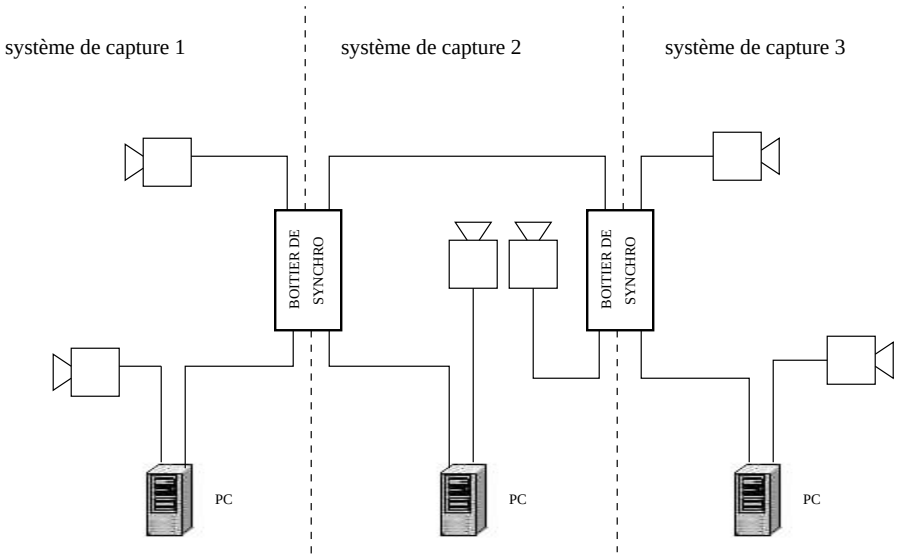

FIG. 5.7 – *Système de capture avec 6 caméras et 3 PC*

Enfin dans cette configuration, une erreur est apparut lors de la fermeture de la connexion aux caméras, ce qui entrainait le blocage du programme de capture. Nous pensons que les bibliothèques *libdc* et *libraw* ne savent pas correctement gérer les réseaux entre boitier de synchronisation.

Dans cette configuration, nous n'avons obtenus des résultats convenables (c'est à dire plus de 95% d'images appariées) que dans 3 test sur 10.

Dans cette configuration, nous avons obtenus les résultats suivant :

vitesse caméras	**total images sauvées PC 1**	**total images sauvées PC 2**	**total images sauvées PC 3**	**sixtètes d'images appariées**	**nombre d'images par sec**
3	30	30	30	15	3.75
7	61	60	59	29	7.25
15	123	121	121	61	15.25
30	241	240	242	120	30

5.4.2 synchronisation de 5 caméras

Comme lors des tests de synchronisation de 6 caméras, nous nous sommes aperçut que nous avions des problèmes avec la deuxième caméras du système 2. Nous avons décidé de l'enlever (figure 5.8, page 54).

Dans cette nouvelle configuration, nous avons observé une bien plus grande stabilité du système. Ainsi, nous n'avons constaté que très épisodiquement (2 test sur 14) un problème

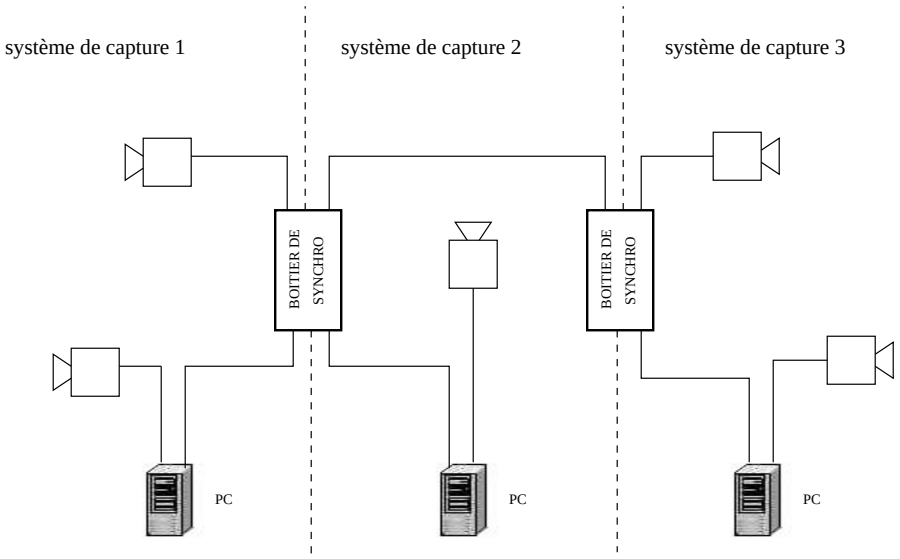

FIG. 5.8 – *Système de capture avec 5 caméras et 3 PC*

(mauvaise acquisition, caméra inacessible, ...) du programme de capture sur l'un des trois système de capture.

Dans cette configuration, nous avons obtenus les résultats suivant :

vitesse caméras	**total images sauvées PC 1**	**total images sauvées PC 2**	**total images sauvées PC 3**	**quintètes d'images appariées**	**nombre d'images par sec**
3	30	15	31	15	3.75
7	61	32	60	29	7.25
15	123	59	120	60	15
30	241	119	240	120	30

Nous observons que nous restons bien dans les spécifications que nous nous sommes fixer. C'est cette configuration, qui nous a permit de synchroniser le plus de caméras, tout en gardant une certaine stabilité du système.

5.4.3 Synchronisation de 6 caméras dans une nouvelle configuartion

On a observé que notre système de capture était très stable, dans la configuration avec 5 caméras reliées à 3 PC par le biais des boitiers de synchronisation. Comme on a put observé que c'était la caméra qui était relié à un PC par un réseau passant par 2 boitiers de synchronisation qui nous possait des problèmes, on a imaginé cette configuration pour synchroniser 6 caméras en utilisant 4 PC et 3 boitiers de synchronisation différents (figure

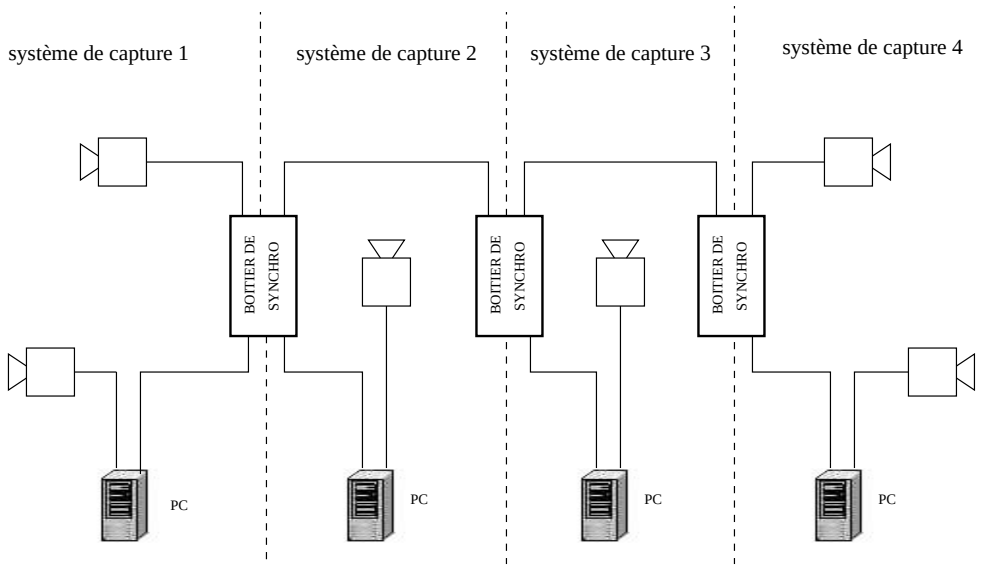

FIG. 5.9 – *Système de capture avec 6 caméras et 4 PC*

5.9, page 55).
Cette configuration est purement théorique, elle n'a jamais put être testé expérimentalement, car nous ne possédions que 2 boitier de synchronisation.

5.4.4 Synchronisation avec plus de 6 caméras

Comme l'on montré les tests précédents, nous pouvons monter un réseau de système de capture, relié entre eux par un réseau ethernet et par les boitiers de synchronisation pour le réseau firewire. Mais au vus des problèmes technique qui sont apparus lors des tests de synchronisation de 6 caméras, on devrait retrouver les mêmes problèmes.

Pour synchroniser plus de 6 caméras, il faut utilisé une configuration dans laquelle chaque PC est relié à une caméra sauf Pour les 2 PC des extrémitées du réseau, qui eux sont connectés à 2 caméras.

Chapitre 6

Modèles de couleurs et calibration de caméra

6.1 Les modèles de couleurs

Cette section décrit quelques unes des représentations numériques de la couleur.

6.1.1 L'espace RGB

Ce modèle code les trois couleurs avec trois composantes : le Rouge, le Vert et le Bleu, chacune étant comprise entre 0 et 1 et ce avec synthèse additive, c'est à dire que, tout comme la lumière du jour, le blanc est composé de toutes les composantes au maximum tandis que le noir d'aucune. L'ensemble de ces couleurs (figure 6.1.1, page 57) décrit un espace orthonormé à trois dimensions : chaque couleur peut être représentée comme un point de cet espace. A noter que sur une même droite linéaire, les couleurs sont invariantes et que ce n'est pas la luminosité de la couleur en question qui varie. . Le codage RVB des couleurs est aussi le codage utilisé par les écrans d'ordinateur pour reproduire les couleurs.

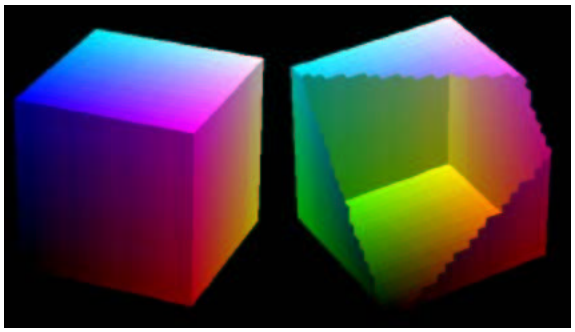

FIG. 6.1 – *Le cube RGB*

6.1.2 L'espace YUV

L'espace YUV est courament utilisé dans de nombreuses application vidéo, telles que la télévision en Europe, ou bien la compression MPEG. Il en existe plusieurs variantes (YIQ,YCrYCb,..) mais elles ont toutes en commun la séparation de la luminance (Y) et de la chrominance (U et V) de l'objet. La *chrominance* est la couleur du pixel, c'est à dire le rapport entre les couleurs primaires qui la composent. La *luminance* est, quant à elle, l'intensité lumineuse du pixel. On passe de RGB à YUV grâce à la matrice suivante :

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 1.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} . \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

et inversement :

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.140 \\ 1 & -0.394 & -0.581 \\ 1 & 2.028 & 0 \end{pmatrix} . \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

6.1.3 Utilisation des différents modèles de couleurs

Les caméras Firewire nous renvoyait des images qui étaient dans plusieurs formats de couleur différents

- En niveau de gris (chaque pixel codé sur 8 ou 16 bits).
- Au format RGB (chaques couleurs codé sur 8 bits).
- Au format YUV444, YUV422 et YUV411.

Il a donc fallut géré la conversion d'un format à un autre. Dans la bibliothèque, nous n'utilisons pour les sauvegardes et l'affichage que des images au format RGB.

Or le temps perdu lors de la convertion d'un format à un autre est assez conséquent.

Pour l'affichage, nous ne pouvions afficher que des images RGB, il fallait donc convertir en RGB les images reçut dans un autre format. Or la conversion d'un format à un autre prend du temps, il faut convertire tous les pixels les uns après les autres. Donc nous avons créé un module d'affichage en temps réel d'image YUV (dans le mode d'affichage XV, c'est la carte graphique qui fait la conversion et non plus le processeur central)

Pour éviter de perdre trop de temp, en conversion, lors de la sauvegarde en temp réel, sur plusieurs caméras, nous sauvions les images en brute et nous les convertisions lors d'un post-traitement où le temp réel n'était plus une condition.

6.2 calibration de caméra

L'étape de calibration a pour but d'obtenir un modèle numérique d'une caméra, correspondant à ses caractéristiques géométriques réelles. Un tel modèle est décrit par deux ensembles de paramètres, qui sont les paramètres intrinsèques (par exemple la distance focale) et les paramètres extrinsèques (par exemple la position de la caméra dans l'espace). Cette étape terminée, il est alors possible de savoir comment la caméra projete un point de l'espace sur l'écran.

6.2.1 Modélisation d'une caméra

Le plus généralement, une caméra est modélisée sous la forme suivante :

$$\begin{pmatrix} \gamma.u \\ \gamma.v \\ \gamma \end{pmatrix} = \mathcal{K} \times \mathcal{M} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

où :

- (u, v) les coordonées en pixel du point dans l'image.
- γ un facteur pour les coordonnées homogènes.
- $\mathcal{K}$ la matrice intrinsèque de la caméra.
- $\mathcal{M}$ la matrice de transformation géométrique.
- $(x, y, z, 1)$ les coordonnées du point dans l'espace.

A : Les paramètres intrinsèques

Ils sont composés des coefficients de la matrice $\mathcal{K}$, dite matrice intrinsèque de la caméra.

$$\mathcal{K} = \begin{pmatrix} k_u.f & c & u_0 \\ 0 & k_v.f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

où :

- k_u et k_v sont les facteurs d'échelles fonction de la taille de l'image en pixels/mm.
- f est la distance focale. On note en général $f.k_u \equiv \alpha_u$.

- (u_0, v_0) sont les coordonnées en pixels de l'intersection de l'axe optique avec le plan rétinien. Ce point n'est généralemant pas situé exactement au centre de l'image (figure 6.2, page 60).
- c est un coefficient qui est null lorsque les axes de l'image sont orthogonaux.
- $k_u.f$ et $k_v.f$ sont souvent noté α_u et α_v.

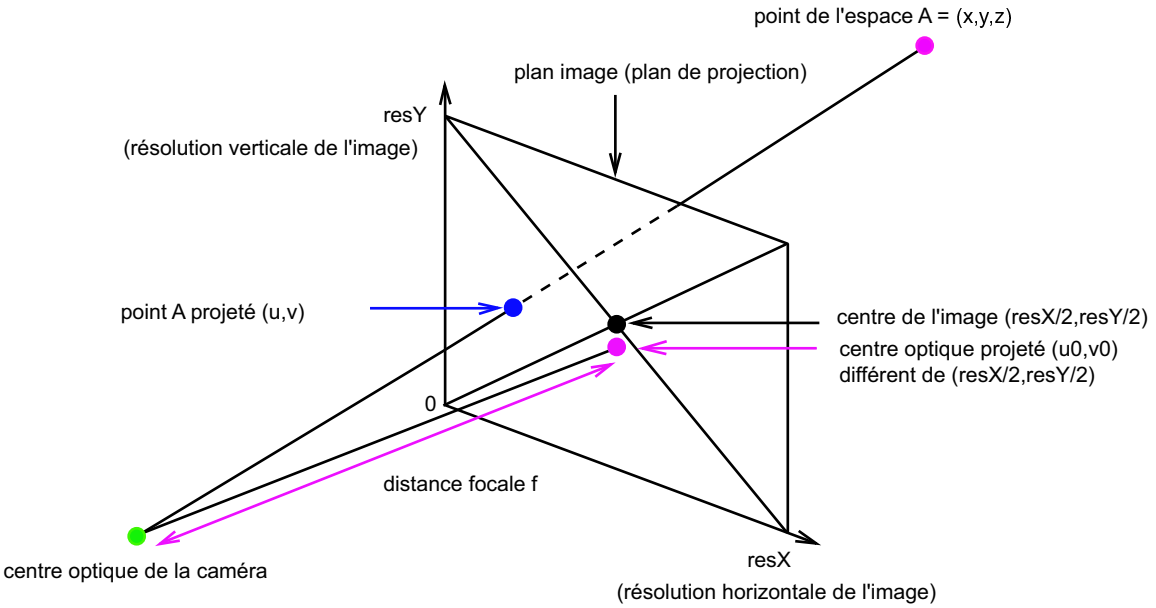

FIG. 6.2 – *Schéma d'une caméra avec la distance focale et la projection de l'axe optique*

B : Les paramètres extrinséques

Ils sont composés par les coefficients de la matrice $\mathcal{M}$, dite matrice de transformation de la caméra.

$$\mathcal{M} = \begin{pmatrix} \mathcal{R} & \mathcal{T} \end{pmatrix}$$

Ils correspondent à la transformation qui passe d'un repère de référence de l'espace, au repère caméra : il s'agit donc d'une translation générée par $\mathcal{T}$ (vecteur à 3 composantes) et d'une rotation générée par $\mathcal{R}$ (matrice 3x3) dans l'espace (figure 6.3, page 61).

6.2.2 Méthode de calibration

La phase de calibration permet d'obtenir de façon automatique l'ensemble de ces paramètres. La calibration est aujourd'hui bien maitriser en vision par ordinateur. Par

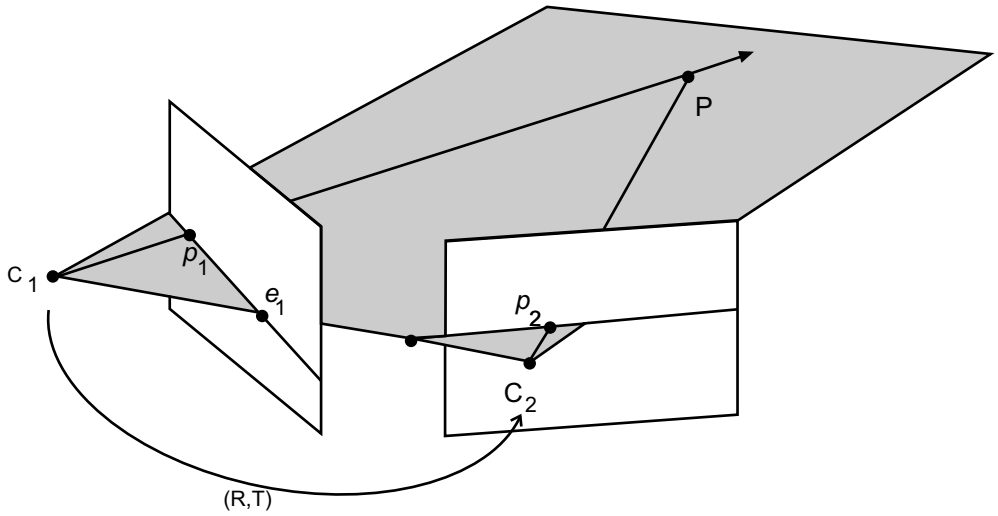

FIG. 6.3 – *Exemple de passage du repère 1 au repère 2*

exemple OpenCV (pour **Open** Source **C**omputer **V**ision Library développé par Intel) propose de déterminer les paramètres intrinsèque et extrinsèque d'une caméra à partie d'un certain nombre de prise de vue différentes d'un damier fait à partir de la caméra que l'on veut calibrer.

En effet, le seul matériel à avoir à disposition, est un simple damier noir et blanc (figure 6.4, page 61).

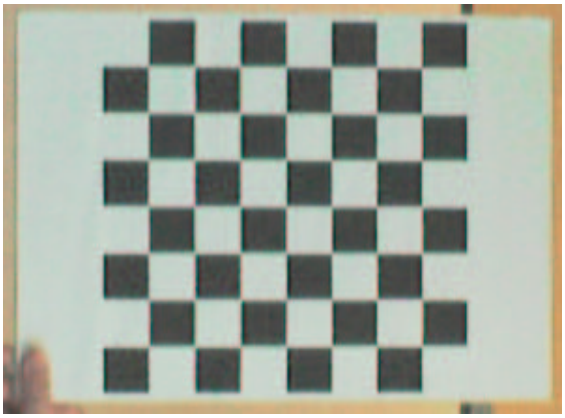

FIG. 6.4 – *La mire de calibration*

6.2.3 Protocole de calibration

La calibration se fait en quatre étapes grâce aux fonctions fournis par la bibliothèque OpenCV :

- Tout d'abord il faut extraire et suivre les points situés aux intersections des cases blanches et noires du damier (figure 6.5, page 62).

- Ensuite il faut réordonner les points trouver, pour qu'ils soient dans un certain ordre.
- Puis connaissant les coordonnées de ces points sur l'image, il est possible de déterminer les paramètres internes de la caméra.

- Enfin il est possible de déterminer les paramètres externes de la caméra

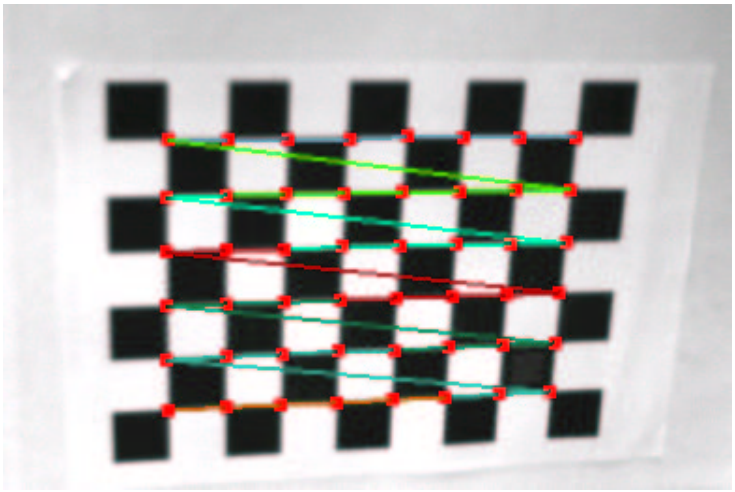

FIG. 6.5 – *Détection des coins du damiers sur la mire de calibration*

Afin de déterminer les coefficients de la matrice intrinsèque, il suffit de capturer au minimum 3 images du damier sous différents angles de vues et de stocker les intersections trouvées afin de pouvoir déterminer l'homographie dans le plan projectif de l'image. Ce sont les coefficients de la matrice d'homographie qui sont sont les coefficients de la matrice intrinsèque. Cette étape est réalisé de façon indépendante pour chaque caméra.

Les coefficients de distortions sont déterminés a postériori en corrigeant par itération successives les paramètres précédents.

En ce qui concerne la matrice de transformation, toutes les caméras doivent capturer la même image du damier afin qu'il soit possible de retrouver leur position relative. En effet, une fois les paramètres internes établis, il est possible de trouver la matrice de transformation qui fait correspondre aux points filmés, des points de l'espace. Bien sûr pour cela, il faut préciser le repère (arbitraire) qui est attribué au damier (les coordonnées dans l'espace virtuel que l'on désire attribuer aux intersections, ainsi, on a un repère de référence sur lequel se baser pour définir les transformations.

Le problème ici est qu'il est impossible de savoir de quel coté du damier la caméra se trouve. En effet, le doute vient du fait que si deux caméras sont placées de part et d'autre du damier, comment déterminer laquelle se trouve à droite et laquelle se trouve à gauche. Par ailleurs quel est le côté droit et le côté gauche du damier ? Une astuce trés simple permet de lever l'ambiguité.

6.2.4 Amélioration apporté au protocole de calibration

Il suffit, en fait, de disposer une pastille de couleur sur un angle du damier, La case marquée sera considérée comme origine du damier. Parmis les intersections trouvées, il ne reste donc plus qu'a déterminer les 16 intersctions qui 4 par 4 encadrent les 4 cases situées aux angles du damier, puis de trouver quel angle contient la pastille de couleur (figure 6.6, page 63).

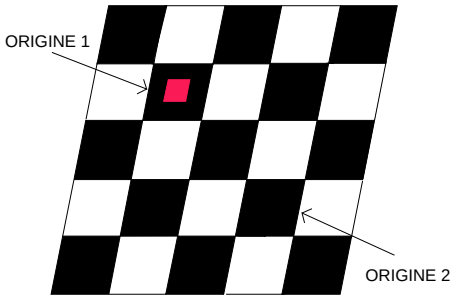

FIG. 6.6 – *Schéma de la mire avec l'origine*

Expérimentalement, nous avons observé que la méthode de la librairie OpenCV poir trouver les intersection du damier dans une image, était très sensible à la lumière. Il était très fréquent que cette fonction ne trouve qu'une partie des intersections. Nous avons découvert de façon expérimentale comment comment augmenté tout en gardant la même image, le nombre de point que cette fonction va trouver, il suffit d'augmenté le gamma de l'image, plus le gamma est haut, plus les chances de trouver toutes les intersections est grande.

Une autre amélioration a été mise au point : la fonction d'OpenCV nous rend les coordonnées des intersection dans un certains ordre qui peut être incompatible avec l'ordre requis pour trouvé les paramètres internes. Nous avons donc écrit un algorithme de réordonnement des points trouvés. Pour le moment cette algorithme, n'a pas été complètement implémenté.

6.2.5 Limites de cette méthode

La limite de cette méthode est que la caméra doit voir et détecter toutes les intersections pour pouvoir déterminer sa position par rapport au repère arbitraire du damier. Or

dans une position rasante par rapport au plan de ce damier, il devient impossible de toutes les extraire. Toutes les caméras doivent donc avoir une vue relativement plongeante sur le damier ce qui restreint la dispertion des caméras autour de la scène et diminue d'autant le nombre de partie observable de l'objet ciblé.

6.2.6 Procédure de calibration

Nous détaillerons içi la procédure retenu pour calibrer les caméras :

1) Applications d'un gamma correcteur, pour une meilleur recherche des points situé, aux intersections des cases blanches et noires du damier.

2) Dans l'image courante, nous cherchons les points situés aux intersections des cases blanches et noires du damier.

3) si nous en trouvons assez (nb de points = (largeur damier - 1) $\times$ (hauteur damier -1)) nous poursuivons le traitement.

4) Nous cherchons les quatres point formant les coins du damier (figure 6.6, page 63).

5) A partir des coins, nous trouvons la pastille de couleur.

6) Nous réorganisons les points pour qu'ils soient dans un ordre bien précis (figure 6.5, page 62)

7) Nous sauvons les points pour un traitement ultérieur.

8) Quant nous avons avons assez d'image nous utilisons une fonction de la librairie OpenCV, pour déduire les paramètres intrinséques et extrinséques de la caméra considérer.

Chapitre 7

Évolutions future et bugs connus

Dans un premier temps, nous détaillerons les futures évolutions qui peuvent être apportées à la librairie *util_1394*. Puis dans un deuxième temps nous donnerons une liste non exhaustive des différents bugs connus et répertoriés à ce jour.

7.1 Évolutions future

7.1.1 Initialisation du système de capture

- 1) Nous ne sommes capables de lancer l'acquisition qu'avec un seul format de données (VGA NONCOMPRESSED). La norme Firewire prévoit plusieurs autres formats de données, qui n'ont pas été implémentés car aucune des caméras de test, ne les supportaient (SVGA NONCOMPRESSED 1, SVGA NONCOMPRESSED 2, STILL IMAGE, SCALABLE IMAGE SIZE).

- 2) Le système de capture ne peut tenir compte que des caméras connectées au moment du lancement du programme. On pourrait modifier la bibliothèque pour tenir compte des fonctionnalitées plug'n play du Firewire.

- 3) Une fois qu'une caméra est initialisé, on ne peut pas changer certain de ses paramètres : format de données (VGA NONCOMPRESSED, SVGA, ...), mode, framerate, vitesse de transmission). Ce serait une fonctionalitée interressante, si on pouvait réinitialisé à la volée une caméra plutot que de redémarer le système de capture.

- 4) Faire un fichier de configuration pour le système de capture, ce fichier devrait contenir tous les champs de la structure preference_camera, plus les autres information que le système doit demandées à l'utilisateur.

7.1.2 Affichage des caméras

Pour géré l'affichage d'une ou de 2 caméras sur l'écran d'un PC en temps réel, plusieurs solutions ont été envisagées, chacune avec ses défauts et ses qualitées. Pour une utilisation optimal, de l'affichage il faut disposer d'un PC le plus puissant possible (le processeur le plus rapide et une grande quantité de RAM), et que l'accélération matériel de la carte graphique soit activé.

Dans cette partie, nous détaillerons les 4 modes créés pour afficher les images en provenance des différentes caméras.

- SDL : Dans ce mode, nous pouvons afficher, deux caméras en même temps (dans la même fenêtre), la fluidité de l'affichage est très dépendant de la puissance du PC. Je ne suis pas totalement sur que l'accélération matériel de la carte graphique soit prise en compte.

- X11 : Dans ce mode, nous pouvons afficher, deux caméras en même temps (dans deux fenêtre séparées). Dans ce mode la fluidité de l'affichage ne dépent que de la puissance du PC. Je pense que dans ce mode il y a moyen d'accéléré la convertion, mon algorithme de conversion se fait en deux passe pour certaine conversion, on pourrait exécuter la conversion en une seul passe.

- XV : Dans ce mode, nous ne pouvons actuellement afficher qu'une caméra (si on en affiche deux, on voit apparaitre un fond bleu pour l'image 1 quant on réaffiche l'image 2, et vice versa). Dans ce mode, nous nous servons des capacitées de conversion hardware présente sur la carte graphique. Ainsi la carte fait la convertion en hardware du YUV en RGB, et affiche le résultat. On se sert d'un nouveau module du serveur graphique XFREE 4.*, le module XVIDEO. C'est se même module qui assure la décompression et l'affichage des DVD sur un système Linux. Bien sur on gagne du temps si les images que l'on veut afficher sont originiellement en YUV, si elle ne le sont pas alors une étape de conversion est alors nécéssaire.

- GLX : Ce mode est l'utilisation du mode OpenGL (avec l'accélération matériel) dans une fenêtre X11. Dans ce mode, nous pouvons afficher deux caméras (dans la même fenêtre). C'est ce mode qui est utilisé lors de la calibration, pour afficher les différents points trouvés. Je ne suis pas sûr, d'avoir les paramètres optimaux dans ce mode.

7.1.3 Acquisition et synchronisation de plusieurs caméras

- 1) Pour le moment, nous ne pouvons synchroniser que 2 PC reliés entre eux par un réseau ethernet et Firewire. Une amélioration intéressante du système de capture serait de pouvoir synchroniser n PC entre eux par le réseau ethernet.

- 2) Une ébauche de socket UDP a été créé pour pouvoir lancer l'acquisition en même temps, je pense qu'il y a moyen de l'amélioré (un client ne recoit pas toujours son ordre, ou le serveur n'arrive pas à s'initialiser).

- 3) L'acquisition sur 6 caméras en parallèle, nous a posé quelques problèmes, il faudrait pousuivre les tests dans cette configuration pour savoir si les nombreux problèmes rencontrés sont dut à une incompatiblitée de la librairie *libdc* dans le cas d'un réseau Firewire (test avec un réseau entre deux boitier de synchronisation en particulier). Ou si le problème viendrait d'une imcompatibilité entre les deux boitier eux-même (pendant les tests un boitier à sembler par moment ne pas fonctionner).

- 4) Comme dans le cas de l'affichage en temps réel, les différents algorithme de conversions utilisés ne sont peut être pas optimaux, certain pourrait être modifié pour diminué le temps de la conversion d'une image d'un format à un autre.

7.2 Bugs connus

7.2.1 Initialisation du système de capture

- 1) Le système n'a été testé qu'avec un nombre limité de caméra différentes. Il peut donc subsitué des erreurs dut à un manque de caméras différentes.

- 2) Il est impossible d'initialiser la caméra sony dans le mode 640*480 YUV444, la cause en est inconnus.

- 3) Il est arriver avec les caméra sony (et uniquement celle-là) que le système de capture cale, mais après il était impossible de réinitialiser les caméras SONY, même en les déconnectant, ou en rebootant le PC. Les caméras redevennaient opérationnel au bout d'un certain temps (environs 5 minute).

- 4) Il est impossible d'initialiser deux caméras d'un même type avec deux fichier de configuration différents. Mais on peut après leur initialisation modifié leurs feature respectifs.

- 5) Si le système de capture cale, il se peut que le drivers Firewire soit atteint, ce qui entraine une impossibilité totale de lancer le système, la seule solution est de relancer le PC.

7.2.2 Fermeture du système de capture

- 1) Lorsque on utilise plusieurs systèmes de capture reliés par un réseau Firewire, le système cale au moment de la fermeture du programme, nous pensons que cela est dut à la librairie *libdc* qui ne sait pas encore bien géré les réseau Firewire.

7.2.3 Affichage des caméras

- 1) Lorsqu'on utilise l'affichage SDL, le programme cale de manière aléatoire, lors de la fermeture du système SDL. La cause en est inconnu.

- 2) Lorsqu'on utilise l'affichage GLX, le programme peut caler lors de la fermeture du système X11, la cause en est identifié, il s'agit d'un bug dans le drivers *NVDriver* fournit par NVIDIA, qui gère l'accélération graphique hardware.

7.2.4 Synchronisation

- 1) Lors de la recherche de l'image complémentaire (prise en même temps) on ne supprime pas les images qui ne peuvent avoir aucun complémentaire. Mais comme le compteur Firewire est cyclique (cycle de 128 seconde), il se peut que l'on trouve deux image possible lors de la recherche mais prise à 128 seconde de diatance. Ce bug ne peut apparaitre que sur des prises de vue supérieur à 128 seconde.

- 2) Lors de la sauvegarde, des images apparié, l'utilisateur n'a pas le choix du format de sauvegarde, c'est obligatoirement le PPM.

- 3) lors de l'appariment par réseau, nous ne gérons qu'un serveur et un client. Il faudrait modifier notre protocole réseau pour que le serveur puisse communiqué avec plusieurs clients en même temps.

7.2.5 Sauvegarde

- 1)

7.2.6 Les différentes bibliothèque annexes

- 1) *OpenCV* : Nous ne savons pas combien de temps cette bibliothèque va rester open source, ue alternative devrait être trouvé, au cas ou intel modifirai les termes de la licence gérant cette bibliothèque. *OpenCV* utilise la bibliothèque *IPL* qui vient de changer de licence, elle vient de passer de opne source à payante

- 2) *libdc* : Cette bibliothèque est encore en dévellopement, une nouvelle version est sortie durant la phase terminal de ce projet, et n'a pas put être testé. Nous recommandons de toujours utilisé la dernière version disponible.
- 3) *libraw* : Cette bibliothèque est elle aussi toujours en dévellopement, mais elle a atteint un certain degrès de maturité, elle peut ê considéré comme stable.

Conclusion

Pendant ce stage j'ai eu la mission de mettre en place un système d'acquisition d'images à partir de caméras Firewire. Ce système devait permettre d'acquérir facilement des images en provenance d'une ou plusieurs caméras numériques connectées sur un bus 1394. Il est destiné à être utilisé par les équipes MOVI et IMAGIS pour la capture de mouvement.

Les objectifs primordiaux étaient de faciliter l'acquisition d'images en développant une bibliothèque spécialisée. Les points importants étaient de permettre un stockage des images sur le disque dur, d'avoir une synchronisation entre les différentes cameras utilisées et de faciliter le traitement en temps réel de ces images.

En développant la bibliothèque util_1394 j'ai atteint une grande partie des objectifs demandés. Cette bibliothèque facilite l'utilisation de caméras numériques en fournissant un ensemble de fonctions de haut niveau. Elle permet d'effectuer des traitements temps réel sur les images ou des les stocker sur le disque dur pour les traiter en différé. Ces résultats ont été obtenus en résolvant constamment des problèmes tels que les conversions de formats d'images ou l'apprentissage d'outils sans documentation.

Ainsi j'ai pu m'enrichir personnellement en mettant à l'œuvre les connaissances acquises durant ma scolarité à l'UFRIMA. La programmation en C et l'étude des systèmes m'ont particulièrement servi pour la réalisation de la bibliothèque et la configuration de LINUX. J'ai ainsi acquis une certaine expérience en programmation sous LINUX et dans les domaines de la vidéo numérique. J'ai pu aussi améliorer mon anglais en assistant à des conférences faites par des chercheurs anglophones et en côtoyant quelques stagiaires étrangers.

En fin de compte lors de ce stage d'étée j'ai pu découvrir un aspect du monde des laboratoires de recherche fort enrichissant. Et dans ce cas précis j'ai eu un aperçu très positif de la manière dont fonctionne un institut de recherche en informatique.

ANNEXES

Annexe A

Fichier de configuration

Exemple de fichier de configuration les commentaires sont en rouges

```
#*****************************************************************
# fichier de configuration par défaut !!
#
# marque : Point Grey Research Marque de la caméra
# model : Dragonfly Model de caméra
# fichier generer le : Tue Jul 23 17 :41 :43 2002 Date et heure ou le fichier à été généré

Format de l'image
# choix du format
#
# 0 FORMAT VGA NONCOMPRESSED
# 1 FORMAT SVGA NONCOMPRESSED
# 2 FORMAT SVGA NONCOMPRESSED
# 3 FORMAT RESERVED
# 4 FORMAT RESERVED
# 5 FORMAT RESERVED
# 6 FORMAT STILL IMAGE
# 7 FORMAT SCALABLE IMAGE SIZE :
# attention, seul le format ""0 FORMAT VGA NONCOMPRESSED""
# est implémenté pour l'instant.

0

Mode de l'image
Selon le type de caméra certain mode seront annoté comme inutilisable.
La génération automatique de fichier de config personnalise chaque fichier en fonction des
options que permet ou ne permet pas la caméra
# choix du mode
#
# 0 MODE 160x120 YUV444 (mode impossible avec cette caméra)
# 1 MODE 320x240 YUV422 (mode impossible avec cette caméra)
# 2 MODE 640x480 YUV411 (mode impossible avec cette caméra)
# 3 MODE 640x480 YUV444 (mode impossible avec cette caméra)
# 4 MODE 640*480 RGB (mode impossible avec cette caméra)
# 5 MODE 640x480 MONO 8
# 6 MODE 640x480 MONO 16

5

Le framerate ainsi que les autres options seront communes à toute les caméras de ce type
branché sur le système de capture
# choix du framerate
```

```
#
# 0 FRAMERATE 1,875
# 1 FRAMERATE 3,75
# 2 FRAMERATE 7,5
# 3 FRAMERATE 15
# 4 FRAMERATE 30
# 5 FRAMERATE 60

4

Par défaut, cette vitesse doit être réglée au maximum
# choix de la vitesse de transmission.
#
# 0 SPEED 100 Mb/s
# 1 SPEED 200 Mb/s
# 2 SPEED 400 Mb/s

2

Maintenant nous donnons les paramètres propres à chacun des 19 features
Nous détaillerons le premier, les autres sont construit de façon similaire
Lors de la génération du fichier de configuration, nous testons si les features sont présent
, le fichier ainsi généré est le reflet exacte de la compositions de cette caméra
# les options propres a chaque camera
##############################################################
# feature brightness :
#********************
La ligne suivante indique si le feature est présent sur ce type de caméra
Si la réponse est non, ce feature sera ignoré et on cherchera le suivant indiqué par le
symbole $
$ option presente : OUI
# information sur le feature
Nous affichons tous les renseignements que nous avons put obtenir lors de la création du
fichier de configuration
# interrupteur on/off : non present
# mode auto : non present
# mode manuel : present
# les valeurs minimum et maximum
# valeur minimal : 0
# valeur maximal : 255
# la config
C'est içi que l'utilisateur peut changer les paramètres du feature
Dans le cas du mode, il ne peut passer comme valeur, que les valeurs permises, elles sont
```

marquées dans la parenthèse
@ mode(MAN) : MAN
Dans le cas de la valeur, il faut que celle ci soit comprise entre la valeur maximal permise et la valeur minimal permise. Un test de validité de cette valeur est fait au momant de la lecture du fichier de configuration par le système de capture
@ valeur : 0
#**********

feature exposure :
#**********
$ option presente : OUI
# information sur le feature
# interrupteur on/off : present
# mode auto : present
# mode manuel : present
# les valeurs minimum et maximum
# valeur minimal : 0
# valeur maximal : 1023
# la config
@ mode(OFF/AUTO/MAN) : AUTO
@ valeur : 634
#**********

feature sharpness :
#**********
$ option presente : NON
#**********

feature white balance :
#**********
$ option presente : NON
#**********

feature hue :
#**********
$ option presente : NON
#**********

#feature saturation :
#**********
$ option presente : NON
#**********

```
# feature gamma :
#*******************
$ option presente : NON
#*******************

# feature shutter :
#*******************
$ option presente : OUI
# information sur le feature
# interrupteur on/off : non present
# mode auto : present
# mode manuel : present
# les valeurs minimum et maximum
# valeur minimal : 2
# valeur maximal : 532
# la config
@ mode(AUTO/MAN) : AUTO
@ valeur : 532
#*******************

# feature gain :
#*******************
$ option presente : OUI
# information sur le feature
# interrupteur on/off : non present
# mode auto : present
# mode manuel : present
# les valeurs minimum et maximum
# valeur minimal : 0
# valeur maximal : 1023
# la config
@ mode(AUTO/MAN) : AUTO
@ valeur : 233
#*******************

# feature iris :
#*******************
$ option presente : NON
#*******************

# feature focus :
#*******************
$ option presente : NON
```

#**********

feature temperature :
#**********
$ option presente : NON
#**********

C'est le feature qui permet d'utilisé un GBF externe pour envoyé les top de prise d'image
feature trigger :
#**********
$ option presente : OUI
# information sur le feature
# interrupteur on/off : present
# polarity : present
Active ou non le trigger, si le trigger est activé mais si aucun GBF n'est connecté au caméra, alors cette feature est ignoré
@ trigger (ON/OFF) : OFF
@ polarity (ON/OFF) : OFF
@ mode du trigger(0/1/2/3) : 0
#**********
feature zoom :
#**********
$ option presente : NON
#**********

feature pan :
#**********
$ option presente : NON
#**********

feature tilt :
#**********
$ option presente : NON
#**********

feature optical filter :
#**********
$ option presente : NON
#**********

feature capture size :
#**********
$ option presente : NON

```
#******************

# feature capture quality :
#******************
$ option presente : NON
#******************

##################################################################
# fin du fichier de configuration
```

Annexe B

Mode d'emploi de l'installation

mise à jour : 30 juillet 2002

Mode d'emplois de l'installation d'un système firewire sous linux

Configuration de départ :

- Processeur compatible x86.
- mandrake 8.2 (noyau d'origine, non patché).
- kernel 2.4.8 ou supérieur.
- XFree86 4.2.0 ou supérieur.

Les différents composants logiciel dont nous avons besoin :

- libraw (version 0.9 ou supérieur)
- libdc (version 0.8.3 ou supérieur)
- util_1394
- GTK (bibliothèque et librairie de dévellopement)
- SDL (bibliothèque et librairie de dévellopement)
- coriander 0.24 ou supérieur
- OpenCv
- Les librairie de dévellopement de XFree86
- Le support OpenGL de la carte graphique
- Le support OpenGL pour XFree86

Étape 1 : installation de OpenGL

En premier lieu, il faut installer les drivers de la carte graphique pour avoir l'accélération matériel. Pour une carte nvidia, il faut récuperer sur le site web de Nvidia les dernières paquetages sources. Les récuperer sous la forme de NVIDIA_GLX-*.src.rpm et NVIDIA_kernel-*.src.rpm

Les recompiler par la commande :
rpm –rebuild NVIDIA_GLX-*.src.rpm
rpm –rebuild NVIDIA_kernel-*.src.rpm

Pour les installer :
aller dans le répertoire /usr/src/RPM/RPMS/i586/ et taper les commandes suivantes :
rpm -ivh NVIDIA_kernel.i686.rpm
rpm -ivh NVIDIA_GLX.i686.rpm

Puis il faut modifier le fichier de configuration de XFree86 :
En root, ouvrir le fichier /etc/X11/XF86Config-4 et remplacer dans la section Device la ligne Driver "nv" par Driver "nvidia".
Dans la section module enlever la ligne load "dri" si elle existe puis rajouter la ligne load "glx".
Relancer le serveur X, vous devez voir apparaitre le logo de nvidia au redémarrage du

serveur graphique.

Étape 2 : vérification du mode OpenGL

Taper (en root) la commande suivante cat /var/log/XFree86.0.log et vérifier que l'on voit bien apparaitre (II) loading extension XVideo à l'écran.
Taper la commande xvinfo pour vérifier que le module xvideo de XFree86 est bien chargé , si il l'est vous devez voir apparaitre pleins de commentaire.
Ce module est indispensable pour pouvoir avoir la visualisation en temp réel.

Étape 3 : installation de la carte firewire

REMARQUE : il faut que la carte d'acquisition soit installée dans le PC, avant de taper les commandes suivantes.

REMARQUE : pour pouvoir exécuter cette étape, il faut être root.

Taper les commandes suivantes :

- mknod -m 600 /dev/raw1394 c 171 0
- modprobe -dk ieee1394
- modprobe -dk ohci1394

Dans /etc/modules.conf, rajouter les lignes :

- alias char-major-171 raw1394
- alias char-major-172 video1394
- pre-install raw1394 /sbin/modprobe -k ohci1394
- pre-install video1394 /sbin/modprobe -k ohci1394

Dans modules rajouter :

- char-major-171
- char-major-172

Relancer l'ordinateur, maintenant les modules doivent se charger automatiquement lors du démarage de l'ordinateur.

Après avoir taper la commande **lsmod**, on doit avoir les résultats suivants :
video1394 15344 1
raw1394 6896 3
ohci1394 17136 4 (autoclean) [video1394]
ieee1394 24848 0 [video1394 raw1394 ohci1394]

Une dernière vérification, taper la commande suivante : **ls -ald /dev/*1394***, on doit avoir avoir le résultat suivant :

crw-rw-rw- 1 root root 171, 0 jan 1 1970 /dev/raw1394
drwxr-xr-x 1 root root 0 jan 1 1970 /dev/video1394/

Si on observe que les droits sont différents, ils faut les mettrent à jour en tapant :**chmod ugo+rw /dev/*1394***.

Étape 4 : installations des librairies annexes

Dans cette section, nous installerons toutes les bibliothèque dont nous aurons besoins par la suite.

Pour pouvoir installer toutes les bibliothèque, on doit être loggé en root.

Il faut installer les bibliothèque suivante :

- GTK et GDK (présent en RPM sur le CD de Mandrake).
- Librairie de XFree86 (présent en RPM sur le CD de Mandrake).
- Librairie SDL (présent en RPM sur le CD de Mandrake).
- OpenCV (présent sur le CD util_1394).
- Libraw (présent sur le CD util_1394).
- Libdc (présent sur le CD util_1394).

Étape 5 : installation de util_1394

Installation de Util_1394.

1) Installer les sources.
2) Taper la commande **make**.
3) Se logger en root puis taper **make install**. Puis se délogger.
4) Compiler les exemples en tapant **make exemple**.

Quelques commandes utiles :

make clean Nettoie les sources de tous les fichiers temporaire et des exécutables.
make image Détruit toutes les images et tous les fichiers temporaires créé par util_1394 pour stocker les images.
make doc Fabrique la documentation relative à la librairie Util_1394.

Annexe C

Liste et description des exemples

descriptif des différents fichier exemples

exemple01 :
Commande : <exemple01> <numéro camera> <dma>
Description : on donne le numero de la camera (entre 0 et nombre de camera-1) et si on veut utiliser le dma (mettre 1) ou non (mettre 0)

exemple02 :
commande : exemple02 <nb de photos> [noir et blanc]
prend le nombre de photos passer en paramètre pour chaque caméra

exemple03 :
Description : donne en sortie les différents paramètres utilisé par la caméra donne leur valeur courante, la valeur max et la valeur min

exemple04 :
Description : affiche le status du système

exemple05 :
Description : programme pour tester l'initialisation des cartes et des caméras

exemple06 :
Description : donne le nombre de cartes connectéee et le nombre de caméra connectées à chaque carte

exemple07 :
Description : pour connaitre le temp de chargemet d'une image et de sa conversion (sans threads)

exemple08 :
Description : pour connaitre le temps de chargemet d'une image et de sa conversion (sans threads)

exemple09 :
Description : pour connaitre le temps d'un cycle processeur et le temps qu'a besoin le processeur pour remplir le registre timer

exemple10 :
Description : pour connaitre le temp de chargemet d'une image et de sa conversion (avec threads)

exemple11 :
Description : pour connaitre le temps de la gestion des threads (sémaphore)

exemple12 :
Description : pour connaitre le temps de la gestion des threads (mutex)

exemple13 :
Description : test de sortie de l'image sous plusieurs formats differents (ppm pgn bmp jpg)

exemple14 :
Description : test des threads lance iso sur 2 caméras et capture sur les 2 caméras

exemple15 :
Description : test convertion séquence en séquentiel

exemple16 :
Description : test malloc sur très grosse taille

exemple17 :
Description : donne le cycle time dans le cas de non thread

exemple18 :
Description : compteur pour version non thread

exemple19 :
Description : test de l'affichage graphique sur 2 caméras

exemple20 :
Description : test de l'affichage graphique sur 1 caméra

exemple21 :
Description : test de la pause pendant l'affichage (sur 1 caméra)

exemple22 :
Description : test de l'appariment des images

exemple23 :
Description : test du changement des features en temp réel (sur 1 caméra)

exemple24 :
Description : test du calibrage d'une caméra

exemple25 :
Description : test serveur client udp

exemple26 :
Description : test serveur client tcp

Annexe D

Documentation de la librairie util_1394

Bibliographie

[1] Rémi Fontan. Mise en place d'un systeme d'acquisition d'images composée d'une caméra iee1394. Master's thesis, Département Informatique, IUT 2 Grenoble, 2001.

[2] Christian Rolland. *LaTeX par la pratique*. O'REILLY, 1999.

[3] IEEE Computer Society. *IEEE Standard for a Hight Performance Serial Bus*. IEEE std 1394-1995, August 1996.

[4] IEEE Computer Society. *IEEE Standard for a Hight Performance Serial Bus - Amendment 1*. IEEE std 1394a-2000, June 2000.

[5] SITE WEB. doxygen. `http://www.doxygen.org/index.html`.

[6] SITE WEB. La bibliothèque dc1394. `http://sourceforge.net/projects/libdc1394/`.

[7] SITE WEB. La bibliothèque raw1394. `http://sourceforge.net/projects/libraw1394/`.

[8] SITE WEB. La techonologie ieee1394. `http://iseea.online.fr/Rapports/a\_jolly/html/bus/bus6.html`.

[9] SITE WEB. La techonologie ieee1394. `http://www.ti.com/sc/docs/products/msp/intrface/1394/tech.htm`.

[10] SITE WEB. La techonologie ieee1394. `http://www.multimania.com/maddog/HARD/HowFW.html`.

[11] SITE WEB. La techonologie ieee1394. `http://www.skipstone.com/compcon.html`.

[12] SITE WEB. Le logiciel coriander. `http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/coriander/`.

[13] SITE WEB. Le projet ieee1394 sous linux. `http://www.linux1394.org/`.

[14] SITE WEB. L'equipe movi. `http://www.inrialpes.fr/movi/`.

[15] SITE WEB. Les formats yuv. `http://www.webartz.com/fourcc/fccyuv.htm`.

[16] SITE WEB. L'inria rhônes-alpes. `http://www.inrialpes.fr`.

[17] SITE WEB. Point grey research, inc. `http://www.ptgrey.com`.

[18] SITE WEB. Qt designer. `http://doc.trolltech.com`.