Glossaire

[1] Système temps réel embarqué : C'est tout d'abord un système dont le comportement dépend, non seulement de l'exactitude des traitements effectués, mais également du temps où les résultats de ces traitements sont produits. Il est dit « temps réel » parce que ses actions logicielles dirigent les activités d'un processus en cours d'exécution. Si de plus, il est physiquement intégré à l'environnement qu'il contrôle et qu'il est soumis aux mêmes contraintes physiques (température, pression, etc.) que son environnement, il est alors dit « embarqué ».

Sommaire

Introduction générale

L'évolution croissante des exigences en termes de fonctionnalités des systèmes embarqués se traduit, aujourd'hui, par une augmentation significative de la complexité de leur cycle de développement. Les systèmes réactifs temps-réel, largement utilisés dans les domaines de sécurité tels que la vidéosurveillance, la stéréoscopie est une parfaite illustration de cette tendance.
Nos jours, les systèmes embarqués sont de plus en plus présents comme constituants de produits industriels ou commerciaux. Que ce soit le four micro-onde, le contrôleur d'injection et d'allumage d'une voiture, le robot industriel, le téléphone intelligent, le système de pilotage automatique d'un avion, tous ces appareils ou systèmes ont au moins un point en commun, ils sont dédiés à des tâches spécifiques. Le marché des systèmes embarqués est en pleine croissance. Un tel système embarqué comprend deux parties essentielles : la partie matérielle et la partie logicielle.
Il est évident que les FPGA sont aujourd'hui utilisés dans un large éventail d'applications. Or, à mesure que le catalogue d'IP disponibles continue de croître, les FPGA ressemblent de plus en plus à des systèmes sur puce.
La possibilité d'obtenir des informations tridimensionnelles dans l'ensemble des directions présente beaucoup d'intérêt en stéréovision, qu'il s'agisse de construire un modèle de l'environnement, de localiser la camera dans cet environnement ou bien d'asservir ses déplacements. Dans un tel contexte, il est indispensable d'obtenir les données tridimensionnelles en un temps minimum.

Il est question dans ce projet, de dimensionner un système embarqué pour l'optimisation des capacités de stockage du système de vidéosurveillance, ainsi d'étudier les algorithmes de la reconstitution 3D, ensuite de l'implémenter sur une architecture reconfigurable telle que les FPGA (Field Programmable Gate Array).

Le premier chapitre portera sur la présentation du laboratoire d'accueil, et ses différents axes de recherche et de développement, ainsi il présente le contexte général dans lequel les travaux décrits dans ce manuscrit ont été menés.

Dans le second chapitre, présente des primitives nécessaires à la compréhension de la vision stéréoscopique.

Dans le troisième chapitre, donnera les différentes démarches pour la conception et la réalisation de l'application.

Le quatrième chapitre portera sur les procédures suivies pour l'implémentation de l'algorithme sur la carte NanoBoard3000.

Chapitre I
Présentation du lieu de stage et contexte du projet

I. Présentation du laboratoire d'accueil

Notre travail a été réalisé au sein du Laboratoire Signaux Systèmes et Composants (L.S.S.C) du département génie électrique créé en 1996 à la Faculté des Sciences et Techniques de Fès**.** L'équipe du laboratoire est composée de 14 Professeurs de l'enseignement supérieur, de 2 Professeurs habilités, de 3 Professeurs assistant et 20 Doctorants**.** Les activités du laboratoire se divisent en quatre axes comme le montre l'organigramme ci-dessous :

Laboratoire signaux et systèmes et composants
Directeur : Mr. **El-Hossein ABARKAN**

Signaux Systèmes et Télécommunications	Composants et Microélectronique	Optélécoms et Imagerie	Compatibilité électromagnétiqu
Responsable de l'équipe : **Pr. Mohcine ZOUAK**	Responsable de l'équipe : **Pr. Ali AHAITOUF**	Responsable de l'équipe : **Pr. Farid ABDI**	Responsable de l'équ : **Pr. N. EL OUAZZA**

I.1: bora stèr

xes velop

Ces équipes développent des recherches fondamentales et appliquées. Elles sont dans la plupart des cas réalisées dans le cadre de projets scientifiques financés par l'état ou par d'autres organismes. Pour plus de détail sur les activités de ce laboratoire, les intéressées peuvent aller sur le site de la FST www.fst-usmba.ac.ma.

II. Présentation du sujet et problématique

II-1. Contexte global du projet

Ce stage a été effectué au sein de l'équipe Composants et Microélectroniques du laboratoire signaux systèmes et composants de la faculté des sciences et techniques. Dans le cadre de projet de fin d'étude, conception et réalisation d'un système embarqué autour de FPGA d'une procédure de mise en correspondance stéréoscopique, en utilisant un algorithme à base de l'évaluation d'une fonction de corrélation. Ce système repose sur la gestion de l'affichage sur un écran VGA, des images importées. Afin de manipuler un processeur embarqué, et des drivers de l'afficheur VGA par la programmation C/VHDL, ensuite la validation de ces drivers sur la carte d'Altium.

II-2. Interprétation de la problématique

Dans de nombreux domaines, l'information apportée par l'imagerie classique est insuffisante. Par exemple, avec une simple camera, l'image apporte des informations sur la couleur de la scène et sur la forme des obstacles existants mais très peu sur leurs dimensions..
La modélisation de scènes 3D à partir de séquences d'images 2D a depuis longtemps été abordée comme un sujet de recherche [1]. Le but de cette modélisation est d'extraire une description compacte de la scène à des fins de reconstruction [2], de la reconnaissance [3], ou de la compression de données [4, 5].
L'algorithme de reconstruction 3D ou de stéréoscopie doit résoudre donc deux problèmes fondamentaux: la correspondance, qui traite de la recherche d'un objet une image de gauche par exemple, qui correspond à un objet dans une image de droite, et la reconstruction, qui cherche à trouver la profondeur et la structure du point d'intérêt correspondant.
Le problème de correspondance est le plus exigeant en termes de complexité de calcul, et implique la recherche et l'appariement technique (pour localiser un objet commun dans les deux images). L'objectif est de concevoir un modèle de système 3D fonctionnel sur support embarqué afin d'évaluer les possibilités d'intégration on chip d'un système de vision 3D.
Le travail à réaliser commence tout d'abord par l'importation de deux images. Dans un second temps, il s'agira de pouvoir porter l'algorithme sur la carte de développement NanoBoard 3000 d'Altium.

III. Cahier des charges

Le but recherché lors du début de la conception de ce programme de vision par système embarqué était l'obtention d'une solution complète, couvrant l'ensemble des étapes de la fonction de corrélation. Il consiste à suivre les étapes suivantes :

- Savoir utiliser les logiciels d'Altium.
- Connaitre les étapes de configuration de processeurs embarqués.
- Connaitre le fonctionnement de l'afficheur VGA.
- Programmer un algorithme à base de l'évaluation d'une fonction de corrélation
- Portage de l'algorithme sur la plate-forme embarquée.
- Interpréter les résultats.

Conclusion

Ce chapitre donne une vue générale de l'organisation générale du laboratoire Signaux Systèmes et Composants (L.S.S.C) du département génie électrique, ainsi le contexte global de notre projet. Dans le chapitre qui suit nous allons introduire la vision stéréoscopique, principe et différentes algorithmes pour réaliser ce projet.

Chapitre II

La vision stéréoscopique

Dans le domaine du divertissement vidéo, d'environnement 3D, et de la vidéosurveillance intelligente en général, on tente sans cesse d'augmenter le réalisme et d'améliorer les moyens de sécurité. Qu'il s'agisse de la vidéo couleur ou de la récente télévision haute définition.
En effet, une vidéo stéréoscopique est réalisée par deux caméras filmant la scène à partir de deux points de vue légèrement distants. Ces deux vues sont alors présentées de manière à ce que l'image de la caméra gauche soit vue uniquement par l'œil gauche, et l'image de la caméra droite par l'œil droit pour donner un effet de stéréoscopie.
Cependant, la manière d'obtenir une information précise de la scène à partir d'images stéréoscopiques reste toujours un défi, surtout dans le domaine de la robotique où une reconnaissance spatiale de l'environnement entourant le robot est nécessaire à son déplacement.
Nous nous intéressons dans ce chapitre au principe de la vision stéréoscopique et notamment : aux algorithmes et techniques utilisés pour reconstruire l'image 3D.

I. Principe de la stéréoscopie

I.1. Principe de la vision stéréoscopique

La stéréoscopie regroupe toutes les techniques à but d'extraction d'informations de la même scène à partir de deux images classiques fournies par deux capteurs (Figure 1.2). En effet, le fait d'avoir deux angles de vues différents d'un même objet nous permet d'estimer la profondeur.

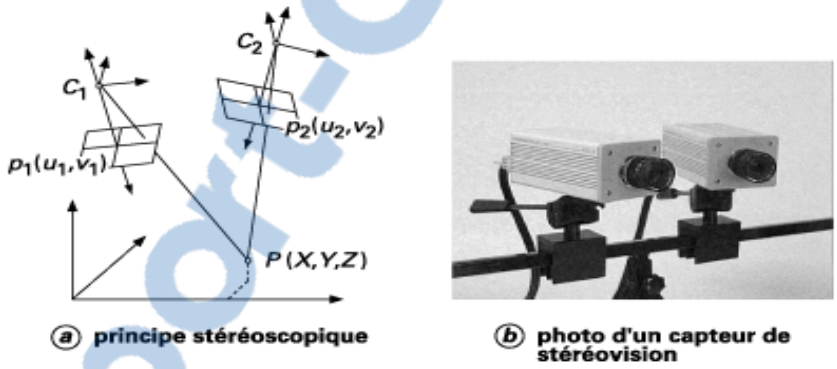


Figure 2.1: Principe de la stéréoscopie

Cette méthode est apparue presque en même temps que la photographie. L'objectif d'un système de vision est de reconstruire la scène 3D observée par la caméra. Cela veut dire récupérer la dimension qui est perdue lors de la projection de la scène tridimensionnelle sur le plan image 2D. Cette reconstruction est faite au moyen d'une chaine de traitements, comme illustré en figure 2.2

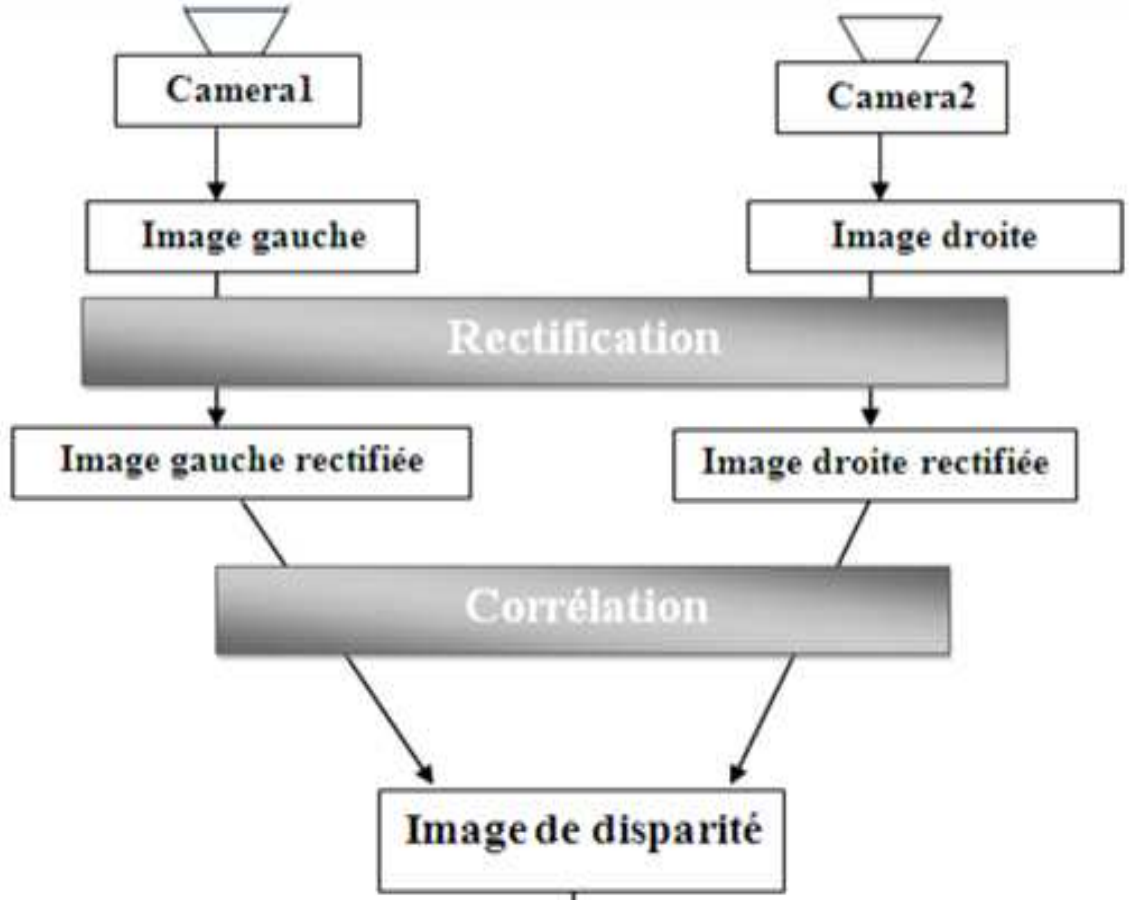


Figure 2.2: Synoptique général de la vision stéréoscopique

Le principe de la vision stéréoscopique se base sur deux étapes principales qui sont : la rectification et la corrélation.
La rectification est le processus qui consiste à appliquer des corrections géométriques sur une image de synthèse à partir d'informations prélevées dans une image source (ou image brute) en fonction du modèle de géométrie choisi. Tandis que la corrélation est l'action de mettre en correspondance les points similaires entre les deux images.
On dit alors que l'on rectifie l'image brute et l'image synthétisée s'appelle image rectifiée.

I.2. Principe de la rectification épi-polaire

I.2.1. Rappels sur la géométrie épi-polaire

La rectification d'une paire d'images stéréoscopiques permet de se ramener à une géométrie épi-polaire simple où les droites épi-polaires sont parallèles aux lignes des images. Ceci permet de réduire la recherche du point correspondant d'une image sur une droite horizontale de la seconde image située à la même ordonnée.
La figure ci-dessous montre que lorsqu'un point M de la scène est visible simultanément par les deux caméras dont les centre C_1 etC_2, sa projection dans les deux images gauche et droite nous donne respectivement $m_1$et m_2. Le stéréo correspondant au m_1 dans l'autre image est une droite appelée droite

épi-polaire, issue de m_2 et notée D_{E_2}. Cette droite est définie par l'intersection du plan support du triangle (C1, C2, m_1) et du plan support de l'image2, tandis que la droite D_{E_1} présente l'intersection du plan support du triangle (C1, C2, m_1) et du plan support de l'image1. Toutes les droites épi-polaires dans l'image2 (resp.image1) convergent vers un point, qui est l'intersection de la droite portant le segment (C_1,C_2) et du plan support de l'image2 (resp. image1). Ce point est le centre épi-polaire de l'image2 C_{E_2} (resp. image1 C_{E_1}).

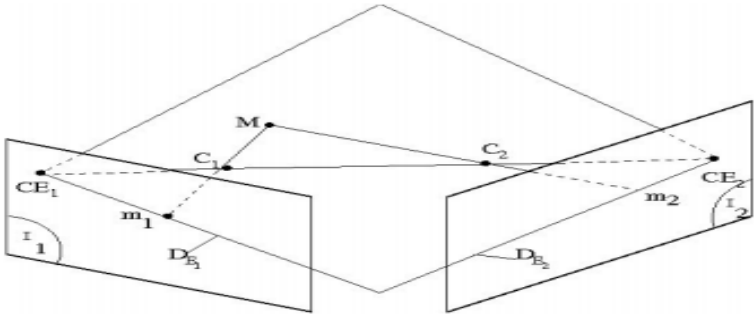


Figure 2.3: Géométrie épi-polaire

I.2.2 Rectification épi-polaire

La méthode de rectification consiste à réorienter les lignes épi-polaires pour qu'elles soient parallèles avec l'axe horizontal de l'image. Cette méthode est décrite par une transformation qui projette les épi-pôles à l'infini et dont les points correspondants sont nécessairement sur la même ordonnée (figure 2.4).

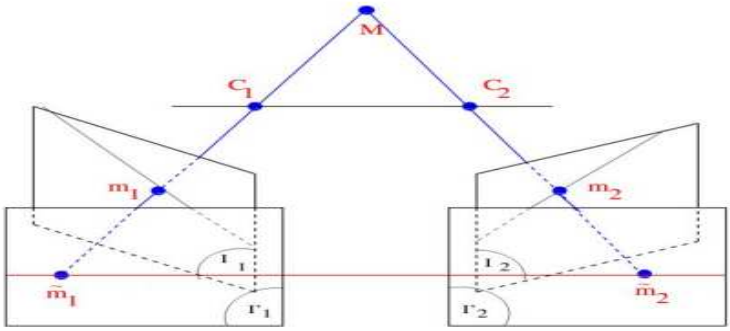


Figure 2.4 : opération de rectification

Le but recherché est la parallélisation des lignes épi-polaires, de façon à simplifier la recherche de correspondance entre deux images. En effet, après rectification chaque point d'une image trouve son correspondant dans l'autre image.
Dans notre cas on a implémenté un algorithme proposé par A. Fusiello [6]. Le principe est de définir un nouveau repère rectifié

Figure 2.5 : Images stéréoscopiques avant et après rectification épi-polaire

Après la rectification des deux images gauche et droite, on se trouve devant la reconstruction qui consiste à savoir quel élément de l'image 1 correspond au point de l'image 2. Cette mise en correspondance de point est ce qu'on appelle l'appariement stéréoscopique.

I.2.3. Contrainte épi-polaire

La contrainte épi-polaire est une contrainte forte due à la géométrie des caméras. Son expression brute est la suivante : tout point de l'image de gauche trouve son correspondant sur une ligne de l'image de droite appelée ligne épi-polaire et réciproquement.

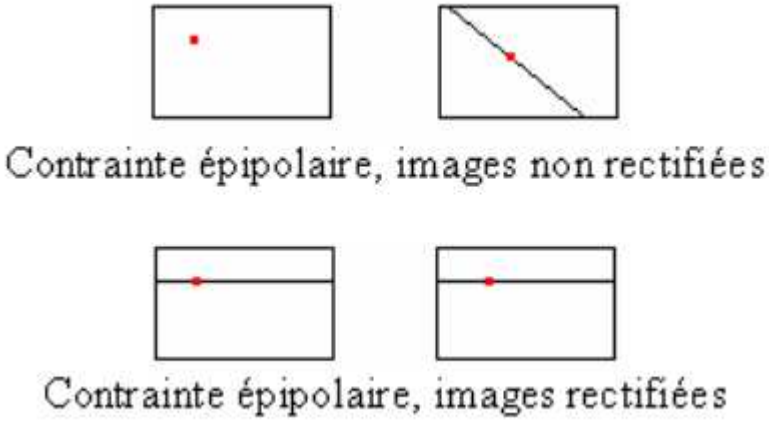


Figure 2.6. Contrainte épi-polaire

La contrainte dépend de la géométrie du système et permet de réduire le problème de l'appariement à la recherche sur une ligne. La rectification permet de simplifier grandement cette recherche d'un point de vue algorithmique.

II. Les algorithmes d'appariement

Les algorithmes d'appariement sont au cœur du processus de la vision 3D. Ils consistent à associer les pixels correspondants entre l'image droite et l'image gauche. En déterminant toutes les paires de pixels appariés dans une image, nous pouvons tracer une carte de disparité. Cette dernière correspond à la distance sur la ligne entre les coordonnées en abscisse de chacune des projections.

II.1. Mesures de similitude

La mesure de similitude constitue la base de l'algorithme d'appariement. En effet, il s'agit d'une mesure algébrique qui permet de déterminer les paires de points et calculer la ressemblance entre deux pixels. En pratique cette mesure de ressemblance est réalisée sur une fenêtre et non pas sur un seul pixel. Cette fenêtre, appelée fenêtre de corrélation présente le voisinage de pixel (figure 2.7).

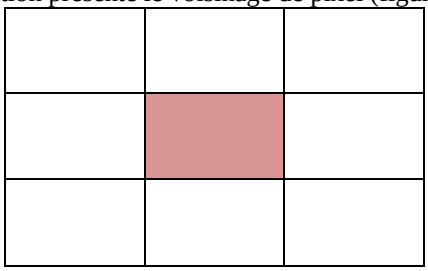


Figure 2.7: fenêtre centrée sur le pixel à apparier

On peut chercher la corrélation entre deux pixels et leurs voisinages avec une taille de voisinage plus ou moins importante comme le montre la figure2.7 de façon:

- à prendre la même fenêtre de corrélation sur l'image gauche et l'image droite ;
- à prendre une fenêtre de corrélation connexe (en un seul morceau) [7];
- que la fenêtre de corrélation contienne le pixel qu'on cherche à apparier.

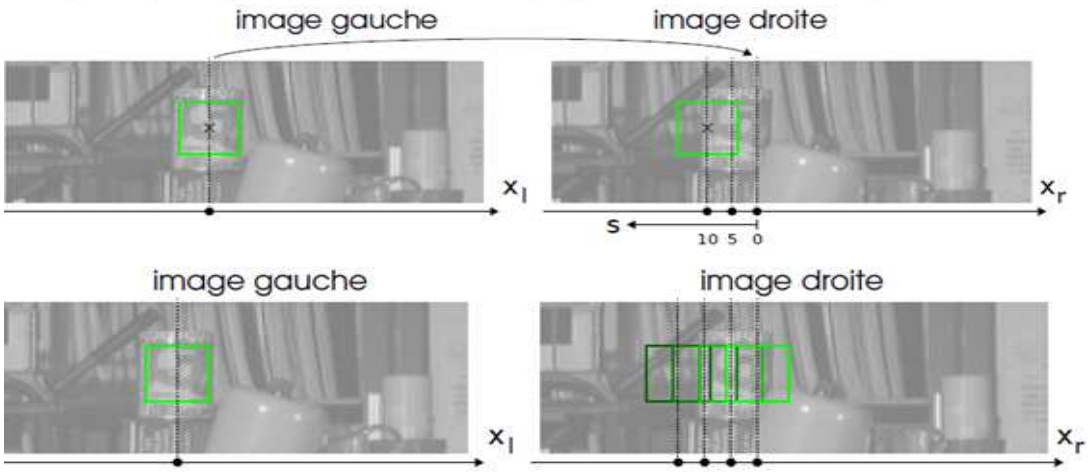


Figure 2.8: principe de ressemblance par la fenêtre de corrélation

II.1.1. Algorithmes de similitude

Les mesures de similitude sont des mesures qui travaillent sur la ressemblance directe en intensité entre les pixels des images à apparier. Afin de mesurer la similitude entre deux voisinages de pixels on va réaliser la somme des similitudes sur un voisinage du pixel à apparier.
Le tableau ci-dessous présente les différentes méthodes de mesures de similitude:

Somme des différences absolues	SAD	$\sum_{(u,v)\in w} \lvert I_1(u,v) - I_2(x+u,y+v)\rvert$
Somme de carré des écarts	SSD	$\sum_{(u,v)\in w} \lvert I_1(u,v) - I_2(x+u,y+v)\rvert^2$
Corrélation croisée normalisée	NCC	$\frac{\sum_{(u,v)\in w}\lvert I_1(u,v).I_2(x+u,y+v)\rvert}{\sqrt{\sum_{(u,v)\in w}\lvert I_1(u,v)\rvert^2.\sum_{(u,v)\in w}\lvert I_2(x+u,y+v)\rvert^2}}$
Somme des écarts absolus centrés	ZSAD	$\sum_{(u,v)\in w} \lvert (I_1(u,v) - \overline{I_1}) - (I_2(x+u,y+v) - \overline{I_2})\rvert$
Somme de carré des écarts centrés	ZSSD	$\sum_{(u,v)\in w} \lvert (I_1(u,v) - \overline{I_1}) - (I_2(x+u,y+v) - \overline{I_2})\rvert^2$
Corrélation croisée normalisée	ZNCC	$\frac{\sum_{(u,v)\in w}\lvert (I_1(u,v) - \overline{I_1}) - (I_2(x+u,y+v) - \overline{I_2})\rvert}{\sqrt{\sum_{(u,v)\in w}\lvert I_1(u,v) - \overline{I_1}\rvert^2.\sum_{(u,v)\in w}\lvert I_2(x+u,y+v) - \overline{I_2(x,y)}\rvert^2}}$

centrée		

Tableau : Mesures de Similitude [7]

$I_1(u,v)$: Valeur de pixel de cordonnées u et v de l'image gauche.

$I_2(u,v)$: Valeur de pixel de cordonnées u et v de l'image droite.

$\overline{I_1}$: Valeur moyenne des pixels de l'image gauche.

$\overline{I_2}$: Valeur moyenne des pixels de l'image droite.

- L'algorithme somme des différences absolues SAD est l'un des plus simple des mesures de similarité, elle se base sur la sommation de différence des pixels en un voisinage carré entre l'image de référence 1 et l'image 2 cible. Si les images gauche et droite correspondent exactement, la résultante sera nulle.
- L'algorithme ZSAD est similaire à SAD, en soustrayant la moyenne des pixels de chaque pixel.
- En somme des carrés des écarts SSD, les différences sont affrontés et regroupées au sein d'une fenêtre carrée. Cette mesure a une plus grande complexité de calcul par rapport à l'algorithme SAD parce que cela implique de nombreuses opérations de multiplication.
- Dans la méthode ZSSD on fait la soustraction de la moyenne de la zone de jeu à partir de chaque valeur d'intensité. Toutefois, la soustraction de la moyenne à partir des intensités au carré ajoute à la complexité de calcul.
- La corrélation croisée normalisée NCC est encore plus complexe car elle implique de nombreuse multiplication, division et les opérations de racine carrée.
- ZNCC est similaire à la CCN avec la seule différence de la soustraction de la valeur moyenne locale d'intensité. Ainsi, la technique de calcul reste a même et de la correspondance peut être obtenue de la même manière que CCN [9,10].

Les techniques de type SAD, SSD, NCC ont un inconvénient majeur [11,12] : elles sont extrêmement sensibles aux différences de gain et de luminosité entre les deux images de la paire stéréoscopique. Pour pallier à ce problème on a développé les techniques de type ZSAD, ZSSD, ZNNC qui normalisent l'intensité par rapport à la valeur de l'intensité moyenne sur la fenêtre étudiée. Cela permet de s'affranchir des problèmes dus aux différences de luminosité entre les images.
Par ailleurs d'un point de vue temps, les techniques SAD et SSD [11] sont les plus simples et donc les moins coûteuses en temps de calcul. Donc les biens applicables en systèmes embarqués dans le domaine de la robotique.

III. Les algorithmes de la Stéréovision dense par corrélation SAD et SSD

Les techniques corrélatives se basent sur l'appariement de points image par minimisation d'un score de corrélation. Ce score est mesuré par la somme des écarts des pixels voisins des deux points à apparier.
On se base dans ce projet sur le calcul de l'indice de corrélation par deux méthodes importantes SAD (Sum of Absolute Differences) et SSD (Sum of Squared Differences).

Le principe est de considérer, pour un pixel $P_g(i,j)$ de l'image gauche, une fenêtre rectangulaire centrée en $P_g(i,j)$ et de calculer sa corrélation/distance avec une fenêtre dans la deuxième image. La fonction de corrélation est alors maximum en $P_d(k,l)$ correspondant de $P_g(i,j)$ dans la deuxième image (distance minimum).

III.1. Les étapes de décomposition de l'algorithme

Nous rappelons brièvement les différentes étapes de la mise en correspondance par corrélation.
Pour chaque pixel $P_g(i,j)$ de l'image de gauche figure (7,8(a)), il faut :

- Déterminer la zone de recherche dans l'image de droite, Z_d c'est-à-dire, la zone dans laquelle on pense trouver le pixel correspondant.
- Calculer le score de corrélation pour chaque pixel $P_d(k,l)$ de la zone de recherche en utilisant la mesure de corrélation choisie (SSD ou SAD).
- Sélectionner le pixel qui donne le meilleur score de corrélation.

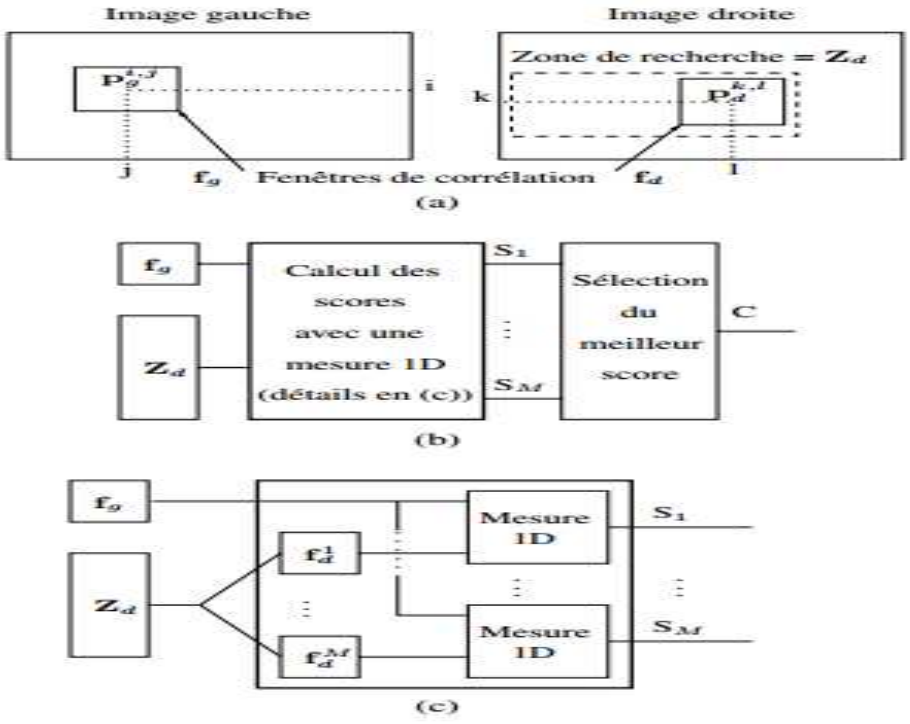


Figure 2.9 : Mise en correspondance par corrélation

De manière générale, les tailles de la zone de recherche et de la fenêtre de corrélation sont déterminées de manière empirique.
Le résultat de la mise en correspondance est une disparité $D_g(k,l)$ pour chaque pixel $P_g(i,j)$ définie par :
Si $P_g(i,j)$ et $P_d(k,l)$ se correspondent alors $D_g(i,j) = (k - i \;\; l - j)^T$
Dans le cas d'images ayant subi une rectification épi-polaire, nous avons : i = k. Une des manières de représenter les résultats de la mise en correspondance est d'utiliser une carte de disparités (figure 2.9).

(S = Score, C = Correspondant) – (a) Zone de recherche et fenêtres de corrélation. (b) Recherche d'un correspondant en utilisant une mesure 1D. (c) Détails du calcul des scores avec une mesure 1D.

Chaque pixel de la carte de disparités représente l'amplitude de la disparité (Figure 2.9), c'est-à-dire, la distance entre le pixel de l'image gauche et son correspondant dans l'image droite. Ici, plus le pixel est clair, plus la disparité est grande et plus le point de la scène est proche des caméras. Les pixels noirs correspondent aux pixels sans correspondant.

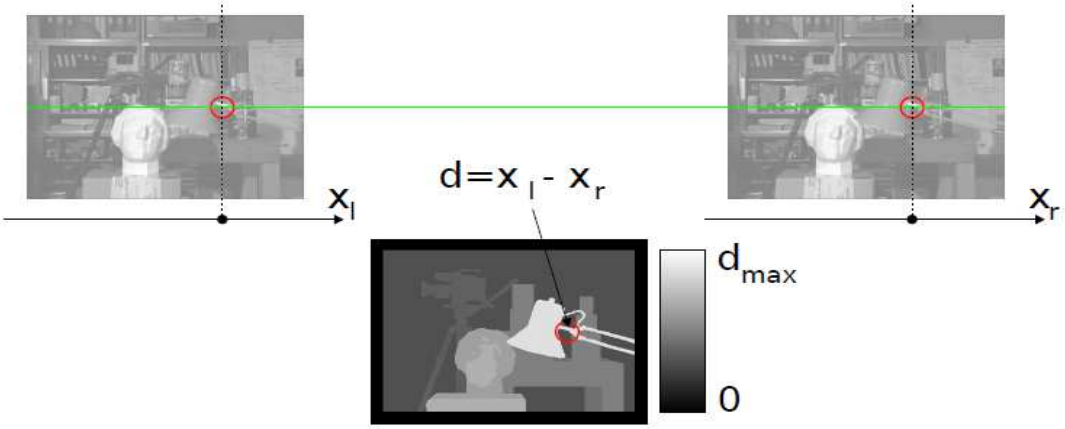


Figure 2.10: la carte de disparité

Conclusion

La conception d'un algorithme d'appariement se fait donc par le choix de la mesure de similitude utilisée. Dans le prochain chapitre on s'intéresse à la conception et l'application des algorithmes SAD et SDD sur l'image numérique en utilisant les langages de programmation Matlab et langage C. En effet, pour pouvoir procéder à un montage numérique de couples stéréo il faut bien sûr disposer d'images numériques, d'un ordinateur et de logiciels adéquats.
On va s'intéresser aux différentes étapes de démarche de projet, pour l'implémentation de l'algorithme sur la FPGA de la carte NanoBoard 3000 d'Altium.

Université Sidi Mohamed Ben Abdellah
Faculté des Sciences et Techniques Fès
Département Génie Electrique

CHAPITRE III

CONCEPTION ET REALISATION DE L'APPLICATION

Année Universitaire 2012- 2013

Ce chapitre est consacré à la vérification des fonctionnalités des algorithmes choisis.
Les tests de validation se déclinent généralement en plusieurs phases:

- **Validation fonctionnelle :**

Les tests fonctionnels vérifient que les différents modules ou composants implémenté répondent correctement aux exigences du cahier de charge. Ces tests peuvent être de type valide, invalide, inopportun.

- **Validation solution :**

Les tests solutions vérifient les exigences du cahier de charge d'un point de vue "use cases", généralement ces tests sont des tests en volumes. L'intérêt est de valider la stabilité d'une solution par rapport aux différents modules qui la composent.

- **Validation performance robustesse :**

Les tests de performance vont vérifier la conformité de la solution par rapport à ses exigences de performance, alors que les tests de robustesse vont essayer de mettre en évidence des éventuels problèmes de stabilité et de fiabilité dans le temps.
Dans les deux parties suivantes nous allons donner une brève description des différents outils utilisés pour l'implémentation, ensuite une étude de faisabilité des algorithmes de stéréoscopie.

I. Outils de conception et réalisation

I.1. Outils de validation fonctionnelle

Ce paragraphe ne décrit pas l'utilisation de chaque logiciel en détail. Il s'agit uniquement de donner une idée de leur utilité, et ainsi justifier leur utilisation.
Nous avons utilisées le DEV C++ et Matlab

I.1.1. Logiciel DEV C++ [8]

Dev-C++ est un environnement de développement intégré (IDE) permettant de programmer en C et en C++. Dev-C++ est disponible uniquement sous Microsoft Windows.
Cet IDE complet comprend entre autres un « répertoire de classes », servant à localiser facilement les fonctions, classes et membres du code source, un « répertoire de fonctions incluses », fonctionnant comme le répertoire de classes mais pour chercher dans les fichiers inclus (header), et un débogueur qui permet de surveiller l'état des variables pendant l'exécution du programme. Il souffre en revanche de l'absence d'un éditeur de ressources, ce qui rend la conception d'applications délicate si on ne fait pas appel à un outil externe.

I.1.2. Logiciel Matlab [9]

MATLAB est un environnement complet, ouvert et extensible pour le calcul et la visualisation. Il dispose de plusieurs centaines (voire milliers, selon les versions et les modules optionnels autour du noyau Matlab) de fonctions mathématiques, scientifiques et techniques. L'approche matricielle de MATLAB permet de traiter les données sans aucune limitation de taille et de réaliser des calculs numériques et symboliques de façon fiable.

I.2. Outils de validation : Solution et Performances

Un des plus grands défis des ingénieurs et des développeurs est de tester une nouvelle idée, et de la concevoir.
Le plus souvent, ces tests ont besoin d'un système complet. Altium Designer, couplé au NanoBoard3000, permet de construire un système complet, avec l'accès aux processeurs soft et hard, la logique programmable, toutes sortes d'entrée/sortie, un tas d'instruments de test cool peuvent être intégrées dans le design, et outils de développement de PCB.

I.2.1. Spécification matériel [10]

Le produit Altium Designer intègre un environnement de conception et de vérification de systèmes au niveau FPGA dédié au développement des applications à base des langages de description Hardware VHDL, verilog ou même en C.
La NB3000AL d'Altium Designer offre :

- Une plate-forme matérielle de développement qui met à profit la puissance d'une composante programmable de hautes capacités à bas coût, permettant ainsi rapidement développer et de mettre en œuvre vos systèmes.
- Un parfait point d'entrée pour découvrir et explorer l'univers de la conception de systèmes embarqués à base de FPGA. L'univers du matériel programmable vous permet de mettre à jour votre conception rapidement et de nombreuses fois sans pénalisation en temps ou en argent.
- Un fonctionnement de manière transparente et en parfaite intégration avec Altium Designer la solution de conception électronique de prochaine génération d'Altium.
- FPGA haute capacité situé sur la carte-mère et possibilité d'une carte périphérique plug-in (Altium ou développée par l'utilisateur) pour une flexibilité accrue du système
- Détection et configuration automatique de la carte périphérique.
- Double système de boot permettant à la carte de mettre elle-même à jour son firmware sur le terrain, via une connexion USB standard - sans nécessiter de port parallèle ni d'adaptateur JTAG US.

Figure 3.1: La NanoBoard 3000AL

I.2.2. Spécification logicielle [11]

Le logiciel Altium Designer version 10 introduit un nouveau concept intéressant : l'Altium Innovation Station, qui combine Altium Designer et l'Altium Desktop NanoBoard pour créer un environnement de développement qui place l'intelligence des composants programmés au cœur du processus de conception.
Altium Designer et l'Altium Desktop NanoBoard opèrent de manière transparente pour offrir:

- Une solution unifiée de conception du logiciel et du matériel
- Un modèle de données de conception
- Une plate-forme de développement
- L'indépendance totale par rapport aux fournisseurs de composants matériels et logiciels.
- De très nombreuses possibilités de déploiement matériel

Altium Designer est un outil complet de développement de produits électroniques incluant :

- un outil de routage PCB
- un outil de simulation SPICE
- un outil de développement FPGA
- un outil de développement de code embarqué

II. Etude de faisabilité des algorithmes choisis

Le but du projet consiste à appliquer un algorithme de stéréoscopie sur deux images pour avoir l'image stéréoscopique.

II.1. Implémentation de stéréoscopie sur une image

Pour cela nous avons essayé de valider l'algorithme de stéréoscopie en premier lieu sur ordinateur, afin de le préparer pour la phase de portage sur la NanoBoard3000Al d'Altium Designer.
Dans un premier temps, nous avons travaillé sur un programme en Matlab, qui lit les matrices de pixels des deux images et extrait la matrice de disparité. Ce logiciel représente l'environnement le plus convenable pour notre réalisation, car il dispose d'une base de fonctions prédéfinies assez importante surtout pour les applications de traitement d'image et de la vidéo. Matlab est principalement un logiciel de simulation, et donc il nous apparaît indispensable de profiter de cet environnement dans notre projet.
Puis on a écrit un code en C avec le logiciel DEV C++, en se basant sur les résultats de Matlab. Ce code sera adapté en C embarqué pour l'implémenter après sur la FPGA.

II.1.1. Programmation en Matlab

Comme nous avons vu dans le chapitre précédent, l'algorithme choisi se base sur la sommation de chaque pixel de la 'fenêtre différence' des deux fenêtres de corrélation gauche et droite.

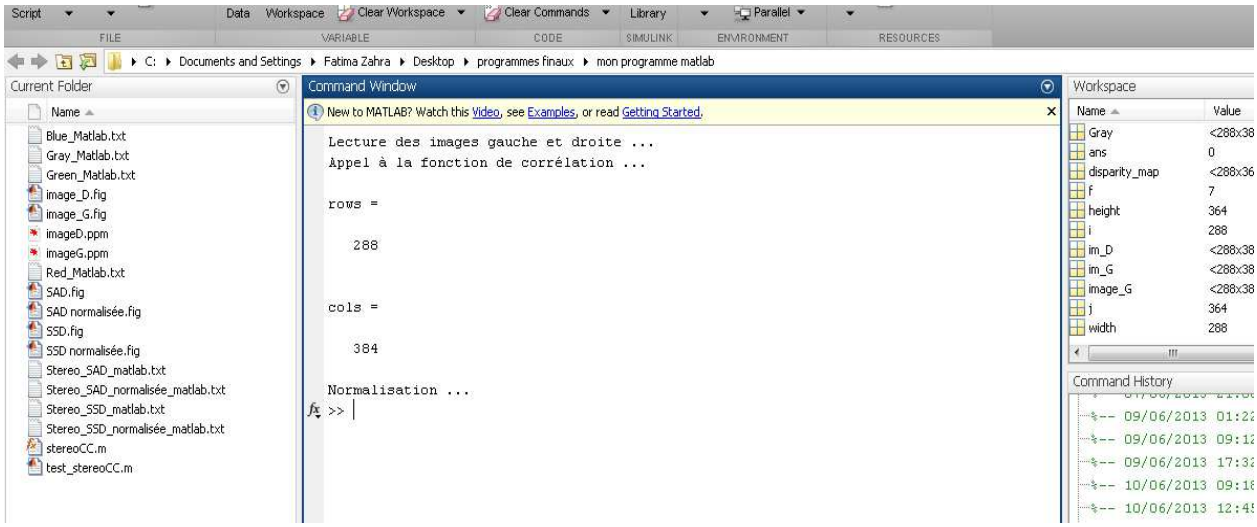


Figure 3.2: commande pour lire les images et appliquer l'algorithme

La figure 3.3 présente un organigramme simplifié l'algorithme.

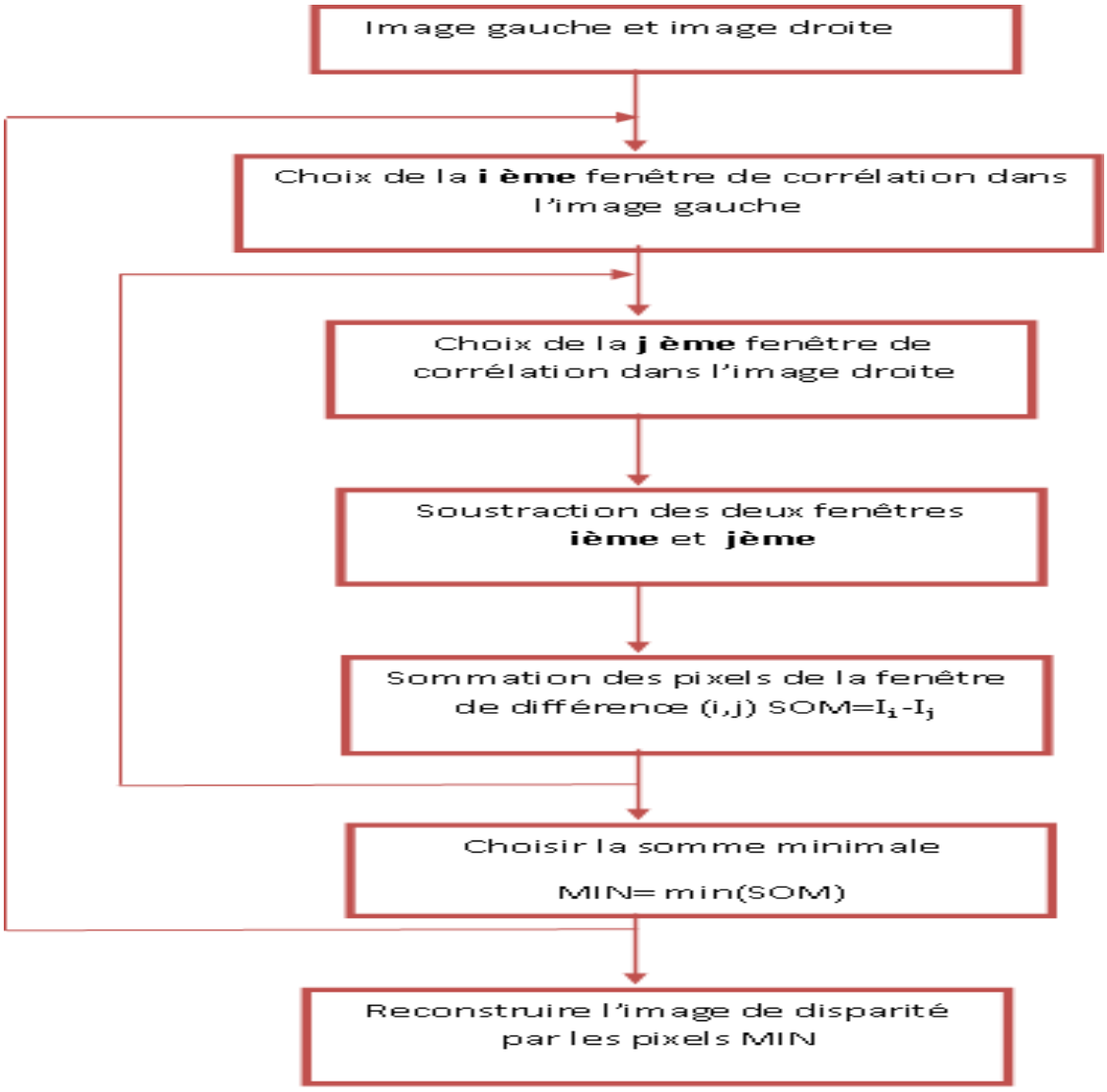


Figure 3.3: Organigramme de l'algorithme de stéréoscopie sur Matlab

- Résultat de la simulation :

La figure suivante montre l'image de reconstruction par les deux méthodes SAD et SSD :

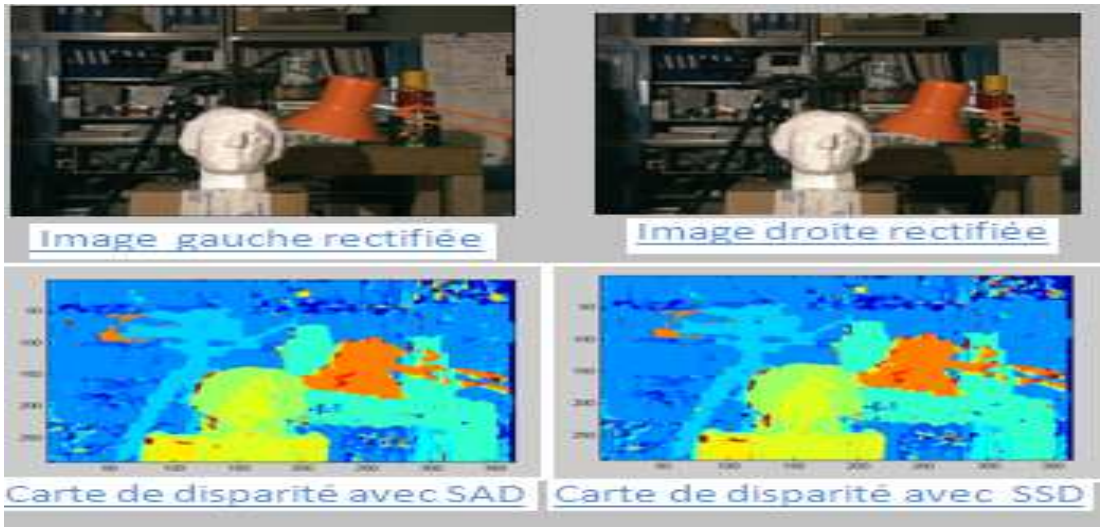


Figure 3.4: image de disparité en utilisant l'algorithme SAD et SSD

Comme c'est déjà mentionné dans le chapitre précédent, il n'existe pas une grande différence entre les deux algorithmes, seulement au niveau de lissage.

II.1.2 programmation en C

L'algorithme en DEVC++ (voir Annexe 2), nous génère deux fichiers textes: un fichier texte contenant les valeurs des pixels de l'image de reconstruction avec l'algorithme SAD et l'autre contient celle de l'algorithme SSD.

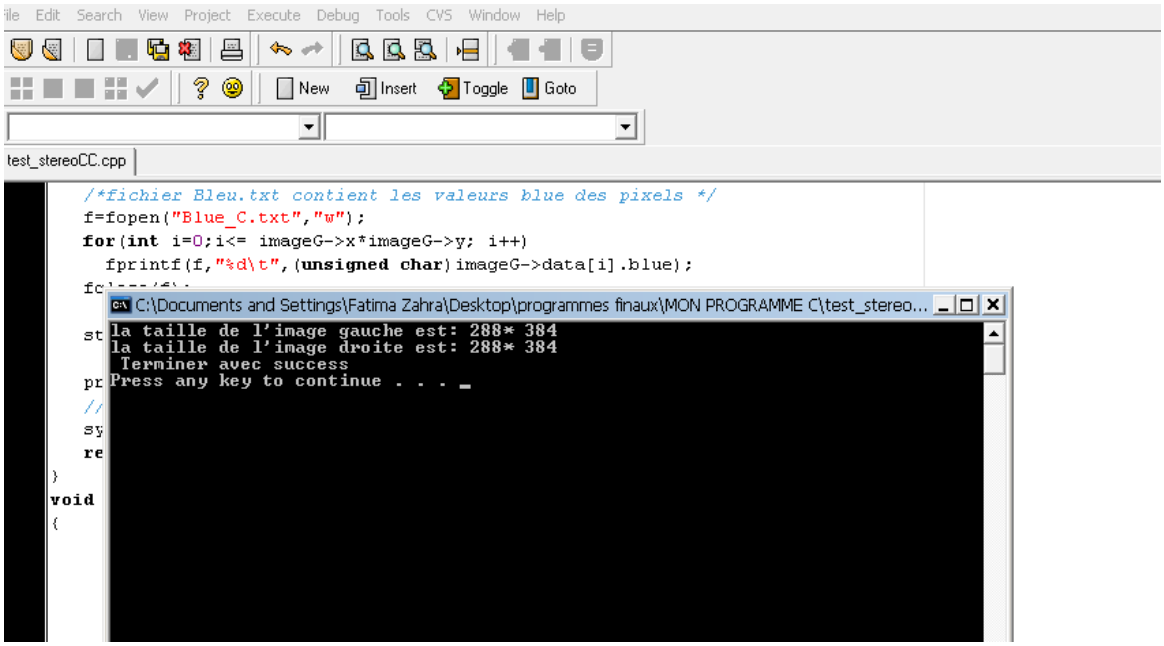


Figure 3.5: commande pour exécuter l'algorithme de stéréoscopie

Cette étape est nécessaire pour connaître la qualité de programme, il nous reste pour l'implémentation une adaptation de ce code C avec les fonctions de logiciel Altium Designer.

Conclusion

Dans ce chapitre, nous avons parlé des logiciels utilisés dans ce projet, ainsi les caractéristiques du logiciel ALTIUM DESIGNER, en dimensionnant le système qui permet d'afficher le L'image 3D sur l'écran de la carte de FPGA.

Université Sidi Mohamed Ben Abdellah
Faculté des Sciences et Techniques Fès
Département Génie Electrique

CHAPITRE IV

Implémentation de l'algorithme sur la carte NanoBoard 3000

Ce chapitre est consacré à un état de l'art sur les méthodologies de développement d'applications pour les systèmes embarqués, et notamment pour les systèmes basés sur composants reconfigurables du type FPGA. L'objectif est de s'appuyer sur les solutions existantes afin d'en extraire les éléments permettant de proposer une approche originale, dédiée aux architectures du type caméra intelligente basées sur FPGA.

I. Méthodologie de travail

Ce travail se concentre sur le dimensionnement d'un système embarqué pour l'optimisation des capacités de stockage du système de vidéosurveillance. La figure suivante illustre les phases du flot de conception adopté dans ce projet.

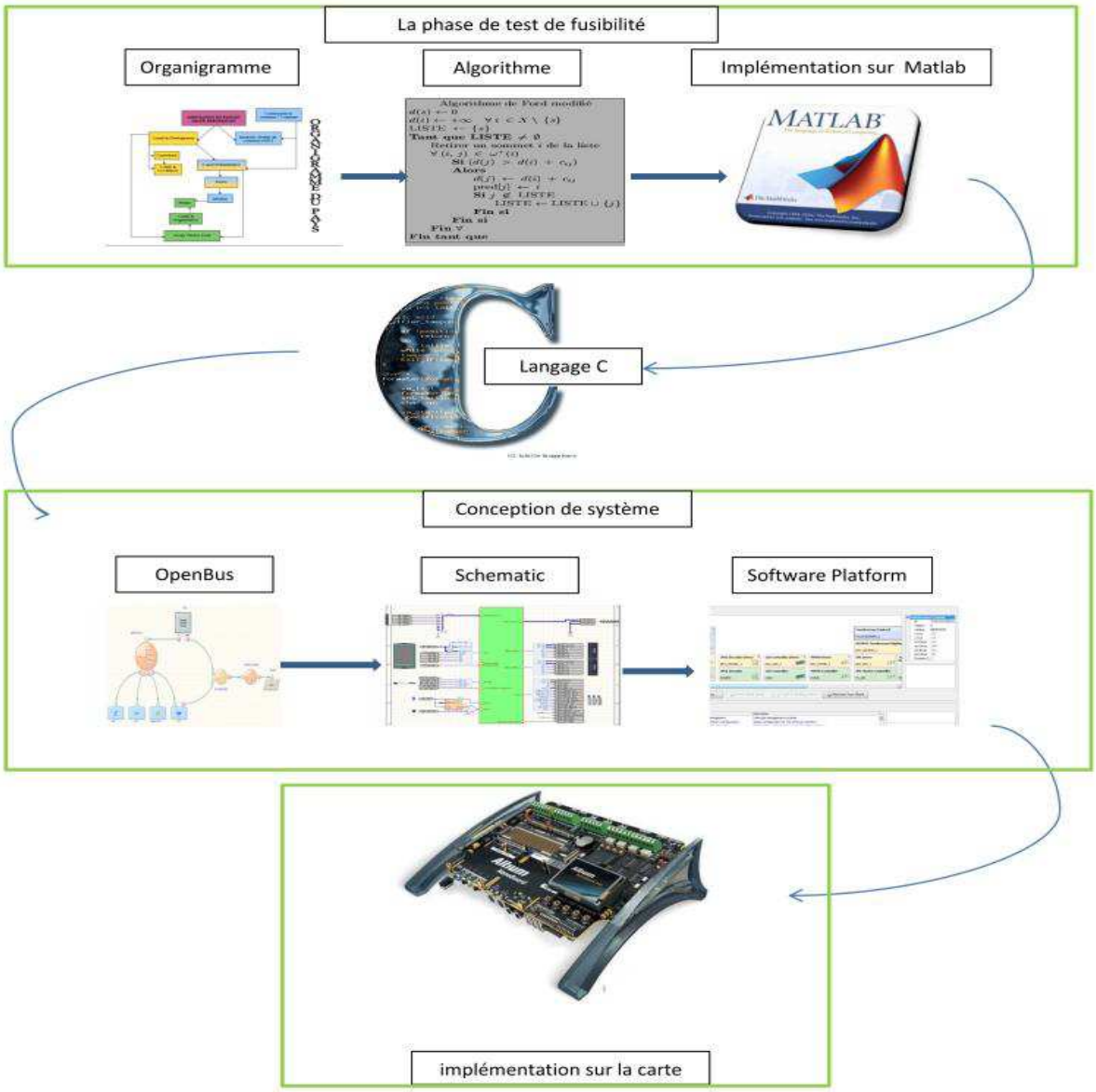


Figure 4.1: Etapes et méthodologie de travail

Notre projet se focalise, après l'étude de l'algorithme de stéréoscopie, sur l'implémentation qui se base sur la réalisation de deux étapes importantes : partie matérielle et autre logicielle. Nous commençons en premier temps par la partie matérielle: faire le dimensionnement de notre système en utilisant l'architecture graphique « OpenBus » permettant de placer les IPs (Intellectual Property) nécessaires pour l'application cible.

Puis, nous sommes passés à la phase logicielle en commençant par la création d'un nouveau projet embarqué qui contient deux parties. La première partie nous permettra de générer la liste des drivers pour chaque IP utilisée dans l'architecture matérielle. Concernant la deuxième partie nous sommes arrivés à la phase intéressante qui place

l'intelligence du programme écrit en langage C embarqué vers des composants électroniques en exploitant les différentes bibliothèques de fonctions disponibles dans la plate-forme logicielle. Enfin, nous compilons tous les documents que nous avons construits dans les deux parties de la conception matérielle et logicielle et le tout est envoyé vers la NanoBoard 3000 via JTAG.

I.1. Logiciel Altium Designer 10

Altium Designer est un puissant outil de CAO électronique développé par l'entreprise Altium. Ce logiciel permet de saisir des schémas électriques, les vérifier, les simuler et aller jusqu'à la conception du circuit imprimé.

I.1.1. Caractéristiques

La grande force d'Altium Designer c'est les nombreux outils qu'il intègre par défaut. Ce logiciel représente une seule solution informatique pour concevoir et développer un montage électronique en allant du schéma jusqu'à la programmation des composants de ce schéma. Voici en outres les principales caractéristiques de ce logiciel:

- L'outil unique permet de simplifier le travail entre l'éditeur de schématique, la simulation, la réalisation du circuit imprimé et toutes les autres fonctionnalités
- Réaliser un prototype via le logiciel grâce aux différents outils, incluant un outil de visualisation 3D
- Adapté pour l'avenir (prend en compte certaines nouvelles technologies et mise à jour régulière)
- Nombreuses bibliothèques de composants
- Mise à jour continue

I.1.2. Références

Ce puissant logiciel est utilisé par de nombreuses organisations, incluant des universités et des entreprises de l'aérospatiale, de l'automobile, de la défense, de la médecine, des télécommunications et de la grande distribution. Parmi les grands groupe qui utilise Altium Designer, il est possible de citer la NASA et Sony.

I.2. Hiérarchie des projets

Le terme «Projet» désigne un groupe de documents qui se combine pour former une mono cible, Il faut être prudent lors de la création d'un projet afin de s'assurer que celui-ci soit bien le type de projet voulu.

La hiérarchie des projets sous l'environnement d'Altium est donnée ci-dessous

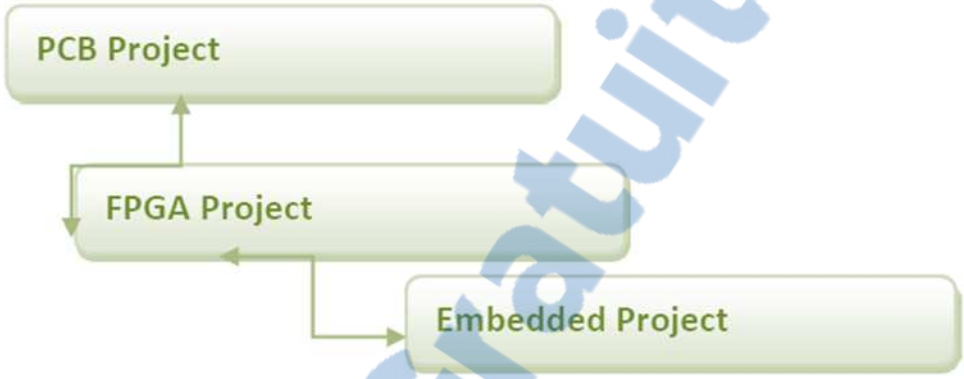


Figure 4.2: Hiérarchie des projets sous Altium

Les projets d'Altium Designer peuvent paraître un peu difficiles, surtout quand les projets contiennent d'autres projets ou sous projets. Une chose importante est que chaque projet ne peut avoir qu'une seule sortie. Un projet de PCB peut contenir un ou plusieurs projets de FPGA, mais jamais l'inverse.

I.3. Les différents types de projet

Altium Designer prend en charge un certain nombre de projets, voici une brève description de quelques un dont nous aurons besoin :

- Projet PCB (*.PrjPcb) : Il représente l'ensemble des documents de conception nécessaires à la fabrication d'un circuit imprimé.
- Projet FPGA (*.PrjFpg) : Ce projet représente l'ensemble des documents de conception qui peuvent être traitées pour un programme de FPGA.

- Projet intégré (*.PrjEmb) : C'est l'ensemble des documents de conception nécessaires pour produire une application logicielle qui peut être intégrée avec son processeur d'exécution dans un produit électronique.
- Bibliothèque intégrée (*. LibPkg) & (*.IntLib) : Représente l'ensemble des documents de conception nécessaires pour produire une bibliothèque intégrée.

II. Etapes de conception de notre projet dans Altium Designer

II.1. Architecture matérielle

Le but de cette partie est d'implémenter des applications sur les images affichées sur l'écran tactile de NanoBoard 3000 et sur l'ecran SVGA, qui sont par la suite appliquées dans les séquences vidéo.
L'architecture matérielle dans l'environnement d'Altium comprend deux parties qui permettent de dimensionner le système, nous commençons par :

Architecture graphique de dimensionnement d'un système « OpenBus »

Une conception de processeurs FPGA peut être divisée en deux parties :

- Système de processeur principal : contenant le processeur et les périphériques.

Les IPs nécessaires pour l'application contiennent un processeur embarqué, les périphériques des mémoires, des entrées/sorties, ainsi que les composants d'interconnexion entre eux.

La figure 4.3 montre la solution hardware qui nous permettra d'afficher les images sur l'écran et ainsi de leur appliquer des traitements :

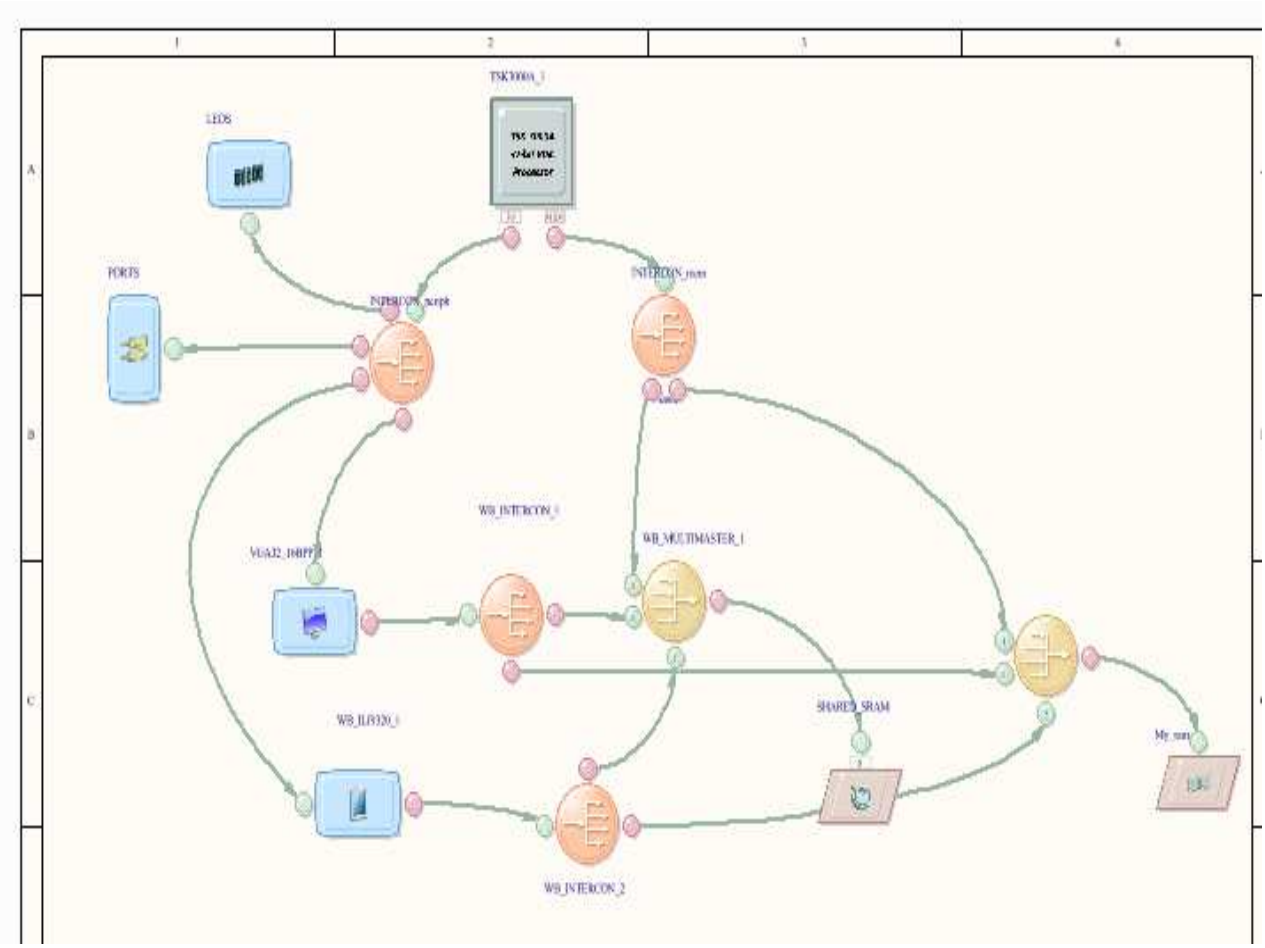


Figure 4.3: Architecture graphique de dimensionnement d'un système « OpenBus »

La liste des IPs utilisés est définie dans l'annexe 4.

- La schématique: cela comprend le câblage entre les périphériques et les broches

physiques du dispositif FPGA (représentées par les composants du port). Il permet de transformer l'architecture prédéfinie dans l'OpenBus en un bloc vert qui a pour objectif de connecter chaque périphérique au composant physique qui existe réellement sur la carte de NanoBoard 3000. La figure 4.4 illustre cette division :

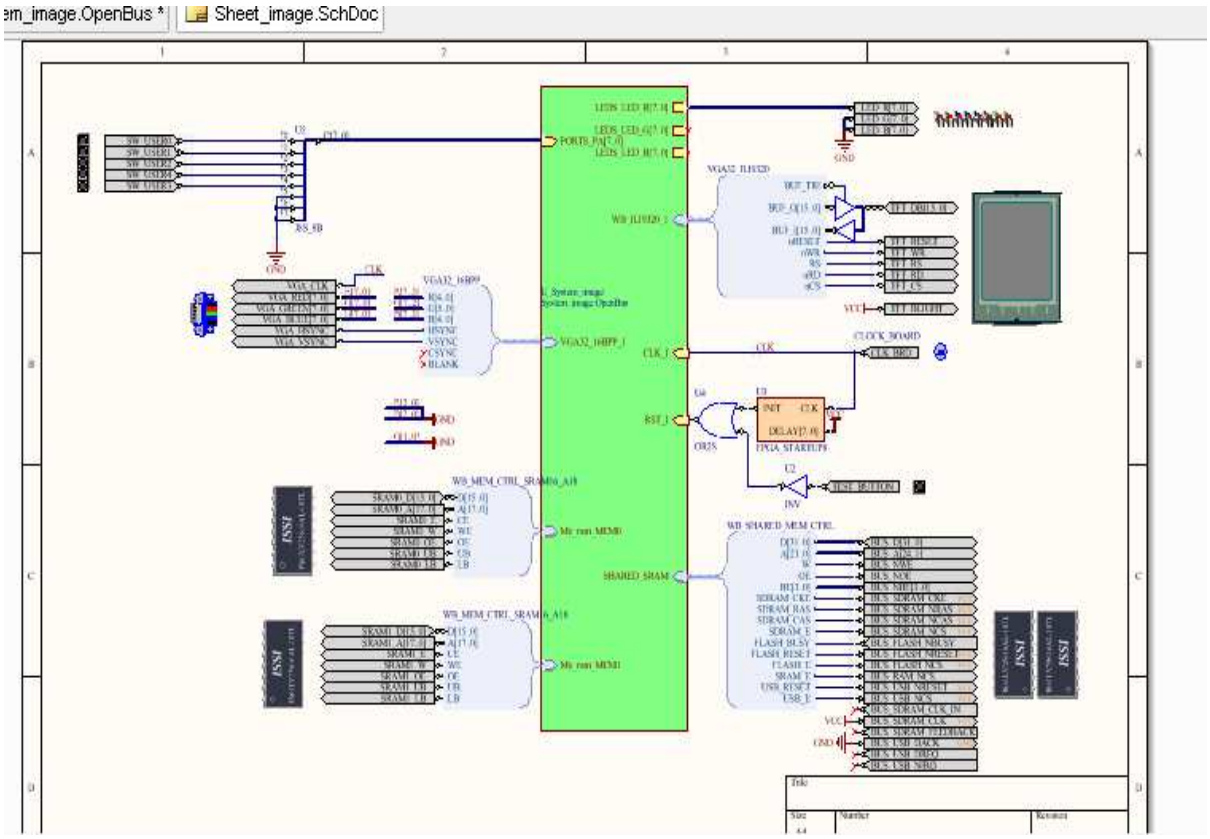


Figure 4.4: Architecture Schématique

La solution matérielle est nommée aussi solution programmable de type logique, les étapes de conception matérielle sont détaillées dans ce qui suit.

Nous avons utilisé (figure 4.3) un processeur embarqué TSK3000A de 32bits parce que le processeur TSK3000 est un «open source » offert par Altium. Par contre les autres processeurs embarqués nécessitent l'achat de la licence (le processeur Nios II de l'ALTERA, Microblaze,...), puis nous avons choisi deux mémoires de taille maximale de 1M ; elles sont connectées au processeur via le port « MEM ». Les autres périphériques d'entrées / sorties tels que les LEDs, l'écran TFT, les buttons poussoirs sont connectés au processeur via le port I / O. Afin d'interconnecter les différents IPs entre eux, nous avons utilisé deux composants d'interconnexion, l'un est appelé « Interconnect component » et est utilisé pour connecter le processeur aux mémoires et l'autre est appelé « component Arbitre » pour le connecter aux autres périphériques d'entrées / sorties.

Par la suite, nous devons posséder le schéma « Top level » (figure 3.8) qui consiste à lier les blocs à l'intérieur de FPGA de l'étape précédente par les différent es ressources qui existent à l'extérieur de FPGA et disponibles sur la carte NanoBoard 3000. Ils sont modélisés sur « schématique » cette connexion est faite soit en utilisant les broches pins) qui lient les LEDs, l'horloge et les interruptions de commande, soit en utilisant le signal Harness qui sert à lier les mémoires et les écrans.

Il ne nous reste maintenant qu'à définir les fichiers de contraintes qui permettent de mapper la conception dans FPGA en spécifiant le nom de chaque entrée / sortie à son correspondant dans la carte.

Altium Designer met à la portée de tous les utilisateurs de NanoBoard un fichier qui regroupe l'ensemble des connexions possibles (voir annexe 4)

II.2. La plate-forme logicielle

OpenBus permet d'assembler rapidement et efficacement le processeur et le matériel du système d'E/S pour les conceptions destinées à être mises en œuvre dans un FPGA. La NanoBoard constitue alors une plate-forme matérielle physique sur laquelle on peut rapidement mettre en œuvre la conception, la tester et la déboguer.

Pour pouvoir rapidement construire l'application qui constitue une partie essentielle du test de notre projet, on dispose également de la plateforme logicielle« Software Platform », il constitue une solution simple qui vient de compléter la construction du système matériel par OpenBus, il permet en

effet au concepteur d'assembler visuellement l'ensemble du code et des pilotes de bas niveau nécessaires aux différents périphériques utilisés dans la conception, en rassemblant automatiquement tout le code de pilotes fourni avec Altium Designer.
La plateforme logicielle nous permet de générer les différents drivers pour chaque IP utilisé dans le dimensionnement du système et d'ajouter d'autres fonctionnalités selon le besoin de l'application.
Dans les systèmes embarqués nous ne disposons pas d'un système d'exploitation auquel on peut accéder à nos données facilement, alors que dans notre application nous avons besoin d'afficher une image sur l'écran, Altium Designer offre un éditeur de lien «--import-object= "image1.bmp"» qui nous permettra d'importer les fichiers nécessaires. La figure suivante montre le portage d'une image :

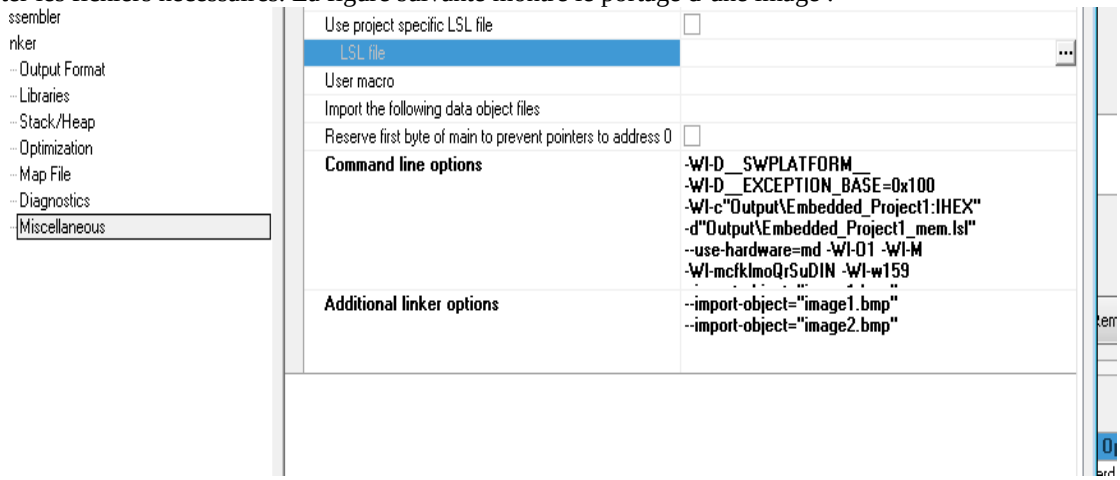


Figure 4.5: illustration du Linker

La conception logicielle est constituée par trois phases principales. La phase de génération des drivers pour chaque IP utilisée dans la solution matérielle. Ces drivers sont configurables et ils sont déclarés sous forme de niveaux (figure 4.6).

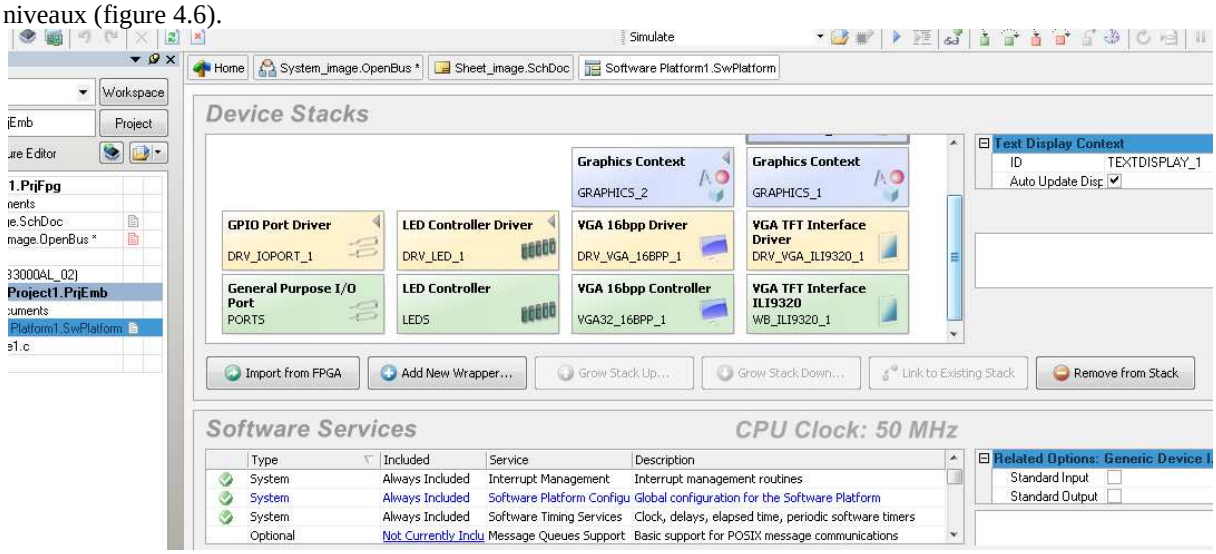


Figure 4.6: niveaux des drivers

La deuxième phase est la configuration des mémoires (figure 4.7) par la détermination de sa taille et les types de mémoires utilisés pour le code et les données.

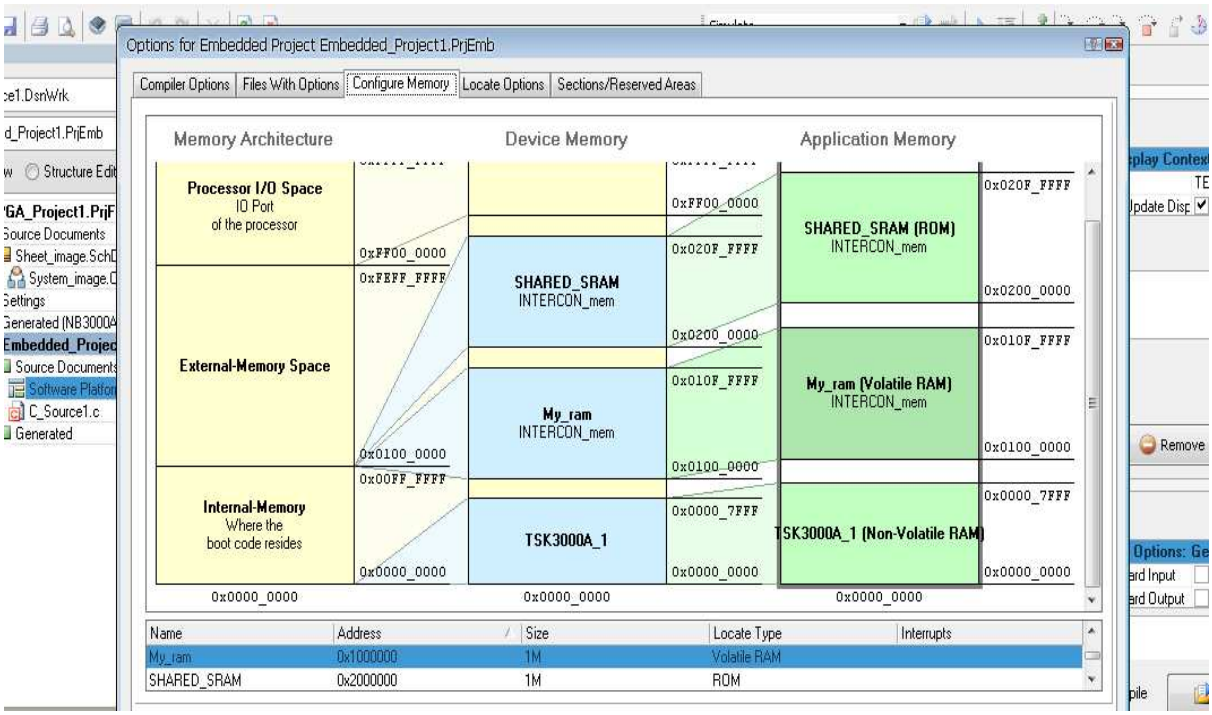


Figure 4.7: L'environnement de la configuration des mémoires

La troisième phase consiste à écrire le programme principal en C embarqué qui a pour but l'affichage de l'image (figure 4.8).

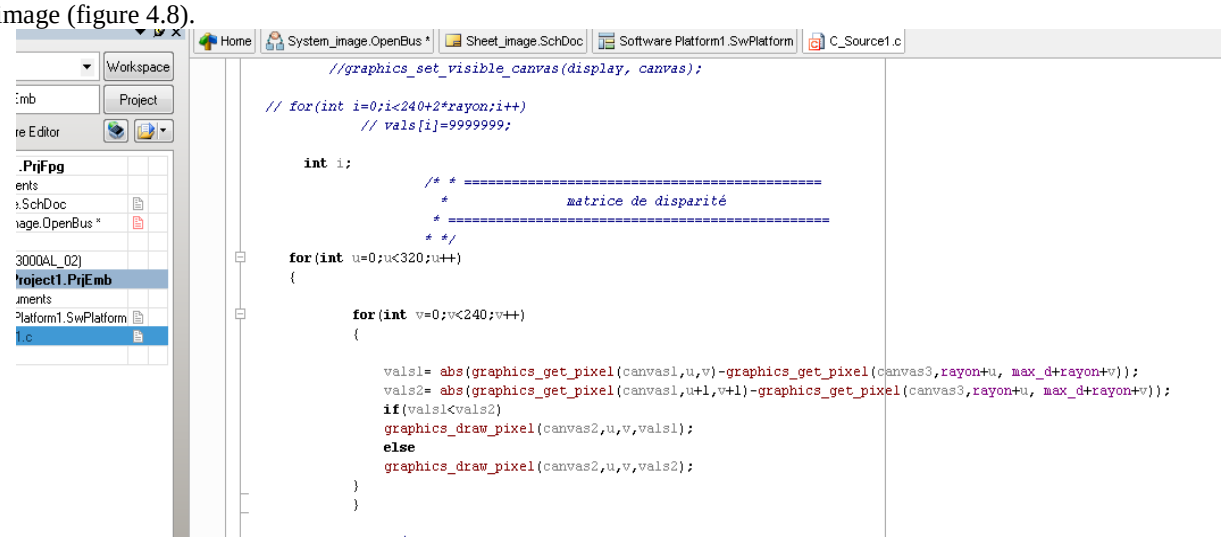


Figure 4.8 : L'environnement de langage C

III. Résultats et interprétations

III.1. soustraction de deux images

Notre programme de stéréoscopie se base sur la soustraction de deux fenêtres, on commence en premier temps par un programme qui fait la soustraction de deux images (figure 4.9).

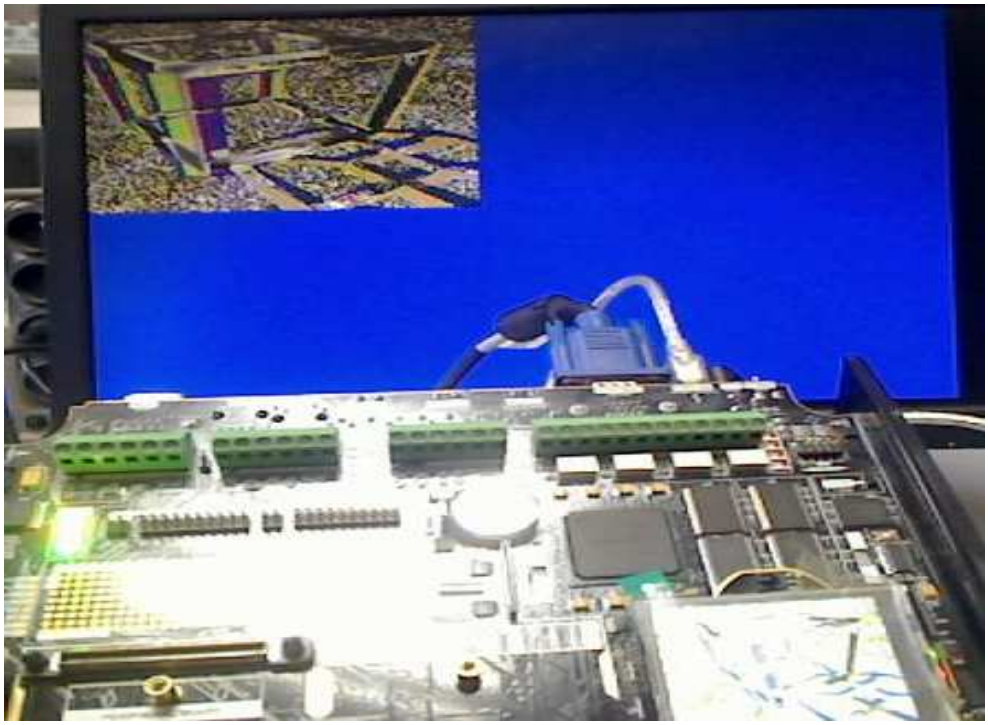
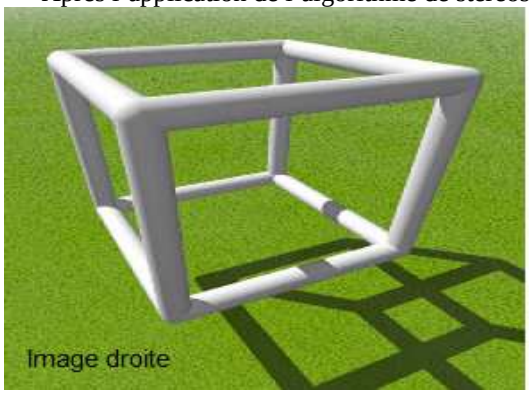
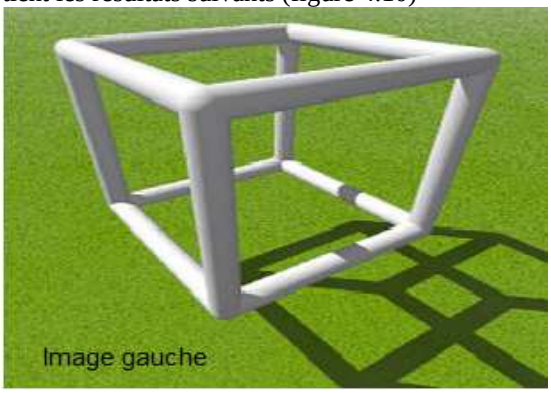


Figure 4.9 : Soustraction de deux images

III.2. image de reconstruction 3D

Après l'application de l'algorithme de stéréoscopie, on obtient les résultats suivants (figure 4.10)





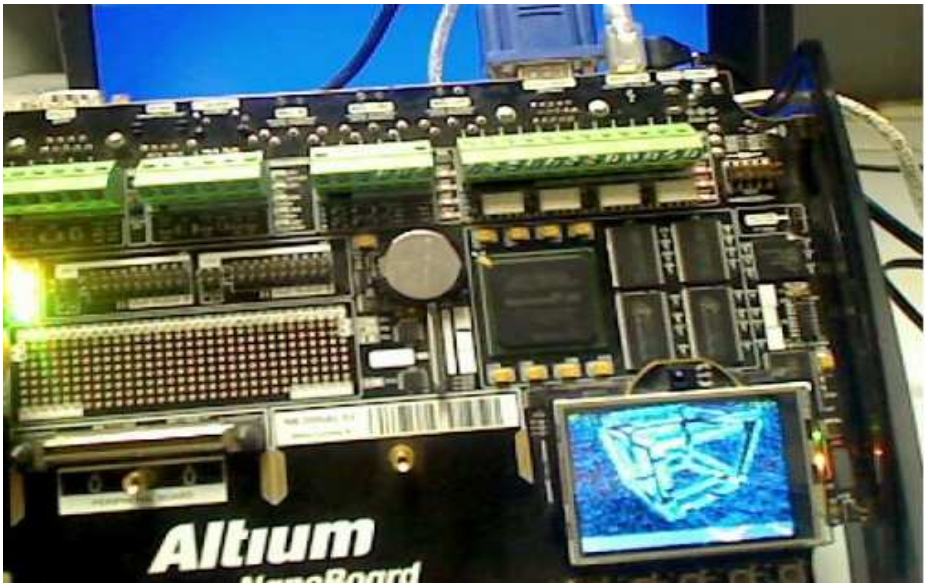


Figure 4.10 : Image de reconstruction 3D

III.3. Résultat

L'objectif de ce projet est d'implémenter un algorithme de stéréoscopie sur FPGA. Une des contraintes que nous avons rencontrées, est le temps d'exécution de l'algorithme puisqu'on plusieurs itérations. Pour résoudre ce problème et afin d'optimiser la mémoire nous avons proposé une solution qui nous a permis de diviser l'image sur quatre parties de tel façon à jouer sur un bloc de pixels et pas seulement sur chaque pixel. La solution proposée est de convertir ce programme de C en langage VHDL, pour ne pas faire le calcul dans le processeur.

Conclusion

Dans ce chapitre nous avons cité les différentes démarches pour implémenter notre algorithme sur la carte NanoBoard3000.

Conclusion générale

Ce stage a été réalisé au sein du Laboratoire Signaux Systèmes et Composants (L.S.S.C) du département génie électrique à la Faculté des Sciences et Techniques de Fès**.** Les objectifs de ce stage étaient doubles : dans un premier temps réaliser la conception d'un programme de vision 3D pouvant servir de modèle et réaliser la caractérisation de ce modèle et dans un second temps étudier et implémenter un certain nombre d'algorithmes et de techniques d'appariement.

L'objectif final était d'étudier la possibilité de s'orienter vers une implémentation silicium d'un dispositif de vision 3D stéréoscopique. La vision stéréoscopique, technique simple de vision qui rejoint par le principe la vision humaine possède en effet un large champ d'applications potentielles, du jeu vidéo à l'aide à la conduite en passant par la robotique et la domotique. Ce champ est actuellement limité par le coût élevé des dispositifs.

L'étude de faisabilité des algorithmes choisis était une phase transitoire vers la conception et l'implémentation sur la NanoBoard3000.

La réalisation et l'implémentation des fonctions de base de stéréoscopie sur une image, était réussie malgré que nous avons encore des améliorations à faire au niveau optionnel. Vue la contrainte de temps.

Finalement, ce stage présente une grande expérience pour ma future carrière, et nous espérons que notre travail témoigne de notre bonne volonté et que les membres de jury trouvent l'expression du grand honneur qu'ils nous font en acceptant de l'évaluer.

Annexes

Annexe 1: lexique

Fenêtre de Corrélation: La fenêtre de corrélation est utilisée par les mesures de similitudes. Il s'agit du voisinage sur lequel va être déterminée la ressemblance ou le non ressemblance entre deux zones. Cette fenêtre est carrée et peut être de taille variable ou fixe.

Disparité: La disparité se définit comme la différence des abscisses de deux point correspondants. L'un appartenant à l'image gauche et l'autre à l'image droite la disparité représente la distance entre les deux points dans l'image. La disparité est inversement proportionnelle à la profondeur.

Épi-polaire (géométrie): la géométrie épi-polaire est la géométrie qui décrit la conformation d'un capteur stéréoscopique. On distingue particulièrement dans cette géométrie les épi-pôles et les lignes épi-polaires.

Ligne épi-polaire : Une ligne épi-polaire est définie grâce à la matrice fondamentale. Elle est le fruit de la contrainte épi-polaire qui dit que tout point d'une image trouve sont correspondant sur une ligne dans l'autre image. Cette ligne se nomme la ligne épi-polaire et passe par l'épi-pôle.

Épi-pôle: Toutes les lignes épi-polaires d'une image sont concourantes en un point. Ce point est l'épi-pôle de cette image.

Rectification: La rectification est une étape du processus de vision 3D qui vise à simplifier la contrainte épi-polaire en rendant les lignes épipolaires parallèles : après la **Rectification :** des images la recherche de correspondant se réduira à une simple recherche sur la ligne correspondante dans l'autre image au lieu de prendre place sur la ligne épi-polaire.

Centre de projection: Le centre de projection est le centre du repère caméra.

Point principal: Le point principal est le centre du repère image, il est généralement situé au point milieu de l'image.

Corrélation: La corrélation est l'action de mettre en correspondance les points similaires.

On forme des paires de points. Les outils pour la corrélation sont les mesures de similitude et l'algorithme d'appariement.

Les systèmes embarqués : sont des systèmes électroniques qui exécutent un nombre limité de tâches fixes, contrairement aux ordinateurs personnels ou station de travail, qui eux sont d'usage général

IP: (Intelectual Properties) Les IP sont des descriptions de fonctions à intégrer dans un FPGA. Cela permet à un développeur un gain de temps considérable, en implantant le code des fonctions IP déjà écrites.

Open Bus: C'est un schéma fonctionnel mettant en œuvre des IP sous ALTIUM

SoC : Système on Chip. C'est le concept d'intégrer une fonction électronique dans un composant programmable.

CAO: Conception assistée par ordinateur

Annexe 2 : code C de l'algorithme de stéréoscopie

```
void stereoCC(PPMImage *imageL, PPMImage *imageR)
{
    FILE *f,*f2;
    float x;
    int dp=0;
    int rayon=3;
    int max_d=20;
    int rows,cols;

     // Rayon du voisinage (pour calculer la similarité (dist/corr) entre 2 points)
     // Un rayon de 2, par exemple, donne une fenêtre 5x5
     //  Taille de l'image et de la fenêtre
    rows=imageL->y;
    cols=imageL->x;
    unsigned char **corrL,**corrR;
    corrL = new unsigned char*[rows];
    corrR = new unsigned char*[rows];
    for(int i=0;i<rows+2*rayon;i++)
    {
          corrL[i] = new unsigned char[cols+2*rayon];
          corrR[i] = new unsigned char[cols+2*rayon];
    }
     // on charge les centres des deux fenetres imL et imR respectivement par les valeurs de l'image gauche te l'image droite
    for(int i=0;i<rows;i++)
    {
          for(int j=0;j<cols;j++)
          {
                x=0.2989 * imageL->data[i*cols+j].red + 0.587 * imageL->data[i*cols+j].green + 0.1140 * imageL->data[i*cols+j].blue;
                //if(x-(unsigned char)x>0.5)
                      corrL[i][j]=(unsigned char)(x) ;
                // else
```

```
                    //      corrL[i][j]=(unsigned char)(x);

                x=0.2989  *imageR->data[i*cols+j].red  +  0.587  *imageR->data[i*cols+j].green  +  0.1140
*imageR->data[i*cols+j].blue;
                // if(x-(unsigned char)x>0.5)
                        corrR[i][j]=(unsigned char)(x) ;
                // else
                    //      corrR[i][j]=(unsigned char)(x);

        }
    }

     // on prend deux fenetres imL et imR de meme taille [rows+(2*rayon)]*[cols+(2*rayon)]
    unsigned char **imL,**imR;
    unsigned char **disparity_map;

    imL = new unsigned char*[rows+2*rayon];
    imR = new unsigned char*[rows+2*rayon];

    disparity_map = new unsigned char*[rows+2*rayon];
    for(int i=0;i<rows+2*rayon;i++)
    {
        imL[i] = new unsigned char[cols+2*rayon];
        imR[i] = new unsigned char[cols+2*rayon];
        disparity_map[i] = new unsigned char[cols+2*rayon];
    }
    for(int i=0;i<rows+2*rayon;i++)
    {
        for(int j=0;j<cols+2*rayon;j++)
        {
            if(i>=rayon&&i<rayon+rows && j>=rayon&&j<rayon+cols)
            {
                imR[i][j]=corrR[i-rayon][j-rayon];
                imL[i][j]=corrL[i-rayon][j-rayon];
            }
            else
            {
                imR[i][j]=0;
                imL[i][j]=0;
            }
            disparity_map[i][j]=0;
        }
    }

  int k=0;

    unsigned char **WindowsR,**WindowsL;
    WindowsR= new unsigned char*[2*rayon+1];
    WindowsL= new unsigned char*[2*rayon+1];

    unsigned long *vals;
    unsigned long min;
    vals = new unsigned long[cols+2*rayon];

    for(int i=0;i<cols+2*rayon;i++)
        vals[i]=9999999;
```

Annexe 3: code de l'algorithme de stéréoscopie en C embarqué

```
#include <string.h>
#include <graphics.h>
#include <canvas.h>
#include "generic_devices.h"
#include "devices.h"
#include <drv_ioport.h>
#include <drv_led.h>
#include <devices.h>
#include "led_info.h"
#define rayon 3
#define max_d 20
// memory

extern __no_sdata graphics_bitmap_t _lc_ub_image1_bmp[];
extern __no_sdata graphics_bitmap_t _lc_ue_image1_bmp[];

extern __no_sdata graphics_bitmap_t _lc_ub_image2_bmp[];
extern __no_sdata graphics_bitmap_t _lc_ue_image2_bmp[];

graphics_t * display, *display1;
canvas_t * canvas,*canvas1,*canvas2,*canvas3,*canvas4,*canvas5,*canvas6;
led_t * ptrLEDs;
ioport_t * IOPorts;
color_t vals1, vals2;
unsigned char min;

int i,j,k,l,dp;

bool * ptr_dip;
bool * SW(uint8_t x);
bool butt_sw[5];



void main(void)

{
    display = graphics_open(GRAPHICS_1);
    canvas = graphics_get_visible_canvas(display);
    display1 = graphics_open(GRAPHICS_2);
    canvas1 = graphics_get_visible_canvas(display1);
```

```
    graphics_fill_canvas(canvas1, BLUE);

    IOPorts = ioport_open(DRV_IOPORT_1);
    ptrLEDs = led_open(DRV_LED_1);
    led_set_all_on_intensity(ptrLEDs, 10);
    led_turn_all_off(ptrLEDs);

    canvas3 = canvas_extract(canvas, 0, 0, 320,240);
    canvas4 = canvas_extract(canvas1, 0, 0, 320,240);
    canvas2 = graphics_get_canvas(display, 1);
    canvas5 = graphics_get_canvas(display, 1);
    canvas6 = graphics_get_canvas(display, 1);
         while (1)
    {


            led_turn_all_off(ptrLEDs);


            ptr_dip = SW(ioport_get_value(IOPorts, PORTS));
            for (uint8_t i = 0; i < 5; i++)
            {
                butt_sw[i] = *ptr_dip;
                if (butt_sw[i] == 1)led_turn_on(ptrLEDs,4-i);

                ++ptr_dip;
            }

            // stereoscopie

                 if(butt_sw[0] == 1)
                {


          //graphics_set_visible_canvas(display, canvas);

  // for(int i=0;i<240+2*rayon;i++)
              // vals[i]=9999999;

       int i;
                        /* * ================================================
                          *                matrice de disparité
                         * ================================================
                        * */
     for(int u=0;u<320;u++)
     {

              for(int v=0;v<240;v++)
              {

                  vals1=                        abs(graphics_get_pixel(canvas1,u,v)-
graphics_get_pixel(canvas3,rayon+u, max_d+rayon+v));
                  vals2=                    abs(graphics_get_pixel(canvas1,u+1,v+1)-
graphics_get_pixel(canvas3,rayon+u, max_d+rayon+v));
                  if(vals1<vals2)
                  graphics_draw_pixel(canvas2,u,v,vals1);
                  else
                  graphics_draw_pixel(canvas2,u,v,vals2);
              }
              }
```

Annexe 4: Liste des IP prêtent à l'emploi livrés par Altium sous OpenBus

IP	Désignation
	Le TSK3000A est un processeur 32-bit, processeur RISC.Le TSK3000A, est unprocesseur "classique RISC" d'une architecture interne Harvard, il dispose d'unestructure de mémoire qui simplifie grandement la gestion des interruptions. Le processeur simplifie également la connexion de périphériques avec un support pour le bus Wishbone.
	C'est une interface de communication série qui fournit une interface SPI Master,permettant à un processeur hôte la communication efficace avec un dispositif SPI esclave - résident à l'extérieur du dispositif physique FPGA sur laquelle la conception est ciblée.
	Le contrôleur VGA32_ILI9023 fournit une interface simple de 32-bit entre un processeur hôte et d'un écran TFT (Thin Film Transistor) panneau LCD. Le contrôleur récupère les données au format 16bpp de la mémoire vidéo externe, et l'affiche sur le panneau TFT LCD connecté.
	Le composant contrôleur de mémoire (WB_MEM_CTRL) fournit une interface simple et configurable, entre un processeur 32-bit, et différents types de mémoires.
	La composante Arbiter fournit un moyen simple de partager un périphérique esclave entre plusieurs maîtres au sein du système OpenBus.
	La composante d'interconnexion constitue un moyen d'accéder à un ou plusieurs périphériques sur une seule interface OpenBus. Il se connecte directement aux ports d'E / S ou MEM de un des IP livré, pour faciliter la communication avec les périphériques ou des dispositifs de mémoire physique.

	Le contrôleur de mémoire partagée (WB_SHARED_MEM_CTRL) fournit une interface entre un processeur 32-bits et des états mémorisés sur un bus partagé. Le contrôleur permet d'accéder à, et l'utilisation de la suite de trois différents types de mémoire: - Asynchrone RAM statique - Single Data Rate Synchronous DRAM - Mémoire Flash

Bibliographie

[1] J.K.Aggarwal, N. Nandhakumar, "On the Computation of Motion from Sequences of Images –A Review ,"Proc.of the IEEE, V ol. 76 (8), pp. 917–935, Aug. 1988.

[2] A. Blake, A. Zissermann, "Visual Reconstruction," MIT–Press, Cambridge, 1987.

[3] D.Marr, K. H. Nishihara, "Representation and recognition of the spatial organization of threedimensional shapes," Proc. Royal. Soc. Lond. B, vol. 200, pp. 269 –294, 1978.

[4] H.G.Musmann, M. Hötter, J. Ostermann, "Object–oriented analysis–synthesis coding of movingimages,"SignalProcessing: Image Communication, V ol. 1 (2), pp. 117–138, Nov .

[5] G. Martínez, "A robust algorithm for 3D–motion estimation of a small surface patch of a 3D object", Workshop on Image Analysis and Synthesis in Image Coding, Berlin, 18–19 October 1994, pp. D1.1–D1.4.

[6] A.Fusiello, E.Trucco, A.Verri. A compact algorithm for rectification of stereo pairs. Machine Vision and Application 2000.

[7] O.Faugeras, B.Hotz, H.Mathieu, T.Viéville, Z.Zhang, P.Fua, E.Théron, L.Moll, G.Berry, J.Vuillemin, P.Bertin, and C.Proy. Real time correlation based stereo: algorithms, implementations and applications. Technical Report 2013, INRIA, 199

[8] C/C++ Langage Reference.

[9] Yehu Shen, "Efficient normalized cross correlation calculation method for stereo vision based robot navigation",Frontiers of Computer Science in China, June2011, Volume 5, Issue 2, pp 227-235.

[10] Banks, Jasmine, Porter, Reid, Bennamoun, Mohammed, & Corke, Peter (1997) "A generic implementation framework for stereo matching algorithms", In Chaplin, Bob I. & Page, Wyatt H. (Eds.) DICTA'97 and IVCNZ'97 Conference Proceedings, The Department of Production Technology, Massey University, Auckland, New Zealand, pp. 29-34.

[11] Kun Wang. Adaptive Stereo Matching Algorithm Based on Edge Detection.

[12] Nalpantidis Lazaros, Georgios Christou Sirakoulis and Antonios Gasteratos, "REVIEW OF STEREO VISION ALGORITHMS: FROM SOFTWARE TO HARDWARE", International Journal of Optomechatronics, 2: 435–462, 2008.

[13] TSK3000A 32-bit RISC Processor. [14] Using the TSK3000 Embedded Tools

(1): www.fst-usmba.ac.ma
(2): www.eduscol.education.fr
(3): www.altium.com