Conception d'un système d'acquisition de séquences d'images basé sur le Shape from Focus

Claude Vivien Toulouse

Rapport de stage

Du 11 mars au 11 aout 2013

Conception d'un système d'acquisition de séquences d'images basé sur le Shape from Focus

Auteur : Claude-Vivien TOULOUSE
Responsable : M. Frédéric COINTAULT
Encadrant : M. Bastien BILLIOT

Toujours par deux ils vont. Ni plus, ni moins.
Le maître et son apprenti.

Raisonnable parole d'un grand maître à la peau céladon dont l'anadiplose reflète la sagesse.

Remerciements

Je remercie tout d'abord Mr Frédéric COINTAULT, maître de conférences, pour m'avoir accueilli au sein d'Agrosup DIJON.

Je remercie également Nicéphore Cité pour les financements de mon stage.

Un grand merci Mr Bastien BILLIOT, actuellement en troisième année de thèse à Agrosup, pour son aide, ses conseils, son encadrement et sa disponibilité tout au long du stage.

Merci aussi à Thomas DECOURSELLE pour ses différents conseils et sa bonne humeur au quotidien.

Enfin je tiens à remercier globalement toutes les personnes présentes à Agrosup pour leur sympathie à mon égard et qui m'ont permis d'effectuer ce stage dans les meilleures conditions.

Sommaire

Table des figures

Introduction

La 3D est devenue quasi-omniprésente dans de nombreux domaines. Ainsi nous retrouvons cette technologie dans des téléviseurs, consoles de jeux, imprimantes, cinéma, … etc. On remarque que, en dehors des imprimantes 3D initialement utilisées pour du prototypage rapide et de plus en plus accessible au grand public, le terme 3D indique bien souvent un mode de perception tridimensionnel. La volonté d'obtention de l'information tridimensionnelle à l'image de la vision humaine n'est pas récente en vision par ordinateur.

Durant ces 30 dernières années, obtenir l'information tridimensionnelle est un axe de recherche vaste et retrouvé au travers de différents termes tels que acquisition 3D, reconstruction 3D ou encore numérisation 3D. Aujourd'hui, le besoin de recourir à ces méthodes est présent dans différents domaines comme la robotique, le contrôle industriel, l'imagerie médical, la reconstruction environnemental ou encore dans l'agriculture de précision. Bien souvent ce besoin se manifeste pour pallier au manque d'information induit par la 2D grâce à une interprétation complète de la scène.

Dans notre cas, nous utiliserons une méthode basée sur le Shape from Focus qui se base sur une information de netteté, technique qui sera détaillée dans une prochaine partie.

Ce rapport se découpe en 4 parties distinctes qui reprennent le déroulement du stage à savoir une étude sur le contexte du stage et pourquoi vouloir une imagerie de précision en agriculture avec quelques généralités d'optique. Par la suite, une partie abordera les problématiques liées au système existant. Une troisième partie fera découvrir les différentes solutions proposées ainsi que le déroulement d'une acquisition et enfin le quatrième point s'articulera au niveau des différents résultats obtenus.

I) Contexte

1) Imagerie appliquée à l'agriculture de précision

L'agriculture de précision se repose sur le principe de la gestion des parcelles agricoles en se basant sur des technologies de pointe telles que l'imagerie satellitaire ainsi qu'a plus courte portée directement sur les parcelles et le traitement des images et informations obtenues par l'informatique. Elle est souvent facilitée par des moyen de localisation dans la parcelle comme le système de positionnement par satellite de type GPS.

Son but principal est d'optimiser la gestion d'une parcelle d'un point de vue :

- agronomique : ajustement des pratiques culturales en fonction des besoins de la plante et du type de sol ;
- environnemental : réduction de l'empreinte de l'activité agricole tel que l'épandage d'engrais ;
- économique : augmentation de la compétitivité et réduction des couts en limitant les zones à enrichir par exemple.

Le système d'acquisition développé lors de ces quelques mois a pour but d'affiner encore plus toutes ces mesures car il pourra être utilisé afin d'avoir une représentation 3D précise et nette de la plante et ainsi pouvoir estimer sa croissance, sa surface de feuille ou encore pouvoir par exemple compter des grains précisément afin d'avoir un aperçu le plus proche possible du tonnage de la future récolte.

2) Shape from Focus

La méthode Shape from Focus (également appelée Depth from Focus) se base sur l'information de netteté. Cette technique de reconstruction 3D se décompose en plusieurs étapes comme représentées ci-dessous

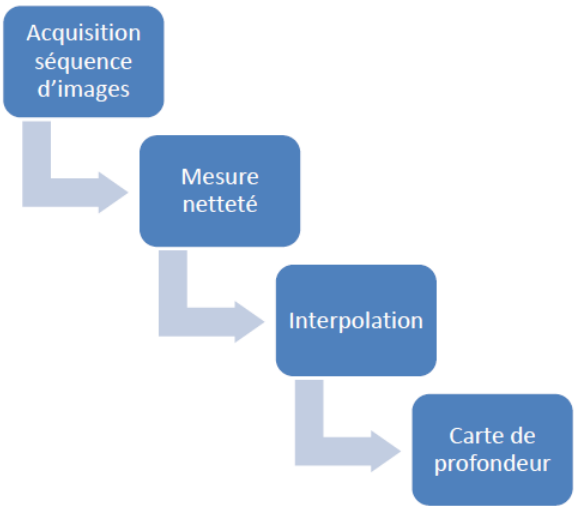

Figure 1 : Principe du Shape from Focus

La première étape est l'acquisition de la séquence d'images de la scène dont on souhaite effectuer la reconstruction 3D. L'idée générale de l'acquisition est de parcourir la scène en faisant un balayage complet par le plan focal. A chaque pixel correspond une image dans laquelle celui-ci est le plus net.

Une fois la séquence d'images obtenue, une évaluation de la netteté de chaque pixel est faite. Le même pixel de chaque image de la séquence doit représenter le même point spatial dans la scène. Une mesure de netteté locale est appliquée à chaque pixel et leur voisinage permettant de déterminer l'image dans laquelle chaque point de la scène est le plus net. On retrouve ici le principe de base de l'autofocus en photographie numérique qui consiste à maximiser une mesure de netteté afin de focaliser correctement sur l'objet visé.

Une valeur de netteté étant déterminé pour chaque pixel de chaque image, il est intéressant d'interpoler la courbe de netteté obtenue pour chaque point de la scène. En effet, en fonction de la précision du système d'acquisition utilisé, la position de la valeur maximum de netteté n'est pas nécessairement la position réelle du point de la scène. Comme le but est d'obtenir une estimation de la profondeur en tout point de la scène le plus précisément possible, il est nécessaire d'approximer la position du maxima de la courbe de netteté.

Enfin, la reconstruction 3D de la scène se déroule en plusieurs étapes en fonction des situations. L'étape commune à toutes les situations est la création de la carte de profondeur. Cette carte de profondeur est constituée des différentes positions spatiales des points de la scène correspondant aux positions approximées lors de l'étape précédente. Ces positions spatiales peuvent être représentées par des niveaux de gris pour une meilleure clarté avec comme légende : le blanc = pixel le plus loin et inversement, le noir = pixel le plus proche. Avec un calibrage du système, ces niveaux de gris renvoient directement à une position métrique de chaque point. Une représentation en fausses couleurs est également possible pour permettre une meilleure visualisation.

Si la situation n'exige que l'obtention de l'information de profondeur en tout point de la scène considérée, alors seule la carte de profondeur est nécessaire. Si une représentation 3D de la scène est souhaitée, la carte de profondeur est projetée sur trois axes et la texture de la scène est plaquée sur la surface projetée. Connaissant l'image où chacun des pixels sont nets, une image de la scène entièrement nette peut être construite et utilisée pour le plaquage de texture, ce qui permet une reconstruction très précise de la scène.

Cette technique de reconstruction 3D est habituellement utilisé dans le domaine de l'infiniment petit comme la microscopie confocale et qui peut être assimilée au Shape from Focus.

3) Généralités optiques

a) Profondeur de champ

La profondeur de champ est la notion sur laquelle il faut jouer pour obtenir une reconstruction 3D la plus précise possible par Shape from Focus. Contrairement au système sténopé, l'utilisation d'un système optique à base de lentille n'entrainera pas une projection nette d'un seul plan objet mais d'un ensemble de plans

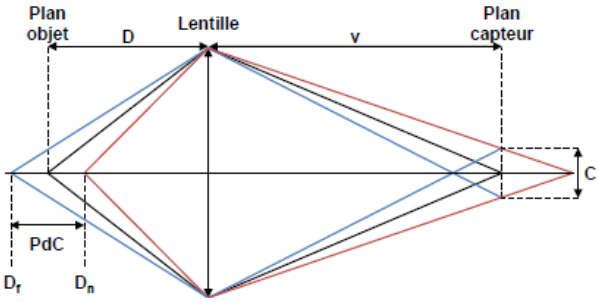

Figure 2 : Illustration de la profondeur de champ

La zone qui sera nette est une zone de la scène ayant une profondeur de champ égale à PdC (Figure 2 : Illustration de la profondeur de champ) et elle est calculée suivant ces équations :

$$\begin{cases} D_n = \frac{D}{1+C.A.\frac{D-f}{f^2}} \\ D_f = \frac{D}{1-C.A.\frac{D-f}{f^2}} \\ PdC = D_f - D_n \end{cases}$$

$$PdC = \frac{2.A.C.f^2.D.(D-f)}{f^4 - A^2.C^2.(D-f)^2}$$

$\boldsymbol{D}$ est la distance entre la scène et le système optique (mm) ;
$\boldsymbol{D_n}$ est la distance entre le premier plan net et l'objectif (mm) ;
$\boldsymbol{D_f}$ est la distance entre le dernier plan net et l'objectif (mm) ;
$\boldsymbol{C}$ est le diamètre du cercle de confusion (mm) ;
$\boldsymbol{A}$ est l'ouverture relative du diaphragme ;
$\boldsymbol{f}$ est la focale de l'objectif (mm).

La profondeur de champ est donc dépendante de quatre paramètres : la focale, l'ouverture du diaphragme, la distance d'acquisition et le diamètre du cercle de confusion.

b) Focale

La focale ou distance focale *f* de l'objectif utilisé est exprimée en millimètre. Ce paramètre est lié à la dimension de la scène à visualiser. En effet, à chaque distance focale correspond un angle de champ particulier qui sera grand pour une focale courte et petit pour une focale longue.

Une grande scène est acquise par un objectif de focale courte alors qu'une petite scène par un objectif de focale élevée. Le zoom n'est donc qu'une augmentation de la distance focale.

Plus la focale est courte et plus la quantité de lumière captée est importante du fait de l'élargissement du champ de vue (effet pris en compte lors de la conception des objectifs en diminuant le diamètre réel de l'ouverture proportionnellement à la focale). Un éclairement constant est obtenu avec une même valeur d'ouverture mais une focale différente. La profondeur de champ est affectée par une variation de la distance focale qui est le paramètre impactant le plus celle-ci : une petite augmentation de la focale entraine une grande diminution de la profondeur de champ.

c) Ouverture

Une ouverture relative notée *A* est égale à *A = f/D* avec f la focale et *D* le diamètre de l'objectif. L'ouverture varie selon une plage dépendante de l'objectif selon une suite géométrique de raison $\sqrt{2}$. Une faible ouverture correspond à une grande ouverture du diaphragme alors qu'une plus

grande valeur indique une ouverture plus faible où, pour chaque augmentation de *A*, deux fois moins de lumière traverse l'objectif. Plus l'ouverture relative *A* est faible et plus la profondeur de champ est réduite.

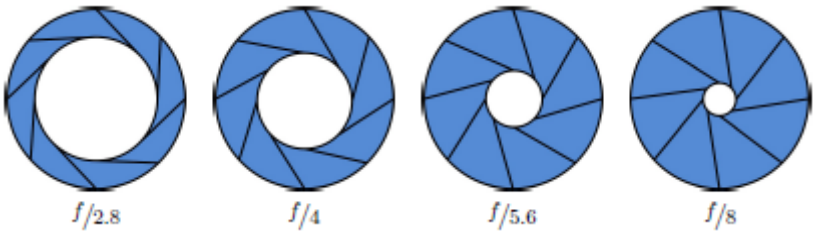

Figure 3 : Ouvertures relatives

d) Distance

La distance *D* est la distance entre objectif et le point de la scène considéré. On considère la distance entre l'objectif et le plan objet le plus net parmi les plans objets composant la profondeur de champ. Cette distance est liée à la valeur de réglage de la bague de mise au point de l'objectif : le focus.

Le réglage de cette distance de mise au point permet de faire coïncider le plan objet avec le point de l'espace où la netteté est la meilleure. En photographie, la mise au point peut être automatique et avoir un comportement similaire au phénomène d'accommodation de l'œil humain, c'est-à-dire que le plan objet net coïncide automatiquement sur le point de l'espace considéré. C'est l'autofocus.

Mécaniquement, une variation du focus est réalisée par une translation du réseau de lentilles constituant l'objectif afin de le rapprocher ou l'éloigner du capteur. Ainsi, plus la distance de mise au point est proche et plus la profondeur de champ est réduite.

II) Problématiques

1) Système existant

La solution de déplacement du système optique a été retenue afin d'obtenir la séquence d'image représentant un balayage de la scène par le plan objet.

L'ensemble du système d'acquisition peut-être découpé en deux parties : la partie optique permettant les acquisitions en tant que telles et la partie électronique permettant le déplacement de l'ensemble optique. L'utilisateur contrôle l'acquisition et le déplacement par le biais d'un ordinateur.

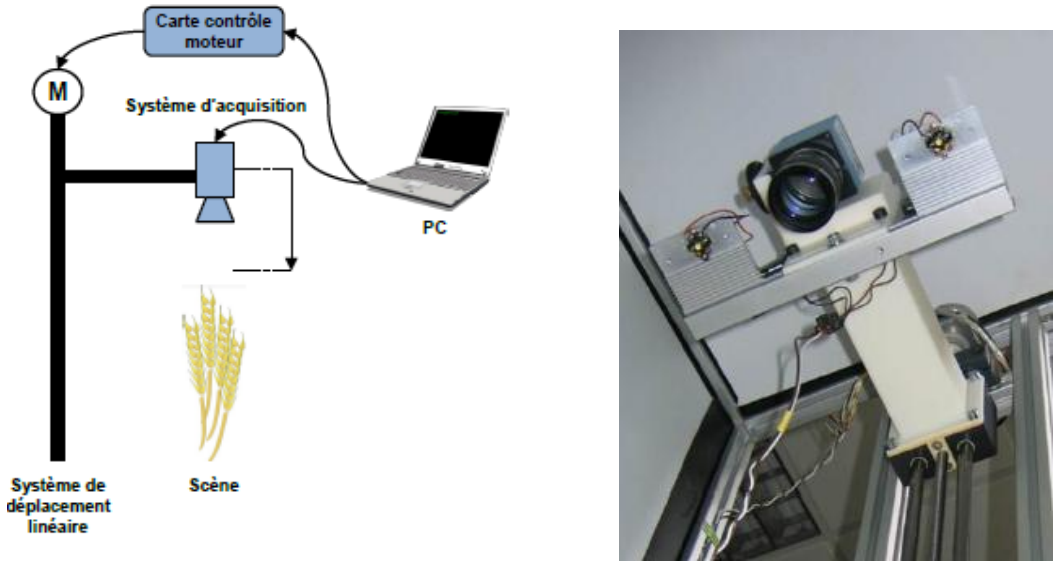

Figure 4 : Premier prototype

Cependant les vibrations provoquées par le déplacement de la partie optique entrainent un léger déplacement du centre optique entre chaque image de la séquence. Ceci est problématique dans le cadre de la reconstruction 3D par Shape from Focus car chaque pixel doit correspondre au même point de la scène dans chacune des images de la séquence. La séquence d'image nécessite donc un recalage afin d'obtenir la correspondance entre les pixels. Le recalage d'images est un problème récurrent en imagerie. Il consiste essentiellement à établir une relation géométrique entre les représentations de deux images permettant une mise en correspondance de celles-ci. Cette relation géométrique peut être de type rigide, dans le cas d'objets non déformables, ou non-rigide dans le cas contraire.

La transformation recherchée est celle pour laquelle la mesure de similarité est maximum. Ce type de méthode est souvent divisé en trois partie qui sont : le choix de la mesure de similarité à utiliser (somme des différences au carré, coefficient de corrélation, entropie mutuelle ...), le type de transformation recherchée (rigide, affine, projective ...) et une méthode d'optimisation assurant le lien entre les deux précédentes étapes.

Dans notre cas, chacune des images de la séquence met en évidence une zone nette différente du fait du balayage par la zone de profondeur de champ. Ainsi, les méthodes de recalage géométriques basées sur une mise en correspondance de points d'intérêt ne sont pas adaptées. Notre type de séquence d'images présente plusieurs similitudes avec des séquences d'images de type IRM. Ainsi, on retrouve des méthodes basées sur l'information mutuelle de souvent utilisée en imagerie médicale. L'intérêt principal de cette technique est sa rapidité et sa convergence robuste vers l'estimation de la transformation liant chaque image.

2) Cahier des charges

Ce premier système ayant permis la validation de la faisabilité de l'utilisation de la méthode de reconstruction 3D Shape from Focus, la conception d'un second système entrainant l'achat de matériel spécifique a été effectuée.

Afin de s'affranchir du déplacement de la partie optique entrainant une légère translation du centre optique, on s'est orienté vers le balayage de la scène par la variation de la distance de mise au point. Une variation de cette distance induit une modification de la profondeur de champ. Cependant, tant que la profondeur de champ est plus faible que la précision de l'estimation de la profondeur recherchée, ce type de déplacement est parfaitement utilisable.

Dans le cadre d'une reconstruction 3D d'une scène basée sur la méthode Shape from Focus, il est nécessaire de disposer d'une séquence d'image de la scène où un plan de netteté est en réalité une zone de la scène considérée comme nette correspondant à la profondeur de champ. La taille de cette zone varie en fonction de l'objectif utilisé (focale, ouverture), la caméra (taille des photosites) ainsi que la distance séparant la scène considérée et le système d'acquisition.

L'objectif du stage est l'implémentation et l'intégration de la commande de l'objectif motorisé au programme de contrôle de la caméra afin de réaliser un cycle d'acquisition automatique. Ce programme devra permettre l'acquisition d'une séquence d'images où le nombre d'image ainsi que le pas entre chaque acquisition sera choisi par l'utilisateur. Un calibrage du système sera donc à réaliser afin d'obtenir un pas constant du plan de netteté entre chaque acquisition. En effet un pas constant de variation de l'objectif motorisé n'entraine pas une variation constante du plan de netteté. Le contrôle de l'objectif motorisé passe par l'envoie de commandes à un boitier de contrôle connecté au PC par liaison série (RS232).

Le but final étant d'avoir un programme permettant de contrôler l'objectif motorisé ainsi que l'acquisition des images par la caméra.

III) Principe du prototype d'acquisition

1) Matériel à disposition

De façon à réaliser cette variation de la distance de mise au point précisément et surtout de façon répétable, l'utilisation d'un objectif motorisé est nécessaire. Cependant, un objectif avec une bague de réglage de la distance de mise au point motorisée n'est pas fréquent contrairement à ceux avec ouverture et focale motorisées. Le fabricant Qioptiq propose des objectifs conçu pour des capteurs à hautes résolutions ainsi qu'une version motorisée de ces objectifs.

Ainsi, nous avons opté pour un objectif de focale 35 mm et d'ouvertures relatives *f*1.6−16 (Qioptiq MeVis-Cm Figure 5 : Objectif motorisé).

Cet objectif monture C assure une correction linéaire de 450 nm à 900 nm et est compatible pour des capteurs de diagonale de un pouce au maximum. La monture C est un type de monture fréquemment utilisé au cinéma avant l'arrivée du numérique. Ils se vissent sur la caméra ont un tirage mécanique, c'est-à-dire une distance entre la bride et le capteur de 0,69 pouce soit 17,526 mm.

La variation de l'ouverture et de la mise au point est motorisée et la commande passe par l'utilisation d'un module de contrôle. Ce module de contrôle permet la commande de l'objectif par une liaison RS-232 avec l'ordinateur. Un jeu d'instruction permet la variation de la position de la bague de réglage de l'ouverture et de mise au point. Dans notre cas, la profondeur de champ souhaitée étant la plus faible possible, on veillera à toujours avoir le diaphragme ouvert au maximum (soit à l'ouverture relative *f*1.6).

Figure 5 : Objectif motorisé

Toujours dans l'optique d'une profondeur de champ réduite, le choix de la caméra (IDS UI-2280SE Figure 6 : Caméra IDS) s'est porté sur un capteur CCD couleur à filtre de Bayer 2/3 pouces (8.446×7.066 mm) de résolution maximale 2448 par 2048 (5 mégapixels) dont les photosites mesurent 3.45 µm². La taille de capteur supérieure comparée au premier système induit également une augmentation du champ de vue disponible de la scène considérée. La liaison avec l'ordinateur est assurée par connexion USB 2.0.

A une distance de mise au point de 1 mètre, cet ensemble objectif/caméra induit une profondeur de champ de 8.7 mm et un champ de vue de 25 par 19 cm. Un autre avantage de ce second système est le contrôle direct de la caméra et de l'objectif par l'ordinateur sans avoir de partie électronique conséquente. Ceci permet par ailleurs une réduction de l'encombrement général du système d'acquisition.

Figure 6 : Caméra IDS

2) Choix d'un langage

Lors du début du projet, aucune contrainte concernant la plate-forme de développement ou du langage n'était imposé. Un choix s'est vite établi pour un développement en C++ avec comme base la bibliothèque OpenCV et une interface graphique développée avec Qt afin de porter le projet sur différentes plates-formes, différents systèmes d'exploitation sans contraintes d'argent.

L'utilisation de Qt et OpenCV n'est pas open-source (qui permet de divulger des sources et les modifier) mais sous licence GNU LGPL (GNU Lesser General Public License) ce qui indique un logiciel libre mais qui oblige de citer les sources ainsi qu'une interdiction de les modifier tout comme les fonctions disponibles et offertes par les différentes bibliothèques.

Ayant déjà eu l'occasion de travailler depuis quelques années avec le langage C++ et étant revendicateur de la solution se rapprochant le plus de l'open source, la combinaison du C++ avec OpenCV et Qt était pour moi la meilleure possible. Voici en détail ces 3 outils avec qui le traitement d'image devient extrêmement intéressant :

- **Le C++**

C'est un langage de programmation qui permet de multiples paradigmes comme la programmation procédurale, orientée objet ou encore générique. Ce langage n'a aucun propriétaire comme le souhaitait son créateur Bjarne Stroustrup ce qui entraine une libre utilisation par quiconque souhaitant s'en servir, sans besoin d'autorisation ou d'obligation à payer une licence.

- **OpenCV**

OpenCV pour Open Computer Vision est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. NVidia développe des fonctions utilisant CUDA pour OpenCV. La bibliothèque OpenCV met à disposition de nombreuses fonctionnalités très diversifiées permettant de créer des programmes partant des données brutes pour aller jusqu'à la création d'interfaces graphiques basiques. Depuis la version 2.1, OpenCV a mis

l'accent sur les matrices et les opérations sur celles-ci pour en faire une alternative sérieuse à Matlab. En effet, la structure de base est la matrice. Une image peut être considérée comme une matrice de pixel. Ainsi, toutes les opérations de bases des matrices sont disponibles, notamment la transposée, le calcul du déterminant, l'inversion, la multiplication (par une matrice ou un scalaire) ou encore le calcul des valeurs propres. Elle propose la plupart des opérations classiques en traitements bas niveau des images comme la lecture, écriture et affichage d'une image, le calcul de l'histogramme des niveaux de gris ou d'histogrammes couleurs , le lissage ou filtrage, le seuillage d'image, la segmentation ou encore la morphologie mathématique.

- **Qt**

C'est une interface de programmation développé en C++ par Qt Development Frameworks et qui offre des composants pour créer des interfaces graphiques et faciliter des transmissions de données comme par exemple des connexions USB, port série, accès à des trames Ethernet,… Le principal avantage de Qt est qu'il est portable quelque soit l'environnement utilisé en ne recompilant que les sources, de plus il supporte le fait d'utiliser des bibliothèques logiciels réalisées dans un autre langage que celui dans lequel elle a été écrite (binding).

Comme dit précédemment, OpenCV est une bibliothèque orientée vers le traitement graphique donc optimisée et facile d'utilisation pour le traitement d'images malgré sa complexité apparente. En effet, une grande communauté ainsi qu'un handbook en ligne extrêmement bien ficelé et détaillé sont disponibles ce qui facilite son utilisation.

La partie la plus longue et la plus complexe fut de faire cohabiter OpenCV et Qt. En effet, malgré les apparences ce n'est pas un simple ajout de bibliothèques. Voulant utiliser les dernières versions disponibles afin de bénéficier des dernières options et des fonctions optimisées pour CUDA, il a fallu recompiler entièrement OpenCV avec les différentes options de compilation que l'on souhaitait bénéficier (CUDA, interface graphique, liaison avec Qt).

Une fois cela fait et n'ayant aucune documentation sur l'intégration entre les dernières versions de Qt en version 5.1 et OpenCV en version 2.4.5 (au début du stage, actuellement la version 2.4.6 est disponible et apporte de nouvelles fonctions pour le GPGPU), il aura fallu plusieurs jours pour comprendre comment remplir le fichier .pro qui est un fichier listant les fichiers du projet et permettant de configurer la compilation de l'application en fonction des bibliothèques statiques (.lib) ou bibliothèques dynamiques (.dll).

En utilisant la documentation en ligne des différentes fonctions disponibles et bénéficiant d'un moteur de recherche très complet, il fut aisé de commencer la programmation pour gérer l'affichage et la gestion d'images.

Une partie plus ardue fut la gestion de flux vidéo. Afin de se passer des API de la caméra pour pouvoir exploiter au maximum les capacités d'OpenCV et engendrer un code portable d'une caméra à l'autre, il fallu d'abord trouver les drivers compatibles ce qui s'avéra être des drivers béta particulièrement instables. N'ayant pas réussi à trouver la cause de cette instabilité surement due à la version de développement des drivers, l'utilisation des API de la caméra fourni par le constructeur fut obligatoire, s'éloignant ainsi de la liberté totale du programme. A partir de là, il fut très simple de pouvoir accéder au flux vidéo et de commencer à manipuler les matrices d'image. Ci-dessous un exemple affichant le flux vidéo :

```
#include <cv.h>
#include <highgui.h>
CvCapture* capture = NULL;
IplImage *frame = NULL;
IplImage *result = NULL;
CvFont font;
double hScale=0.5;
double vScale=0.5;
int lineWidth=2;
cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX, hScale, vScale, 0, lineWidth);
capture = cvCreateCameraCapture(index);
if (capture)
{
cvNamedWindow("Capture", CV_WINDOW_AUTOSIZE);
cvSetCaptureProperty(capture, CV_CAP_PROP_FPS, 30.0);
}
while(true)
{
frame = cvQueryFrame(capture);
if( !frame )
break;
result = cvCloneImage(frame);
cvPutText(result, text_fps, cvPoint(0,20), &font, cvScalar(255,255,0));
cvShowImage("Capture", result);
if(cvWaitKey(20) != -1)
break;}
```

De même pour le port série RS-232, de nombreux exemples étaient disponibles et les instructions à envoyer au contrôleur étant basique, cela ne fut pas problématique. Ci-dessous un exemple de classe gérant l'envoi et la réception de données à envoyer à l'objectif motorisé :

```
class RS232
{
public:
RS232();
virtual ~RS232();

bool open(int numPort, long speedInBaud);
bool open(int numPort, long speedInBaud,
int nbBit, int parity, float nbStopBit);
void closeCom();
bool setTimeOut(DWORD ms);
bool setSpeed(DWORD baudrate);

int sendData(DWORD lg, LPBYTE data);
int sendData(string* data);
int receiveData(DWORD lg, LPBYTE data);
int receiveData(string* data);

string getErrorMsg();
private:
HANDLE hcom;
_COMMTIMEOUTSct;
DCB          dcb;
int           bufferSize;
};
```

Il ne restait plus qu'à combiner les différents codes dans une interface graphique afin de faire bénéficier aux utilisateurs d'une facilité de prise en main et une assistance par rapport aux différentes actions à effectuer.

Pour cela, il fallait pouvoir visualiser à la fois la caméra pour pouvoir contrôler visuellement le bon déroulement de l'acquisition et aussi enregistrer des images, interagir sur les options de la

caméra via l'API du constructeur ainsi que d'envoyer des données au contrôleur de l'objectif motorisé pour faire varier la position du focus. Il a donc fallu avoir recours à des threads.
Pour définir un thread, nous avons besoin de définir ce qu'est un processus :

- Un processus est une tâche qui est en train de s'exécuter. Par exemple quand un programme se lance, le système d'exploitation crée un nouveau processus et celui-ci exécutera une suite d'instructions sur l'ordinateur (qui correspond au code d'un programme compilé). Un processus est défini comme étant un ensemble d'instructions à exécuter qui a besoin d'un espace mémoire pour les données de travail ainsi que d'autres ressources, comme des descripteurs de fichiers, des ports réseaux...

- Un même processus peut se décomposer en plusieurs parties, qui vont s'exécuter simultanément en partageant les mêmes données en mémoire. Ces parties se nomment des threads. Du point de vue de l'utilisateur, les threads semblent se dérouler en parallèle.

Pthread sera la bibliothèque retenu pour gérer les threads car malgré son apparence elle est très simple d'utilisation. Le terme Pthread est une abréviation de "POSIX Threads". POSIX est lui un acronyme de "Portable Operating System Interface for UniX". Nous avons donc choisi d'utiliser Pthread car c'est une très bonne bibliothèque stable et surtout portable entre différents système d'exploitation (Windows, Linux…) permettant de manipuler les threads, les processus voir les mutex assez facilement.

La mémoire de tous les threads d'un même processus peut être partagée ou non. Dans le cas de la mémoire partagée, cela signifie que si l'on crée dynamiquement une variable dans un thread, elle peut être lue dans un autre (c'est le cas des variables globales par exemple). Dans le cas de la mémoire privée, les variables crées dans un thread ne peuvent pas être lue dans un autre (c'est le cas des objets locaux par exemple). Il faut donc bien surveiller la mémoire qui est allouée dans un thread car elle n'est pas automatiquement libérée à la fin du thread. Il faut donc penser à libérer la mémoire qui a été allouée dynamiquement au cours du thread avant sa destruction, si besoin est. Il faut aussi veiller à ne pas dés-allouer la mémoire qui sera utilisée par un autre thread durant la suite de l'exécution programme.

Voici un exemple de l'utilisation de Pthread :

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS     2
void *PrintHello(void *threadid)
{
long tid;
tid = (long)threadid;
printf("Thread num #%ld!\n", tid);
pthread_exit(NULL);
}
int main (int argc, char *argv[])
{   pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t=0; t<NUM_THREADS; t++){
printf("Creating thread num %ld\n", t);
rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
if (rc){
printf("ERROR%d\n", rc);
exit(-1);
}    }
pthread_exit(NULL);}
```

Une solution d'acquisition semi automatique utilisant deux programmes distincts dont l'un permettait d'afficher et enregistrer les images en utilisant un thread et l'autre permettant de gérer l'objectif fut rapidement développée.

3) CUDA

Cherchant à avoir le maximum de performances sur du traitement d'images et bénéficiant d'une séquence d'images à analyser, il était forcé d'essayer au moins quelques codes basiques en utilisant la puissance des cartes graphiques : le GPGPU (General-purpose processing on graphics processing units). Pour cela, 3 choix étaient possibles :

- CUDA (Compute Unified Device Architecture) développé par Nvidia ;
- ATI Stream développé par AMD ;
- OpenCL (Open Computing Language) développé par Khronos Group.

Le choix de la technologie a testé s'est très vite tourné vers Nvidia et CUDA pour deux raisons très simples, les cartes graphiques à disposition étaient des Nvidia et la documentation était plus fournie que les ATI Stream et OpenCL. Il aurait été intéressant de faire des tests en OpenCL car cette technologie a la faculté de s'adapter au nombre de cœurs disponibles dans un ordinateur, et unifie les cœurs des processeurs aux unités de calculs des cartes graphiques pour améliorer ses performances d'où une portabilité sans aucune contrainte matériel. Malgré tout, la documentation est peu dense ou alors trop complexe pour des néophytes.

ATI Stream aurait pu être une bonne technologie, tout simplement car AMD produit les cartes graphiques les plus puissantes en termes de performance brute (ce sont des GPUs AMD qui sont utilisés pour du mining du fait de leur puissance équivalente à des ASIC entrée de gamme). L'avantage du GPGPU est de fournir une des performances nettement supérieures à celles des processeurs pour des prix dérisoires tout en permettant une parallélisation des calculs lourds. Ci-dessous un comparatif sur le débit binaire entre processeurs et cartes graphiques au cours du temps (Figure 7 : Comparatif CPU - GPU).

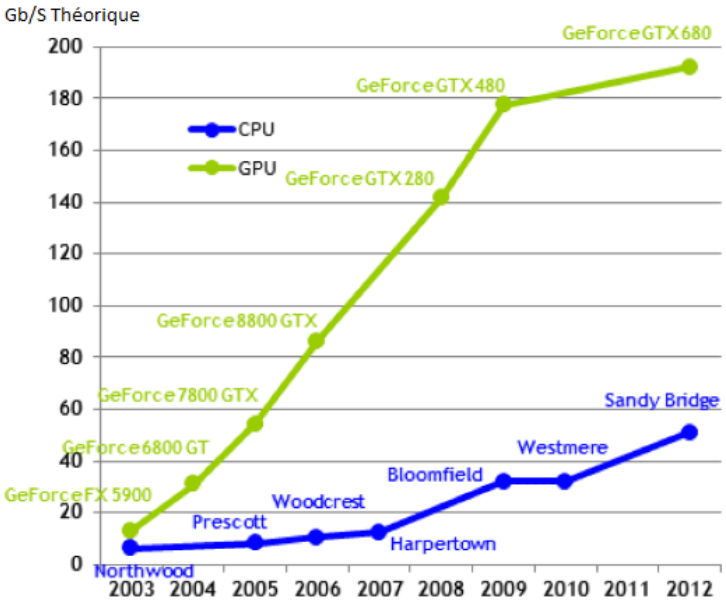

Figure 7 : Comparatif CPU - GPU

Maintenant voici un code simple pour entrevoir la puissance de calcul permise par l'utilisation de GPGPU. Le code dans les deux cas charge une image en niveau de gris, applique un filtre bilatéral pour lisser l'image en préservant les contours puis applique un filtre de Canny pour détecter les contours et sauvegarde l'image transformée :

- Code CPU

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main() {
Mat src = imread ("image.bmp", 0);
if (!src.data) exit(1);
Mat dst;
bilateralFilter(src,dst,-1, 50, 7);
Canny(dst,dst,35,200,3);
imwrite("canny.bmp",dst);
return 0;}
```

Le filtre bilatéral prend environ 1500ms pour s'exécuter sur l'image de définition 2000x2000. Le filtre de Canny prend quant à lui 20ms ce qui nous donne une durée d'environ 1700ms (une moyenne de 1691ms sur 10 exécutions du programme). Le temps d'accès au disque pour enregistrer étant compris entre 35 et 45ms.

- Code GPU

```
#include <opencv2/opencv.hpp>
#include <opencv2/gpu/gpu.hpp>
using namespace cv;
int main() {
Mat src = imread("image.bmp",0);
if (!src.data) exit(1);
gpu::GpuMatd_src(src);
gpu::GpuMat d_dst;
gpu::bilateralFilter(d_src,d_dst,-1,50,7);
gpu::Canny(d_dst,d_dst,35,200,3);
Mat dst(d_dst);
imwrite("canny.bmp",dst);
return 0;}
```

Le chargement de la même image depuis la ram vers la ram de la carte graphique prend 0,5ms. L'allocation temporaire de la mémoire est d'après la console de l'interface de développement de 0ms mais le temps ne peut-être nulle donc il sera considéré de 1 ms. Le filtre bilatéral est exécuté en 57ms et le filtre de Canny en 6.5ms. Le chargement de l'image vers la ram GPU vers la ram CPU prend environ 0,5ms. Le temps d'accès au disque est toujours compris entre 35 et 45ms. Le temps total est d'environ 100 ms (102 ms de moyenne sur 10 exécutions).

On peut donc voir qu'un code très simple prend 17 fois moins de temps en utilisant le GPGPU. Cela parait aisé d'utiliser les GPU pour faire du traitement d'images, mais seules quelques fonctions ont été transcrites, le reste doit se faire manuellement en allouant les zones mémoires et en faisant tout à la main en C, ce qui pour des personnes habituées à de la programmation plus haut niveau devient vite fastidieux et long et CUDA ne sera pas utilisé pour le reste du projet sauf en l'état de test, le passage des frames de la caméra vers la mémoire GPU en temps réel n'ayant pas réussi à être réalisé durant le laps de temps imparti.

4) Solution proposée

Un gros problème arriva lors de la création du logiciel d'acquisition automatique. Il fut impossible de gérer en même temps dans un même programme le port série, l'enregistrement des images, l'affichage de la caméra et les différentes actions de l'utilisateur. Un problème au niveau des threads et la gestion de l'interface produite en Qt en était la cause.

Après plusieurs jours passés, le temps passant rapidement et des résultats devant être rapidement produits, je pris la décision de développer un programme dans un autre langage, Matlab, pour avoir rapidement un programme d'acquisition automatique afin de répondre aux demandes et de pouvoir faire des acquisitions en mode automatiques pour ensuite exploiter les données.

Matlab est un langage matriciel assez simple à prendre à main, permettant d'obtenir des résultats beaucoup plus rapidement que dans les autres langages une fois habituée mais est aussi beaucoup plus lent pour du traitement de données importantes.

a) Calibrage

Le fait de faire varier plusieurs paramètres d'un objectif engendre automatiquement des aberrations ainsi qu'un grossissement. Parmi les aberrations, on retrouve deux grandes familles, les aberrations géométriques et chromatiques.

Les aberrations géométriques regroupent :

- les aberrations sphériques : provient des rayons au bord de l'objectif qui ne focalisent pas à la même distance que ceux du centre ;
- le coma : des points de la scène qui se focalisent en forme de comète ;
- l'astigmatisme : focalisation différente des rayons dans le plan sagittale ;
- la courbure de champ : projection d'un plan sous forme parabolique sur le capteur ;
- les distorsions : proviennent du mauvais alignement des lentilles.

Les aberrations chromatiques correspondent à une différence de focalisation des rayons lumineux à des longueurs d'ondes différentes.

L'objectif à disposition étant un objectif de précision prévu pour des mesures de précisions, différents aberrations sont éliminées. Une correction linéaire de 450 à 900nm corrige les aberrations chromatiques. La plupart des aberrations géométriques sont corrigées par la conception de l'objectif, mais il fallait le vérifier. Pour cela, nous avons utilisé un outil de calibrage développé par Bouguet sous Matlab.

Cet outil implémente une méthode de calibrage avec l'acquisition de plusieurs images d'une mire en faisant varier la position de celle-ci (Figure 8 : Calibration en utilisant des mires).

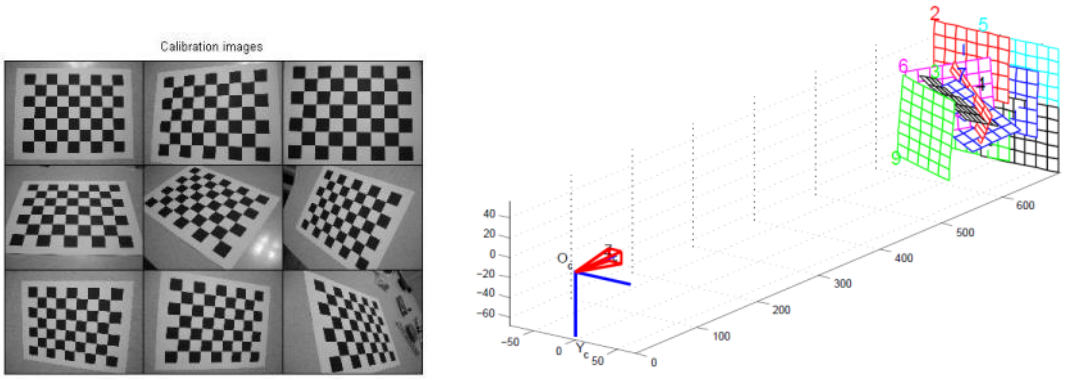

Figure 8 : Calibration en utilisant des mires

Les points d'intérêt des images sont détectés puis les paramètres intrinsèques (focale, centre optique) et extrinsèques (position repère monde de la caméra par rapport à la scène) sont déterminés. Les coefficients de distorsions radiales et tangentielles sont ensuite estimés avant une optimisation de l'estimation des différents paramètres par minimisation de la fonction de maximum de vraisemblance.

Le vecteur estimé contenant les coefficients de distorsions se présente sous la forme *kc* = [*k*1 *k*2 *p*1 *p*2 *k*3] avec *k*1, *k*2 et *k*3 les coefficients de distorsions radiales et *p*1, *p*2 les coefficients de distorsions tangentielles. Nous utiliserons un modèle du 4ème ordre donc seulement avec les coefficients k1 et k2 sans utiliser de distorsions radiales (corriger par notre objectif de précision).

Une fois le calibrage réalisé pour notre objectif, on obtient la modélisation des distorsions et on peut constater une très légère distorsion de 4 pixels en bordure d'image (50 mm et 35 mm motorisé), les modélisations des distorsions radiales sont obtenues On constate qu'une distorsion légère est présente sur les deux objectifs.

Ainsi, on a 9 pixels de décalages maximum en bordure d'image pour l'objectif 50 mm et 4 pixels pour le 35 mm (Figure 9 : Distorsions radiales). Le calibrage met en avant le fait que la focale de l'objectif 35 mm est en réalité de 37.5 mm.

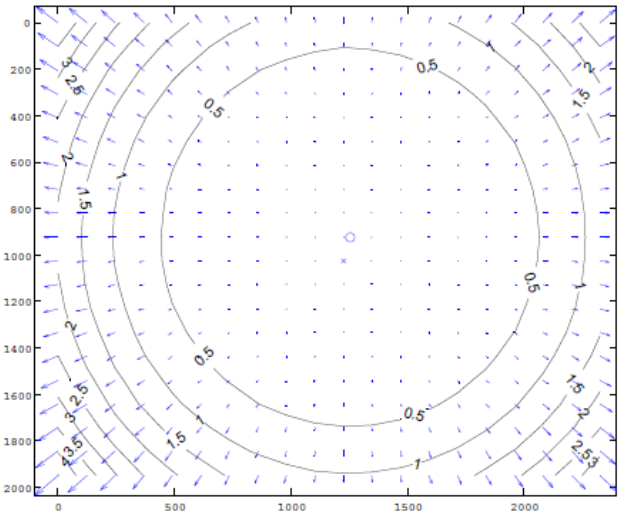

Figure 9 : Distorsions radiales

b) Correction du grossissement

La correction du grossissement est assez simple et passe par une méthode avec une interface pour redimensionner des images et l'adapter à la fenêtre de mesure. L'estimation du grossissement se fait grâce au déplacement connu entre les deux images et simplement avec le théorème de Thalès on peut obtenir un coefficient de grossissement (Figure 10 : Grossissement entre deux images).

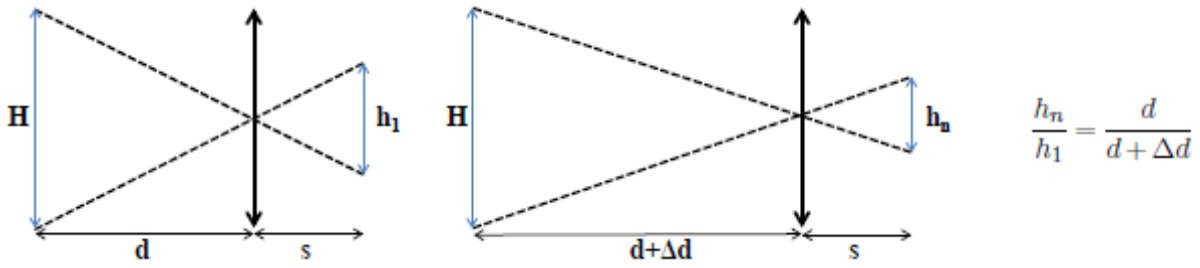

Figure 10 : Grossissement entre deux images

Ce coefficient est calculé et sert à rogner chaque image pour avoir toujours la même zone. Une mise à l'échelle de toutes les images est ensuite réalisée en fonction de la taille de l'image la plus éloignée du capteur. Ainsi, une sous-interpolation des autres images est appliquée. L'information est certes dégradée mais appliquer un redimensionnement dans ce sens ne génère pas d'informations erronées contrairement à une sur-interpolation. La résolution du capteur étant de base élevée, une réduction de la taille des images implique toujours une bonne résolution. On obtient de ce fait une séquence d'images représentant la même zone.

c) Etalonnage de l'objectif motorisé

L'objectif motorisé du second système nécessite un étalonnage afin de maitriser la commande. En effet, une rotation à intervalle constant de la bague de variation de distance de mise au point n'entraine pas une variation constante de cette distance dans l'espace. Cette variation doit être maitrisée afin de garantir une reconstruction 3D fidèle à la réalité. L'étape de mesure de netteté permet de déterminer la position de l'image où chaque pixel est le plus net et donc le pas séparant chaque image de la séquence doit être connu.

L'objectif motorisé permet une variation de la distance sur 96 positions de la distance de mise au point minimum (45 mm selon constructeur) à l'infini. Pour notre application, nous souhaitons balayer la scène considérée de la distance de mise au point la plus proche jusqu'à une distance de 1 mètre. Ainsi, pour chaque position du moteur, la distance de mise au point correspondante est déterminée et représentée ci-dessous (Figure 11 : Evolution la distance objet-> objectif en fonction de la position).

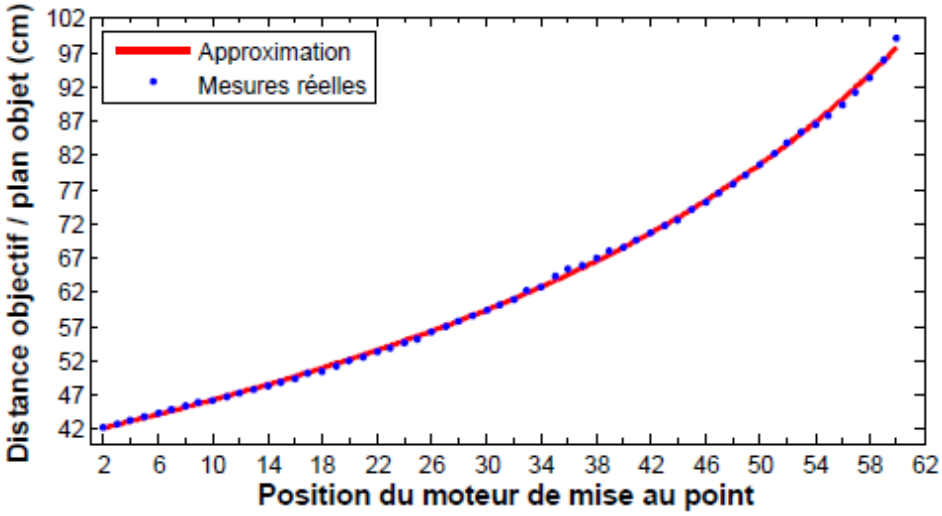

Figure 11 : Evolution la distance objet-> objectif en fonction de la position

Pour déterminer avec précision à quelle distance exacte correspond une position du moteur, un opérateur de mesure de netteté (la variation de l'amplitude du gradient : le Tenegrad Variance abrégé TENV) est utilisée afin de mesurer précisément cette distance (cf code Matlab ci-dessous pour le calcul du TENV). D'autres opérateurs existent mais le TENV est l'opérateur ayant le meilleur compromis rapidité/résultat, d'où son utilisation.

Le gradient est une grandeur vectorielle qui indique la variation d'une grandeur physique en fonction de différents paramètres. Pour le cas d'une image, c'est la variation d'intensité entre chaque pixel de l'image qui nous intéresse. Concrètement, un gradient dans une image indique la direction de la plus grande variation du champ scalaire et l'intensité de cette variation. On considère qu'un pixel correspond à un maximum local de la norme du gradient ce qui engendre le fait qu'une valeur non nulle indique une zone de transition entre une partie claire et une sombre de l'image (
Figure 12 : Lien avec le contour du gradient).

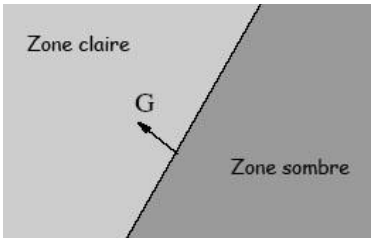

Figure 12 : Lien avec le contour du gradient

Les opérateurs de netteté se basent sur la quantification des composantes hautes fréquences d'une image comme les primitives (coins, arrêtes) d'où l'intérêt de se baser sur des techniques liés au gradient.

Pour cela, un logiciel a été développé en Matlab et a pour but d'enregistrer à chaque position de la potence, sur laquelle est positionné l'ensemble caméra/objectif, les différentes valeurs de l'opérateur de netteté. Les différentes valeurs, c'est-à-dire la position du focus pour laquelle les mesures étaient réalisées, ainsi que les valeurs de l'opérateur de netteté ainsi que les positions en

millimètre, étaient enregistrées dans des matrices afin de pouvoir chercher le maximum et faire des représentations sur des courbes plus parlantes et aussi pouvoir vérifier l'intégrité des données.

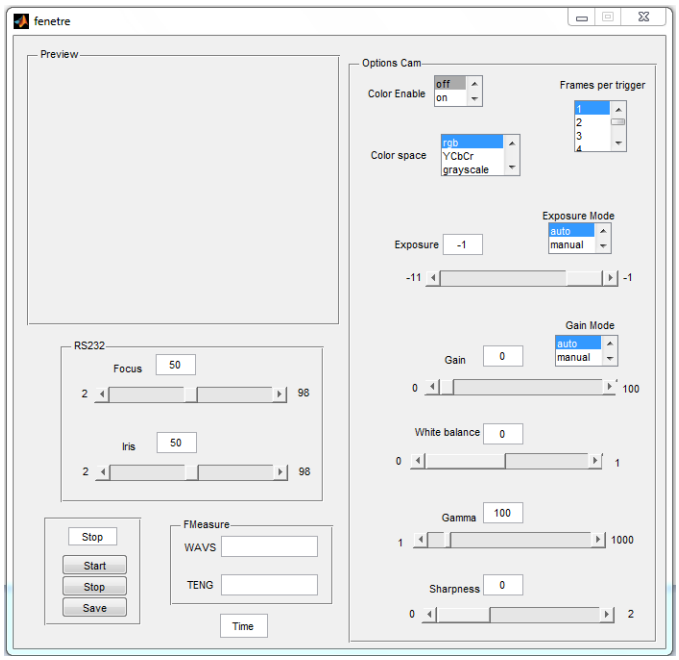

Figure 13 : Logiciel de calibration

Ainsi, pour chaque position du moteur du focus on obtient une courbe analogue (Figure 14 : Mesure de netteté pour la position initiale du moteur).

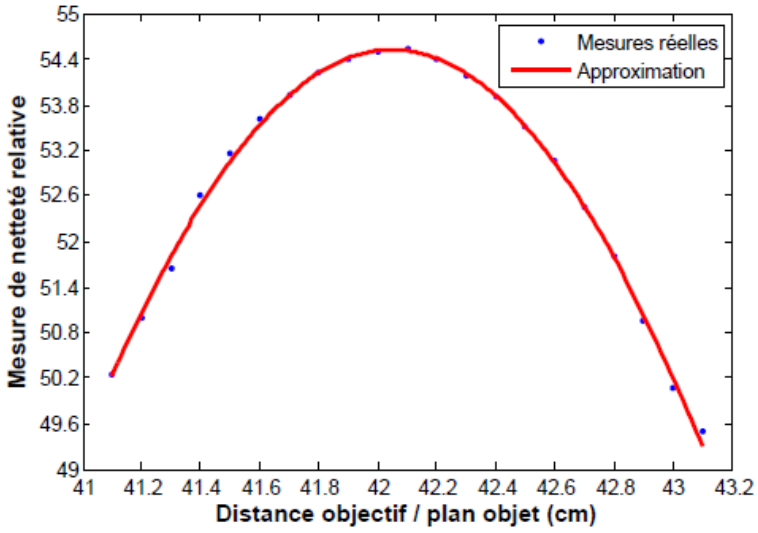

Figure 14 : Mesure de netteté pour la position initiale du moteur

On remarque une non-linéarité entre l'évolution de la distance de mise au point en fonction de la position du moteur, la courbe ayant une allure de gaussienne il est facile de déterminer à la fois visuellement et numériquement le maxima, donc la position à laquelle il faut placer l'objet pour avoir

le premier plan de netteté. De plus, après mesure, la distance minimum de mise au point est de 42.1 cm et non 45 cm.

Une fois l'étalonnage de l'objectif terminé, on peut calculer la profondeur de champ associée à chaque distance de mise au point (Figure 15 : Profondeur de champ en fonction de la distance de mise au point).

Avec un balayage de la scène inférieur à 1 mètre, la profondeur de champ n'excède pas 7,5 mm. Ceci permet une précision de l'estimation de la profondeur par la méthode Shape from Focus car cela garanti un non recouvrement des mêmes zones nettes pour deux images successives. En effet, l'espacement entre deux images est contrôlé de façon à être toujours supérieur à la profondeur de champ pour garantir une précision des mesures.

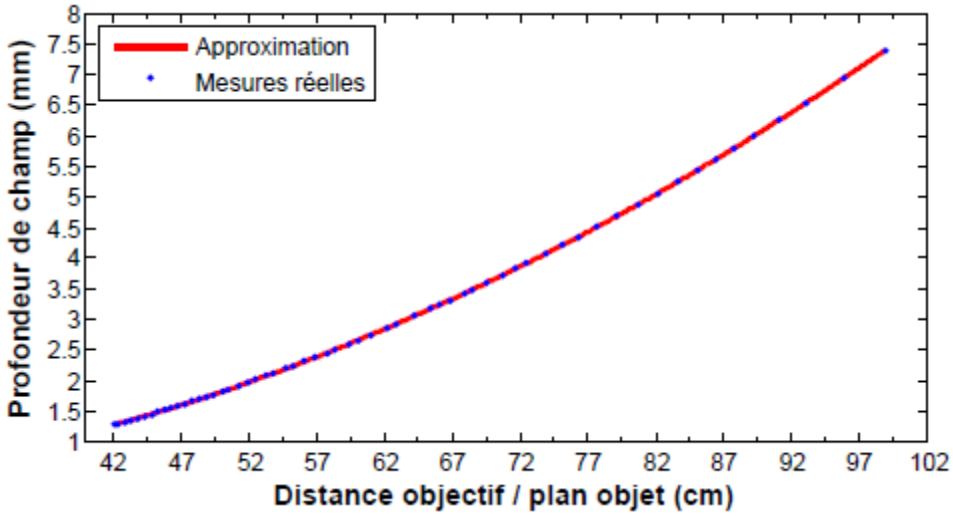

Figure 15 : Profondeur de champ en fonction de la distance de mise au point

Quant au champ de vue induit par un balayage de 42 cm jusqu'à 1 mètre, il varie de 9,5 par 8 cm à une distance de 42 cm jusqu'à 22,3 par 18,6 cm à 1 m de distance.

5) Acquisition

La procédure d'acquisition va dépendre principalement du type de scène à visualiser ainsi que de la précision de la reconstruction souhaitée (Figure 16 : Système d'acquisition final).

Figure 16 : Système d'acquisition final

Ainsi le logiciel développé (Figure 17 : Logiciel final du système d'acquisition) pour le contrôle du système permet de choisir le pas d'acquisition souhaité, la distance de départ du balayage ainsi que le nombre d'images à acquérir. Le choix du pas entre chaque acquisition va être choisi en fonction de la profondeur de champ induite par le système. En effet, pour ne pas avoir d'effet de chevauchement des zones dans le plan de netteté, le pas doit être choisi au minimum égal à la profondeur de champ.

Par exemple, pour une profondeur de champ de 1 cm, le déplacement minimal entre deux images sera de 1 cm. La distance de départ du balayage permet de placer le premier plan de balayage au début de la scène à reconstruire. Enfin, le nombre d'images à acquérir va évidemment dépendre de la profondeur de reconstruction souhaitée de la scène considérée. Une fois l'ensemble des paramètres choisi par l'utilisateur, l'acquisition est entièrement automatisée.

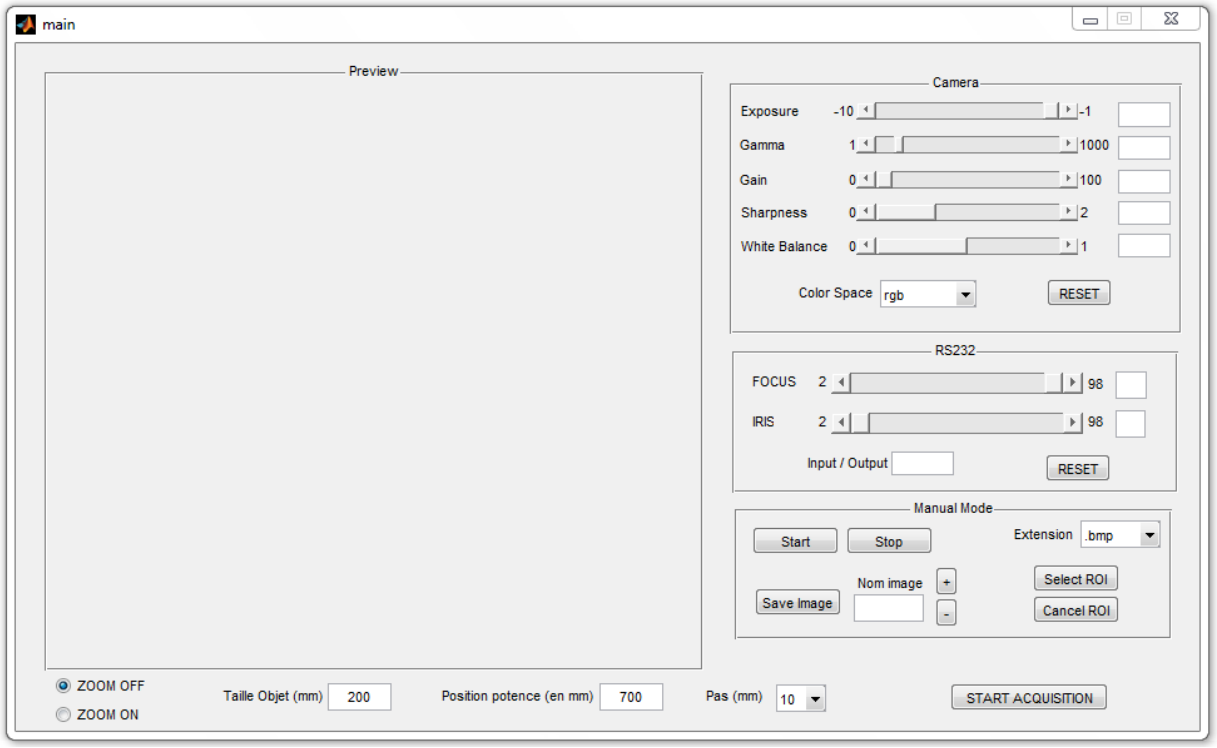

Figure 17 : Logiciel final du système d'acquisition

Le logiciel comprend plusieurs parties qui sont complémentaires les unes des autres. Elles sont au nombre de 4 et permettent d'affiner l'acquisition afin de correspondre au mieux à ce que cherche l'utilisateur. Les différentes parties sont détaillées ci-dessous.

a) Réglages de la caméra

La partie qui concerne les réglages de la caméra permet de gérer différentes options de la caméra comme on peut le voir ci-dessous (Figure 18 : Réglages de la caméra).

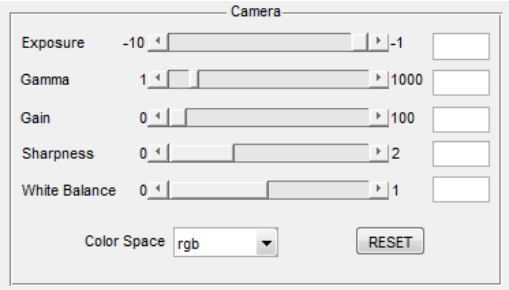

Figure 18 : Réglages de la caméra

On peut notamment y régler l'exposition, qui se peut se nommer aussi la lumination et qui permet de régler la quantité totale reçue par le capteur pendant la prise de vue.

Le gamma est aussi ajustable et il correspond à un facteur de contraste qui est fonction de la densité de l'image en fonction de la lumination reçue.

Le gain est par définition, le rapport de luminance lumineuse d'une image avec une direction de contemplation connue sur la luminance lumineuse de la surface complètement réfléchie (blanc de référence) pour une direction d'irradions fixée au préalable.

La netteté peut être réglée, elle permet d'augmenter le contraste que le long des bords de l'image tout en laissant des zones lisses de l'image.

On peut aussi effectuer une balance des blancs ce qui permet d'étalonner le capteur et ainsi de pouvoir corriger la dominante de couleur en fonction de l'éclairage ambiant.

b) Réglages de l'objectif

L'objectif motorisé se contrôle par port série par norme RS232 (Figure 19 : RS232 : prise femelle et mâle) et peut se faire manuellement par le biais de la partie RS232 du logiciel (Figure 20 : Réglages de l'objectif).

Figure 19 : RS232 : prise femelle et mâle

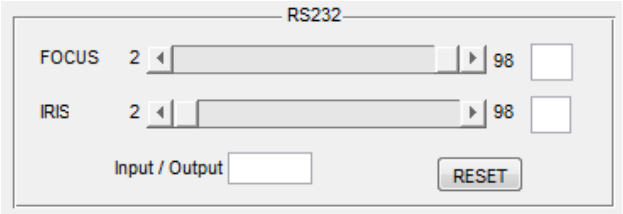

Figure 20 : Réglages de l'objectif

La liaison série RS 232 est utilisée dans tous les domaines de l'informatique (ex : port de communication com1 et com2 des PC, permettant la communication avec des périphériques tels que des modems, des scanners, ...), mais également dans le domaine de l'automatisme pour relier des capteurs aux automates, pour programmer un automate par un PC …

Elle est de type liaison série, permettant la communication entre deux systèmes numériques en limitant le nombre de fils de transmission par rapport à des liaisons parallèles. Elle est de type asynchrone, c'est à dire qu'il n'y a aucun transfert de signal horloge contrairement aux liaisons synchrones.

Ci-dessous se trouve le code Matlab pour envoyer des commandes via le port série à l'objectif motorisé. Les commandes sont assez basiques et en voici une liste non exhaustive :

- LCPF* → Focus à la position * → renvoi : LCNReady
- LCPI* → Iris à la position * → renvoi : LCNReady
- LCCD → Position du Focus → renvoi : LCPosF:*
- LCCDI → Position de l'Iris → renvoi : LCPosI:*

```
function rs232( iris,focus)
obj = instrfind('Type', 'serial', 'Port', 'COM1', 'Tag', '');
if isempty(obj)
    obj = serial('COM1');
else
    fclose(obj);
    obj = obj(1);
end
fopen(obj);
obj.Timeout = 0.1;
send_iris = strcat ('LCPI',char(iris));
query(obj, send_iris);
send_focus = strcat ('LCPF',char(focus));
query(obj, send_focus);
fclose(obj);
delete(obj);
end
```

On peut donc gérer la position de l'iris mais aussi la position du focus afin de s'adapter à la zone que l'on souhaite observé.

c) Mode manuel d'acquisition

Le logiciel dispose aussi d'un mode manuel d'acquisition qui permet de choisir l'extension et de sauvegarder une seule ou plusieurs images. L'utilisateur peut dans ce mode, utilisé les différents réglages précédents, qui sont le réglage de la caméra ainsi que le réglage de l'objectif motorisé.

L'affichage de la caméra se fait dans la zone de prévisualisation dédiée à cet effet (Figure 17 : Logiciel final du système d'acquisition) et elle peut être démarré et arrêté en appuyant sur les boutons dédiés à cet utilisation (cf ci-dessous le code basique d'une gestion de la caméra sous Matlab).

Les images peuvent être sauvegardées en choisissant leur extension et une zone d'intérêt peut être choisie en utilisant les fonctions de sélection de ROI pour ne s'intéresser qu'a une seule partie de l'image.

Lors de la sauvegarde, un dossier est créé et il contient à l'intérieur un fichier d'information où est contenue l'heure de l'acquisition manuel ainsi que les différents réglages de la caméra et du port série. Chaque image est numérotée automatiquement à chaque enregistrement si aucun nom n'est saisi et le fichier d'information permet de relier l'image à la position du focus afin de pouvoir, si l'utilisateur le souhaite ce resservir des données dans un futur proche.

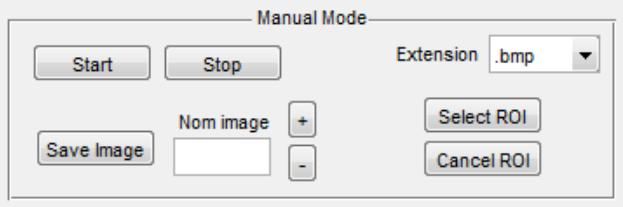

Figure 21 : Mode manuel d'acquisition

```
global src;
global vid;
global hImage;
vid.FramesPerTrigger = 1;
triggerconfig(vid, 'manual');
vidRes = get(vid, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid, 'NumberOfBands');
hImage = image(zeros(imHeight, imWidth, nBands));
preview(vid, hImage);
stoppreview(vid);
```

d) Mode d'acquisition automatique

Le mode le plus utile et qui a été développé en priorité est le mode d'acquisition automatique (Figure 17 : Logiciel final du système d'acquisition).

Une fois la position de la potence choisie ainsi que la taille de l'objet, l'extension des images et le pas, le système va automatiquement choisir la position du focus pour ensuite faire la série d'acquisition.

Les différentes calibrations ont permis de nous apporter une matrice de correspondance qui nous donne la position du focus de l'objectif motorisé à atteindre afin d'avoir le plan de netteté correspondant à la hauteur de la scène et d'acquérir une séquence complète de la scène.

Lors du lancement de ce mode d'acquisition automatique, tout comme le mode manuel d'acquisition, un répertoire est créé et il contiendra le fichier de configuration avec l'heure, les réglages de la caméra et des différentes options que le logiciel propose ainsi qu'a chaque ligne, le numéro des images ainsi que la position du focus afin de pouvoir se resservir de ses informations lors de la reconstruction 3D de la scène. Les résultats de ces différentes acquisitions seront présentés dans la dernière partie de ce manuscrit.

IV) Résultats

Pour finaliser ce rapport, voici deux séquences qui ont été faites sur 2 plantes différentes, afin de tester les possibilités de ce prototype dans le domaine agronomique. Le blé ainsi que la plante de l'arbuste Millepertuis Hidcote présente à proximité d'Agrosup.

Voici les premières images avec le blé (Figure 22 : Acquisition d'épis de blés):

Figure 22 : Acquisition d'épis de blés

De gauche à droite on a la première image de l'acquisition, la 11ème et la dernière image. On voit que sur chaque image, les éléments nets de la scène ne sont pas la même.

Une fois les images recalés, et en utilisant le meilleur opérateur de netteté, on obtient une image ou les éléments sont complètement nets ainsi qu'une carte de profondeur ou les éléments noirs sont les plus près de nous et les éléments blancs les plus éloignés.

Figure 23 : Image reconstruite

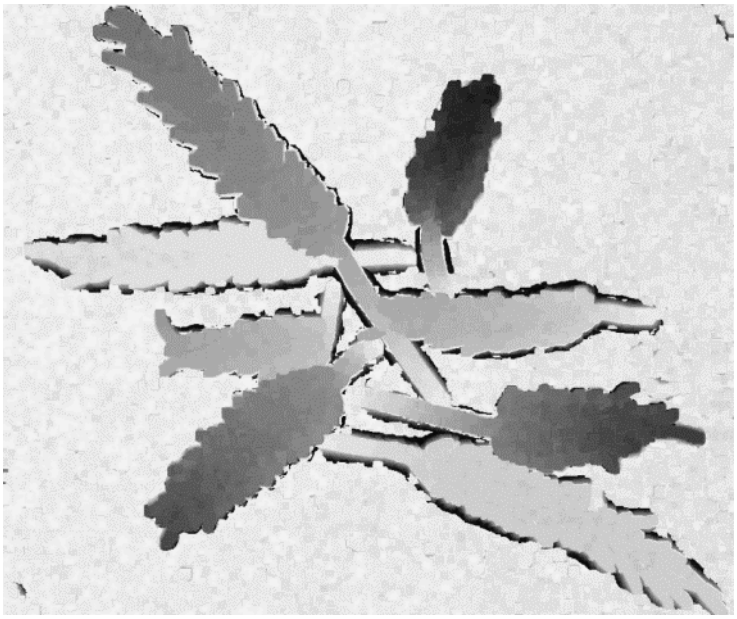

Figure 24 : Carte de profondeur des épis de blés

Cette carte de profondeur peut-être projeté en 3D et on peut appliquer dessus la texture de l'image complètement nette pour avoir la représentation 3D de la scène (Figure 25 : Représentation 3D de la scène d'épis de blés).

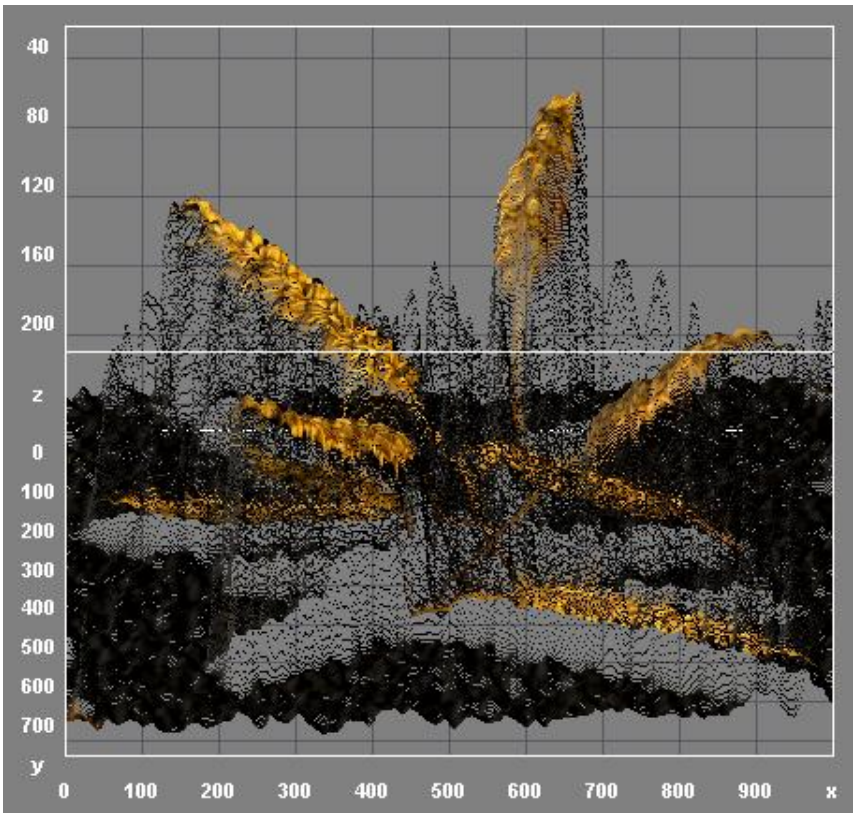

Figure 25 : Représentation 3D de la scène d'épis de blés

De même, voici les résultats sur une fleur de Millepertuis Hidcote :

Figure 26 : Acquisition d'une fleur de Millepertuis Hidcote

Figure 27 : Carte de profondeur

Figure 28 : Image reconstruite

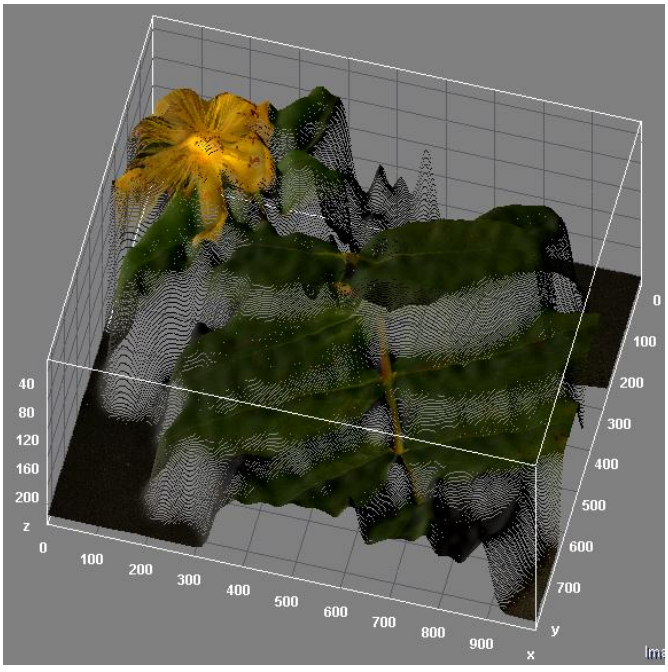

Figure 29 : Reconstruction 3D avec texture

Conclusion

Durant ce manuscrit, nous avons pu voir qu'il fallait un système spécifique pour pouvoir acquérir une séquence d'images nécessaire à la reconstruction 3D par la méthode de Shape from Focus.

Le logiciel permettant de faire l'acquisition ainsi que les différents traitements effectués sur la séquence d'images acquises est la clé pour permettre une reconstruction fiable et robuste.

L'automatisation de toutes ces tâches a permis d'avoir une utilisation aisée par des personnes non spécialistes du domaine grâce à l'intuitivité et la facilité de la prise en main du logiciel développé.

Le prototype final à la possibilité de s'affranchir de différentes erreurs d'alignement et grâce à son encombrement réduit, il peut être intégré dans différents appareils mobiles ou des espaces confinés.

Malgré cela, le prototype reste un prototype et une version finale aurait pu être développé en combinant les deux systèmes en C++. De plus il aurait été très intéressant de pouvoir bénéficier de la technologie CUDA au sein du code pour accélérer et threader les tâches à effectuer pour avoir un rendu le plus rapide possible et se rapprocher de l'ambitieux objectif du temps réel.

De plus, on peut penser que l'avenir du traitement d'image étant la parallélisation des données à traiter en utilisant le GPGPU et malgré tout les progrès fait au niveau des algorithmes, filtres et autres méthodes pour transformer, analyser ou recaler les images, il reste une grosse partie d'optimisation des systèmes traitants ou analysants des flux d'images à optimiser.

Références

[Bouguet, 2010] Bouguet, J. (2010). Camera calibration toolbox.

[Minhas et al., 2009] Minhas, R., Mohammed, A. A., Wu, Q. J., and Sid-Ahmed, M. A. (2009). 3d shape from focus and depth map computation using steerable filters. In *Image Analysis and Recognition*, pages 573–583. Springer

[Nayar, 1989] Nayar, S. K. (1989). Shape from focus. Technical report.

[Pentland, 1987] Pentland, A. (1987). A new sense for depth of field. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4) :523–531.

[Pertuz et al., 2012] Pertuz, S., Puig, D., and Angel Garcia, M. (2012). Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*.

[Rovira-Más et al., 2008] Rovira-Más, F., Zhang, Q., and Reid, J. F. (2008).
Stereo vision three-dimensional terrain maps for precision agriculture. *Computers and Electronics in Agriculture*, 60(2) :133–143.

[Zitova and Flusser, 2003] Zitova, B. and Flusser, J. (2003). Image registration methods : a survey. *Image and vision computing*, 21(11) :977–1000.