Université de Montréal

Introduction à la reconstruction 3D par lumière structurée

par

Nicolas Hurtubise

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

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

novembre, 2020

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

Introduction à la reconstruction 3D par lumière structurée

présenté par

Nicolas Hurtubise

a été évalué par un jury composé des personnes suivantes:

_____

Jean Meunier
(Professeur)

_____

Derek Nowrouzezahrai
(Professeur)

_____

Sébastien Roy
(Directeur)

Mémoire accepté le _____

RÉSUMÉ

Ce mémoire porte sur la reconstruction 3D active par la lumière structurée et se veut une introduction au travers d'une vue d'ensemble du domaine.

L'accent est mis sur une compréhension des différents aspects du scan 3D, en commençant par l'acquisition du matériel et en allant jusqu'au calcul d'un modèle 3D. Les défis techniques les plus fréquents dans un contexte de scan 3D avec de l'équipement à faible coût sont abordés, de façon à donner une idée générale des problèmes à anticiper à quelqu'un qui souhaiterait se lancer dans la conception ou l'utilisation de ce type de scanneur. Afin d'aider à peser les forces et faiblesses de chaque algorithme formant l'état de l'art actuel en lumière structurée, une revue des différentes stratégies existantes est également donnée.

L'étude du domaine a mené à une contribution sous la forme d'un article, qui a été intégré dans le chapitre 6. Cet article propose une amélioration en ce qui concerne la performance et la précision d'un algorithme de scan 3D existant et a été publié dans la *International Virtual Conference on 3D Vision* en novembre 2020.

Mots clés: reconstruction active, sous-pixel, discontinuités de profondeurs

ABSTRACT

This thesis focuses on active 3D reconstruction with structured light and is intended to be an introduction through an overview of the field.

It is focused on understanding the different aspects of a 3D scanner, from the acquisition of the material to the final mesh reconstruction. The most common technical difficulties that arise in the context of low-cost 3D scanning are addressed, to give a general idea of the potential technical problems that may occur when using a DIY 3D scanner built from scratch. To help assess the strengths and weaknesses of each algorithm of the current state of the art in structured light, a review of the different strategies is also proposed.

The study of this domain led to the contribution of a paper, presented here in its entirety in chapter 6. This paper describes an improvement of an existing algorithm, both in terms of performance and precision, and was published in the *International Virtual Conference on 3D Vision* in November 2020.

Keywords: active reconstruction, subpixel, depth-discontinuities

TABLE DES MATIÈRES

LISTE DES FIGURES

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, Sébastien Roy, pour son encadrement précieux tout au long de mon cheminement, pour son écoute, pour ses patientes explications, et surtout pour ses mille et une idées farfelues qui m'ont permi de voir le travail de recherche comme un travail extrêmement ludique et motivant.

Je tiens à remercier Chaima El-Asmi, Emir Chouchane et Pierre-André Brousseau pour leur précieuse aide et pour leur présence au lab tout au long de ma maîtrise. Un deuxième merci tout spécial à Pierre-André pour ses relectures et précieux conseils pendant la course à la publication.

Merci à Emma Parent Senez pour son amour et ses bons petits plats. Merci à Sulliman Aïad, Guillaume Riou, Abdelhakim Qbaich et à Gabriel Beauchamp pour avoir m'écouté radoter sur mes idées d'esquisses d'algorithmes à de nombreuses reprises. Merci à Audrey Champagne pour le dessin d'alpaga (2.1). Merci au Céramic Café, qui permet de se procurer des objets parfaitement lambertiens, idéaux pour tester et comparer différents algorithmes de lumière structurée. Merci encore une fois à mon directeur de recherche, Sébastien Roy, pour avoir tenté d'expliquer (sans trop de succès) à une serveuse du Céramic Café un peu confuse pourquoi il comptait acheter des objets juste comme ça, sans les peindre. Merci à la serveuse d'avoir hoché de la tête tout le long en faisant semblant de comprendre ce que c'était que de la réflectance lambertienne.

Merci aux personnes qui maintiennent des logiciels libres comme Debian, Emacs, git, TeX et LaTeX, sans qui ce mémoire aurait été écrit à la main, sur du papier.

Chapitre 1

INTRODUCTION

La reconstruction 3D est un vaste domaine dont les applications sont nombreuses. Du côté industriel, on peut penser par exemple à l'automatisation de l'assurance-qualité : elle permet de détecter automatiquement des défauts de fabrication dans des pièces manufacturées. Du côté médical, le scan 3D peut assister à la conception d'orthèses adaptées aux patients. D'un point de vue artistique, il permet d'intégrer fidèlement des objets et personnes du monde réel dans la création d'un film d'animation 3D ou d'un jeu vidéo.

La démocratisation des imprimantes 3D donne une raison de plus pour s'intéresser au sujet : le scanneur 3D est complémentaire à l'imprimante 3D. Si l'imprimante 3D ouvre la porte au téléchargement d'une statue antique pour la présenter dans son salon, le scanneur 3D permet à l'inverse de créer un modèle virtuel d'un objet physique existant, permettant ainsi sa copie et sa ré-impression à l'infini.

Figure 1.1. Un scanneur 3D dans l'imaginaire du grand public tient plus de la science-fiction que du réel

Malgré le fait que les imprimantes 3D soient de plus en plus connues du grand public, l'idée d'un scanneur 3D reste un concept quelque peu ésotérique. Parlez de scan 3D à quelqu'un, et vous verrez probablement votre interlocuteur s'imaginer une panoplie de lasers, d'appareils industriels de haut niveau ou encore carrément des systèmes dignes

d'oeuvres de science-fiction. Les principes derrière les scanneurs 3D les plus répandus restent pourtant relativement simples et sont réalisables avec de l'équipement à faible coût que beaucoup de gens possèdent déjà chez eux sans le réaliser.

1.1 Un scanneur 3D primitif : nos yeux

La reconstruction 3D est fondée sur les mêmes principes que la vision binoculaire chez un bon nombre d'animaux. En plus de donner un champ de vision plus large dans l'ensemble, posséder deux yeux plutôt qu'un seul est ce qui rend la vision stéréoscopique possible : observer deux fois la même scène depuis des angles différents permet de déduire les profondeurs des différents objets présents.

Une expérience très simple qui illustre ce principe est la suivante : placez votre doigt à quelques centimètres de votre visage et ouvrez un seul oeil à la fois : chaque oeil verra une image différente, donnant l'impression que le doigt se déplace de gauche à droite, selon l'oeil que vous ouvrez. Il y a un déplacement horizontal, que l'on appelle aussi *disparité*, entre les positions de votre doigt telles que perçues par chaque oeil. L'oeil gauche devrait observer votre doigt plus "*à droite*" dans son image que l'oeil droit.

Cette disparité diminue à mesure que l'on éloigne le doigt de nos yeux et atteindrait éventuellement zéro (si on avait des bras assez longs), avant de recommencer à augmenter, mais en sens inverse. En refaisant le même exercice, mais en regardant plutôt une montagne au loin, vous verrez probablement que la position relative gauche/droite de cette montagne dans chaque oeil est dans le sens contraire. L'ensemble des points de l'espace où la disparité atteint zéro est appelé l'*horoptère*.

Une autre façon d'illustrer ce phénomène est de considérer un *anaglyphe* tel que la Figure 1.2. Il s'agit d'une image "3D" constituée de deux points de vue en même temps : la vue pour l'oeil gauche est superposée à celle pour l'oeil droit en filtrant les couleurs de chacune. En regardant une telle image avec des lunettes bicolores, l'oeil gauche filtrant les couleurs pour garder seulement le rouge dans l'image et l'oeil droit pour garder seulement

le cyan, on obtient une image différente pour chaque oeil, ce qui permet de créer une illusion de profondeur. En regardant une telle image sans lunettes colorées, on peut voir les fantômes des régions dont les couleurs devraient être annulées pour l'un ou l'autre des yeux. Ces ombres cyan/rouges permettent de visualiser la relation entre la profondeur des objets et la disparité entre les positions obtenues dans les deux vues.

Figure 1.2. La disparité entre deux images est clairement visible sur des *anaglyphes*. **En considérant la position relative des ombres cyan vs rouge, on peut déduire que le cactus à droite de l'image (A) se situe** *avant l'horoptère* **(l'oeil gauche, qui utilise un filtre rouge, voit l'ombre rouge comme étant un bout de l'arrière-plan, et donc observe ce cactus plus à droite que l'oeil gauche), le cactus au milieu en haut (B) situe après l'horoptère (l'oeil gauche en cyan voit le cactus plus à gauche que l'oeil droit) et le cactus en bas de l'image (C) se situe à peu près sur l'horoptère du système (puisqu'il n'y a à peu près pas d'ombre visible).**

La disparité horizontale entre deux vues est une information critique pour percevoir

le monde en 3D et s'avère être un point de repère plutôt efficace pour notre cerveau.

1.2 Donner la vue 3D aux machines

Si on s'intéresse à faire faire l'analyse de scènes 3D automatiquement par des machines, on peut partir de ces mêmes principes et les appliquer avec des images capturées par des caméras. Quantitativement, la disparité peut être utilisée pour *trianguler* des points et obtenir une distance exacte. Si on connaît la distance qui sépare deux points de vue, et que l'on connaît la disparité entre chaque paire de points correspondant entre l'image gauche et l'image droite, on peut déduire les angles du triangle que forment l'objet et ses deux images. Tel qu'illustré dans la Figure 1.3, un peu de trigonométrie de niveau secondaire permet de mesurer la distance en trois dimensions[1].

Pour percevoir une scène en 3D, on devrait donc en théorie avoir besoin de : 1) deux points de vue d'une même scène, placés à des endroits précisément connus dans le monde; 2) une façon de déterminer quels sont les points correspondants entre les deux images. Un scanneur 3D pourrait donc être composé de (1) deux caméras *calibrées*, c'est-à-dire, dont la position et la physique optique sont connues, et de (2) un algorithme chargé de trouver automatiquement les correspondances entre les points des deux images. Si (1) est un problème en bonne partie résolu, (2) est beaucoup moins trivial à concevoir de façon robuste et reste un domaine de recherche encore actif à ce jour.

Une idée simple pour déterminer si deux points forment une correspondance serait de calculer la corrélation entre chaque région des deux images, par exemple à coups de blocs de 32×32 pixels : chaque point de l'image A sera associé au point de l'image B si les régions autour des points sont maximalement corrélées. Cette stratégie est illustrée dans la Figure 1.4.

Sur la scène présentée dans la Figure 1.4, cet algorithme donne des résultats raisonnablement acceptables. Les problèmes arrivent lorsqu'on considère d'autres scènes plus diffi-

[1] Dans un monde idéal, sans erreurs sur les mesures. Voir le chapitre 7 pour un peu plus de détails.

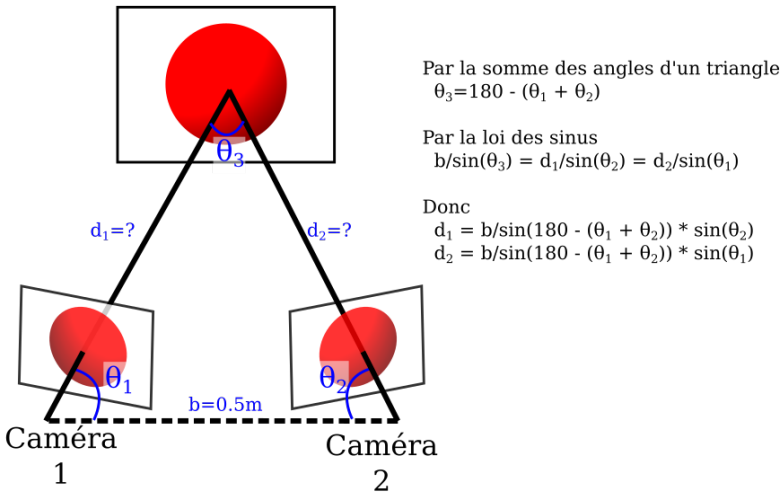

Figure 1.3. Triangulation : si les angles θ_1 et θ_2, formés dans le triangle qui relie les points correspondants dans les plans image au point de la scène, et la distance entre les deux caméras b sont des valeurs connues, on peut alors déduire la distance entre chaque point dans l'image d'une caméra et l'objet réel

Figure 1.4. Corrélation entre des régions 32×32 de deux points de vue différents pour la scène *Teddy* du jeu de données *Middlebury (2003)* [38]. *De gauche à droite*, région choisie dans la vue de droite, région correspondante dans la vue de gauche, et carte de corrélation entre la région choisie et les différentes positions possibles (blanc: forte corrélation, noir: peu de corrélation).

ciles. En choisissant d'utiliser la corrélation entre deux régions ici, on fait implicitement beaucoup d'hypothèses qui ne sont pas forcément vraies. D'abord, on fait l'hypothèse que l'objet a une réflexion *lambertienne*, donc que la couleur perçue pour un point est constante, peu importe l'angle d'observation. Ensuite, on fait également l'hypothèse que le ratio de pixels est le même entre les deux caméras, donc qu'une région de 32×32 pixels représente la même surface sur l'objet dans chaque point de vue, ce qui implique que les deux caméras ont à peu près la même résolution et ne sont pas trop éloignées l'une de l'autre. Finalement, l'hypothèse qui est probablement la plus forte ici est de croire que la scène scannée est nécessairement texturée : tenter de scanner un mur blanc avec cette technique donnera un match probable entre toutes les paires de points possibles d'une image à l'autre. Des scènes texturées, mais dont la texture se répète à l'extérieur d'une région de 32×32 pixels poseront le même problème : un mur de brique parfaitement régulier scanné de face présentera beaucoup d'ambiguïtés pour notre algorithme. Ce problème est illustré dans la Figure 1.5.

1.3 Reconstruction passive et reconstruction active

Scanner une scène telle quelle, en utilisant uniquement des caméras et sans lui apporter de modifications, entre dans la catégorie du scan 3D *passif*. Selon le contexte, il est parfois possible de se permettre de contrôler l'illumination de notre scène. On pourrait par exemple ajouter un projecteur à notre configuration pour contourner les problèmes reliés aux textures : une idée simple serait *d'ajouter de la texture artificiellement en illuminant la scène à scanner*. Notre mur blanc ambigu deviendrait ainsi une surface plane sur laquelle on verrait des textures spécialement conçues pour aider notre algorithme à scanner la scène correctement. Le fait d'introduire de l'illumination contrôlée lors du scan entre dans la catégorie du scan 3D *actif*.

On pourrait cependant pousser l'idée précédente un peu plus loin : quitte à introduire un projecteur dans le processus, on peut l'utiliser plus judicieusement. Un projecteur

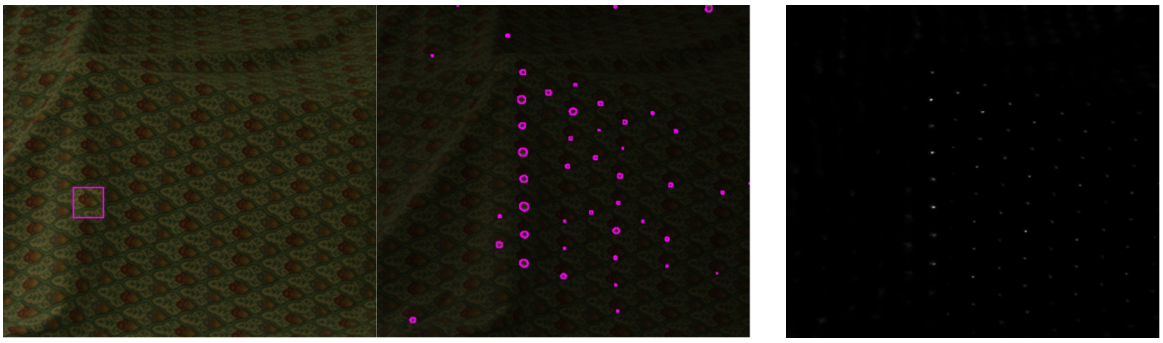

Figure 1.5. Tentative de correspondance en utilisant la corrélation entre des régions pour la scène *Cloth1* **du jeu de données** *Middlebury (2006)* **[38].** *De gauche à droite*, **région choisie dans la vue de droite, régions possiblement correspondantes dans la vue de gauche, et carte de corrélation entre la région choisie de l'image de droite et toutes les positions possibles dans l'image de gauche. Contrairement à la scène montrée à la Figure 1.4, la corrélation ne donne pas un maximum clair ici, puisqu'un motif unique se répète un peu partout dans l'image.**

peut être mathématiquement modélisé de façon similaire à une caméra, en considérant qu'il s'agit d'un *sténopé inversé*. Plutôt que de capturer les photons émis par la scène sur un plan image, le projecteur va projeter son plan image sur la scène. Avec seulement un projecteur et une caméra, on a donc tout ce qu'il faut pour scanner une scène en 3D, sans avoir à se soucier des textures. Différentes techniques de reconstruction 3D sont basées sur cette idée. Ces techniques, qui diffèrent principalement dans la façon de concevoir les images projetées, sont regroupées sous le terme **lumière structurée**.

En pratique, la reconstruction active donne des résultats nettement meilleurs que la reconstruction passive : en 2003, Scharstein et Szeliski [39] vont jusqu'à se servir d'un algorithme de lumière structurée pour obtenir un *"ground truth"* pour un jeu d'images stéréo utilisées pour comparer la performance de différents algorithmes de scan passif. En 2014, un nouveau jeu de tests plus précis a été introduit par Scharstein et al. [37], toujours en utilisant de la lumière structurée pour construire les *ground truths*. Ces jeux de données [38] acquis par lumière structurée sont devenus un standard du domaine

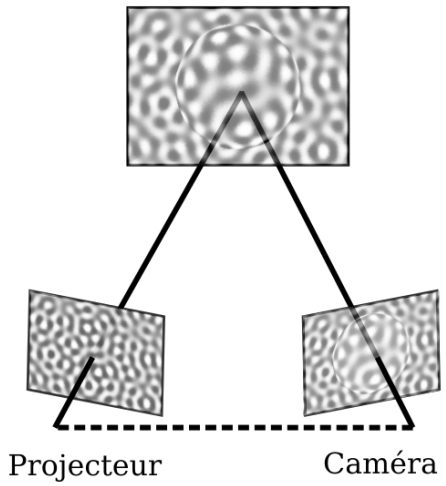

Figure 1.6. Le modèle de formation d'image sur lequel se base la triangulation est le même en reconstruction *active* qu'en reconstruction *passive*

pour l'évaluation des algorithmes de reconstruction passive, ce qui laisse croire que l'utilisation d'un projecteur est une façon simple d'obtenir des scans de bien meilleure qualité qu'en se limitant à des caméras.

1.4 Problème résolu?

Le problème est cependant loin d'être résolu. Comme on le verra dans le chapitre 5, la reconstruction active fait encore plusieurs hypothèses qui peuvent se révéler fausses dans certains contextes, en plus d'apporter son propre lot de désavantages. Pour ne citer qu'un seul exemple : un projecteur bon marché sera rarement assez puissant pour compétitionner avec la lumière du soleil, ce qui rend la reconstruction des scènes en extérieur en plein jour infaisable sans inventer de nouveaux algorithmes mieux adaptés[15] ou sans acheter du matériel spécialisé pour ces situations[32].

L'exemple de la lumière du jour est un détail mineur qui peut être contourné dans beaucoup de situations. Un artiste qui souhaiterait scanner une de ses sculptures peut

probablement le faire depuis son atelier, les rideaux fermés plutôt que de le faire à l'extérieur. On verra cependant dans les chapitres 4 et 5 que même dans une pièce parfaitement contrôlée qui ne contiendrait aucune autre source de lumière que le projecteur, il y a encore beaucoup de petits détails qui peuvent dégrader la qualité des scans obtenus en reconstruction active.

Ce mémoire est une introduction théorique et pratique au scan 3D par lumière structurée. Les prochains chapitres détailleront les différentes étapes du scan en focalisant sur une compréhension intuitive du domaine, sur les problèmes fréquents et sur les différents algorithmes qui forment l'état de l'art actuel.

Figure 1.7. Un scanneur 3D en vrai. Vraiment moins intéressant que la Figure 1.1

Chapitre 2

VUE D'ENSEMBLE : LE SCANNEUR 3D DE A À Z

Avant d'entrer dans les détails de chacune des étapes de la conception d'un scanneur 3D à lumière structurée, ce chapitre présente une vue d'ensemble des différents défis à relever.

2.1 Matériel

La toute première étape de la conception d'un scanneur 3D actif est l'acquisition du matériel approprié : minimalement, une caméra et un projecteur. Beaucoup d'options sont disponibles, dans toutes les gammes de prix.

Pour du scan 3D de hobbyiste, qui ne demanderait pas une précision incroyable, on peut se permettre d'utiliser du matériel bon marché : n'importe quels caméras et projecteurs disponibles en grand magasin vont faire l'affaire. Le matériel pour construire un scanneur 3D raisonnablement bon peut être acquis pour la modique somme de $\approx 200\$$, par exemple en utilisant un mini-projecteur portatif [1] et une webcam haute définition dans la moyenne de ce qui se vend [2].

Il faut cependant savoir que ce type d'équipement est conçu avant tout pour donner des résultats *visuellement jolis* pour le commun des mortels. Puisque les images produites ne sont pas a priori destinées à être analysées par ordinateur, rien n'empêche les fabricants de caméras d'appliquer certains filtres pour "embellir" les photos capturées. Ces différents filtres peuvent introduire des artéfacts visibles dans les reconstructions. Un "embellissement" fréquent dans les caméras bon marché est le rehaussement des contours, illustré à la Figure 2.1. Ce genre d'effet est souvent présent par défaut et est possiblement désactivable dans les paramètres.

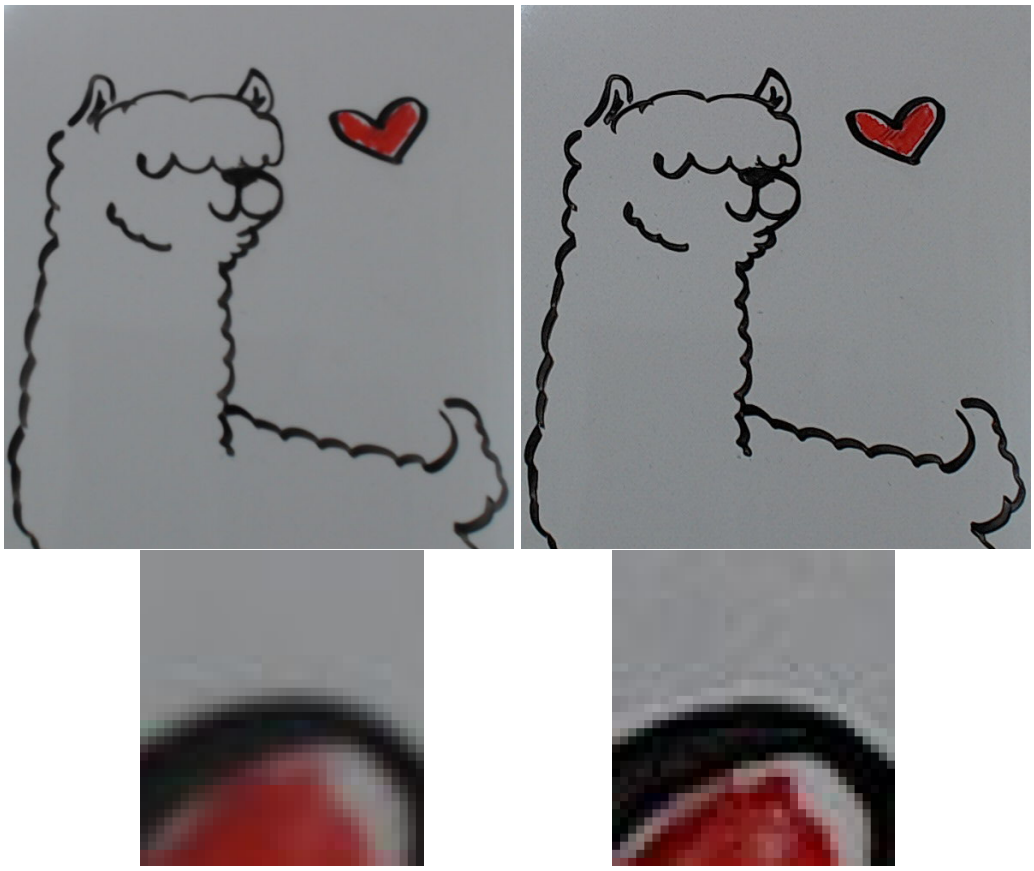

Figure 2.1. Effet d'un rehausseur de contours appliqué automatiquement par beaucoup de caméras sur les images capturées, pleine image (*rangée du haut*) et zoom sur le coeur (*rangée du bas*). *À gauche*, l'image non filtrée est représentative des photons capturés par le senseur de la caméra, *à droite*, l'image a été modifiée après la capture pour paraître beaucoup plus jolie aux yeux d'un humain. L'image filtrée ajoute un contour blanc et un intérieur noir au vrai contour du coeur, ce qui peut potentiellement dégrader une analyse automatique de l'image par un algorithme.

Pour avoir une certaine garantie sur la précision de l'équipement, ce qui peut être nécessaire dans certains contextes tels que celui de l'assurance-qualité industrielle, le mieux est d'acquérir de l'équipement professionnel spécialisé. Les caméras industrielles sont, par opposition aux caméras grand public, conçues pour donner un contrôle total sur la capture des images, souvent avec une meilleure résolution ou avec une vitesse de capture beaucoup plus grande. Les images capturées par ces caméras sont donc parfaites pour l'analyse par des machines. Ces dernières viennent cependant généralement avec un coût plus élevé.

À l'inverse, si la perte de précision n'est pas considérée comme un problème trop grave, on peut se permettre d'aller plus loin en recyclant de l'équipement déjà disponible. Beaucoup de gens possèdent déjà au moins une caméra, puisqu'elles sont souvent directement intégrées à même les ordinateurs et téléphones portables, et théoriquement, un petit écran de cellulaire, une boîte à soulier, une loupe et un élastique sont suffisants pour concevoir un projecteur maison [3] pour moins de 5$.

2.2 Calibrage du système

Une fois l'acquisition du matériel faite, la seconde étape est de le *calibrer*. Le calibrage complet du système inclut différentes étapes :

1. **Calibrage des paramètres intrinsèques :** ce premier calibrage permet de déterminer comment la caméra projette le monde sur son plan image. Dans le cas du projecteur, on calcule plutôt comment le plan image est déprojeté sur le monde, ce qui revient au même d'un point de vue mathématique. Un exemple de paramètre intrinsèque serait *l'angle de vue*.

2. **Calibrage des paramètres extrinsèques :** ce second calibrage permet de déterminer la *pose relative du système dans le monde 3D*. Pour trianguler des points, on doit connaître la position de la caméra et du projecteur dans le monde (voir Figure 1.6).

Lorsqu'on déplace un des éléments du système, on doit recommencer ce calibrage.

3. **Calibrage de différents filtres correctifs :** ce dernier calibrage englobe toutes les méthodes qui ont pour but de corriger au niveau logiciel certaines imperfections du système. On va entre autres chercher à corriger la distorsion radiale, la distorsion gamma des intensités projetées/capturées images, la distorsion des couleurs... Dans le contexte où on utilise du matériel de basse qualité, cette étape peut devenir particulièrement importante, tout dépendant des hypothèses faites par l'algorithme de scan utilisé.

À ces trois calibrages s'ajoutent possiblement d'autres calibrages qui servent à intégrer de l'information a priori supplémentaire dans le système. Ces différents calibrages dépendent de la méthode exacte utilisée, et non du matériel. L'exemple le plus important ici est le calibrage épipolaire, qui permet dans beaucoup de cas de réduire le nombre total d'images à capturer ou qui est nécessaire au fonctionnement de certains algorithmes. D'autres calibrages plus spécifiques ont été imaginés pour répondre à certains besoins inhabituels. On peut par exemple penser à [46], qui demande de connaître la fonction de transfert de modulation du système caméra-projecteur, puisqu'elle extrait de l'information supplémentaire sur la scène en analysant les harmoniques présentes dans la séquence temporelle d'images.

2.3 Acquisition des images

Bien que l'étape de capture des images puisse sembler simple (on projette une image et on capture le résultat, facile non?), il est crucial de comprendre quels problèmes peuvent survenir à cette étape. Un problème technique récurrent lors de la capture touche la synchronisation du projecteur avec la caméra. Si la caméra et le projecteur sont contrôlés indépendamment, on peut facilement se retrouver dans une situation où une ou plusieurs images se retrouvent capturées pendant la transition entre deux projections, ce qui rend le

résultat inutilisable (Figure 2.2).

En pratique, avoir un même logiciel qui contrôle à la fois la capture et la projection est une façon simple d'éviter des problèmes. En alternant la capture et la projection et en ajoutant un délai supplémentaire pour éviter des défauts dans les images capturées, on peut facilement s'attendre à recevoir autour de 5 images par seconde sur de l'équipement bon marché.

Un autre facteur à considérer au moment de la capture est la *configuration* de l'équipement. Pour éviter les problèmes d'"embellissement" automatique mentionnés plus haut, on devrait désactiver tous les filtres, auto-focus, balance des blancs, ... et tout basculer en mode manuel autant que possible. Beaucoup de caméras font une auto-exposition par défaut, ce qui peut également corrompre les données capturées. À noter ici que plusieurs caméras bas de gamme *réinitialisent* ces paramètres à chaque redémarrage, d'où la mention dans cette section : c'est une bonne idée d'inclure cette configuration dans le processus de capture pour éviter toute surprise.

2.4 Mise en correspondance

Une fois les images capturées, l'étape suivante est de mettre en correspondance les points capturés avec les points projetés. C'est à cette étape que les différentes façons de concevoir un scanneur divergent.

L'objectif de la mise en correspondance est d'associer à chaque pixel de caméra une position dans le plan image du projecteur. Typiquement, on représente cette information par une *carte de correspondance* sous forme de bitmap 16-bit (Figure 2.3), où les canaux rouge et vert encodent respectivement les positions en x et en y dans le plan image du projecteur. Le canal bleu n'est pas forcément utilisé, mais peut contenir de l'information additionnelle au besoin. Certaines méthodes peuvent par exemple calculer une *mesure de qualité* ou de *confiance* pour chaque correspondance, qui peut être stockée dans ce canal. Il est à noter qu'avec un scanneur 3D bon marché, 16-bits (65 535 valeurs distinctes) est

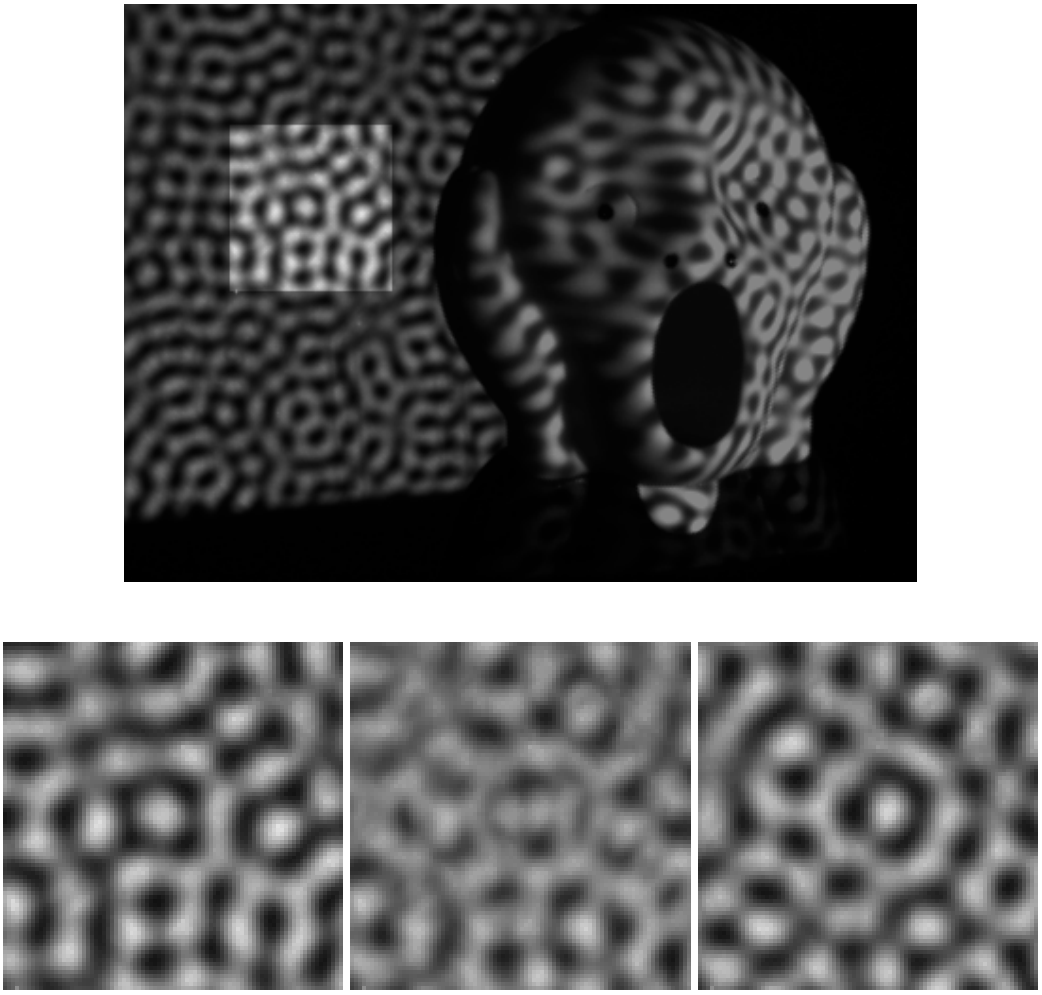

Figure 2.2. Projection des images sur une scène (*haut*) et zoom sur le carré en subrillance (*bas*), respectivement pour la première image projetée (*gauche*), pour un mélange de deux images en même temps (*milieu*) et pour la deuxième image projetée (*droite*). La capture de deux images mélangées se produit lorsque le temps d'exposition de la caméra contient le moment de transition d'une image à l'autre, lorsque la caméra et le projecteur sont désynchronisés. Une capture mélangée est typiquement inutilisable lors du décodage.

plus que suffisant pour la précision que l'on peut espérer obtenir pour chaque dimension.

Le sujet de la mise en correspondance sera abordé extensivement dans le chapitre 5. L'article *Fast Discontinuity-Aware Subpixel Correspondence in Structured Light* publié dans le cadre de cette maîtrise (chapitre 6) répond à l'un des problèmes qui se posent à cette étape.

2.5 Triangulation

L'objectif de la mise en correspondance est d'obtenir des paires de points à utiliser lors de la triangulation, ce qui permet d'extrapoler une position 3D à partir de mesures 2D. Bien qu'une version simple de la triangulation soit présentée dans l'introduction (Figure 1.3), cette méthode est un peu trop naïve pour fonctionner dans le vrai monde.

En supposant que le point 3D recherché correspond *précisément* aux projections 2D capturées dans les plans images, on peut être tenté d'utiliser de la trigonométrie de base pour calculer l'intersection des lignes de déprojections dans l'espace. Cette hypothèse est souvent trop forte en pratique [18]. Une toute petite erreur de mesure suffit pour faire en sorte que les lignes en 3D passent l'une à côté de l'autre sans se croiser, ce qui rend la solution naïve inutilisable. En pratique, on utilise plutôt des algorithmes numériquement stables comme la *direct linear transformation*, présentée rapidement dans le chapitre 7.

2.6 Maillage

Avec la triangulation, on se retrouve avec un *nuage de points 3D*, ce qui n'est malheureusement pas utilisable dans la plupart des applications concrètes intéressantes. Un tas de points dans l'espace ne contient d'information ni sur la *surface sous-jacente* ni sur la façon dont les points sont connectés entre eux. Pour avoir un modèle 3D qui peut être importé dans un logiciel d'animation ou qui peut être envoyé à l'imprimante 3D, la dernière étape requise est le *maillage*.

Plusieurs stratégies permettent de créer un modèle 3D complet à partir de nuages de

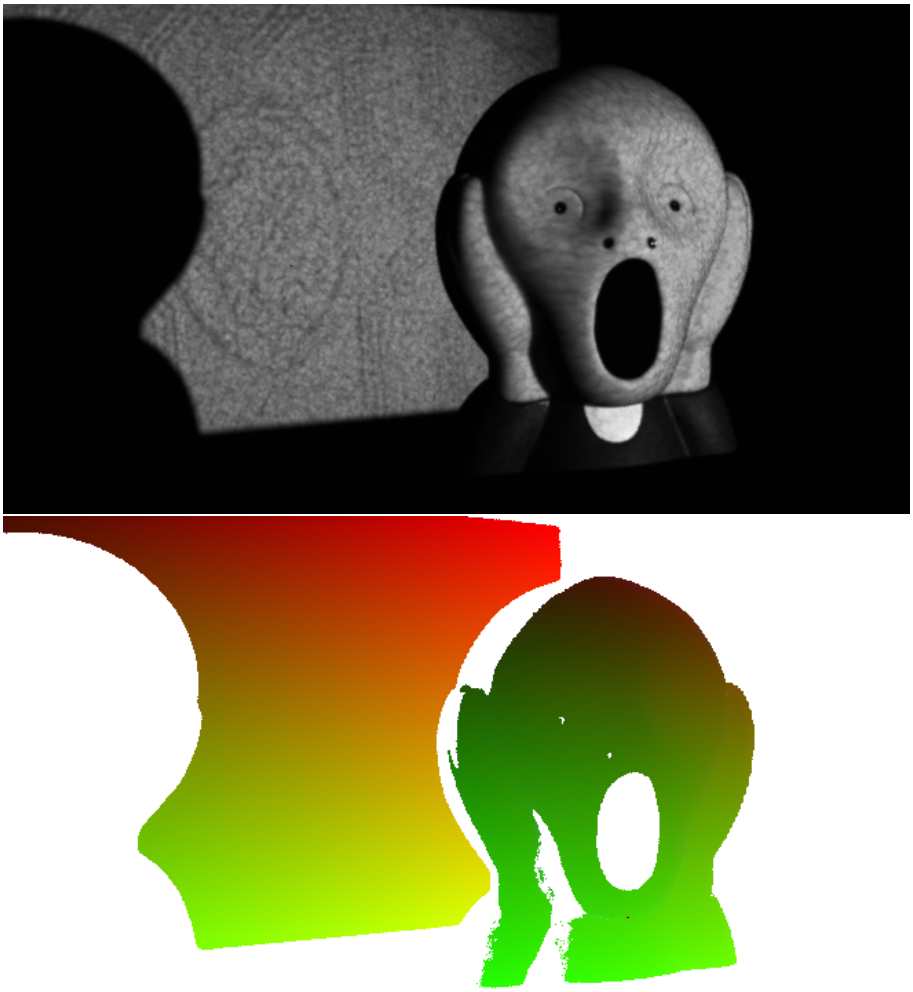

Figure 2.3. Carte de correspondance calculée pendant le scan 3D d'un jouet inspiré par *Le Cri*, d'Edvard Munch

points. On peut par exemple utiliser la connectivité des pixels de caméras ainsi qu'une *carte de discontinuités* pour générer directement un modèle 3D, ou encore considérer le nuage de points comme définissant une surface implicite, que l'on peut résoudre de façon globale [23].

2.7 Conseils pratiques

Le scan 3D complet comporte plusieurs étapes et chacune d'entre elles peut être une source de problèmes. Pour éviter de se retrouver avec un tas de points 3D complètement n'importe comment, il est essentiel de *valider le scan à mesure* en visualisant les résultats – même partiels – dès que possible. Une inspection visuelle rapide du résultat est de mise à chaque étape. Dès le calibrage, on devrait avoir une façon de dessiner en 3D le système caméra/projecteur pour valider que le calibrage reflète de façon réaliste la géométrie du système. Une simple validation du fait que, par exemple, le projecteur est bel et bien à droite de la caméra comme dans le vrai monde, peut sauver beaucoup de temps de débogage versus calculer tout le nuage de points avant de commencer à chercher l'étape qui a bien pu causer une erreur.

De façon générale, il vaut mieux avoir une façon rapide d'inspecter les données, en les écrivant sur le disque plutôt qu'en les gardant dans la mémoire du programme seulement. Par exemple, après la capture, on veut pouvoir inspecter visuellement les images pour détecter des anomalies (sous-exposition/surexposition, mix temporel des images...) si elles ont lieu.

Automatiser les tâches séparément est également une très bonne façon de pouvoir recommencer le processus non pas de zéro, mais depuis la dernière étape réussie. Plutôt que d'avoir un programme unique qui se charge de réaliser les étapes de `capture` + `correspondance` + `triangulation` + `maillage`, utiliser quatre programmes distincts appelés un après l'autre depuis un `script shell` laisse la flexibilité nécessaire pour ne pas avoir à recommencer de zéro lorsqu'il y a un problème à l'une des étapes.

Chapitre 3

CALIBRAGE D'UN SYSTÈME À LUMIÈRE STRUCTURÉE

Puisque le scan 3D repose sur la triangulation de points entre deux plans image, c'est absolument crucial de savoir *comment* les points du monde réel y sont projetés.

3.1 Calibrer la caméra

Le modèle *sténopé* permet de modéliser la façon dont les points 3D du monde sont projetés sur le plan image de la caméra. Mathématiquement, la *matrice de caméra P* représente la transformation complète d'un point 3D du monde $\mathbf{x}_w$ vers un point 2D du plan image $\mathbf{x}$:

$$\mathbf{x} = P\mathbf{x}_w \tag{3.1}$$

$$P = K\left[I_3 \;\middle|\; 0\right]RT \tag{3.2}$$

$$= \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

avec

$K =$ matrice 3×3 des paramètres intrinsèques

$\left[I_3 \;\middle|\; 0\right] =$ matrice 3×4 de projection

(coordonnées 3D homogènes $\rightarrow$ coordonnées 2D homogènes)

$RT =$ matrice 4×4 (coordonnées homogènes) des paramètres extrinsèques

(**R**otation et **T**ranslation de la caméra dans le monde)

Considérons chaque matrice une à la fois, de la droite vers la gauche.

3.1.1 Paramètres extrinsèques

La matrice des paramètres extrinsèques combine la rotation et la translation de la caméra dans le monde. Pour un point du monde $\mathbf{x}_w$ exprimé en coordonnées homogènes, ces deux matrices permettent de calculer :

$$\mathbf{x}_{cam} = RT\mathbf{x}_w \tag{3.4}$$

$$(x_{cam}\ y_{cam}\ z_{cam}\ 1)^T = RT\,(x_w\ y_w\ z_w\ 1)^T \tag{3.5}$$

soit le même point 3D en coordonnées homogènes, mais exprimé dans le système de coordonnées de la caméra.

L'utilisation des coordonnées homogènes ici – $(x,y,z,1)^T$, au lieu de $(x,y,z)^T$ en coordonnées euclidiennes – permet de combiner la translation et la rotation en une seule matrice.

On pourrait arriver au même résultat en utilisant plutôt des coordonnées euclidiennes avec

$$\mathbf{x}_{cam} = R\mathbf{x}_w + T \tag{3.6}$$

$$(x_{cam}\ y_{cam}\ z_{cam})^T = R\,(x_w\ y_w\ z_w)^T + (t_x\ t_y\ t_z)^T\,. \tag{3.7}$$

L'utilisation des coordonnées homogènes a cependant d'autres avantages, en particulier la possibilité de représenter des choses qui n'ont pas de représentation pas en géométrie euclidienne, par exemple les lignes et les points situés à l'infini [1].

3.1.2 Projection

La projection 3D vers 2D est ensuite exprimée avec

$$\mathbf{x}_{cam2D} = \left[\, I_3 \;\middle|\; 0 \,\right] \mathbf{x}_{cam}. \tag{3.8}$$

[1] Un *point de fuite* dans une image, soit un point où deux droites parallèles convergent, est un exemple de point à l'infini

où $\mathbf{x}_{cam2D}$ représente le même point, mais écrasé le long de l'axe z en 2D dans le système de coordonnées de la caméra.

3.1.3 Paramètres intrinsèques

Finalement, la matrice K des paramètres intrinsèques ajuste les coordonnées homogènes de $\mathbf{x}_{cam2D}$ pour en faire des coordonnées dans le plan image de la caméra, en considérant la *longueur focale* f et la coordonnée dans le plan image du point principal (c_x, c_y) :

$$\mathbf{x} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{cam2D}. \tag{3.9}$$

Le point principal correspond au point où l'axe optique de la caméra croise le plan image. La longueur focale est la distance entre le centre optique de la caméra et le point principal. Le point (c_x, c_y) est souvent localisé près du milieu de l'image.

Il s'agit ici du modèle simple de K. Ce modèle suppose des pixels carrés, ce qui peut ne pas être le cas, par exemple pour des caméras utilisant des capteurs CCD. Dans le cas où la caméra a des pixels rectangulaires plutôt que carrés, dans une proportion m_x par m_y, on devrait également inclure cette proportion :

$$\begin{bmatrix} a_x & 0 & c_x \\ 0 & a_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

où

$$a_x = f m_x \tag{3.11}$$

$$a_y = f m_y. \tag{3.12}$$

3.2 Distorsion radiale

Le modèle de caméra sténopé est simple, mais il n'est pas tout à fait réaliste : la majorité des caméras utilisent des lentilles pour condenser la lumière et ainsi réduire le temps d'exposition nécessaire à la capture d'une image. Ces lentilles peuvent malheureusement introduire des déformations non linéaires dans la géométrie de l'image, plus précisément sous forme de distorsion radiale. En présence de distorsion radiale, les droites du monde ne restent pas tout à fait droites une fois projetées dans le plan image (Figure 3.1). À moins d'utiliser des caméras à grand angle, cet effet est généralement suffisamment léger pour permettre d'utiliser le modèle de caméra sténopé auquel on ajoute une étape supplémentaire de "dé-distorsion" des points de l'image.

3.3 Procédure pour le calibrage

La matrice P est une matrice 3×4. Elle contient 12 entrées, mais est définie à un facteur d'échelle près, puisqu'elle travaille avec des points en coordonnées homogènes. On a donc 11 degrés de liberté dans une matrice de caméra. En prenant des points 3D connus dans le monde, on peut capturer et identifier leurs images dans la caméra et ainsi poser des contraintes sur P. Chaque correspondance $\mathbf{p}_w \leftrightarrow \mathbf{x}$ donne deux contraintes, donc 6 correspondances 3D $\leftrightarrow$ 2D sont nécessaires au minimum. Une seule image capturée peut suffire, pour autant que six points connus soient présents et que les six points ne soient pas situés sur un même plan. Les coins d'un cube pourraient suffire, à condition de choisir l'angle de capture de façon très spécifique, sinon un dodécahèdre de dimensions connues est un choix idéal.

Cette méthode est moyennement pratique dans un contexte de hobbyiste. Peu de personnes possèdent déjà d'avance un objet 3D aux dimensions précisément connues pour répondre à ces contraintes. En revanche, beaucoup de gens ont accès à une imprimante, ce qui ouvre la porte au calibrage *planaire* : plutôt que d'utiliser la projection de points 3D connus, on peut utiliser un objet planaire connu, par exemple un damier dont on connaît

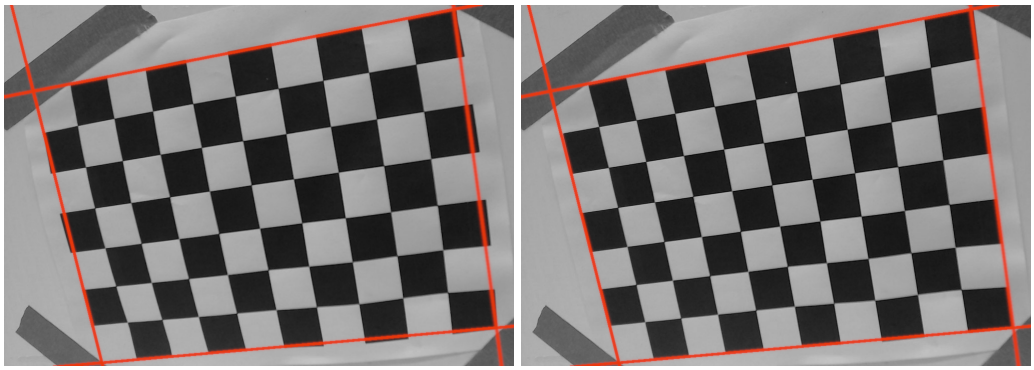

Figure 3.1. Impact de la distorsion radiale sur les droites dans une image : le cadre rouge est le même d'une image à l'autre, le damier apparaît légèrement courbé en présence de distorsion radiale (*à gauche*). On peut corriger cette déformation avec une rectification logicielle de l'image (*à droite*).

les dimensions des carrés. Cette méthode a été proposée par Zhang [47] et est depuis la méthode de calibrage la plus connue et la plus vastement utilisée. Cette méthode est implantée dans OpenCV, avec la fonction `cv::calibrateCamera`.

L'idée est la suivante : on peut fixer le système de coordonnées du monde comme étant aligné avec l'objet planaire. Un des coins du plan sert d'origine en trois dimensions en coordonnées homogènes, $(0,0,0,1)$. Sans perte de généralité, les axes x et y sont définis comme étant perpendiculaires dans le plan de l'objet, et l'axe est perpendiculaire à ce plan plan, qui fixe la position de référence en $z = 0$. Fixer l'axe $z = 0$ permet de travailler en 2D, et donc de considérer la transformation de quatre points dans l'objet planaire comme étant une homographie. On peut réexprimer la transformation $\mathbf{x} = P\mathbf{x}_w$ avec P qui est une matrice 3×3, calculer plusieurs homographies qui donnent chacune de l'information sur P, et ainsi résoudre la matrice complète.

Bien que cette méthode reste la référence, d'autres stratégies ont été proposées au fil du temps, par exemple [44], qui propose un calibrage des paramètres internes et externes automatique directement à partir d'une mise en correspondance d'images d'un objet 3D à scanner.

3.4 Calibrer le projecteur

En lumière structurée, le projecteur remplace une des caméras. Le modèle de projection reste cependant le même :

$$P = K \left[I_3 \;\middle|\; 0 \right] RT \tag{3.13}$$

La façon d'obtenir cette matrice pour la caméra était de *capturer* plusieurs points de vue d'un objet connu. On a besoin de faire la même chose, mais pour le point de vue du projecteur.

L'étape de la mise en correspondance des points entre l'image du projecteur et l'image de la caméra (chapitre 5) peut nous aider ici. Sachant que chaque pixel de caméra a capturé un bout de la scène et que chaque pixel de caméra est également associé à un pixel de projecteur, on peut transposer les intensités capturées dans les coordonnées correspondantes du projecteur. Le résultat de ce processus montré à la Figure 3.2.

Les méthodes indirectes peuvent calculer la correspondance projecteur $\rightarrow$ caméra directement (ex.: [7]), mais la majorité des méthodes de lumière structurées sont des méthodes *directes*, qui se limitent à générer la correspondance caméra $\rightarrow$ projecteur.

Dans ce cas, on peut toujours tenter d'inverser la carte de correspondance. Chaque pixel de caméra est associé à un pixel de projecteur, mais l'inverse est un peu plus compliqué : certains pixels de projecteur pourraient ne pas avoir de correspondance dans la caméra, tandis que d'autres pixels pourraient très bien être associés à plusieurs pixels de caméra. Dans une situation où la caméra a une résolution plus basse que le projecteur, le résultat est une carte de correspondance éparse telle qu'illustré à la Figure 3.3.

3.5 Calibrage gamma

Comme c'est le cas pour notre audition, notre système visuel humain ne perçoit pas les intensités de façon linéaire, mais plutôt de façon (à peu près) logarithmique : les quantités relatives sont beaucoup plus importantes pour nous que les quantités absolues.

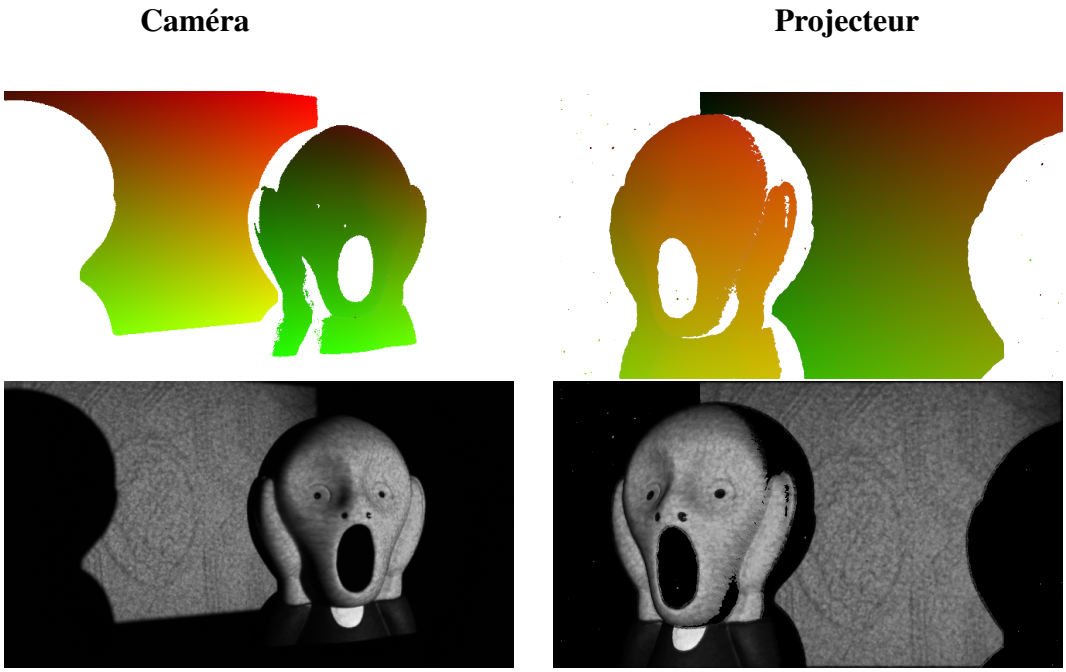

Figure 3.2. *Rangée du haut :* **carte de correspondance caméra $\rightarrow$ projecteur (*gauche*) et carte de correspondance projecteur $\rightarrow$ caméra (*droite*). *Rangée du bas :* la deuxième seconde peut être utilisée comme carte de transformation d'une image capturée par la caméra (*gauche*) pour générer la vue projecteur (*droite*)**

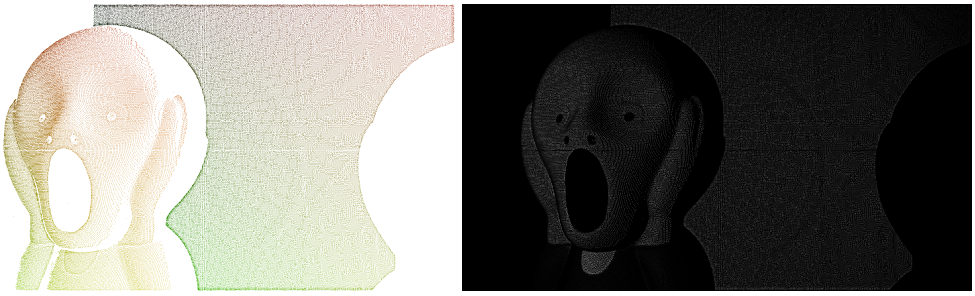

Figure 3.3. Inverser une carte de correspondance caméra $\rightarrow$ projecteur ne fonctionne pas dans toutes les situations, puisque la correspondance n'est pas forcément $1 \leftrightarrow 1$, ce qui peut laisser des trous dans la carte inversée

Dans cette optique, avoir des caméras qui stockeraient des intensités lumineuses de façon linéaire serait un peu absurde : sur 8 bits, la plage de valeurs allant de 0 à 10 contiendrait énormément de nuances perceptibles à l'oeil, tandis que les intensités de 245 à 255 seraient à peu près toutes perçues de la même façon.

Pour compenser cet effet, on peut appliquer une transformation gamma sur les intensités absolues de la forme $I' = I^{\gamma}$, où I et I' sont des intensités normalisées entre 0 et 1. En utilisant un $\gamma < 1$, les valeurs basses se retrouvent à occuper plus de place dans l'encodage que les valeurs élevées. Un tel encodage est illustré dans la Figure 3.4. On peut décoder cette intensité au besoin pour retrouver la valeur originale en utilisant la formule inverse, avec plutôt $\frac{1}{\gamma}$. Ce décodage est fait par le projecteur, qui devrait *à peu près* utiliser un γ qui est l'inverse de celui de la caméra. On a donc $0 < \gamma_c < 1$ pour la caméra et $\gamma_p > 1$ pour le projecteur.

Si on considère que chaque capture I_c d'une intensité projetée I_p peut être modélisée avec :

$$I_c = (\alpha {I_p}^{\gamma_p} + o)^{\gamma_c} \tag{3.14}$$

où α correspond à l'albédo de l'objet qui reçoit l'intensité et o est l'illumination globale en ce point, on peut retrouver le meilleur modèle aux moindres carrés pour un minimum de quatre observations $I_{c,i} \leftrightarrow I_{p,i}$:

$$\underset{\alpha,o,\gamma_p,\gamma_c}{argmin} ||I_c - (\alpha {I_p}^{\gamma_p} + o)^{\gamma_c}||^2. \tag{3.15}$$

Un choix simple peut être de capturer l'ensemble des tons de gris utilisables entre 0 et 255, et de trouver une solution pour chaque pixel. On peut cependant noter que, si α, o, I_c et I_p varient pour chaque pixel, les paramètres γ_p et γ_c devraient rester les mêmes sur toute l'image. On devrait donc pouvoir estimer ces paramètres globalement sur tous les pixels en même temps.

Alternativement, on peut considérer utiliser une caméra dont la réponse est linéaire, ou dont le gamma est réglable. Dans le cas, le modèle d'image à calibrer devient plus

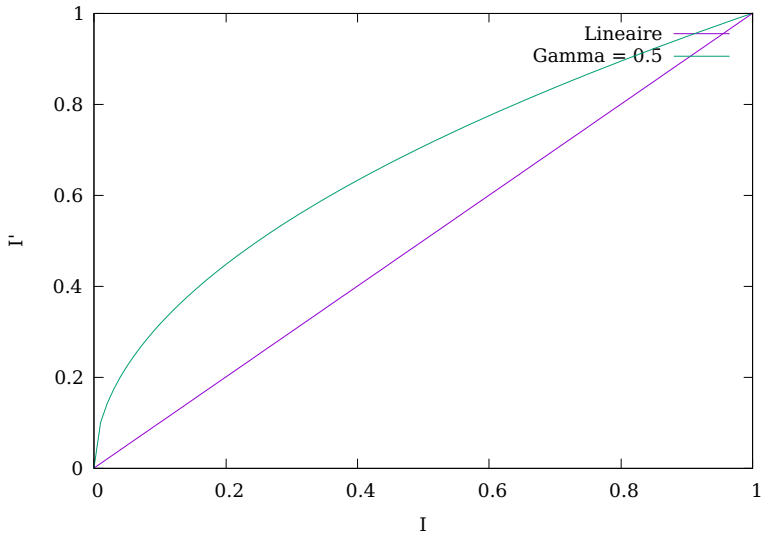

Figure 3.4. Exemple de non-linéarité gamma appliquée aux intensités capturées

simple :

$$I_c = \alpha {I_p}^{\gamma_p} + o. \tag{3.16}$$

Dans cette version, en utilisant des intensités normalisées entre 0 et 1, les points extrêmes ne sont pas affectés par le gamma, ce qui permet de retrouver l'albédo et l'illumination ambiante :

$$I_{c\,min} = (\alpha 0^{\gamma_p} + o) \tag{3.17}$$

$$\Longrightarrow I_{c\,min} = (\alpha 0 + o) \tag{3.18}$$

$$\Longrightarrow o = I_{c\,min} \tag{3.19}$$

$$I_{c\,max} = \alpha 1^{\gamma_p} + o \tag{3.20}$$

$$\Longrightarrow I_{c\,max} = \alpha 1 + o \tag{3.21}$$

$$\Longrightarrow I_{c\,max} = \alpha + o \tag{3.22}$$

$$\Longrightarrow \alpha = I_{c\,max} - I_{c\,min} \tag{3.23}$$

En ayant capturé quelques images dans l'ensemble des tons de gris utilisables *qui in-*

cluent les intensités minimum et maximum, on peut alors récupérer pour des intensités quelconques :

$$I_c = \alpha {I_p}^{\gamma_p} + o \tag{3.24}$$

$$\Longrightarrow \frac{I_c - o}{\alpha} = {I_p}^{\gamma_p} \tag{3.25}$$

$$\Longrightarrow \log(\frac{I_c - o}{\alpha}) = \gamma_p \log I_p \tag{3.26}$$

$$\Longrightarrow \gamma_p = \frac{\log(\frac{I_c - o}{\alpha})}{\log I_p} \tag{3.27}$$

On peut moyenner les valeurs de γ_p récupérées sur l'ensemble des pixels de l'image pour estimer le paramètre. Cette procédure peut être incorporée à même certains algorithmes de scan. La méthode présentée au chapitre 6 s'y prête bien.

Une fois le modèle récupéré, on peut soit modifier les images projetées pour compenser la distorsion non linéaire (compensation active), ou soit déduire les valeurs des images qui auraient dû être capturées si le système était linéaire (compensation passive). En pratique, la compensation active donne des résultats significativement meilleurs que la compensation passive [45].

On doit cependant noter que les projecteurs et caméras ne sont pas tenus de respecter un modèle de gamma parfait, et l'équation 3.14 n'est pas forcément représentative de la réalité. Dans [45], Zhang note que la grande majorité des projecteurs ne suivent pas un modèle non linéaire aussi simple. Zhang propose plutôt de modéliser les transformations non linéaires par des polynômes de degré 7.

3.5.1 Contourner le problème

Une bonne correction des non-linéarités peut devenir relativement complexe à faire. Une façon simple de contourner ce problème est d'abandonner l'idée d'analyser les tons de gris directement. En choisissant plutôt d'utiliser des mesures binaires, on gagne en précision dans les mesures. Par exemple, en considérant seulement la projection de blanc ou de noir, la question de la relation exacte entre une intensité projetée et une intensité

capturée n'a plus tellement d'importance. Une autre approche est de considérer seulement la relation d'ordre entre les intensités, qui est généralement respectée, peu importe la distorsion non linéaire : si $I_{p1} < I_{p2}$, on peut s'attendre à capturer $I_{c1} < I_{c2}$, peu importe la distorsion non linéaire qui est appliquée.

Choisir une telle stratégie permet d'éviter le calibrage des non-linéarités du système et est à considérer dans un contexte de scan 3D avec de l'équipement à bas coût. Le défaut de cette approche est cependant la perte d'information. Utiliser une méthode qui n'utilise pas le plein intervalle de tons de gris disponible va typiquement devoir compenser en projetant plus d'images qu'une méthode qui les utilise au maximum.

3.6 Calibrage épipolaire

Il existe une relation géométrique qui relie les points correspondants d'une image à l'autre. N'importe quel plan dans l'espace qui contient la ligne reliant le centre de la caméra et le centre du projecteur est un *plan épipolaire*. Pour un plan épipolaire donné, son intersection avec le plan image de la caméra donne une ligne l_c et son intersection avec le plan image du projecteur donne une ligne l_p. Les lignes l_c et l_p sont des *lignes épipolaires correspondantes* et donnent une contrainte sur la correspondance des points de l'image.

Si un point est projeté dans une ligne épipolaire l_c dans la caméra, il est **nécessairement** projeté quelque part sur la ligne l_p du projecteur. Cette contrainte peut s'avérer particulièrement utile pour contraindre l'espace de recherche des correspondances lors d'un scan. Elle permet également de limiter la recherche des correspondances en une seule dimension, pour ensuite en déduire la correspondance 2D.

Il est important de noter que cette contrainte est indépendante de la scène et ne dépend que de la géométrie des caméras. Algébriquement, la géométrie épipolaire peut s'exprimer par la matrice 3×3 F, appelée la *matrice fondamentale*. Les points cor-

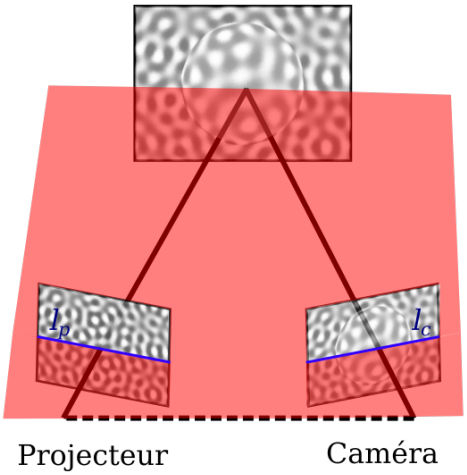

Figure 3.5. Plan épipolaire (*rouge*) qui relie les centres de la caméra et du projecteur à un point de la scène et lignes épipolaires (*bleu*) l_c et l_p correspondantes pour ce plan.

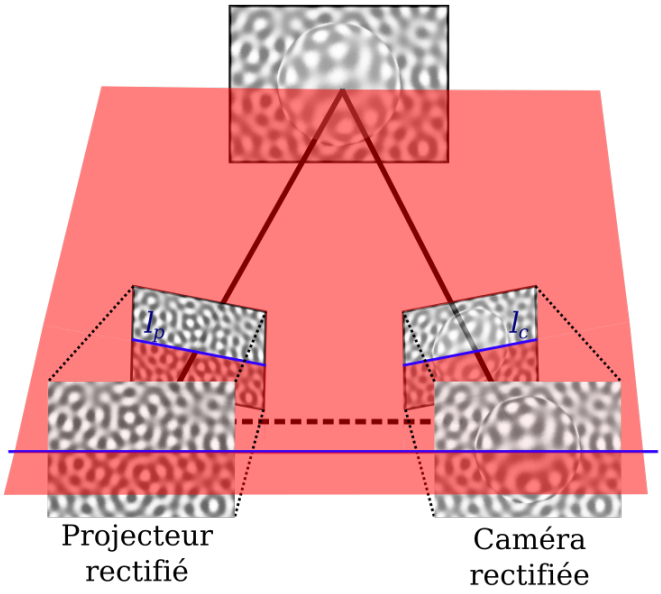

Figure 3.6. Rectification stéréo : les plans images sont reprojetés sur un plan commun dans lequel les lignes épipolaires correspondantes sont alignées

respondants doivent satisfaire l'équation :

$$\mathbf{x}'F\mathbf{x} = \mathbf{0} \tag{3.28}$$

où $\mathbf{x}'$ et $\mathbf{x}$ sont des points correspondants dans la caméra et le projecteur, exprimés en coordonnées homogènes dans les plans images. On peut calculer F à partir d'au minimum sept correspondances $\mathbf{x}' \leftrightarrow \mathbf{x}$. Avec OpenCV, la matrice fondamentale peut être récupérée via la fonction `cv::findFundamentalMat`.

Concrètement, la matrice fondamentale peut être utilisée pour effectuer une rectification stéréo avant d'avoir obtenu un calibrage complet de la caméra et du projecteur. Une rectification stéréo correspond à reprojeter les plans images de la caméra et du projecteur sur un plan commun dans lequel les lignes épipolaires correspondantes sont alignées. Cette rectification est illustrée dans la Figure 3.6.

La fonction `cv::stereoRectifyUncalibrated` d'OpenCV permet d'effectuer cette rectification à partir de la matrice fondamentale, sans avoir besoin de la matrice de calibrage complet P. Pour une paire de caméra et projecteur déjà calibrés, la fonction `cv::stereoRectify` peut également effectuer cette rectification, en utilisant plutôt les paramètres internes et externes comme guides.

Chapitre 4

PROJECTION ET CAPTURE DES IMAGES

Bien que la capture puisse sembler être la partie la plus simple du scan, cette section est particulièrement importante. En plus d'être l'étape qui cause le plus de problèmes techniques, c'est également ici que beaucoup de petits détails peuvent s'accumuler et finir par dégrader les scans de façon significative.

En contrôlant bien son équipement et en anticipant les problèmes qui peuvent survenir, on peut améliorer de beaucoup la qualité des scans et éviter de perdre du temps.

4.1 Problèmes reliés à la capture

Le problème de la distorsion non linéaire des tons de gris a été couvert au chapitre précédent. Il s'agit là d'un bon exemple du fait que les caméras et projecteurs grand public sont conçus avant tout pour nos yeux, pas pour les ordinateurs. Lorsqu'on les utilise pour le traitement automatique des images, on doit s'attendre à faire face à quelques problèmes.

4.1.1 Auto-focus et auto-exposition

La personne moyenne n'aime pas régler manuellement le point focal de sa caméra entre chaque photo : la majorité des caméras vont, au moment de capturer une photo, faire la mise au point pour que les éléments sailliants apparaissent de façon nette dans l'image.

Lorsqu'on s'amuse à projeter des textures artificielles sur une scène, les algorithmes simples d'auto-focus ont souvent de la difficulté à suivre et tentent de se réajuster un peu n'importe comment. De la même façon, le temps d'exposition nécessaire pour faire apparaître une image ni trop claire ni trop sombre est souvent ajusté automatiquement pour chaque photo. Encore une fois, les textures artificielles ont tendance à mêler la

détection automatique de l'exposition, surtout lorsqu'un mélange d'images à hautes et à basses fréquences est utilisé lors d'un même scan.

Dans la mesure où la caméra le permet, les fonctionnalités d'*auto-focus* et d'*auto-exposition* doivent être désactivées avant de commencer un scan. Le focus doit être réglé manuellement, de façon à ce que la plus grande partie des points de la scène soient au focus. Le défocus des images agit comme un filtre passe-bas sur les intensités, ce qui peut dégrader les résultats du scan. L'exposition devrait également être réglée manuellement, de façon à capturer des images qui ne sont ni surexposées ni sous-exposées. Une surexposition crée de la distorsion dans les intensités : les pixels trop exposés ont tendance à "déborder" dans les pixels voisins et corrompre une bonne partie de l'image. À l'inverse, une exposition inutilement basse entraîne une perte de précision dans les intensités capturées, puisque les nuances entre les intensités les plus sombres sont perdues.

4.1.2 Gain et rehaussement de contours

Tous les filtres appliqués après la capture devraient être désactivés, question d'avoir une image aussi proche que possible de ce que les senseurs ont perçu.

Parmi ces filtres, on retrouve entre autres le rehausseur de contours donné en exemple au chapitre 2, dans la Figure 2.1. Le *gain*, qui correspond simplement à un multiplicateur numérique du signal reçu, est un autre filtre commun. L'image résultante apparaît plus *lumineuse* sans modification du temps d'exposition, mais le bruit numérique de la caméra se retrouve également amplifié.

4.1.3 Bruit numérique

Les circuits électroniques qui composent les senseurs des caméras ne sont pas parfaits et introduisent du bruit. Bien que l'on pourrait inclure un modèle de bruit lors de l'analyse des images, la façon la plus simple d'aborder ce problème est de minimiser directement l'impact du bruit si possible, en réduisant l'ouverture de l'objectif et en augmentant le

temps d'exposition.

4.1.4 Balance des blancs et couleurs

Toutes les techniques dont on va parler travaillent avec des intensités en gris, noir et blanc seulement pour une bonne raison : le même genre de distorsions non linéaires problématiques mentionnées dans la section 3.5 est présent dans les couleurs, souvent en pire. Dans le but de répliquer le plus possible la vision humaine, les caméras ont tendance à prioriser certaines couleurs plus que d'autres. Des filtres de Bayer sont souvent intégrés aux senseurs, dans le but de donner plus de sensibilité au vert, qui a un plus grand poids dans le système visuel humain.

Dans le cas où on souhaiterait tout de même profiter de tout l'espace de couleurs, un calibrage colorimétrique supplémentaire serait nécessaire, et on devrait désactiver la *balance des blancs automatique* qui est souvent activée par défaut. La balance des blancs est une correction appliquée à l'espace de couleurs de l'image pour retirer la contribution de l'éclairage ambiant aux couleurs de l'image. Puisque nos yeux font une correction dans ce style, une mauvaise balance des blancs dans une image apparaît tout de suite bizarre aux yeux d'un humain.

4.2 Synchroniser le système

La dernière problématique à considérer est la synchronisation de la caméra et du projecteur. Le temps requis pour l'affichage d'une image par le projecteur et le temps d'exposition de la caméra doivent être considérés, car simplement lancer une projection continue et, la plupart du temps, une capture vidéo des images va mener à acquérir des images intermédiaires *mixées*, plutôt que des images correctes (Figure 2.2).

Le plus simple, dans la mesure du possible, est d'utiliser un même ordinateur pour contrôler la capture et la projection, ce qui permet d'alterner les deux de façon synchrone. Il est bon de noter ici qu'envoyer une commande `capture` à la caméra ou une commande

`affiche` au projecteur ne va pas forcément s'exécuter aussi rapidement au niveau du matériel que dans le code du programme qui envoie la commande, donc forcer un temps d'attente entre chaque commande peut s'avérer nécessaire.

Alternativement, s'il n'est pas possible d'utiliser une seule et même machine pour contrôler le projecteur et la caméra à cause de contraintes techniques, on peut toujours se baser sur une synchronisation des horloges des machines via un *serveur de temps*, qui permet de garder un temps de référence commun à plus ou moins quelques milliseconde. On peut ensuite envoyer une commande d'avance à chaque machine, indiquant à quels *timestamps* on devrait afficher/capturer.

Une autre possibilité est de passer par une synchronisation matérielle. Certaines caméras disposent d'une entrée *genlock*, qui permet de synchroniser la capture d'une image avec un signal analogue arbitraire. Ces dispositifs sont cependant seulement disponibles sur de l'équipement professionnel.

Finalement, comme plusieurs problèmes mentionnés jusqu'ici, on a toujours l'option de le contourner : certains algorithmes de scan 3D modélisent directement la mixture temporelle causée par une acquisition non synchronisée des images [28, 9].

Chapitre 5

CORRESPONDANCE

Lorsque l'on parle d'un "algorithme de scan 3D", dans la majorité des cas, on fait référence à l'étape de la *mise en correspondance*.

Calibrer puis placer un système caméra-projecteur devant une scène, capturer des images et trianguler des points sont des étapes nécessaires à tous les scans à lumière structurée. La place restante pour la créativité et l'inventivité se trouve dans la façon de choisir les images à projeter – *le codage* – et dans la façon d'analyser les images capturées – *le décodage*.

5.1 Intuition

Stratégie simple

Comment établir la correspondance? Une stratégie très simple serait d'allumer un pixel de projecteur à la fois, puis d'analyser l'image de caméra pour trouver le pixel ayant la plus forte réponse. Plus d'un seul pixel de caméra risque de recevoir de l'illumination : on doit penser aux interréflexions causées par les différentes surfaces de la scène, à l'éclairage ambiant, ou encore à d'autres effets qui peuvent diffuser la lumière (ex.: lorsque la lumière passe à travers des objets plus ou moins translucides), mais le pixel qui reçoit l'illumination la plus forte devrait être la meilleure correspondance possible.

Cette méthode est optimale en ce qui concerne la qualité de la correspondance. Seul petit problème : si on calcule environ 5 images par seconde, récupérer la correspondance complète pour une image HD prendrait $\frac{1920\times1080 \text{ pixels}}{5 \text{ images par seconde}} = 414\,720$ secondes, soit plus d'une centaine d'heures... Uniquement pour la capture, avant même d'avoir analysé les 2 millions d'images.

Augmenter la vitesse de capture, par exemple, avec de l'équipement spécialisé et synchronisé à 1000fps [43], réduirait le temps à environ une demi-heure, ce qui reste plutôt long pour scanner une seule scène... La force brute n'est pas la meilleure solution ici.

Dégradé de tons de gris

Une idée simple pour établir la correspondance entre les pixels de projecteur et les pixels de caméra est d'attribuer un *code* unique à chaque pixel. On pourrait par exemple choisir de projeter une rampe horizontale allant linéairement de noir à blanc, en passant par tous les tons de gris : en observant simplement l'intensité d'un pixel de caméra, sa colonne correspondante dans le projecteur pourrait être retrouvée directement. Le pixel exact pourrait être déduit en répétant l'opération une deuxième fois, avec une rampe verticale.

Il y a deux problèmes majeurs à cette approche : d'abord, l'intensité capturée ne correspond pas directement à l'intensité projetée. Assumant une réponse linéaire de la part de la caméra et du projecteur[1], pour une intensité projetée I_p, l'intensité capturée I_c correspondante inclut minimalement :

$$I_c = \alpha I_p + o \tag{5.1}$$

où α est l'albédo de l'objet au point capturé et o est l'illumination ambiante au point capturé. Même en supposant des senseurs idéaux, on ne peut pas utiliser l'intensité capturée directement sans passer par une étape d'annulation de ces paramètres.

On peut corriger l'intensité capturée en ajoutant deux images, une entièrement noire

[1] L'équipement n'est généralement pas linéaire de base, voir le chapitre 3 pour calibrer la compensation nécessaire.

(intensité normalisée à 0.0), et une entièrement blanche (intensité normalisée à 1.0) :

$$I_{c\,min} = \alpha 0.0 + o \tag{5.2}$$

$$I_{c\,max} = \alpha 1.0 + o \tag{5.3}$$

$$\implies o = I_{c\,min} \tag{5.4}$$

$$\implies \alpha = I_{c\,max} - I_{c\,min} \tag{5.5}$$

$$\implies I_p = \frac{I_c - I_{c\,min}}{I_{c\,max} - I_{c\,min}} \tag{5.6}$$

On reste cependant avec l'autre problème, qui concerne la profondeur des intensités capturées (le *bit depth*) : le standard le plus répandu pour les caméras et projecteurs est de travailler avec des intensités sur 8 bits, donc avec 256 valeurs discrètes. Encoder la position horizontale dans une image *full HD* à 1920 colonnes de pixels impliquerait d'avoir des colonnes de 7 à 8 pixels partageant un même code. En considérant les effets d'albédo et d'illumination ambiante, certains tons de gris distincts dans le projecteur se retrouvent compressés en un même ton de gris du côté de la caméra, surtout si elle est également limitée à 8 bits. En pratique, cette solution simple ne donne pas des résultats précis.

Codes binaires

Si on utilise plutôt un code *binaire* [34] pour encoder la position x, puis un autre pour la position y de chaque pixel (Figure 5.1), on peut utiliser une suite de noir/blanc pour représenter la suite de 0 et 1 qui forment ce code binaire. Pour un pixel de caméra donné, le pixel de projecteur correspondant est retrouvé en reconstruisant le numéro associé à la séquence capturée par ce pixel. Puisque chaque image de la séquence d'images projetées contient une partie du code total, on dit qu'il s'agit d'un code *multiplexé dans le temps*.

Avec cette stratégie, la taille d'un code de pixel n'est pas limitée par la profondeur des images, mais grandit plutôt avec le nombre de lignes/colonnes dans le projecteur. La longueur totale de chaque code est de $log_2(N)$, où N représente le nombre de colonnes ou

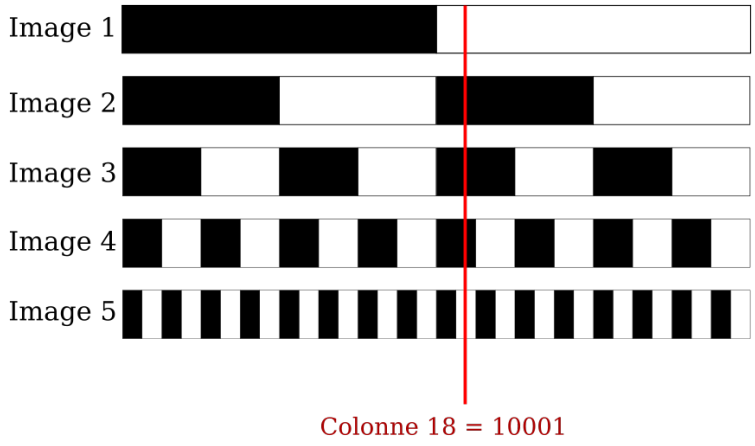

Figure 5.1. Exemple d'encodage binaire pour chaque colonne d'une image

de rangées (selon la dimension récupérée).

5.2 Problèmes potentiels

On peut s'intéresser à l'impact qu'auraient les erreurs lors de la capture. En ce qui concerne les erreurs causées par le matériel, le bruit numérique dans l'image ou la non-linéarité, se limiter à détecter simplement la présence ou absence de lumière en un point est beaucoup plus robuste que d'essayer d'interpréter directement le niveau de gris capturé.

Une erreur dans la première méthode se traduit directement par un déplacement de la position décodée. Une erreur dans l'intensité d'un code binaire ne va pas forcément affecter le code binaire capturé, mais le cas échéant va se traduire par un plus ou moins grand déplacement, selon si l'erreur se trouve dans un des bits plus ou moins significatifs du code. Dans la Figure 5.1 pour la colonne 18, une erreur sur le bit de l'image 5 mènerait à décoder la séquence capturée comme étant 17 plutôt que 18. À l'inverse, une erreur à l'image 1 mènerait ce même pixel à être décodé comme 2 plutôt que 18, déplaçant le pixel de la moitié de l'image totale.

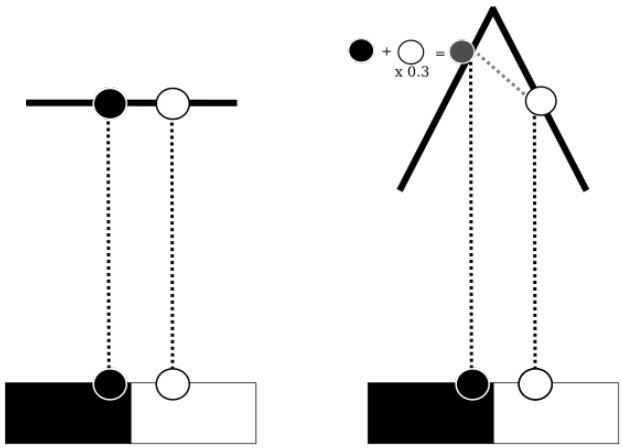

Figure 5.2. *(Gauche)* **illumination d'une scène sans réflexions : l'intensité lumineuse en un point est le résultat de l'illumination directe seulement. (***Droite***) réflexion d'un rayon vers un autre point de la scène : l'intensité lumineuse résultante en un point est modifiée par les rayons réfléchis sur le reste de la scène, ce qui va possiblement corrompre le code capturé.**

Dans le cas des images binaires, la source d'erreur la plus importante n'est pas le bruit, mais plutôt *l'illumination indirecte*, soit l'ensemble des rayons lumineux qui affectent l'intensité d'un point de la scène sans avoir été directement projetés dessus. Elle inclut entre autre l'illumination réfléchie entre les objets de la scène, ou encore la transluminescence des surfaces, qui va diffuser la lumière tout autour du point illuminé.

Réflexion

Dans le cas des codes binaires en particulier, si la longue bande blanche de l'image 1 dans la Figure 5.1 se retrouve reflétée vers un point de la scène qui devrait être noir (Figure 5.2, côté gauche de la scène), l'intensité résultante se retrouve de l'ordre du gris moyen, qui peut être plus facilement confondu pour du blanc lors du décodage. Si on inversait l'image horizontalement, on aurait plutôt un problème du côté droit de la scène à la place.

L'éclairage total en un point dépend donc de la géométrie de la scène ainsi que de

l'image projetée. Si on considère cependant la même situation de réflexion, mais avec l'image 5 à la place, chaque point de la scène devrait recevoir un mélange d'environ 50% de rayons blancs réfléchis et 50% de rayons noirs réfléchis, autant à droite qu'à gauche. La quantité d'illumination indirecte en chaque point ne dépendrait alors que de la géométrie de la scène : inverser ou déplacer horizontalement l'image projetée ne ferait aucune différence, puisque l'illumination réfléchie en chaque point est toujours moitié blanche, moitié noire.

L'image 1 contient des basses fréquences (variations lentes dans l'image), tandis que l'image 5 contient des hautes fréquences (variations rapides). Une stratégie qui pourrait réduire de beaucoup l'impact de l'illumination réfléchie serait de construire des codes pour les pixels qui génèrent *uniquement des images à hautes fréquences* [30]. De cette façon, le passage d'une image à l'autre ne modifierait pas la quantité d'illumination réfléchie reçue en un point donné, et on pourrait donc la considérer comme faisant partie de l'illumination ambiante o. Cette stratégie est l'idée principale utilisée par Gupta et al dans [16]. Des codes binaires à haute fréquence peuvent par exemple être construits en appliquant un ou-exclusif entre un code arbitraire à haute fréquence et chaque code binaire traditionnel.

Limite physique

Les images à hautes fréquences règlent le problème de l'illumination réfléchie, mais le matériel ne permet généralement pas de projeter et capturer des fréquences arbitrairement hautes. Au final, la plus petite largeur de bande contiguë qu'il est possible de projeter et capturer de façon suffisamment précise est d'au moins 4 ou 5 pixels.

La fréquence maximale théorique de $\frac{1}{2px}$, où chaque colonne passe successivement de blanc à noir à blanc ..., n'est généralement pas projetée correctement. Les différentes intensités ont tendance à se diffuser sur les pixels avoisinants, ce qui donne plutôt un gris moyen uniforme sur toute l'image et qui n'est pas utilisable en pratique. Un effet simi-

laire se produit dans la caméra, où les pixels ne capturent pas nécessairement une surface parfaitement alignée à la surface couverte par un pixel projeté. Un pixel de caméra va généralement moyenner les intensités d'une surface contenant plusieurs pixels de projecteur.

Afin de réussir à établir une correspondance aussi précise que possible, on peut se débarrasser des bandes discontinues et plutôt opter pour un motif *continu*. Si on retourne à l'idée initiale de projeter une rampe linéaire, on peut constater que ces effets physiques sont mitigés. Du côté du projecteur, si les intensités des pixels se diffusent un peu sur les pixels avoisinants, par exemple avec

$$\hat{I}_p[x] = \frac{\beta}{2} I_p[x-1] + (1-\beta) I_p[x] + \frac{\beta}{2} I_p[x+1], \tag{5.7}$$

où β est la contribution des pixels avoisinants, entre 0 et 1. Avec une rampe linéaire, la valeur résultante au pixel x reste inchangée, puisque cette image utilise pour l'encodage :

$$I_p[x] = x \tag{5.8}$$

donc

$$\hat{I}_p[x] = \frac{\beta}{2}(x-1) + \beta x + \frac{\beta}{2}(x+1) \tag{5.9}$$

$$= \frac{\beta}{2}x - \frac{\beta}{2} + (1-\beta)x + \frac{\beta}{2}x + \frac{\beta}{2} \tag{5.10}$$

$$= x \tag{5.11}$$

$$\implies \hat{I}_p[x] = I_p[x]. \tag{5.12}$$

Un effet similaire se produit avec le pixel de caméra, qui pourrait moyenner plusieurs pixels voisins avec une proportion de β' :

$$\hat{I}_c[x'] = \alpha\left(\frac{\beta'}{2} I_p[x-1] + (1-\beta') I_p[x] + \frac{\beta'}{2} I_p[x+1]\right) + o \tag{5.13}$$

$$= \alpha\left(\frac{\beta'}{2}(x-1) + (1-\beta')x + \frac{\beta'}{2}(x+1)\right) + o \tag{5.14}$$

$$= \alpha x + o \tag{5.15}$$

$$\implies \hat{I}_c[x'] = I_c[x']. \tag{5.16}$$

Encore une fois, l'effet s'annule ici.

La rampe n'est cependant pas pratique à cause de la profondeur limitée des intensités. Pour avoir de mesures assez précises, on peut augmenter la fréquence, en répétant plusieurs rampes plus minces dans la même image (Figure 5.3), avec

$$I_p[x] \equiv x \mod \frac{1}{f} \tag{5.17}$$

où f est la fréquence de l'image projetée. Avec une fréquence de $\frac{1}{10\text{px}}$, on aurait une rampe de 255 intensités pour encoder chaque bloc de 10 pixels de large, ce qui limiterait en même temps l'illumination réfléchie. Cette nouvelle fonction d'encodage introduit cependant deux nouveaux problèmes :

- Chaque changement de période introduit une discontinuité dans les intensités, ce qui pose le même problème que pour les bandes binaires en ce qui concerne la diffusion des valeurs sur les pixels voisins, et qui ne fonctionnent pas dans l'équation 5.13

- La valeur $I_c[x]$ permet de retrouver x en modulo $\frac{1}{f}$ seulement.

5.3 Déphasage (Phase-shifting)

Pour éviter les discontinuités dans l'image projetée, on peut utiliser une fonction d'encodage périodique et continue, comme un sinus :

$$I_p[x] = 0.5sin(2\pi fx) + 0.5. \tag{5.18}$$

Cette fonction est continue et se répète à chaque période de $\frac{1}{f}$ pixels. La position x dans l'image est encodée dans la *phase* du sinus projeté. On peut récupérer la phase en un point capturé en projetant $N \geq 2$ déphasages d'une même fréquence [41] avec :

$$I_{p,k}[x] = 0.5sin(2\pi fx + \frac{2\pi k}{N}) + 0.5. \tag{5.19}$$

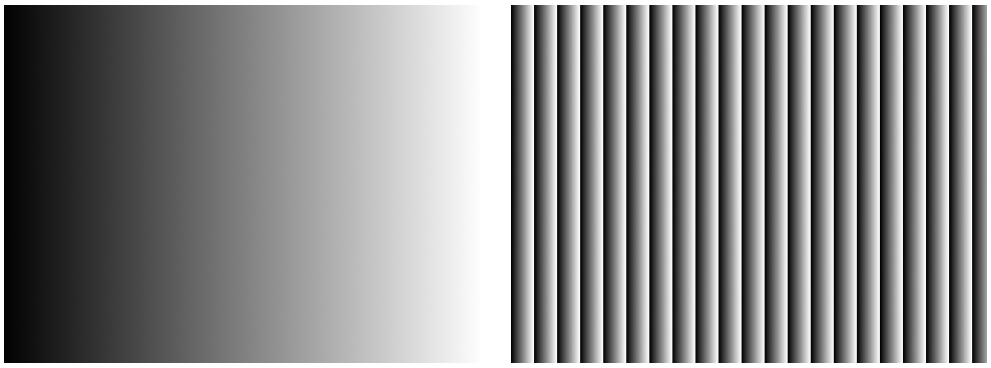

Figure 5.3. Rampe horizontale à basse fréquence (*gauche*) et à haute fréquence (*droite*).

On peut retrouver la phase de départ aux moindres carrés en éliminant la contribution de l'illumination ambiante et de l'albédo avec :

$$\phi = \arctan \frac{\sum_i^N sin(2\pi i/N) I_{c_i}}{\sum_i^N cos(2\pi i/N) I_{c_i}}, \quad \phi \in [-\pi, \pi] \tag{5.20}$$

$$\phi \equiv fx \mod 2\pi \tag{5.21}$$

Le minimum est de $N = 3$ images, mais plus d'images permet une meilleure robustesse au bruit et aux autres déformations. La phase ϕ récupérée est dite "*wrapped*", puisqu'elle permet de retrouver la position x seulement à un multiple de $\frac{1}{f}$ près. En d'autres termes, ϕ correspond à la position à l'intérieur d'une période, mais on ne sait pas quelle période contient le point. Le décodage de la position complète, ou *phase unwrapping*, requiert de l'information supplémentaire.

Une idée très simple est l'*unwrapping spatial*, qui consiste à supposer que chaque discontinuité dans la carte de phase sépare une seule période : pour une rangée donnée, on peut partir de la gauche et graduellement ajouter un multiple de 2π à la fois de façon à obtenir un ϕ *unwrapped* strictement croissant. Cette méthode est malheureusement très peu robuste, puisqu'elle propage le bruit de chaque pixel vers le reste de la carte de phase. Pire : l'hypothèse de base est fausse dans beaucoup de cas. Tant que la scène est

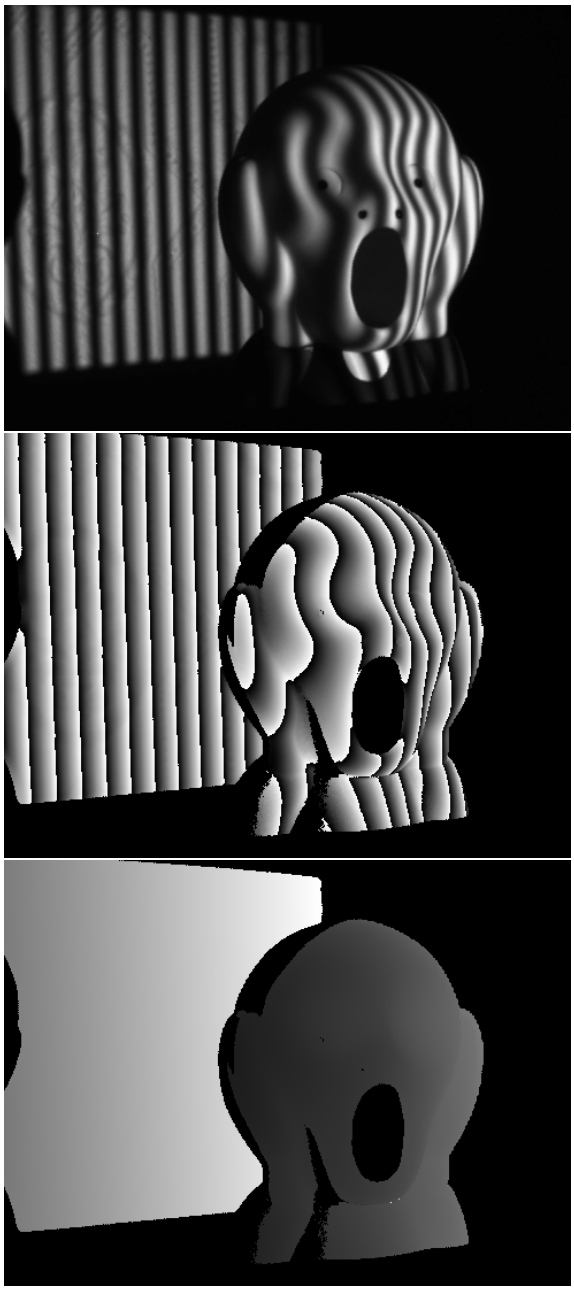

Figure 5.4. Phase-shifting sinusoïdal à haute fréquence (*haut*), carte de phase *wrapped* décodée (*milieu*) et carte de phase *unwrapped* (*bas*).

parfaitement continue, on devrait obtenir un ϕ strictement croissant de gauche à droite, mais cette hypothèse devient fausse en présence de discontinuités de profondeur. Cet effet est observable dans la carte de phase *wrapped* de la Figure 5.4.

Une meilleure alternative est l'*unwrapping temporel*, qui demande plutôt d'ajouter des images supplémentaires. Sansoni et al. [11] proposent d'utiliser des codes binaires pour l'*unwrapping*, donc d'utiliser le code binaire pour la position approximative, et les images sinusoïdales pour la précision. Une autre option est d'utiliser uniquement des images sinusoïdales, mais en variant la fréquence utilisée [20].

On peut par exemple utiliser deux cartes de phase, une récupérée avec des images à très basse fréquence, qui couvre toute l'image en une seule période, et une autre avec des images à très haute fréquence. La carte de phase à basse fréquence fournit un ϕ_{approx}, qui souffre du problème de profondeur limitée des intensités et qui est donc approximatif, mais qui est suffisant pour guider l'*unwrapping* de la carte de phase à haute fréquence ϕ_{precis}. Plus spécifiquement, on peut déterminer la position x exacte avec :

$$x_{approx} = \frac{\phi_{approx}}{2\pi f_{basse}} \tag{5.22}$$

$$Num_{periode} = \lfloor x_{approx} * f_{haute} \rfloor \tag{5.23}$$

$$\hat{\phi}_{precis} = \phi_{precis} + 2\pi Num_{periode} \tag{5.24}$$

$$x_{precis} = \frac{\hat{\phi}_{precis}}{2\pi f_{haute}}. \tag{5.25}$$

Six images projetées sont suffisantes pour récupérer la position horizontale en chaque point. Une seule petite erreur dans la carte de phases approximative peut cependant avoir des impacts importants sur la position finale. On peut augmenter la robustesse en appliquant plutôt ce processus avec une séquence de fréquences de plus en plus basses. Huntley et Saldner ont comparé dans [20] l'utilisation de fréquences suivant une série harmonique :

$$f_i = \frac{1}{f_{base} + ki}, \quad k > 0 \tag{5.26}$$

avec une série géométrique :

$$f_i = \frac{1}{2^i f_{base}} \tag{5.27}$$

et ont trouvé que la seconde minimise la probabilité d'erreur, principalement parce qu'elle réduit le nombre total de fréquences nécessaires pour arriver à couvrir toute l'image en une seule période. Utiliser beaucoup de fréquences réduit la probabilité d'erreur à chaque niveau d'*unwrapping*, mais le fait de faire une erreur à une seule des étapes dégrade les résultats. La série géométrique minimise l'erreur à chaque étape tout en minimisant la probabilité d'erreur due au nombre d'étapes total.

Ces deux méthodes utilisent un minimum de $3N_{freqs}$ images, puisque chaque fréquence est décodée séparément avec l'équation 5.20, qui requiert 3 images au minimum.

Défocus

Appliquer l'équation 5.20 à chaque fréquence peut sembler redondant : en un point donné, l'illumination ambiante et l'albédo de l'objet devraient rester constants, donc on devrait pouvoir récupérer ces paramètres une seule fois, par exemple avec 5.2. Un effet qui est négligé dans cette formule est ce qui se passe quand le projecteur n'est pas au focus pour un point de la scène particulier : le défocus du projecteur agit comme un filtre passe-bas sur l'image. Dans [14], Gupta et Nayar montrent que l'amplitude effective du projecteur dépend de la fréquence de l'image projetée, ce qui modifie en pratique les paramètres o et α capturés.

Si on se limite à une seule fréquence, ou quelques fréquences très proches les unes des autres, o et α peuvent être résolus de façon globale pour toutes les images, ce qui réduit le nombre d'images à utiliser. *Micro Phase-Shifting* [14] propose de faire cette résolution explicitement pour trois déphasages d'une première fréquence, et d'utiliser ces paramètres pour normaliser les intensités des autres images, qui sont des sinusoïdes toutes proches les unes des autres en termes de fréquences, sans déphasages. Avec cette approche, aussi peu que $N_{freqs} + 2$ images peuvent être utilisées. En pratique, les auteurs

montrent des résultats pour des scans réalisés avec seulement 7 images.

On devrait rappeler ici qu'en choisissant des fréquences assez élevées, on peut se sauver de l'impact de l'illumination indirecte due aux interréflexions, ce qui rend Micro Phase-Shifting plutôt robuste dans l'ensemble.

Temps de calcul

Le principal inconvénient de cette méthode est que la position x dans l'image n'est pas calculée directement. La façon de choisir les fréquences est seulement pensée en termes de ce qui maximise la distance entre les codes de chaque pixel, et les intensités résultantes pour chaque pixel ne sont pas suffisantes pour calculer directement une phase ou une position en chaque point. La façon de récupérer la position x correspondant à un pixel de caméra donné est de faire une recherche le long des lignes épipolaires dans le projecteur pour trouver le vecteur d'intensités qui est maximalement corrélé avec l'observation, ce qui augmente considérablement le temps de calcul pour récupérer la correspondance. De plus, cette méthode nécessite absolument d'obtenir au préalable un calibrage épipolaire pour le système (voir la section 3.6).

On peut toutefois retravailler le principe de base et l'améliorer en choisissant une façon plus intéressante de déterminer l'ensemble de fréquences des images à projeter. Moreno et al. [29] proposent une stratégie pour choisir un ensemble de hautes fréquences qui peuvent être décodées en basses fréquences "*embedded*", qui augmentent de façon exponentielle et qui peuvent être utilisées pour le décodage complet de la position x.

Un ensemble de valeurs entières $T_1, ...T_m$ est choisi arbitrairement. Les fréquences *embedded* sont choisies de façon à suivre une relation exponentielle :

$$F_m = \frac{1}{T_1} ... \frac{1}{T_m}. \tag{5.28}$$

Ces fréquences *embedded* servent à générer les hautes fréquences à projeter :

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_m \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & & 0 \\ \vdots & & & \ddots & 0 \\ 1 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_m \end{bmatrix} \tag{5.29}$$

ou de façon plus concise :

$$f_1 = F_1 \tag{5.30}$$

$$f_i = F_1 + F_i. \tag{5.31}$$

Les fréquences f_i sont toutes élevées et très proches les unes des autres, alors que les fréquences *embedded* F_i sont graduellement de moins en moins hautes, similairement aux fréquences utilisées dans l'*unwrapping temporel* traditionnel.

Chaque fréquence peut utiliser un minimum de 2 shifts, à condition que le nombre d'images total soit $N_{images} \geq 2N_{freqs} + 1$, question que le système linéaire utilisé pour le décodage ne soit pas sous-déterminé :

$$\mathbf{u} = [o, c_1, s_1, c_2, s_2, ..., c_m, s_m]^T, \tag{5.32}$$

$$c_m = \alpha \cos(2\pi f \phi_m) \tag{5.33}$$

$$s_m = \alpha \sin(2\pi f \phi_m) \tag{5.34}$$

$$A_m = \begin{bmatrix} cos(\theta_{m,1}) & -sin(\theta_{m,1}) \\ \vdots & \vdots \\ cos(\theta_{m,N_{freqs}}) & -sin(\theta_{m,N_{freqs}}) \end{bmatrix} \tag{5.35}$$

$$\mathbf{u} = \underset{\mathbf{u}}{argmin} ||\mathbf{I}_c - A\mathbf{u}|| \tag{5.36}$$

où $\mathbf{I}_c$ est le vecteur des intensités capturées par la caméra. On peut utiliser le vecteur $\mathbf{u}$ récupéré pour calculer chaque carte de phase à haute fréquence :

$$\phi_m = \arctan \frac{s_m}{c_m} \tag{5.37}$$

et les cartes de phases *embedded* peuvent être rapidement décodées avec :

$$\begin{cases} \Phi_1 = \phi_1 \\ \Phi_m = \phi_m - \phi_1 \quad \text{si } m > 1. \end{cases} \tag{5.38}$$

On peut finalement obtenir un *unwrapping* des cartes de phases Φ_2 à $\Phi_{N_{freqs}}$ en utilisant l'*unwrapping temporel* à plusieurs fréquences, Φ_2 correspondant à la fréquence f_2, qui est la plus élevée. On peut également profiter du fait que les cartes de phases ϕ_m sont *toutes* à haute fréquence, ce qui implique qu'elles sont toutes plutôt précises. On peut améliorer la qualité de la correspondance de plus belle en agrégeant les ϕ_m (par exemple, avec la moyenne ou la médiane), puis en faisant un *unwrapping* de la carte de phases résultante avec la carte Φ_2 *unwrapped*.

Les opérations mathématiques nécessaires pour effectuer ces calculs sont toutes simples. Le temps de calcul en est significativement réduit par rapport à la recherche demandée par Micro Phase-Shifting. Cette méthode peut soit utiliser un calibrage épipolaire et résoudre la correspondance en 2D à partir d'images 1D seulement, soit utiliser deux correspondances 1D orthogonales.

Modèle général et codage/décodage optimaux

Dans *Optimal Structured Light a la Carte* [27], Mirdehghan et al. s'intéressent à concevoir une façon optimale d'encoder et de décoder les positions, dans le but de minimiser le nombre d'erreurs de correspondance.

L'encodage des positions se fait sur mesure pour un système caméra-projecteur donné, en cherchant à minimiser le nombre d'erreurs de décodage probables pour des contraintes données (nombre d'images, fréquence maximale, distance entre la caméra et le projecteur, ...) dans l'espace des images possibles.

Pour le décodage des images, leur conclusion est que la meilleure mesure à utiliser en termes de maximum de vraisemblance est la *corrélation croisée entre les vecteurs*

d'intensités centrés à zéro et normalisés, ou *ZNCC* :

$$ZNCC(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}-\bar{\mathbf{a}}}{||\mathbf{a}-\bar{\mathbf{a}}||} \cdot \frac{\mathbf{b}-\bar{\mathbf{b}}}{||\mathbf{b}-\bar{\mathbf{b}}||} \tag{5.39}$$

Pour décoder une position, l'algorithme cherche simplement à maximiser cette mesure, en considérant tous les pixels possibles le long des lignes épipolaires.

Bien que beaucoup plus robuste à toutes sortes de dégradations, la méthode souffre cependant des mêmes problèmes que Micro Phase-Shifting, soit la nécessité de disposer d'un calibrage épipolaire et le temps nécessaire pour la recherche des correspondances.

5.4 Images arbitraires et scan non-synchronisé

La qualité de la correspondance n'est pas forcément la seule chose à considérer dans le choix d'un algorithme. Dans certaines situations, la possibilité de s'éviter une partie des calibrages peut être intéressante. Si on prend l'exemple des non-linéarités introduites sur les intensités par le projecteur ou par la caméra, comme c'est le cas avec un γ (chapitre 3.5), les codes binaires peuvent être utilisés directement, sans avoir à calibrer de correction sur les intensités, tandis que cette déformation dégrade significativement les résultats d'un phase-shifting sinusoïdal. Les images binaires posent cependant problème avec la résolution possible du scan ainsi qu'avec les discontinuités d'intensités.

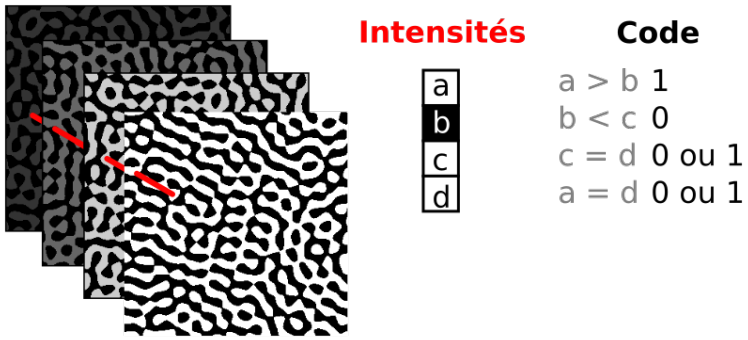

Figure 5.5. Images non-structurées et code binaire associé à l'un des pixels.

Dans *Robust Unstructured Light* [7], Couture et al. se basent sur l'algorithme *Unstructured Light* [24] (lumière non-structurée) et proposent de construire des images binaires aléatoires en 2D, en restant dans une bande mince de fréquences élevées. La Figure 5.5 illustre les images projetées résultantes. Pour éviter d'avoir à résoudre l'albédo explicitement, le code binaire associé à chaque pixel est déterminé à partir de différences d'intensités consécutives I_1 et I_2, avec :

$$bin(I_1, I_2) = \begin{cases} 1 & \text{si } I_1 > I_2 \\ 0 & \text{si } I_1 < I_2 \\ 0 \text{ ou } 1 \text{ tiré au hasard si } I_1 = I_2. \end{cases} \tag{5.40}$$

La première et la dernière image sont considérées "consécutives", ce qui donne un code de la taille du nombre d'images projetées N_{images}.

Puisque les images sont générées au hasard, on ne peut pas décoder une position directement à partir des intensités observées. Comme pour OptimalSL [27] et pour MicroPS [14], une recherche dans le domaine des intensités est nécessaire. Puisque cette méthode fonctionne directement en 2D, une recherche exhaustive sur toute l'image serait trop coûteuse ($\approx$ 2 millions de candidats par pixel pour une image HD). La recherche est plutôt faite de façon approximative avec l'algorithme *LSH* [4], un algorithme de Monte-Carlo. La méthode LSH consiste à *hacher* des données avec une fonction qui calcule un même hachage pour des valeurs similaires, puis de chercher des collisions entre les données dans une table de hachage.

Dans [7], les codes binaires projetés sont hachés et insérés dans une table en utilisant simplement un sous-ensemble des bits tiré au hasard, puis les codes binaires capturés sont hachés de la même façon en vue de trouver des collisions. Lorsqu'une collision est trouvée entre un pixel de caméra et un pixel de projecteur, cette paire de pixels devient un candidat pour la correspondance. En variant la fonction de hachage (en sélectionnant un nouveau sous-ensemble des bits au hasard), on peut répéter le processus et obtenir des collisions différentes, qui vont générer des candidats supplémentaires. Pour chaque paire

de pixels candidate, la qualité de la mise en correspondance de ces deux pixels est évaluée avec la distance de Hamming sur les codes binaires complets, et le meilleur candidat est conservé à la fin.

Grâce à sa nature probabiliste, et en utilisant des codes plus gros que le strict minimum de bits requis pour encoder uniquement chaque pixel (puisqu'il s'agit de codes binaires, le minimum serait de $\log_2(W \times H)$), *Robust Unstructured Light* offre une certaine redondance dans les codes : une partie des codes peut être complètement erronée, sans que ça empêche l'algorithme de récupérer la mise en correspondance correctement. Une telle corruption des bits peut se produire avec les discontinuités dans les intensités noir/blanc, ce qui ne pose pas problème à l'algorithme.

Une particularité de cet algorithme est le fait de pouvoir faire un *décodage inverse*, en cherchant le pixel de caméra associé à chaque pixel de projecteur, plutôt qu'en décodant le pixel de projecteur associé à un pixel de caméra. Les algorithmes qui décodent directement les intensités capturées ne peuvent pas faire ce décodage, qui peut avoir une application dans le calibrage du projecteur (voir la section 3.4 à ce sujet). Cette particularité est discutée plus en détail dans la section 6.3.6 au prochain chapitre.

Bien que les images binaires aient été conçues pour être à haute fréquence en deux dimensions, la mise en correspondance n'utilise pas d'information a priori sur le contenu des images capturées et projetées, ce qui rend l'algorithme utilisable avec n'importe quel jeu d'images. L'algorithme *Unstructured Light*, duquel est dérivé *Robust Unstructured Light* utilisait d'ailleurs cette propriété pour démontrer la possibilité de réaliser un scan à partir de photos de famille. Une conséquence importante de cette propriété est la possibilité de résoudre le problème des images temporellement mixées lors d'un scan avec de l'équipement désynchronisé.

L'image présentée à la Figure 2.2 contient un mélange de deux images consécutives, capturées dans un même temps d'exposition par la caméra. À condition de connaître la proportion de chaque image dans le mélange temporel, on peut en théorie re-mélanger les images projetées originales pour retrouver une image correspondante du côté du pro-

jecteur. On peut ensuite exécuter l'algorithme LSH sur ce nouvel ensemble d'images caméra/projecteur, et récupérer une correspondance malgré le fait que l'exposition de la caméra n'est pas synchronisée avec le projecteur. Cette idée est proposée par El Asmi et Roy dans [9], ce qui leur permet de capturer un scan complet avec 60 images non-structurées en 1 à 2 secondes.

5.5 Sous-pixel

Jusqu'ici, la correspondance est sous-entendue comme étant de pixel à pixel, autrement dit, que chaque pixel de caméra est associé à un pixel de projecteur. Le vrai monde n'a cependant pas cette contrainte, et un pixel de caméra n'a pas à être parfaitement aligné avec un pixel de projecteur.

Certaines méthodes mentionnées jusqu'ici peuvent attribuer à un pixel de caméra une *position arbitraire* dans l'image du projecteur, plutôt que de devoir forcément associer des positions discrètes. Si on reprend l'exemple de la rampe linéaire, on peut considérer ce qui se passe lorsqu'un pixel de caméra capture un mélange linéaire de deux pixels voisins. Pour une correspondance $x' \leftrightarrow x$, on a :

$$\hat{I}_c[x'] = \alpha x + o. \tag{5.41}$$

Si la caméra capture plutôt un mélange linéaire de deux voisins en proportion $\lambda \in [0,1]$, on a plutôt :

$$\begin{aligned}
\hat{I}_c[x'] &= \alpha((1-\lambda)I_p[x] + \lambda I_p[x+1]) + o && (5.42)\\
&= \alpha((1-\lambda)x + \lambda(x+1)) + o && (5.43)\\
&= \alpha(x - \lambda x + \lambda x + \lambda) + o && (5.44)\\
&= \alpha((1-\lambda+\lambda)x + \lambda) + o && (5.45)\\
&= \alpha(x+\lambda) + o && (5.46)\\
\implies & x' \leftrightarrow x + \lambda. && (5.47)
\end{aligned}$$

Ce décodage donne effectivement une correspondance fractionnaire entre deux pixels de projecteur, qui donne directement la proportion λ de l'interpolation linéaire. La relation est simplifiée ici, mais les algorithmes de Phase-shifting sinusoïdal amènent au même résultat.

De base, les algorithmes [27] et [7] se limitent à établir une correspondance pixel à pixel, mais des algorithmes de *raffinement sous-pixellique* peuvent être appliqués pour augmenter leur résolution. En particulier, [25], [10] et [21] tirent profit des images déjà capturées pour la correspondance pixel à pixel pour interpoler un déplacement supplémentaire plus petit que 1px.

Bien qu'une différence inférieure à un pixel puisse sembler négligeable dans la correspondance entre deux images HD, la différence entre un scan avec une résolution au pixel près et un scan au sous-pixel près est visuellement perceptible sur la reconstruction 3D. La Figure 5.6 montre la différence entre un scan réalisé au pixel près, avec [7], et le même scan après un raffinement sous-pixellique avec [21].

Sous-pixel et discontinuités

Un dernier détail à considérer est au sujet de la relation entre le déplacement sous-pixel causé par l'alignement des pixels de caméra avec ceux du projecteur, et ce qui se passe lorsque des discontinuités de profondeurs sont capturées *au sein d'un même pixel*. Dans le cas d'une discontinuité de profondeur, l'intensité $I_c[x']$ capturée par la caméra ne correspond pas à un mélange d'intensités contiguës dans le projecteur, mais plutôt un mélange quelconque de deux intensités projetées :

$$I_c[x'] = \lambda(\alpha_1 I_p[x_1] + o_1) + (1-\lambda)(\alpha_2 I_p[x_2] + o_2) \tag{5.48}$$

Ce sujet est examiné en détail dans l'article présenté au chapitre qui suit.

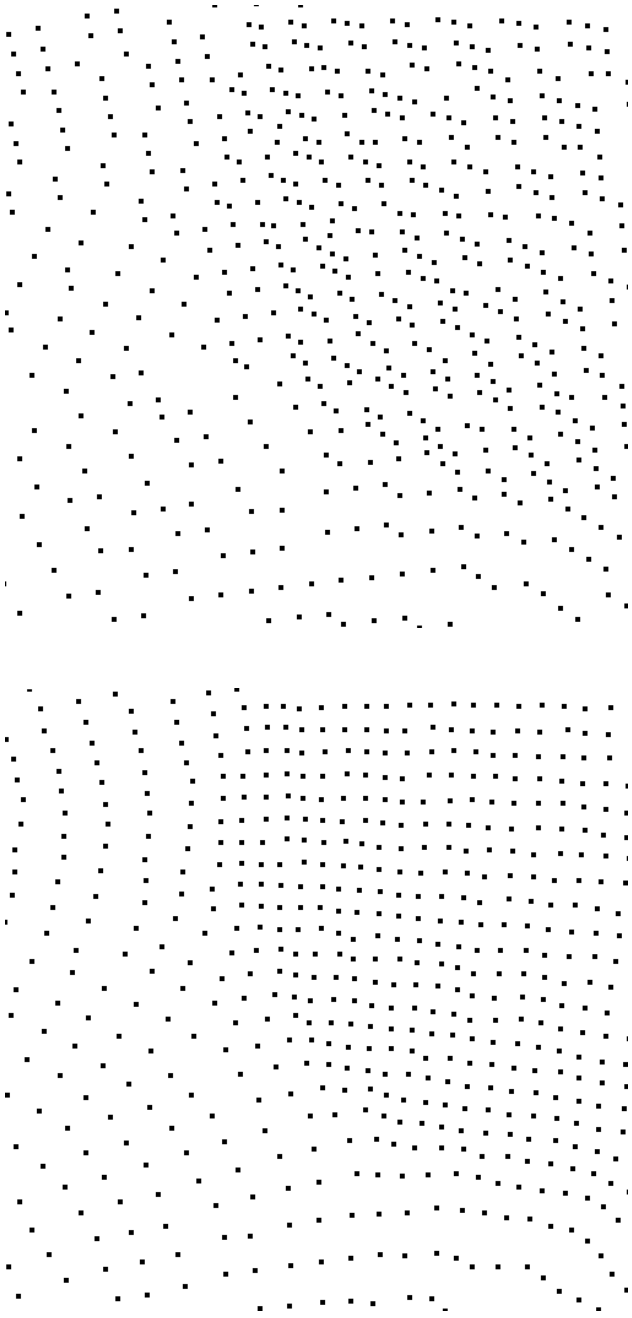

Figure 5.6. Scan d'une surface continue et courbée avec un algorithme de scan au pixel près (*haut*) et raffinement sous-pixellique de ce scan (*bas*). Les points 3D issus du scan au pixel près sont contraints à un alignement avec la grille de positions entières propre aux pixels de la caméra, ce qui cause des artefacts visibles à l'oeil sur la reconstruction.

Chapitre 6

(ARTICLE) FAST DISCONTINUITY-AWARE SUBPIXEL CORRESPONDENCE IN STRUCTURED LIGHT

Cet article [21] a été publié tel qu'indiqué dans la bibliographie :

> Nicolas Hurtubise et Sébastien Roy. Fast discontinuity-aware subpixel correspondence in structured light. Dans *2020 International Virtual Conference on 3D Vision (3DV)*, pages 1108–1116, Novembre 2020.

Cet article est la continuité de ce qui a été présenté dans le chapitre précédent et considère le lien à faire entre le rafinement sous-pixellique d'une correspondance entière et la détection des discontinuités de profondeurs dans une scène.

L'article est présenté ici dans sa version originale.

Abstract

Structured light-based 3D scanning presents various challenges. While robustness to indirect illumination has been the subject of recent research, little has been said about discontinuities. This paper proposes a new discontinuity-aware algorithm for estimating structured light correspondences with subpixel accuracy. The algorithm is not only robust to common structured light problems, such as indirect lighting effects, but also identifies discontinuities explicitly. This results in a significant reduction of reconstruction artifacts at objects borders, an omnipresent problem of structured light methods, especially those relying on direct decoding. Our method is faster than previously proposed robust subpixel methods, has been tested on synthetic as well as real data and shows a significant improvement on measurement at discontinuities when compared with other state-of-the-art methods.

6.1 Introduction

In the field of 3D reconstruction, structured light denotes a family of techniques where a known image or set of images is projected onto a scene and captured back with a camera. Unlike stereovision algorithms, these methods allow for scene-independent dense reconstructions, since the number of acquired points is not limited by the presence of textured elements in the scene [35]. The precision of a structured light system is related to its hardware components as well as the algorithm used to retrieve the correspondences. While some algorithms work at pixel-level, where the precision is limited by the camera and projector pixel grids, others can retrieve correspondence at a finer level which is known as *subpixel* precision.

Although structured light methods can achieve dense 3D reconstruction fairly well, most traditional algorithms make some inaccurate assumptions about the captured scene, which can degrade reconstructions in some situations. For instance, neglecting the impact of indirect illumination (inter-reflection, subsurface scattering, defocus, ...) can lead to

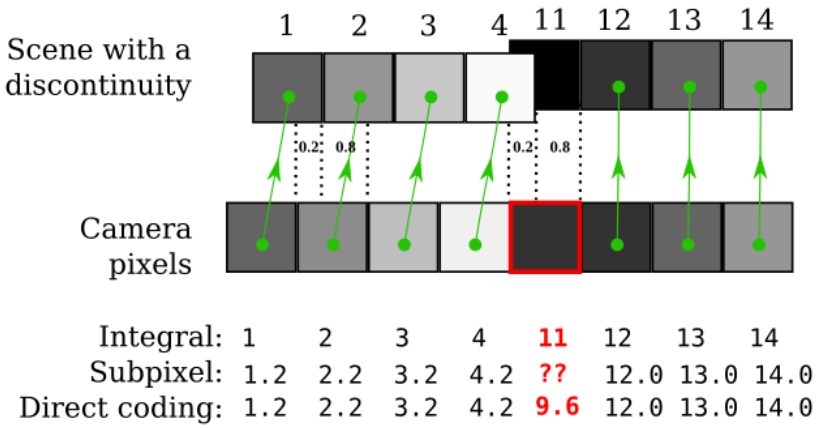

Figure 6.1. Subpixel displacement and discontinuities both appear as intensity interpolation. Interpolation at a depth discontinuity (*pixel in red*) corrupts the captured intensity.

severe degradation of some scenes and has been adressed numerous times over the past years [30, 16, 7, 14, 29, 17].

Another problem which has received much less attention occurs at scene discontinuities : as illustrated in Fig. 6.1, a camera pixel does not always capture a mixture of intensities from neighboring projected points. At a discontinuity, the mixture of multiple scene depths results in a corrupted intensity measure (displayed as a red square in Fig. 6.1). For structured light methods with a *direct decoding* strategy, where the projected code at a given pixel directly encodes its position in the image, this typically results in a decoded position which is unrelated to the depths observed at that camera pixel. In practical situations, the typical approach to deal with noisy discontinuities is to post-process the acquired data with a spatial median filter on the correspondence map. However, this strategy can introduce arbitrary displacement of 3D reconstructed scene points as well as causing the loss of small details [25].

Since the source of these errors is the direct decoding of noisy pixel codes, algorithms

using an *indirect decoding strategy*[24, 7, 27] provide an alternative to increase robustness at discontinuities. One such method based on unstructured light [7], has been shown to accurately find pixel correspondences at discontinuities. A big downside of this method is that its accuracy is limited to pixel-to-pixel discrete matches. While this problem has been addressed before, current proposed solutions require slow operations such as a hierarchical voting scheme [25] or a per pixel finite differences gradient descent [10], and none of them consider the accuracy of their subpixel matches for pixels located at scene discontinuities.

6.1.1 Interpretation of a "subpixel discontinuity"

As most methods struggle to estimate even a *pixel*-accurate match (or integral match) at discontinuities, little has been said about how a subpixel position at a discontinuity should be interpreted. A subpixel position refers to a displacement smaller than one pixel, often resulting from the misalignment of the camera pixel with the projected pixel.

In the case of a depth discontinuity, a camera pixel will however observe multiple surfaces, each with its own subpixel displacement, with one surface partially occluding the other one. Moreover, the occluding surface border is itself localized at some subpixel location. These three subpixel values, i.e. the displacement of both surfaces and the extent of the occluding surface, cannot be expressed as a single subpixel measurement at the discontinuity. We therefore argue that computing a single "subpixel correspondence" for a pixel observing a discontinuity is meaningless, as depicted in Fig. 6.1 by the "*??*" subpixel match. However, the integral match still has some usefulness as it represents the dominant surface in that pixel (see Fig. 6.1, *Integral match 11*). Notice that at a discontinuity, direct coding methods will provide a wrong match (Fig. 6.1, *Direct decoding match 9.6*) resulting from the interpolation of unrelated scene pixels.

The naive approach to identify discontinuities would be to consider that a camera pixel contains a discontinuity if its neighbors' matches are discontinuous, i.e. vary more

than, for example, a single pixel. In Fig. 6.1, such an approach would flag integral match 4 as a discontinuity (since its left and right neighbors are matching 3 and 11) as well as match 11 (with neighbors matching 4 and 12). It is clear, however, that not all pixels with discontinuous neighbor matches are actual discontinuities. In Fig. 6.1, only match 11 is a true discontinuity, where two surfaces are seen in the same pixel, and pixel 4 is part of a surface.

This paper proposes to flag discontinuities by relying on subpixel matching evaluation. Once identified, these pixels can either be removed from the final reconstruction, or kept at a pixel-accurate value, thus avoiding the extrapolation of false subpixel data.

Our main contributions for this paper are the introduction of a mathematical formulation with a fast and simple analytic solution for the computation of subpixel displacement from a pixel-accurate 2D correspondence, and a method to accurately address scene discontinuities.

6.2 Previous work

According to the classification proposed by Salvi et al. in their surveys of structured light patterns [35, 36], our method falls in the continuous time-multiplexing category. At the time of these surveys, the vast majority of algorithms relied on directly decoding the captured images. Since then, multiple methods based on *indirect decoding* have been proposed [24, 7, 27].

6.2.1 Direct decoding methods

Direct decoding methods, by far the most well-known and most widely used methods, rely on codes that can be decoded to directly recover the corresponding projector position of a camera pixel. These methods tend to require a relatively low number of patterns compared to indirect decoding methods, but are more sensitive to noise in the captured intensities, which induce errors in the decoded positions. Most direct decoding strategies

are either based on discrete binary codes [34] or sinusoidal phase shifting [41].

Binary coded patterns use black and white stripes [34] to create a unique code for each projector column. The resulting patterns range from low to high spatial frequencies. As the captured patterns are binarized during the decoding stage, the impact of camera noise and other small interference becomes negligible. The main drawback of binary codes is in their discrete resolution : additional patterns must be projected in order to achieve subpixel accuracy [13, 40]. Sinusoidal phase shifting consists in projecting shifted versions of a periodic sinusoidal grading onto the scanned surface. Each point can be decoded by computing the phase at this position. Since the phase is a continuous value, positions are obtained with subpixel accuracy. High-frequency phase-maps allow for more precise measurements, but because the pattern is periodical, they have to be unwrapped. This is typically achieved by using additional lower-frequency images [20].

Both strategies rely on low-frequency illumination to obtain the full codes. Low-frequency patterns are unfortunately a major source of error in structured light [30, 16], as they are a source indirect illumination which can corrupt the observed codes. Multiple techniques have been introduced to solve this problem, ranging from explicit separation of direct and global illumination [6, 12] to minimizing the indirect illumination itself by constraining the projected patterns to use a narrow band of high frequencies [7, 16, 14, 29, 27, 17].

6.2.2 Indirect decoding methods

Indirect decoding methods consist in projecting codes that do not directly encode projector positions, thereby allowing for arbitrary patterns to be used [24]. The patterns can thus be designed to target specific problems, such as indirect illumination [7].

One indirect decoding algorithm [7] has been shown to accurately retrieve correspondences at discontinuities. In this method, 2D patterns using a narrow spatial frequency band are generated randomly. Binary pixel codes are then formed from the differences

of pattern intensities. Since the projected codes are random, the correspondence must be solved with a nearest neighbors search between the observed camera codes and the projected codes. This is done efficiently through the probabilistic LSH algorithm [4].

The binary nature of the codes allows for increased robustness, which becomes especially important at discontinuities : when two projector codes are mixed together, the resulting code is the one associated to the *strongest* of those codes, rather than an unrelated interpolation between the two. It is interesting to note that unlike direct decoding strategies, the nearest neighbors search between codes can be done *both ways*, allowing to "decode" projector matches using the camera as a reference, leading to a dense correspondence map for the projector view as well as for the camera.

While the original method works at pixel-level, Martin et al. [25] adapted it to recover subpixel information, at the cost of using a time-consuming voting scheme. El Asmi and Roy [10] proposed to compute subpixel-precise matches through a gradient descent instead. To simplify the optimization, continuous codes are derived from the unstructured patterns to replace the original discrete binary codes. Neither methods explicitly consider the problem of *subpixel* correspondence for pixels located at depth discontinuities.

Recently, some authors have focused on algorithms to *design* optimal patterns for a given scene [27, 17]. Mirdehghan et al. [27] proposed an algorithm to design smooth patterns that maximize the distance between projected codes. They propose to use the zero-mean normalized cross-correlation as a decoding function, which they show to be optimal with respect to the number of decoding errors regardless of the projected pattern. This decoding function is indirect, as it requires a linear search for correspondences on epipolar lines. One aspect of this method is that a prior epipolar calibration of the system must be performed, which makes it unsuitable in some situations. For instance, computing correspondence maps for setups equipped with unusual tricky-to-calibrate cameras [42] or even the epipolar calibration itself [44] are impossible using this method. Furthermore, the patterns are optimized for pixel-level accuracy and no subpixel computation is proposed.

6.2.3 Depth discontinuities

The specific problem of depth discontinuities has received little attention over the years. Related work include [33], in which Park et al. rely on custom patterns to identify the scene edges without requiring a full 3D reconstruction. This methods exploits the intensity corruption illustrated in Fig. 6.1 to construct a much more accurate edge map than what can be obtained with standard image processing filters.

The problem of retrieving correspondences for pixels observing depth discontinuities has recently been addressed by Zhang et al. in [46], where this optical phenomenon as well as a few others are modeled as *bimodal multi-path* light transport. Although their method requires a supplementary calibration step, they obtain very impressive results by considering the impact of constructive and destructive interference when two signals are mixed into a single camera pixel. Because of the reliance on a spectral cue, their approach unfortunately cannot easily be applied to arbitrary structured light patterns [7, 27] which have other interesting properties such as enabling scans using unsynchronized equipment [9].

6.3 Fast subpixel correspondence

In order to efficiently find the subpixel-accurate correspondence, we start with a pixel-accurate match obtained from the robust unstructured light algorithm [7]. While our subpixel method does not strictly dependent on using this as a starting point, we chose this algorithm because of its robustness to depth discontinuities and to global illumination, among other things.

Because the pixel-accurate match is already known, the problem to resolve is to recover the fractional part. A pixel-accurate match gives us information about the corresponding camera and projector codes at every pixel. The pixel code V at a position (x, y)

is simply defined as the zero-mean normalized vector of image intensities I:

$$V[x,y] = \frac{I[x,y] - \bar{I}[x,y]}{||I[x,y] - \bar{I}[x,y]||}. \tag{6.1}$$

The length of this vector is the same as $N_{patterns}$, the number of projected patterns. Normalizing serves to effectively cancel the impact of the global illumination and of the object albedo on the codes. The best correspondence between two given codes $V[x,y]$ and $V'[x,y]$ is defined as in [10], when the approximate angle between the two vectors is minimal:

$$cost(V[x,y], V'[x,y]) = 1 - V[x,y] \cdot V'[x,y]. \tag{6.2}$$

Note that minimizing this cost value is the exact same as maximizing the zero-mean normalized cross-correlation between codes, as proposed by Mirdehghan et al. [27]. They showed this cost function to be the optimal way of evaluating the quality of a correspondence between a camera code and a candidate projector code with respect to the number of matching errors [26].

6.3.1 Improving the pixel-accurate match

An important aspect of the matching algorithm used for unstructured light [7] is the fact that all pixels are independently matched, without any assumption of smoothness. While this allows for very small regions to be accurately reconstructed, it does not take into account the probable spatial structure of the scene when looking for new matches. This can result in few pixels with a wrong correspondence in the middle of large smooth regions. Another problem arises from the fact that codes are locally correlated. Because of this, pixels can end up being matched with a slightly off position, for instance, 2 or 3 pixels away from their actual match, since neighboring codes are almost perfect matches for them [10].

To ensure that each camera pixel has a locally-optimal match, we propose a simple heuristic to propagate good match across neighbors. If a camera pixel (x,y) is a good

match for a projector pixel (x', y'), then $(x \pm 1, y)$ is probably a good match for $(x' \pm 1, y')$ – and respectively $(x, y \pm 1) \rightarrow (x', y' \pm 1)$. The quality of this new match can be assessed with Eq. 6.2, updating the match only when the new cost is lower. This can be done recursively with each newly created match, which would propagate good matches all over their connected regions.

However, directly using this recursive definition results in a very inefficient algorithm. Instead, we propose to propagate good matches in a way that is heavily inspired by the propagation phase of PatchMatch [5]. Once the LSH algorithm has completed, we iterate over the camera pixels in the scanline order *top to bottom, left to right* and we try to create new matches along the scanline direction (*right* and *bottom*). Every other iteration, the reverse scanline order is used with matches created in the corresponding direction (*left* and *top*), as shown in Fig. 6.2.

Restricting new matches to be created along the scanline order is an efficient way of propagating new matches to the neighbors without resorting to a recursive algorithm. This is repeated until no better match is found, and typically only a few iterations are necessary. Contrary to the typical soft smoothness constraints often found in stereovision algorithms, this heuristic adds prior information about the observed scene without degrading real discontinuities. Correspondences are only be updated if the propagation yields a better match with respect to Eq. 6.2.

6.3.2 Subpixel accuracy

Assuming that a region is smooth (ie. not discontinuous) and that the pixel ratio is favorable to the camera, camera pixels will always see at most a mixture of four projector pixels on a 2×2 grid. For subpixel matching, we model this mixture as bilinear since the displacement is in two dimensions.

If the pixel ratio is instead favorable to the projector, each camera pixel will average a region of locally correlated pixels. Using the same subpixel model regardless of the

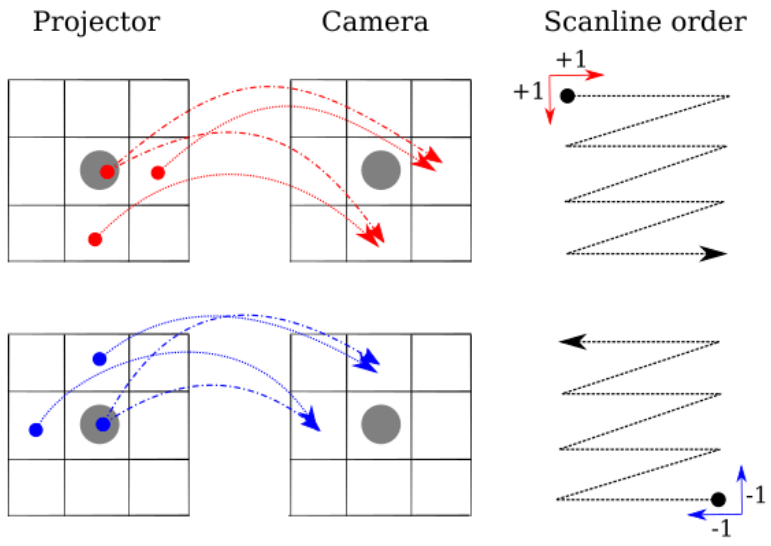

Figure 6.2. Spatial propagation heuristic during even (*top*) and odd iterations (*bottom*). Arrows represents potentials new matches.

pixel ratio would require that the average intensity of a small correlated neighborhood in a pattern is approximately equal to the intensity at its midpoint. This hypothesis is implicitly made when scanning a scene using phase shifting with a camera having a lower resolution than the projector. While it does not strictly hold true for all regions of either sine patterns or unstructured patterns, it is an acceptable approximation for most pixels (see Fig 6.3). Thus, we propose to use the same bilinear model for both situations. The validity of this hypothesis is verified empirically in Section 6.5, where our experimental setup has a pixel ratio favorable to the projector.

This bilinear model combined with the cost function in Eq. 6.2 greatly simplifies the task of retrieving the subpixel position, as it leads us to a simple analytic solution.

6.3.3 A note on quadrants

Given a discrete match between camera pixel p and projector pixel p', we compute the subpixel displacement (δ_x, δ_y) such that $p + (\delta_x, \delta_y)$ is the exact camera match for p'. Since (δ_x, δ_y) can be located in any of the four quadrants around p' {*top left*, *top right*, *bottom left*, *bottom right*}, possible values for the subpixel match will be computed in each quadrant, and only the best final value will be kept.

We should note that the displacement can always be computed in an arbitrary quadrant, simply by adjusting the position of the initial discrete match. For the following subsections, subpixel computation will be shown for the *bottom right* quadrant only, with a subpixel displacement $(\lambda_x, \lambda_y) \in ([0,1],[0,1])$.

6.3.4 Subpixel computation

Our key insight is that for a subpixel position (λ_x, λ_y), the captured camera intensities should all be bilinear mixes of the projected intensities weighted by (λ_x, λ_y). Each captured intensity will thus be a constraint on where the subpixel could fall.

Let (x, y) be an integer position in the camera, let the vector $V[x, y]$ be the observed

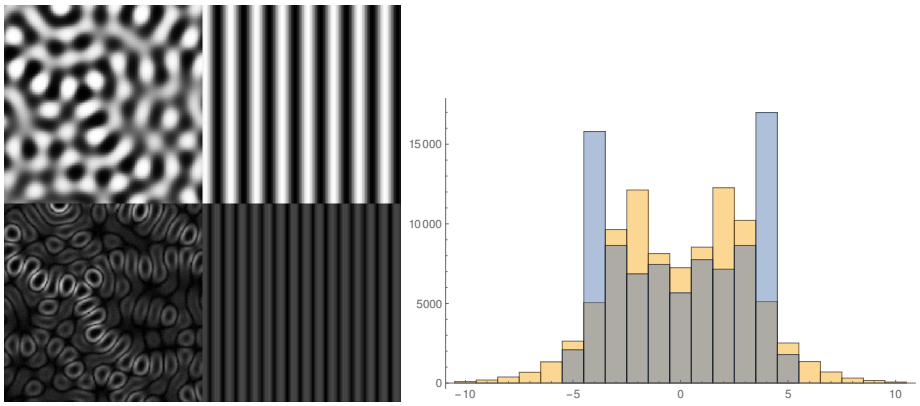

Figure 6.3. (*Left*) an unstructured pattern used in our method and a sine pattern used by phase shifting methods, with their respective heatmaps showing the differences between each pixel's intensity and the average of its 3×3 neighborhood. Pixels in dark regions have intensities that better approximate their neighborhood average than lighter regions. (*Right*) Distribution of those differences for the unstructured pattern (*orange*) and for the sine pattern (*blue*). Differences are shown for 8-bit grayscale patterns using the same spatial frequency.

camera code at this position and let $V'[x,y]$ be the corresponding code for its integer match in the projector. $V[x,y]$ is maximally correlated with the bilinearly interpolated code $V'[x+\lambda_x, y+\lambda_y]$ which corresponds to its subpixel position (λ_x, λ_y):

$$\begin{aligned} & V'[x+\lambda_x, y+\lambda_y] = \\ & \quad (1-\lambda_y)((1-\lambda_x)\, V'[x,y] \quad\quad + \lambda_x V'[x+1,y]) \\ & + \quad\quad \lambda_y \;((1-\lambda_x)\, V'[x,y+1] \; + \lambda_x V'[x+1,y+1]). \end{aligned} \tag{6.3}$$

This provides one bilinear surface per projected intensity. Let us define $S_i(\lambda_x, \lambda_y) = V_i'[x+\lambda_x, y+\lambda_y]$ for clarity. By intersecting $S_i(\lambda_x, \lambda_y)$ with $V_i[x,y]$,we obtain a 2D curve on which the subpixel displacement lies.

By expanding Eq. 6.3 and subtracting $V_i'[x,y]$ to both S_i and $V_i[x,y]$, we obtain

$$C_i(z_i) = \{(\lambda_x,\lambda_y)|b_i\lambda_x + c_i\lambda_y + (d_i - b_i - c_i)\lambda_x\lambda_y = z_i\} \tag{6.4}$$

where

$$\begin{bmatrix} b_i \\ c_i \\ d_i \\ z_i \end{bmatrix} = \begin{bmatrix} V_i'[x+1,y] \\ V_i'[x,y+1] \\ V_i'[x+1,y+1] \\ V_i[x,y] \end{bmatrix} - V_i'[x,y]. \tag{6.5}$$

This curve $C_i(z_i)$ defines a constraint for our subpixel displacement. Since we have two unknowns, intersecting a pair of curves $C_i(z_i)$ and $C_j(z_j)$, $i \neq j$ should be sufficient to solve for λ_x and λ_y.

The intersection is given by the quadratic formula:

$$(\lambda_x,\lambda_y) = \left(\frac{-B_1 - B_2}{2A_1}, \frac{-B_1 + B_2}{2A_2}\right) \pm (\sqrt{\Delta}, \sqrt{\Delta}) \tag{6.6}$$

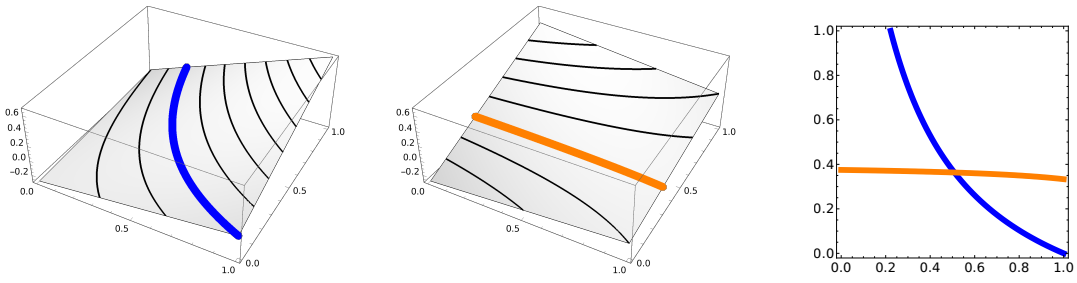

Figure 6.4. Example surfaces S_i (*left*) and S_j (*middle*) with the corresponding curves $C_i(z_i), C_j(z_j)$. The intersection of two curves (*right*) corresponds to the subpixel position and is solved in 2D.

where

$$A_1 = b_j(c_i - d_i) + b_i(-c_j + d_j) \tag{6.7}$$

$$A_2 = c_j(-b_i + d_i) + c_i(b_j - d_j) \tag{6.8}$$

$$B_1 = -b_j c_i + b_i c_j \tag{6.9}$$

$$B_2 = z_i(b_j + c_j - d_j) + z_j(-b_i - c_i + d_i) \tag{6.10}$$

$$\Delta = (B_1 + B_2)^2 - 4A_1(c_i z_j - c_j z_i). \tag{6.11}$$

For any real solution (λ_x, λ_y), if λ_x and λ_y do not lie in the interval $[0,1]$, this solution is the result of extrapolation beyond the boundaries of the surface and should be discarded. In all other cases, Eq. 6.2 directly estimates the quality of this solution and should be a minimum.

Since each pattern provides a single curve and we need to intersect two curves, we can pick any pair $C_i(z_i), C_j(z_j)$ to compute the subpixel position. The total number of available pairs is $N_{patterns}(N_{patterns} - 1)/2$. While a single pair of intersecting curves could be sufficient, some pairs of curves can give degenerate solutions. To avoid those, we can pick $N_{candidates}$ random pairs to compute candidate solutions, and keep the position that minimizes Eq. 6.2 as the final subpixel displacement. Using a larger $N_{candidates}$ increases the quality of the subpixel measurement but takes more time to compute. This tradeoff is shown in Table 6.1.

Note that all this is done locally, without relying on prior knowledge of the patterns. Our method could thus be retroactively applied to refine previously acquired pixel-accurate scans. The only (quite important) limitation is that 2D variations are required across the image sequence. If only 1D subpixel information is required (e.g. with an epipolar calibration), a simpler linear interpolation would be sufficient and our method is not necessary.

6.3.5 Sensitivity to image degradation

While our algorithm expands on a robust matching method [7], it should be noted that the robustness to various degradation factors is not as strong during the subpixel computation phase. The most concerning factor would be global illumination, which should not impact our method. We refine integer matches by looking at neighboring pixels, which tend to receive similar indirect illumination. This illumination should blend in with the object albedo in Eq. 6.1.

However, using the captured intensities instead of binarizing can make the subpixel computation more sensitive to projector defocus, to digital noise and to non-linearity introduced by the camera-projector system. The sensitivity of different methods to blurred patterns and to gamma parameters is compared in Fig. 6.5. The choice of the $N_{candidates}$ parameter in our method controls the tradeoff between quality and speed, with larger values converging towards the quality obtained with a global solution as in [10].

While the method shows no bias in the presence of blur or a gamma non-linearity (data not shown), its precision is reduced. Note that the loss of quality will at worst result in a scan with the same precision as the starting pixel-accurate scan. Solutions for achieving high precision with low-cost structured light systems include longer exposure time to reduce image noise, a gamma-correction strategy and avoiding to scan unreasonably out-of-focus objects.

6.3.6 Bidirectional subpixel

As noted in Section 6.3.2, computing a camera-to-projector subpixel correspondence when the pixel-ratio is unfavorable to the camera requires making an approximation. Interestingly, the unstructured light algorithm [7] can compute correspondences both ways, allowing to reconstruct a scene from the projector's point of view, which is not possible with direct decoding methods.

The subpixel correspondence algorithm described earlier can use either the camera

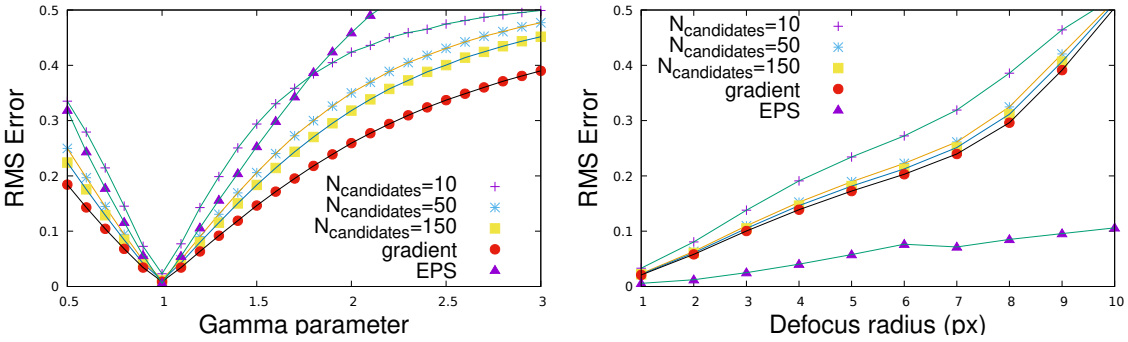

Figure 6.5. Sensitivity to an exponential non-linearity that could be introduced by low-cost cameras or projectors (*left*) and impact of a Gaussian blur that could be introduced by defocus (*right*). RMS error is shown for various choices of $N_{candidates}$ in our method, for a gradient descent optimization [10] and for embedded phase shifting [29] (EPS) using 2 or 3 shifts of 9 frequencies. All methods used pattern periods between $20px$ and $40px$ and a budget of 20 images.

or the projector pixel-accurate match as a starting point and can compute the subpixel for both views. Since a pixel-ratio unfavorable to the camera must be, by definition, favorable to the projector, it is possible in such cases to compute a projector-to-camera subpixel correspondence instead. Computing the subpixel match the other way around would then use a more accurate model for the subpixel interpolation and would lead to a denser reconstruction due to multiple projector pixels being matched at different subpixel locations "inside" a shared camera pixel.

6.4 Depth discontinuities

Computing the subpixel displacement for the camera view can be seen as quantifying the contribution of each projector pixel captured in a single camera pixel. This same principle can be used to accurately detect depth discontinuities. When a camera pixel is observing a discontinuity in a scene, the result is an mixture of projector intensities belonging to unrelated codes.

If the code of a camera pixel best matches the interpolation of discontinuous codes, it is flagged as a discontinuity and either removed, or kept at its original pixel-accuracy. Using Fig. 6.1 as an example again, the code of integral camera pixel 11 would have an optimal "subpixel localization" when matched with interpolated projector codes of pixels 4 and 11, whereas integral camera pixel 4 would have an optimal match with the interpolation of codes for projector pixels 3 and 4. Pixel 11 would be flagged as a discontinuity, but pixel 4 would not.

We should note that discontinuities cannot be strictly interpreted as positions more than $1px$ appart. Adjacent camera pixels could be matched to projector pixels within a distance greater than $1px$ and still lie on a smooth inclined surface. For instance, with a camera-projector pixel-ratio of 1:2, there would likely be gaps of approximately $2px$ between each match of adjacent camera pixels on a planar surface. Furthermore, even at 1:1 pixel-ratios, a surface can be slanted relative to the projector as well as to the camera, which will impact the observed pixel-ratio, as projected images can appear on them as if they were being stretched.

In order to decide if two pixel correspondences are adjacent or not, we rely on the pattern period chosen by the user. Past this distance, the pixels become uncorrelated [10] and aligning at subpixel-level a mixture of uncorrelated pixels according to neighboring projector pixels will not reflect a real-world subpixel displacement. As with any method, users should choose the highest practical pattern frequency for their use case, as higher pattern frequencies lead to more accurate results.

6.5 Experiments

Since discontinuities often form a very small part of reconstructed scenes, we validate our algorithm on synthetic data first and then on real world scenes.

6.5.1 Simulated results

We generated patterns for MicroPS [14], EmbeddedPS [29] and for our method, and introduced artificial grid-aligned discontinuities at every $50px$. We then added a subpixel displacement of $0.15px$ along the x-axis. The patterns were all chosen to contain frequencies high enough to ensure no correlation between neighbors around a discontinuity. Results of the different matching algorithms are shown in Fig. 6.6 using 10 and 100 patterns. Since EmbeddedPS can compute multiple estimates per pixel and aggregate them, it is expected to show a smaller error than MicroPS overall. For EmbeddedPS, the median of all yielded values is used.

It is clear from the plotted error that both MicroPS and EmbeddedPS have systematic errors at discontinuities that cannot be fully accounted for by adding more patterns. Using a small number of patterns, our method shows slightly less precise measurement than EmbeddedPS on smooth regions, but depth discontinuities are correctly handled. The precision can be increased by using more patterns and increasing $N_{candidates}$, as shown in the bottom row.

To measure the limits on the subpixel-accuracy of our method, we tested it on synthetic images with uniformly sampled random subpixel displacements. We compare in Table 6.1 the quality and computation time for a subpixel match over 1M pixels acquired with different choices of $N_{candidates}$, with a gradient descent as in [10] and with the state-of-the-art EmbeddedPS algorithm [29]. We choose not to show the computation time for the latter as the implementation we used was not coded with performance in mind, but it should be noted that since it is a *direct decoding method* using simple arithmetic operations, the full correspondence (pixel + subpixel) is acquired multiple times faster than our subpixel correspondance, at the cost of more reconstruction errors around discontinuities.

We can observe a significant speedup for our method over the gradient descent, at the cost of slightly less precise correspondances. Depending on the application, this loss of precision might not be a problem, as even for a relatively low $N_{candidates}$ of 20, the RMS

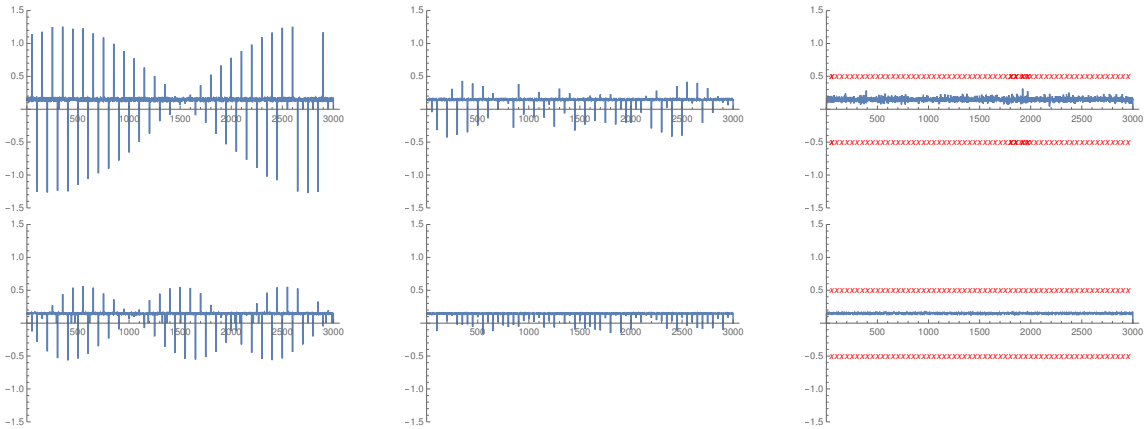

Figure 6.6. Computed subpixel displacement for the simulated experiment (zoom in the electronic copy to see details). Ground truth is $0.15px$ on smooth sections. Data is shown for $N_{patterns} = 10$ (*top row*) and $N_{patterns} = 100$ (*bottom row*) for MicroPS (*left*), EmbeddedPS (*middle*) and our method (*right*). *Red* markers on our method show which points are flagged as discontinuities.

Table 6.1. Time/quality of subpixel matches using $N_{patterns} = 20$

Method	Bias (px)	RMS Error (px)	Time (s)
Ours, $N_{candidates} = 1$	-0.000089	0.177	14
Ours, $N_{candidates} = 10$	0.000001	0.023	25
Ours, $N_{candidates} = 20$	-0.000009	0.016	33
Ours, $N_{candidates} = 50$	-0.000002	0.013	72
Ours, $N_{candidates} = 100$	0.000001	0.011	123
Ours, $N_{candidates} = 150$	-0.000004	0.011	184
Gradient descent [10]	0.005457	0.009	367
EmbeddedPS [29]	-0.000019	0.006	*Fast*

error remains smaller than 0.02 pixel.

6.5.2 Real world scenes

Our experimental setup consisted of a low-cost AAXA HD Pico Projector set at its native resolution of 1280x720 and a Prosilica industrial camera with a resolution of 659x493. Our pixel ratio is favorable to the projector, with around 2 or 3 projector pixels per camera pixel. Even though most structured light systems have the inverse ratio, we chose this setup to highlight the bidirectional nature of our algorithm (Section 6.3.6). In real life scenarios, this ratio could arise from pixel binning, which can be used to increase the signal-to-noise ratio, allowing shorter exposure.

We compared our method to EmbeddedPS [29], using $N_{patterns} = 40$ for both and $N_{candidates} = 100$ for our method. Since our method does not rely on an epipolar calibration and computes 2D correspondances directly, we retrived the x and y directions in two independant EmbeddedPS scans, using 20 patterns per dimension. We used 2 or 3 shifts of 9 frequencies per scan, as this configuration gave us the best robustness to discontinuities (see our supplementary material for details).

To ensure that the subpixel displacement is correct, we present scans of a toy based on Edvard Munch's *The Scream* and of a ceramic egg holder in Fig. 6.7. These results show that different albedos on *The Scream* do not affect our method, and neither do the interreflections caused by the shape of the egg holder. The results are consistent with our synthetic experiments, showing slightly smoother point clouds for EmbeddedPS than for our method on continuous surfaces. As a baseline, the original pixel-accurate scans are compared and we can see that the grid-like artifacts present in them are no longer visible after our subpixel refinement. We also include a reconstruction from the projector view for our method. With our setup, this results in a denser scan that cannot be achieved through direct decoding methods.

To test our discontinuity localization algorithm, we scanned both a simple and a com-

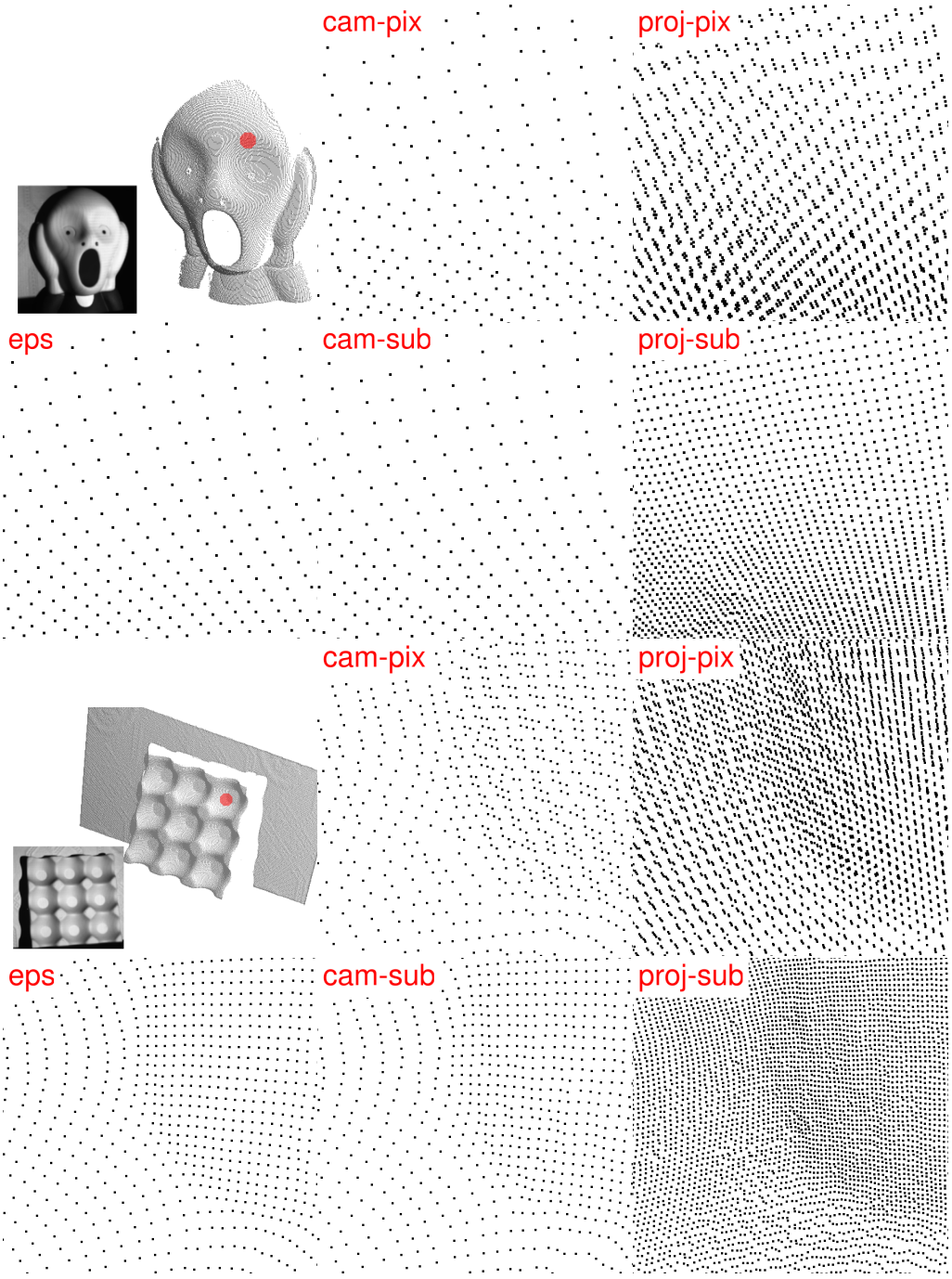

Figure 6.7. Reconstruction of a toy with varying albedos based on the painting *The Scream*, of a ceramic egg holder, prone to interreflections. Reconstructions zoomed in on the red dot are shown for EmbeddedPS (*eps*), for our pixel-accurate input (*pix*) and for our subpixel method (*sub*) for both the camera and projector views (*cam* and *proj*).

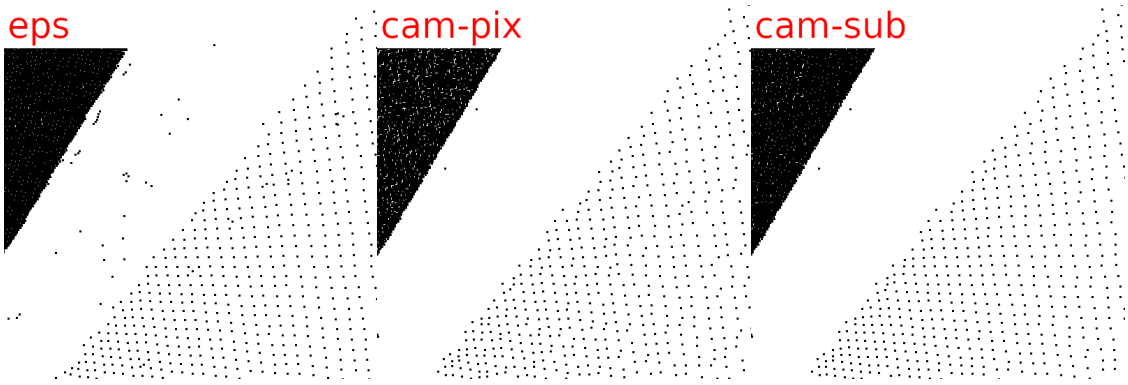

Figure 6.8. Reconstruction of a simple discontinuity at the edge of a box in front of a wall, zoomed in on the discontinuity. EmbeddedPS (*eps*) and our method (*cam-sub*) both show a smooth profile on the surface when compared to the integer-match (*cam-pix*), but EPS show large displacements for points near the discontinuity.

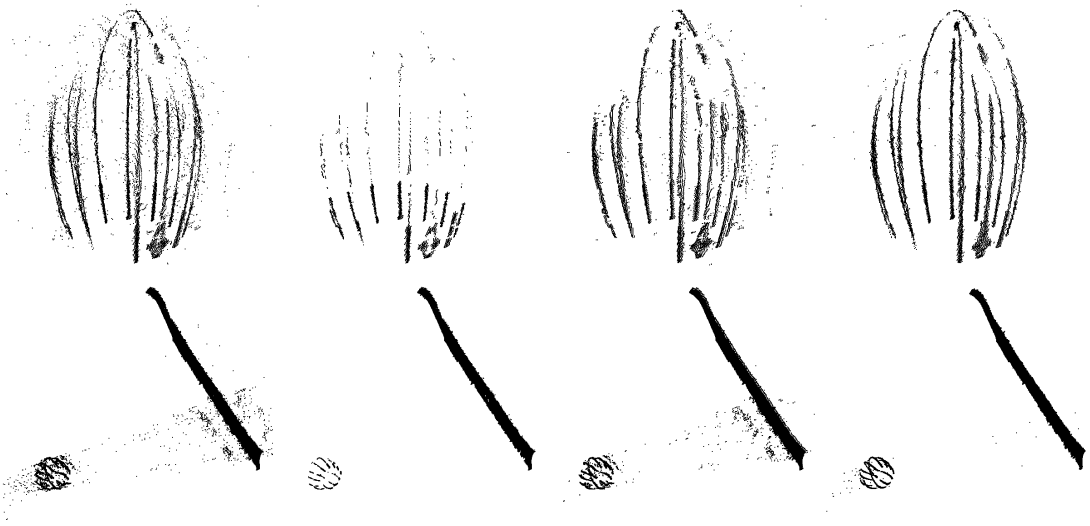

Figure 6.9. Reconstruction of a cooking whip in front of a wall, with a zoom on the whip (*top row*) and an overview of the sceen from the top (*bottom row*). *From left to right*, raw point cloud reconstructed with EPS, EPS after filtering out every potential discontinuity, EPS after applying a 3x3 median filter on the matches, and our reconstruction.

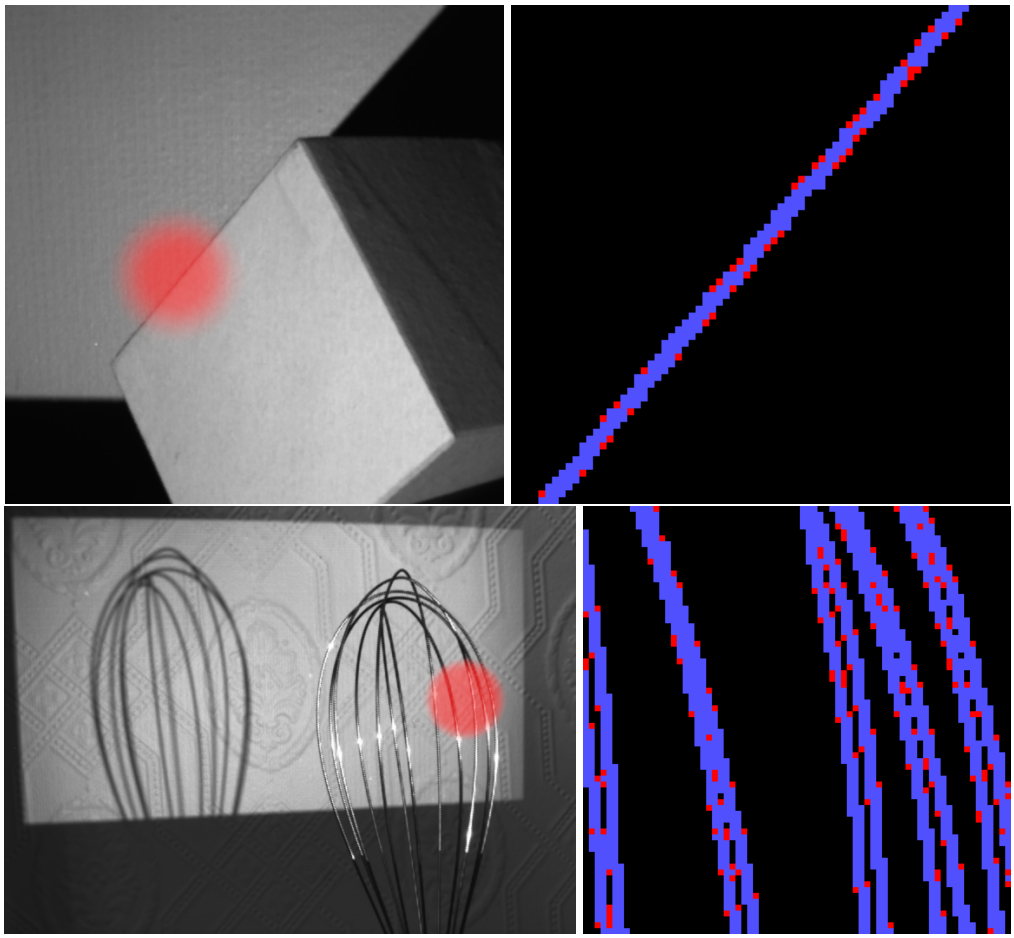

Figure 6.10. Camera point of view for the box and the cooking whip used in experiments (*left*) and map of the pixels that would be falsely flagged as a discontinuity when using a naive discontinuity detection method (*right*). Pixels in red do not observe a discontinuity but are adjacent to disconnected regions while pixels in blue observe a mix of projected pixels.

plex object in front of a wall. The first was the flat side of a box (Fig. 6.8), the second was a metallic cooking whip (Fig. 6.9). Difficulty in this last scene arises from the specular nature of the material, as well as from the fact that each metallic wire were $\approx 5px$ of width in the camera image, including the discontinuous pixels. The point clouds for both scenes highlight the tendency of phase shifting methods to interpolate between unrelated points at discontinuities. To show that errors on complex scenes cannot be trivially corrected, we also show the whip reconstructed using the naive strategy consisting in removing every pixel with a non-neighboring match, which removes all outliers but removes most of the whip too. We also show the effect of a median filter of size 3x3 on the correspondence map, which removes only a small part of the outliers. Larger filter sizes were used, none of which corrected the presence of outliers, while having a more damaging effect on the reconstructed whip (results not shown).

Other points of view of the scenes are available in our supplementary material, along with our source code.

6.6 Conclusion

In this paper, we presented a subpixel computation algorithm which accurately identifies discontinuities in captured scenes. This method allows for precise reconstructions of difficult scenes and is faster than previous robust subpixel approaches. While it has not been tested on unsynchronized equipment, it could easily be integrated in the algorithm described in [9] as it uses the same cost function. Although our tests used the robust unstructured light patterns [7], our cost function and imaging model are not pattern dependent, with the caveat of using 2D-varying patterns. As future work, one could imagine designing a 2D variant of the optimal patterns generated in [27] and using the method presented here to compute subpixel matches while remaining accurate around discontinuities.

Chapitre 7

TRIANGULATION ET MAILLAGE

Une fois la correspondance entre les points projetés et capturés obtenue, on peut l'utiliser pour retrouver les points 3D qui forment la scène.

7.1 Triangulation

En deux dimensions, la triangulation est assez directe : étant donné un objet observé depuis deux endroits différents séparés d'une distance connue, il suffit de trouver les angles formés dans le triangle qui relie les deux points de vue et l'objet pour déduire les distances inconnues. Cette idée est illustrée au chapitre 2, dans la Figure 1.3.

On pourrait tenter d'appliquer cette méthode à partir de l'information acquise jusqu'ici. Soient un point $\mathbf{p}_c$ dans la caméra et sa correspondance $\mathbf{p}_p$ dans le projecteur, soient les matrices P_c et P_p, respectivement les matrices de transformation de la caméra et du projecteur et soit $\mathbf{x}_w$ le point du monde à retrouver par triangulation.

Les points $\mathbf{p}_c$ et $\mathbf{p}_p$ sont exprimés en 3D, dans les systèmes de coordonnées respectifs de la caméra et du projecteur. Les deux premières dimensions correspondent à des positions (x,y) exprimées en pixels dans l'image, la troisième est $z = 1$, qui correspond à la distance du centre de la caméra ou du projecteur au plan image virtuel. On peut facilement réexprimer ces points dans le système de coordonnées du monde, en les passant en coordonnées projectives et en utilisant les inverses des matrices de transformation :

$$\mathbf{x}_c = P_c^{-1} \, (\mathbf{p}_c \; 1)^T \tag{7.1}$$

$$\mathbf{x}_p = P_p^{-1} \, (\mathbf{p}_p \; 1)^T \, . \tag{7.2}$$

De façon similaire, on peut exprimer les centres de la caméra et du projecteur dans le

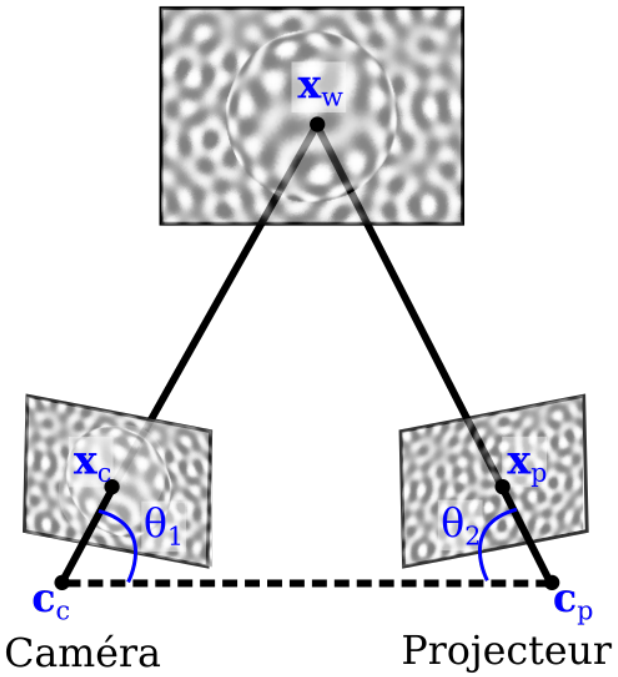

Figure 7.1. Triangle formé en 3D par les points $\mathbf{x}_c, \mathbf{x}_p, \mathbf{c}_c, \mathbf{c}_p$ et $\mathbf{x}_w$. En retrouvant les angles θ_1 et θ_2, on peut calculer la position du point $\mathbf{x}_w$.

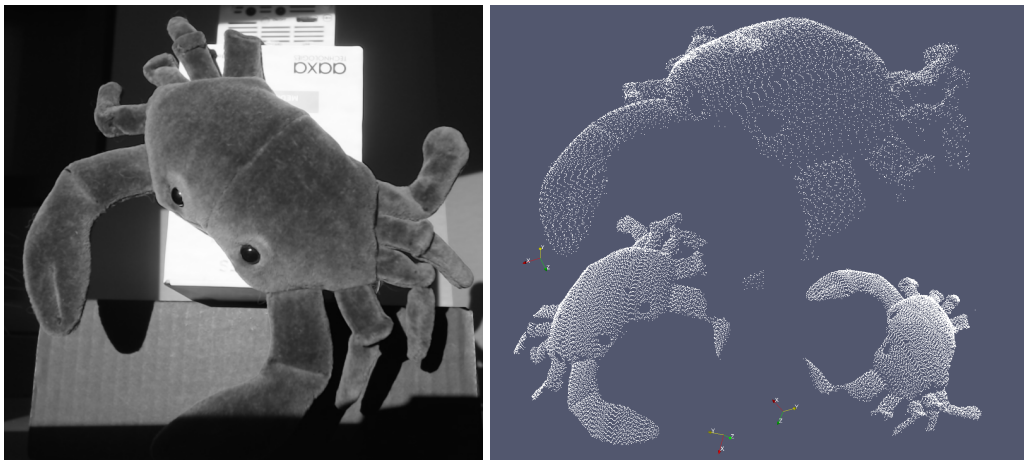

Figure 7.2. Nuage de points reconstruit pour un scan d'un toutou de crabe, vu sous différents angles. La triangulation des points utilise la *direct linear transformation* implantée dans OpenCV.

monde avec :

$$\mathbf{c}_c = P_c^{-1}\,(0\ 0\ 0\ 1)^T \tag{7.3}$$

$$\mathbf{c}_p = P_p^{-1}\,(0\ 0\ 0\ 1)^T\,. \tag{7.4}$$

Les points $\mathbf{x}_c, \mathbf{x}_p, \mathbf{c}_c, \mathbf{c}_p, \mathbf{x}_w$ sont tous exprimés en coordonnées projectives dans le système du monde et se situent sur un même triangle dans un même plan (illustré à la Figure 7.1). En tenant compte de cette information, on peut déduire la valeur de $\mathbf{x}_w$ à partir des autres, qui sont connus :

$$\theta_1 = arccos\left(\frac{(\mathbf{x}_c - \mathbf{c}_c)\cdot(\mathbf{c}_p - \mathbf{c}_c)}{||(\mathbf{x}_c - \mathbf{c}_c)||||(\mathbf{c}_p - \mathbf{c}_c)||}\right) \tag{7.5}$$

$$\theta_2 = arccos\left(\frac{(\mathbf{x}_p - \mathbf{c}_p)\cdot(\mathbf{c}_c - \mathbf{c}_p)}{||(\mathbf{x}_p - \mathbf{c}_p)||||(\mathbf{c}_c - \mathbf{c}_p)||}\right) \tag{7.6}$$

$$b = ||\mathbf{c}_p - \mathbf{c}_c|| \tag{7.7}$$

$$||\mathbf{x}_w - \mathbf{c}_p|| = \frac{sin\theta_1}{b} \tag{7.8}$$

$$||\mathbf{x}_w - \mathbf{c}_c|| = \frac{sin\theta_2}{b} \tag{7.9}$$

$$\mathbf{x}_w = \mathbf{c}_p + \frac{sin\theta_1}{b}(\mathbf{x}_p - \mathbf{c}_p) = \mathbf{c}_c + \frac{sin\theta_2}{b}(\mathbf{x}_c - \mathbf{c}_c). \tag{7.10}$$

Très simple... À condition d'avoir des points parfaits. L'hypothèse de base ici est que les points en correspondances dans les plans images sont directement issus d'un même point 3D du monde. Avec des mesures imparfaites, par exemple, au pixel près, il est plutôt probable que les droites $\mathbf{x}_c - \mathbf{c}_c$ et $\mathbf{x}_p - \mathbf{c}_p$ *ne se croisent pas en 3D*. Le plan formé par les trois points $\mathbf{c}_c, \mathbf{c}_p$ et $\mathbf{x}_w$ n'est alors pas celui qui contient les points $\mathbf{x}_c, \mathbf{x}_p$ retrouvés.

La bonne façon de trianguler de points en 3D est plutôt de considérer le problème comme une minimisation : on cherche le point 3D $\widehat{\mathbf{x}}_w$ qui est le plus cohérent possible avec les points $\mathbf{p}_c$ et $\mathbf{p}_p$. La formulation "le plus cohérent possible" est délibérément vague ici, puisque plusieurs critères différents peuvent être utilisés.

On pourrait par exemple tenter de minimiser la distance euclidienne de $\widehat{\mathbf{x}}_w$ aux droites de déprojection des points dans le monde, $(\mathbf{x}_c - \mathbf{c}_c)$ et $(\mathbf{x}_p - \mathbf{c}_p)$.

Considérons les droites exprimées comme étant la position des points $\mathbf{p}_c$ et $\mathbf{p}_p$, reprojetés sur des plans images à des distances respectives λ_c et λ_p des centres de projection, le tout exprimé en coordonnées projectives du monde :

$$l_c(\lambda_c) = P_c^{-1} \, (\lambda_c \mathbf{p}_c \;\; 1)^T \tag{7.11}$$

$$l_p(\lambda_p) = P_p^{-1} \, (\lambda_p \mathbf{p}_p \;\; 1)^T \, . \tag{7.12}$$

On cherche alors :

$$\underset{\widehat{\mathbf{x}}_w}{argmin} \left(||l_c(\lambda_c) - \widehat{\mathbf{x}}_w||^2 + ||l_p(\lambda_p) - \widehat{\mathbf{x}}_w||^2 \right) \tag{7.13}$$

Ce point correspond au point milieu de la perpendiculaire commune aux deux droites dans un système euclidien. Cette formulation est utilisée tôt dans la littérature [31], mais ne donne pas des résultats particulièrement bons.

Une autre approche est de plutôt considérer *l'erreur de reprojection 2D* : pour des points $\mathbf{p}_c$ et $\mathbf{p}_p$, on cherche :

$$\underset{\widehat{\mathbf{x}}_w}{argmin} \left(||P_c \hat{\mathbf{x}}_w - \mathbf{p}_c||^2 + ||P_p \hat{\mathbf{x}}_w - \mathbf{p}_p||^2 \right) \tag{7.14}$$

où $||...||$ est la distance euclidienne dans le plan image. Il s'agit ici non pas de minimiser une distance euclidienne en 3D, mais plutôt une distance euclidienne *dans les plans images*, donc exprimée en pixels.

Plusieurs méthodes permettant de résoudre cette formulation sont décrites dans [19]. La méthode la plus commune est la *direct linear transformation*, qui est l'algorithme derrière la fonction *cv::sfm::triangulatePoints* d'OpenCV.

Assumant que les matrices de transformation P_c et P_p soient connues de façon exacte, on a la contrainte supplémentaire que $\hat{\mathbf{x}}_w$ doit être projeté sur des lignes épipolaires correspondantes dans les deux plans images (ce qui n'est pas forcément précisément le cas lorsqu'on considère les points originaux $\mathbf{p}_c$ et $\mathbf{p}_p$ mesurés). Une méthode optimale prenant en considération cette contrainte supplémentaire est proposée par Hartley et Sturm dans [19].

7.2 Maillage

La triangulation donne un nuage de points, ce qui n'est pas tout à fait utilisable en pratique. Un nuage de points triangulé ne contient pas d'information sur la connexité des points. Pour la plupart des applications, obtenir la *surface* du modèle 3D est nécessaire.

Dans le cas d'un scan 3D par lumière structurée, la grille régulière de pixels peut être utilisée pour inférer la connexité : des points générés à partir de pixels voisins devraient habituellement être connectés, et donc reliés par une arête. La seule exception est dans le cas des discontinuités de profondeur, que l'on peut trouver en calculant une carte de discontinuités (par exemple, avec l'algorithme proposé au chapitre 6) et l'utiliser pour éviter de relier entre eux des points pour des pixels qui ne correspondent pas à des points réellement adjacents dans la scène.

Cette méthode présente cependant certains désavantages : premièrement, elle est limitée à la reconstruction de surfaces *partielles*, puisque le maillage se fait avec la connexité récupérée depuis un seul point de vue. Si on souhaitait considérer un nuage de points construit à partir de l'alignement de plusieurs scans d'un même objet sous différents angles, il faudrait une méthode un peu plus avancée pour obtenir un maillage consistant pour l'ensemble des points. Deuxièmement, les points sont utilisés directement pour créer le modèle. Si les points comportent du bruit, ce bruit est directement présent sur les sommets à la surface du modèle.

D'autres algorithmes plus généraux pour le maillage de nuages de points évitent ces problèmes, par exemple, la *reconstruction de surface de Poisson* [23], qui propose plutôt d'utiliser un nuage de points orientés en tant que guide pour inférer une surface implicite lissée.

L'idée est la suivante : une surface implicite peut être triangulée avec un algorithme comme *Marching tetrahedra* [8] à partir d'une fonction indicatrice (1 si on se trouve à l'intérieur de la surface, 0 si on se trouve à l'extérieur). Cette fonction indicatrice n'est pas connue, mais le nuage de points orienté correspond à peu près au champ vectoriel des

gradients de cette fonction.

Étant donné un champ vectoriel $\vec{V}$ en 3D, on cherche la fonction indicatrice χ dont le gradient est $\vec{V}$:

$$\chi \text{ t.q. } \nabla\chi = \vec{V}. \tag{7.15}$$

Cette fonction n'existe pas forcément, mais une approximation peut être retrouvée :

$$\widehat{\chi} = \underset{\chi}{argmin}\left(||\nabla\chi - \vec{V}||\right). \tag{7.16}$$

En appliquant l'opérateur de divergence (∇) aux termes, on obtient une jolie formulation du problème en équation de Poisson :

$$\nabla \cdot \nabla\chi = \nabla \cdot \vec{V} \tag{7.17}$$

$$\Longrightarrow \Delta\chi = \nabla\vec{V}. \tag{7.18}$$

On cherche χ, tel que son laplacien est égal à la divergence du champ vectoriel $\vec{V}$, ce qui peut être résolu aux moindres carrés avec un système linéaire épars et bien conditionné, en utilisant des algorithmes de résolution généraux (par exemple, la méthode du gradient conjugué). La surface implicite est ainsi calculée de façon globale, ce qui permet de trouver une solution minimisant le bruit pour tout l'ensemble de points en même temps.

À noter : un scanneur 3D à lumière structurée ne donne qu'un nuage de points *non orientés*. On peut cependant calculer des normales très approximatives en utilisant la structure du nuage de points, ce qui fonctionne tout de même, puisque l'algorithme est robuste au bruit dans les normales. Cet algorithme est implanté dans le logiciel MeshLab et dans la librairie de calcul géométrique CGAL.

Chapitre 8

CONCLUSION

Ce mémoire se veut une introduction au domaine du scan 3D par la lumière structurée. Il présente des introductions aux composantes de la vision par ordinateur qui sont essentielles à la mise en oeuvre d'un scanneur 3D à lumière structurée, ainsi qu'un résumé plus détaillé des différentes stratégies possibles et des différents défis associés à chacune.

En guise de contribution principale, ce mémoire propose une méthode permettant le calcul efficace du raffinement sous-pixellique d'une correspondance en lumière non-structurée, tout en considérant correctement les effets dus aux discontinuités de profondeurs. Une implantation de cet algorithme est disponible sous licence MIT et est accessible sur Internet [22], en espérant qu'elle puisse servir à faciliter l'accès à des scans plus robustes et plus précis.

Suite aux travaux réalisés pendant ce mémoire, plusieurs voies d'exploration restent ouvertes. L'augmentation de la précision des scans, ainsi que la diminution du temps total requis pour compléter un scan restent les deux plus grandes avenues de recherche.

La précision est probablement l'élément le plus important dans les applications industrielles. Beaucoup d'applications en contrôle de qualité nécessitent des scans à très haute résolution, et plusieurs facteurs jouent en même temps sur la qualité des reconstructions dans une scène donnée. Un des plus gros facteurs est au niveau du choix de la fréquence spatiale des images projetées. Autant une fréquence spatiale trop élevée que trop basse dégradent les résultats d'un scan, et aucun article n'a pour le moment étudié la façon dont les différentes parties de la scène scannée affectent la fréquence optimale à utiliser. La question des surfaces inclinées, qui modifient le ratio de pixels perçu entre la caméra et le projecteur, a été abordée dans la section 6.4, mais mériterait plus d'attention. Puisqu'une surface inclinée "condense" plusieurs pixels projetés en peu de pixels cap-

turés, il serait particulièrement intéressant de considérer la possibilité d'utiliser différentes fréquences pour différentes parties d'une même scène en fonction de la densité de pixels capturés en un point. Cette idée pourrait être réalisée soit en augmentant le nombre total d'images projetées, soit en ajustant la structure même des images générées pour varier les fréquences spatiales selon la densité de pixels. Permettre un choix de fréquence optimal pour chaque point de la scène plutôt que de choisir une unique fréquence qui fonctionne bien en moyenne seulement aurait le potentiel d'améliorer la qualité de certaines scènes traditionnellement plus difficiles.

Au niveau de l'augmentation de la vitesse des scans, il serait important de valider expérimentalement l'applicabilité de l'algorithme de raffinement sous-pixellique 6 lors d'un scan désynchronisé tel que décrit dans [9]. Ce dernier algorithme propose une accélération majeure de la capture, au détriment du temps de calcul lors de la correspondance. En accélérant à la fois la phase initiale de capture et la phase finale de raffinement sous-pixel, un scanner qui combinerait ces deux algorithmes pourrait donner des résultats précis dans un temps réduit par rapport aux algorithmes de l'état de l'art actuel.

RÉFÉRENCES

[1] aaxa technologies. https://www.aaxatech.com/. [En ligne; accès le 9 septembre 2020].

[2] Webcam Logitech C270 HD. https://www.logitech.com/en-ca/product/hd-webcam-c270. [En ligne; accès le 9 septembre 2020].

[3] Build A Smartphone Projector! (Using Shoebox). https://www.youtube.com/watch?v=Tx4vPeL9y2g, Jul 2015. [En ligne; accès le 9 septembre 2020].

[4] Alexandr Andoni et Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, Janvier 2008.

[5] Connelly Barnes, Eli Shechtman, Adam Finkelstein, et Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Août 2009.

[6] Tongbo Chen, Hans-Peter Seidel, et Hendrik Lensch. Modulated phase-shifting for 3d scanning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008), IEEE Computer Society, 1-8 (2008)*, 06 2008.

[7] Vincent Couture, Nicolas Martin, et Sébastien Roy. Unstructured light scanning robust to indirect illumination and depth discontinuities. *International Journal of Computer Vision*, 108(3):204–221, Juillet 2014.

[8] A. Doi et A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. Dans *IEICE Transactions on Information and Systems*, pages 214–224, 1991.

[9] Chaima El Asmi et Sebastien Roy. Fast unsynchronized unstructured light. *15th Conference on Computer and Robot Vision (CRV)*, pages 277–284, 05 2018.

[10] Chaima El Asmi et Sébastien Roy. Subpixel unsynchronized unstructured light. *14th International Conference on Computer Vision Theory and Applications*, pages 865–875, 01 2019.

[11] Matteo Carocci Giovanna Sansoni et Roberto Rodella. Three-dimensional vision based on a combination of gray-code and phase-shift light projection: analysis and compensation of the systematic errors. volume 38, pages 6565–6573. Appl. Opt., Nov 1999.

[12] J. Gu, T. Kobayashi, M. Gupta, et S.K. Nayar. Multiplexed Illumination for Scene Recovery in the Presence of Global Illumination. Dans *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, Nov 2011.

[13] Jens Guehring. Dense 3-d surface acquisition by structured light using off-the-shelf components. *Proceedings of SPIE - The International Society for Optical Engineering*, 4309, 01 2001.

[14] M. Gupta et S.K. Nayar. Micro Phase Shifting. Dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Jun 2012.

[15] M. Gupta, Q. Yin, et S.K. Nayar. Structured Light in Sunlight. Dans *IEEE International Conference on Computer Vision (ICCV)*, Dec 2013.

[16] Mohit Gupta, Amit Agrawal, Ashok Veeraraghavan, et Srinivasa Narasimhan. Structured light 3d scanning in the presence of global illumination. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 713 – 720, 07 2011.

[17] Mohit Gupta et Nikhil Nakhate. A geometric perspective on structured light coding. Dans *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Richard Hartley et Andrew Zisserman. *Multiple View Geometry In Computer Vision, Second Edition*. Cambridge University Press, 2005.

[19] Richard I. Hartley et Peter Sturm. Triangulation. volume 68, pages 146 – 157. Computer Vision and Image Understanding, 1997.

[20] Jonathan Mark Huntley et Henrik O. Saldner. Shape measurement by temporal phase unwrapping and spatial light modulator-based fringe projector. Dans Otmar Loffeld, editeur, *Sensors, Sensor Systems, and Sensor Data Processing*, volume 3100, pages 185 – 192. International Society for Optics and Photonics, SPIE, 1997.

[21] Nicolas Hurtubise et Sébastien Roy. Fast discontinuity-aware subpixel correspondence in structured light. Dans *2020 International Virtual Conference on 3D Vision (3DV)*, pages 1108–1116, November 2020.

[22] Nicolas Hurtubise et Sébastien Roy. Fast discontinuity-aware subpixel correspondence in structured light (code source). `https://gitlab.com/316k/fast-discontinuity-aware-subpixel-correspondence-in-structured-light`, 2020. [En ligne; accès le 24 novembre 2020].

[23] Michael Kazhdan, Matthew Bolitho, et Hugues Hoppe. Poisson surface reconstruction. Dans *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association.

[24] A. Kushnir et N. Kiryati. Shape from unstructured light. Dans *2007 3DTV Conference*, pages 1–4, May 2007.

[25] Nicolas Martin, Vincent Chapdelaine-Couture, et Sébastien Roy. Subpixel scanning invariant to indirect lighting using quadratic code length. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1441–1448, 12 2013.

[26] Parsa Mirdehghan, Wenzheng Chen, et Kiriakos N. Kutulakos. Optimal structured light à la carte: Supplemental document.

[27] Parsa Mirdehghan, Wenzheng Chen, et Kiriakos N. Kutulakos. Optimal structured light à la carte. Dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] Daniel Moreno, Fatih Calakli, et Gabriel Taubin. Unsynchronized structured light. volume 34, New York, NY, USA, Octobre 2015. ACM Trans. Graph.

[29] Daniel Moreno, Kilho Son, et Gabriel Taubin. Embedded phase shifting: Robust phase shifting with embedded signals. Dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[30] S.K. Nayar, G. Krishnan, M. D. Grossberg, et R. Raskar. Fast Separation of Direct and Global Components of a Scene using High Frequency Illumination. *ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH)*, Jul 2006.

[31] A Zisserman P A Beardsley et D W Murray. Navigation using affine structure from motion. Dans *Proc 3rd European Conf on Computer Vision, Stockholm*, Lecture Notes in Computer Science, pages 85–96. Springer, 1994.

[32] B. Park, Y. Keh, D. Lee, Y. Kim, S. Kim, K. Sung, J. Lee, D. Jang, et Y. Yoon. Outdoor operation of structured light in mobile phone. Dans *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2392–2398, 2017.

[33] Jiyoung Park, Cheolhwon Kim, Jaekeun Na, Juneho Yi, et Matthew Turk. Using structured light for efficient depth edge detection. *Image and Vision Computing*, 26(11):1449 – 1465, 2008.

[34] J.L Posdamer et M.D Altschuler. Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1 – 17, 1982.

[35] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, et Xavier Llado. A state of the art in structured light patterns for surface profilometry. *Pattern Recognition*, 43:2666–2680, 08 2010.

[36] Joaquim Salvi, Jordi Pagès, et Joan Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, apr 2004.

[37] D. Scharstein, H. Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, X. Wang, et P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. Dans *GCPR*, 2014.

[38] Daniel Scharstein, Richard Szeliski, et Heiko Hirschmüller. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. `https://vision.middlebury.edu/stereo/`, 2020. [En ligne; accès le 9 septembre 2020].

[39] Daniel Scharstein et Rick Szeliski. High-accuracy stereo depth maps using structured light. Dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pages 195–202. IEEE Computer Society, June 2003.

[40] Z. Song, R. Chung, et X. Zhang. An accurate and robust strip-edge-based structured light means for shiny surface micromeasurement in 3-d. *IEEE Transactions on Industrial Electronics*, 60(3):1023–1032, March 2013.

[41] V. Srinivasan, H. C. Liu, et Maurice Halioua. Automated phase-measuring profilometry: a phase mapping approach. *Appl. Opt.*, 24(2):185–188, Jan 1985.

[42] J.-P Tardif, S. Roy, et M. Trudeau. Multi-projectors for arbitrary surfaces without explicit calibration nor reconstruction. Dans *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 217– 224, 11 2003.

[43] Yoshihiro Watanabe, Gaku Narita, Sho Tatsuno, Takeshi Yuasa, Kiwamu Sumino, et Masatoshi Ishikawa. High-speed 8-bit image projector at 1,000 fps with 3 ms delay. 12 2015.

[44] Shuntaro Yamazaki, Masaaki Mochimaru, et Takeo Kanade. Simultaneous self-calibration of a projector and a camera using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 60 – 67, 07 2011.

[45] Song Zhang. Active versus passive projector nonlinear gamma compensation method for high-quality fringe pattern generation. Dans *Proceedings of SPIE*, volume 9110, page 911002. The International Society for Optical Engineering, 05 2014.

[46] Yu Zhang, Daniel L. Lau, et Ying Yu. Causes and corrections for bimodal multi-path scanning with structured light. Dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[47] Zhengyou Zhang. A flexible new technique for camera calibration. Dans *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1330 – 1334, 12 2000.