ÉCOLE DES PONTS PARISTECH

PROJET D'INITIATION À LA RECHERCHE 2017

COURS D'OUVERTURE 1A :
Imagerie numérique – Application en mécanique des matériaux

Mathieu Aubry (LIGM-Imagine) – Michel Bornert (Navier)

Mesure de forme et de déformation par imagerie numérique

Reconstitution 3D de la surface d'un objet à partir d'images 2D

Stéphane Allado – Alexandre Pacaud – Caroline Pascal – Diane Pugès – Pierre Uginet

1er Juin 2017

ABSTRACT : Our project aims to developp a process allowing to visualize the exterior surface of a tri-dimensionnal object in a 3D view, using as input a sequence of 2D pictures of this object. This process is structured around two main thematics : over a first phase, the research in each pictures of matching points between different interest features spotted on the real object, and, over a second phase, the triangulation of the previously spotted points in the 3-dimensional space, taking in account the cameras properties and calibration, and considering an orthographic projection model. This computer stereo vision process, aiming to model the shape and deformations of objects whose size is close to the millimeter, requires a high accuracy and robustness : therefore, we led a quantitavive scaling of these two factors, noise transmission and disparity between the reconstruction and the real object, on our own implementation of this process.

KEY-WORDS : Digital image, feature detector, descriptor, matching, SIFT, kd-tree, computer stereo vision, triangulation, structure from motion, orthographic projection, 3D reconstruction from multiple images, RANSAC, orthographic projection

Table des matières

Chapitre 1

Introduction

Il existe de nombreuses méthodes de visualisation en trois dimensions d'un échantillon de matériau, comme par exemple la tomographie à rayon X, ou la visualisation par microscope à balayage électronique. Ces techniques utilisent le plus souvent l'interaction entre un rayonnement et l'échantillon étudié. Elles permettent, après reconstitution, de visualiser en trois dimensions la structure interne de l'échantillon. Ces méthodes présentent de nombreux avantages comme être non destructives par exemple. Elles nécessitent en revanche une quantité colossale de données et de nombreuses heures de traitement avant de donner un résultat utilisable. De plus, la visualisation de la surface des échantillons peut se révéler être nécessaire à la compréhension des déformations et des contraintes, qu'elles soient locales ou globales. Des phénomènes de glissements inter-granulaires, de cisaillement ou encore de mouvements hors plan d'une surface ne sont pas révélés précisément avec des méthodes telles que la tomographie à rayon X. Ces informations sont pourtant très importantes pour caractériser l'échantillon, notamment pour des matériaux non homogènes. L'étude des déformations par imagerie numérique permet de visualiser ces déformations de surface et donne une bonne idée des propriétés mécaniques du matériau. Dans le cadre d'une thèse pourtant sur l'étude des glissements aux joints de grains dans l'aluminium à haute température, un suivi des déformations en surface par imagerie numérique a été prôné par les chercheurs. Pour ce faire, une grille formée d'un maillage régulier a été imprimée sur les échantillons d'aluminium (1.1), et l'analyse du déplacement relatif des noeuds permet l'étude des déformations de l'échantillon dans le plan de l'image. L'objectif de notre projet est donc apparu comme la reconstitution en trois dimensions de la surface de ces échantillons, photographiés sous plusieurs angles, afin de permettre l'étude de leurs déformations tri-dimensionnelles en surface.

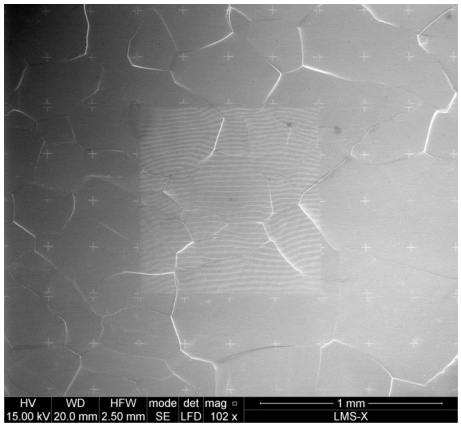


FIGURE 1.1 – *Visualisation de la surface d'un échantillon*

Afin d'éprouver nos reconstitutions 3D sur des données réelles, une série de photographies d'un échantillon d'aluminium déformé et une liste de correspondances de coordonnées ont été mises à notre disposition.

Chapitre 2

Recherche de correspondances

Pour pouvoir reconstituer un objet en 3D à partir de différentes prises de vue en 2D, il est nécessaire d'identifier des points-clés que l'on va retrouver sur chaque image : on parle de correspondances. Cette recherche de correspondances se déroule en trois étapes :

1. Recherche de points-clés
2. Calcul des descripteurs
3. Identification des correspondances

Pour ce projet, nous avons choisi d'utiliser principalement un algorithme de détection de correspondances nommé SIFT (*Scale-Invariant Feature Transform*)

2.1 Recherche de points-clés

Pour identifier les zones identiques sur deux prises de vue différentes d'un même objet, on travaille sur des points-clés (en nombre fini) plutôt que d'effectuer des calculs sur tous les points de l'image, ce qui serait trop long et complexe à implémenter et n'améliorerait pas le résultat final.

Pour ce projet, nous avons choisi de travailler principalement avec un algorithme de détection de correspondances nommé SIFT. Dans cet algorithme, les points-clés sont définis par leurs coordonnées (*x,y*) dans l'image ainsi que par leur facteur d'échelle σ, qui correspond au rayon de la zone circulaire (*patch*) que l'on va considérer autour de ce point. On travaille dans *l'espace des échelles*, espace discret qui correspond à la donnée de ces trois coordonnées.

La recherche de points clés est cruciale au bon calcul des correspondances : en effet, il est nécessaire qu'ils soient choisis de manière à être identifiables le mieux possible sur les différentes prises de vue, c'est-à-dire qu'ils soient relativement invariants aux changements d'échelles, rotations, transformations affines... Ils doivent être suffisamment nombreux pour permettre une bonne triangulation de l'image et donc une reconstitution précise, mais un trop grand nombre conduirait à une perte de robustesse. Comment sont-ils choisis ?

Le but étant de choisir les patchs les plus facilement et certainement reconnaissables entre les images, on va sélectionner les patchs avec des gradients élevés, qui correspondent à de forts changements de contrastes.

On peut calculer les gradients de façon classique, en calculant simplement pour chaque point du patch la différence avec ses voisins immédiats.

La méthode SIFT, quant à elle, travaille dans l'espace des échelles et utilise le *gradient de facteur d'échelle*, qui est la convolution de l'image étudiée I avec un filtre gaussien G de paramètre σ :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Ceci a pour but de "lisser" l'image de façon à supprimer les détails trop petits, c'est-à-dire de rayon inférieur à σ. On étudie ensuite l'image appelée *différence de gaussiennes*, définie de la manière suivante :

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

avec k un paramètre de l'algorithme caractérisant la finesse de la discrétisation voulue. Les extréma locaux de cette image seront les points-clés retenus pour la suite.

On observe en général que les patchs ayant des gradients élevés dans plusieurs directions sont les plus faciles à identifier, tandis qu'il existe un problème avec les lignes simples qui posent un *problème d'ouverture* : il est difficile de les identifier car plusieurs points des images ont les mêmes caractéristiques.

Voici un exemple simple de recherche (à la main) de points-clés sur une photo que nous avons utilisé tout au long de cette partie pour tester notre programme :

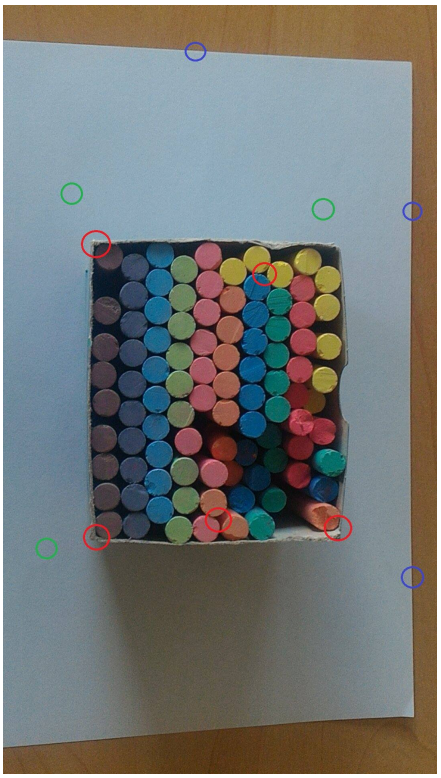


FIGURE 2.1 – *Exemple de différents points-clés potentiels*

De façon intuitive, les patchs entourés en vert sont très difficiles à identifier : en effet, il ne présentent aucun contraste caractéristique et n'importe quel point de la feuille pourrait y correspondre : ces points ne seront pas retenus pour la suite de l'algorithme.
Ceux entourés en bleu présentent plus de contrastes (i.e de différences de gradient) et sont plus facilement identifiables, cependant il peut encore y avoir des confusions entre certains points : il s'agit ici du problème du fort gradient, mais dans une seule direction qui ne permet pas toujours une description efficace.
En revanche, les points entourés en rouge présentent de forts gradients dans plusieurs directions : ils sont facilement identifiables et présentent peu de risques d'être confondus avec d'autres patchs. Il s'agit de ce type de points-clés que l'on va chercher à privilégier pour notre programme.

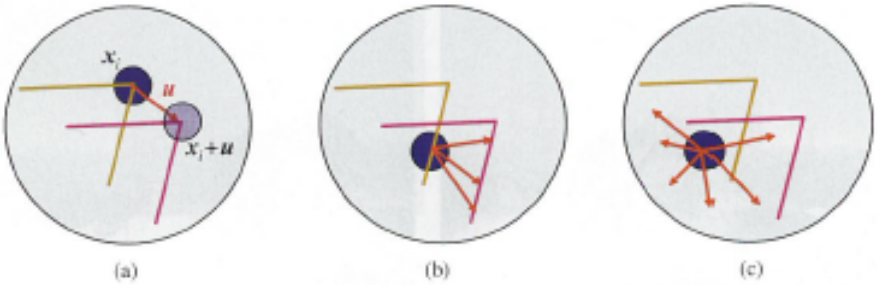


FIGURE 2.2 – *Problèmes d'ouverture pour différents patchs : (a) Situation "en coin", stable ; (b) Situation "en ligne", problème d'ouverture ; (c) Région "sans texture".*
Les deux images sont superposées ; le vecteur rouge u indique le déplacement entre les vecteurs des patchs. [9]

2.2 Calcul des descripteurs

Une fois les points-clés sélectionnés, il faut pour chacun calculer un *descripteur*, qui va caractériser numériquement le point. Le principe sur lequel repose l'algorithme est que le descripteur d'un même point varie le moins possible d'une prise de vue à une autre. Comme on l'a vu précédemment, les points-clés sélectionnés le sont de façon à optimiser l'invariance des descripteurs.
Il existe de très nombreux types de descripteurs, parmi lesquels :

Le descripteur MOPS (*Multi-Scale Oriented Matches*)

Calculé dans l'espace des échelles, ce descripteur est calculé simplement en normalisant les intensités du patch, à une fréquence d'échantillonnage basse pour réduire la prise en compte du bruit dans la position du point d'intérêt. Il est raisonnablement performant pour des tâches simples qui ne requièrent pas une prise en compte importante de la profondeur [1].

Le descripteur HOG (*Histogram of Oriented Gradient*)

Cette méthode consiste à diviser l'image en *cellules*, puis à calculer l'histogramme de l'orientation des gradients pour chacune de ces cellules. On normalise ensuite cet histogramme en fonction de la norme d'un *bloc* de plusieurs cellules auquel appartient la cellule considérée ;

ceci pour pallier aux transformations affines entre les images pouvant être causées notamment par une différence de luminosité. Cette méthode est surtout reconnue pour la détection de personnes ou d'objets.

Le descripteur SIFT (Retenu par la suite pour nos calculs)

Pour calculer ce descripteur, on commence par se placer dans un système de coordonnées lié au point, toujours dans un but d'invariance du descripteur en fonction de la prise de vue. Pour cela, on effectue une rotation de l'image d'angle inverse à celui de l'orientation du point dans l'image calculé à l'étape précédente. On va ensuite considérer une zone de 16x16 pixels autour du point. Le gradient de chacun de ces 256 pixels est alors calculé. De plus, la magnitude du gradient de chaque pixel est pondérée par une fonction gaussienne (de facteur égal à 1,5 fois le facteur d'échelle σ du point-clé) pour réduire le poids des gradients les plus éloignés du centre du patch.
On subdivise ensuite cette fenêtre en 4x4 quadrants de 4x4 pixels chacun. Sur chacun de ces quadrants, on calcule l'histogramme à 8 intervalles des orientations des gradients des 16 pixels qui le composent (sur le même principe que la méthode HOG). Les 16 histogrammes sont ensuite concaténés dans un vecteur de dimension 128 qui est ensuite normalisé. Afin d'améliorer la robustesse du descripteur relativement aux variations photométriques (effets de lumière, reflets...) qui perturbent l'appariement des points, on définit un seuil sur la norme des gradients : il est ici de 0.2. Une nouvelle normalisation de l'histogramme fournit le descripteur SIFT du point considéré.
Les descripteurs SIFT et leurs variantes font partie aujourd'hui des descripteurs les plus robustes et sont donc très utilisés, particulièrement en imagerie 3D.

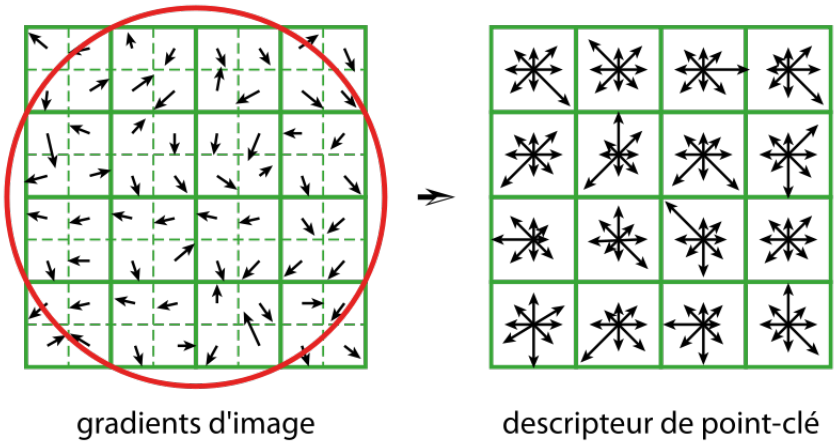


FIGURE 2.3 – *Schéma du calcul des descripteurs* SIFT

2.3 Identification des correspondances

Une fois les descripteurs calculés et stockés dans une base de données, il faut relier les points-clés des différentes prises de vue en comparant leurs descripteurs. Normalement, deux points identiques sur deux images différentes auront des descripteurs très proches (plus proches que de ceux de n'importe quel autre point) : c'est cette propriété que l'on va utiliser pour identifier les différents points. On va alors implémenter un programme de recherche du plus

proche voisin dans la base de données des descripteurs.
L'implémentation naïve de cette recherche consisterait à parcourir, pour chaque point d'une image, la liste des descripteurs d'une autre image pour trouver celui qui s'en rapproche le plus. Cette méthode est bien sûr impossible en pratique car de complexité quadratique en la taille de la liste de descripteurs, et ce, pour seulement deux images ! Il est donc nécessaire de trouver une façon plus efficace d'apparier les descripteurs. Pour cela, deux outils sont couramment utilisés : une table de hachage multidimensionnelle, utilisée par exemple pour la méthode MOPS, ou bien un arbre de recherche multidimensionnel, utilisé pour la méthode SIFT et que nous allons détailler ici.

Description des arbres kd et utilisation

Les descripteurs sont tous indexés dans un arbre à k dimensions nommé *arbre kd*. Le principe des arbres kd, qui sont des arbres binaires de recherche, est de stocker l'ensemble des données dans les hyperplans alignés le long des axes de l'arbre. Plus précisément, chaque noeud de l'arbre contient un point en dimension k (ici un descripteur, en dimension 128). Le noeud sépare ensuite l'espace en deux "demi-espaces", représentés par les branches gauche et droite qu'il engendre. Par exemple, si un noeud divise l'espace selon un plan normal à la direction *(Ox)*, alors tous les points dont la coordonnée *x* est inférieure à celle du noeud seront situés sur la branche gauche, et tous ceux dont l'abscisse est supérieure seront sur la branche droite. L'arbre est construit de manière à avoir la profondeur la plus réduite possible, c'est-à-dire à être équilibré pour que la recherche soit plus rapide et efficace.
La recherche du plus proche voisin d'un descripteur *d* s'effectue alors de la manière suivante, appelée *Best bin first*, littéralement "meilleure boîte en premier" :

1. On cherche à placer *d* dans l'arbre, à l'aide d'un algorithme d'insertion (non décrit ici)
2. Lorsqu'on parvient à une feuille (la fin d'une branche), on la considère comme le "meilleur candidat", appelé *descripteur question*.
3. L'algorithme cherche un descripteur plus proche en explorant les noeuds de l'arbre dans l'ordre de leur distance(euclidienne) au descripteur question.

Ce procédé est généralement très efficace (notamment grâce au fait d'utiliser la distance euclidienne) et fournit des résultats très convaincants. L'algorithme SIFT recherche les deux plus proches voisins du descripteur question. Lorsque le rapport des distances entre ces deux voisins est supérieur à 0.8, la correspondance est considérée comme ambiguë (soupçonnée notamment d'être due à du bruit) et est supprimée. Ceci permet d'éliminer plus de 90% des fausses correspondances.

2.4 Obtention de la liste des correspondances sur un modèle simple

Afin de tester la recherche de correspondances par l'algorithme SIFT, nous avons testé notre implémentation de cette méthode sur différentes projections de points multicolores situés sur les trois faces d'un cube tri-dimensionnel de côté $c = 40\ pixels$.

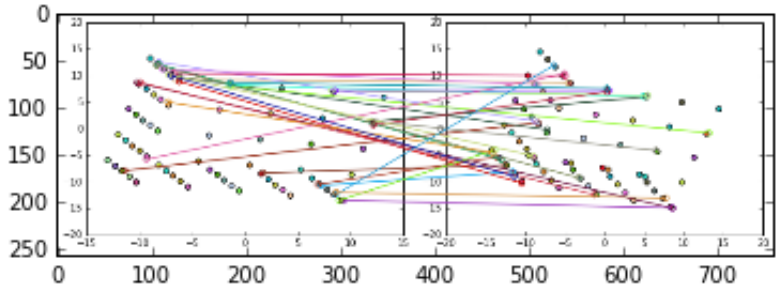


FIGURE 2.4 – *Inclinaison* $-10/-5$ *degrés*

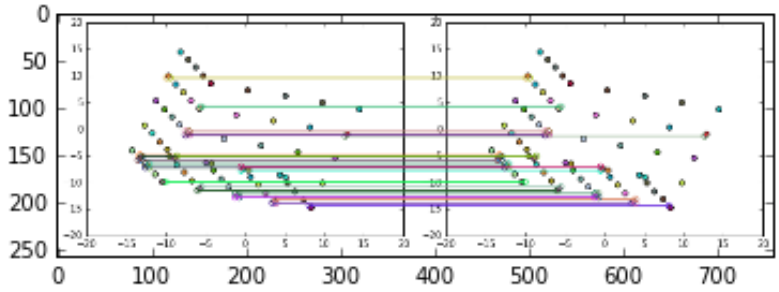


FIGURE 2.5 – *Inclinaison* $-5/0$ *degrés*

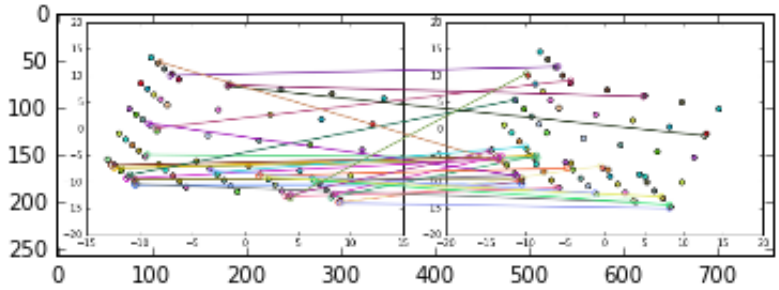


FIGURE 2.6 – *Inclinaison* $0/5$ *degrés*

Comme nous pouvons le voir, nous n'avons pas 100% de correspondances justes même si le résultat est loin dêtre décevant. Il aurait été nécessaire dimplémenter un filtre supplémentaire afin de ne conserver que les correspondances qui étaient justes. Ensuite, lalgorithme peut nous retourner les coordonnées des points-clés qui correspondent sur chacune des deux images. Cette liste de coordonnées peut être utilisée afin de procéder à la reconstruction 3D de l'objet photographié.

Chapitre 3

Reconstitution 3D de l'objet

Une fois les correspondances entre les points d'intérêt des différentes images établies, la deuxième partie de notre procédure consiste à en tirer une représentation tri-dimensionnelle de l'objet photographié. Dans cette partie, nous allons détailler comment, à partir de ces correspondances et de paramètres propres à la caméra utilisée, nous avons abouti à une reconstruction 3D de l'objet initial.

3.1 Principe de triangulation

Toute reconstitution 3D d'un objet à partir d'un ensemble de photographies repose sur le principe fondamental de triangulation. Un point repéré sur une image plane est la projection d'un point réel dans l'espace, que l'on peut alors relier par une droite (Figure 3.1). En considérant plusieurs photographies de ce même point, la position du point réel associé peut être alors obtenue en déterminant le point d'intersection de ces droites : c'est le problème de triangulation.

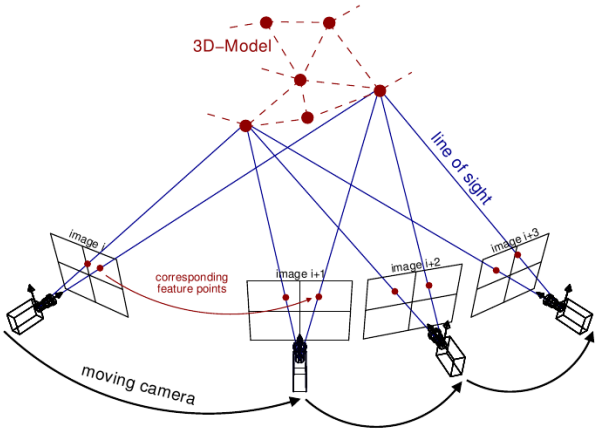


FIGURE 3.1 – *Illustration schématique du principe de triangulation [8]*

Si ce problème semble a priori géométriquement simple, la question de la calibration de la caméra le rend beaucoup plus complexe, et souvent non linéaire. En effet, lors de la projection d'un point de l'espace sur le capteur de la caméra, son orientation et ses paramètres de construction sont autant de paramètres à prendre en compte lors de la triangulation.

Dans le cadre de l'élaboration de notre procédure, nous nous sommes donc intéressés aux différents modèles de caméra, et en particulier à la modélisation de leur projection.

3.2 Modélisation du problème : deux types de projection

3.2.1 Projection perspective

Plusieurs modélisations de la projection sont possibles, ils dépendent en partie du type d'objectif utilisé. Dans la plupart des objectifs le modèle est celui de la projection perspective. Nous considérons ici une caméra à *sténopé*, c'est à dire que les rayons vont tous se croiser en un point, le centre optique, ou centre de projection, de la caméra. Ce point est analogue au point de fuite que l'on retrouve dans l'art pictural avec la perspective linéaire. La projection des points sur le plan image se fait à partir de droites passant par le centre optique et des points.

Projection **perspective** du point $P = (X, Y, Z)$ par la caméra de centre C et distance focale f :

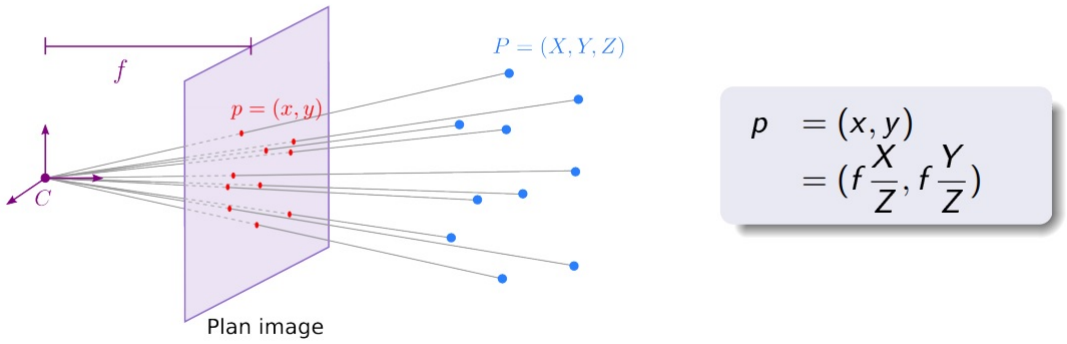


FIGURE 3.2 – *Schéma explicatif du modèle de projection perspective [2]*

L'expression de la matrice de projection perspective sera exprimée dans cette partie. Les notations suivantes seront utilisées :

— Les coordonnées en majuscules correspondent à des coordonnées dans un repère lié à la caméra $M = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$

— Celles, en minuscules, correspondent aux coordonnées sur le plan image $m = \begin{pmatrix} x \\ y \end{pmatrix}$.

— f est la distance focale de l'appareil

— On note C le centre optique. On adopte un système de coordonnées homogènes qui sera utile pour le calcul matriciel.

Par le théorème de Thalès on obtient aisément que :

$$m = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix}$$

En introduisant la matrice de la focale cela revient à :

$$\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}$$

Notons que si l'on cherche à obtenir les coordonnées sur le plan image en pixel, il suffit de multiplier la matrice de la focale par un coefficient correspondant au nombre de pixel par unité de longueur.

Les coordonnées de M sont actuellement relatives à un repère ayant pour origine C, le centre de la caméra, si l'on souhaite partir des coordonnées dans le repère de l'espace 3D, il faut faire intervenir la matrice de rotation du repère caméra par rapport au repère monde ainsi que les coordonnées de C dans le repère de l'espace que lon note C_{lab}.

$$M_{lab} = R^{-1}M + C_{lab}$$

d'où :

$$A = R(M_{lab} - C_{lab})$$

En produit matriciel en coordonnées homogènes on a :

$$A = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{d_3} & -C_{lab} \\ 0 & 1 \end{bmatrix} M_{lab} = \begin{bmatrix} R & -RC_{lab} \\ 0 & 1 \end{bmatrix} M_{lab}$$

En effectuant le produit de la matrice de la focale, de la matrice de rotation et de translation on obtient la matrice de projection perspective :

$$\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} R & -RC_{lab} \\ 0 & 1 \end{bmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \sim \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$P = \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} R & -RC_{lab} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} KR & -KRC_{lab} \end{bmatrix}$$

P correspond à la matrice de projection. Pour reconstituer les points en 3D à partir des points d'une image il faut donc inverser la matrice de projection. Cependant il est alors nécessaire de connaître avec précision les paramètres de la caméra : position de la caméra ainsi que son orientation dans le repère du laboratoire et sa focale. La calibration devient ainsi une étape primordiale à la reconstitution 3D. Nous n'avons malheureusement pas une telle précision dans les intruments à notre disposition c'est ce qui nous a motivé à utiliser un autre modèle de projection, la projection orthographique avec échelle.

3.2.2 Projection orthographique avec échelle

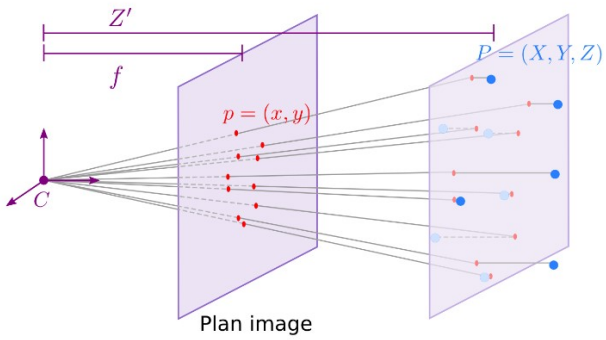
Les difficultés rencontrées vis-à-vis de la projection perspective nous ont poussé à étudier un modèle moins général, mais ne nécessitant pas une connaissance précise de la calibration de la caméra : le modèle de projection orthographique avec échelle.

Défini pour la première fois dans l'article de Poelman et Kanade [6], ce modèle de projection se veut être une approximation du modèle de projection perspective dans le cas où l'objet se trouverait à l'infini sur l'axe de la caméra.

D'un point de vue géométrique, ce modèle de projection consiste, dans un premier temps, à projeter orthogonalement les points sur un plan donné, puis, à effectuer une projection perspective de ces points projetés. En accord avec les modèles trouvés dans la littérature, nous avons décidé de prendre le plan défini par la coordonnée $(P) : Z = \frac{1}{N}\sum_i Z_i$.

Cette méthode de projection, bien que plus éloignée de la réalité que la projection précédente, possède l'avantage de se proscrire des non linéarités. Nous pouvons alors nous rapporter à une fonction de projection linéaire, comme décrit dans [3] pour un point de l'espace X et son projeté sur le capteur de la caméra x :

Projection **orthogonal avec échelle** du point $P = (X, Y, Z)$ par la caméra de centre C et distance focale f :



$$Z' := \frac{1}{N} \sum Z_i$$

$$\begin{aligned} p &= (x, y) \\ &= (\frac{f}{Z'} X, \frac{f}{Z'} Y) \end{aligned}$$

FIGURE 3.3 – *Schéma explicatif du modèle de projection orthographique avec échelle [2]*

$$x = \frac{f}{\vec{k}^T O + t^z} [\begin{pmatrix} \vec{i}^T \\ \vec{j}^T \end{pmatrix} X + \begin{pmatrix} t^x \\ t^y \end{pmatrix} + \begin{pmatrix} c^x \\ c^y \end{pmatrix}] \tag{3.1}$$

où :

— O est l'origine du repère de l'epace, $O = (0, 0, 0)$

— f est la distance focale de la caméra, et $c = \begin{pmatrix} c^x \\ c^y \end{pmatrix}$ son point principal (ici, $c = (0, 0)$)

— $R = \begin{pmatrix} \vec{i} \\ \vec{j} \\ \vec{k} \end{pmatrix}$ est la matrice de rotation et $\vec{t} = \begin{pmatrix} t^x \\ t^y \\ t^z \end{pmatrix}$ le vecteur translation de la caméra par rapport au repère de l'espace

Pour replacer ce modèle de projection dans le cadre de la triangulation, il nous faut exprimer l'équation 3.1 pour un ensemble de N points, photographiés par M caméras de paramètres $C_i, \vec{i_i}, \vec{j_i}, \vec{k_i}, f_i$. La projection x du point X sur la caméra i s'écrit alors :

$$x = \frac{f_i}{t_i^z} [\begin{pmatrix} \vec{i_i}^T \\ \vec{j_i}^T \end{pmatrix} X + \begin{pmatrix} t_i^x \\ t_i^y \end{pmatrix}] = [\begin{pmatrix} \vec{m_i}^T \\ \vec{n_i}^T \end{pmatrix} P^i + \begin{pmatrix} a_i \\ b_i \end{pmatrix}]$$

En considérant les M projections des N points, nous obtenons donc l'équation matricielle suivante :

$$W = \begin{pmatrix} x_1^1 & \dots & x_1^N \\ \vdots & & \vdots \\ x_M^1 & \dots & x_M^N \end{pmatrix} = \begin{pmatrix} \vec{m_1}^T \\ \vec{n_1}^T \\ \vdots \\ \vec{m_M}^T \\ \vec{n_M}^T \end{pmatrix} \begin{pmatrix} X^1 & \dots & X^N \end{pmatrix} + \begin{pmatrix} a_1 \\ b_1 \\ \vdots \\ a_M \\ b_M \end{pmatrix} \begin{pmatrix} 1 & \dots & 1 \end{pmatrix} = RS + T(1 \dots 1) \tag{3.2}$$

Remarques (cf. [3])

— Nous avons besoin d'au minimum 3 vues distinctes de l'objet pour reconstruire l'objet sans ambiguïté : $M \geq 3$

— La matrice W doit être de rang 3 pour permettre à l'algorithme de reconstruction de fonctionner : $N \geq 4$

Si ce modèle de projection nous évite les problèmes rencontrés lors de l'établissement de la matrice de calibration de la caméra, il est d'autant plus pertinent qu'il représente une bonne approximation de la caméra à sténopé à grande distance focale devant l'objet photographié [3]. En effet, les photographies de l'échantillon que nous souhaitons étudier ont été prises avec une caméra de grande distance focale, $f = 5960\ mm$, le modèle de projection orthographique avec échelle nous a semblé être une bonne alternative au modèle général précédent.

3.3 Mise en place de la reconstitution à partir du modèle orthographique avec échelle

Après avoir décidé de se placer dans l'hypothèse d'une caméra à grande focale suivant un modèle de projection orthographique avec échelle, nous avons abordé la question de la reconstruction 3D de l'objet photographié.

Ce problème de triangulation peut être résolu à partir de l'équation matricielle 3.2, de laquelle il nous faut extraire la matrice R et le vecteur T, qui correspondent aux matrices de rotation et au vecteur de translation des M caméras, ainsi que la matrice S qui contient les coordonnées dans l'espace des N points d'intérêt à reconstruire.

Pour ce faire, nous avons décidé d'implémenter [1] la méthode de *Pose Estimation* décrite dans [3], et brièvement reprise ici :

1. Calcul de la matrice $W*_0 = W - T(1 \quad ... \quad 1)$ à partir des correspondances entre les N points d'intérêt
2. Calcul de la décomposition en valeurs singulières de $W*_0$: $W*_0 = U\Sigma V^T$, dont on extrait la meilleure approximation de $W*$ de rang 3 : U', Σ', V'
3. Calcul de $R_0 = U'\Sigma'^{1/2}$ et $S_0 = \Sigma'^{1/2}V'^T$
4. Recherche d'une matrice Q inversible telle que $R = R_0Q$ et $S = Q^{-1}S_0$ vérifiant les contraintes imposées par R, matrice de rotation, et résumées en un système linéaire sur les coefficients de la matrice $\Pi = QQ^T$ définie positive
5. Calcul de R et$S$ à partir de la matrice Q et calcul des paramètres d'orientation de la caméra

De manière analogue à l'implémentation présentée dans [3], nous avons tenu compte de l'ambiguité observée sur la dernière factorisation, étape 4, où l'algorithme peut renvoyer deux reconstructions 3D de l'objet, avec deux orientations opposées (cf. Figure 4.3) : si ce cas est rencontré, le script renverra les deux reconstructions 3D possibles de l'objet.

3.4 Amélioration de l'implémentation : utilisation d'un filtre AC-RANSAC

Lors du premier essai de notre implémentation en langage *Python* de l'algorithme précédent, sur les correspondances pré-calculées sur les images de l'échantillon à étudier, la présence de

1. L'ensemble des scripts rédigés dans le cadre de ce projet sont accessibles sur *GitHub* : `https://github.com/CarolinePascal/PIR_ImagerieNumerique_2017`

valeurs aberrantes ou intraitables a mené l'algorithme à calculer une matrice Π non définie positive, empêchant la reconstruction 3D. Pour pallier ce genre d'incident, nous avons alors décidé d'incorporer un filtrage de type RANSAC des correspondances avant de calculer la triangulation de l'objet.

3.4.1 Principe du filtre AC-RANSAC

RANSAC (RANdom SAmple Consensus) [4], [5], est une méthode itérative qui permet d'estimer les données pertinentes (*inliers*) à un modèle dans un ensemble de données qui contiennent également des données décorélées ou aberrantes (*outliers*). Une méthode des moindres carrés est à proscrire car elle produit un modèle qui cherche à s'ajuster à chaque point même les plus aberrants. L'algorithme RANSAC ne renvoit que des points pertinents, tant que la probabilité de tirer un *inlier* dans lensemble des données est élevée. L'algorithme procède ainsi :

1. Un sous-ensemble de données est choisi dans l'ensemble total, ces données sont considérées comme hypothétiquement pertinentes. Le modèle est ainsi ajusté à ces valeurs.
2. On teste chacune des autres valeurs, si une valeur s'accorde bien avec le modèle alors elle est considérée comme pertinente.
3. En fonction du nombre de nouvelles données pertinentes, le modèle (et donc le sous-ensemble) est considéré comme correct ou non. Le modèle est alors ré-estimé avec les autres valeurs pertinentes.

On répète cela un nombre fixé de fois. On ne garde que le modèle estimé le meilleur.
L'algorithme RANSAC a l'avantage d'être robuste aux valeurs aberrantes. Cependant il n'est pas adapté lorqu'il existe plusieurs modèles optimaux possibles. C'est le cas de notre problème, car il peut exister deux orientations possibles des points en sortie de l'algorithme de reconstruction 3D. De plus il n'y a pas de majoration du nombre d'itérations, ainsi on ne peut pas savoir si l'on a la meilleure solution possible. Il faut alors choisir de manière heuristique un seuil d'acceptation du nombre de données pertinentes : en cas de sur-estimation on inclut donc des valeurs bruitées. Pour pallier à cet inconvénient, une variante de l'algorithme RANSAC est utilisé, c'est l'algorithme AC-RANSAC dit RANSAC a contrario, qui identifie un modèle ainsi que sa précision automatiquement. L'algorithme AC-RANSAC repose sur le principe qu'une forte déviation du modèle de base est très probablement révélatrice. La précision est évaluée à partir d'une mesure de qualité NFA, le nombre de faux pas. Un faux pas correspond à un modèle qui est du à la chance, pour définir un tel modèle il faut partir d'un modèle de base, c'est l'hypothèse 0 qui correspond à une distribution aléatoire et indépendante des points. Cet outil NFA permet également de fournir un critère de validation sous un seuil ϵ puisque l'on cherche à minimiser NFA [5].

3.4.2 Implémentation

Le filtre AC-RANSAC implémenté dans notre procédure suit le fonctionnement théorique explicité plus haut, sur une base d'un seuil NFA unitaire et de 100 itérations au maximum, comme celui implémenté dans [3]. Le filtrage est effectué sur trois images parmi les M considérés : L'algorithme calcule la reconstitution 3D (orthographique) d'un point d'intérêt à partir des deux premières vues et re-projette le point projeté ainsi calculé selon la troisième vue. A partir de ces coordonnées calculées et de celles initialement repérées lors de l'établissement des correspondances, nous pouvons calculer une erreur que l'algorithme AC-RANSAC va pouvoir minimiser en choisissant l'échantillon de points le plus pertinent. Une fois le filtrage effectué, nous pouvons alors calculer, avec plus de précision et de robustesse, la triangulation de l'objet à partir des correspondances entre les points d'intérêts compris dans les inliers.

3.5 Exemple de reconstitution 3D d'objets

Afin de tester notre implémentation de l'algorithme de triangulation, nous avons simulé la prise de photographies et l'établissement de correspondances entre quelques points d'intérêt d'un objet tridimensionnel virtuel, selon un protocole explicité à la section 5.2.1.

3.5.1 Reconstitution 3D d'un demi-cube

Le premier objet tri-dimensionnel que nous avons reconstruit en 3D est un cube de côté $c = 40\ pixels$, comportant $N = 75\ points$ sur ces trois faces et centré sur l'origine du repère $O(0, 0, 0)$. Pour simuler les photographies, nous avons projeté les différents points du cube selon la matrice de projection perspective d'une caméra à sténopé de distance focale $f = 50\ pixels$, orientée selon le vecteur $-\vec{Z}$ et dont le centre de projection est situé en $C(0, 0, 100)$, et le point principal au centre de l'image. Les différentes vues sont alors obtenues en faisant pivoter la caméra autour de l'axe $(O, \vec{Z})$ pour des angles variant de -10 à 10.

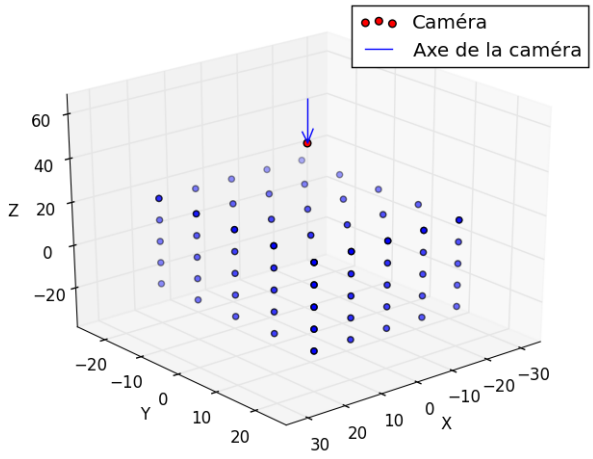


FIGURE 3.4 – *Premier objet tridimensionnel : un demi-cube*

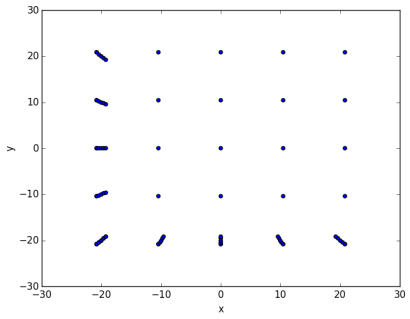
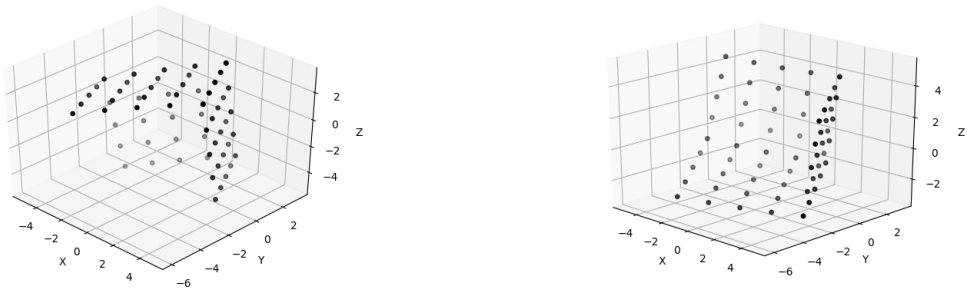


FIGURE 3.5 – *Projection du demi cube sur le capteur de la caméra*



(a) *Orientation 1* (b) *Orientation 2*

FIGURE 3.6 – *Reconstruction 3D du demi-cube selon les deux orientations possibles*

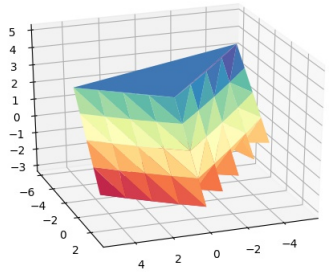


FIGURE 3.7 – *Reconstruction de la surface d'une orientation par triangulation de Delaunay*

3.5.2 Reconstitution 3D d'une coque

Le deuxième objet tri-dimensionnel que nous avons reconstruit en 3D est une coque de largeur $l = 8\ pixels$, de hauteur $h = 80\ pixels$ et de profondeur $p = 20\ pixels$, comportant $N = 200\ points$ sur sa surface. Pour simuler les photographies, nous avons repris la méthode utilisée pour le cube avec la même caméra orientée selon l'axe $-\vec{Z}$ mais dont le centre de projection est situé en $C(10, 0, -600)$.

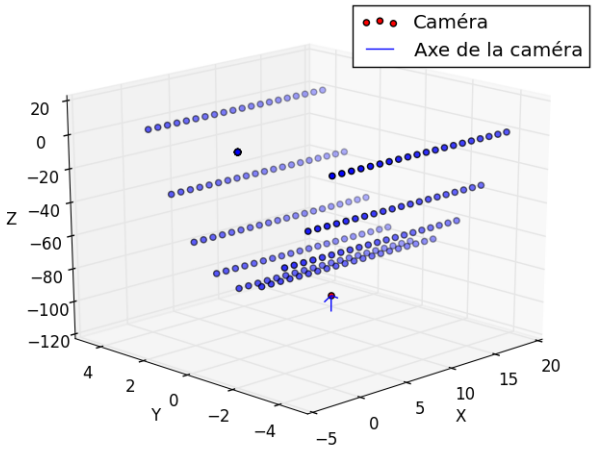


FIGURE 3.8 – *Deuxième objet tridimensionnel : une coque*

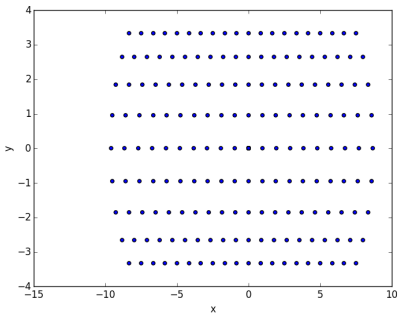
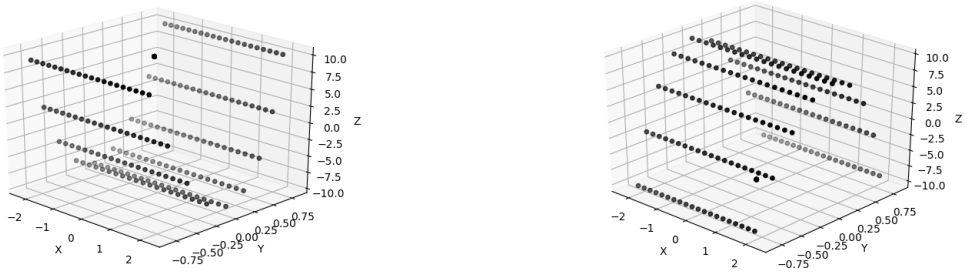


FIGURE 3.9 – *Projection de la coque sur le capteur de la caméra*



(a) *Orientation 1* (b) *Orientation 2*

FIGURE 3.10 – *Reconstruction 3D de la coque selon deux orientations possibles*

3.6 Reconstitution de la surface

Une fois les points restitués en 3 dimensions il faut ensuite reformer la surface de l'objet étudié. C'est un problème de restitution de surface à partir d'un nuage de points désorganisés. Nous avons choisi d'utiliser une approche basée sur la triangulation de Delaunay en 2D et une interpolation. La triangulation de Delaunay permet de créer un maillage des points en 3 dimensions

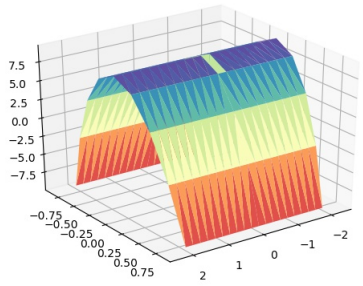


FIGURE 3.11 – *Reconstruction de la surface d'une orientation par triangulation de Delaunay*

sur un plan. L'idéal est de choisir un plan où les projections des points se superposent le moins possible. La triangulation s'explique facilement à partir du diagramme de Voronoi qui est son problème dual. Prenons un espace métrique E de dimension quelconque et S un ensemble fini de points de E. Le diagramme de Voronoï correspond à une subdivision de E en sous-ensemble convexe contenant un unique point s de S et tous les points de E étant plus proche de s que de n'importe quel autre point de S.

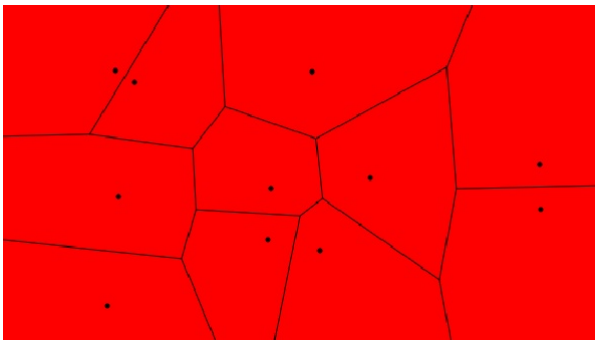


FIGURE 3.12 – *Diagramme de Voronoi en 2D*

On obtient la triangulation de Delaunay de en reliant les points dont les domaines de Voronoï sont adjacents.

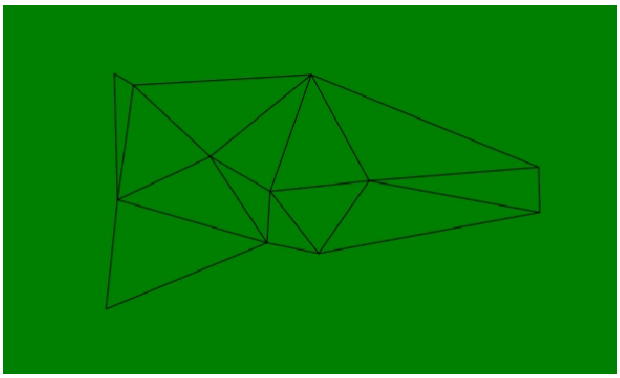


FIGURE 3.13 – *La triangulation de Delaunay correspondante*

La triangulation de Delaunay définit le maillage triangulaire sur le plan de projection des

points en 3 dimensions sur lequel on effectuera l'interpolation de la surface. Il faut ensuite effectuer une interpolation biliénaire sur chaque élément du maillage afin de totalement reconstituer la surface. Cela consiste à évaluer la valeur de la position de la coordonnées orthogonal au plan de projection d'un point de l'élément à partir d'une combinaison linéaire des coordonnées des sommets. Pour un point de coordonnée (x, y) appartenant à un élément du maillage, on obtient sa 3ème coordonnée, selon l'axe orthogonal au plan ainsi : $z = ax + by + c$ avec a, b, c des points obtenus à partir des sommets de l'élément. Il est alors évident de voir que plus le nombre de points est élevé plus la surface reconstituée sera fidèle à la réalité.

Chapitre 4

Résultats

4.1 Calcul des correspondances entre deux vues de l'échantillon

Afin d'avoir la liste des correspondances, on applique la méthode SIFT comme décrit plus haut. Faute de temps, nous n'avons pas pu utiliser la liste de correspondances obtenue pour la reconstruction 3D. Nous obtenons graphiquement le résultat ci-dessous pour deux projections différentes de l'échantillon :

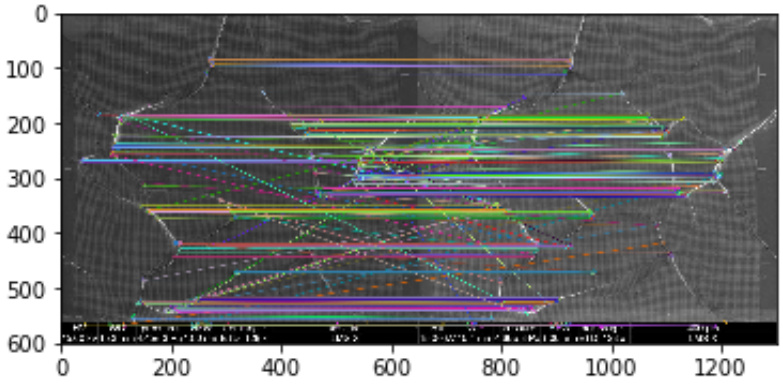


FIGURE 4.1 – *Correspondances pour deux projections différentes de l'échantillon*

4.2 Reconstitution 3D de l'échantillon à partir des correspondances pré-calculées

Afin de restituer au mieux la surface de l'échantillon considéré, nous avons effectué un post-traitement ([3]) des points tri-dimensionnels calculés par notre algorithme : le repère est modifié afin de ramener le centre de projection de la première caméra (resp. projection) à l'origine du repère 3D et l'échelle est réglée de sorte que la distance entre le centre de projection de la première caméra (resp. projection) et de la deuxième soit unitaire.

Le calcul de triangulation de la surface de l'échantillon a été réalisé partir des correspondances calculées entre cinq sites différents du maillage imprimé sur celui-ci, comme illustré sur la figure 4.2.

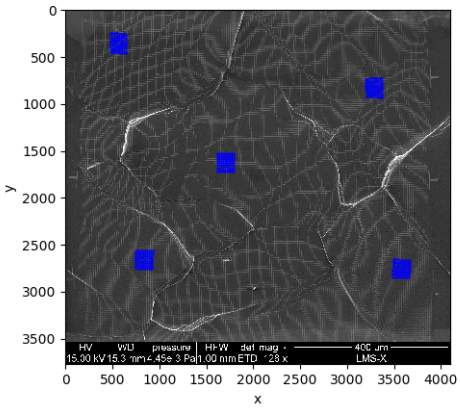
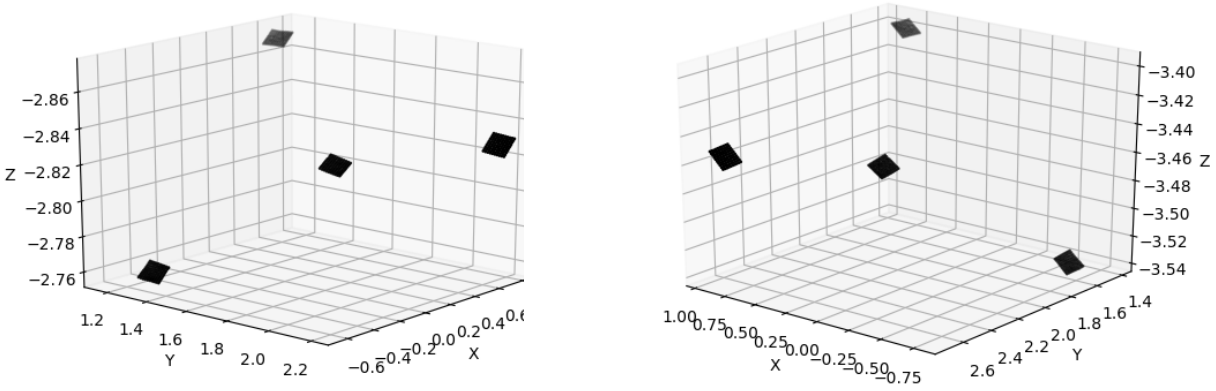
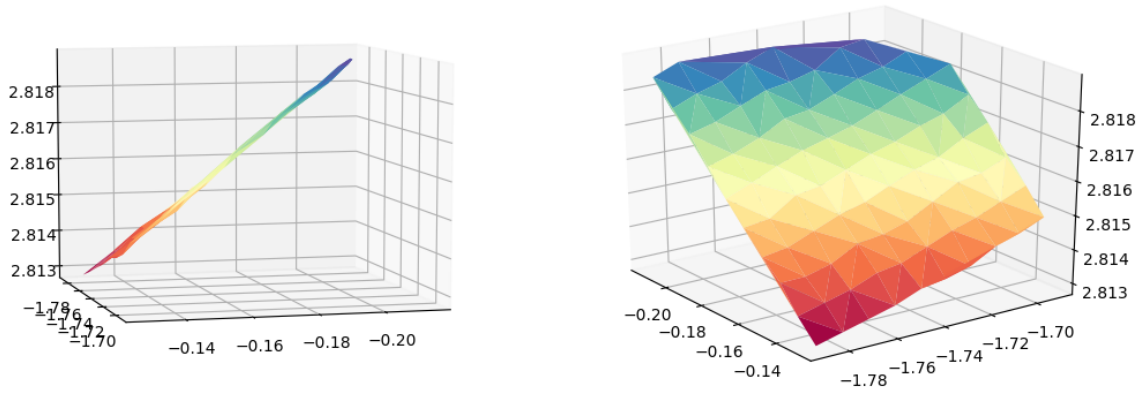


FIGURE 4.2 – *Cartographie des zones sur la surface de l'échantillon dont les correspondances ont été pré-calculées*



(a) *Orientation 1* (b) *Orientation 2*

FIGURE 4.3 – *Reconstruction 3D des différentes zones de la surface de l'échantillon selon les deux orientations possibles*



(a) *Résultat en coupe*

(b) *Résultat en face*

FIGURE 4.4 – *Reconstruction 3D de l'une des 4 zones après triangulation de Delaunay et interpolation*

Une fois les points reconstitués en trois dimensions, on peut maintenant chercher à reformer la structure apparente de l'échantillon. Pour cela on utilise la triangulation de Delaunay expliquée dans 3.6.

Chapitre 5

Analyse de la précision et de la robustesse des algorithmes

5.1 Etude de la précision et de la robustesse de l'algorithme de recherche de correspondances

Impact du bruit généré par la caméra sur les correspondances Nous voulions calculer l'influence du bruit sur la précision des correspondances entre points-clés. Afin de rendre compte de cela, nous avons pris deux images d'une même scène qui diffèrent de 90 degrés. Cette scène est constituée de points de différentes couleurs afin de différencier les descripteurs SIFT autour de chaque point pour faciliter la correspondance entre les points-clés des deux images. A partir des points dintérêt de la première image, nous avons pu calculer les cordonnées de ceux correspondants théoriquement après une rotation de 90 degrés. Nous avons par la suite pu étudier la précision des correspondances à deux pixels près en comparant les coordonnées des points-clés de la deuxième image exposée à un bruit gaussien avec ceux calculés théoriquement.

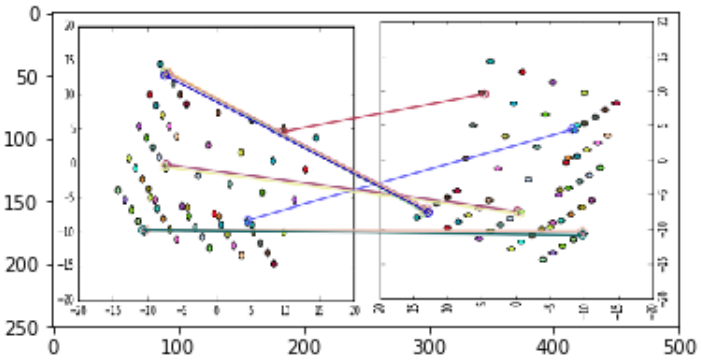


FIGURE 5.1 – *Correspondances entre les deux images avec un bruit gaussien d'écart-type nul*

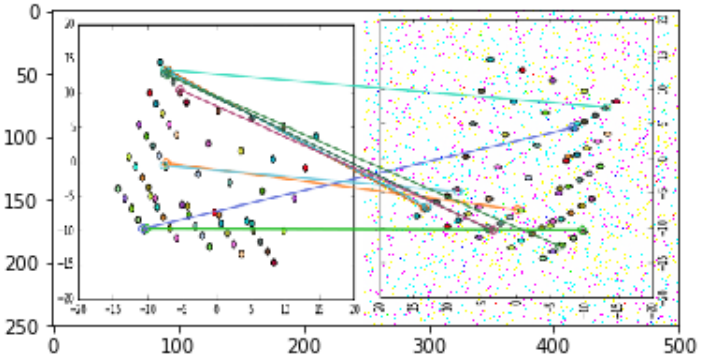


FIGURE 5.2 – *Correspondances entre les deux images avec un bruit gaussien d'écart-type 0.5*

Nous voyons donc que la présence d'un bruit faible généré par le capteur ne diminue pas la précision des correspondances. Ceci est cohérent avec la théorie car la popularité de la méthode SIFT est surtout due à sa robustesse. Néanmoins, au-dessus d'un certain seuil, nous voyons que la précision des correspondances diminue drastiquement. En effet, le bruit modifie l'orientation et l'amplitude du gradient ce qui a pour cause de modifier les descripteurs SIFT de la deuxième image. Les descripteurs entre points-clés correspondants sont donc de moins en moins similaires ce qui a comme conséquence de diminuer le nombre de correspondances réussies.

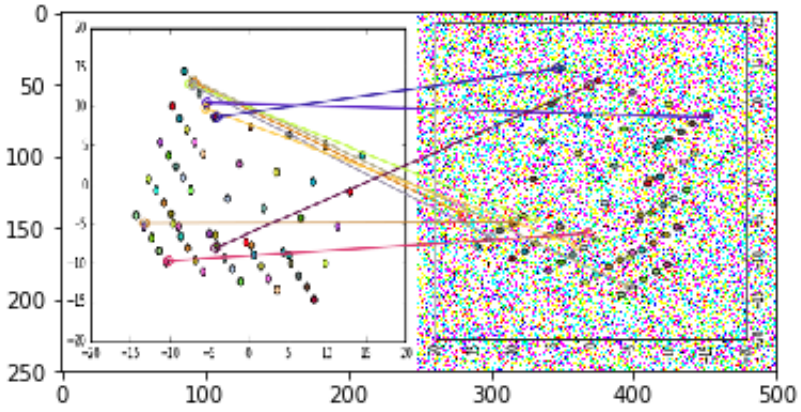


FIGURE 5.3 – *Correspondances entre les deux images avec un bruit gaussien d'écart-type 1*

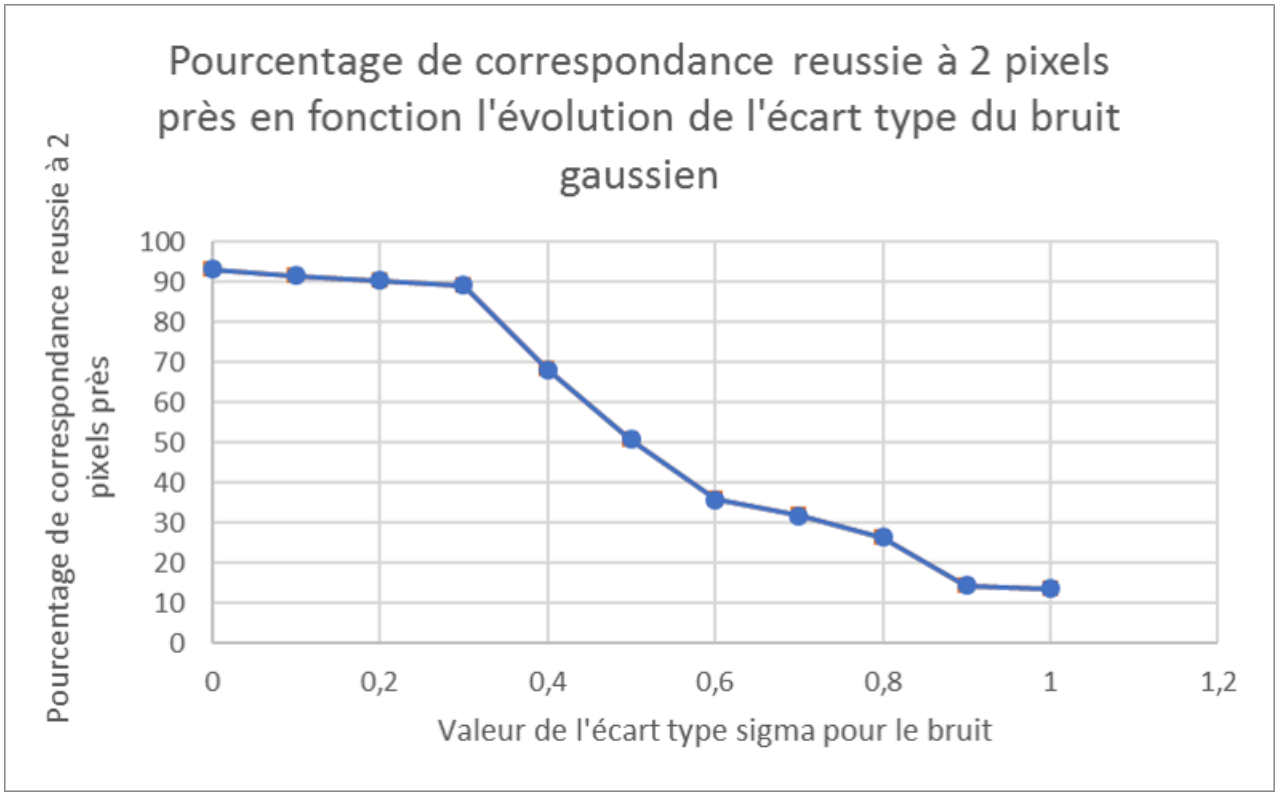


FIGURE 5.4 – *Pourcentage*

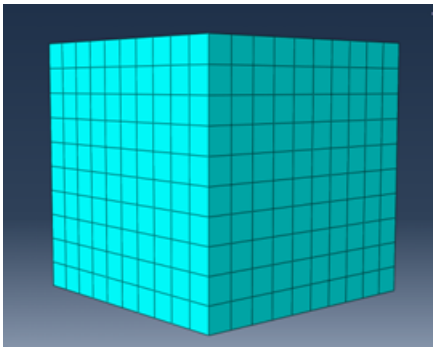


FIGURE 5.5 – *Capture d'écran du cube réalisé sur Abaqus avec un maillage régulier.*

5.2 Etude de la précision et de la robustesse de l'algorithme de reconstitution 3D

Après avoir établi le programme de reconstitution, nous avons voulu le tester. Pour cela, nous avons décidé de créer un objet simple sur lequel nous pourrions marquer des points. On note les coordonnées de ces points dans les différentes positions de l'objet. Nous avons choisi un cube de coté $c = 40\ pixels$. C'est un objet simple à simuler mais également à vérifier : il est facile de voir si la reconstitution est fidèle à l'objet réel. À l'aide du logiciel *Abaqus* nous avons donc pu créer un cube virtuel. Nous avons ensuite relevé les coordonnées de certains points du cube. Afin de reconstituer l'objet, et en s'inspirant des fichiers dont nous disposions, nous avons fait le choix de prendre 25 points par face. On utilise un repère orthonormé direct que l'on centre au centre du cube.

Les huit sommets du cube :

$$A_1 = (a, -a, a) A_2 = (a, a, a) A_3 = (-a, a, a) A_4 = (-a, -a, a)$$
$$A_5 = (-a, -a, -a) A_6 = (a, -a, -a) A_7 = (a, a, -a) A_8 = (-a, a, -a)$$
$$, a = \frac{c}{2}$$

A partir de ces sommets, on peut exprimer les points de faces. En effet, un plan étant entièrement caractérisé par trois points on a :

$$\forall P \in F, \exists (t, u) \in [0, 1], \underline{OP} = t(\underline{OB} - \underline{OA}) + u(\underline{OC} - \underline{0A})$$

où A,B,C sont trois sommets définissant et F la face contenant ces trois points.

On prend t et u allant de 0 à 1, avec un pas de 0,2. On obtient ainsi 25 points par face, ce qui nous donne un bon échantillon de la surface (d'autant qu'elle est plane !). On a ainsi les coordonnées d'un nombre conséquent de points définissant le cube. On peut utiliser différents modèles de caméra pour projeter les points dans le plan image.

5.2.1 Précision

La première étape de l'analyse des erreurs de notre algorithme est l'étude de la précision des reconstructions 3D qu'il calcule. Pour ce faire, nous avons repris le modèle du demi-cube

illustré dans la section 3.5, dont nous avons pu constater que la reconstruction 3D était assez proche du cube initial.

Or, en voulant comparer les deux motifs tri-dimensionnels, nous nous sommes aperçus que la reconstruction 3D calculée ne possédait ni les même dimensions, ni la même orientation que l'objet initial, rendant délicate tout quantification de la ressemblance en terme de forme des deux ensembles de points.

5.2.2 Robustesse

Dans un second temps, nous avons mené une évaluation quantitative de la robustesse de notre algorithme de reconstruction 3D vis à vis du bruit induit par le capteur de la caméra. Pour évaluer la robustesse de l'algorithme, nous avons repris le modèle du demi-cube que nous avons testé de la manière suivante :

- Génération d'un demi-cube tri-dimensionnel de côté $c = 40 pixels$, comportant 25 points par face
- Simulation de la prise de photographie avec une caméra à sténopé de distance focale f fixée à $50\ pixels$: projection des N points selon une projection perspective, avec rotation d'un angle $\theta \in -10, -5, 0, 5, 10$ de l'objet 3D entre chaque prise de vue
- Simulation du bruit associé au capteur de la caméra : ajout d'un bruit blanc gaussien d'écart-type σ aux coordonnées des projections des N points
- Reconstruction 3D du demi-cube non-bruité, demi-cube de référence, et du demi-cube bruité à partir des correspondances, connues, entre les différents points projetés
- Évaluation des écarts entre le demi-cube de référence et le demi-cube bruité

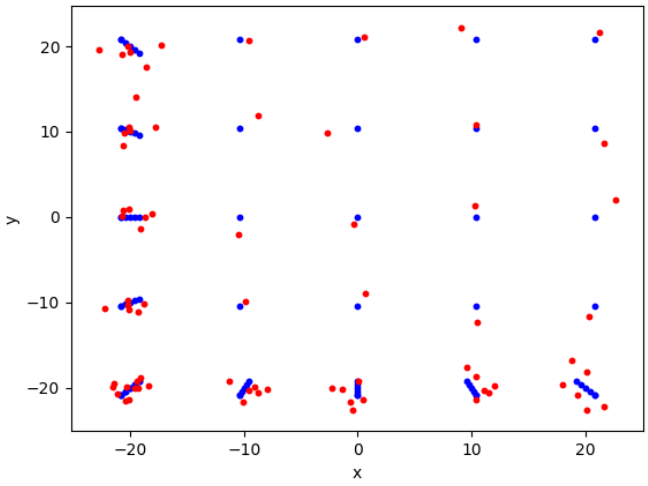


FIGURE 5.6 – *Superposition de la projection du demi-cube de référence (bleu) et de la projection du demi-cube bruité (rouge)*

Afin d'évaluer les écarts entre les deux reconstructions tri-dimensionnelles, nous avons sommé la norme des distances euclidiennes entre les points de référence et les points bruités, puis divisé le résultat par le nombre de points N multiplié par le côté du cube c, pour obtenir un résultat homogène à une erreur moyenne :

$$Err_moy = \frac{1}{N} \sum_{i=1}^{N} ||p_ref - p_bruit||$$

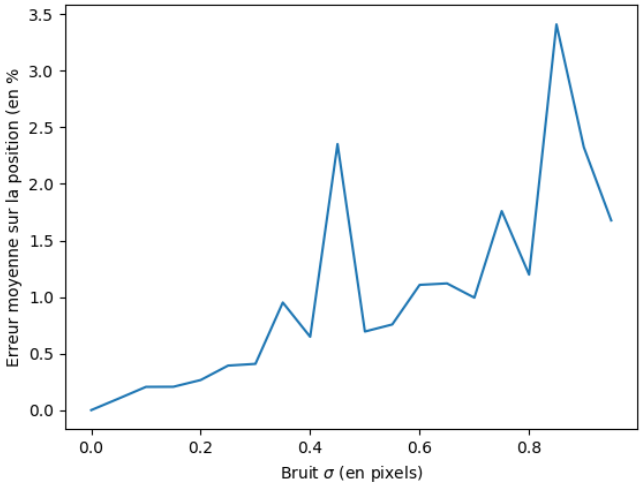


FIGURE 5.7 – *Évolution de l'erreur moyenne de la reconstruction 3D du demi-cube (avec et sans filtre AC-RANSAC) en fonction de l'écart-type du bruit gaussien* σ

Robustesse face au bruit du capteur de la caméra La courbe 5.7 nous indique que le comportement de notre algorithme face au bruit induit par le capteur de la caméra est très irrégulier, et fluctue de manière importante avec les valeurs de σ. Toutefois, on remarque sur les deux courbes que l'erreur moyenne a tendance à augmenter avec l'écart-type, ce qui, couplé avec les pics irréguliers, nous mène à une erreur maximale à hauteur de 3.5% du côté du cube, soit $0.7\ pixel$, avec et sans le filtre AC-RANSAC, pour un bruit d'environ 1 pixel.

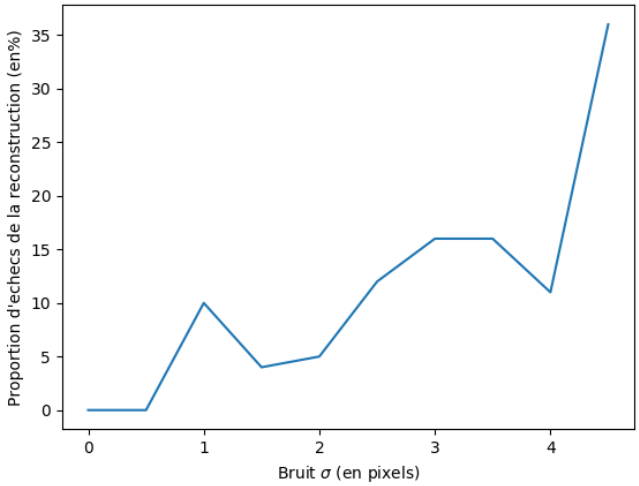


FIGURE 5.8 – *Evolution de la proportion d'échecs de l'algorithme (avec et sans filtre AC-RANSAC) en fonction de l'écart-type du bruit gaussien* σ

L'étude de la réponse de l'algorithme face a un bruit gaussien n'a été menée que pour une plage d'écarts-types assez petite, $\sigma \in [0, 1]$, à cause du nombre grandissant d'échecs (cf. section 3.4) de la reconstruction 3D en fonction de celui-ci. En effet, comme le montre la figure 5.8, la proportion d'échecs de notre algorithme, avec ou sans filtre AC-RANSAC, augmente progressivement avec les valeurs de l'écart-type, constituant un autre défaut de robustesse pour notre algorithme.

Les erreurs et la proportion d'échec induites par la présence de bruit en amont de l'algorithme sont donc assez importantes, il nous faut donc essayer de réduire au maximum le bruit créé à la fois au niveau du capteur de la caméra, mais aussi lors du calcul des correspondances entre les points d'intérêts des différentes images.

Chapitre 6

Discussions

6.1 Algorithme de calcul de correspondances

6.1.1 Optimisations possibles de le méthode SIFT et reconstruction 3D

Les résultats obtenus par l'algorithme que nous avons implémenté sont satisfaisants mais améliorables. Pour les améliorer, nous aurions souhaité implémenter un filtre supplémentaire afin de conserver uniquement les correspondances justes. Grâce à ce filtre, nous aurions pu générer une liste de correspondances bien plus fiables qui auraient pu nous permettre une reconstruction 3D pertinente. Si nous avions eu un peu plus de temps, nous aurions souhaité tester d'autres méthodes de correspondances pour pouvoir les comparer et conserver la plus efficace.

6.2 Algorithme de reconstruction 3D

6.2.1 Non-concordance des dimensions et de l'orientation

Lors de la reconstitution 3D, nous obtenons effectivement un cube, mais un problème survient : le cube reconstitué est bien plus petit que le cube initial et orienté différemment. Ce problème est probablement dû à l'absence de post-traitement des points 3D que calcule l'algorithme, notre script se contente de renvoyer la matrice S de l'équation matricielle 3.2, sans se poser la question du passage du repère de l'objet réel, liée à l'espace, à celui de la caméra lié à l'image. Un post-traitement de ces points, similaire à celui implémenté dans [3] pourrait alors être une solution à ce problème.

De plus, comme nous l'avons remarqué dans la section 5.2.1, cette différence de taille et d'orientation nous empêche d'évaluer la précision de notre algorithme, selon la méthode classique mise en place pour la quantification de sa robustesse. Pour palier ce problème, il nous faudrait un outil capable de vérifier la ressemblance entre deux objets, sans tenir compte de sa dimension et de son orientation, en quelques sorte, une quantification de sa forme.

6.2.2 Modèle de caméra : la projection orthographique avec échelle

Dans le cadre de la modélisation du problème de triangulation, nous avons adopté, au regard des données concernant la caméra utilisée pour photographier les échantillons, l'hypothèse d'une caméra à projection orthographique avec échelle (cf. section 3.2.2).

Cette projection, qui nous permet par la suite de reconstituer l'objet en trois dimensions, repose sur une hypothèse : celle d'une distance focale importante devant les dimensions de l'objet photographié. Si dans notre situation, la caméra utilisée possède une distance focale $f = 5690mm$ assez grande devant la taille des échantillons, cette hypothèse rend notre modèle très imprécis lorsqu'il s'agit de traiter des images prises avec des caméras à faible distance focale.

D'autre part, lorsqu'il est confronté à un objet dont la surface est plane, le modèle orthographique s'avère être défectueux, car incapable de distinguer les variations de relief selon l'axe de visée de la caméra [3]. C'est le cas, par exemple, de la zone en bas à droite de l'échantillon étudié, représenté sur la figure 4.2, qui est filtré comme *outlier* par le filtre AC-RANSAC et donc non représentée sur la reconstruction 3D.

Ainsi, face à un problème de reconstruction 3D impliquant des caméras à faible distance focale, ou des déformations quasi-planes, il faudra avoir recours à un modèle de projection perspective, plus général, mais dont l'implémentation est plus complexe, car nécessitant une très bonne calibration de la caméra utilisée [7].

Chapitre 7

Conclusions

Le modèle de projection utilisé s'est révélé efficace pour les focales et déformations considérées. La reconstitution en trois dimensions des objets est plutôt fidèle à la réalité, malgré certains problèmes d'échelle et d'orientation. Notre programme de correspondances de points nécessite un filtre AC-RANSAC qui n'est efficace que si la transformation est proche de celle choisie en théorie. Ce filtrage nécessite donc un bon modèle de transformation au préalable. On pourrait également imaginer d'autres moyens de filtrer les fausses correspondances. Le temps nous a également manqué pour créer un programme complet, qui prend une série de photographies en argument et renvoie sa reconstitution tri-dimensionnelle. La connexion entre la correspondance des points et l'algorithme de reconstitution à partir des séries de coordonnées n'a pas pu être faite dans les temps.

Bibliographie

[1] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. San Diego, June 2005.

[2] Laura F. Julia. Estimation de pose à trois vues : les mérites respectifs du tenseur trifocal et du modèle orthographique. 2016.

[3] Laura F. Julia, Pascal Monasse, and Marc Pierrot-Deseilligny. The Orthographic Projection Model for Pose Calibration of Long Focal Imges. *Image Processing On Line*, March 2017.

[4] Lionel Moisan, Pierre Moulon, and Pascal Monasse. Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers. *Image Processing On Line*, 2 :56–73, May 2012.

[5] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive Structure from Motion with a contrario model estimation. volume 7727, pages 257–270. Springer, November 2012.

[6] Conrad Poelman and Takeo Kanade. A Paraperspective Factorization Method for Shape and Motion Recovery. *DTIC Document*, 1993.

[7] Peter Sturm. *Vision 3D non calibrée : contributions à la reconstruction projective et étude des mouvements critiques pour l'auto-calibrage*. phdthesis, Institut National Polytechnique de Grenoble - INPG, December 1997.

[8] Chris Sweeney. Theia vision library.

[9] Richard Szeliski. *Computer Vision, Algorithm and Applications*. Springer.