DEPARTEMENT INFORMATIQUE – IUT 2 GRENOBLE

Année Universitaire 2000-2001

MEMOIRE DE STAGE

MISE EN PLACE D'UN SYSTEME D'ACQUISITION D'IMAGES COMPOSEE D'UNE CAMERA IEEE1394
INRIA Rhône-Alpes

(Stage du 2 avril au 30 juin 2001)

Présenté par
Rémi FONTAN

Jury

IUT : Mr Laurent BONNAUD
IUT : Mme Anne PENSO
Société : Mr Hervé MATHIEU

Remerciements

Je remercie Radu Horaud 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 Radu Horaud et Hervé Mathieu pour m'avoir accueilli chaleureusement et de m'avoir suivi avec attention tout au long du stage.

Je remercie également Bill Triggs et Frédérick Martin pour leurs aides précieuses. Leurs coopérations ont été très utiles et m'ont beaucoup aidé à résoudre de nombreux problèmes.

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 de tels projets pendant dix semaines.

Table des matières

Introduction

L'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. Créée 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 et d'Hervé MATHIEU que s'est déroulé le stage. Radu HORAUD est directeur de recherche du projet MOVI et Hervé MATHIEU est ingénieur de recherche au Service Robotique.

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.

Le projet MOVI forme une des équipes de recherche du laboratoire GRAVIR. Et à ce titre, appartient à l'INPG, à l'Université Joseph Fourier et est associé au CNRS. MOVI est un projet commun à l'IMAG et à l'unité de recherche Rhône-Alpes de l'INRIA.

L'équipe de MOVI est composée de 6 chercheurs permanents et d'une dizaine de thésards et post doctorats. En période de fin d'année scolaire une dizaine de stagiaires viennent généralement compléter les rangs de l'équipe.

L'équipe du Service Robotique

Le Service Robotique possède un statut particulier au sein de L'INRIA. Il est charger de mettre à disposition des projets de recherche le plus de ressources matérielles et logicielles possibles. Leur but est de fédérer l'effort expérimental en favorisant les expérimentations inter-projets, la mise en commun des moyens expérimentaux et des outils réutilisables (environnement de développement, machine de vision...). L'équipe est composé de quatre ingénieurs de recherche, d'un technicien et d'une assistante de projet ainsi que de nombreux stagiaires.

Description du sujet de stage

Utilité du stage

Ce stage consiste à mettre en place un système d'acquisition d'images à partir de caméras numériques sur bus 1394 sous Linux. Ce système permettra d'utiliser des images de bonne qualité et de simplifier la chaîne d'acquisition en supprimant l'étape de conversion en numérique. Etant donné que la tendance actuelle est de migrer progressivement vers le tout numérique, il est normal de s'intéresser à ce type de technologie qui commence à être exploitable sous Linux.

Le bus 1394 est particulièrement intéressant en vision car il permet d'utiliser des caméras numériques transmettant des images de bonnes qualités ce qui est indispensable pour pouvoir faire du traitement dessus. Il permet aussi de synchroniser plusieurs caméras et ainsi d'effectuer de la stéréovision.

A qui cela va servir :

Les résultats de mon stage serviront plus particulièrement à l'équipe du projet MOVI et au service Robotique mais aussi à toute personne de l'INRIA voulant se servir de camera 1394. Le Service Robotique en plus de faire de la recherche en robotique s'occupe de mettre en place des systèmes matériels et logiciels pour d'autre projet tel que MOVI dans le cadre de mon stage. Le projet MOVI est spécialisé dans la recherche en vision et stéréovision. C'est donc dans le projet MOVI mais encadré par le service robotique que ce passe mon stage.

En quoi cela consiste :

Le système d'acquisition d'images est divisé en plusieurs parties. Tout d'abord il est composé d'une partie matérielle qui comprend l'ordinateur, la carte et les caméras. Mettre en place le système reviendra à installer et configurer ce matériel. Ensuite il y a la partie logiciel qui est composée de bibliothèques et de programmes permettant d'acquérir des images.

L'objectif principal est de réaliser une bibliothèque en C qui permettra d'automatiser et de simplifier certaines taches.

Les objectifs attendus sont:

- acquisition d'images
- stockage d'image au format "ppm"
- possibilité de traitement d'images à la volée ou en différé
- séparation des phases d'initialisation, d'acquisition et de fermeture

Cette bibliothèque s'intitulera " util_1394 " et fournira un ensemble de fonctions facilitant l'utilisation des caméras 1394.

I Evaluation de la situation initiale

1 Présentation de la technologie IEEE1394

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 adopte en 1995 par le consortium IEEE en tant que standard identifié par le nombre 1394. 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 mis à jour sous le nom de 1394a, et sera de nouveau mis 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.

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 model OSI, 1394 s'étend sur 3 niveaux : le niveau physique, le niveau liaison et le niveau réseau.

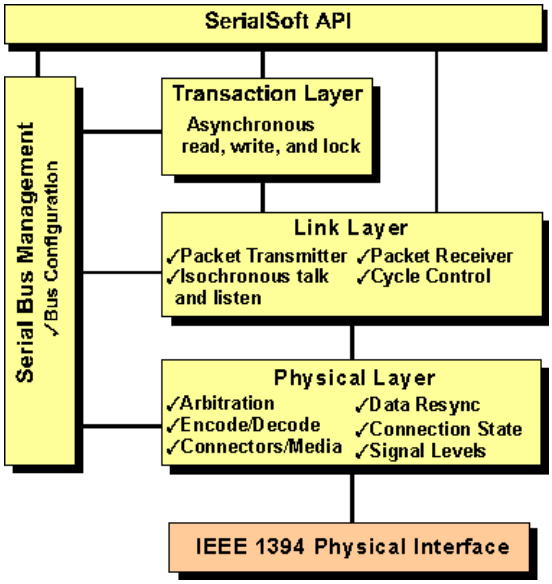


fig.1

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

Niveau liaison:

Ce niveau s'occupe de gérer l'envoie et la réception de paquets de données. La transmission isosynchrone est gérée à ce niveau.

Le mode de transmission isochrone est particulier. 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 espace-temps demandé à chaque cycle. Un seul 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 asynchrone. Ce mode de transmission garanti 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 quantifier car il dépend de taux l'utilisation du bus 1394 par d'autres transmissions pour d'autres appareils communicants 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.

Parallèlement à ces trois couches il subsiste une couche de gestion de bus série. Cette couche s'étend sur les trois niveaux précèdent 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. La seule restriction est qu'il ne faut pas avoir plus de 16 câbles entre deux appareils. On peut connecter à un tel réseau 63 appareils.

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.

- les connecteurs des câbles:

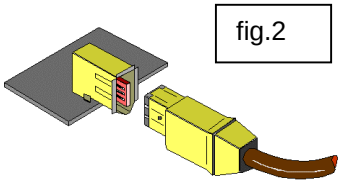


fig.2

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 inoffensives vis a 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.

- 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.

- 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.

- topologie flexible:

  Le bus 1394 ne doit pas nécessairement suivre une topologie spéciale du moment que l'on ne connecte pas plus de 16 câbles entre deux appareils.

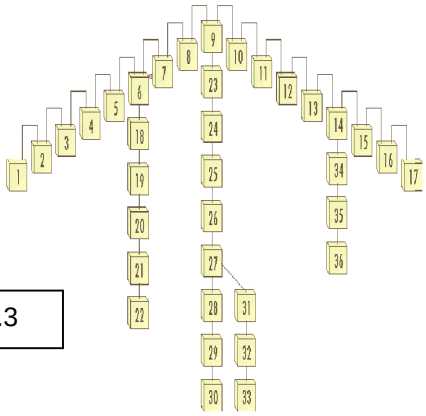


fig.3

- 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 destine à se diffuser rapidement.

2. 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 est toujours au stade expérimentale. Toutefois il ne répond pas entièrement au cahier des charges et ne peut pas être utilisé de manière optimale par MOVI.

XVision permet d'afficher à l'écran une vidéo ou le flux que reçoit une caméra numérique. Néanmoins il ne semble pas fonctionner correctement sur certaines machines du laboratoire et semble trop dépendant d'un matériel spécifique. De plus sa conception est relativement compliquée en raison des optimisations que ses concepteurs ont effectuées.

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 types 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 tache.

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

3. prise de contacte avec les outils

Outils "hardware" :

On a mis à ma disposition un PC compatible 75 Mhz, une carte interface 1394 de marque Orange Micro et une camera SONY de model DFW-VL500. Des terminaux X sont également disponibles pour accéder au compte personnel et pour utiliser les possibilités offertes par le réseau de l'INRIA.

Installation de Linux :

Le PC mis à ma disposition ne possédait pas de système d'exploitation. Je me suis donc chargé d'installer Linux Red Hat 6.2 et de le configurer.

Normalement je n'ai accès à des comptes Linux ou Unix qu'en temps que simple utilisateur, mais dans ces circonstances je dus devenir administrateur pour pouvoir faire l'installation et la configuration.

L'installation de Linux s'est passé sans aucun problème, tout le matériel a été reconnu. Le plus dur a été de configurer le réseau. Mais en demandant de l'aide et en me documentant j'ai pu obtenir un accès à Internet. Je n'ai pas cherché utiliser le NFS ou d'autres possibilités liées au réseau. Seule l'accès au réseau Internet était important.

Outils "software" :

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 existant 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 arborescence particulière. Les flèches de la figure 4 indiquent le sens des dépendances.

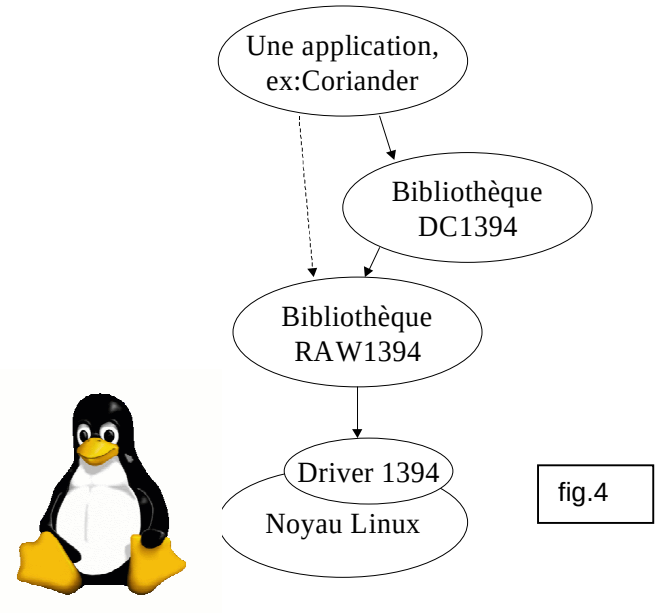


fig.4

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 en directement dans le noyau. 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.

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 et de très bas niveau et très compliquée.

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.

Utilisation de quelques logiciels :

En plus des outils standards pour la programmation et l'accès à Internet j'ai eu l'occasion d'en utiliser des particuliers:

“ animate ” pour visualiser une vidéo à partir d'un ensemble d'images.

“ wincenter ” pour pouvoir utiliser les outils Microsoft et plus particulièrement ceux de bureautique sur des stations UNIX et Linux.

II Travail effectué

1. Installation du système d'acquisition sur plusieurs ordinateurs

Pendant le déroulement du stage j'ai du procéder à l'installation et la configuration du bus 1394 sur plusieurs ordinateurs. J'ai commencé par installer le système sur l'ordinateur mis à ma disposition au début du stage puis en fin de stage sur un portable mis à la disposition de l'équipe de MOVI. L'installation du système sur le portable permettra d'utiliser les caméras numériques sans être dépendant d'un lieu précis et permettra ainsi d'effectuer des démonstrations facilement en dehors des locaux de l'INRIA.

L'installation du système se déroule en trois étapes successives, la configuration du noyau, l'installation de quelques bibliothèques et l'exécution de plusieurs tests pour valider l'ensemble.

Configuration du noyau :

Linux supporte la technologie 1394 mais encore de manière expérimentale. Les drivers ne sont pas encore finalisés et il subsiste probablement quelques bugs. L'état expérimental de l'implémentation du bus empêche son utilisation systématique dans les distributions Linux standards. Il est par conséquent indispensable faire une mise à jour de Linux en recompilant le noyau avec les options adéquates. Généralement on choisira la version la plus récente compatible avec la distribution utilisée.

Compilation du noyau:

Pendant la compilation du noyau, cocher l'option " Prompt for development and/or incomplete code/driver " permet d'activer les drivers encore à l'état expérimental. Lors de l'activation des drivers du bus 1394 on peut choisir de les utiliser sous forme de modules. Dans ce cas il est important de noter que les modules seront compilés sous les noms " ohci1394 ", " raw1394 ","video1394 ".

" ohci1394 " est le module correspondant au driver générique de la carte interface 1394. Certaines cartes ne sont pas compatibles avec ce driver et en utilisent un autre. Néanmoins la plupart des cartes sont compatibles avec le driver générique.

" raw1394 " correspond au module qui permet de communiquer sur le réseau 1394. Les bibliothèques bas niveau tel que(RAW1394) utilise ce module.

" video1394 " est un module spécialisé dans l'acquisition d'images à partir de caméras. Etant donné que le bus 1394 est particulièrement utilisé pour la vidéo, les développeurs Linux ont jugés intéressant de créer un tel module.

Installation des bibliothèques :

Dans les distributions Linux standard qui existent n'incorporent pratiquement jamais les bibliothèques " RAW1394 " et " DC1394 ". Il est par conséquent indispensable de procéder à leur installation. Généralement cela ne pose aucun problème en suivant la documentation fournie.

Une fois l'installation des bibliothèques effectuée, il est nécessaire de créer des points d'entrées dans le répertoire " /dev ". Les points d'entrées sont des fichiers spéciaux qui permettent de communiquer directement avec le matériel. Ils sont généralement utilisés par les drivers.

Dans ce cas deux points d'entrées sont créés :

- Le point " /dev/raw1394 " qui permet de communiquer avec la carte 1394 et par conséquent avec n'importe quels périphériques 1394. Il est utilisé par la bibliothèque " RAW1394 ".

- Eventuellement le point " /dev/video1394 " suivant les options de compilation du noyau. Ce point est spécialisé dans la communication avec les caméras numériques. Il permet d'acquérir des images très rapidement et sans surcharger le processeur. C'est la bibliothèque " DC1394 " qui l'utilisera principalement.

Premiers programmes de tests :

Pour pouvoir valider l'installation du système, il est nécessaire de procéder à quelques tests. J'ai commencé par réaliser plusieurs programmes. Chacun de ces programmes teste quelques caractéristiques. Le choix du langage C pour l'écriture de ces programmes s'est fait pour rester dans la continuité des bibliothèques utilisées.

- Test01 : Ce programme se contente d'inclure les bibliothèques et de créer un lien (" handle ") vers la carte 1394. L'intérêt est de tester si les bibliothèques sont bien installées et si le matériel (la carte 1394) est bien reconnu et supporté par l'ordinateur.

- Test02 : Le programme récupère les numéros identifiant de chaque caméra connectée à la carte. Il affiche pour chaque caméra quelques informations tel que le nom du vendeur, du model, le numéro de série…

- Test03 : A peu près le même fonctionnement que le test précédant mais utilise des fonctions différentes. Il comporte en plus une boucle infinie qui affiche à chaque fois que l'on connecte une caméra ses informations. Le but de ce test est de vérifier le bon fonctionnement du " hot plug'n play ".

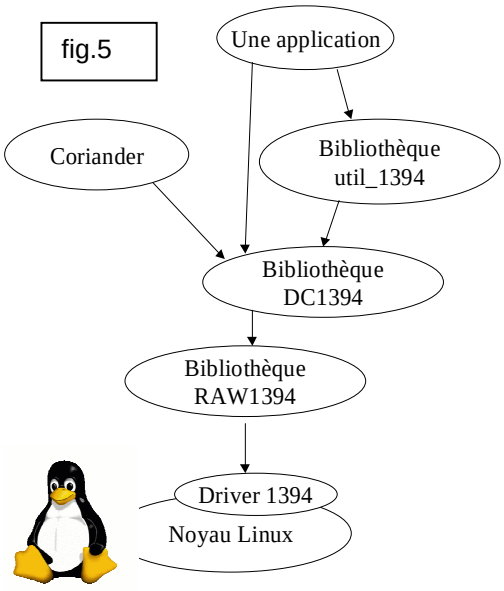
- Test04 : Pratiquement identique que le test02 mais affiche une description plus détaillée des caméras. En plus d'afficher le nom du model et du vendeur il affiche toutes les caractéristiques des caméras. C'est à dire l'état du zoom, de la mise au point (" focus "), de la balance des couleurs… Il existe une vingtaine de caractéristique pouvant être modifier. Toutes les caméras ne supportent pas toutes ces caractéristiques, certaine ne

peuvent pas changer de zoom ou d’autres sont autofocus. Ce test affiche aussi quelles caractéristiques sont supportées par les caméras connectées.

Ces quatre tests m'ont permis de me rendre compte de la réussite de l'installation du bus 1394. Ils ont été très utiles pour prendre connaissance avec les bibliothèques et ainsi apprendre à les utiliser.

2. Conception et programmation de la bibliothèque d'acquisition d'images “ util_1394 ”

Util_1394 est plus ou moins une surcouche de DC1394. Elle permet de faciliter et d’automatiser certaines taches redondantes pour le programmeur. En reprenant le schéma numéro 4 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 proche de l'homme et donc plus simple à appréhender.



Conception de la bibliothèque

Avant de se lancer dans la programmation quel que soit le projet une étape de d'analyse et de conception est nécessaire. Elle permet de 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 procède 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. On peut regrouper ces opérations sous trois grandes partie 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 model de la caméra, quel est son numéro de série mais aussi quelle résolution supporte et quel débit elle supporte. 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 ”.

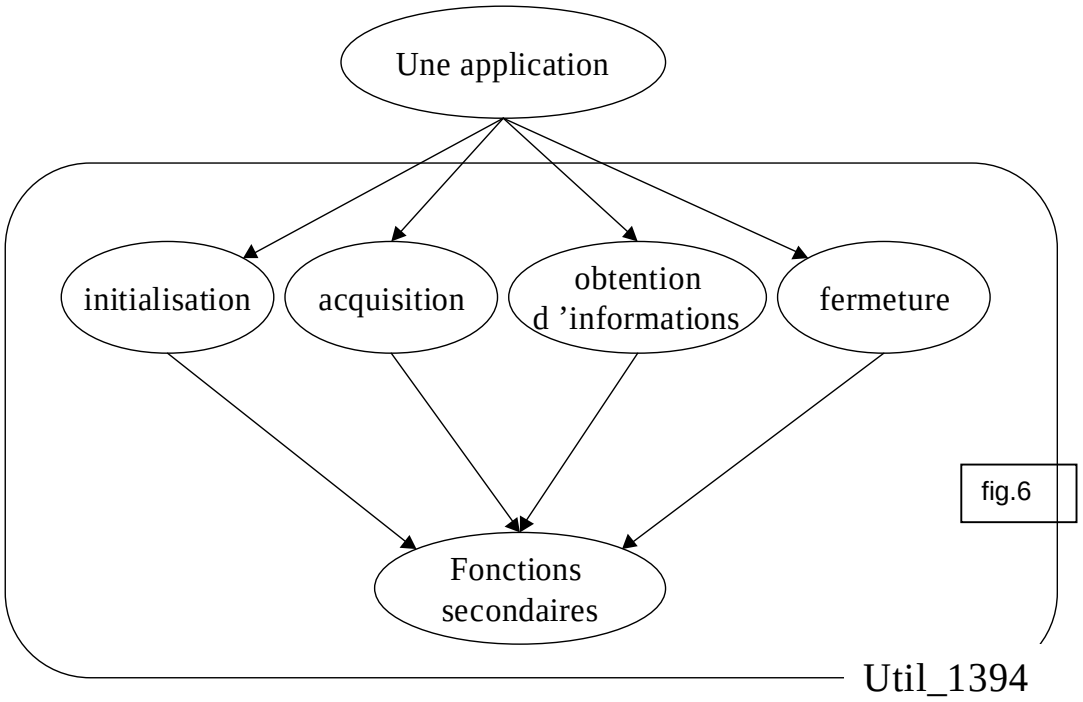


fig.6

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

Mise en évidence de plusieurs scénarios par partie :

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 qu'est ce qu'il souhaite utiliser comme configuration. Ce scénario est surtout utile lorsque l'on souhaite changer régulièrement de configuration.

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 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 standard.

Pour la partie concernant l'acquisition j'ai défini deux scénarios:

- Acquisition d'une image à partir d'une caméra.
- Acquisition d'une image à partir de chaque caméra simultanément.

L'acquisition d'une image à partir de chaque caméra permet de faire de la stéréovision ou de la vision avec n caméras. Bien sur plus il y a de caméras plus il y aura de décalage temporel entre chaque image. Il est très probablement possible de faire de la stéréovision dans de bonnes conditions bien qu'il reste à effectuer encore certains tests pour s'en assurer.

Si l'on souhaite capturer un flux vidéo il suffit de faire plusieurs captures d'image suffisamment rapprochées. Avec deux caméras on arrive à obtenir 30 images par seconde et par caméras.

Pour la partie concernant l'obtention d'information sur les caméras j'ai aussi deux scénarios différents.

- Obtention d'information pour une caméra.
- Obtention d'information pour toutes les caméras.

Comme pour l'acquisition, on peut obtenir une information sur une caméra en particulier ou sur toutes les caméras en même temps. Le deuxième scénario est présent simplement pour faciliter la tache du développeur.

Pour la partir concernant la fermeture il n'y a qu'un seul scénario possible. Le programme se fermera toujours de la même manière.

Description des fonctions secondaires:

La partie concernant les fonctions secondaires est un peu différente des autres parties. Il n'y a pas a proprement parler de scénario mais on peut quand même la décomposer en plusieurs sous-parties:

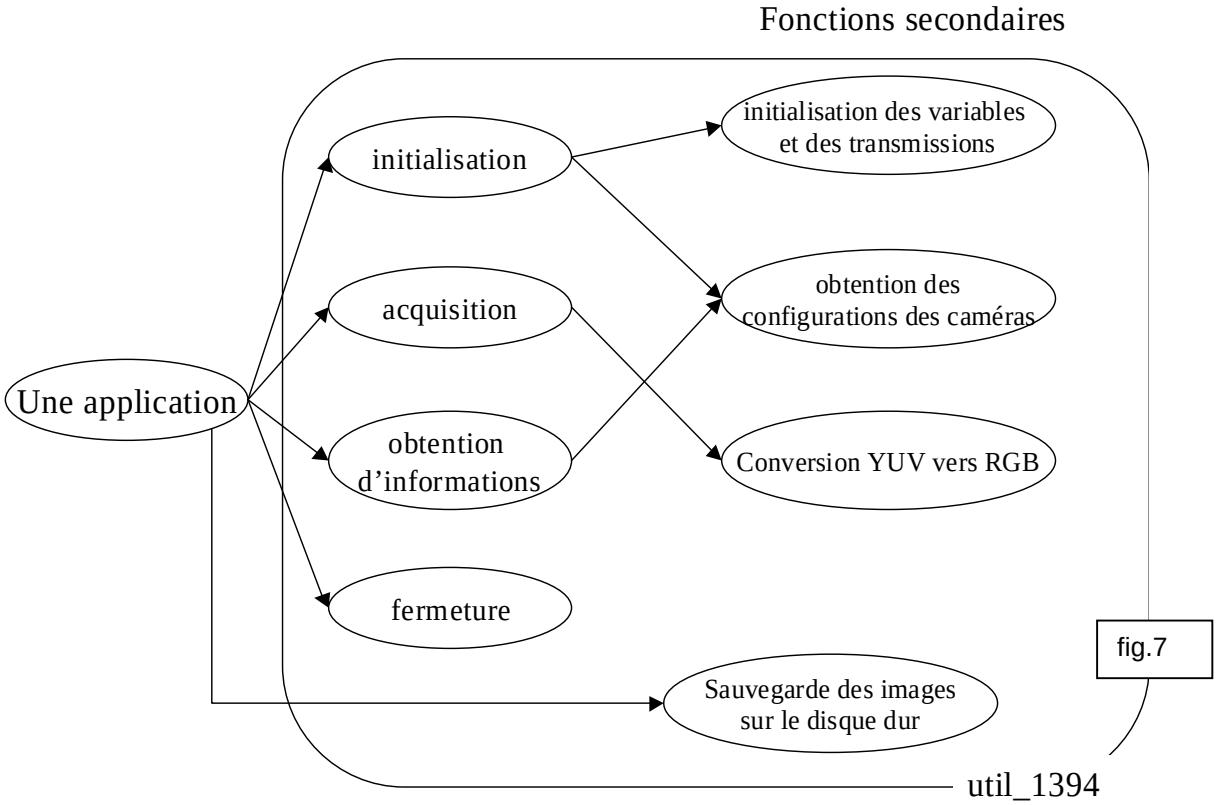
Il y a une sous-partie correspondant aux fonctions d'initialisation de la transmission et des variables. Ces fonctions ne font appel qu'à la bibliothèque DC1394. Elle est chargée d'automatiser ces appels en les regroupants dans quelques fonctions.

Une autre sous-partie regroupe les fonctions interagissant avec l'utilisateur ou avec des fichiers. Elles sont spécialisées dans l'obtention des configurations des caméras en début de programme.

Une autre sous-partie comporte exclusivement les fonctions de conversion d'images des formats YUV au format RGB. Ces fonctions sont indispensables si l'on souhaite pouvoir visualiser ces images.

La dernière partie ne comporte jusqu'à présent qu'une seule fonction. Elle sert à sauvegarder sur le disque dur une image. Cette fonction est particulière et est un peu en dehors des fonctions dites secondaires car elle est généralement directement appelée par l'utilisateur et non par une autre fonction.

Organisation finale:



Sur ce schéma on observe les dépendances entre les différentes parties et les groupes de fonctions secondaires. On peut remarquer que fermeture ne possède aucune dépendance avec des fonctions secondaires. La partie fermeture ne comportant qu'un seul scénario il aurait été inutile de créer des fonctions secondaires. Dans le cas de la partie acquisition on constate qu'il n'y a qu'une dépendance avec les fonctions de conversions. Ceci est dû au fait qu'il existe plusieurs scénarios d'acquisition utilisant certaines fonctions identiques.

Les dépendances entre les fonctions d'util_1394 et DC1394 ne sont pas représentées sur ce schéma en raison de leurs grands nombres. L'utilisation de la bibliothèque DC1394 est quasi omniprésente dans util_1394.

Programmation de la bibliothèque

Choix du langage de programmation :

Pour procéder à la programmation de la bibliothèque " util_1394 " j'ai choisi d'utiliser le langage de programmation C pour plusieurs raisons.

Les bibliothèques dont je dispose sont écrites en C ainsi que la plus grande partie des fonctions standards fournies par Linux. Ainsi utiliser le C permet de rester cohérent avec les autres outils à dispositions.

Le C est très utilisé au sein de l'équipe MOVI et de l'équipe robotique. Ainsi utiliser le C permet de ne pas dépayser les personnes souhaitant utiliser " util_1394 ".

La bibliothèque ne nécessitant pas une architecture orientée objet, l'utilisation de langage comme le C++ ne serait pas justifié. Dans ce cas le C satisfait le plus de conditions possibles et par conséquent c'est ce langage qui a été choisi.

Programmation par parties:

Le déroulement de la programmation s'est fait par partie. J'ai implémenté une partie à la fois pour pouvoir mieux percevoir les erreurs et ainsi gagner du temps.

Au début j'ai réalisé plusieurs tests pour vérifier et valider la configuration du bus 1394. J'ai continué ces tests, donc à partir du test05 jusqu'au test09, pour tester progressivement l'évolution de la bibliothèque util_1394.

- Test05 : C'est à partir de ce test que j'ai essayé d'acquérir des images. J'ai tout d'abord vérifié le fonctionnement des fonctions d'initialisation puis celles d'acquisition. Pour l'instant les captures récupérées ne sont ni stockées ni traitées, ce programme ne fait que des vérifications. Ce programme se charge de demander à l'utilisateur comment il veut initialiser les caméras puis prend une capture et arrête le programme.

- Test06: Ici j'ai testé l'implémentation d'une partie des scénarios d'initialisation, notamment concernant le choix des configurations des caméras. En plus j'ai inclus dans le programme de tests quelques algorithmes de conversion YUV vers RGB et de sauvegarde d'images sur le disque pour vérifier leur fonctionnement.

- Test07 : Ce programme vérifie que l'implémentation des précédants algorithmes dans la bibliothèque fonctionnent bien.

- Test08 : Le plus gros du travail a été effectué au niveau de ce programme. Dans le test précédant, il fallait compter une centaine de lignes de codes pour faire l'acquisition d'une image tandis que là seul une trentaine de lignes est suffisante.

- Test09 : C'est l'aboutissement de la bibliothèque. Tous les scénarios précédemment présentés ont été programmés. Ce dernier programme de tests est en fait pratiquement une petite application permettant de stocker sur le disque dur une séquence d'images à partir d'une caméra ou de plusieurs. N'ayant que deux caméras ma disposition je n'ai pas pu essayer avec trois caméras ou plus.

3. Problèmes résolus

La réalisation de cette bibliothèque ne s'est pas passé sans problèmes. Au contraire, au fur et à mesure du stage j'ai rencontré des problèmes quelque fois paraissant insolubles. Ces problèmes m'ont permis de beaucoup apprendre et de mettre en pratique les connaissances que j'ai pu acquérir pendant l'I.U.T.

Configuration de Linux

La configuration de Linux avec le support du bus 1394 n'a pas été évidente. A configuration d'un tel bus nécessite de faire d'importante mise à jour. Le noyau doit être recompilé avec les options adéquates mais évidemment en gardant les options précédantes. C'est notamment sur ce point là que j'ai eu des problèmes. Pour éviter de faire de nombreux tests plus ou moins aléatoires pour trouver la bonne configuration j'ai du rechercher sur d'autre machine ayant une configuration similaire et la modifier pour l'utiliser.

Un autre problème est apparu lors de la compilation du noyau. Lorsque l'on choisit quels drivers utiliser, on peut soit les utiliser sous forme de module soit les inclure directement dans le noyau. L'ordinateur mis à ma disposition possède une vielle version de Linux. Utiliser les nouveaux drivers du bus 1394 sous forme de modules posent des problèmes de compatibilité difficiles à détecter lorsque l'on ne possède pas beaucoup d’expérience dans le domaine de Linux.

Utilisation de la bibliothèque DC1394

La technologie 1394 est récente sous Linux. Les bibliothèques sont encore en cours d'écriture. Le problème avec la bibliothèque c'est qu'elle ne possède pratiquement pas de documentation. Cela a été relativement difficile pour en comprendre le fonctionnement étant donné qu'elle comporte une centaine de fonctions. Ce manque de documentation m'a fait perdre du temps notamment pour faire une capture d'image. Je me suis aidé de forum de discussion et aussi de programmes existants tel que Coriander pour apprendre à m'en servir.

Conversion du format YUV au format RGB

Les caméras numériques mises à ma disposition fournissent toutes des images dans différents formats. Les formats les plus utilisés sont basés sur le codage YUV. N'ayant pas beaucoup d'expériences dans le domaine de l'image et notamment dans les caméras j'ai eu quelques problèmes pour convertir ces images dans un format plus conventionnel.

Le format YUV code les couleurs en utilisant trois composantes représentant la luminance par Y et la chrominance par U et V. En comparaison, le format RGB ne sépare pas le codage des couleurs et de la luminosité. Tout est codé dans trois composantes représentant le rouge, le vert et le bleu.

Dans le cas du YUV, la composante Y correspond en fait à l'image en noir et blanc et les deux autres composantes codent simplement les couleurs. Pour pouvoir convertir un point codé en YUV en RGB il faut multiplier chaque composante par quelques coefficients.

R = 1,164(Y - 16) + 1.596(U - 128)
G = 1,164(Y - 16) - 0.813(U - 128) - 0.391(V - 128)
B = 1,164(Y - 16) + 2.018(V - 128)

Jusque là j'ai parlé du codage d'un point, mais les caméras utilisent plusieurs formats d'images suivant la qualité que l'on souhaite obtenir. Il existe notamment les formats YUV411, YUV422 et YUV444. La différence entre ces formats ne provient pas de la manière dont on code les couleurs mais de l'organisation des composantes des points dans l'image.

Les chiffres suivant YUV représente la proportion d'utilisation des composantes pour quatre points. Le format 444 utilise quatre composantes Y, U et V pour coder quatre points. Le format 422 utilise quatre composantes Y mais seulement deux composantes V et U pour coder quatre points. Le format 411 utilise toujours quatre composantes Y et qu'une seule composante U et V.
En représentant l'organisation en mémoire on obtient ceci:

444 : [**Y1**, U1, V1, **Y2**, U2, V2, **Y3**, U3, V3,**Y4**, U4, V4]

422 : [U1, **Y1**, V1, **Y2** ,U2, **Y3**, V2, **Y4**]

411 : [U, **Y1**, **Y2**, V, **Y3**, **Y4**]

Le plus dur a été d'identifier comment fonctionne ce format et de trouver des algorithmes optimisés. Pour cela je me suis basé sur des programmes existant en libre source. Les algorithmes que j'ai utilisés ont été dans la plus grande partie écris par des développeurs indépendants mais ont été modifiés pour fonctionner avec util_1394.

Implémentation du mode DMA

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 don cutiliser les deux modes.

Evolution de la bibliothèque DC1394

La bibliothèque DC1394 est en constante modification. En milieu de stage, alors que j'avais déjà bien avancé la réalisation d'util_1394, DC1394 a changé de version. Le problème étant qu'il y a eu de nombreuses modifications incompatibles avec la version précédante. Par conséquent j'ai du modifier dans de nombreux endroit mon code pour pouvoir le faire fonctionner à nouveau.

Utilisation d'util_1394 sur un autre ordinateur

Une fois la conception et la réalisation finie j'ai installé ma bibliothèque sur une autre machine possédant une carte 1394. Cette machine est à disposition de l'équipe MOVI notamment pour utiliser le système d'acquisition 1394.

L'installation a posé problème en raison d'un conflit de bibliothèque. Sur la même machine était installé deux fois DC1394 mais avec deux versions différentes. Le problème étant que les programmes de tests se compilaient avec la version la plus récente et s'exécutaient avec l'ancienne version de DC1394. Ceci engendre de nombreux bugs très difficile à comprendre. C'est après cette épreuve que je me suis rendu compte que ce ne sont pas les erreurs les plus simples qui sont les plus faciles à résoudre.

III Mise en fonction de la bibliothèque

1. Premières impressions

Ce n'est que très proche de la fin du stage que l'on a commencé à utiliser le fruit de mon travail. Un stagiaire de l'équipe MOVI a eu besoin d'utiliser un système d'acquisition vidéo pour pouvoir faire du traitement temps réel avec et util_1394 correspondait à ses besoins

La mise en fonction a été un succès partiel. C'est un succès du point de vue du fonctionnement qui semble être tout à fait correcte mais la bibliothèque ne correspondait pas tout à fait aux exigences de l'utilisateur.

- Images en noire et blanc: dans le domaine de la vision de nombreux calculs se font en noir et blanc pour accélérer la vitesse. Ma bibliothèque ne permet pas d'acquérir des images directement en noire et blanc, mais simplement en couleur. Par conséquent une conversion est nécessaire ce qui alourdit la charge du processeur inutilement.

- Visualisation temps réel : La visualisation en temps réel des images est intéressante pour bien cadrer la caméra et éventuellement afficher des résultats en temps réel sur l'écran. Ma bibliothèque ne permet pas ce type de fonctions, il est nécessaire de recourir à une bibliothèque spécialisée dans l'affichage graphique.

2. Perspectives envisagées

La réalisation d'util_1394 n'étant pas parfaite et nécessitant des changements mineurs il est important de s'intéresser à ces changements envisagés. Ces modifications seront très probablement effectuées entre la soutenance et la fin de la rédaction du rapport ou simplement pendant l'éventuelle période post stage que j'effectuerai.

Simplification de la mise en œuvre d'util_1394

Util_1394 est à l'heure actuelle fournie sous forme de fichier sources ainsi son utilisation impose de la compiler avec les programmes qui l'incluent. De plus lorsque l'on fait une modification sur la bibliothèque tous les programmes en dépendant doivent être recompilés pour tenir comptes de ses changements. Une manière d'éviter ces inconvénients est de la mettre sous forme de bibliothèque dynamique.

Simplification de l'installation d'util_1394

Pour l'instant la bibliothèque ne comporte pas d'installation simplifiée ou automatisée. Une telle solution est envisagée en utilisant dans un premier cas un système de scripts ou de

manière plus complète en utilisant un système de gestion de paquetage comme le système RPM de Red Hat ou DEB de Debian.

Répondre aux nouveaux besoins

Comme j'ai pu le constater lorsque l'on a commencé à utiliser util_1394, de nouveaux besoins sont apparus en plus des petites finitions qu'elle nécessite.

Pour l'instant util_1394 ne supporte pas tous les modes prévus par les caméras à ma disposition. Leur implémentation n'a pas été faite faute de temps mais elles sont prévues à court terme.

L'implémentation d'une visualisation en temps réel sur l'écran est envisagée en utilisant une bibliothèque spécialisée dans l'affichage graphique comme la SDL ou la GLUT.

Le support des images en noir et blanc est prévu à court terme en rajoutant des nouveaux scénarios dans la partie acquisition de la bibliothèque.

Tenir compte de l'évolution du bus 1394

L'implémentation de la technologie 1394 sous Linux est constamment en changement et nécessite par conséquent une attention particulière. De tels efforts seront fait pour qu'util_1394 reste cohérent vis à vis des bibliothèques qu'elle utilise.

Conclusion

Pendant le stage j'ai eu la mission de mettre en place un système d'acquisition d'images à partir de caméras sur bus 1394. 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 l'équipe MOVI spécialisée dans la recherche en vision et l'équipe robotique.

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 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 tel 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'I.U.T. La programmation en C et l'étude des réseaux 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 entraîner un petit mon anglais en allant à des conférences faites par des chercheurs anglophones et en côtoyant quelques stagiaires étrangers.

En fin de compte lors de ce stage de dix semaines j'ai pu découvrir un aspect du monde de l'entreprise fort enrichissant. Et dans ce cas précis j'ai eu un aperçu très positif de la manière dont se déroule un institut de recherche en informatique. Avec un accueil très agréable et une ambiance de travail très sympathique, j'ai eu un rare privilège en pouvant offrir mes services à l'équipe robotique et MOVI.

ANNEXES

Bibliographie

La technologie IEEE1394 :

http://iseea.online.fr/Rapports/a_jolly/html/bus/bus6.html
http://www.ti.com/sc/docs/products/msp/intrface/1394/tech.htm
http://www.multimania.com/maddog/HARD/HowFW.html
http://www.skipstone.com/compcon.html

Le projet IEEE1394 sous Linux :

http://linux1394.sourceforge.net/index.html
Installation de la cardbus : http://cs.jhu.edu/~burschka/downloads/firewire.html

La bibliothèque DC1394 :

http://sourceforge.net/projects/libdc1394/

Les formats YUV :

http://www.webartz.com/fourcc/fccyuv.htm

Le logiciel CORIANDER :

http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/coriander/

L'INIRA Rhônes Alpes :

http://www.inrialpes.fr/

Equipe robotique :

http://www.inrialpes.fr/iramr/

Equipe MOVI :

http://www.inrialpes.fr/movi/

Ouvrages :

La chaîne d'acquisition d'images.
Ecrit par Hervé MATHIEU
ISRN INRIA/RT—0246—FR

Documentation Util_1394

Bibliothèque d'acquisition d'images pour caméras IEEE1394

Ecrite en **C** pour **Linux 2.4**, surcouche des bibliothèques **dc1394 0.8.1** et **raw1394 0.8.2**
Licence: **GPL**
Rédacteur:

Rémi Fontan
remi.fontan@inrialpes.fr
Avril-Juin 2001

Sommaire

La bibliothèque "util_1394" a été écrite pour faciliter la capture d'images en provenance de caméras FireWire (ou IEEE1394). Elle propose un ensemble de fonctions spécialisées et faciles d'utilisations. C'est une surcouche de la bibliothèque "dc1394" qui au contraire propose un vaste ensemble de fonctions complexes servant à gérer des caméras numériques.

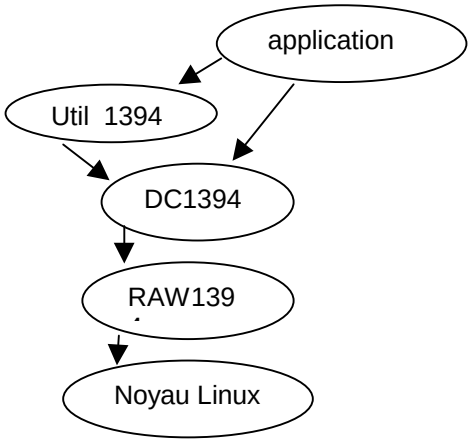
"util_1394" a été écrite à partir de la version 0.8.1 de la bibliothèque "dc1394" et de la version 0.8.2 de la bibliothèque "raw1394".

Les bibliothèques dc1394 et raw1394 ainsi que les dernières versions du noyau de Linux peuvent être télécharger ici:

- noyau Linux 2.4: http://kernel.org/pub/kernel/v2.4/
- dc1394 : http://sourceforge.net/projects/libdc1394
- raw1394 : http://linux1394.sourceforge.net./download.html

I Fonctionnement général de la bibliothèque

Organisation dans Linux:



La bibliothèque util_1394 est une surcouche de DC1394 mais elle ne la remplace pas complètement. Elle permet de simplifier quelques actions comme l'initialisation. DC1394 reste indispensable si l'on souhaite faire des réglages tel que la balance couleur ou la luminosité.

Structure principale:

La bibliothèque util_1394 s'utilise avant tout par l'intermédiaire de plusieurs fonctions dites de haut niveau. Ces fonctions permettent d'initialiser les différentes variables globales et locales ainsi que de faire des captures ou d'obtenir des informations concernant les caméras.

Ces fonctions se regroupent en quatre catégories.
- les fonctions d'initialisation.
- les fonctions de capture d'images.
- les fonctions pour obtenir des informations.
- une fonction dite de fermeture servant à libérer la mémoire des variables utilisées.

Il faut savoir que la plupart des fonctions renvoie un entier pour informer l'utilisateur du bon déroulement de la fonction. Si celle si renvoie 0 alors tout c'est bien passé, sinon dans la plupart des cas elle renvoie 1 s'il y a eu un problème. Certaines fonctions ont plusieurs code d'erreurs, il est nécessaire de regarder la description des ces fonctions.

Lors de l'écriture d'un programme, on retrouvera toujours une organisation de ce type:

- appelle d'une fonction d'initialisation.
- appelle de fonctions de capture ou d'obtention d'informations.
- appelle de la fonction de fermeture du programme

On retrouvera toujours la déclaration de ces trois variables en début de programme:

```
int *x;
int *y;
char **rgb;
```

Ces variables seront utilisées lors de la capture d'images et lors de la sauvegarde d'images dans des fichiers.

***x** et ***y** sont des tableaux d'entiers, ils contiendront les résolutions utilisées par chaque caméra connectée.

****rgb** est un tableau de buffeurs image. Il contiendra pour chaque caméra connectée un buffeur image.

De la mémoire leur sera alloué lors de l'appel d'une fonction d'initialisation et sera libéré lors de l'appel de la fonction de fermeture.

Les variables globales:

Plusieurs variables globales sont déclarées dans util_1394. Elles permettent de gérer la carte 1394 ainsi que les cameras connectées. Normalement si l'on utilise seulement les fonctions dites de haut niveau, il n'est pas nécessaire d'en tenir compte.

raw1394handle_t	**carte1394**;
nodeid_t *	**id_all_cameras**;
int	**nb_cameras**;
dc1394_camerainfo *	**camera**;
dc1394_miscinfo *	**camerainfo**;
dc1394_cameracapture *	**capture**;
int	**iso_channel**;

"**carte1394**" permet de créer un lien vers la carte 1394. La plupart des fonctions de "util_1394" l'utilisent.

"**id_all_cameras**" est un tableau comportant toutes les adresses ou "nodes" des caméras connectées. Un "node" permet d'identifier de manière unique un périphérique sur un bus IEEE1394.

"**nb_cameras**" est un entier indiquant le nombre de caméras connectées.

"***camera**" est un tableau comportant pour chaque caméra une structure de données indiquant des informations de bases sur la caméra. Le nom du vendeur et du modèle ainsi que le numéro de série y sont inscrit. Le "handle" et le "node" y sont aussi inscrit.

"***camerainfo**" est un tableau comportant pour chaque caméra des informations concernant son mode d'acquisition. La résolution, le format, le nombre d'images par seconde y sont inscrit.

"***capture**" est un tableau comportant pour chaque caméra une structure de données ou sont stockés le buffeur image ainsi que la résolution.

"**iso_channel**" est un compteur servant à affecter à chaque caméra un iso channel diffèrent.

Les fonctions d'initialisations:

int **util_1394_init_utilisateur**(int *x[], int *y[], char ***rgb);

int **util_1394_init**(int *x[], int *y[], char ***rgb,
int format, int mode, int framerate,
int iso_speed);

int **util_1394_init_fichier**(int *x[], int *y[], char ***rgb);

Chacune de ces fonctions ont un rôle similaire, tel que initialiser les variables et la carte, mais elles le font d'une façon différente.

"**util_1394_init_utilisateur**(...)" demande a l'utilisateur quels modes utiliser pour chaque caméra.

"**util_1394_init**(...)" initialise chaque caméra connecte avec les même modes. Cette fonction est à utiliser si l'on est au courant des modes supportés par les cameras.

"**util_1394_init_fichier**(...)" initialise chaque caméra á partir d'un fichier texte de configuration. Chaque caméra possède son propre fichier de configuration. Chaque fichier de configuration a comme nom : "camera-vendeur-model.config". Par exemple: "camera-SONY-DFW-VL500 v1.00 .config" C'est certainement la méthode la plus agréable à utiliser étant donné qu'il n'est pas nécessaire de recompiler le programme pour changer la configuration.

paramètres communs:

dans chacune de ces fonctions les paramètres **int *x[], int *y[], char ***rgb** sont en communs. Ces paramètres sont nécessaires pour stocker pour chaque caméra sa résolution utilisée et son buffeur image. Il n'est pas nécessaire d'allouer de la mémoire á l'avance pour ces paramètres, les fonctions d'initialisation s'en charge.

Valeurs renvoyées:

Chacune de ces fonctions renvoie un entier pour que l'on puisse savoir si l'exécution c'est bien passé.

-1 : problème d'ordre matériel, une camera a été déconnectée, la carte n'est pas reconnue, …

-2 : problème d'ordre logiciel, un mode non supporté a été affecté à une camera, …

1 : problème d'ordre logiciel, une transmission iso n'a pas démarré, …

0 : aucun problème, tout c'est bien passé.

Lorsque la valeur **1** est renvoyée, il est nécessaire d'appeler la fonction de fermeture pour libérer la mémoire allouée.

Les fonctions de capture d'images:

int **util_1394_get_image_camera**(char **RGB, int numero_camera);
int **util_1394_get_image_all_cameras**(char **RGB);

"**util_1394_get_image_camera**(...)" fait une capture d'une image à partir de la camera identifiée par le numéro "num". L'image est stocke sous forme de buffeur dans le tableau "RGB" a l'indice "num". Aucune allocation de mémoire n'est faite mais seulement des modifications.

"**util_1394_get_image_all_camera**(...)" fait une capture d'une image pour chaque caméra connectée. Toutes les images sont stockées dans le tableau "RGB".

Il est important de noter que ces fonctions sont bloquantes jusqu'à l'acquisition d'une image par caméra. Par contre si l'ordinateur n'est pas assez rapide ces fonctions, entre deux captures, peuvent sauter des images.

Les fonctions concernant l'obtention d'informations:

void **util_1394_print_info_camera**(int num);

void **util_1394_print_info_all_cameras**();

"util_1394_printf_info_camera(...)" inscrit dans un fichier texte le nom, le modèle ainsi que les modes supportes par la camera. Le fichier texte a comme nom :"camera-vendeur-model.info"

"util_1394_print_info_all_cameras(...)" appelle la fonction précédante pour chaque caméra connectée.

La fonction de fermeture (ou pour libérer la mémoire utilisée):

void **util_1394_close**(int x[], int y[], char **rgb);

Cette fonction libère la mémoire alloue lors de l'appel d'une fonction d'initialisation. Elle possède les mêmes paramètres que les fonctions d'initialisation mais au lieu de faire de l'allocation de mémoire, elle la libère.

II Structure interne

Description des fonctions secondaires

Fonctions secondaires servant á l'initialisation.

int util_1394_init_carte_et_variables(int *x[], int *y[], char ***rgb);
Initialise la carte1394 ainsi que les variables globales.
Valeurs retournées:

1	Problème avec l'initialisation d'au moins une caméra, nécessite l'appel de la fonction util_1394_close(…) pour libérer la mémoire utilisée.
0	Aucun problème
-1	Problème avec la carte FIREWIRE
-2	Problème avec les caméras

void util_1394_init_resolution_et_buffer(int x[], int y[], char **rgb, int num);
Initialise les tableaux de résolution et les buffers image en fonction des modes de chaque caméra.

int util_1394_init_camera_utilisateur(int num);
Initialise les modes d'une camera suivant les paramètres rentrés par l'utilisateur.

int util_1394_init_camera(int format, int mode, int framerate, int iso_speed, int num);
Initialise les modes d'une caméras en fonctions des modes donnés en paramètres. Chaque caméra utiliseras les même modes, il est donc nécessaire de faire attention a ce que les caméras supportent bien ces modes.

int util_1394_init_camera_fichier(int num);
Initialise les modes d'une caméra en fonctions d'un fichier texte intitulé:
"camera-vendeur-model.config"

int util_1394_init_iso_transmission(int num);
Initialise la transmission "iso" et alloue de la mémoire pour le buffeur image d'une caméra.

void util_1394_afficher_miscinfo(dc1394_miscinfo *miscinfo);
Affiche les caractéristiques d'une caméra contenue dans la variable "miscinfo". Cette fonction est un complément de la bibliothèque "dc1394". Elle est surtout destinée à être utilisé pour faire des tests.

int util_1394_saisir_miscinfo(dc1394_camerainfo camera, dc1394_miscinfo *miscinfo);
Saisie le format, le mode, le framerate et le débit pour une caméra donnée et le stock dans la variable "miscinfo".

Ce sont des fonctions intermédiaires utilisées par **util_1394_saisir_miscinfo(…)**.

int util_1394_saisir_format(dc1394_camerainfo camera, unsigned int *format);

int util_1394_saisir_mode(dc1394_camerainfo camera, unsigned int format, unsigned int *mode);

int util_1394_saisir_framerate(dc1394_camerainfo camera, unsigned int format, unsigned int mode, unsigned int *framerate);

void util_1394_defaut_miscinfo(dc1394_miscinfo *miscinfo);
Initialise par défaut la caméra en 320x240 YUV422 a 7,5imgs/s avec un débit de 400Mb/s. La plupart des caméras supportes ces modes, néanmoins il n'est pas impossible que le débit soit trop grand pour certain modèle de bas de gamme.

Fonctions de conversion du YUV en RGB.

int util_1394_capturetorgb(char *rgb, dc1394_cameracapture camera,int mode);
Cette fonction convertie le buffeur d'une caméra en un buffeur RGB. La structure de donnée d'un buffeur RGB est très simple. Elle consiste en un tableau d'octets(en général des caractères) ou les octets correspondent successivement à une couleur.
Exemple: Si je possède une image composée de trois pixel, mon tableau ressemblerai à cela:
[rouge, vert, bleu, rouge, vert, bleu, rouge, vert, bleu]

Attention, il est nécessaire que de la mémoire ait été alloué á la variable "rgb".

Ces fonctions servent à convertir un pixel codé en YUV en RGB.
inline void util_1394_yuv444torgb (char *YUV, char *RGB, int NumPixels);

inline void util_1394_yuv422torgb (char *YUV, char *RGB, int NumPixels);

inline void util_1394_yuv411torgb (char *YUV, char *RGB, int NumPixels);

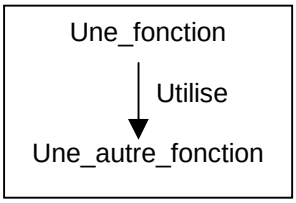
int util_1394_rgb2ppm(char *rgb, int largeur, int hauteur, char *nomfichier);
Sauvegarde un buffeur rgb dans un fichier au format ppm (raw).

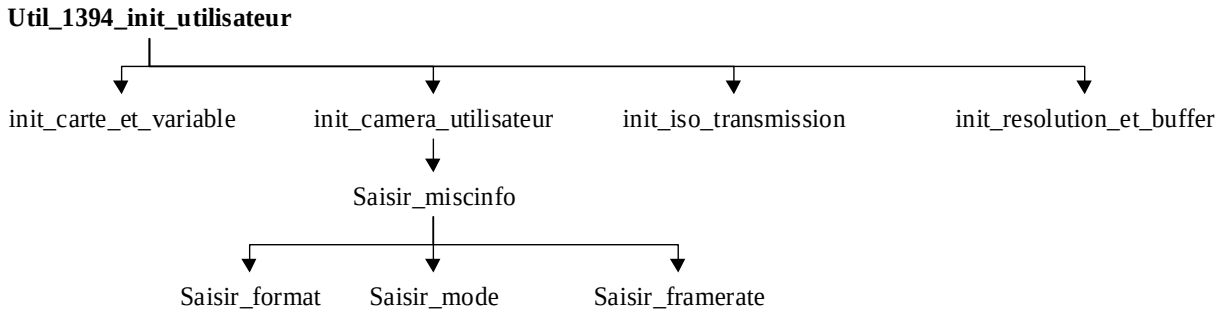
Arbres des fonctions

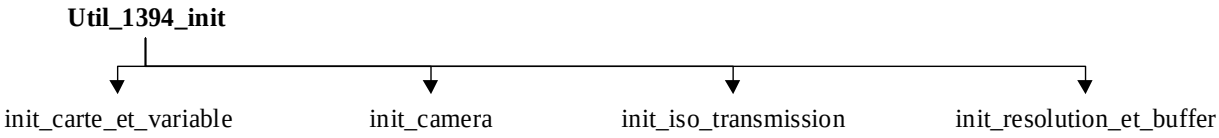
Ces arbres permettent de mieux visualiser les dépendances entre les fonctions de la bibliothèque util_1394. On part d'une fonction de haut niveau et on descend vers les fonctions de plus bas niveau.

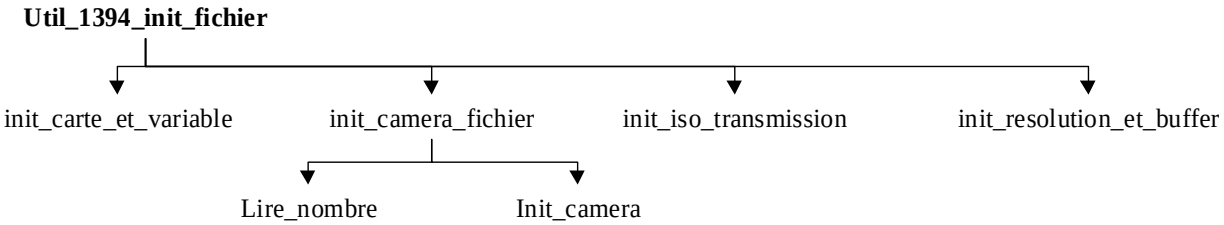
J'ai volontairement omis d'écrire à chaque fois "util_1394_"par soucis de lisibilité.

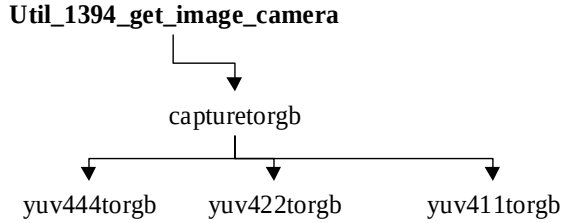
Sens de lecture:

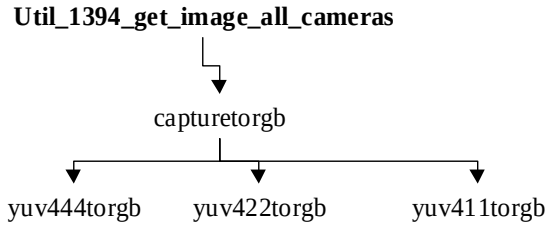


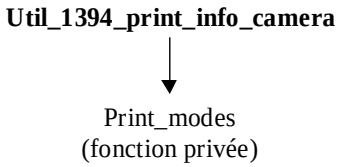


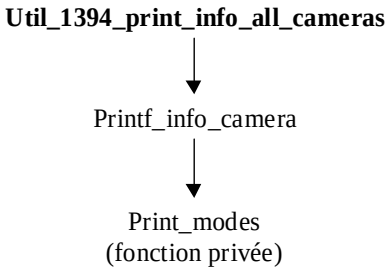







Util_1394_get_image_all_cameras
capturetorgb
yuv444torgb
yuv422torgb
yuv411torgb


Util_1394_print_info_camera
Print_modes
(fonction privée)


Util_1394_print_info_all_cameras
Printf_info_camera
Print_modes
(fonction privée)

III Bug liste

Problèmes connus:

Le problème le plus gênant est un blocage inexpliqué de la caméra lorsque l'on effectue pour la première fois une capture avec. C'est au niveau de la capture d'une image que le programme se bloque, on peut avoir l'impression qu'il manque une sorte initialisation.

Pour résoudre ce problème: Avec la caméra SONY DFW-VL500 il semble que en changeant légèrement le focus avant de lancer le programme peut éviter ce blocage. Sinon en débranchant la caméra et en la rebranchant on arrive au bout d'un certain temps à éviter le blocage.

Une fois qu'une capture a été faite avec une caméra, il n'y aura plus de problème tant que l'on ne débranche pas la caméra.

Dans certains cas, la caméra se bloque car l'ordinateur n'est pas assez puissant pour supporter le débit imposé par la caméra. En réessayant avec un framerate ou une résolution inférieure le problème se résoudra probablement.

Il est souvent difficile de discerner la différence entre les deux problèmes précédants, ainsi il vaut mieux essayer toutes les possibilités pour "résoudre" le problème.

Un autre problème étrange: Les captures réalisées avec la camera SONY DFW-VL500 avec une résolution de 640x480 sont de qualité médiocre. Je ne pense pas que mes algorithmes de conversion de YUV en RGB soient en cause.

Lorsque l'on utilise la camera PYRO WEBCAM-API-200, la première image est pratiquement toujours inutilisable. Il vaut mieux de préférence faire deux captures consécutives pour réserver la première image pour initialiser la camera et la deuxième pour l'utiliser.

Ce qui n'a pas été fait:

Cette bibliothèque n'est pas complète. Elle ne supporte pas tous les formats de caméra disponible mais seulement le format VGA non compressé. Etant donné qu'elle est basée sur l'utilisation de bibliothèques non finies, il faudra très probablement la mettre à jour.

La bibliothèque "DC1394" n'ayant aucune documentation et m'étant inspiré seulement des commentaires des ".h" pour en comprendre le fonctionnement, il est très probable qu'il reste des bugs.

Lors de l'acquisition d'une image, celle ci est automatiquement convertie au format RGB grâce à quelques fonctions logiciels. Néanmoins Xfree86 propose des fonctions de conversion de YUV en RGB accéléré matériellement si la carte vidéo

propose cette option. La plupart des cartes vidéo récentes proposant ce type de fonctions, il serait intéressant de les utiliser.

Le stockage d'images non compressées sur le disque dur pose comme problème la quantité de place disponible et la vitesse d'écriture sur le disque dur. Il pourrait être intéressant d'utiliser une carte de compression MPEG pour réduire la taille de la vidéo.

IV Exemple simplifié:

Cet exemple permet de capturer une image de la première caméra connectée et de sauvegarder cette image dans un fichier intitulé "mon_image.ppm". L'initialisation des caméras se fait par l'intermédiaire de fichier de configuration.

```
/*************************************************/
/*************************************************/
#include "util_1394.h"

int main(int argc, char **argv){

  char **rgb;  // tableau contenant les buffers de chaque camera
  int *x,*y;   // tableaux contenant les résolutions de chaque camera
  int err;     // variable servant á stocker les codes erreurs

  // initialisation des caméras
  err=util_1394_init_fichier(&x, &y, &rgb);
  // gestion des erreurs de l'initialisation
  switch(err){
  case -2:// problème matériel
  case -1:{
    // problème: soit il n'y a aucune camera connectée!, soit il y a
    // un problème d'initialisation
    return(1);
  }
  case 1:{
    // problème d'initialisation d'au moins une camera
    util_1394_close(x, y, rgb);
    return(1);
  }
  case 0:{
    // aucun problème
    break;
  }
  }
  // Cette fonction n'est pas indispensable, elle permet juste
  // d'activer le réglage automatique de l'iris et par conséquent
```

```
  // d'avoir une bonne lumière.
  // Attention toutes les cameras ne possèdent pas cette fonction.
  dc1394_auto_on_off(camera[0].handle, camera[0].id,
                             FEATURE_IRIS, 1);



  // on effectue deux captures pour être certain de ne pas avoir de problème
  err=util_1394_get_image_camera(rgb, 0);
  err=util_1394_get_image_camera(rgb, 0);
  if(err==0){
    // il n'y a eu aucun problème lors de la capture de l'image
    err=util_1394_rgb2ppm(rgb[0], x[0], y[0], "mon_image.ppm");
    if(err!=0){
      printf("problème lors de la sauvegarde de l'image\n");
    }
  }

  // libération de la mémoire
  util_1394_close(x, y, rgb);

  exit(0);
}


/*************************************************/
 *************************************************/
```

makefile:

```
###################################
# le nom de mon application:
APP = exemple

###############################

GCC = cc
LIB = -ldc1394_control -lraw1394
OBJ = $(APP).o util_1394.o

# les fichier util_1394.c et .h doivent être dans le répertoire
# ~/includes/

INC = -I$(HOME)/includes

$(APP):$(OBJ)
        $(GCC) -o $(APP) $(OBJ) $(LIB)

$(APP).o:$(APP).c
        $(GCC) -c -Wall $(INC) $(APP).c

util_1394.o:$(HOME)/includes/util_1394.c $(HOME)/includes/util_1394.h
```

```
	$(GCC) -c -Wall $(INC) $(HOME)/includes/util_1394.c

clean:
	rm *.o
	rm $(APP)

#################################
```

Résumé

Ce stage se déroulait au sein de l'INRIA Rhône–Alpes. Il consistait à mettre en place un système d'acquisition d'images à partir de caméras sur bus 1394. Dans un premier temps une petite étude a été effectuée sur la technologie 1394 puis la conception et la réalisation d'une bibliothèque ont été effectuées.

Cette bibliothèque doit permettre de simplifier la programmation d'application utilisant des caméras 1394. Les objectifs principaux étaient de faciliter la mise en œuvre des caméras, de pouvoir stocker les images sur le disque dur et d'aider le traitement des images en temps réels.

Cette bibliothèque fonctionne sous Linux, est écrite en C et est accompagnée d'une documentation aidant sa compréhension.

Abstract

I have done my training at the research institute INRIA Rhône–Alpes. The goal was to set-up picture acquisition system with digital cameras on a 1394 bus. At the beginning I have done a study about the IEEE-1394 bus norm. Then I developed and programmed a specific library.

This library is intended to assist the programming of software which uses digital cameras connected through the IEEE-1394 bus. The purposes were to provide functions facilitating the control of digital cameras, the storage of pictures on hard disk and for real time picture processing.

This library operates under Linux, is written in C and is provided with a complete documentation.