Université de Montréal

Reconstruction tridimensionnelle pour projection sur surfaces arbitraires

par

Louis Bouchard

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts des sciences
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.) en informatique

février, 2013

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé :

Reconstruction tridimensionnelle pour projection sur surfaces arbitraires

présenté par

Louis Bouchard

a été évalué par un jury composé des personnes suivantes:

Derek Nowrouzezahrai
(Professeur)

Gena Hahn
(Professeur)

Sébastien Roy
(Directeur)

Mémoire accepté le _____

RÉSUMÉ

Ce mémoire s'inscrit dans le domaine de la vision par ordinateur. Elle s'intéresse à la calibration de systèmes de caméras stéréoscopiques, à la mise en correspondance caméra-projecteur, à la reconstruction 3D, à l'alignement photométrique de projecteurs, au maillage de nuages de points, ainsi qu'au paramétrage de surfaces. Réalisé dans le cadre du projet LightTwist du laboratoire Vision3D, elle vise à permettre la projection sur grandes surfaces arbitraires à l'aide de plusieurs projecteurs. Ce genre de projection est souvent utilisé en arts technologiques, en théâtre et en projection architecturale.

Dans ce mémoire, on procède au calibrage des caméras, suivi d'une reconstruction 3D par morceaux basée sur une méthode active de mise en correspondance, la lumière non structurée. Après un alignement et un maillage automatisés, on dispose d'un modèle 3D complet de la surface de projection.

Ce mémoire introduit ensuite une nouvelle approche pour le paramétrage de modèles 3D basée sur le calcul efficace de distances géodésiques sur des maillages. L'usager n'a qu'à délimiter manuellement le contour de la zone de projection sur le modèle. Le paramétrage final est calculé en utilisant les distances obtenues pour chaque point du modèle. Jusqu'à maintenant, les méthodes existante ne permettaient pas de paramétrer des modèles ayant plus d'un million de points.

Mots clés: paramétrage de surfaces, multi-projection, stéréoscopie, calibration, reconstruction 3D, maillage, vision par ordinateur, alignement photométrique, lumière non structurée.

ABSTRACT

This thesis falls within the field of computer vision. It focuses on stereoscopic camera calibration, camera-projector matching, 3D reconstruction, projector blending, point cloud meshing, and surface parameterization. Conducted as part of the LightTwist project at the Vision3D laboratory, the work presented in this thesis aims to facilitate video projections on large surfaces of arbitrary shape using more than one projector. This type of projection is often seen in theater, digital arts, and architectural projections.

To this end, we begin with the calibration of the cameras, followed by a piecewise 3D reconstruction using an active unstructured light scanning method. An automated alignment and meshing of the partial reconstructions yields a complete 3D model of the projection surface.

This thesis then introduces a new approach for the parameterization of 3D models based on an efficient computation of geodesic distances across triangular meshes. The only input required from the user is the manual selection of the boudaries of the projection area on the model. The final parameterization is computed using the geodesic distances obtained for each of the model's vertices. Until now, existing methods did not permit the parameterization of models having a million vertices or more.

Keywords: surface parameterization, multi-projection, stereo, camera calibration, 3D reconstruction, meshing, computer vision, projector blending, unstructured light.

TABLE DES MATIÈRES

LISTE DES FIGURES

REMERCIEMENTS

Je tiens d'abord à exprimer ma profonde gratitude à mes parents, ainsi qu'à mon directeur, Sébastien Roy, pour leur soutien indéfectible et leur patience, sans lesquels ce mémoire n'aurait jamais vu le jour.

Je souhaite aussi remercier Vincent Chapdelaine-Couture et Nicolas Martin du laboratoire Vision3D, vos travaux ont rendu les miens possible.

Et finalement, merci à Jean Piché et Patrick Saint-Denis du laboratoire d'Électro-acoustique pour m'avoir prêté votre espace et vos bras pour mes expérimentations.

Chapitre 1

INTRODUCTION

Le travail présenté dans cette thèse s'inscrit dans le domaine de la vision par ordinateur. Plus spécifiquement, on va s'intéresser à des thèmes centraux du domaine comme la calibration de systèmes de caméras stéréoscopiques, la mise en correspondance caméra-projecteur, la reconstruction 3D et l'alignement photométrique de projecteurs. On va aussi toucher à des sujets de modélisation tridimensionnelle, soit le paramétrage de surfaces et le maillage de nuages de points.

1.1 Contexte

Ce mémoire a été réalisé dans le cadre du projet LightTwist [2]. LightTwist est un ensemble d'outils logiciels pour la réalisation d'installations visuelles multi-projecteur. Ces outils sont développés depuis 2004 par le Laboratoire Vision3D du Département d'Informatique et Recherche Opérationnelle de l'Université de Montréal. Avant les travaux effectués pour ce mémoire, LightTwist se limitait à des géométries de surfaces de projection spécifiques, soient des plans, cylindres ou dômes. La mise en correspondance se faisait à l'aide d'une seule caméra, placée à un point de vue unique. La reconstruction était faite dans le but d'optimiser la projection pour ce point de vue, considéré comme celui de l'observateur idéal. L'objectif de ce mémoire est d'étendre la méthode à des écrans de formes quelconques qui ne peuvent être vus entièrement d'un seul point de vue et pour lesquels le concept de l'observateur idéal ne s'applique plus.

1.2 Problème

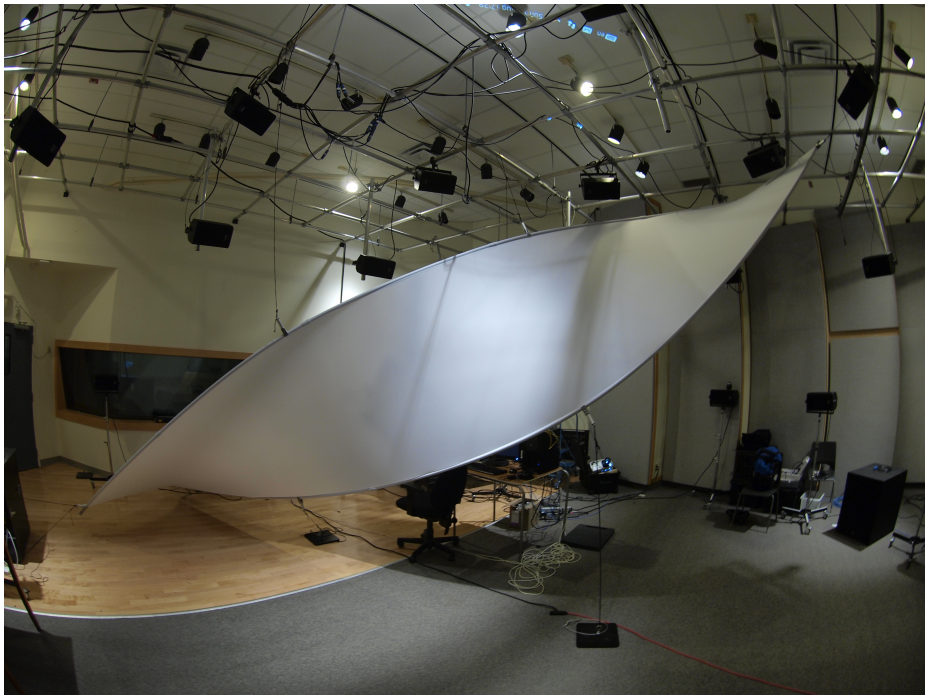
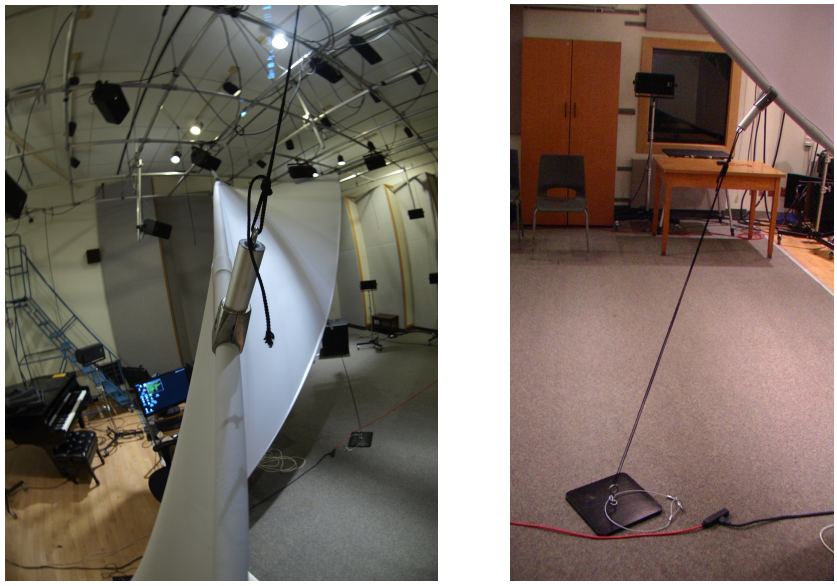


Figure 1.1. Écran sans projection

On veut ici projeter sur un écran en spandex conçu par Extension-Concept. L'écran, illustré à la figure 1.1 mesure 30 mètres par 1.5 mètres et est retenu par une armature tubulaire flexible insérée dans un ourlet le long du haut et du bas de l'écran. À l'aide de serres et de câbles, on peut déformer cet écran à notre guise, en faire un cylindre, voire même un ruban Möbius. La plupart des exemples présentés dans ce document sont faits sur des sections d'un quart de la longueur totale.

On peut considérer cet écran comme un ruban, ou un rectangle allongé qui est ensuite déformé mais sans en altérer la topologie outre mesure. Le but est de projeter des images et des vidéos sur cette surface de façon à donner l'impression que l'image est collée à ce ruban. Pour arriver à cette fin, on va utiliser plusieurs projecteurs positionnés de manière à ce que la surface de projection soit couverte complètement,



(a) (b)

Figure 1.2. Serres avec câbles tenseurs

avec une zone de chevauchement raisonnable entre chaque image de projecteur.

À l'aide d'un système calibré de caméras stéréoscopiques, on va mettre en correspondance les pixels du projecteur avec les pixels des deux caméras dans le but de trouver la position 3D de tous pixels de chaque projecteur, donc obtenir un modèle 3D de l'écran à partir duquel on pourra calculer des cartes de déformation pour chaque projecteur.

L'écran peut prendre des formes très complexes, ce qui pose plusieurs défis:

- Il est souvent impossible de placer une seule caméra (ou dans notre cas, un système de caméras stéréoscopiques) à un seul endroit où l'écran serait visible au complet et avec la surface de projection suffisamment perpendiculaire à la caméra. La caméra va devoir se déplacer afin d'observer l'écran en plusieurs parties. Ces multiples reconstructions devront êtres fusionnées de façon automatique.

- Comme la surface de l'écran couverte par un projecteur risque d'être reconstruite en deux morceaux ou plus, et comme ces reconstructions partielles on étés prises avec des points de vue différents, on ne peut plus utiliser simplement les correspondances dans l'image des caméras. On va devoir fusionner les différentes vues en 3D.

- Le paramétrage de la surface de projection, pose aussi un problème. Dans le système de projection LightTwist original, on optimisait la projection pour un point de vue privilégié, en l'occurrence la position de la caméra. Aussi, le système ne se prêtait qu'à des géométries d'écrans prédéterminées, soient des écrans planaires ou des cylindres plus ou moins parfaits. On pouvait également adapter une projection à une surface quelconque pour qu'elle ait une apparence planaire du point de vue de la caméra ayant servi à la reconstruction, et donc du spectateur. Le paramétrage de la surface était donc fait à partir d'une homographie pour une configuration planaire, ou d'un passage aux coordonnées polaires pour une projection cylindrique.

  Dans le cas présent, l'écran n'est plus destiné à être observé d'un point de vue optimal unique, et on ne peut plus modéliser le paramétrage simplement par une ou plusieurs homographies ou autres transformations géométriques ordinaires. On veut coller l'image à l'écran comme si l'écran lui-même était rectangulaire et planaire et ensuite déformé. Pour ce faire, on aura besoin du modèle 3D.

1.3 Méthodologie

La réalisation et la mise en oeuvre d'un système multi-projecteur tel que décrit dans cette thèse, doit se faire par étapes:

- **La calibration du système de caméras stéréoscopiques**. La calibration procède à l'aide de la méthode traditionnelle par damiers de OpenCV.

- **La mise en correspondance entre les pixels des projecteurs et des caméras**. On va déplacer les caméras à divers endroits jusqu'à ce qu'on ait couvert la totalité de l'écran. Pour chaque point de vue des caméras, on va envoyer des motifs au projecteurs qui sont visibles pour cette vue et obtenir des mises en correspondances partielles.

- **L'alignement photométrique**. Pour chaque point de vue des caméras, lorsqu'on observe un chevauchement entre deux projecteurs ou plus, on va calculer automatiquement un fondu entre les projecteurs.

- **Le raffinement de la calibration**. Cette étape optimise globalement la calibration du système complet à partir de la calibration initiale.

- **La triangulation des points en 3D pour les mises en correspondances partielles**. On obtient des modèles partiels sous forme de nuages de points pour toutes les prises de vue des caméras.

- **La fusion des nuages de point pour chaque projecteur**. Puisque l'écran est reconstruit par morceaux, cet étape est nécessaire pour obtenir un modèle 3D cohérent.

- **Le maillage du nuage de points complet**. Le modèle 3D doit être polygonal, il faut établir un maillage pour permettre la paramétrage de la surface.

- **Le paramétrage du modèle maillé**. Ce paramétrage est requis pour obtenir les cartes de déformation de chaque projecteur.

1.4 Contribution

Dans le domaine des arts technologiques, on observe un recours de plus en plus fréquent de la projection multi-projecteur sur des surfaces de plus en plus complexes.

Il devient donc important de simplifier et d'automatiser le processus d'alignement des projecteurs. Bien qu'il soit toujours possible de réaliser des projections complexes par un alignement manuel, cette opération reste en pratique trop longue et trop couteuse. De plus, l'expertise requise pour un tel aligement dépasse souvent largement les connaissances des artistes et techniciens de scène.

Plusieurs systèmes similaires ont étés proposés dans le passé pour automatiser l'alignement des projecteurs. Les travaux précurseurs de Raskar et al. démontrent les premiers véritables efforts pour réaliser la multi-projection sur des surfaces non-prédéterminées [14]. Ses travaux subséquents délaissent les surfaces arbitraires pour se concentrer des écrans planaires ou légèrement courbes [15], [13].

D'autres se sont attaqués au problème, mais toujours en restreignant la geométrie de la surface de projection à des plans [20] ou des portions de cylindres [16]. Les méthodes proposées pour des écrans non-planaires font presque toujours appel à une calibration paramètres internes et externes des projecteurs, ainsi qu'à des connaissances *a priori* sur la géométrie de la surface de projection [17], [10], deux contraintes que notre système évite. À notre connaissance, notre méthode est la seule à n'utiliser que la calibration des caméras et évite la calibration des projecteurs autrement plus difficile à réaliser [6].

Un aspect important justifiant la contributions de ce mémoire est la démocratisation de la technologie en augmentant l'accessibilité de telles projections à des artistes de talent plutôt qu'à ceux qui en ont les moyens.

On propose donc une méthode complète qui ne requiert qu'un minimum d'intervention de l'usager et qui s'applique à des géométries d'écran arbitraires. Plus spécifiquement, la technique de paramétrage de la surface à l'aide de distances géodésiques est une innovation. Elle rend possible le paramétrage efficace de maillages contenant des millions de points.

Chapitre 2

CALIBRATION

La calibration est une étape incontournable de tout système qui vise a obtenir une reconstruction 3D d'un objet à partir d'images.

En pratique, dans des problèmes de lumière structurée, on doit d'habitude faire la calibration d'un système caméra-projecteur. Si la calibration des paramètres internes d'une caméra est facilement réalisable, il n'en est pas de même pour un projecteur. Pour une caméra, on peut utiliser diverses méthodes avec des objets de calibration connus captés par celle-ci, et ainsi mettre en relation les points 2D de l'image de la caméra avec les points 3D du monde [19, chapitre 2]. Pour un projecteur, c'est plus complexe parce qu'il ne capte pas d'images, on doit mettre en relation les points du projecteur avec les points du monde par l'intermédiaire d'une caméra, ce qui est souvent peu pratique [6].

Dans notre cas, on souhaite se restreindre à la calibration de caméras seulement. C'est possible parce qu'on utilise toujours deux caméras pour faire la triangulation et on n'est pas contraint à modéliser les projecteurs eux-mêmes. On va utiliser une méthode par damiers pour calibrer les paramètres internes et externes du système rigide de caméras stéréoscopiques [19, section 6.3], [3].

En fait, la décision d'utiliser un système calibré de deux caméras plutôt qu'un système caméra-projecteur vient non seulement du fait qu'il est plus facile de calibrer des caméras que des projecteurs, mais aussi du fait qu'on veut laisser la possibilité d'utiliser des projecteurs qui ne sont pas calibrables, comme lorsqu'on utilise des miroirs convexes ou autre déformation de la projection.

De plus, l'utilisation de deux caméras permet d'obtenir des modèles 3D cohérents, c'est-à-dire qui ont toujours la même précision puisque la position relative des caméras

Figure 2.1. Système de caméras stéréoscopiques

(baseline) est constante. Dans le cas d'une reconstruction caméra-projecteur, la position relative change selon le projecteur utilisé et les reconstructions sont de précision variables. Si, par exemple, on a besoin de plus de détail à un endroit particulier, on peu approcher les deux caméras. Par contre, un projecteur est généralement fixe dans la scène et ne peut donc pas être déplacé.

2.1 Coordonnées projectives

Les points 2D et 3D sont généralement exprimés en coordonnées projectives (ou *homogènes*). Un vecteur dans un espace à n dimensions $\mathcal{R}^n$ peut être représenté dans l'espace projectif correspondant $\mathcal{P}^n$ par un vecteur de $n+1$ composantes. Par exemple, un vecteur 3D serait représenté dans $\mathcal{P}^3$ par $\mathbf{x}_h = (x, y, z, w)$. Ce même vecteur s'exprime dans $\mathcal{R}^3$ en divisant $\mathbf{x}_h$ par son dernier élément: $\mathbf{x} = (x/w, y/w, z/w)$. À l'inverse, on peut homogénéiser un vecteur en y ajoutant un 1: $\mathbf{x}_h = (x/w, y/w, z/w, 1)$. Deux vecteurs homogènes ne différant que par un facteur d'échelle sont donc considérés équivalents. Un point projectif dont la composante $w = 0$ est considéré comme étant à l'infini et ne peut être représenté dans $\mathcal{R}^3$.

On utilise les coordonnées projectives pour harmoniser la translation avec autres transformations affines. Dans un espace $\mathcal{R}^n$, si les transformations affines telles la rotation ou le cisaillement peuvent être représentées par une matrice $n \times n$, il n'est pas de même pour la translation, que l'on exprime par une addition de deux vecteurs. On ne peut donc combiner dans une même matrice . Le passage à l'espace projectif règle ce problème. On peut représentér une transformation affine 3D par une matrice 4×4:

$$\begin{bmatrix} & & & 0 \\ & \mathbf{A}_{3\times 3} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On peut aussi représenter une translation en 3D similairement:

$$\begin{bmatrix} x+t_x \\ y+t_y \\ z+t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Il est donc possible de combiner translations et transformations affines en enchaînant les multiplications de matrices.

Les coordonnées projectives permettent aussi de faciliter le passage du monde 3D vers une projection sur un plan 2D à l'aide d'une matrice 4×3:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2.2 Modèle de Caméra

Le modèle de caméra généralement utilisé en vision est celui de la caméra sténopé (ou *pinhole*) [19, chapitre 2]. Ce modèle décrit la relation entre un point 3D et sa projection sur un plan image. Toute la lumière captée par la caméra passe par un point unique. L'ouverture de la caméra *pinhole* étant 0, on la considère donc comme une caméra idéale dont la profondeur de champ est infinie. C'est un modèle simplifié mais suffisant pour nos besoins.

Considérons un point 3D dans le système de coordonnées de la caméra, p_w, et sa projection en un point 2D sur le plan image de la caméra, p_c, ces deux points en coordonnées projectives. La relation entre p_w et p_c est décrite par une matrice 3×4 $\mathbf{W}$, qui contient les paramètres internes de la caméra:

$$\mathbf{p_c} \propto \mathbf{W} \cdot \mathbf{p_w} \tag{2.1}$$

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \propto \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.2}$$

où f_x et f_y sont les distances focales exprimées en pixels, et (c_x, c_y) est le centre de la caméra, également en pixels. Pour une caméra dont les pixels du capteur sont carrés, f_x et f_y sont égaux.

Dans notre cas, on a deux caméras, l et r, avec chacune leur matrice interne, $\mathbf{W}_l$, $\mathbf{W}_r$. La transformation entre les systèmes de coordonnées des deux caméras est décrite par les matrices de rotation et translation, $\mathbf{R}_{lr}$ et $\mathbf{T}_{lr}$, de sorte que:

$$\begin{aligned} p_l &\propto \mathbf{W}_l \cdot p_w \\ p_r &\propto \mathbf{W}_r \mathbf{R}_{lr} \mathbf{T}_{lr} \cdot p_w \end{aligned}$$

où p_l et p_r sont les points 2D projectifs dans les images des caméras de gauche et de droite.

Distorsion radiale

Le modèle de caméra *pinhole* fait abstraction de toute notion de distorsions non linéaire dues à la lentille, et en particulier la plus commune, la distorsion radiale. Pour des caméras avec lentilles de très haute qualité, sans distorsion, c'est suffisant. En pratique, toute lentille amène au moins une légère distorsion radiale. Cette distorsion, si on ne la prend pas en compte, peut grandement affecter la calibration. Un système calibration qui tente de minimiser l'erreur de reprojection va tenter d'ajuster les paramètres internes pour compenser cette distorsion, ce qu'on veut éviter.

Selon le modèle *pinhole*, un point 3D projeté sur le plan image nous donne un point sans distorsion radiale (x_u, y_u). On veut connaître la position réelle (x_d, y_d) de ce point dans l'image de la caméra après avoir été affecté par la distorsion radiale de la lentille. On définit un point normalisé (x', y') et le facteur de distorsion k comme

$$\begin{aligned} x' &= \frac{x_u - c_x}{f_x} \\ y' &= \frac{y_u - c_y}{f_y} \\ k &= 1.0 + k_1 r^2 + k_2 r^4 \quad \text{où} \quad r^2 = x'^2 + y'^2 \end{aligned}$$

pour obtenir la position réelle:

$$\begin{aligned} x_d &= f_x k x' + c_x \\ y_d &= f_y k y' + c_y \end{aligned}$$

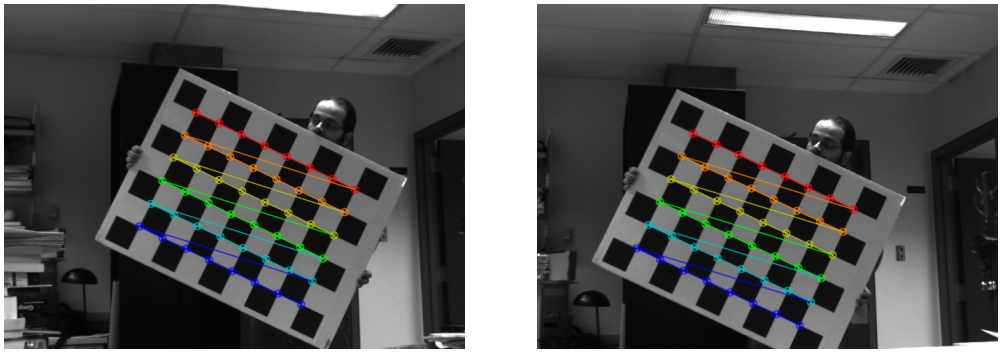
Ici, k_1 et k_2 sont les coefficient de distorsion radiale de premier et second ordre. Le modèle admet des coefficients d'ordre supérieur (jusqu'à k_6), ainsi que des paramètres de distorsion tangentielle, mais ils ne sont pas d'une grande utilité dans notre cas. Chaque caméra a donc son vecteur de coefficients de distorsion radiale $\mathbf{K}_l$ et $\mathbf{K}_r$, contenant les coefficients de premier et second ordre seulement.

En général, on cherche plutôt à enlever la distorsion radiale d'un point image. Comme l'équation n'est pas inversible, on doit utiliser une méthode itérative pour approximer la position du pixel non distorsionné via OpenCV.

2.3 Première calibration (phase 1 : calibration planaire)

Lorsqu'on ne dispose d'aucune information sur les caméra, on doit effectuer une première calibration pour obtenir une estimation préliminaire de la calibration. Les méthodes de calibration classiques font appel à un objet de calibration de dimensions connues. De tels objets permettent de faire le lien entre des coordonnées 3D réelles mesurées sur l'objet lui-même, avec les coordonnées en pixel des points d'intérêt de l'objet dans l'image de la caméra.

On utilise une méthode simple et rapide, mais imprécise, appelée calibration par plans [19, section 6.3.1]. L'objet de calibration en question dans cette méthode est un damier imprimé sur un plan. Cette calibration utilise des images de ce damier prises sous différents points de vue. Les coins du damier sont détectés automatiquement [3, chapitre 11] et on procède à une calibration dite *planaire* .



(a) (b)

Figure 2.2. Damiers observés en stéréo

L'imprécision de cette méthode provient du fait que la distorsion radiale de la lentille peut être interprétée par angle de vue légèrement plus grand ou plus petit. Aussi, les coins du damier peuvent être mal détectés, leurs coordonnées erronées dans l'image vont bien sûr affecter le résultat.

Impact de la qualité de la calibration sur les résultats

La précision d'une calibration peut se mesurer par l'erreur de reprojection. Après une première calibration, notre erreur est typiquement de 5 pixels. Ce résultat est obtenu en modélisant la distorsion radiale.

Puisque l'erreur de reprojection est déterminante pour la qualité des reconstructions 3D finales, il est important de procéder à une seconde calibration pour la réduire le plus possible. Après cette étape, l'erreur sera réduite à environ 0.1 pixels. On peut voir plus loin dans la figure 2.4 qu'une calibration avec erreur de reprojection minimale est cruciale pour être en mesure de bien aligner les nuages de points entre eux. On remarque que la calibration initiale est trop imprécise pour donner un résultat acceptable.

2.4 Seconde calibration (phase 2: calibration non-linéaire)

Comme la calibration planaire stéréo d'OpenCV peut être inexacte, on doit procéder au raffinement de la solution. On va prendre les résultats de deux mises en correspondances d'un même projecteur de deux points de vue différents.

Typiquement, on va prendre la première mise en correspondance qui est en général cadrée pour occuper tout le champ de vision des deux caméras, ainsi qu'une seconde mise en correspondance prise d'un point de vue différent mais qui englobe la même portion d'écran (voir figure 2.3). Cette seconde mise en correspondance ne servira pas dans la reconstruction modèle 3D de l'écran de projection, elle ne sert qu'au raffinement de la calibration. On va utiliser la calibration du système de caméras stéréoscopiques obtenue précédemment pour produire deux modèles 3D sous forme de nuages de points (voir figure 5.2, chapitre 5, p. 40). Comme ces deux mise en correspondances ont été faites pour le même projecteur, on a une bijection entre point des deux modèles. Si on aligne les deux modèle du mieux qu'on peut (least squares), la distance (l'erreur) entre chaque paire de points 3D nous donne une bonne idée de

la qualité de la calibration. C'est à partir de cette idée générale qu'on va procéder.

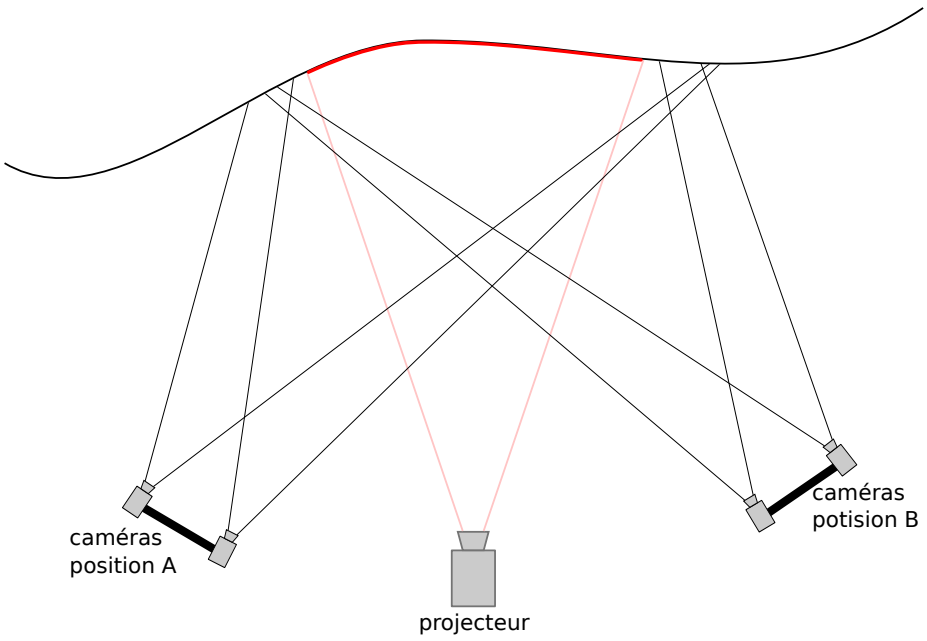


Figure 2.3. Disposition des caméras pour la seconde calibration.

On utilise ici l'algorithme de minimisation de fonction non-linéaire de Levenberg-Marquardt pour trouver une meilleure solution à la calibration [12, chapitre 10.2]. Cette méthode consiste à chercher dans l'espace des paramètres d'une fonction à partir d'une estimation initiale, jusqu'à ce que la somme des carrées des résultats de la fonction sur un ensemble de données arrive à un minimum local.

Les paramètres soumis à l'algorithme sont les variables de la calibration stéréo, soit les paramètres internes des deux caméras, leurs coefficients de distorsion radiale, ainsi que la rotation et translation entre les deux caméras:

- les deux matrices des paramètres internes $\mathbf{W}_l$, $\mathbf{W}_r$, constituées chacune des paramètres f_x, f_y, c_x et c_y. La forme de ces matrices suit celle de $\mathbf{W}$ aux l'équations 2.1 et 2.2.

- les deux vecteurs de coefficients de distorsion radiale $\mathbf{K}_l$, $\mathbf{K}_r$.

- la rotation et translation $\mathbf{R}_A^{lr}$ et $\mathbf{T}_A^{lr}$ entre les systèmes de coordonnées des caméras pour le premier point de vue A.

- la rotation et translation $\mathbf{R}_B^{lr}$ et $\mathbf{T}_B^{lr}$ entre les systèmes de coordonnées des caméras pour le second point de vue B. Ça peut sembler superflu, mais le montage de caméras stéréo n'est pas nécessairement aussi rigide qu'on le croit, et pourrait avoir bougé légèrement entre deux prises de vues.

- le rotations et translation $R_{B\to A}$ et $T_{B\to A}$ entre les points 3D reconstruits des deux points de vue A et B.

Les rotations $\mathbf{R}_A^{lr}$, $\mathbf{R}_B^{lr}$ et $R_{B\to A}$ sont exprimées en angles selon les trois axes, et de même pour les translations $\mathbf{T}_A^{lr}$, $\mathbf{T}_B^{lr}$ et $T_{B\to A}$, exprimés seulement en (x, y, z). Ça nous donne un espace de paramètres de 26 dimensions.

Soit un point d'une scène illuminé par une projecteur P, observé par deux caméras (L et R), successivement placées en deux points de vue différents (A et B). Ce point peut être triangulé par les caméras L et R indépendamment dans chaque point de vue, pour donner les points 3D $\mathbf{p}_A$ et $\mathbf{p}_B$. Ces points représentent un même point 3D de la scène, mais dans des systèmes différents qui sont reliés par $\mathbf{R}_{B\to A}$ et $\mathbf{T}_{B\to A}$. Lorsqu'il est visible de l'ensemble des caméras, ce point de la scène est dit *AB-visible* et définit une correspondance entre les deux vues A et B et du même coups permet d'estimer la relation entre ces vues ($\mathbf{R}_{B\to A}$ et $\mathbf{T}_{B\to A}$).

Les données utilisées pour la minimisation sont les deux ensembles de mises en correspondances prises de deux points de vues différents (A et B) pour chaque caméra (du point de vue du projecteur): $\mathbf{M}_i^{PL}$, $\mathbf{M}_i^{PR}$, $i = \{A, B\}$, où P est le projecteur, L la vue gauche, et R la vue droite. Seuls les points *AB-visibles* seront utilisés pour la minimisation.

Soit j un index dans l'ensemble des points *AB-visibles* du projecteur P. On a:

$$\begin{aligned}
\mathbf{p}_A(j) &= Triangule(\mathbf{M}_A^{PL}(j), \mathbf{M}_A^{PR}(j), \mathbf{W}_l, \mathbf{W}_r\mathbf{RT}_A^{lr}, \mathbf{K}_l, \mathbf{K}_r) \\
\mathbf{p}_B(j) &= Triangule(\mathbf{M}_B^{PL}(j), \mathbf{M}_B^{PR}(j), \mathbf{W}_l, \mathbf{W}_r\mathbf{RT}_B^{lr}, \mathbf{K}_l, \mathbf{K}_r) \\
\mathbf{p}_{AB}(j) &= TrianguleSD(\mathbf{M}_A^{PL}(j), \mathbf{M}_A^{PR}(j), \mathbf{W}_l, \mathbf{W}_r\mathbf{RT}_A^{lr}, \\
&\quad \mathbf{M}_B^{PL}(j), \mathbf{M}_B^{PR}(j), \mathbf{W}_l\mathbf{RT}_{B\to A}, \mathbf{W}_r\mathbf{RT}_B^{lr}\mathbf{RT}_{B\to A}, \\
&\quad \mathbf{K}_l, \mathbf{K}_r)
\end{aligned}$$

où

$$\begin{aligned}
\mathbf{RT}_A^{lr} &= \mathbf{R}_A^{lr}\mathbf{T}_A^{lr} \\
\mathbf{RT}_B^{lr} &= \mathbf{R}_B^{lr}\mathbf{T}_B^{lr} \\
\mathbf{RT}_{B\to A} &= \mathbf{R}_{B\to A}\mathbf{T}_{B\to A}
\end{aligned}$$

La fonction $Triangule(\mathbf{p}_1, \mathbf{p}_2, \mathbf{M}_1, \mathbf{M}_2, \mathbf{K}_1, \mathbf{K}_2)$ calcule par triangulation le point 3D qui se projette en $\mathbf{p}_1$ et $\mathbf{p}_2$ si on lui applique respectivement la matrice de projection $\mathbf{M}_1$ et $\mathbf{M}_2$ et la distorsion $\mathbf{K}_1$ et $\mathbf{K}_2$.

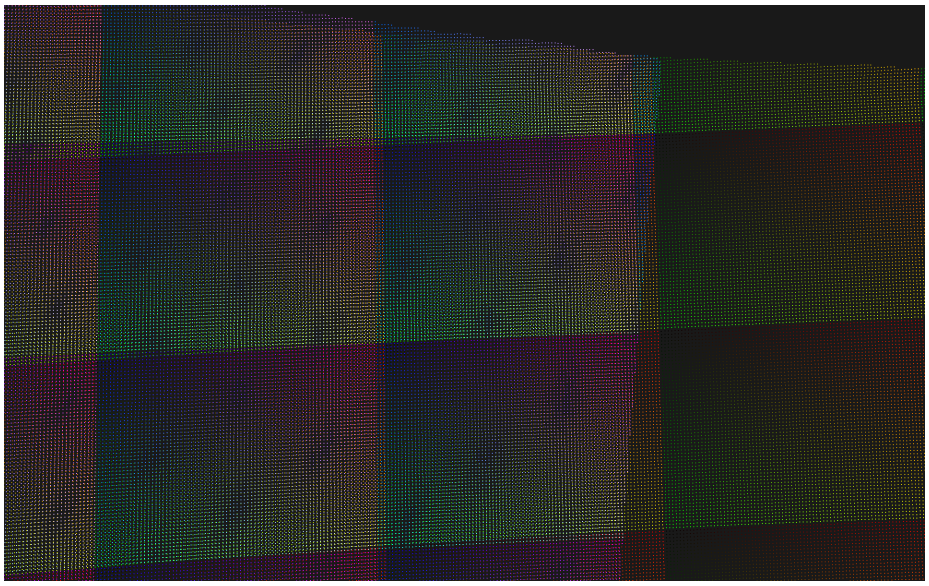
La fonction $TrianguleSD(\mathbf{p}_1, \mathbf{p}_2, \mathbf{M}_1, \mathbf{M}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{M}_3, \mathbf{M}_4, \mathbf{K}_1, \mathbf{K}_2)$ effectue une triangulation surdéterminée à partir de 4 correspondances de 4 caméras vers un même point d'un projecteur.

La fonction à minimiser est la norme carrée du vecteur suivant:

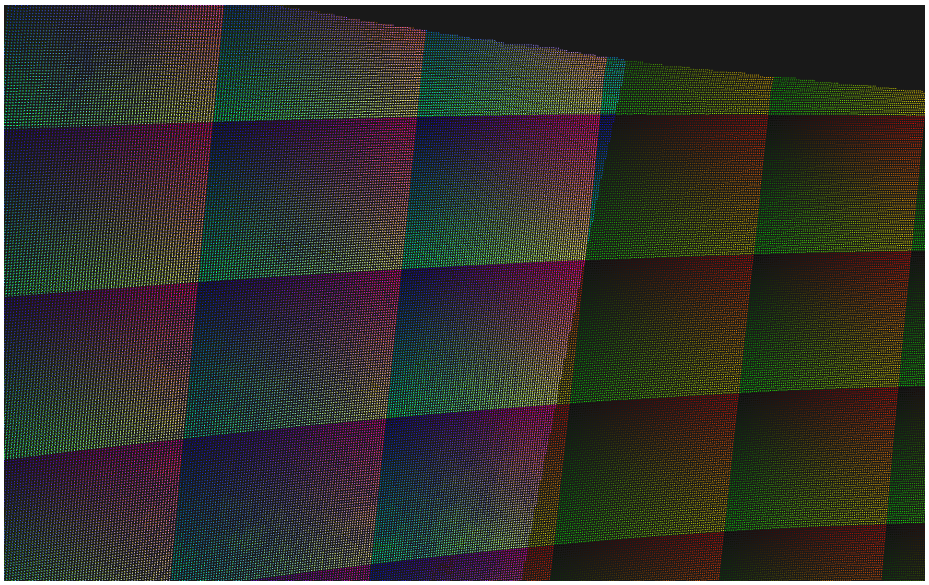
$$\mathbf{v}_{err} = \begin{bmatrix}
\|\mathbf{p}_A(j) - \mathbf{p}_{AB}(j)\| \\
\vdots \\
\|\mathbf{p}_B(j) - \mathbf{p}_{AB}(j)\| \\
\vdots \\
\|\mathbf{M}_A^{PL}(j) - distort(\mathbf{W}_l\mathbf{p}_A(j), \mathbf{K}_l)\| + \|\mathbf{M}_A^{PR}(j) - distort(\mathbf{W}_r\mathbf{p}_A(j), \mathbf{K}_r,)\| \\
\vdots \\
\|\mathbf{M}_B^{PL}(j) - distort(\mathbf{W}_l\mathbf{p}_B(j), \mathbf{K}_l)\| + \|\mathbf{M}_B^{PR}(j) - distort(\mathbf{W}_r\mathbf{p}_B(j), \mathbf{K}_r,)\|
\end{bmatrix}$$

On peut très bien voir l'amélioration des résultats lorsqu'on aligne deux nuages de points partiels d'un même projecteur. Si la calibration est mauvaise, deux reconstructions de points de vue différent des mêmes points d'un projecteur auront une forme différente et ne pourront être alignés parfaitement utilisant la méthode décrite dans la section 5.4. On le remarque très clairement dans la figure 2.4(a). Avec une calibration raffinée (figure 2.4(b)), l'alignement peut être fait beaucoup plus précisément.

De toute évidence, la sophistication de la méthode de calibration ne peut suffire à obtenir de bons résultats. Les correspondances projecteur-caméra doivent d'abord être d'une grande qualité. L'obtention des correspondances sera présenté dans le prochain chapitre.



(a)



(b)

Figure 2.4. Deux nuages de points partiels d'un projecteur alignés avec et sans raffinement de la calibration

Chapitre 3

MISE EN CORRESPONDANCE

Pour obtenir un modèle tridimensionnel d'une scène, il faut procéder à une mise en correspondance suivie d'une triangulation. Le modèle 3D est incontournable puisqu'il servira ensuite à calculer comment projeter des images *corrigées* sur la surface.

Généralement, on utilise une paire de caméras, ou une paire caméra-projecteur pour la mise en correspondance. Une paire de caméras seule oblige le recours à une reconstruction stéréoscopique, qui est imprécise et dépend de la texture des objets observés (reconstruction passive) [19, section 11.3]. Un projecteur peut aussi *augmenter* une scène et rendre la reconstruction plus facile et précise (reconstruction active) [11]. Dans cette situation, le projecteur ne participe pas directement à la mise en correspondance et le choix de ce qu'il projette n'a que peu d'importance.

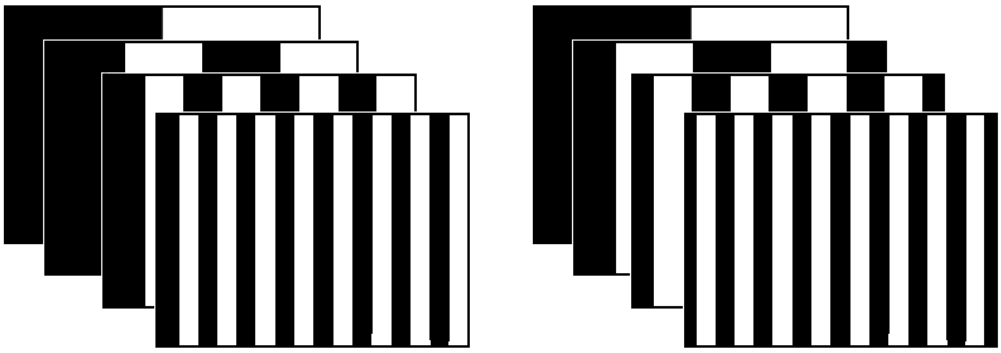
Dans notre cas, on dispose de deux caméras et d'un ou plusieurs projecteurs. On va donc utiliser une méthode active où le projecteur participe activement à la mise en correspondance, la *reconstruction à lumière non-structurée*.

3.1 Lumière structurée

La lumière structurée consiste à encoder d'une manière ou d'une autre les coordonnées 2D des pixels d'un projecteur dans des motifs. Ces motifs sont ensuite projetés puis captés par une caméra. On cherche à mettre en correspondance les pixels du projecteur avec les pixels de la caméra en décodant les motifs tels que captés par la caméra.

La façon la plus simple d'encoder les positions des pixels de projecteur est d'utiliser directement leur code binaire. Par exemple, pour un projecteur d'une résolution de

256 × 256, on peut encoder les coordonnées dans les deux axes sur 8 bits. Une position de (123, 45) a le code binaire (01111011, 00101101). A partir de ces codes, on crée un motif pour chaque bit contenant des barres noires et blanches selon la valeur du bit. Comme illustré dans la figure 3.1, on encode les positions selon l'axe horizontal sous forme de barres verticales où l'intensité, noir ou blanc, correspond à la valeur du bit. On fait de même pour l'axe vertical avec des motifs dans l'autre sens.



(a) (b)

Figure 3.1. Motifs binaires (a) et en code de Gray (b) selon l'axe des x pour les 4 bits les plus significatifs

Pour améliorer la robustesse de ce type de type de motifs, on peut encoder les coordonnées des pixels sous forme de code de Gray. Les codes de Gray sont aussi des codes binaires, mais qui ont la particularité que les codes de deux valeurs successives ne diffèrent que d'un seul bit. Ce sont les motifs les plus couramment utilisés pour faire de la reconstruction 3D [8].

Avec ces méthodes structurées, la correspondance se fait toujours du point de vue de la caméra: chaque pixel de la caméra se voit attribuer un code correspondant à une position dans l'image du projecteur. Or, on a parfois besoin de la mise en correspondance du point de vue du projecteur. C'est effectivement notre cas. Comment

faire? On peut *inverser* le mapping caméra vers projecteur, mais ce mapping n'est généralement pas inversible, parce qu'il peut contenir des trous ou des matchs multiples. On va donc aller vers une méthode non structurée, qui permettra à chaque pixel de la caméra de trouver un correspondant dans le projecteur, et symétriquement à chaque pixel du projecteur de trouver son correspondant dans la caméra.

3.2 Lumière non structurée

La différence majeure entre la lumière structurée classique et la présente méthode est le fait qu'on n'encode pas l'information de position des pixels du projecteur dans les motifs [4]. On va plutôt créer des motifs de bandes aléatoires (figure 3.2). Ces motifs on plusieurs avantages par rapport à ceux vus précédemment. En premier lieu, ces motifs sont résistants à l'illumination indirecte. Si on prend le premier motif issu d'un encodage binaire (fig 3.1), la moitié de la scène est illuminée directement par le projecteur, l'autre moitié restant dans le noir. Or, la lumière du projecteur sur la moitié de la scène est réfléchie et illumine indirectement l'autre moitié de la scène, ce qui cause invariablement des erreurs quand on tente de seuiller les valeurs de tons de gris de la caméra pour déterminer la valeur du bit. Les présents motifs ont une largeur de bande de fréquences (choisie en fonction de la scène) assez petite, et les zones noires et blanches sont uniformément distribuées. L'illumination indirecte résultant de la projection d'un tel motif reste constant pour toute la scène.

Ces motifs sont également spatialement cohérents, c'est-à-dire que deux pixels voisins se verront attribuer des codes très similaires. On peut constater dans l'exemple que deux pixels voisins se situent le plus souvent dans la même bande noire ou blanche. Lorsqu'une correspondance est établie entre un pixel du projecteur et de la caméra, il est donc probable qu'on puisse en trouver une meilleure dans les voisinages des deux positions.

La méthode n'a pas que des avantages: on doit projeter plus de motifs que pour

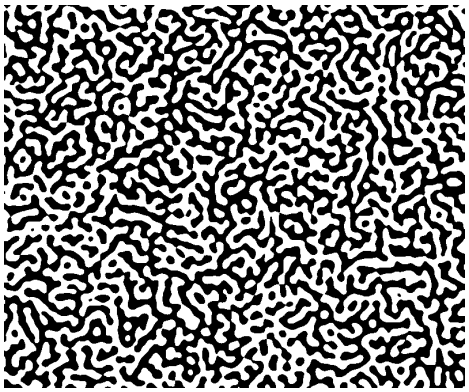


Figure 3.2. Motifs de lumière non-structurée.

les codes de Gray. Étant donné la nature aléatoire de ces motifs, il faut assez de motifs pour s'assurer qu'on ait assez de bits pour être en mesure de différencier les pixels entre eux.

Aussi, comme l'information de la position des pixels n'est pas encodée dans les motifs, on doit chercher soi-même pour trouver les pixels correspondants, ce qui n'est pas nécessaire avec les méthodes structurées. Par contre, parce que l'on obtient une mise en correspondance dans les deux sens en même temps (de la caméra vers le projecteur et vice-versa), c'est un faible prix à payer.

Choix des motifs

Pour générer tels motifs, on sélectionne une bande de fréquences en fonction du ratio entre la taille des pixels des projecteurs et des caméras. Pour chacune de ces fréquences, on fixe l'amplitude à 1 et on choisit la phase de façon aléatoire. On construit une image dans le domaine des nombres complexes contenant la phase des fréquences choisies. On applique une transformée de Fourrier inverse sur cette image, et on binarise le résultat, ce qui nous donne un motif à l'aspect voulu.

Pour obtenir un bit d'information pour un pixel donné, on prend simplement le signe de la différence entre la valeur du motif et de l'intensité moyenne observée sur

tous les motifs utilisés. Après avoir projeté puis capté n motifs, on a des codes de n bits pour chaque pixel du projecteur et de la caméra.

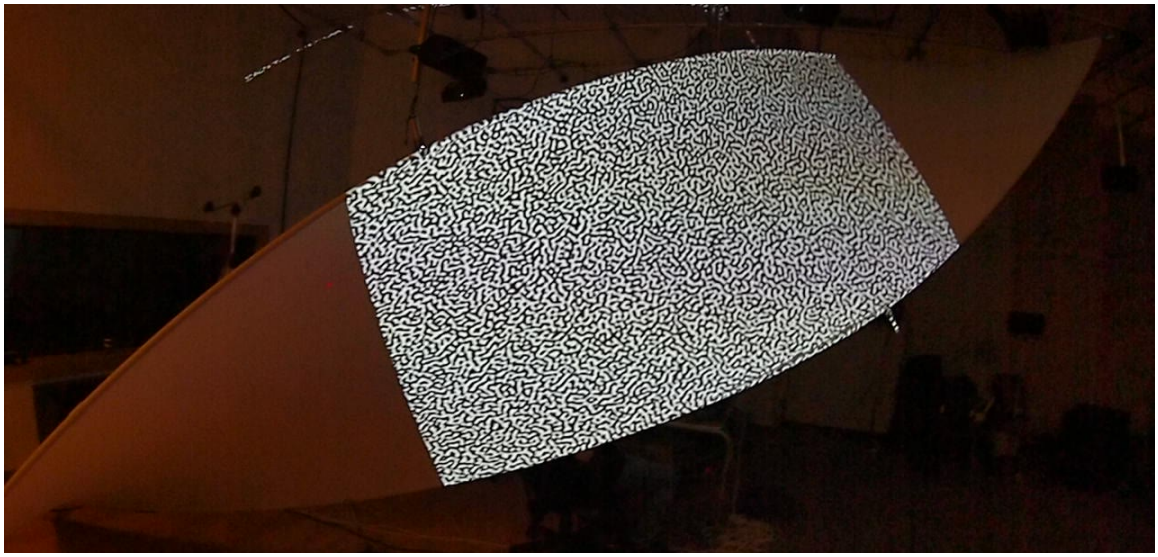


Figure 3.3. Motif projeté sur l'écran lors de la mise en correspondance.

Mise en correspondance

La véritable mise en correspondance entre les pixels des projecteurs et des caméras se fait de façon itérative.

On débute par sélectionner $b = \lceil log(w_{proj} \times h_{proj}) \rceil$ positions de bits au hasard. On bâtit une table de hachage pour b bits qu'on remplit avec les codes de n bits en prenant comme clé de hachage les bits choisis au hazard parmi ces codes. On fait de même pour les codes de la caméra et du projecteur.

On parcourt ensuite les deux tables de hachage simultanément pour comparer les codes complets du projecteur et de la caméra là où ces codes se sont vus attribuer la même clé. On calcule un coût de mise en correspondance égal aux nombre de bits différents entre les codes du projecteur et de la caméra. Si le coût est inférieur à celui de l'itération précédente, on met à jour le pixel correspondant.

On choisit ensuite b nouvelles positions de bits, et on recommence le processus de hachage. On arrête quand la solution se stabilise.

Précision Sous-pixel

Pour obtenir les coordonnées en sous-pixel, on obtient les codes différemment [5]. Pour une mise en correspondance sans précision sous-pixel, le signe de la différence entre les valeurs d'un motif avec celles de l'image moyenne nous donne des codes de n bits pour n motifs. Pour le sous-pixel, on va avoir besoin de beaucoup plus de bits par code. On va plutôt aller chercher le signe de la soustraction de toutes les paires de motifs possible, ce qui donne des codes de longueur $\binom{n}{2}$.

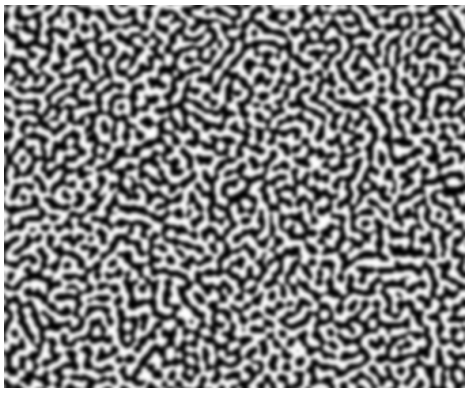
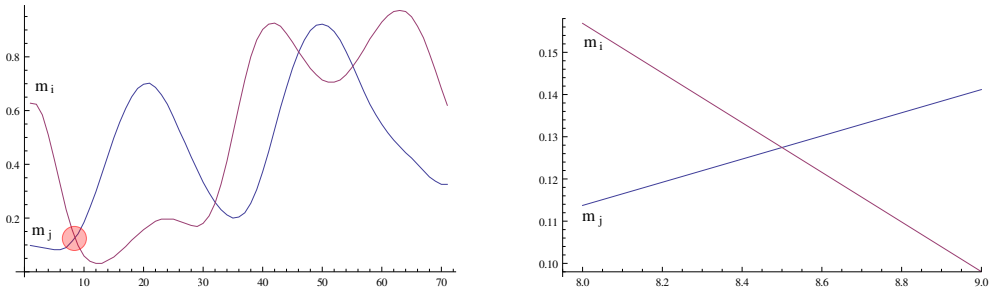


Figure 3.4. Exemple d'un motif utilisé pour obtenir une précision sous-pixel.

Il faut par contre flouer les patterns pour éviter les ambiguïtés quand les soustractions de motifs donnent zéro pour un pixel donné. Le match se fait de la même façon. Les patterns on la propriété d'être spatialement cohérents: les codes de deux pixels voisins sont toujours très similaires, a moins d'une discontinuité sur la surface de projection. Comme les pixels de caméra et de projecteurs ne sont jamais parfaitement alignés, encore moins avec une correspondance un pour un, chaque code vu par la caméra est en fait un mélange de multiples codes du projecteur.

Pour expliquer la suite de la méthode, on va se limiter à une seule dimension.

On cherche les passages par zéro entre les paires de motifs. Pour une paire de motifs donnée, m_i et m_j, si deux pixels voisins x et $x+1$ se voient attribuer un bit différent, on en déduit que la différence interpolée des intensités des deux motifs pour ces deux pixels passe par zéro. Plus précisément, on a un passage par zéro entre deux pixels voisins quand $m_i[x] > m_j[x]$ et $m_i[x+1] < m_j[x+1]$ ou vice-versa.



(a) (b)

Figure 3.5. Passage par zéro. L'image (b) montre avec plus de détail la zone mise en évidence en rouge dans l'image (a).

On suppose que les intensités interpolées évoluent linéairement entre les deux pixels voisins 3.5(b). On peut donc trouver la localisation Λ du passage par zéro:

$$(1-\Lambda)m_i[x] + \Lambda m_i[x+1] = (1-\Lambda)m_j[x] + \Lambda m_j[x+1] \tag{3.1}$$

On construit l'histogramme h de tous les Λ pour chaque paire de motifs qui génère un bit différent pour deux pixels voisin. À partir de h, on construit un histogramme cumulatif H à partir duquel on peut directement aller cherche la véritable position du match en sous-pixel. On n'a qu'a compter le nombre de bits différent t entre le code du point de vue de la caméra et le code correspondant du point de vue du projecteur, puis trouver la première classe (bin) i dans l'histogramme cumulatif H telle que $H(i) \geq t$. Le déplacement en sous-pixel λ d'un pixel par rapport à son voisin est:

$$\lambda = i + \frac{t - H[i-1]}{H[i]} \tag{3.2}$$

2D

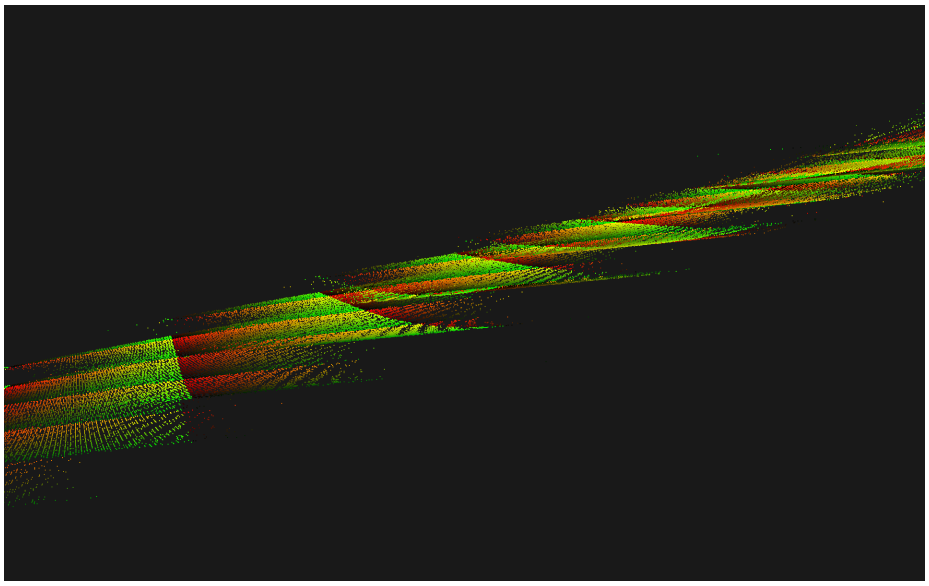
Pour obtenir la précision sous-pixel en deux dimensions, on va considérer quatre pixels voisins plutôt que deux. On va trouver le déplacement en sous-pixel (λ_x, λ_y) en calculant l'intersection entre deux plans bilinéaires définis par les valeurs en niveau de gris des motifs des quatre pixels voisins. Cette intersection correspond au passage par zéro vu précédemment, mais en 2D cette fois. Le calcul des valeurs de (λ_x, λ_y) est expliqué dans [5].

3.3 Triangulation

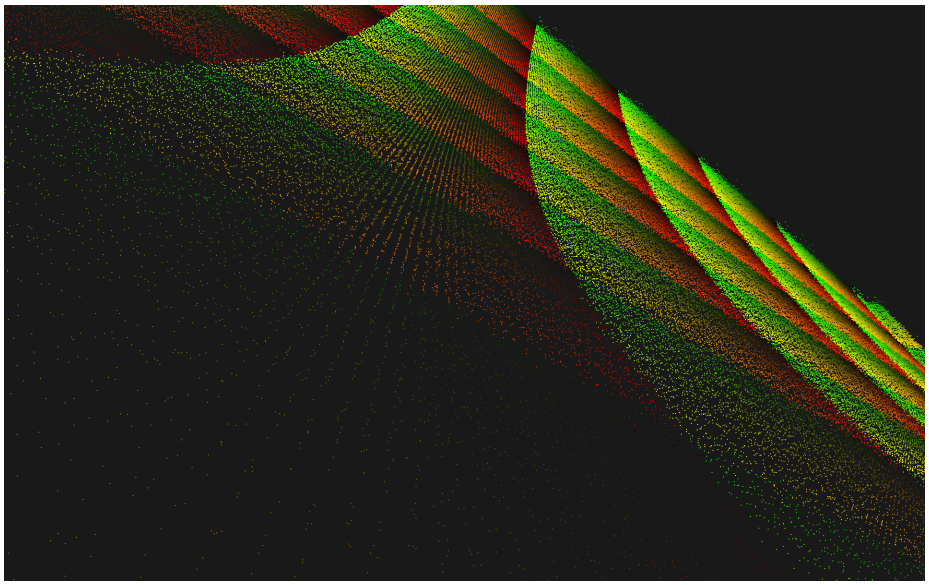
On capture les motifs du projecteur avec deux caméras calibrées en même temps. Comme on fait le match entre les pixels de chaque caméra des mêmes motifs des projecteurs, on a pour chaque points du projecteur une correspondance dans les deux caméras qu'on peut utiliser directement pour trianguler en 3D. L'avantage de la mise en correspondance en stéréo vient du fait que les caméras sont déjà calibrées entre elles. Si on faisait la mise en correspondance avec une seule caméra, on serait forcé de calibrer la pose entre les projecteurs et chaque position de caméra, ce qui est peu pratique et difficile à faire étant donné que les points reconstruits ne sont pas nécessairement propices à une bonne calibration.

3.4 Scènes de grandes tailles

Souvent, la situation en projection sur surface arbitraire implique une surface de grande dimension illuminée section par section par de multiples projecteurs. Idéalement, on utiliserait une seule caméra pour reconstruire l'ensemble de la scène à partir d'un seul point de vue. Cela est malheureusement impossible. La caméra n'aura



(a)



(b)

Figure 3.6. Nuages de points résultant de la triangulation à partir des mises en correspondance avec et sans précision sous-pixel.

généralement une résolution suffisante, et donc le modèle sera moins précis. Aussi, la surface à reconstruire peut être impossible à voir d'un seul point de vue, comme lorsqu'il y a de l'occlusion, des obstacles, ou des formes concaves.

On va donc reconstruire la surface par morceaux en déplaçant la caméra, comme illustré dans la figure 3.7. On aura alors plusieurs reconstructions pour un même projecteur que l'on devra fusionner par la suite (voir section 5.6).

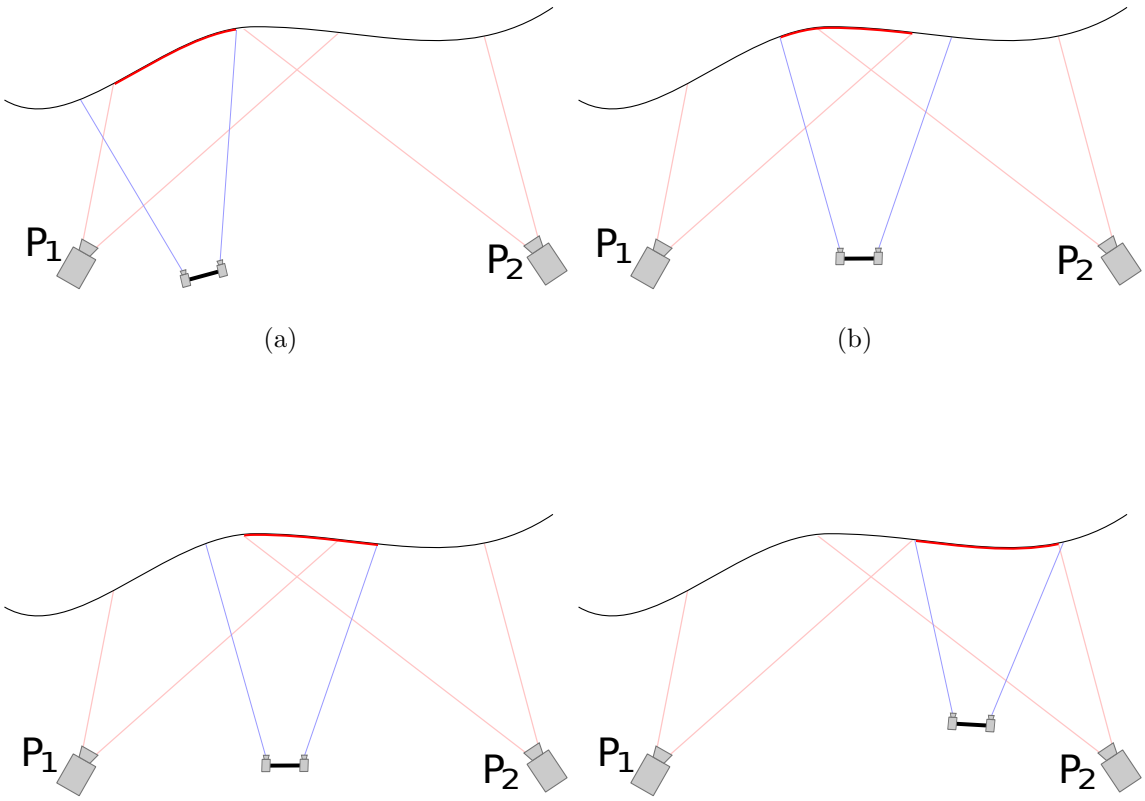
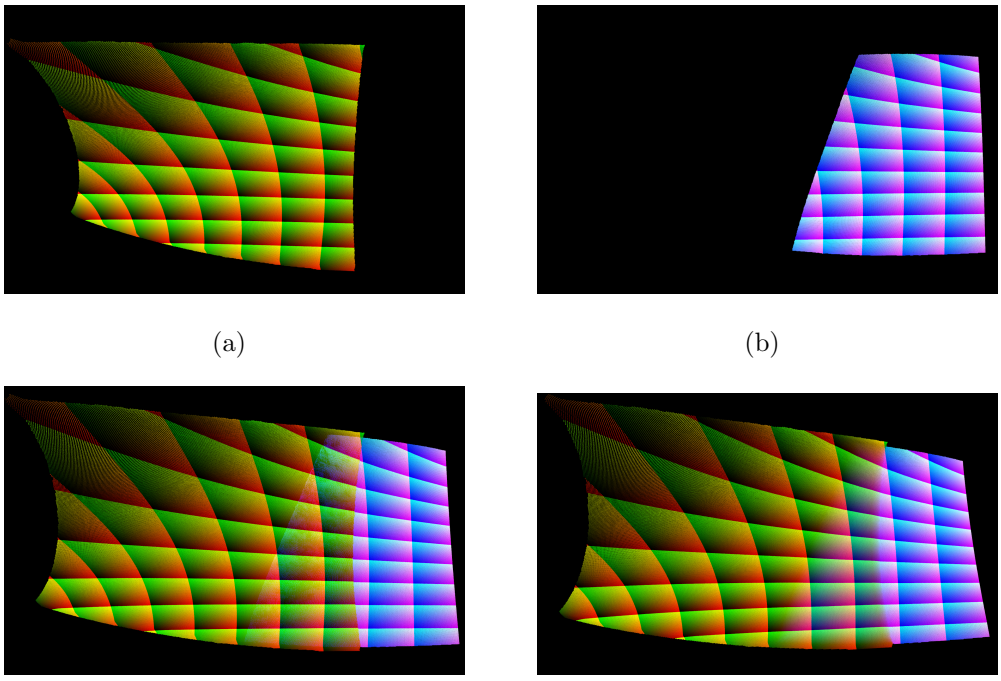


Figure 3.7. Reconstruction par morceaux. Les projecteurs P_1 et P_2 sont fixes. Les caméras sont déplacées séquentiellement jusqu'à ce qu'on ait entièrement couvert la surface. Notez qu'entre (b) et (c), les caméras n'ont pas bougé: il est nécessaire de reconstruire la zone de chevauchement d'un seul point de vue pour les deux projecteurs pour qu'ils soient déjà alignés en 3D.



(a) (b)

(c) (d)

Figure 3.8. (a) et (b): deux reconstruction partielles pour un projecteur. (c) Les deux mêmes reconstructions alignées et superposées. (d) Les deux modèles fusionnés

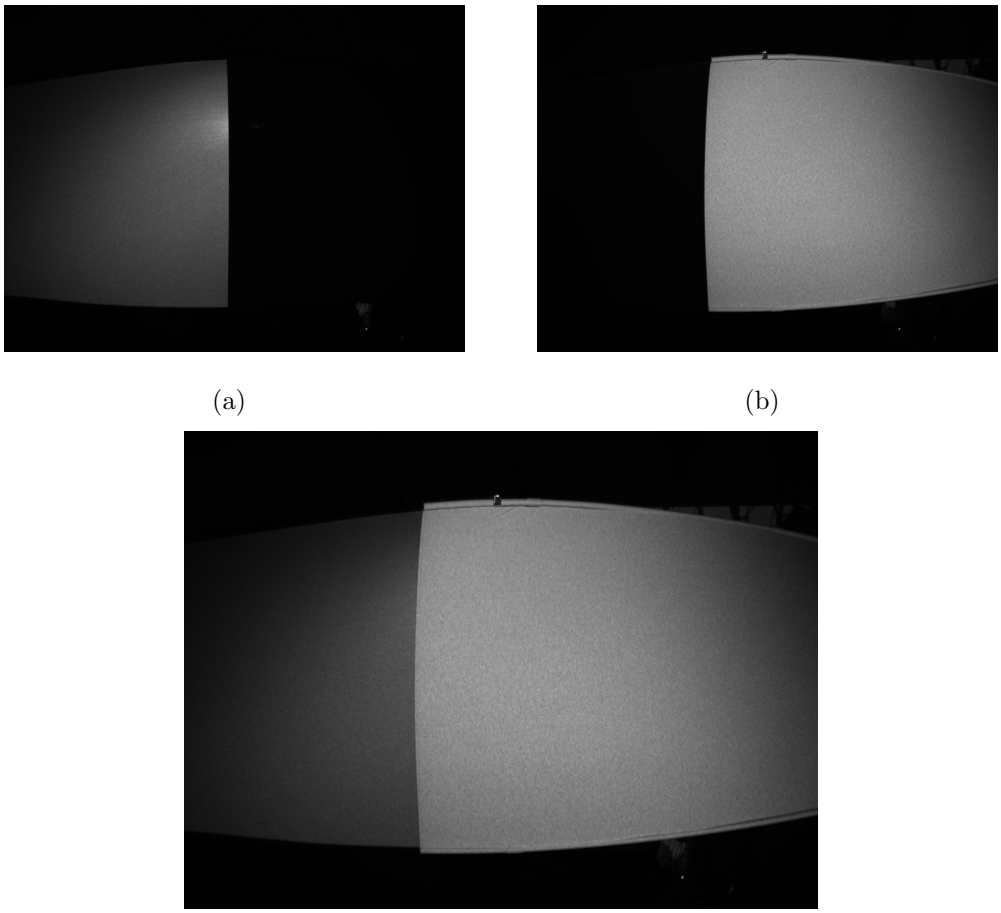
Chapitre 4

ALIGNEMENT PHOTOMÉTRIQUE

Avant de reconstruire un modèle 3D de l'écran, on doit procéder à l'alignement photométrique les projecteurs entre eux. Cette étape s'effectue en 2D et ne requiert pas de disposer d'un modèle 3D de l'écran. Les projecteurs ont des zones de chevauchement pour éviter les trous dans la projection. L'alignement photométrique consiste à ajuster l'intensité de chaque pixel des projecteurs dans ces zones de chevauchement de manière à ne obtenir une projection continue sans transitions visible entre les projecteurs. En d'autres mots, il s'agit de faire un fondu progressif entre deux projecteur ou plus. Cet ajustement d'intensité se fait sous la forme d'un masque de mixage qui module la valeur des pixels pour l'image du projecteur.

Lorsqu'on fait une mise en correspondance d'un point de vue où les caméras voient l'image de plus qu'un projecteur à la fois, on va calculer ce fondu à partir des intensités des pixels des projecteurs telles que perçues par les caméras. L'alignement photométrique n'est calculé que pour les pixels des projecteurs effectivement vus par les caméras. Donc pour un projecteur donné, on peut avoir plusieurs masques de mixage partiels que l'on devra fusionner par la suite.

L'algorithme proposé est itératif. On veut calculer un facteur d'échelle de l'intensité pour chaque pixel de chaque projecteur dans les zones de chevauchement de façon à ce que la somme des intensités telles que captées par la caméra soit lisse. Cette somme S, initialisée à la valeur maximale des intensités, ainsi que les intensités maximale de chaque projecteur individuel I_i, sont lissés à chaque itération. Les valeurs de S et des I_i hors des zones de chevauchement sont ensuite remises à leur valeur initiale. Les pixels des I_i à l'intérieur de la zone de chevauchement sont multipliés par le ratio entre S et la somme s des I_i, pour assurer que la somme des intensités des projecteurs



(a) (b)

(c)

Figure 4.1. (a) et (b) Intensités maximum des deux projecteurs vu de la caméra. (c) Intensités maximum pour tous les projecteurs en même temps.

ne dépassent pas S. L'algorithme termine lorsque la différence entre deux solutions d'itérations successive est suffisamment petite.

On a donc:

- I_i^{max} : intensité maximale captée par la caméra lors de la captation pour le projecteur i.

- I^{map}: pour chaque pixel de caméra, le nombre de projecteurs qui projettent à

cet endroit.

- I_{all}^{max} : valeur maximum dans la caméra pour tous les projecteurs à la fois.

$$I_{all}^{max}(x,y) = max(I_i^{max}(x,y), \forall i \in [0..N_{proj}])$$

- S^k : somme désirée des intensité des projecteurs du point de vue de la caméra, à l'itération k. S^0 est initialisés à I_{all}^{max}.

- I_i^k : le mixage normalisé du projecteur i à l'itération k du point de vue de la caméra. I_i^0 est initialisé à I_i^{max}.

Le lissage de la somme S^k est défini comme suit:

$$\begin{aligned} S^0(x,y) &= I_{all}^{max}(x,y) \\ S^k(x,y) &= \begin{cases} 0 & \text{si } I^{map}(x,y) = 0 \\ I_{all}^{max}(x,y) & \text{si } I^{map}(x,y) = 1 \\ flou(S^{k-1})(x,y) & \text{si } I^{map}(x,y) > 1 \end{cases} \end{aligned}$$

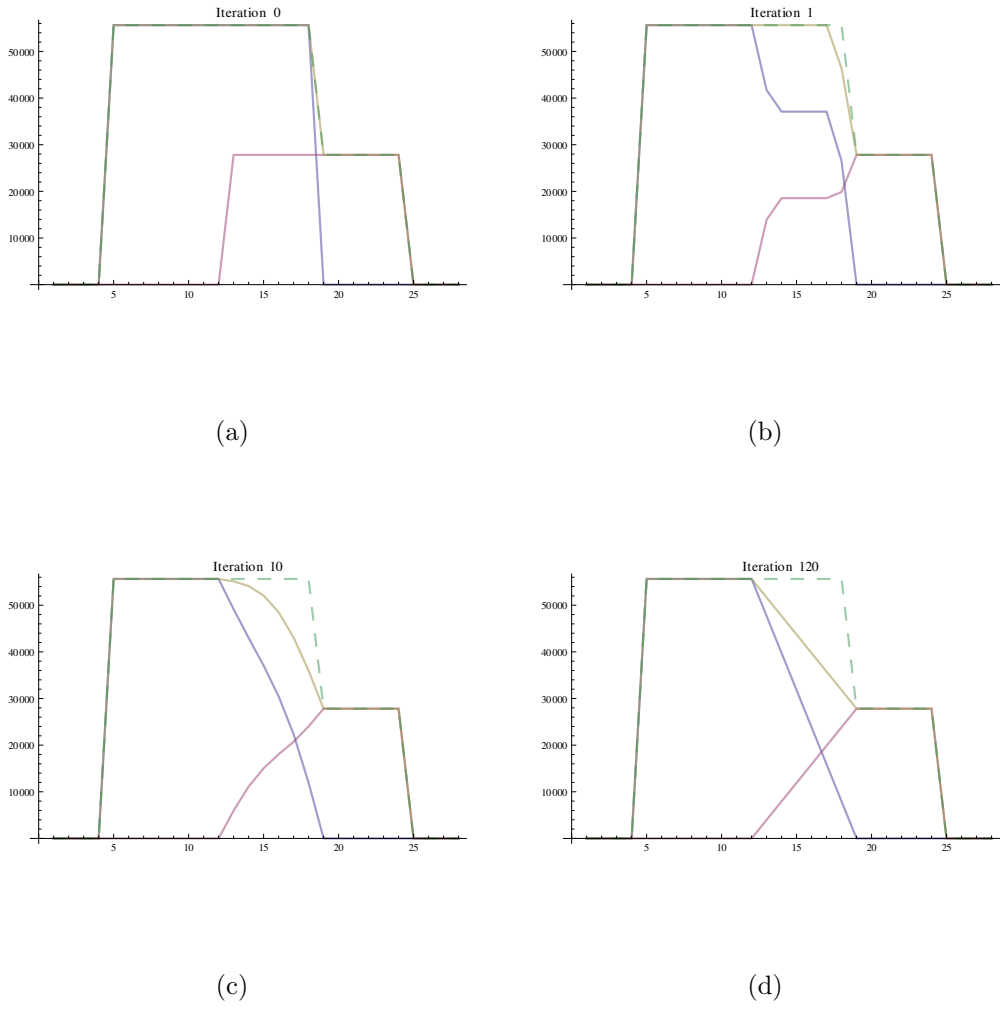
où $flou()$ applique un lissage spatial gaussien. Le mixage non normalisé $\hat{I}_i^k$ des projecteurs est calculé comme suit:

$$\hat{I}_i^k(x,y) = min\left(flou(I_i^{k-1})(x,y), I_i^{max}(x,y)\right)$$

Ici, on ne garde que le minimum entre I_i^{max} et les valeurs résultant du lissage de I_i^{k-1} parce que l'intensité d'un pixel ne doit pas être plus élevée que l'intensité maximale telle que captée par la caméra. Évidemment, dans les zones ou $I_i^{max} = 0$, le mixage doit rester 0.

Le mixage normalisé I_i^k est définit par

$$I_i^k(x,y) = \begin{cases} \frac{\hat{I}_i^k(x,y)}{s} S^k(x,y) & \text{si } s > 0 \\ 0 & \text{sinon} \end{cases} \tag{4.1}$$



(a) (b)

(c) (d)

Figure 4.2. Évolution du mixage au fil des itérations. Les courbes en bleue et rouge sont les intensités maximum de deux projecteurs telles que captées par la caméra, la courbe en beige est la somme désirée des intensités, et la courbe pointillée I_{all}^{max}

où

$$s = \sum_{i=1}^{N_{proj}} \hat{I}_i^k(x, y) \tag{4.2}$$

Lorsque $s > 0$, au moins un projecteur projette sur ce point, sinon $I_i^k(x, y) =$

$I_i^{max}(x,y) = 0$ parce qu'il n'y a aucune projection à cet endroit.

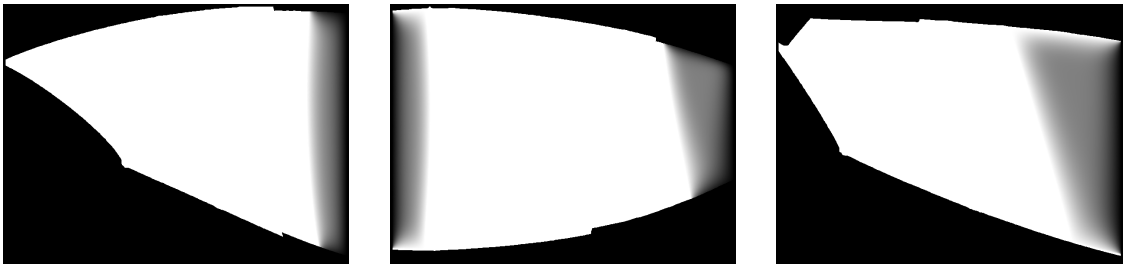
Le calcul itératif du mixage s'arrête lorsque le critère

$$\sum_{i}^{N_{proj}} \sum_{x}^{x_s} \sum_{y}^{y_s} \|I_i^k(x,y) - I_i^{k-1}(x,y)\| \leq C$$

est satisfait. C est fixé arbitrairement à une valeur suffisamment petite pour considérer que la solution s'est stabilisée. La remise à l'échelle garantie qu'à chaque itération, la solution est *valide*, c.-à-d. que la somme des intensités des projecteurs ne dépasse jamais S^k. On choisit donc C pour obtenir un résultat suffisamment lisse.

Le mixage final I_i^{mix} doit être normalisé pour chaque projecteur en fonction de l'intensité maximale qu'il peut générer:

$$I_i^{mix}(x,y) = \begin{cases} \frac{I_i^k(x,y)}{I_i^{max}(x,y)} & \text{si } I_i^{max}(x,y) > 0 \\ 0 & \text{sinon} \end{cases}$$



(a) (b) (c)

Figure 4.3. Résultats du calcul du mixage final, du point de vue des projecteurs.

Puisque la reconstruction d'une surface de projection se fait par morceaux, on va avoir plusieurs I^{mix} partiels pour un même projecteur. On verra plus tard comment fusionner les différentes vues pour chaque projecteur.

Chapitre 5

RECONSTRUCTION 3D

La reconstruction d'objets 3D à partir de paires d'images stéréo est un problème classique de la vision par ordinateur. Une fois une correspondance établie entre deux vue d'un même objet, la triangulation peut être utilisée pour calculer la coordonnée 3D du point associé à cette correspondance.

Évidemment, la triangulation requiert une calibration préalable. Puisque dans notre cas seules les caméras sont calibrées, la triangulation devra se faire à partir de deux caméras. Les caméras n'étant pas mise en correspondance directement, on passera par l'intermédiaire d'un projecteur. Ainsi, seuls les points visibles des deux caméras et du projecteur seront reconstruits.

Le résultats d'une triangulation est un nuage de points 3D dans le système de la caméra gauche associé à un point de vue donné, et pour un projecteur choisi. On obtiendra donc un maximum de (nombre de points de vues) $\times$ (nombre de projecteurs) nuages de point 3D. La triangulation sera décrite dans les sections 5.1, 5.2 et 5.3.

Puisque notre but est d'obtenir une seule reconstruction cohérente de la surface de reprojection, un processus de réalignement sera appliqué aux nuages reconstruits.

Deux nuages obtenus de deux points de vue différents pour un même projecteur devront être alignés entre eux. Notons que ces nuages sont déjà mis en correspondance par le fait qu'ils sont associés au même projecteur. Cette étape sera décrite à la section 5.4.

En théorie, deux nuages obtenus d'un même point de vue avec des projecteurs différents devraient déjà être alignés puisqu'ils sont exprimés dans le même système de coordonnées (celui de la caméra gauche de ce point de vue). Cette situation est décrite a la section 5.5.

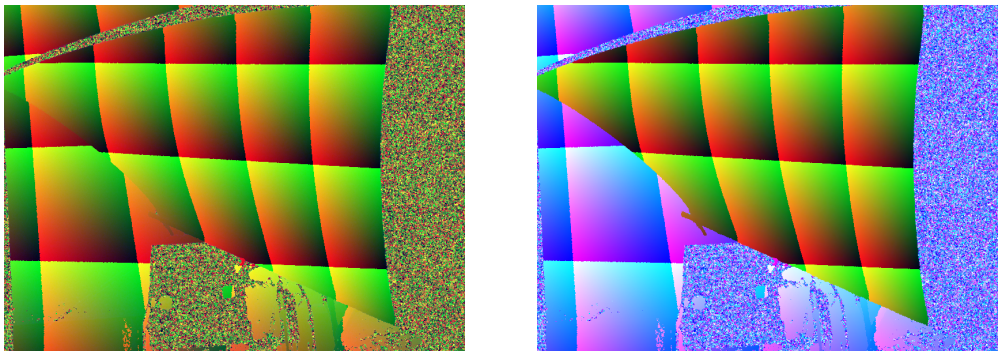
Puisque la reconstruction 3D sera ultimement utilisée pour déformer des projections, il est important de n'avoir qu'un seul point reconstruit par pixel d'image de projecteur. Ainsi, il faudra fusionner les reconstructions issues de points de vue multiple. Cette étape sera décrite à la section 5.6.

5.1 Prétraitement des mises en correpondance

On a des images de mise en correspondance, mais ce ne sont pas tous les points de ces images qui nous intéressent. On va devoir faire une élimination de l'arrière-plan pour se débarrasser de tous les points hors de l'écran.

On fait un remplissage par diffusion de l'image pour déterminer les zones connexes dans les images de mise en correspondance. Chacune de ces zones se voit attribuer une étiquette e_z, qui est incrémentée chaque fois que le remplissage termine et qu'on tombe dans une nouvelle zone. Pour un pixel p donné, on poursuit le remplissage chez un voisin q si la distance entre les valeur (x, y) des correspondances de p et q est en deçà d'un seuil de tolérance. La région ainsi remplie ne contient donc que des pixels connexes, sans discontinuités. Pour chaque pixel, on incrémente un compteur associé au e_z de la région.

Étant donné que la prise d'image est faite en sorte que la région reconstruite la plus vaste est celle qui nous intéresse, on choisit la région qui contient plus grand nombre de pixels. Les zones non-reconstruites ne contiennent que des données très bruitées qui résultent en une multitude de très petites zone qu'on peut aisément ignorer. Les zones qui ne sont pas dans la zone choisie se voient associer un coût de mise en correspondance maximal pour qu'il ne soient plus considérés dans les traitements subséquents.



(a) (b)

Figure 5.1. Élimination des zones non-reconstruites. Les régions teintées de bleu dans l'image de gauche seront ignorées dans les traitements subséquents.

5.2 Triangulation

Une triangulation sera effectuée pour chaque point de vue et chaque projecteur. On part des images de correspondance du projecteur. Pour chaque projecteur, on a deux mises en correspondance, une vers chaque caméra. Pour chaque point du projecteur, on a donc les coordonnées en pixel (ou sous pixel) dans les caméras de gauche et droite. On enlève la distorsion radiale des coordonnées image des deux caméras, puis on les triangule par la minimisation d'un système d'équation linéaire [19, section 7.1]

On a donc deux points en coordonnées image sans distorsion radiale $p_l = (x_l, y_l)$ et $p_r = (x_r, y_r)$, et on cherche le point 3D $p_w = (x_w, y_w, z_w, w_w)$ tel que $p_l = \mathbf{K}_l p_w$ et $p_r = \mathbf{K}_r \mathbf{R}_{lr} \mathbf{T}_{lr} p_w$. Si on pose

$$\mathbf{M}_l = \mathbf{K}_l \text{ et } \mathbf{M}_r = \mathbf{K}_r \mathbf{R}_{lr} \mathbf{T}_{lr}$$

ou a donc pour p_l

$$\begin{aligned} x_l &= \frac{m_{l00}x_w + m_{l01}y_w + m_{l02}z_w + m_{l03}w_w}{m_{l20}x_w + m_{l21}y_w + m_{l22}z_w + m_{l23}w_w} \\ y_l &= \frac{m_{l10}x_w + m_{l11}y_w + m_{l12}z_w + m_{l13}w_w}{m_{l20}x_w + m_{l21}y_w + m_{l22}z_w + m_{l23}w_w} \end{aligned}$$

et une relation similaire pour p_r (remplacer l par r).
Pour trouver $p_w = (x_w, y_w, z_w, w_w)$, on a donc le système d'équations linéaires

$$\begin{aligned}
x_l(m_{l20}x_w + m_{l21}y_w + m_{l22}z_w + m_{l23}w_w) - m_{l00}x_w - m_{l01}y_w - m_{l02}z_w - m_{l03}w_w &= 0 \\
y_l(m_{l20}x_w + m_{l21}y_w + m_{l22}z_w + m_{l23}w_w) - m_{l10}x_w - m_{l11}y_w - m_{l12}z_w - m_{l13}w_w &= 0 \\
x_r(m_{r20}x_w + m_{r21}y_w + m_{r22}z_w + m_{r23}w_w) - m_{r00}x_w - m_{r01}y_w - m_{r02}z_w - m_{r03}w_w &= 0 \\
y_r(m_{r20}x_w + m_{r21}y_w + m_{r22}z_w + m_{r23}w_w) - m_{r10}x_w - m_{r11}y_w - m_{r12}z_w - m_{r13}w_w &= 0
\end{aligned}$$

qui peut s'exprimer sous forme matricielle

$$\mathbf{A}\begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

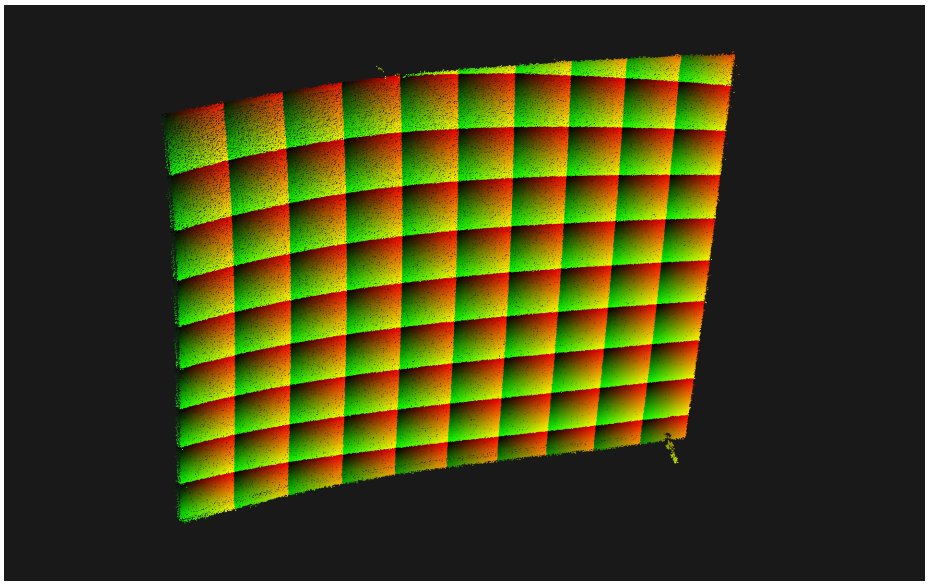
avec

$$\mathbf{A} = \begin{bmatrix}
m_{l20}x_l - m_{l00} & m_{l21}x_l - m_{l01} & m_{l22}x_l - m_{l02} & m_{l23}x_l - m_{l03} \\
m_{l20}x_l - m_{l10} & m_{l21}x_l - m_{l11} & m_{l22}x_l - m_{l12} & m_{l23}x_l - m_{l13} \\
m_{r20}x_r - m_{r00} & m_{r21}x_r - m_{r01} & m_{r22}x_r - m_{r02} & m_{r23}x_r - m_{r03} \\
m_{r20}x_r - m_{r10} & m_{r21}x_r - m_{r11} & m_{r22}x_r - m_{r12} & m_{r23}x_r - m_{r13}
\end{bmatrix}$$

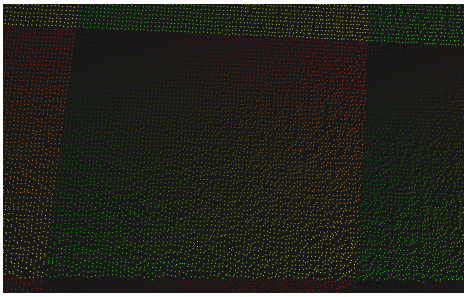
Pour résoudre ce système linéaire homogène, on applique une décomposition en valeurs singulières (SVD) sur $\mathbf{A}$ pour obtenir directement la position 3D triangulée. La figure 5.2 illustre une triangulation typique obtenue des correspondances.

5.3 Post-traitement: Lissage

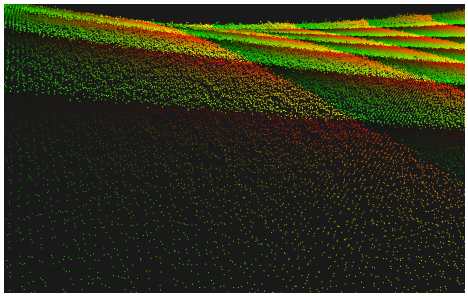
La mise en correspondance en sous pixel nous donne certes une reconstruction plus lisse que si on utilisait des match entiers. Or, comme l'illustre la figure 5.2, la reconstruction reste tout de même légèrement bruitée. Il est donc souhaitable de faire un lissage du nuage de points. On va faire un simple filtre moyenneur sur les positions 3D des points de la reconstruction. Cette technique de lissage est la plus rudimentaire, mais étant donné que la taille du filtre utilisée est très petite, soit un rayon de



(a)



(b)



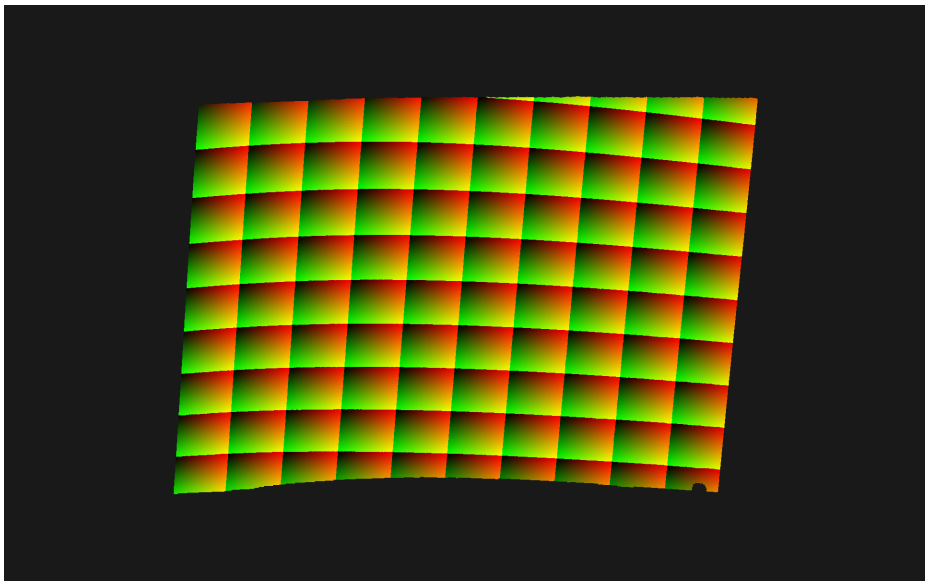
(c)

Figure 5.2. Nuage de points non filtrés

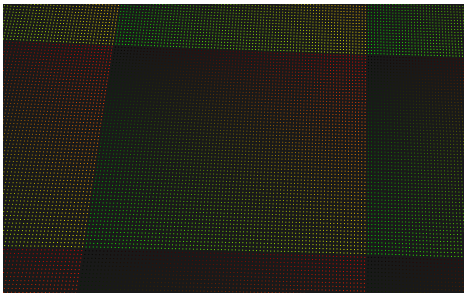
5 pixels dans l'image du projecteur, on va la préférer à d'autres méthodes comme les filtres gaussien et bilatéral pour la simple raison du temps de calcul.

5.4 Alignement de points de vues différents

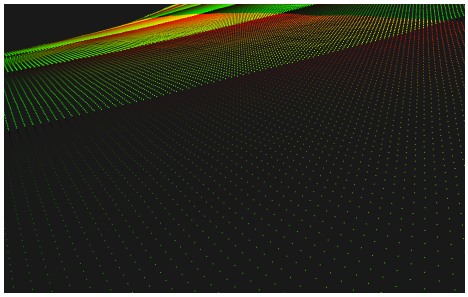
Pour fusionner les nuages de points de vues multiples d'un même projecteur, on doit d'abord aligner entre eux les points qu'ont en commun chaque nuage de points 3D associés aux différents points de vue. Si, par exemple, on a pour un projecteur deux



(a)



(b)



(c)

Figure 5.3. Nuage de points lissés, rayon de 5 points

nuages de points partiels $\mathbf{P}_a$ et $\mathbf{P}_b$ avec une zone de chevauchement entre les deux, on va sélectionner les points communs entre les deux projecteurs dans la zone de chevauchement, $\mathbf{Q}_a$ et $\mathbf{Q}_b$, et en déduire la rotation et la translation.

On soustrait de chaque point de $\mathbf{Q}_a$ et $\mathbf{Q}_b$ leur centre de masse respectif pour

obtenir une matrice $\hat{\mathbf{Q}}_\mathbf{a}$

$$\hat{\mathbf{Q}}_a = \begin{bmatrix} \mathbf{q}_a(1) - \mathbf{c}_a \\ \mathbf{q}_a(2) - \mathbf{c}_a \\ \vdots \\ \mathbf{q}_a(n) - \mathbf{c}_a \end{bmatrix}$$

où $\mathbf{q}_a(j)$ est le point j de l'ensemble $\mathbf{Q}_a$ et où n est le nombre de points dans $\mathbf{Q}_a$, et $\mathbf{c}_a$ le centre de masse de $\mathbf{Q}_a$ défini comme

$$\mathbf{c}_a = \frac{1}{n} \sum_{j=1}^{n} \mathbf{q}_a(j)$$

La matrice $\hat{\mathbf{Q}}_\mathbf{b}$ est définie de façon similaire. On va maintenant trouver la rotation entre $\hat{\mathbf{Q}}_\mathbf{a}$ et $\hat{\mathbf{Q}}_\mathbf{b}$, qui revient à résoudre le problème de Procrustes orthogonal [18]. On forme la matrice de covariance $\mathbf{H}$ des points 3D des deux vues du projecteur $\mathbf{P}$

$$\mathbf{H} = \hat{\mathbf{Q}}_a^T \hat{\mathbf{Q}}_b$$

qui sera ensuite décomposée en valeurs singulières (SVD) pour obtenir une première estimation de la matrice de rotation $\mathbf{R}^*$ qui minimise au sens des moindres carrés la distance entre les paires de points correspondants $\mathbf{q}_a(j)$ et $\mathbf{q}_b(j)$, pour $1 \leq j \leq n$. On a

$$\begin{aligned} [U, W, V^T] &= SVD(\mathbf{H}) \\ \mathbf{R}^* &= (VU^T)^{-1} \end{aligned}$$

Dans un monde idéal, on pourrait s'arrêter ici, et utiliser $\mathbf{R}^*$ directement. En pratique, il est possible que le système de caméras stéréo ait bougé légèrement entre deux prises de vues. Dans un tel cas, deux prises de vues d'un même projecteur serait reconstruites à des échelles différentes et donc impossible à aligner si on suppose que la rotation et la translation entre les deux caméras sont immuables.

On va donc se servir de la matrice de rotation $\mathbf{R}^*$ et des deux matrices de translation décrites par les centres de masse des deux reconstructions $\mathbf{T}_a = T[\mathbf{c}_a]$

et $\mathbf{T}_b = T[\mathbf{c}_b]$ comme point de départ, et on va ajuster ces transformations de façon à minimiser la distance réelle entre chaque paire de points correspondant via l'algorithme minimisation non-linéaire de Levenberg-Marquardt (voir section 2.4).

On va aussi inclure dans la minimisation la rotation et la translation entre les deux caméras du système stéréoscopique pour la seconde reconstruction, $\mathbf{R}_b^{lr}$ et $\mathbf{T}_b^{lr}$. $\mathbf{R}_b^{lr}$ et $\mathbf{T}_b^{lr}$ sont initialisées à $\mathbf{R}_a^{lr}$ et $\mathbf{T}_a^{lr}$ qui, elles, resteront constantes.

On a donc un système de n équations à 12 variables initialisées à zéro:

- les différences d'angle dans les trois axes par rapport à $\mathbf{R}^*$: θ_x, θ_y, θ_z.
- la translation par rapport à $\mathbf{T}_b$.
- les différences d'angle dans les trois axes par rapport à $\mathbf{R}_a^{lr}$: ϕ_x, ϕ_y, ϕ_z.
- la translation par rapport à $\mathbf{T}_b^{lr}$.

$$\mathbf{R}_\Delta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_\Delta^{lr} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_x) & -\sin(\phi_x) \\ 0 & \sin(\phi_x) & \cos(\phi_x) \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi_y) & 0 & -\sin(\phi_y) \\ 0 & 1 & 0 \\ \sin(\phi_y) & 0 & \cos(\phi_y) \end{bmatrix} \cdot \begin{bmatrix} \cos(\phi_z) & -\sin(\phi_z) & 0 \\ \sin(\phi_z) & \cos(\phi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{R} &= \mathbf{R}^*\mathbf{R}_\Delta \\ \mathbf{C}_{RT} &= \mathbf{R}^{lr}\mathbf{R}_\Delta^{lr}\mathbf{T}^{lr}\mathbf{T}_\Delta^{lr} \\ \mathbf{A}_{RT} &= \mathbf{T}_a\mathbf{R}^*\mathbf{R}_\Delta\mathbf{T}_b\mathbf{T}_\Delta \\ \mathbf{v}_{err} &= \left[\ \|\mathbf{P}_a(j) - \mathbf{A}_{RT}^{-1} Triangule(\mathbf{W}_l, \mathbf{W}_r\mathbf{C}_{RT}, \mathbf{P2d}_b^l(j), \mathbf{P2d}_b^r(j) \ \right] \end{aligned}$$

$\forall j$ où on a une correspondance entre deux points dans les nuage de points A et B. En pratique, on ne prend qu'un sous ensemble des points dans la zone de chevauchement.

Une fois le système minimisé, il ne reste qu'à trianguler de nouveau tous les points de la prise de vue B selon l'alignement $\mathbf{A}_{RT}$ et la nouvelle transformation entre les caméras $\mathbf{C}_{RT}$. Comme les mise en correspondance sont prises séquentiellement, on part de la première prise de vue, et on ajuste toutes les autres de façon incrémentale. C'est-à-dire, on ajuste la seconde prise de vue à partir de la première, et chaque prise de vue subséquente avec la précédente.

Dans le cas où l'ordre des caméras serait cyclique ou plus complexe qu'une simple séquence ordonnée, il faudrait alors résoudre tous les systèmes de tous les points de vue en même temps.

5.5 Alignement de projecteurs différents

Pour un même point de vue, on souhaite aligner deux nuages de points 3D associées chacun à un projecteur différent.

Dans le processus d'alignement incrémental décrit précédemment, on va invariablement arriver à la situation dans laquelle les deux prises de vues à aligner sont celles de deux projecteurs différents. Dans ce cas, on se rappelle que les deux mises en correspondance en question ont été prises de la même position de caméras. On a donc rien à faire, en théorie. On peut simplement attribuer les rotations/translations de caméras et d'alignement de la prise de vue du premier projecteur au second et poursuivre avec les vues subséquentes.

Si on s'apercevait que les deux reconstructions ne sont pas parfaitement alignées, on pourrait procéder comme pour l'alignement des reconstructions partielles de projecteurs. On n'aurait qu'à prendre les coordonnées des mises en correspondances des caméras directement plutôt que celles de projecteurs. En effet, on a une correspondance directe entre les deux projecteurs via les pixels des caméras. En pratique, cette situation ne se présente jamais.

5.6 Fusion

Une fois que tous les nuages de points 3D des différentes vues pour un même projecteur sont alignés, on doit les fusionner. En effet, les points se situant dans les zones de chevauchement du projecteur ont étés projetés de plusieurs points de vues. Pour obtenir une correspondance un-pour-un entre les points 3D et les pixels du projecteur, on calcule une moyenne pondérée de tous les points 3D reconstruits pour chaque pixel du projecteur:

$$\mathbf{P}_i = \frac{\sum_{j=1}^{n} \mathbf{W}_{ji}\mathbf{V}_{ji}}{\sum_{j=1}^{n} \mathbf{W}_{ji}}$$

Cette pondération $\mathbf{W}_j$ doit être calculée pour chaque vue du projecteur. Cette situation est similaire à celle de l'alignement photométrique (chapitre 4), et on calculera donc les poids de la même façon.

Chapitre 6

MAILLAGE

Dans ce travail, on considère que la surface de projection est fait une surface rectangulaire qui a été déformée par étirement. On doit donc attribuer à chaque point 3D de la surface reconstruite une correspondance vers la surface rectangulaire de référence. Ceci correspond à attribuer une coordonnée *texture* 2D à chaque point 3D. Pour établir la correspondance, il sera nécessaire de transformer les nuages de points 3D en une surface 3D, un maillage.

Le maillage ne fait qu'ajouter une propriété de connectivité à un nuage de points 3D. Cela permettra ensuite de mesurer des distances géodésiques sur le modèle 3D de l'écran. Ces distances serviront à associer des coordonnées textures 2D à chaque point de la surface.

Le maillage consiste en une série de triangles formés à partir des points du nuage, de façon à créer une surface cohérente en 3D. C'est sur ces triangles qu'on pourra ensuite calculer des distance sur la surface 3D elle-même.

Puisque les points du nuage sont chacun associé à un pixel d'un projecteur, ils peuvent être ordonnés selon les axes x et y de l'image du projecteur.

Sans cet ordre, on serait forcés d'utiliser d'autres méthodes beaucoup plus couteuses qui ne nous garantissent aucunement un maillage le plus régulier possible. On connaît donc exactement le voisinage de chaque point 3D, donc on peut faire un maillage régulier sans utiliser de techniques avancées.

Cette situation idéale ne prévaut que pour un seul projecteur à la fois. Dans les zones de chevauchement entre deux projecteurs, on va devoir entrecroiser les maillages en modifiant les triangles existant pour préserver une surface continue et régulière.

6.1 Partition de l'espace à l'aide d'un octree

Pour accélérer le maillage des zones de chevauchement, on construit un octree dans lequel seront insérées les triangles du modèle. En effet, on aura besoin d'un moyen efficace pour trouver l'arête du maillage la plus proche d'un point 3D donné. Les voxels de cet octree ont la particularité que tout triangle est inclu entièrement dans le voxel qui le contient. Le voxels seront donc agrandis pour englober les sommets des triangles qu'il contient. Cela évite de chercher des triangles ou des arêtes dans plus d'un voxel à la fois.

6.2 Maillage d'une grille réguliere

Le maillage se fait un projecteur à la fois. Le premier projecteur est maillé selon l'ordre naturel de son nuage de points, c'est-à-dire selon l'ordre des pixels de son image.

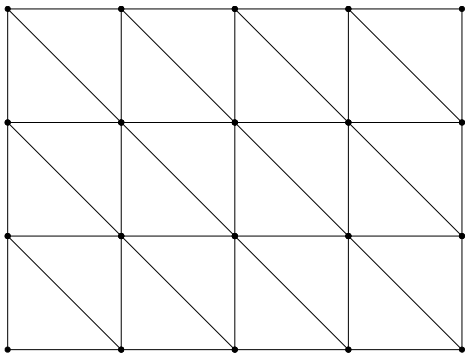


Figure 6.1. Maillage régulier

Pour obtenir un maillage régulier comme dans la figure 6.1, on parcourt chaque coordonnée (x, y) du projecteur. Pour chaque (x, y), on obtient le point 3D correspondant $\mathbf{P}_i(x, y)$, ainsi que trois voisins immédiats $\mathbf{P}_i(x + 1, y)$, $\mathbf{P}_i(x, y + 1)$, $\mathbf{P}_i(x + 1, y + 1)$. Si ces quatre points existent, on forme deux triangles, l'un connectant les points $\mathbf{P}_i(x, y)$, $\mathbf{P}_i(x+1, y)$, $\mathbf{P}_i(x+1, y+1)$, et l'autre les points $\mathbf{P}_i(x, y)$,

$\mathbf{P}_i(x+1, y+1)$, $\mathbf{P}_i(x, y+1)$. Si seulement trois de ces points existent, alors on forme un triangle avec ceux-ci. Si moins de trois de ces points existent, on ne peut évidemment pas former de triangle. Le ou les triangles résultants sont insérés dans le voxel dans lequel se trouve le point original $\mathbf{P}_i(x, y)$. Le voxel en question est agrandi au besoin pour accommoder les autres points du triangle. Cela cause évidemment un léger chevauchement entre les voxels adjacents, mais lorsqu'on insérera de nouveaux points dans le maillage, il sera important que chaque triangle d'un voxel y soit inclus entièrement.

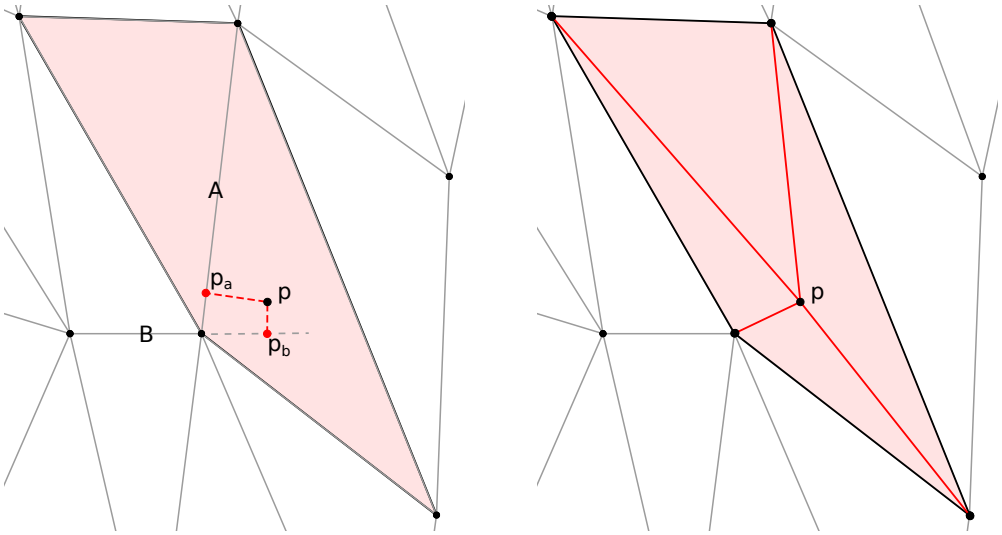
Une fois le premier projecteur ainsi maillé, on peut procéder à l'insertion des points autres projecteurs dans le maillage.

6.3 Zones de chevauchement

Après le maillage du premier projecteur, on veut insérer dans le maillage existant les points 3D de chaque autre projecteur. Chaque nouveau point se situant dans la zone de chevauchement entre deux projecteurs est inséré en modifiant les triangles avoisinants qui existent déjà. On cherche donc les deux triangles adjacents les plus appropriés (pas nécessairement les plus proches). On va couper l'arête commune entre les deux triangles trouvés et former quatre nouveaux triangles. Ça donne une surface plus lisse que de trouver simplement un seul triangle dans le centre duquel on insère le point en faisant trois nouveaux triangles (voir figure 6.2). Chaque nouveau triangle est inséré dans le voxel approprié.

Pour trouver les triangles qui doivent être *brisés*, on cherche parmi les arêtes des triangles du voisinage du point à insérer celle qui lui est la plus proche (voir figure 6.2). La distance d entre un point $\mathbf{p}$ et une arête $(\mathbf{a}, \mathbf{b})$ est définie par la distance minimale entre ce point et les points de la droite formée par l'arête:

$$\begin{aligned} \mathbf{n} &= \frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|} \\ d &= \|\mathbf{p}-\mathbf{a}-\mathbf{n}\,((\mathbf{p}-\mathbf{a})\cdot\mathbf{n})\| \end{aligned}$$



(a) (b)

Figure 6.2. Insertion du point $\mathbf{p}$ dans le maillage. Même si la distance entre $\mathbf{p}$ et $\mathbf{p}_b$ est la plus courte, on choisit de briser les triangles le long de l'arête A parce que $\mathbf{p}_a$ se situe sur celle-ci.

Toutes les arêtes ne sont pas considérées. Une arête est ignorée dans les cas suivants:

- Le point le plus près de $\mathbf{p}$ sur la droite ne se situe pas entre les extrémités de l'arête ($\mathbf{a}$ et $\mathbf{b}$), c.-à-d. si $0 \leq (\mathbf{p} - \mathbf{a}) \cdot \mathbf{n} \leq \|\mathbf{b} - \mathbf{a}\|$ n'est pas satisfaite.
- La distance d est supérieure à la longueur de l'arête $\|\mathbf{b} - \mathbf{a}\|$,
- L'arête ne fait partie que d'un seul triangle.

Si l'arête est valide, on détermine si les quatre nouveaux triangles potentiels sont bien formés. Pour ce faire, on fait le produit scalaire entre la normale de chaque triangle potentiel et la moyenne des normales des deux triangles originaux. Si ces produits n'ont pas tous le même signe, on peut décréter que la pyramide résultante est dégénérée et doit être rejetée.

Cette vérification est particulièrement importante pour les points qui tombent tout juste à l'extérieur de la zone de chevauchement, comme l'illustre la figure 6.3.

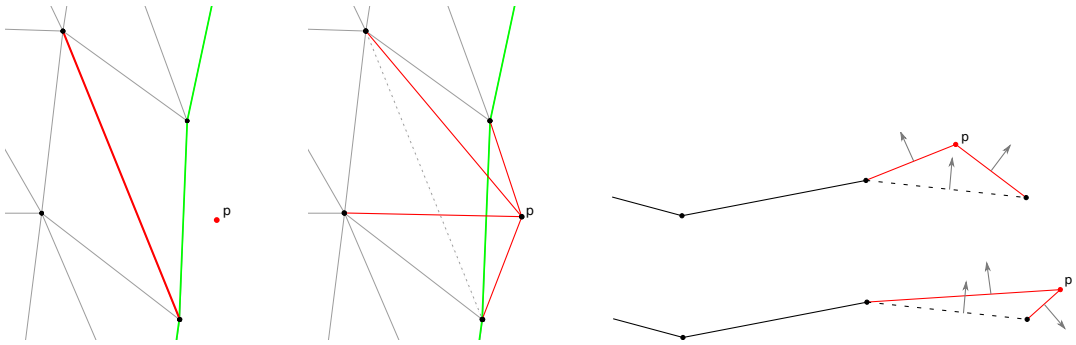


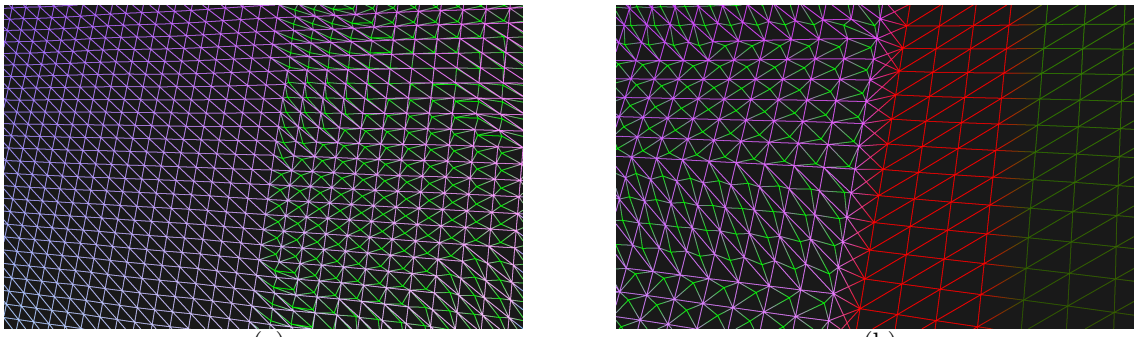
Figure 6.3. Insertion d'un point extérieur. (a) point à insérer. (b) résulats de l'insertion régulière. Le maillage obtenu est dégénéré. (c) vue de profil d'une insertion acceptable (en haut) et d'une rejetée (en bas). On note l'inversion de la normale dans le cas rejeté.

Un point tel qu'illustré dans la figure 6.3 ne doit donc pas être inséré dans le maillage de cette façon car il ne se trouve pas dans la zone de chevauchement. On va plutôt le mailler régulièrement comme dans la section précédente, en faisant attention de ne le connecter qu'à des points qui n'ont pas déjà été connectés. On se retrouve donc avec un projecteur dont les points ont tous étés maillés sauf pour la bordure entre la zone de chevauchement et la zone de maillage régulière.

6.4 Bordures

Pendant l'insertion des points du nouveau projecteur dans la zone de chevauchement, on étiquette chaque point voisin du point inséré comme bordure potentielle. Au fil des insertions, la bordure se propage jusqu'à ce que la zone de chevauchement soit complétée.

Avant de fusionner les points d'un projecteur avec le maillage existant, on compte



(a) (b)

Figure 6.4. Zone de chevauchement maillée. Les points en vert ont été insérés dans le maillage. On peut aussi voir dans l'image (b) la bordure entre le nouveau maillage régulier et le reste du maillage.

le nombre de triangles connectés à chaque point du maillage. Comme le maillage régulier connecte chaque points à six triangles, tout point connecté à moins de six triangles est étiqueté comme faisant partie de la bordure. Le maillage régulier tel que décrit précédemment possède donc une propriété importante: tous les points ne se situant pas sur la bordure du maillage sont connecté à exactement six points voisins. Quand de nouveaux points sont insérés dans les zones de chevauchement pour former de nouveaux triangles, cette propriété n'est plus respectée.

Étant donnée la disposition des projecteurs suggérée par la géométrie de l'écran, on n'a pas à traiter de cas où plus de deux projecteurs se chevauchent au même endroit. Si ce devait être le cas, un autre critère de sélection de la bordure devrait être utilisé.

Donc pour les points du nouveau projecteur, de même que pour les points du maillage existant, on trouve des listes de points consécutifs le long des bordures précédemment étiquetées. Pour ce faire, on cherche parmi les points de bordures les points qui se trouvent à une extrémité d'une bordure, c.-à-d. les points de bordure qui n'ont qu'un seul voisin également étiqueté comme bordure.

Ensuite, pour chaque point de bordure du nouveau projecteur, on cherche le point

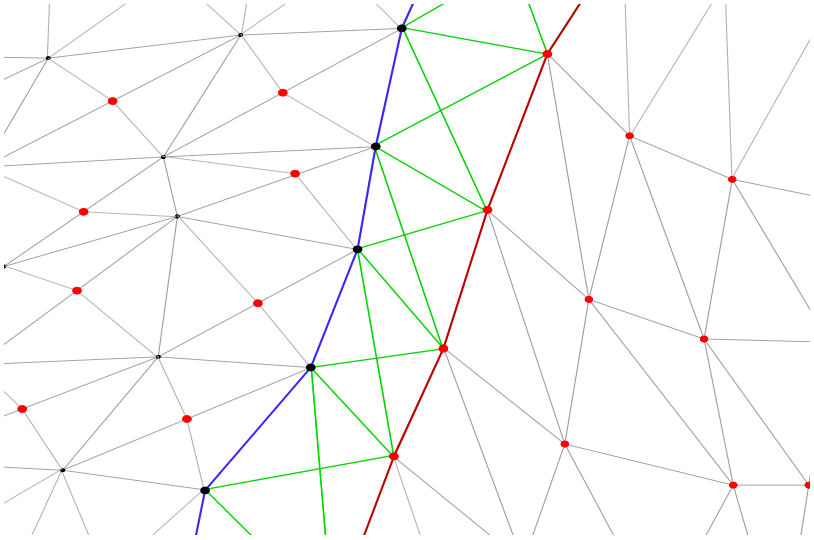


Figure 6.5. Maillage des bordures de la zone de chevauchement. On a ici en bleu la bordure du maillage des projecteurs jusqu'ici, et en rouge, la bordure du maillage régulier des points du nouveau projecteur qui n'on pas été précédemment insérés.

le plus proche parmi les points de bordure du maillage existant. On fait de même pour les points du maillage par rapport aux points du nouveau projecteur.

Pour connecter les deux bordures, on parcourt chaque liste de points et on connecte chaque paire de points voisins d'un côté de la bordure avec les points correspondants de l'autre côté, comme on peut le voir dans la figure 6.5. Certains triangles peuvent être enchevêtrés, mais cela n'est d'aucune importance car le modèle 3D résultant n'est pas destiné à être affiché tel quel. On va plutôt calculer des distances géodésiques sur celui-ci, et la présence de quelques triangles incorrectement connectés ne gêne aucunement la suite des choses.

Chapitre 7

PARAMÉTRAGE DE TEXTURE

On dispose à présent d'un modèle 3D complet. Il reste encore à associer une coordonnée de texture à chaque point du maillage, car celui-ci représente une surface rectangulaire déformée. Ce paramétrage pourrait se faire à l'aide d'un logiciel d'édition 3D tel que Blender [1]. Pour un modèle 3D quelconque, c'est en général une option viable, mais pas dans notre cas. Les méthodes de paramétrage automatique de ces logiciels ne s'intéressent pas a un mappage physiquement réaliste, plutôt que simplement valide. De plus, notre modèle possède typiquement des millions de polygones, ce qui rend les algorithmes classiques de systèmes masse-ressort inutilisables [7].

Or, on a l'avantage d'avoir un *a priori* qui nous permet d'automatiser une grande partie du processus. Le modèle de l'écran est d'une forme contrainte, soit un carré déformé. On va dégager un mappage qui est plus proche possible de la réalité physique d'une surface étirée.

Dans ce chapitre, on propose une nouvelle approche pour le paramétrage automatique de textures carrées attachées à un modèle 3D d'un carré déformé.

7.1 Algorithme d'approximation de la distance géodésique

On va considérer le maillage obtenu jusqu'ici comme étant un graphe. On va calculer des distances géodésiques sur ce graphe à partir de ses quatre côtés et utiliser celles-ci pour obtenir les paramètres (u, v) de chaque point du maillage. Il est impossible d'obtenir les distances exactes mais seulement une approximation de celles-ci. Pour quelles soient exactes, il faudrait disposer d'un maillage infiniment dense.

Pour obtenir la meilleure approximation possible, on va subdiviser les arcs en ajoutant des noeuds intermédiaires le long de ceux-ci. Pour chaque triangle du maillage, on va connecter entre eux ces noeuds intermédiaires pour former de nouveaux arcs qui permettront de suivre un chemin le plus direct possible entre deux points du maillage [9].

Sélection des points de départ

C'est la seule véritable intervention manuelle de tout le projet, mise à part le positionnement des caméras. On cherche ici à délimiter la véritable zone de projection dans le modèle 3D. On obtient cette zone en sélectionnant des points qui représente les 4 cotés du quadrilatère (tordu) de projection. Ces points deviendront les noeuds de départ de l'algorithme de calcul des distances géodésiques.

On choisit quelques points sur le maillage correspondant à la frontière que l'on veut définir. Ces points sont utilisés comme points de contrôle pour une courbe NURBS $\mathbf{C}_s$.

On définit une autre courbe NURBS $\mathbf{C}_n$ dans l'espace des normales avec les normales des points de contrôle sélectionnés. On parcourt la courbe $\mathbf{C}_s$ avec un échantillonage raisonnablement élevée et on intersecte chaque vecteur ($\mathbf{C}_s(i)$, $\mathbf{C}_s(i) + \mathbf{C}_n(i)$) avec le maillage. On ajoute les 3 points de chaque triangle du maillage ainsi intersecté dans la liste de points de départ pour l'algorithme de distances géodésiques. La distance géodésique initiale des points de départ est égale à la distance minimale de ces points avec les points d'intersection dans le maillage.

On fait ainsi pour le haut, le bas, et les côtés gauche et droite du modèle.

Initialisation

Commençons par définir quatre structures de données, soit des tableaux de noeuds **N**, de triangles **T** et d'arcs **A**, et un tas (ou *heap*) d'arcs H avec la longueur de l'arc

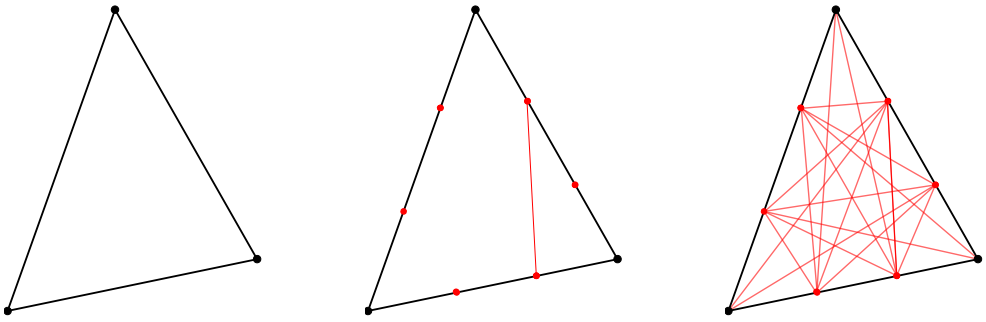
comme clé.

On remplit **N** avec les points 3D du maillage. On assure ensuite la connectivité du graphe en remplissant **T** avec les triangles qui connectent les noeuds de **N**. **A** est formé de l'ensemble des arcs des triangles. Chaque arc est directionnel, donc entré deux fois, l'un pointant vers l'autre. Chaque arc pointe vers les deux noeuds qui le forme ainsi que le ou les triangles desquels il fait partie.

Comme on cherche la distance géodésique de tous les points du maillage à partir d'un ou plusieurs points de départ, on marque les points de départ comme étant *visités*.

Subdivision de triangles

On utilise les noeuds *visités* pour subdiviser les triangles auxquels ils sont connectés. La subdivision d'un triangle commence par la subdivision de chacun de ses arcs en ajoutant un nombre n_{split} de noeuds intermédiaires (noeuds de Steiner) le long de l'arc pour en faire $n_{split} + 1$ arcs de longueurs égales. On connecte ensuite chaque noeud du triangle original à chacun des nouveaux noeuds intermédiaires (voir figure 7.1). Chaque noeud intermédiaire est ajouté à **N** et est étiqueté comme tel, pour qu'il

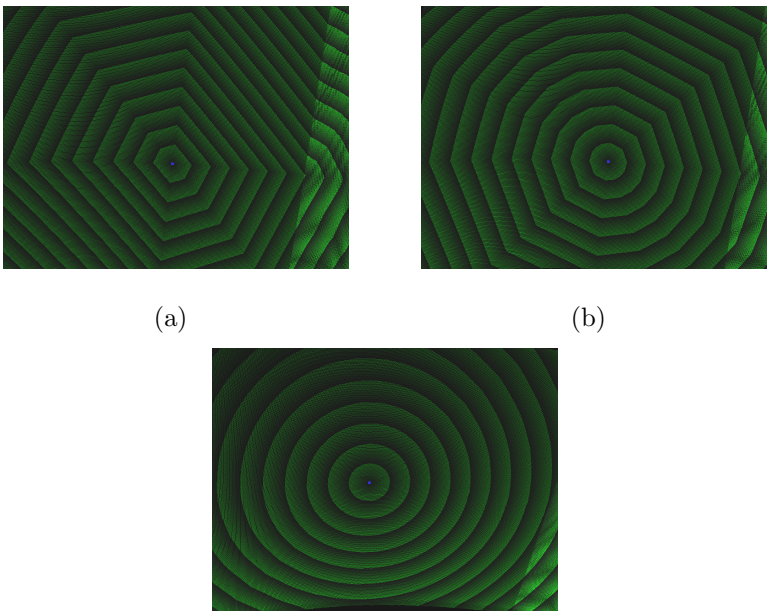


(a) (b) (c)

Figure 7.1. Subdivision d'un triangle pour $n_{split} = 2$.

puisse être éliminé du graphe lorsque il ne sera plus utile. Chaque arc intermédiaire est

ajouté à **A**. Si il relie un noeud non *visité* à un noeud *visité*, alors il est aussi ajouté au tas **H**, avec comme clé sa longueur additionné de la distance totale parcourue jusqu'au noeud *visité*.



(a) (b)

(c)

Figure 7.2. Propagation des distances géodésiques à partir d'un seul point pour différents nombres de noeuds intermédiaires. (a) $n_{split} = 0$, (b) $n_{split} = 1$, (c) $n_{split} = 7$

On voit dans la figure 7.2 l'effet du nombre de noeuds de Steiner ajoutés à chaque arc. Pour $n_{split} = 0$, c'est-à-dire aucune subdivision des arcs, on voit que les distances géodésiques calculées suivent la forme du maillage régulier du modèle. Or, en augmentant le nombre de noeuds de Steiner, on remarque que les distances géodésiques s'approchent de la vraie distance, qui dans ce cas-ci est circulaire autour du point de départ. Ainsi, l'effet du maillage est de moins en moins perceptible. En pratique, $n_{split} = 5$ ou 7 est suffisant.

Propagation

A chaque itération, on sélectionne en retirant du tas **H** l'arc dont la distance totale d'un point de départ jusqu'à son noeud de destination est la plus courte. Si ce noeud a déjà été visité, on l'ignore et on continue car sa distance géodésique est déjà minimale. Si il n'est pas un noeud intermédiaire, alors on subdivise les triangles auxquels il appartient, si ils ne sont pas déjà subdivisés. On ajoute ensuite tous les arcs sortant du noeud destination au tas **H**.

Avant de procéder à l'itération suivante, on élimine du tas tous les arcs intermédiaires que l'on considère inutiles, c'est-à-dire qui relient deux noeuds déjà visités. En effet, si on conservait ces arcs, le tas deviendrait rapidement très lourd. Aussi, dans le cas où l'un de ces arcs serait sélectionné comme étant le plus court, la distance géodésique du noeud destination resterait inchangée, parce que la contribution d'un tel arc ne pourrait plus donner une distance plus petite que celles déjà obtenues.

L'algorithme itère tant qu'il reste des arcs dans le tas. Une fois le tas vide, on sait que le maillage a été traité en son entièreté et qu'on a assigné à chaque point du modèle 3D sa distance géodésique avec les points de départ.

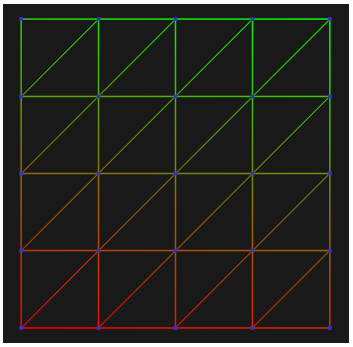
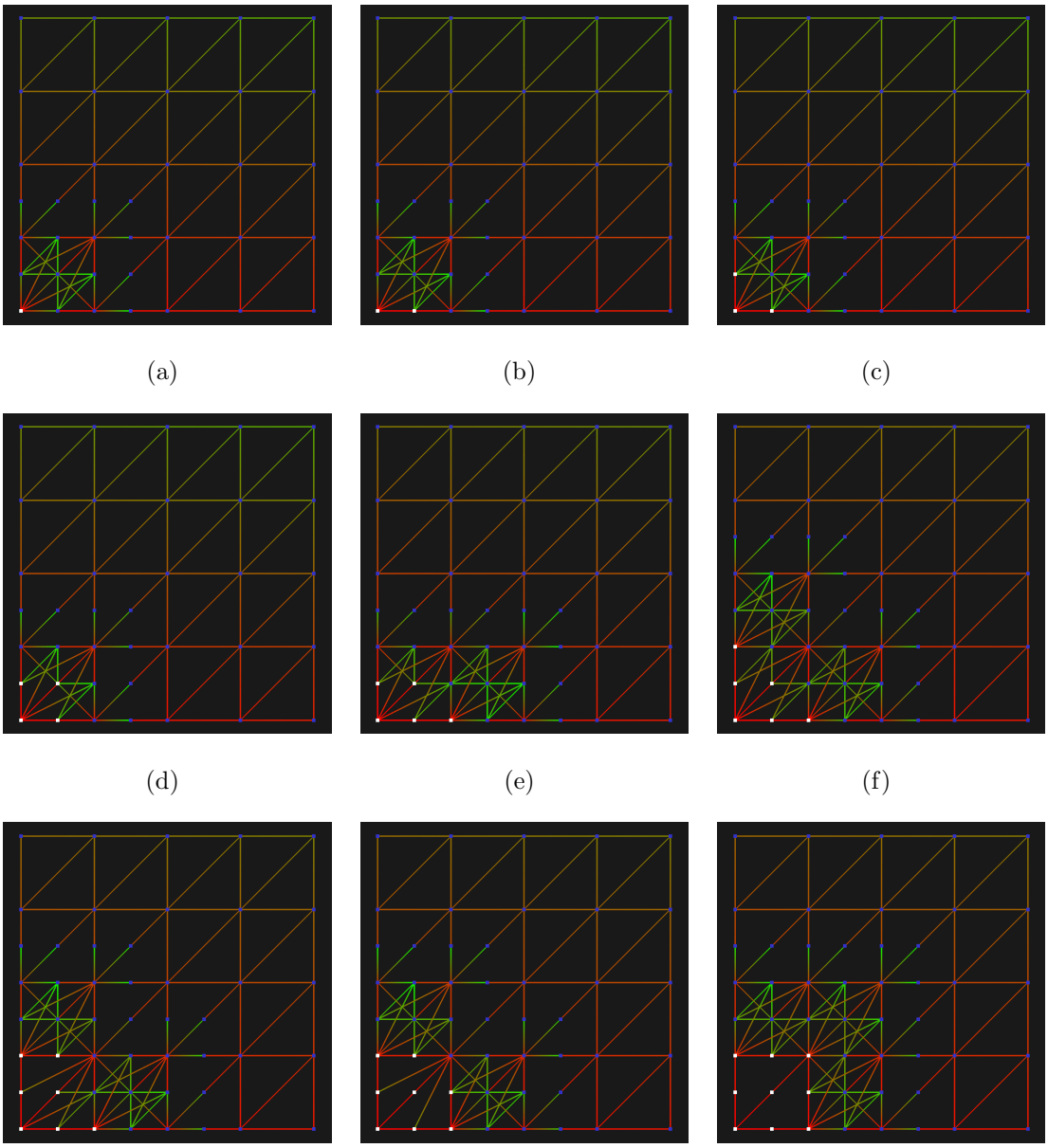


Figure 7.3. Modèle simplifié pour illustrer la propagation de l'algorithme.

Dans la figure 7.4, on peut voir l'évolution de l'algorithme à partir du début. Les noeuds pour lesquels la distance a été calculée sont identifiés en blanc. Les noeuds



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 7.4. Propagation de l'algorithme d'approximation des distances géodésiques, Exemple pour une seule subdivision ($n_{split} = 1$).

non-visités sont en bleu. Le noeud en bas a gauche étant désigné comme noeud de départ, on ajoute les noeuds de Steiner sur les arcs composant les deux triangles qui

le contiennent, ainsi que sur tous les arcs sortant des noeuds de ces triangles. Les noeuds des deux triangles sont connectés et les nouveaux arcs sont ajoutés au tas.

À chaque itération, l'arc le plus court est retiré du tas et le noeud est étiqueté comme visité avec une distance minimale. Les arcs intermédiaires ajoutés précédemment qui rejoignent ce noeud sont aussi éliminés du tas, car la distance minimale pour ce noeud a déjà été déterminée. Lorsqu'un noeud du maillage original est visité, on subdivise les triangles qui s'y rattachent et qui n'ont pas déjà été subdivisés (voir figure 7.4(e)).

À la huitième itération (figure 7.4(i)), on voit que le dernier noeud des deux premiers triangles vient d'être visité. Les arcs intermédiaires dans ces deux triangles ont tous étés retirés. Les noeuds de Steiner, maintenant inutiles car ne participant plus aux calculs de distance, seront eux aussi éliminés du graphe.

Temps d'exécution

Pour un modèle ayant 1 millions de points, 1.9 millions de triangles donc 5.7 millions d'arcs, avec $n_{split} = 7$, sans optimisation multi-processeur, le temps de calcul est de 12 minutes pour un seul ensemble de points de départ, donc 48 minutes pour trouver les distances géodésiques dans les quatre directions.

Avec $n_{split} = 5$, l'approximation est un peu moins bonne, mais le temps total d'exécution passe à 22 minutes pour les quatre directions du même modèle.

7.2 Calcul du paramétrage

Chaque point du modèle a donc un distance géodésique pour chaque ensemble de points de départ. Si l'écran n'était qu'un rectangle déformé mais non étiré, on pourrait très bien n'utiliser que les valeurs normalisées des distances géodésique pour les points de départ de gauche et du haut. Cela est dû a fait que l'algorithme effectue une propagation rigide sur le maillage. Lorsque l'écran peut subir des déformations non rigides

(comme des étirements), alors on observera des différences entre des paramétrages effectués dans des directions opposées. Dans ce cas les paramétrages doivent être combinés de façon à obtenir un paramétrage souhaité.

Pour un point i dans le maillage, on a:

$$\begin{aligned} d_{gauche} &= \mathbf{D}_{gauche}(i)/max(\mathbf{D}_{gauche}) \\ d_{droite} &= \mathbf{D}_{droite}(i)/max(\mathbf{D}_{droite}) \\ d_{haut} &= \mathbf{D}_{haut}(i)/max(\mathbf{D}_{haut}) \\ d_{bas} &= \mathbf{D}_{bas}(i)/max(\mathbf{D}_{bas}) \end{aligned}$$

On a donc, pour chaque dimension du paramétrage, deux distances entre zéro et

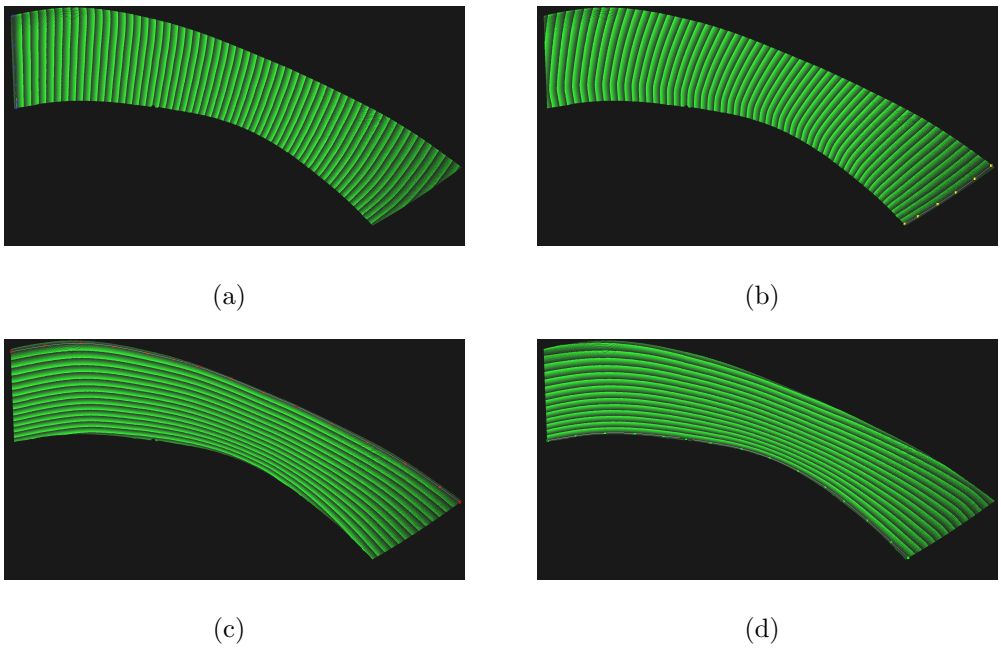


Figure 7.5. Modèle sur lequel sont illustrées les isolignes des distances géodésiques calculées à partir des quatre directions.

un. Ces distance sont re-normalisées entre elles de façon à obtenir une seule distance

entre zéro et un:

$$(u, v) = \left(\frac{d_{gauche}}{d_{gauche} + d_{droite}}, \frac{d_{haut}}{d_{haut} + d_{bas}} \right) \tag{7.1}$$

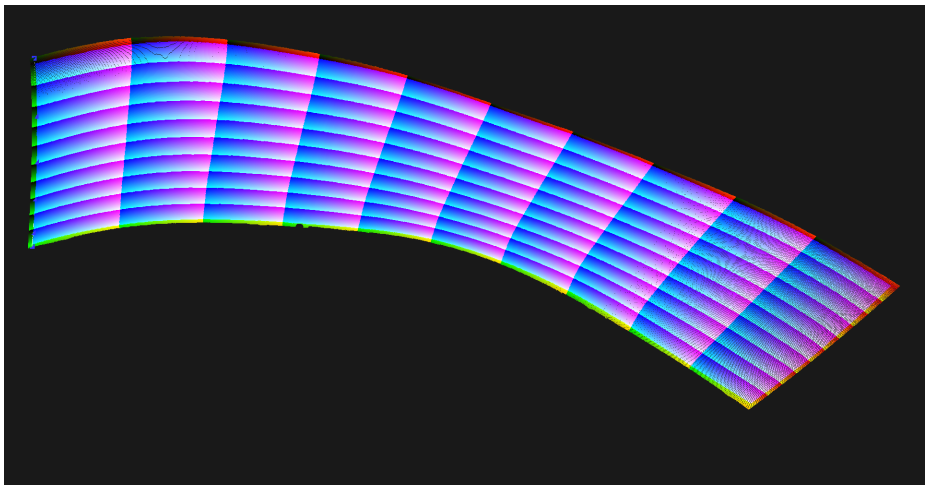
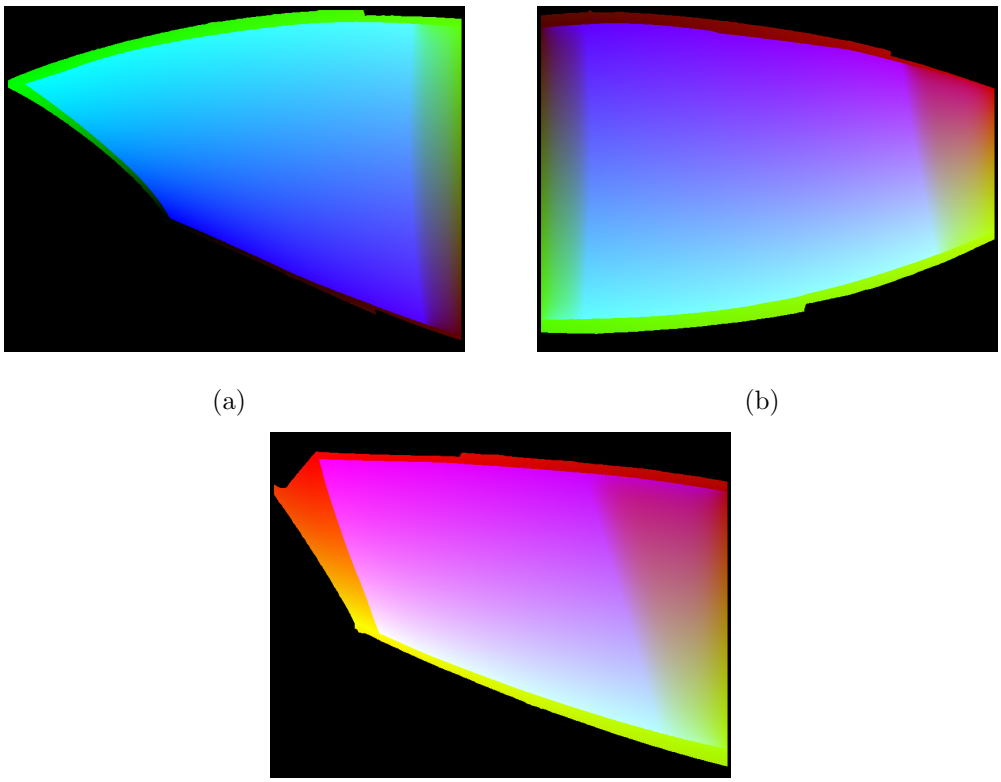


Figure 7.6. Modèle sur lequel on peut voir le paramétrage (u,v). La région bleutée est la zone délimitée par les quatre ensembles de points de départ.

7.3 Cartes de déformation

Nous disposons maintenant d'un modèle 3D paramétré. Puisque nous souhaitons calculer une carte de déformation pour chacun des projecteurs, on doit transférer ce paramétrage vers leurs images. On sait que la triangulation utilisée pour créer le modèle 3D est basée sur des mises en correspondances qui incluent toujours un projecteur. On dispose donc pour chaque pixel de l'image d'un projecteur d'un index qui nous permet de connaître le point 3D du modèle qui lui correspond. On va se servir de cet index pour construire les cartes de déformation, ou *lookup tables*, sous forme d'images RGB. On note que pour garantir qu'on a un seul index pour chaque pixel, les reconstructions multiples sont fusionnées lors de la création du modèle 3D (voir section 5.6).

Pour chaque pixel d'un projecteur, on retrouve le point 3D dans le modèle et on encode ses coordonnées (u, v) obtenue à l'équation 7.1 dans les canaux rouge et vert de l'image, ainsi que la valeur de mixage dans le canal bleu (voir figure 7.7).



(a) (b)

(c)

Figure 7.7. Tables de conversion pour une installation à trois projecteurs.

Chapitre 8

RÉSULTATS

Ce chapitre présente les résultats obtenus lors de la réalisation d'un montage expérimental d'un écran souple en spandex de 12.5 par 1.5 mètres, déformé à l'aide de câbles tenseurs, illuminé par trois projecteurs, et reconstruit à partir de quatre points de vue. Les trois projecteurs haute définition sont réglés à une résolution

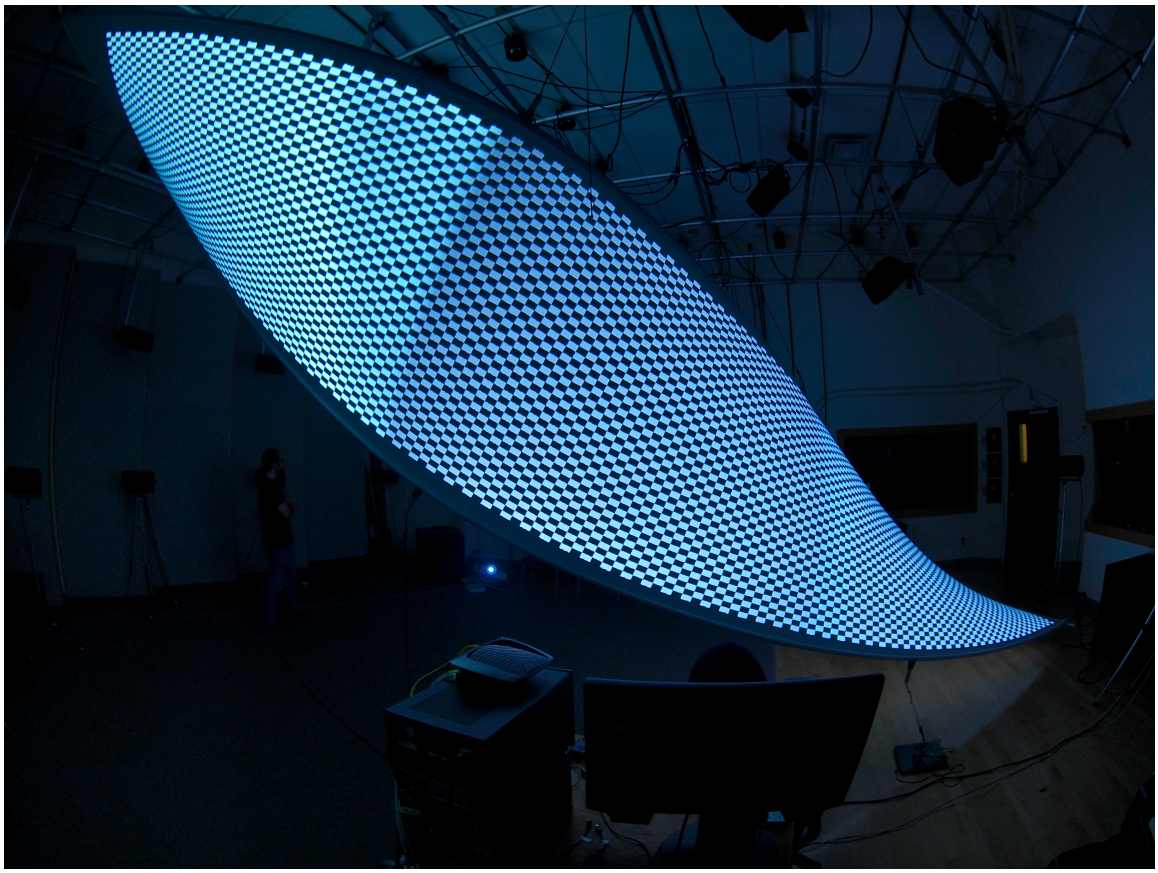


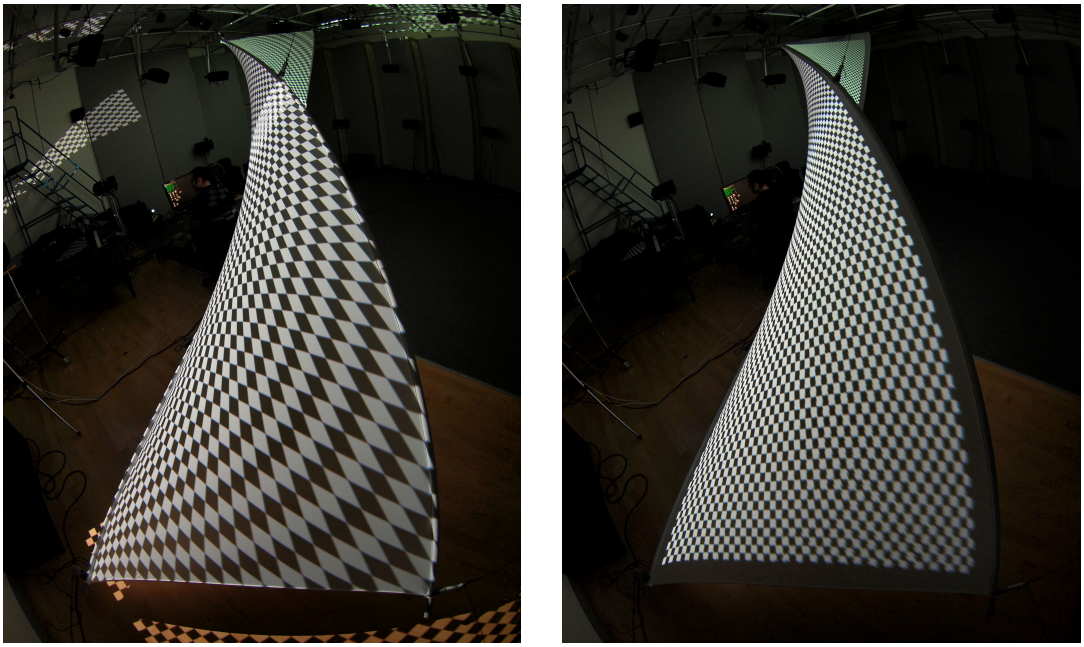
Figure 8.1. Projection sur l'écran d'un damier pour mettre en évidence l'alignement de la projection.

de 800 $\times$ 600 pixels. À partir des quatre points de vue, on a obtenu six mises en correspondance partielles, deux par projecteur. Le maillage résultant comporte 944880 sommets et 1887738 triangles. Dans cette première image (figure 8.1), on a projeté un damier régulier pour illustrer la qualité de l'alignement de la projection par rapport à l'écran. On constate en effet que la projection donne à l'observateur l'impression que l'image est *collée* sur un écran plat qui aurait été par la suite déformé. On remarque aussi une légère démarcation entre les intensités des projecteurs dans les zones de chevauchement. Cela est dû au fait que le gamma des projecteurs n'a pas été calibré, et que l'algorithme d'alignement photométrique tel qu'implanté lors de cette expérimentation n'ajuste pas l'intensité totale de la projection, mais seulement dans les zones de chevauchement.

L'image suivante (figure 8.2) nous montre le même damier que précédemment projeté sur l'écran. À gauche, le damier est projeté sans utiliser les cartes de déformation. À droite, on voit du même point de vue le damier déformé comme dans la figure 8.1. On peut observer dans l'image de gauche que le damier devient progressivement plus flou vers le bord, phénomène dû à l'angle d'incidence de la projection qui devient de plus en plus rasant à cet endroit.

La figure 8.3 illustre avec plus de détails l'effet de l'alignement photométrique. La main se trouve dans le faisceau d'un seul des deux projecteurs qui se chevauchent au centre de l'image. On voit clairement le fondu progressif vers le noir de la projection du côté gauche. On peut également remarquer les carrés du damier qui restent bien alignés dans la zone de chevauchement. Comme pour la première image (figure 8.1), l'intensité totale dans la zone de chevauchement diminue légèrement à cause du gamma des projecteurs, qui n'ont pas été calibrés.

Enfin, les figures 8.4 et 8.5 présentent l'écran sur lequel on projette du contenu autre que des damiers. Visuellement, on perçoit une projection uniforme et cohérente.



(a) (b)

Figure 8.2. Projection sur l'écran d'un damier (a) sans déformation, et (b) avec déformation.

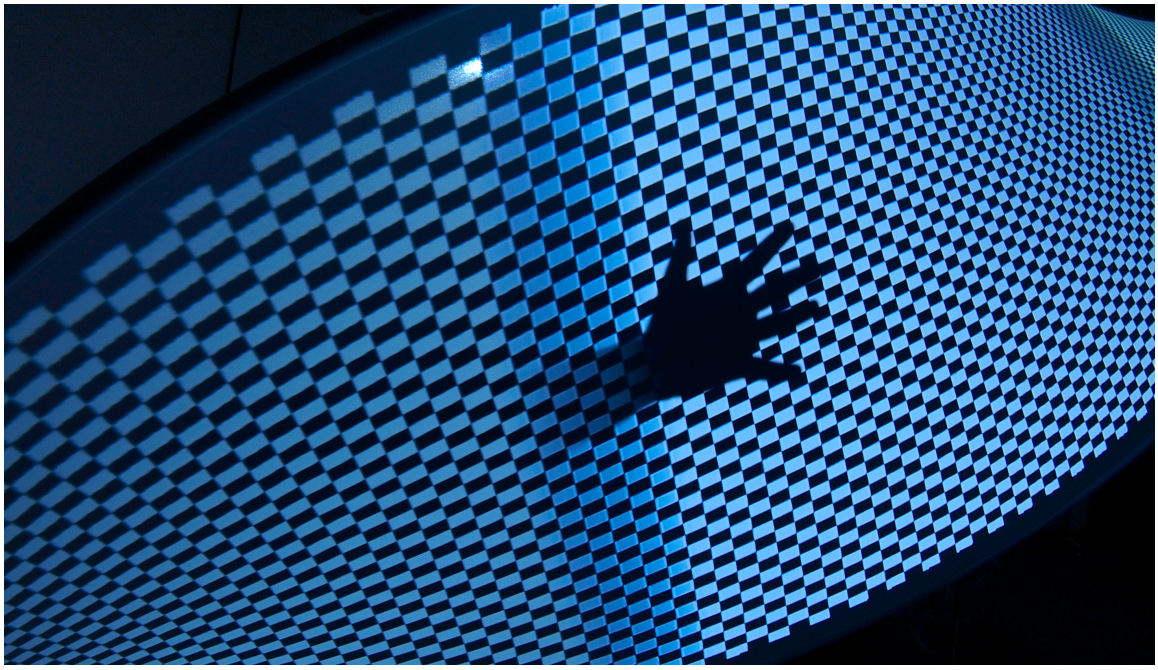


Figure 8.3. Démonstration de l'effet de l'alignement photométrique.

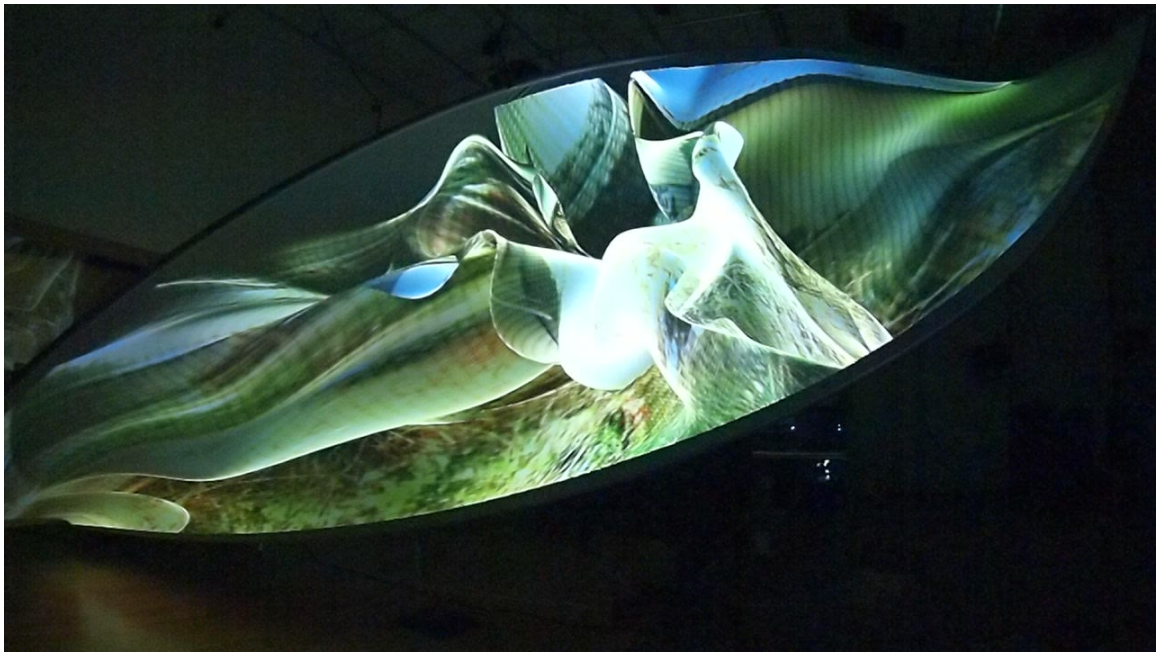


Figure 8.4. Projection de contenu sur l'écran

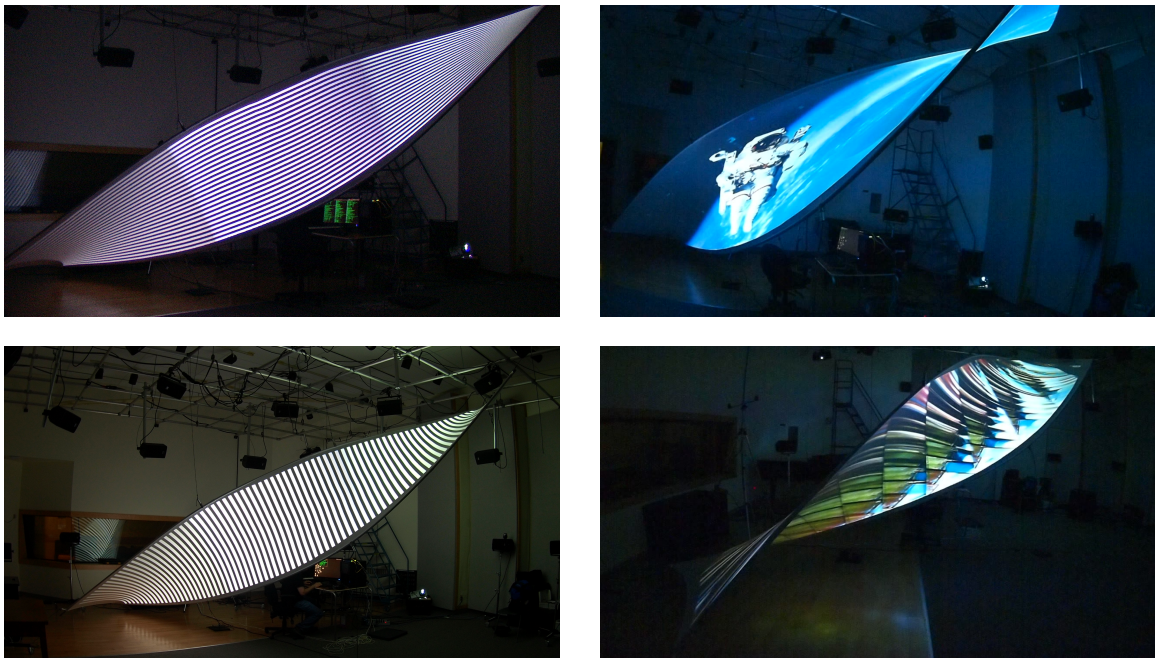


Figure 8.5. Résultats variés.

Chapitre 9

CONCLUSION

Le but de ce mémoire était de présenter une méthode complète pour la réalisation de projections sur des écrans de formes arbitraires. En particulier, on s'intéressait aux écrans dont la forme s'apparente à un rectangle déformé. La méthode présentée coordonne deux caméra et un certain nombre de projecteurs de façon à obtenir une reconstruction 3D de l'écran pour ensuite calculer la déformation à appliquer aux projections.

Cette méthode a plusieurs avantages:

- On n'a qu'à calibrer un système de caméras stéréo. On évite ainsi de calibrer les projecteurs.
- L'alignement photométrique, c.-à-d. le mixage entre les intensités des projecteurs dans les zones ou ils se chevauchent, se fait de façon automatique.
- La mise en correspondance par lumière non structurée établit le lien entre les caméras vers les projecteurs et vice versa. On obtient donc une correspondance stéréo pour chaque point de vue.
- Les reconstructions partielles obtenues de différents points de vue sont alignées et fusionnées automatiquement pour obtenir un modèle 3d complet de l'écran sous la forme d'un nuage de points.
- Le maillage du nuage de points se fait aussi automatiquement.
- Le modèle maillé respecte la topologie rectangulaire de l'écran, et son paramétrage en coordonnées texture ne requiert qu'une intervention humaine minimale.

On a introduit une nouvelle approche pour le paramétrage de modèles 3D se basant sur le calcul efficace de distances géodésiques sur des maillages. L'usager n'a qu'à délimiter manuellement le contour de la zone de projection sur le modèle. Le paramétrage final est calculé en utilisant les distances obtenues pour chaque point du modèle. Jusqu'à maintenant, les méthodes existante ne permettaient pas de paramétrer des modèles ayant plus d'un million de points.

La méthode de paramétrage à l'aide de distances géodésiques présentée ne fonctionne que sur des modèles rectangulaires déformées. Dans le futur, il serait utile d'étendre la méthode à d'autres formes d'écrans, tels un cylindre, un dôme, ou un ruban de Möbius. La propagation de notre algorithme se fait de façon rectiligne, on ne peut l'utiliser directement pour trouver un paramétrage d'une surface qui n'a pas quatre côtés.

D'autres limitations du système pourraient faire l'objet d'améliorations. D'une part, l'alignement des reconstructions partielles se fait de manière incrémentale. Il n'est pas possible de reconstruire un écran de topologie cylindrique en procédant de cette façon car lorsque le dernier morceau d'écran est àjouté au modèle, il n'est pas garanti qu'il soit possible de l'aligner avec la fin et le début du cylindre. Il serait alors pertinant de repenser le processus pour aligner tous les morceaux en même temps.

Dans le but de simplifier encore le processus de mise en correspondance, on pourrait substituer le système de caméras stéréoscopiques pour une caméra unique et calibrer la position et l'orientation des projecteurs à l'aide de la mise en correspondance directement. Il serait aussi intéressant de tester d'autres méthodes de calcul des distances géodésiques, en particulier pour obtenir des distances géodésiques exactes pour déterminer l'impact réel de leur effet sur la qualité du paramétrage.

Étant donné que la méthode présentée simplifie beaucoup la multi-projection, on s'attend à ce qu'elle trouve de nombreux usages pratiques en arts technologiques, en théâtre ou en projection architecturale.

RÉFÉRENCES

[1] Blender 3d content creation suite. `http://www.blender.org/`.

[2] Lighttwist. `http://vision3d.iro.umontreal.ca/en/projects/lighttwist/`.

[3] G. Bradski et A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[4] V. Couture, N. Martin, et S. Roy. Unstructured light scanning to overcome interreflections. Dans *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1895–1902. IEEE, 2011.

[5] V. Couture, N. Martin, et S. Roy. Subpixel unstructured light scanning. Dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (soumis).

[6] J. Draréni, S. Roy, et P. Sturm. Geometric video projector auto-calibration. Dans *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 39–46. IEEE, 2009.

[7] K. Hormann, B. Lévy, et A. Sheffer. Mesh parameterization: Theory and practice. Dans *SIGGRAPH 2007 Course Notes*, numéro 2, pages vi+115, San Diego, CA, Août 2007. ACM Press.

[8] S. Inokuchi, K. Sato, et F. Matsuda. Range imaging system for 3-d object recognition. Dans *International Conference on Pattern Recognition*, 1984.

[9] M. Lanthier, Maheshwari A., et J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. Dans *In 6th Annual Video Review of Computa-*

tional Geometry, Proc. 13th ACM Symp. Computational Geometry, pages 274–283. ACM Press, 1996.

[10] B.B. May, N.D. Cahill, et M.R. Rosen. Calibration of a multi-projector system for display on a cylindrical surface. Dans *Image Processing Workshop (WNYIPW), 2010 Western New York*, pages 6–9, 2010.

[11] H. K. Nishihara. PRISM: A practical real-time imaging stereo matcher. 1984.

[12] J. Nocedal. *Numerical optimization*. Springer, New York, 2006.

[13] M. Ogata, H. Wada, J. van Baar, et R. Raskar. A unified calibration method with a parametric approach for wide-field-of-view multiprojector displays. Dans *Virtual Reality Conference, 2009. VR 2009. IEEE*, pages 235–236, 2009.

[14] R. Raskar, M.S. Brown, Ruigang Yang, Wei-Chao Chen, G. Welch, H. Towles, B. Scales, et H. Fuchs. Multi-projector displays using camera-based registration. Dans *Visualization '99. Proceedings*, pages 161–522, 1999.

[15] R. Raskar et J. van Baar. P-153: Low-cost multi-projector curved screen displays. *SID Symposium Digest of Technical Papers*, 36(1):884–887, 2005.

[16] B. Sajadi et A. Majumder. Auto-calibration of cylindrical multi-projector systems. Dans *Virtual Reality Conference (VR), 2010 IEEE*, pages 155–162, 2010.

[17] B. Sajadi et A. Majumder. Autocalibrating tiled projectors on piecewise smooth vertically extruded surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 17(9):1209–1222, 2011.

[18] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1–10, 1966.

[19] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 1st edition. édition, Novembre 2010.

[20] F. Teubl, C. Kurashima, M. Cabral, et M. Zuffo. Fastfusion: A scalable multi-projector system. Dans *Virtual and Augmented Reality (SVR), 2012 14th Symposium on*, pages 26–35, 2012.