Université Lille1
Sciences et Technologies

Université Lille 1

Affichage d'une image numérique sur une géométrie physique

Rapport technique

soutenu le 20 février 2014

par

Alexis Linke, Jonathan Mathieu et Bruno Ordozgoiti

Encadrants : Laurent Grisoni
François Cabestaing

Table des matières

Introduction

Ce projet est inspiré du projet Illumiroom de Microsoft qui permet d'étendre l'affichage autour d'un écran. En effet, un projecteur placé devant l'utilisateur projette autour de l'écran l'environnement qui se trouve en dehors des limites de l'écran.

C'est donc sur ce principe qu'est basé notre projet. Nous souhaitions apporter une nouvelle dimension au projet de Microsoft et permettre à un utilisateur de projeter une image sur un support sans avoir à ce se soucier de celui-ci, de l'emplacement du projecteur ou de l'emplacement de l'observateur. Cette technologie permettrait alors de faire une présentation avec un projecteur non fixé donc de pouvoir placer ce dernier là où on le souhaite. D'autre part, certaines configurations de pièces ne permettent pas de projeter une image de façon correcte. Par exemple, dans les pièces exigües il peut être plus judicieux d'avoir un affichage dans un coin de pièce ou au contraire mettre le projecteur dans un coin pour gagner en recul et afficher sur un seul pan de mur. Dans les greniers aménagés, par exemple, il serait possible de projeter à la fois sur le mur et le toit sans être gêner par la déformation.

Afin de rendre ces exemples possible, nous avons tenté de corriger les déformations dues au support avec seulement deux plans connus, blanc et de type lambertien, le point de vue était connu et fixe. Cette étape a été légèrement modifiée par rapport aux ambitions de base. En effet, l'objectif original était de projeter sur un nombre inconnu de plans.

Ensuite, nous avons essayé de projeter l'image tout en prenant en compte le positionnement de l'observateur à l'aide d'une caméra. L'affichage dynamique de la correction est complexe et demande un temps de calcul très court pour ne pas gêner la visualisation.

Pour finir nous souhaitions rendre notre projet "conscient de son environnement", lui permettant d'adapter l'affichage en fonction de celui-ci. Cette étape n'a pu être traitée car très complexe et par conséquent très chronophage. Cependant, nous avons essayé d'approcher cette idée en expérimentant certaines techniques.

Calibration

Calibration inter-caméra

Plusieurs aspects envisagés en préambule du projet ont mis en évidence la nécessité d'une calibration inter-caméras (RGB et infrarouge) du capteur de mouvement ASUS Xtion PRO LIVE (*cf.* Fig.1) utilisé. Plusieurs méthodes ont été étudiées comme la calibration forte, la calibration faible et la calibration jointe [1]. Les outils proposés par OpenNI [2] ont cependant permis de répondre à cet impératif plus rapidement. Conaissant les caractéristiques des caméras, les méthodes telles que *GetAlternativeViewPointCap* et *SetViewPoint* à disposition permettent de changer le point de vue à partir duquel la scène doit être capturée. De cette manière, donner à la caméra infrarouge le point de vue de la caméra RGB (ou inversement) connu compense les décalages constatés lors des premières manipulations. Les soucis liés à la précision des relevés effectués par les caméras sont alors moindres et tolérables dans le cadre des nos experimentations. Néanmoins, les méthodes que nous avions étudiées restent fiable pour une système de caméras stéréoscopique indépendantes.

FIGURE 1 – ASUS Xtion PRO LIVE.

Calibration relative à la surface de projection

L'ASUS Xtion PRO LIVE a été placé dans la scène de manière à visualiser entièrement le support de projection mais également à pouvoir suivre les déplacements de l'utilisateur, soit en face de la surface d'affichage et une distance permettant à l'utilisateur d'évoluer dans la scène sans quitter le champ de vision de la caméra. Cette distance a été détérminée manuellement en

évaluant la distance à partir de laquelle l'utilisateur est trop loin pour être suivi correctement. Nous avons alors cherché à calibrer la caméra dans la scène dans le but de déterminer sa position relative au support de projection pour notamment évaluer précisément la position de l'utilisateur. Pour cela nous nous sommes servi d'une mire (*cf.* Fig.2) que nous avons plaqué sur chacun des plans de notre support dans le but, dans un premier temps, de déterminer les paramètres intrinsèques et extrinsèques de la caméra RGB à l'aide des fonctions de la librairie OpenCV [3]. Les méthodes *findChessboardCorners* (permettant de trouver la mire dans une photo prise par la caméra, et plus particulièrement les coins de celle-ci) et *calibrateCamera* (déduisant les paramètres de la caméra à partir de plusieurs images de mire sous différents angles et position) nous ont permis de trouver ces paramètres. Cependant ces résultats se sont avérés incorrects du fait que les mires figurants sur le plan parallèle au sol n'étaient pas détectées et donc pas prises en compte lors de la calibration. Nous avons donc préféré la méthode manuelle consistant à prendre les mesures nécessaires afin de bien déterminer la position de la caméra, c'est-à-dire sa distance en profondeur (z) par rapport à l'origine (le coin) du support de projection et sa hauteur (y) par rapport au sol. La troisième coordonnées (x), perpendiculaire à la profondeur, s'obtient fidèlement en plaçant le capteur axactement en face du support. Cette solution a été préférée de manière à s'assurer de la crédibilité des données transmise à la caméra, mais dépend essentiellement de la disposition de la scène.

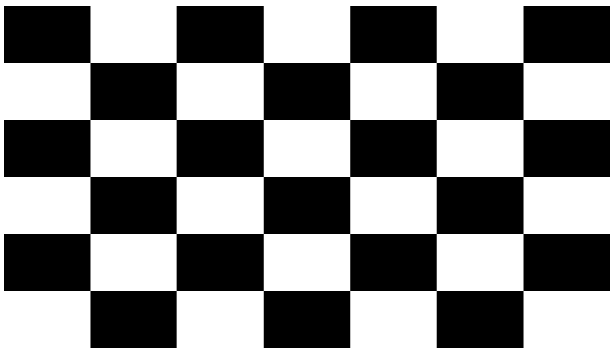


FIGURE 2 – Mire de calibration.

User Tracking

Afin d'approximer de façon efficace le point de vue de l'utilisateur à tout instant, la solution retenue et mise en place lors de l'élaboration de ce projet est de suivre la position de sa tête dans le scène par rapport à l'origine de la scène. Les méthodes implémentés par OpenNI [2], nécessitant une position de calibration (*cf.* Fig.3) , permettent de détérminer puis de suivre le squelette d'un utilisateur. Il est alors possible, en se servant des constantes prédéfinies (*SKEL_HEAD*), de déduire la position de la tête. Les coordonnées nous sont données dans le repère local de la caméra. A l'aide de la calibration on détérmine alors la position de la tête de l'utilisateur, cette fois par rapport à l'origine de la scène. Il faut donc inverser la coordonnée en x et augmenter la coordonnée en y en fonction de la position en hauteur du capteur. Pour la coordonnée de profonduer z, on se sert de la position du support de projection selon cet axe pour en déduire la position de l'utilisateur. De cette manière, on obtient une estimation fidèle du point de vue de l'utilisateur dans le repère sur support.

FIGURE 3 – Pose de calibration pour le tracking de l'utilisateur.

Correction des distorsions

La projection d'une image sur deux plans implique des déformations dues à plusieurs facteurs le premier étant l'orientation de ces plans et la position de l'utilisateur.

Afin de corriger ces déformations, nous avons utilisé le principe de l'homographie. Cet outil mathématique traduit la transformation géométrique nécessaire pour passer d'un plan A à un plan B. L'idée était donc d'utiliser l'homographie qui permet de passer du plan de l'image source à celui de l'image projetée. OpenCV [3] une méthode (*findHomography*) permettant de trouver l'Homographie entre deux plans. Ces plans doivent alors être définit (*cf.* Fig.4).

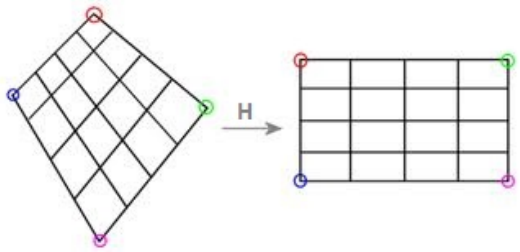


FIGURE 4 – Représentation d'une transformation homographique.

Pour cela, on commence par définir un plan d'observation qui contiendra ce que voit l'utilisateur. Ce plan est centré par rapport au centre de gravité 3D des 6 coins de l'image créés par la projection et la normale en son centre est orientée vers la tête de l'utilisateur (*cf.* Fig.5).

On récupère ensuite les coordonnées des 6 coins de l'image projetée en les mesurant. Ensuite, pour chacun des coins, on calcule la droite passant par la tête de l'utilisateur et le coin puis on conserve les points d'intersections de ces droites avec le plan d'observation (*cf.* Fig.6).

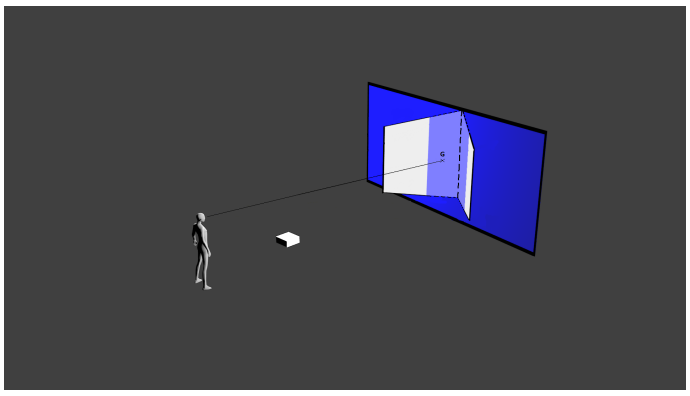


FIGURE 5 – Plan d'observation (en bleu).

On obtient donc 6 points sur le plan d'observation, ces 6 points nous permettent de dé-

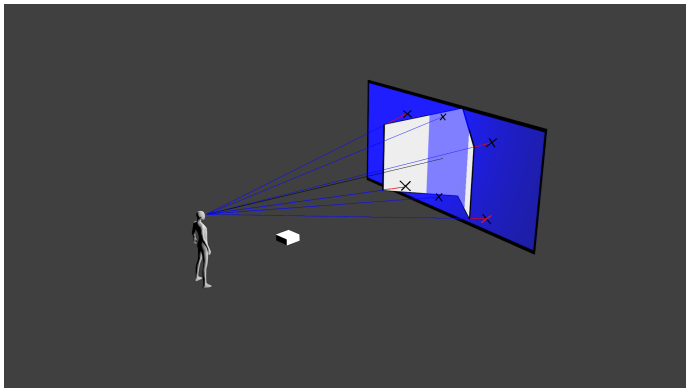


FIGURE 6 – Points d'intersection du plan d'observation.

finir deux déformations adjacentes. L'origine
des coordonnées est le centre de gravité 2D des coins projetés sur le plan d'observation (*cf.* Fig.7).

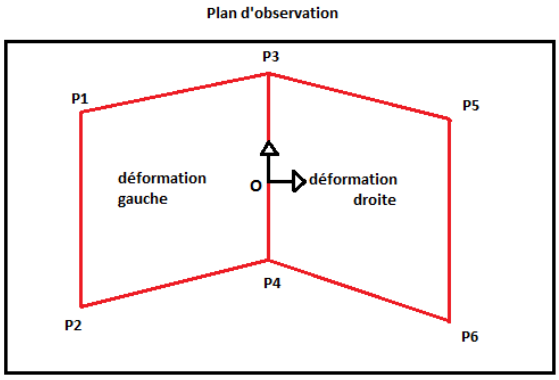


FIGURE 7 – Représentation de la projection vue par l'utilisateur.

Nous traitons tout d'abord la partie gauche de l'image.

Nous avons donc désormais un plan source et un plan destination, il ne reste plus qu'à utiliser la méthode *findHomography* afin de calculer l'homographie entre les deux plans puis à appliquer l'inverse de cette homographie à la moitié de l'image source à l'aide de la méthode *warpPerspectiveTransform* (*cf.* Fig.8).

L'image résultante de cette méthode est bien l'image déformée à projeter sur le support pour contrer la déformation. Seulement l'image se trouve être décalée et sort même parfois de la matrice de l'image. Le principal problème est que dans les informations hors de la matrice se trouvent être perdues. En effet, dans une matrice il est impossible d'accéder à une valeur dont au moins un indice est négatif (*cf.* Fig.9).

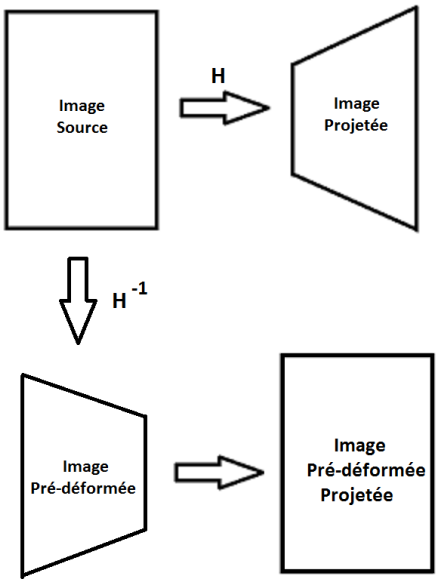


FIGURE 8 – Relations entre les différentes images.

Pour remédier à ce problème nous avons cherché une solution permettant de trouver la translation à appliquer pour replacer l'image dans la matrice.

Pour cela nous avons créé une Bounding Box autour de l'image. Cette Bounding Box est définie par ses coins, eux-mêmes définis en appliquant l'homographie aux coins de l'image sources à l'aide de la méthode *perspective-Transform* et en créant une boite englobant ces quatre coins (*cf.* Fig.10).

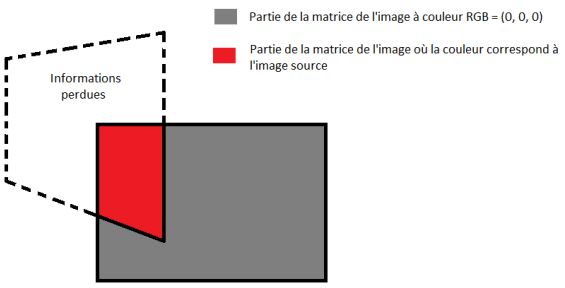


FIGURE 9 – Représentation des informations perdues.

Une fois les coins de la Bounding Box calculés nous pouvons déduire la translation à appliquer pour que l'image soit dans la matrice. Cette translation est la différence entre les coordonnées du coin supérieur gauche de la Bounding Box et le point où l'on souhaite déplacer l'image (ici l'origine). On applique alors cette translation dans la matrice de l'homographie (*cf.* Fig.11).

Une fois la partie gauche traitée nous pouvons travailler sur la partie droite.

Pour la partie droite nous calculons et appliquons l'homographie de la même manière que la partie gauche. Nous obtenons donc deux parties déformées mais celles-ci sont distinctes et

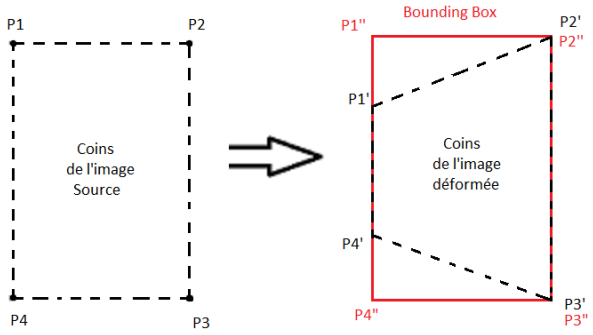


FIGURE 10 – Application de la Bounding Box.

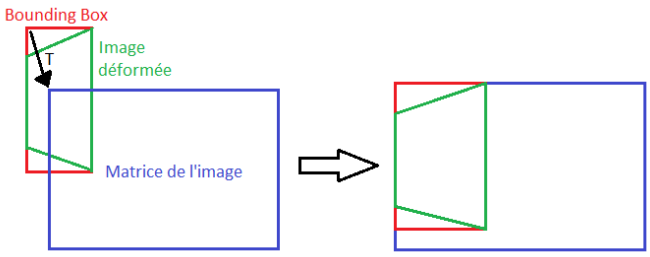


FIGURE 11 – Application de la translation.

doivent donc être jointes (*cf.* Fig.12).

Pour joindre les deux parties nous translatons la partie droite de manière à ce que le coin supérieur droit de la partie gauche et le coin supérieur gauche de la partie droite soient confondus. De plus, on vérifie que le coin le plus soit dans la matrice, en constatant le signe de la composante Y. Si ce signe est négatif c'est que le coin est hors de la matrice, il faut donc translater l'image afin qu'elle soit entièrement dans la matrice. Pour joindre les parties ensembles nous avons mis en place une méthode spécifique. Cette méthode ajoute un canal alpha aux deux parties. Ainsi, lors de la composition de l'image finale, on ne garde l'opacité des pixels qui appartiennent au plan transformé. Cela nous permet de superposer les images de façon efficace. Cette méthode est cependant très lente pour l'instant (chaque image prend autour de 0.5 secondes pour être calculée). Ceci nous empêche d'avoir un système qui fonctionne en temps réel. C'est pour cela que nous avons

FIGURE 12 – Parties gauche et droite déformées.

mis en place une méthode ad hoc qui n'ajoute pas ce canal alpha. Puisque on coupe l'image toujours au milieu, ça nous pose de léger problèmes lorsque le joint entre les images n'est pas parfaitement verticale.

Résultats

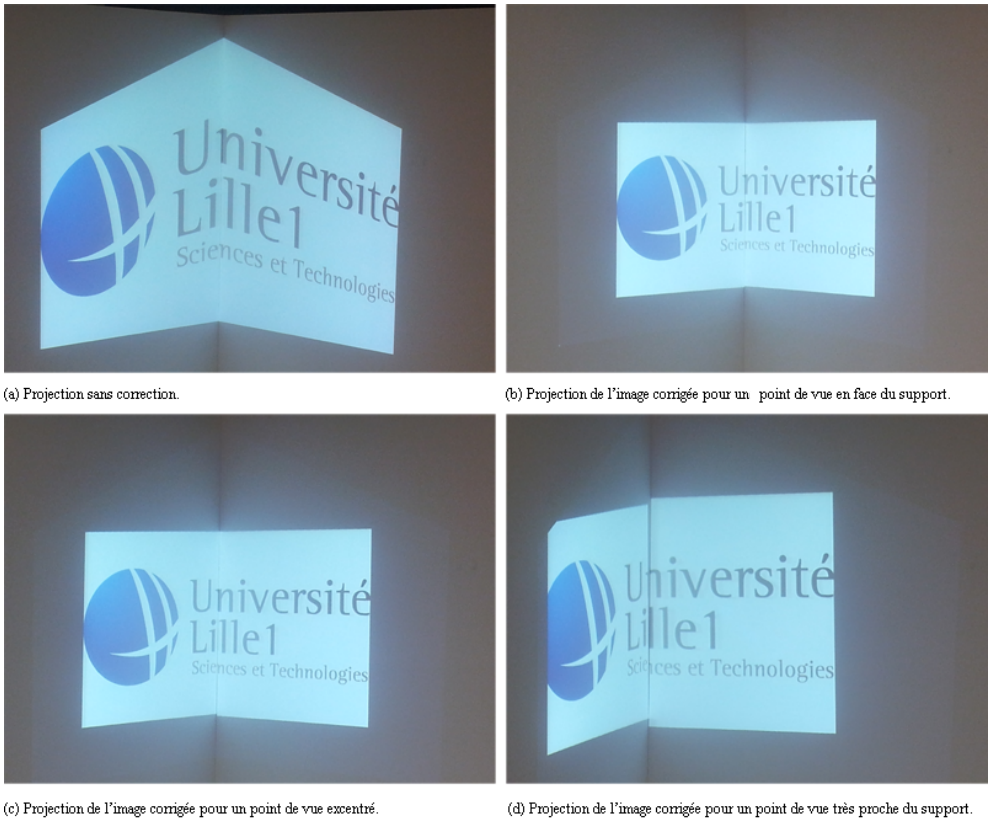
L'efficacité de notre méthode de correction des distorsions est globalement satisfaisante et confirme la viabilité des démarches scientifiques que nous avons développé. Les meilleurs résultats interviennent pour des utilisateurs positionnés raisonnablement dans la scène, ni trop près, ni trop excentré par rapport à la surface d'affichage (*cf.* Fig.13 (b) et (c)). Certaines pertes sont constatées lorsque ce n'est pas le cas (*cf.* Fig.13 (d)) et que le point de vue est trop rapproché en profondeur de l'origine de la scène avec un angle trop important entre l'axe optique de l'utilisateur et la normale du support de projection. Les transformations très importantes subient par l'image dans cette situation explique ces pertes, les pixels débordants des parties de l'image étant omis lors de la superposition. L'ajout du canal alpha peut compenser ces pertes mais n'a pas été ici implanté car coutant trop de temps de calcul et étant par cousequent un obstacle au suivi en temps réel de l'utilisateur. Par ailleurs, le degré d'imprécision finalement plus élevé que prévu des données transmisse par le capteur n'a pas entravé la qualité des corrections appliquées à l'image. En effet, la calibration manuelle agissant sur le suivi de l'utilisateur s'est avéré très limitée. La position en hauteur du centre de gravité de la tête varie considérablement en fonction de sa profondeur dans la scène, la position selon l'axe y de la tête augmentant plus elle est proche du capteur. Néanmois, dans le cadre de nos experimentations, ces aléas n'ont pas influencé nos résultats mais doivent être pris en compte dans l'optique d'une exploitation à plus grande échelle.

Travaux futurs

L'essentiel des travaux effectués lors de ce projet se sont concentrés sur les corrections des distorsions. Dans cette optique, certaines parties annexes au sujet n'ont pas été entièrement traitées. Cependant, le système a été développé de manière à pouvoir prendre en charge ces aspects lors de travaux additionnels.

Calibration automatique

Le programme tel qu'il a été implémenté est fonctionnel dans la scène que nous avons imposée, mais très sensible au moindre changement de disposition des outils utilisés. Ainsi, si le projecteur



(a) Projection sans correction.

(b) Projection de l'image corrigée pour un point de vue en face du support.

(c) Projection de l'image corrigée pour un point de vue excentré.

(d) Projection de l'image corrigée pour un point de vue très proche du support.

FIGURE 13 – Résultats.

ou le capteur n'est pas positionné correctement par rapport au support de projection, les résultats ne seront pas corrects. Dans ce cas, afin de palier à ce problème, l'application a été pensée de manière à pouvoir s'adapter à une autre configuration de la scène. Comme vu précédemment dans la partie concernant la correction des distorsions, les points 3D traduisant les plans desquelles les homographies à appliquer à l'image sont déduites ont été mesurées et fixés. En tenant compte de cela, on peut ajouter une interface permettant à l'utilisateur de sélectionner les points du polygone formé par le projecteur sur la surface d'affichage depuis une capture des caméras couleur et infrarouge (en exploitant donc la calibration inter-caméra développée plus haut). De cette manière il possible d'obtenir une définition semi-automatique des plans pour une scène quelconque, en faisant attention à ce que les caméras puissent voir l'intégralité du polygone.

Division de l'image et affichage sur plus de deux plans

Du fait des contraintes dû au temps de calcul, nous avons implémentés un système ad hoc pour l'affichage sur un support spécifique, compris de deux plans dont la droite commune est parfaitement verticale et centrée sur la surface de projection. Néanmoins, l'application telle que

nous l'avons conçue doit être capable de s'adapter à différentes scènes.

Dans les limites de notre dispositif dans lequel nous projetons sur deux plans, l'image non corrigée est divisée en deux parties égales correspondant aux deux homographies à appliquer. La projection étant centrée sur le support, l'image est divisée en son milieu par l'intermédiaire d'une fonction *divideImageInTwo(img)*. Cependant la projection peut ne pas être centrée. Dans ce cas, la fonction doit être adaptée afin de préciser la verticale à partir de laquelle l'image doit être divisée. Une fonction *divideImageInTwo(img,offset)* a donc été intégrée afin de répondre à ce problème (*cf.* Fig.14).

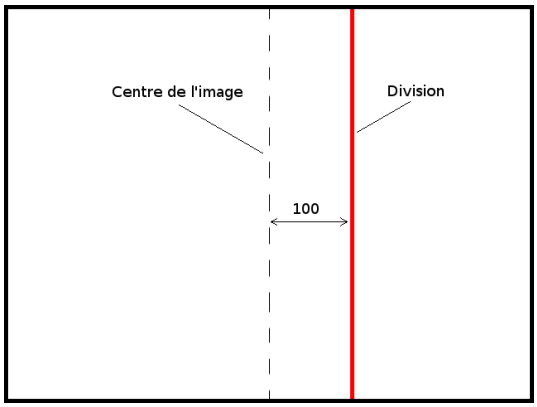


FIGURE 14 – Exemple d'application d'une division non centrée de l'image (offset = 100).

En considérant la liste des points définissant les différents plans sur lesquels l'image doit être projetée, cette fonction peut être utilisée en tenant compte des coordonnées des points ainsi que la taille de l'image. De la même manière, elle peut être adaptée afin de diviser l'image selon divers orientations. Ainsi il devrait être possible d'afficher l'image sur plus de deux plans.

Parties non traitées

Le cahier des charges que nous avions produit concernant ce sujet mettait en lumière certains aspects qui n'ont pas été traités lors de la réalisation du projet. C'est le cas de la modélisation 3D de la scène. Bien que convaincu de la réalisabilité des méthodes proposées tel que RANSAC [4], les difficultées rencontrées lors de l'utilisation d'une scène connue nous ont poussé à assurer la qualité de nos méthodes dans cette situation. Ainsi, cette partie a été laissée de côté mais devrait pouvoir malgré tout compléter efficacement notre système si elle est correctement réalisée. De la même manière, nous avions envisagés de prendre en charge les distortions liées à la luminosité, mais le support sur lequel nous avons travaillé présentant des plans blancs et lambertiens ne nous a pas donné l'occasion de nous confronter à ce problème. Ainsi, apporter ces corrections dans cette situation n'avait aucun interet. Par ailleurs, l'option visant à empêcher les pertes

d'informations dans l'image a également été négligée, le problème ne se présentant presque jamais durant nos experimentations.

Bien entendu, nous nous sommes permis de ne pas développer ces parties car elles n'étaient pas essentielles à la réalisation d'un système répondant avant tout au sujet.

Conclusion

L'objectif du projet était d'être capable de retranscrire correctement la projection d'une image sur une géométrie quelconque. Avec ce but en tête, nous sommes parvenus à appréhender et répondre correctement aux problèmes posés par ce sujet en complexifiant progressivement nos tests. En travaillant avec une surface propice à la validation de nos résultats, nous avons dans un premier temps pris soin d'affirmer nos méthodes permettant la correction de la perception de l'image pour un point de vue fixe et choisi. C'est une fois notre système fonctionnel dans ces conditions que nous l'avons étendu de manière à être capable de suivre le point de vue d'un utilisateur en temps réel. Si tous les objectifs envisagés lors de l'étude précédent ce projet n'ont pas été développés, nous avons pris soin de mettre en place notre dispositif de manière à pouvoir l'adapter et lui permettre de prendre en charge les améliorations que nous avons considérées. Des travaux complémentaires devraient donc pouvoir être apportés facilement.

Finalement ce projet nous a permis mettre en application les méthodologies étudiées à l'occasion d'un sujet concret et le travail en équipe nous a donné la possibilité de confronter nos idées et d'en tirer le meilleur afin d'atteindre les objectis que nous nous étions fixés.

Bibliographie

[1] Daniel Herrera C., Juho Kannala, and Janne Heikkila. Joint depth and color camera calibration with distortion correction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10) :2058–2064, 2012.

[2] OpenNI organization. *OpenNI User Guide*, November 2010. Last viewed 19-01-2011 11 :32.

[3] G. Bradski. *Dr. Dobb's Journal of Software Tools.*

[4] Patrick Quirk, Tyler Johnson, Rick Skarbez, Herman Towles, Florian Gyarfas, and Henry Fuchs. Ransac-assisted display model reconstruction for projective display. In *Proceedings of the IEEE Conference on Virtual Reality*, VR '06, pages 318–, Washington, DC, USA, 2006. IEEE Computer Society.