UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
OUSSAMA MEGOUAS

RECONSTRUCTION 3D POUR LA DETECTION DES MAUVAISES HERBES DANS UN SYSTÈME ROBOTIQUE

(JUIN 2021)

Maîtrise en Mathématique et Informatique Appliquée

RECONSTRUCTION 3D POUR LA DETECTION DES MAUVAISES HERBES DANS UN SYSTÈME ROBOTIQUE:

Megouas Oussama

Superviseur:	Meunier François DMI, Université de Quebec a Trois-Rivieres
Examinateur:	Dominic Rochon Fathallah Nouboud

Abstract

Depth estimation using stereo images is an important task in many computer vision applications. A stereo camera contains two image sensors that observe the scene from slightly different viewpoints, making it possible to find the depth of the scene. An active stereo camera also uses a laser projector that projects a pattern into the scene. The advantage of the laser pattern is the additional texture that gives better depth estimations in dark and textureless areas.

Recently, deep learning methods have provided new solutions producing state- of-the-art performance in stereo reconstruction. The aim of this project was to investigate the behaviour of a deep learning model for active stereo reconstruction, when using data from different cameras. The model is self-supervised, which solves the problem of having enough ground truth data for training the model. It instead uses the known relationship between the left and right images to let the model learn the best estimation.

The model was separately trained on datasets from three different active stereo cameras. The three trained models were then compared using evaluation images from all three cameras. The results showed that the model did not always perform better on images from the camera that was used for collecting the training data. However, when comparing the results of different models using the same test images, the model that was trained on images from the camera used for testing gave better results in most cases.

Résumé

L'estimation de la profondeur à l'aide des images stéréo est une tâche importante dans de nombreuses applications de vision par ordinateur. Une caméra stéréo contient deux capteurs d'image qui observent la scène depuis des points de vue différents, permettant de retrouver la profondeur de la scène. Une caméra stéréo active utilise également un projecteur laser qui projette un motif dans la scène. L'avantage du motif laser est la texture supplémentaire qui donne une meilleure estimation de la profondeur dans les zones sombres et sans texture.

Récemment, les méthodes d'apprentissage en profondeur ont fourni des nouvelles solutions produisant des performances de pointe en matière de reconstruction stéréo. L'objectif de ce projet était d'étudier l'application d'un modèle d'apprentissage en profondeur pour la reconstruction stéréo active, lors de l'utilisation de données provenant de différentes caméras. Le modèle est auto-supervisé. Il utilise la relation connue entre les images gauche et droite pour permettre au modèle d'apprendre la meilleure estimation.

Le modèle a été entraîné séparément sur des ensembles de données provenant de trois caméras stéréo actives différentes. Les trois modèles entraînés ont ensuite été comparés à l'aide des images d'évaluation des trois caméras. Les résultats ont montré que le modèle n'était pas toujours plus performant sur les images de la caméra utilisée pour collecter les données d'entraînement. Cependant, lors de la comparaison des résultats de différents modèles utilisant les mêmes images de test, le modèle formé sur les images de la caméra des tests a donné de meilleurs résultats dans la plupart des cas.

Table des matières

Notation

Abbreviations

ANN	Artificial neural networks
CLR	Cyclic learning rate
CNN	Convolutional neural network
LCN	Local contrast normalization

1

Introduction

Récupérer la structure 3D d'une scène à partir d'images est depuis longtemps l'un des principaux intérêts de la vision par ordinateur. Bien qu'un travail considérable dans ce domaine a été consacré à la réalisation de reconstructions de haute qualité, quelle que soit la durée de reconstruction, d'autres travaux ont porté sur l'aspect de la reconstruction en temps réel et ses applications. Surtout au cours de la dernière décennie, l'intérêt pour les systèmes de reconstruction 3D en temps réel a considérablement augmenté en raison des nombreuses applications, tant industrielles que scientifiques, qui sont rendu possible par cette technologie. Ce memoire s'inscrit dans ce sujet en présentant un système de reconstruction 3D en temps réel utilisant de nouveaux algorithmes de reconstruction. En utilisant ce système, nous présentons plusieurs applications innovantes, en se concentrant en particulier sur les milieux agricoles afin de montrer les avantages et les spécifications des systèmes de reconstruction 3D multi-caméras dans l'avenir des robots agricoles.

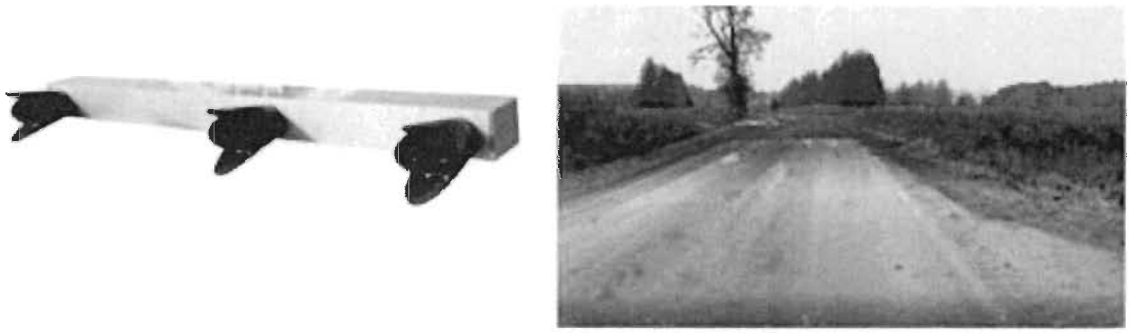


Figure 1.1 Système de caméra stéréo pour véhicules, superposition d'image de caméra avec données de profondeur (blue=proche, rouge=loin) [https://ercim-news.ercim.eu/en95/special/visual-3d-environment-reconstruction-for-autonomous-vehicles]

1.1 Domaines d'application de la reconstruction 3D en temps réel

Avant d'aborder de plus près le sujet de la reconstruction 3D en temps réel dans le domaine agricole dans la section suivante, nous voulons d'abord donner un bref aperçu de certaines utilisations populaires des systèmes de reconstruction 3D afin de montrer les applications possibles de cette technologie. En particulier, nous verrons la robotique, la technologie des voitures autonome et la capture de mouvement.

1.1.1 Technologie des voitures autonomes

Dans un avenir prévisible, il sera courant que divers véhicules soient équipés de capteurs et des systèmes 3D qui reconstruisent la zone avoisinante de ces véhicules en 3D. Cette technologie peut être utilisée dans le cadre d'un système avancé d'aide à la conduite (ADAS) pour un fonctionnement semi-autonome (pilote automatique), ou pour un fonctionnement totalement autonome, en fonction du niveau de maturité technologique et des réglementations. Les systèmes robotiques existants sont principalement équipés de capteurs 3D actifs tels que des dispositifs de scanner laser ou des capteurs de time-of-flight (TOF)

[1]. Les capteurs 3D basés sur des caméras stéréo (voir Figure 1.1) coûtent moins cher et fonctionnent bien même dans une lumière ambiante intense, mais le processus de reconstruction 3D est plus complexe [1].

Le système de reconstruction et de cartographie 3D basé sur la vision stéréo, également connu sous le nom de «localisation et cartographie simultanées» (SLAM), a été largement abordé avec les plateformes robotiques dans les scénarios d'intérieur [1]. Dans de tels environnements créés par l'homme, beaucoup d'hypothèses simplifiées peuvent être faites (sol horizontal, murs plat et géométrie perpendiculaire) [1].

Ainsi, de simples appareils laser, qui scanne un seule plan horizontal sera suffisants pour créer une carte 2D (floor plan). Lors du déplacement vers des environnements extérieurs, qui est la zone de travail de la grande majorité des véhicules, ce type de représentation 2D devient insuffisant et doit être étendu à une représentation 2.5D appelée carte d'élévation. Les capteurs 3D sont désormais nécessaires pour capturer la scène dans son ensemble [1]. Les systèmes de vision stéréo ont cette capacité.

Note : Les systèmes de vision stéréo projettent les valeurs de distance dans un angle solide sur un plan d'image le résultat est une image en profondeur. Une gamme de facteurs dans la conception d'un système de vision stéréo détermine la qualité et la précision des images en profondeur, qui déterminent la faisabilité de la stéréo pour l'application envisagée : L'algorithme de correspondance stéréo doit fonctionner à des fréquences d'images interactives et à une résolution d'image élevée [1]. La géométrie stéréo et la résolution de la caméra doivent fournir une résolution de profondeur suffisante, même pour de plus grandes plages. [1]

Les vues uniques générées par les capteurs 3D souffrent de champs de vision et de résolutions limitées ainsi que d'occlusions. Par conséquent, obtenir une reconstruction plus complète de l'environnement d'un véhicule nécessite la combinaison de plusieurs vues de différentes positions. Les données 3D doivent être traitées et fusionnées dans un cadre de coordonnées commun, mais il est généralement capturé par rapport à un système de coordonnées de capteur [1]. Par conséquent, il y a une tâche importante afin de récupérer le ego-motion du

véhicule entre des mesures consécutives et pour la localiser de véhicule dans le cadre commun [2].

1.1.2 Robotique

Comprendre la géométrie de la scène est essentiel pour la vision artificielle en robotique. Aussi, la modélisation 3D précise d'objets spécifiques ou d'une scène peut être très bénéfique pour le processus de planification ou du Mapping pour un robot. En raison de la richesse de leurs résultats, les algorithmes de reconstruction 3D sont largement utilisés dans ce type de technologie [3].

Et ce type de technologie est tout autant importante en robotique que pour la vision humaine et ce pour l'évolution de l'humanité, toutes les tâches comme la planification, le mapping ou la capture d'objets, nécessitent une reconstruction 3D propre pour l'environnement extérieur.

Par exemple, les humains ont une capacité remarquable à saisir et à reconnaître des objets inconnus avec peu de connaissances préalables. Ils sont une inspiration constante pour les chercheur en robotique. Les solutions classiques pour titre d'exemple le ramassage de pièces (robotic picking) nécessitent une reconnaissance et une estimation de la pose préalable par un processus de planification de ramassage orienté modèle (model-based grasp planning), ou exiger une segmentation d'objets pour associer des détections d'agrippement couple avec identification d'objet [3] (voir figure 1.2). Ces solutions ont tendance à échouer lorsqu'il s'agit d'un nouvel objet dans des environnements encombrés, car ils s'appuient sur des modèles d'objets 3D et / ou de grandes quantités de données d'entraînement pour obtenir des performances robustes [3].

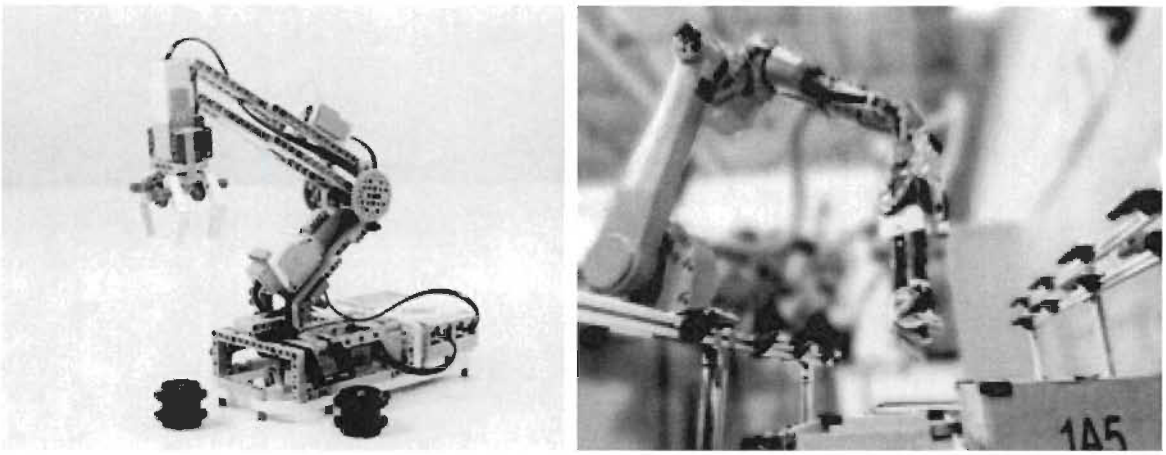


Figure 1.2 Bras robotiques qui utilisent la reconstruction 3D pour détecter les formes et les positions des objets [https://arxiv.org/abs/1710.01330]

1.1.3 Capture de mouvement

Le but de la capture de mouvement est de transférer les mouvements d'un acteur vers un personnage animé afin d'obtenir des animations réalistes dans les films ou les jeux vidéo. Pour atteindre cet objectif, la configuration conjointe des acteurs pour récupérer les scènes. Cela se fait traditionnellement en attachant des marqueurs à l'acteur qui sont suivis et utilisés pour déduire la configuration conjointe.

Bien que courants dans la pratique, l'inconvénient de ces systèmes est qu'ils sont lourds à utiliser et nécessitent généralement que l'acteur porte des vêtements serrés, puisqu'un squelette doit être adapté aux observations. Idéalement, on aimerait retirer les marqueurs et faire porter des vêtements arbitraires à l'acteur. Des recherches dans ce sens ont notamment été poussées par le groupe autour d'Adrian Hilton a l'université de Surrey.

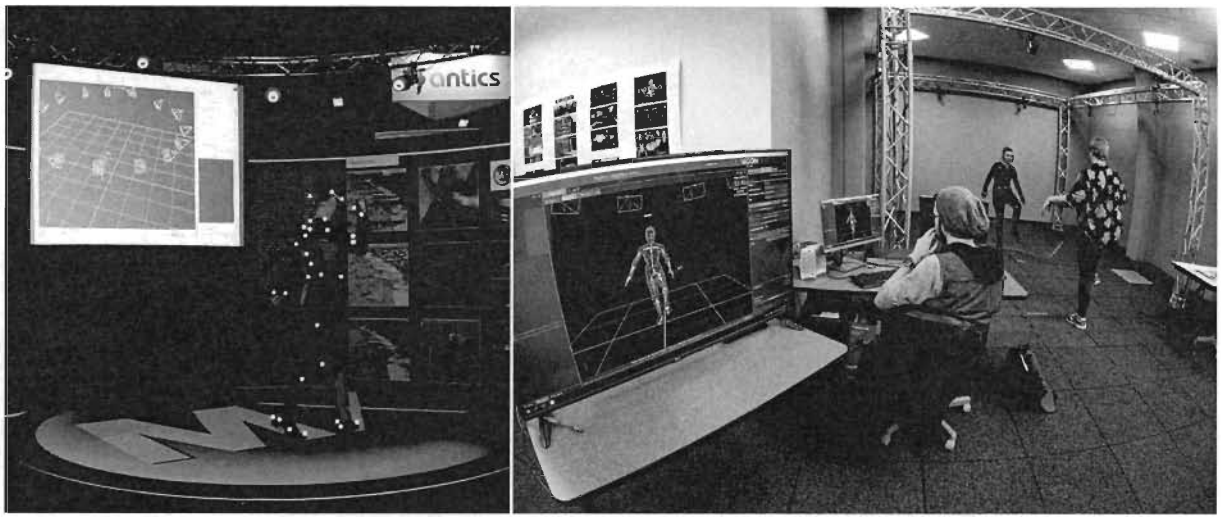


Figure 1.3 Capture de mouvement à l'aide de techniques de reconstruction 3D [https://www.wikiwand.com/en/Motion_capture]

Ces chercheurs reconstruisent en 3D le corps d'acteurs à l'aide d'un système de reconstruction 3D (voir figure 1.3) et utilisent des algorithmes de Mesh tracking pour inférer le mouvement de l'acteur en établissant des correspondances denses par rapport au temps [4] qui à son tour permet de récupérer la configuration commune. Une autre approche de la capture de mouvement consiste à adapter un squelette aux observations (les reconstructions 3D indépendantes) et déduit ainsi la pose de l'acteur [4].

Motion Capture est un exemple d'application dont le but ultime va au-delà de la simple reconstruction d'objets, mais qui utilise les résultats de la reconstruction pour des tâches plus sophistiquées. À ce stade, les applications de capture de mouvement existantes ne fonctionnent pas en temps réel.

1.2 Systèmes de reconstruction 3D en temps réel dans l'agriculture

Contrairement aux domaines d'applications discutés précédemment, l'utilisation de ce genre de technologies est très récente dans le monde agricole. Bien que il y a plusieurs défis intéressants associés à la mise en place d'un système de reconstruction 3D pour des solutions agricoles, les avantages et les possibilités sont énormes et l'emportent facilement sur les défis. Dans ce qui suit, nous examinerons de plus près les deux côtés du problèmes, discuter des avantages et des défis de l'intégration de la reconstruction 3D en temps réel dans les solutions agricoles.

Figure 1.4 Des robots pour éliminer les mauvaises plantes [https://www.rsipvision.com/wp-content/uploads/2018/01/Robots-for-agriculture.jpg]

1.2.1 Avantages

L'intégration d'un système de reconstruction 3D en temps réel dans un robot agricole pour entre autres l'éradication des mauvaises herbes présente plusieurs avantages. En effectuant une reconstruction 3D de la scène, de nombreuses et nouvelles applications surgissent. Par exemple, il devient possible de créer une carte en hauteur (height map) pour que le robot détecte précisément la position d'une mauvaise herbe et la profondeur de chaque composante 3D de cette mauvaise herbes. Une autre application basée sur une reconstruction 3D de la scène est la Cartographie des plantes qui donnera par exemple une carte de toutes les bonnes plantes et ce qui aidera à éviter ces dernières lors des d'éradication des mauvaises herbes.

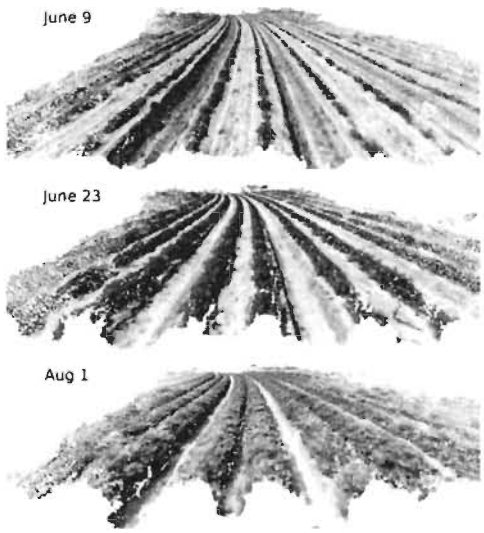


Figure 1.5 Reconstruire un modèle 3D d'un champ d'arachides.

1.2.2 Défis

Bien que les horizons des nouvelles applications offertes par les systèmes de reconstruction 3D en temps réel soient illimités, le secteur agricole pose également plusieurs défis a ces systèmes. La plupart des systèmes existants

fonctionnent dans un environnement de laboratoire dont les conditions sont favorables, éclairage contrôlé et avec peu d'occlusion et d'encombrement. Dans des environnements comme le nôtre (Quebec) ces conditions ne peuvent pas être contrôlées. Le fond a souvent une couleur similaire à celle des plantes ou des herbes que nous souhaitons reconstruire et encombré par d'autres herbes et plantes indésirables. Les conditions d'éclairage peuvent également changer brusquement, comme nous le verrons dans la suite de ce mémoire, la prise en compte de ces conditions est une tâche difficile.

1.3 Contribution

La principale contribution de cette partie de ce mémoire est la conception d'un système de reconstruction 3D en temps réel pour un robot agricole. Sur le plan technique, notre algorithme de reconstruction 3D est présenté, qui est une méthode efficace GPU-based et une méthode incrémentale dans les scènes dynamiques. La liste suivante détaille une fois de plus la contribution apportée dans cette partie du mémoire.

- Conception et mise en œuvre d'un système de reconstruction 3D en temps réel ciblé système robotique agricole.
- Développement d'un algorithme rapide GPU-based pour la reconstruction 3D.
- Discussion sur les défis de l'introduction d'un système de reconstruction 3D dans l'agriculture et proposition des solutions possibles

1.4 Plan

Le reste du mémoire est structuré comme suit. Dans le chapitre 2, nous discutons des différentes techniques de reconstruction 3D et des systèmes multi-caméras. Nous nous concentrons en particulier sur les techniques qui permettent des performances en temps réel. Notre système de reconstruction 3D proposé est présenté au chapitre 3. Nous abordons le chapitre 4 en résumant et en discutant les résultats obtenus de notre système, et nous conclurons avec le chapitre 5 en discutant des défis rencontrés dans les systèmes et des solutions futures possibles.

2

Revue de littérature

Il existe un grand nombre de techniques de reconstruction 3D. Malheureusement, la majorité des méthodes ne fonctionnent pas en temps réel donc la plupart des méthodes disponibles ne résolvent pas nécessairement notre problématique. dans ce chapitre nous jetons un coup d'oeil sur toutes les principales méthodes afin de donner notre perspective et nos choix (dans le chapitre 3).

2.1 Techniques de reconstruction 3D

Il existe une grande variété de techniques de reconstruction 3D. Nous pouvons les séparer en deux catégories méthodes actives et passives. Les méthodes actives utilisent des capteurs actifs (comme le laser) pour récupérer la structure 3D de la scène. En revanche, les méthodes passives ne communiquent pas avec la scène, mais utilisez uniquement la lumière émise par la scène (capturé sous forme d'images) pour effectuer une reconstruction. Comme on peut s'y attendre, les méthodes actives fonctionnent mieux en général, fournissent une plus grande precision mais avec une grande complexité.

2.1.1 Méthodes actives

Les méthodes de reconstruction 3D actives incluent le Laser Range scanning, lumière structurée, stéréo photométrique et time-of-flight (TOF) caméra. Dans la section suivante, chaque méthode sera brièvement décrite.

Laser Range Scanner

Le principe des Scanneur laser est d'utiliser un faisceau laser (*Laser beam*) pour échantillonner la scène. Dans sa forme la plus simple, le rayon laser est projeté sur la scène et crée un point unique qui est enregistré par une caméra

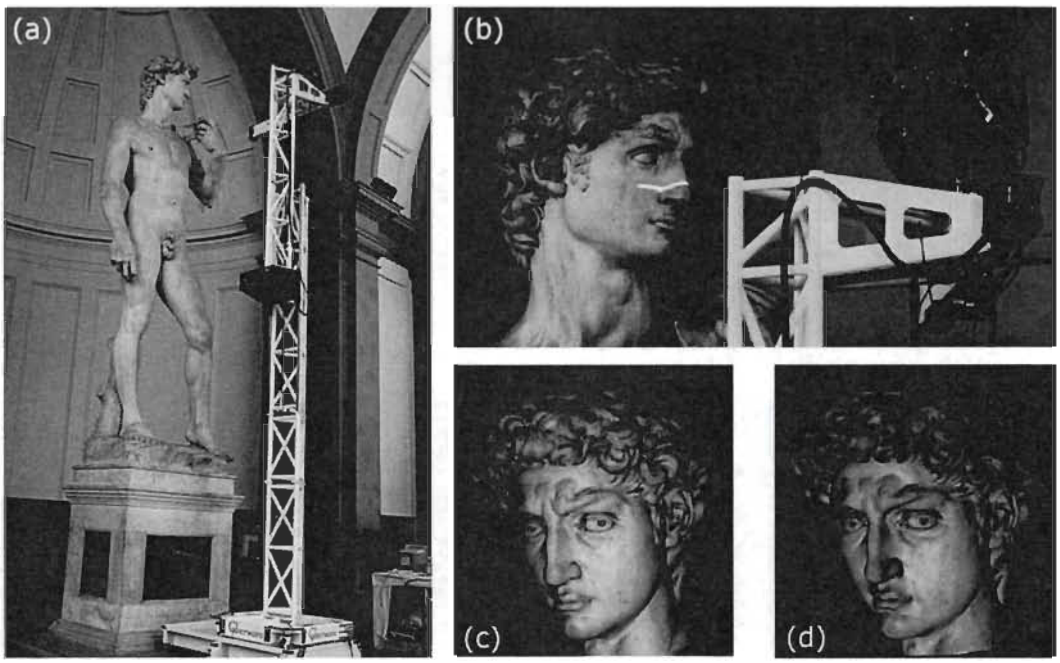


Figure 2.1 Scanneur laser utilisé dans le projet numérique de Michel-Ange [*https://www.researchgate.net/figure/a-Structure-employee-pour-le-scan-du-David-de-Michel-Ange-b-Zoom-sur-le-scanner-3D_fig3_333103024/download*]

calibrée à la source laser. En utilisant la géométrie stéréo du système laser-caméra, la position 3D du point peut être calculée par triangulation. La projection

d'un seul point présente un inconvénient pour la source laser dans le point où la source a besoin de scanner la scène entière point par point, par contre il existe une technique plus efficace pour projeter une ligne comme illustré dans la Figure 2.1. Cela permet au scanner de récupérer une plus grande partie de la scène en un seul balayage. En général, plusieurs analyses (qui devrait être fait ensemble), sont nécessaires pour capturer un objet. La sortie d'un scanner laser dans un Dense nuage de points 3D d'une scène, en utilisant des algorithmes de Maillage de nuage de point peut être converti en représentation sous forme de maille polygonale qui ensuite va être rendu. Les Figure 2.1 c et d montrent un rendu d'un processus de balayage avec un laser. Les scanners laser sont très précis et souvent utilises pour créer des modèles 3D pour des objets réels dans des utilisations différents comme la préservation du patrimoine cultural ou l'industrie cinématographique [5]. Cependant, puisque le laser doit scanner la surface, cette méthode n'est pas une méthode temps réel. De plus, les surfaces réfléchissantes et semi-transparent posent un problème pour cette méthode en raison de leurs propriétés optiques.

Lumière structurée

Le principe de la lumière structurée [6-a] est similaire à celui du scanner laser. Cependant, au lieu de projeter un seul point ou une seule ligne, un pattern lumineux bidimensionnel est projeté sur l'objet. Ce pattern est souvent choisi pour être un ensemble des bandes parallèles. Pour aider à identifier les bandes uniques dans le pattern, des patterns alternés sont projetés formant un code gris pour chaque bande [6-c] (Figure 2.2). En observant la déformation du pattern lumineux, la forme de l'objet peut être déduite, la précision peut être améliorée en effectuant plusieurs acquisitions avec des patterns lumineux légèrement décalés.

La lumière structurée a été utilisée par exemple pour obtenir l'aspect fondamentale pour l'évolution stéréo de middlebury [6-b] qui est à ce jour la norme d'évaluation standard pour les algorithmes de reconstruction (multi-view). Étant donné que la projection et l'acquisition du pattern lumineux peuvent être effectuées assez rapidement, cette méthode est utilisée dans les processus d'inspection et de production industrielle en temps réel. Comme avec les

scanners laser, les surfaces réfléchissantes et semi-transparent peuvent être problématiques.

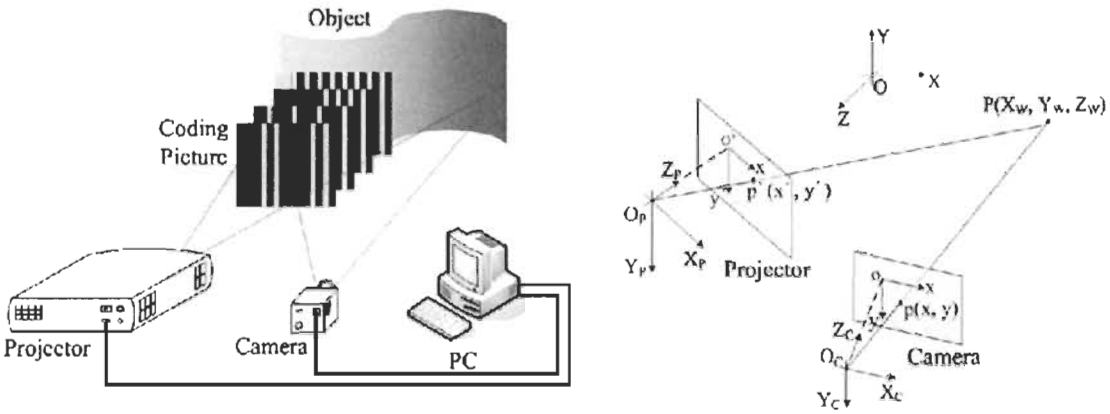


Figure 2.2 Système de caméra de projecteur pour lumière structurée [https://www.researchgate.net/figure/The-principle-of-structured-light-3D-measurement-with-fringe-encoding-a-system_fig1_258295023/download]

Stéréo photométrique

La stéréo photométrique [7-a] est différente des méthodes discutées précédemment elle n'utilise que des informations photométriques pour déterminer la forme d'un objet, tandis que les méthodes actives décrites précédemment utilisent des informations géométriques pour déterminer la surface. L'idée de base derrière la stéréo photométrique est de faire varier la direction de la source de lumière tout en gardant la direction de vision de la caméra constante. Cela crée des intensités différentes à chaque élément de surface en raison des propriétés de réflexion de la surface. Étant donné que la direction de visualisation de la caméra est constante, les pixels sont générés automatiquement et aucune fonctionnalité n'est requise. En utilisant les intensités récupérées à partir de différentes sources de lumière avec un modèle de réflectance de la surface, La moyenne de surface peut être déterminée pour chaque pixel. En integrant les normales, la forme de l'objet sera récupérée.

La stéréo photométrique ne peut pas être utilisée pour récupérer la forme d'un objets en mouvement car il est nécessaire d'acquérir plusieurs images de l'objet

dans une direction de visualisation constante mais avec une direction d'éclairage dynamique. Une solution à ce problème a été proposée par Hernandez [7-b]. Ils utilisent trois sources de lumière de couleurs différentes et séparent les canaux de couleurs dans les images afin d'acquérir simultanément trois images de l'objet sous différentes directions d'éclairage. Cela permet également de capturer des objets en mouvement. La figure 2.3 montre quelques résultats de reconstruction. Les temps de reconstruction qu'ils rapportent se situent dans la plage de 20 secondes par image, ce qui rend cette méthode non compatible en temps réel. De plus, la grande complexité de la configuration et la nécessité de conditions d'éclairage contrôlées avec peu de lumière ambiante rendent cette méthode difficile à utiliser dans un cadre pratique.

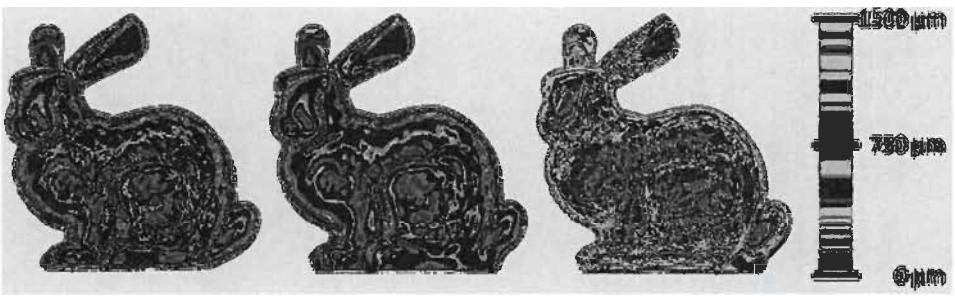


Figure 2.3 Résultats sur une reconstruction photométrique stéréo [https://www.researchgate.net/figure/Results-on-a-photometric-stereo-dataset-In-particular-the-painted-regions-of-the-clay_fig3_262310308]

TOF Caméra

Contrairement aux scanners laser et à la lumière structurée, TOF caméras ne nécessitent pas de configuration complexe et n'éclairent pas la scène dans le spectre visible. d'un autre coté, ils viennent en format pratique et peuvent être utilisés presque rapidement. Ces avantages ont conduit à une adoption rapide de leur utilisation dans la communauté. La caméra produira une carte de profondeur (depth-map) de la scène en envoyant un flash infrarouge modulé. En mesurant le déphasage du flash réfléchissant, la profondeur de la scène par rapport à la caméra peut être calculée. Actuellement, la résolution de la caméra TOF est encore assez limitée et leur précision ne peut être comparée à

celle des scanners laser et de la lumière structurée. Les caméras TOF fonctionnent à des fréquences d'images en temps réel, et il utilise également des propriétés réfléchissantes favorables qui conduisent à des contours partiels dans l'estimation de la profondeur. ceux-ci doivent être corrigés par un post-traitement approprié. La figure 2.4 montre les résultats d'une depth-map traitée obtenue avec une caméra TOF.

Figure 2.4 Caméra TOF et carte de profondeur par la caméra [https://en.wikipedia.org/wiki/Time-of-flight_camera#/media/File:TOF_Kamera_3D_Gesicht.jpg]

2.1.2 Méthodes passives

Les méthodes de reconstruction 3D passives ne scanne pas activement la scène, mais utilisent uniquement des informations photométriques. Cela rend ces méthodes plus générales et faciles à utiliser. Contrairement aux méthodes actives, il existe des cas de problèmes supplémentaires tels que des objets non testés.

Stéréo

La stéréo est l'une des méthodes les plus anciennes en vision par ordinateur pour récupérer la structure 3D d'une scène. En général, deux caméras sont assemblées dans une configuration stéréo. Pour la recuperation de la scène 3D, il y a des correspondance de points entre l'image de gauche et droite pour la

récupération de l'image. Pour simplifier cette tâche les images sont rectifiées, organizer les images de telle manière avec une ligne de balayage enter la gauche et la droite, appelé la disparité et peut être utilise avec la géométrie stéréo pour calculer la profondeur du point. La figure 2.5. donne un exemple tiré de l'évolution stéréo de middlebury [8-a] montrant l'image d'entrée et la Map de disparité.

Figure 2.5 Image d'entrée et reconstruction à partir du dataset Middlebury [https://perso.telecom-paristech.fr/boubek/papers/AMR/]

Le but étant d'obtenir une (*Carte de disparité dense*), chaque pixel de la première image doit correspondre à un pixel dans la deuxième image en supposant qu'il n'y ait pas d'occlusions présentes. La classe simple de la méthode de correspondance utilise une fenêtre locale autour du point d'intérêt dans l'image de gauche et chercher la ligne de correspondence et image globale droite pour trouver la meilleure fenêtre de correspondance [8-b]. En raison de leur simplicité, ces méthodes ne sont pas assez rapides et même pas capables en temps réel. Cependant leur précision est limitée, car ils n'appliquent aucun type de règlements, ce qui les rend problématiques dans les régions de faible texture. D'autres méthodes effectuent la régularisation avec le (*scan lines*) en utilisant les méthodes de programmation dynamique [8-c]. Il existe également des méthodes de découpe graphique (*graph-cut*) et de variation qui considèrent également la régularisation entre les (*scan-lines*) et image global [8-d]. Leurs résultats sont plus précis que les résultats des méthodes locales. Cependant, ils ne sont pas capables en temps réel. En utilisant une seule paire stéréo, on ne peut obtenir

qu'une carte de profondeur de la scène. Afin d'obtenir une reconstruction 3D complète, plusieurs paires stéréo doivent être acquises et leurs cartes de profondeur doivent être fusionnées.

Stereo Multi-vues

Contrairement aux méthodes stéréo traditionnelles, la stéréo à multi-vues utilise un ensemble d'images pour récupérer la structure 3D de la scène. Un aperçu récent de ce domaine est donné par Seitz et al. [9-a]. Souvent, une estimation initiale de la structure de la scène est donnée, cette estimation initiale est ensuite affinée à l'aide de différentes techniques.

Dans la sculpture spatiale, la scène est discrétisée en un voxel pour ca photo-cohérent. Le voxel incohérent est retiré de la reconstruction jusqu'à ce qu'il ne reste que la partie photo-cohérente de la scène. L'inconvénient de cette approche est qu'elle ne contient aucune régularisation et qu'elle est greedy, de sorte qu'un voxel qui a été supprimé une fois ne puisse pas être restauré plus tard. Cela peut conduire à la sculpture de toute la scène, lorsque la cohérence de la photo. Comme toutes les méthodes à gradient décent, les méthodes variationnelles sont locales et il tomber dans des minima.

la méthode est basée sur la coupe graphique [8-d, 9-b] comme les approches variationnelles tentent de trouver la reconstruction photo-cohérente maximale. Cependant, ils sont de nature mondiale et sont donc moins susceptibles de tomber dans les minima locaux, comme tout les méthodes photométrie, une mesure photo précise est nécessaire pour que nous puissions assumer la conception standard (comme que les propriétés de surface lambertiennes) doivent être remplies.

Avec ces méthodes qui déforment une reconstruction, il existe également des méthodes de reconstruction qui commencent par trouver des caractéristiques d'un objet à l'aide de détecteurs de caractéristiques. Ensuite, les correctifs sont développés autour de ces fonctionnalités [9-c]. Un algorithme stéréo à multi-vues sur les paires d'images fusionner les cartes de profondeur [9-d].

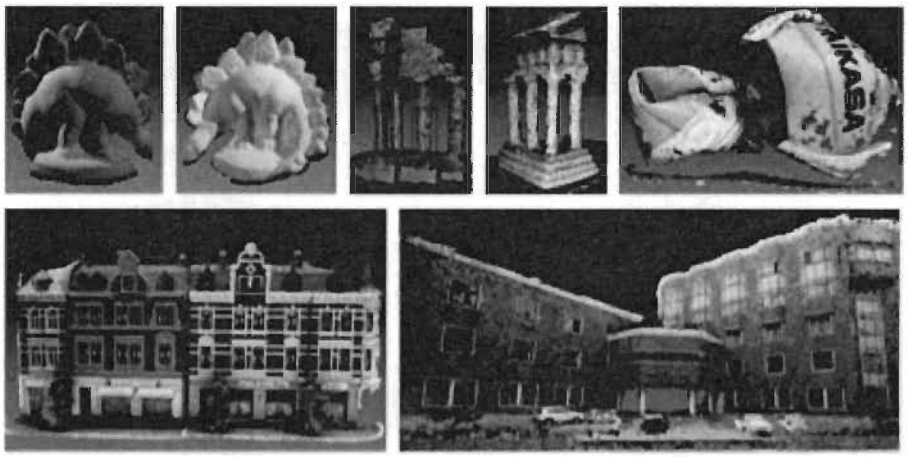


Figure 2.6 Exemples de reconstruction 3D multi-vues [*Article: A Multi-View Stereo Algorithm Based on Homogeneous Direct Spatial Expansion with Improved Reconstruction Accuracy and Completeness*]

Les methods multi-vues stéréo ne sont pas applicables en temps réel mais fournissent les meilleurs résultats possible lorsque des information photométrique sont uniquement utilisées. Quelques résultats de reconstruction sur l'évolution multi-vues sont présentés dans la figure 2.6.

Forme de Silhouette

Contrairement à la méthodes stéréo et à la stéréo multi-vues, les méthodes de forme à partir de la silhouette [10-a, 12-b] n'utilisent pas directement d'informations photométriques sur la scène et en particulier ne se concentrent pas de la cohérence photométrique de la reconstruction. Par contre, ils supposent la disponibilité d'images de silhouette en montrant les objets de premier plan dans la scène et posent le problème de manière purement géométrique comme celle de trouver la forme qui est au maximum cohérente avec les images observées de silhouette [9-a].

En raison de la simplicité du processus de reconstruction, les méthodes de forme a partir de la silhouette sont capable en temps réel. Cependant, ils ont l'inconvénient de nécessiter des images de silhouette, qui ne sont pas toujours disponible. En revanche les objets a texture uniforme ne posent pas de problème

pour cette méthode tant que nous pouvons récupèrer les images silhouette (e.g. par soustraction de fond). Cela rend les approches *shape-from-silhouette* interessantes pour les systèmes en temps réel qui nécessitent plus de complexité.

2.2 Système de reconstruction multi-caméras

Au cours de la derrière décennie, de nombreux systèmes de reconstruction 3D ont été proposés, la plupart d'entre eux utilisant le concept de voxels. Bien que les premiers systèmes n'aient pas atteint les performance en temps réel en raison de la technologie limitée de leur temps.

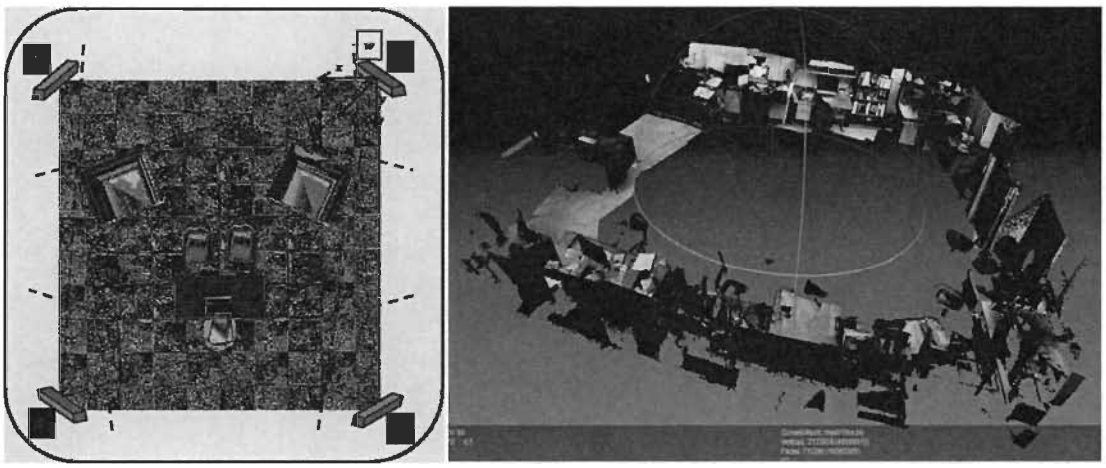


Figure 2.7 Architecture et résultats du studio de caméra à vues multiples [*https://www.researchgate.net/figure/RGB-D-Multi-View-System-with-full-scene-3D-reconstruction-in-a-simulated-setup-a-RGB-D_fig1_333393581*]

Un premier système construit vers 1995 qui utilisant la stereo [10-b] composé par 51 cameras installées sur un dôme de 5 metres de diamètre (Figure 2.8). Les cameras utilisées étaient de type analogique avec des objectifs de 3,6 mm afin d'obtenir un champ de vision de près de 90 degrés. Cela a entraîné un chevauchement 3m x 3m x 2m en taille. Les cameras ont été synchronisées et un signal de synchronisation en mode vidéo. Video et enregistrer sur des bandes S-VHS pour une numérisation et un traitement ultérieurs. Ce système n'était pas en temps réel.

En 1998, la salle 3D a été construite a CMU [10-b]. En utilisant l'acquisition d'images numériques a l'aide d'une caméras 49 couleurs. Cependant, en raison de la taille limite de la mémoire, la durée d'enregistrement était limitée a environ 800 images, soit environ 18 seconde a 15 ips.

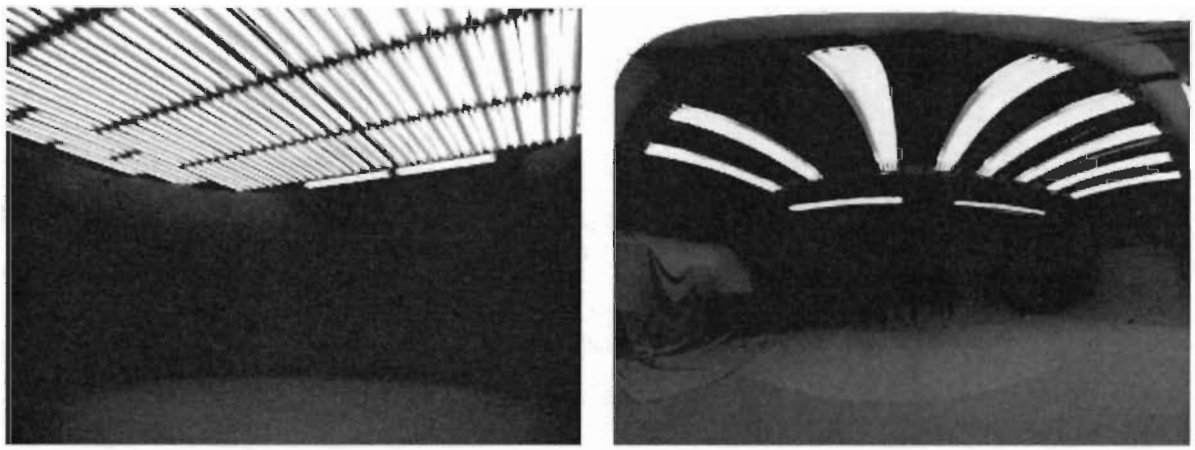


Figure 2.8 Systèmes à multi-vues à Surrey, Kyoto [*https://www.semanticscholar.org/paper/The-Multiple-Camera-3-D-Production-Studio-Starck-Maki/0c16e42ebe28ae2fea011a2f1df67e1319d044cf/figure/0*]

A l'université de Kyoto, un système composé de 15 caméras a été développé [10-c]. Cette méthode applique une approche basée sur l'algorithme visual hull. Le visual hull est amélioré constamment en déformant la maille polygonale récupère en fonction de la coherence photo et la force silhouette [10-c]. Wu *et al.* [10-d] ont utilise le système avec un test d'intersection dans le plan pour la reconstruction de Visual Hull. Ils rapportent des valeurs de 12 IPS avec 9 caméras. Ce système utilise également un éclairage contrôlé et un fond uniforme afin de simplifier la soustraction du fond (figure 2.8).

2.3 Concept de voxels dans la reconstruction 3D

Les données volumétriques sont généralement données par des valeurs scalaires, ces valeurs sont attribuées à des sommets avec une grille de voxels régulière [11-

a]. Afin de visualiser et de traiter efficacement les informations géométrique qui sont implicitement représentées par un ensemble de données volumétrique, nous devons souvent extraire des informations de surface explicites. Il existe deux concepts fondamentalement différent pour accéder et modifier les informations de surface représentées par un ensemble de données volumétriques.

Le premier concept est basé sur la topologie du réseau voxels et ignore principalement les données scalaires associées aux voxels [11-a]. Étant donné que le voisinage de chaque voxel est régulier, nous ne prendrons pas en compte les cas particuliers topologiques. Les opérateurs morphologiques bien connus sont un exemple d'opérateurs qui utilisent des voisinages de réseau grille pour determiner le statut d'un voxel [11-a].

Le deuxième concept considère l'ensemble des dogmes volumétriques comme une fonction à valeurs scalaires définies dans un espace continue à 3-axes en interpolant les valeurs de la grille [11-a]. Grâce au théorèmes de la fonction implicite, ce champ scalaire définit la géométrie locale de toutes ses surfaces [11-a]. Le calcul de dérivée du champ scalaire nous donne le vecteur de gradient qui est le vecteur normal a la surface [11-a]. Le gradient apparait parfois sous forme déguisée, lorsque nous transformons une surface en une autre surface S ($v \pm \varepsilon$) nous, en principe, déplaçons tous les points de surface dans la direction du gradient.

$$S(v) = \{ (x,y,z) \mid f(x,y,z) \} = v$$

2.3.1 Topologie (Les Concepts Mathématique)

Nos éléments de base $s \in \mathbb{Z}^3$ sont appelés voxels. Les voxels peuvent être considérés comme des points d'échantillonnage uniformes d'un champ scalaire ou comme des cellules cubiques unitaires centrées sur la grille entière. Les ensembles $S \subset \mathbb{Z}^3$ sont appelés les ensembles de voxels. Nous supposerons que tous les ensembles de voxels sont bornés, Les voxels $s \in S$ sont appelés solides, les voxels en $\mathbb{Z}^3 \setminus S$ sont appelés vides. Deux voxels solides $s, t \in S$ sont appelés

voisins $s \sim t$, si $\| s - t \|_\infty = 1$, deux voxels vides $a, b \in \mathbb{Z}^3 \setminus S$ sont appelés voisins, $a \sim_E b$ si $|a - b|_1 = 1$.

Comme chaque voxel solide peut avoir jusqu'à 26 voisins $\sim_s$ est souvent appelé la relation de 26-voisinage. Analogiquement $\sim_t$ est appelé la relation des 6-voisinage (Figure 2.9).

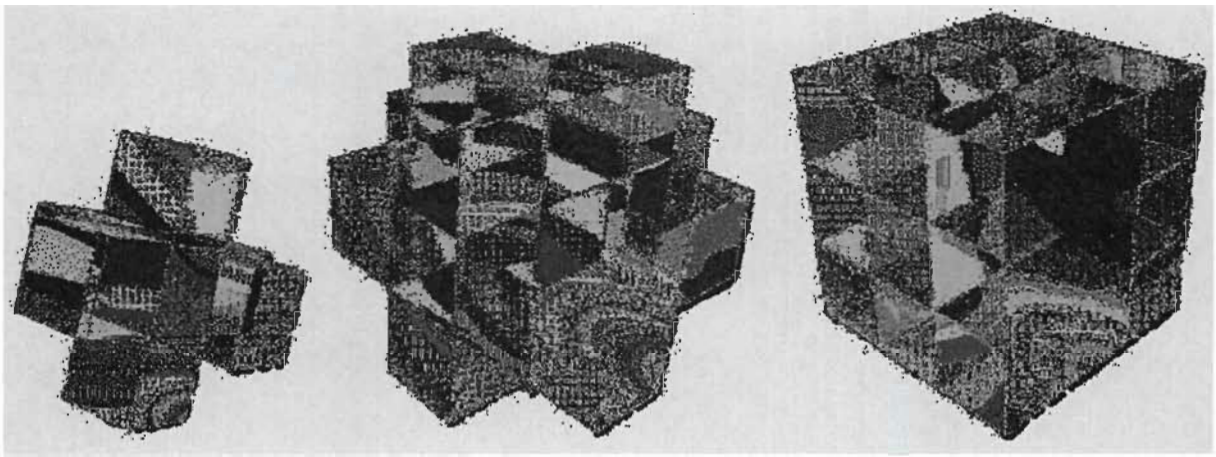


Figure 2.9 Droite: Voxel avec 26 voisins. milieu: Voxel avec 18 voisins. Gauche: Voxel avec 6 voisins [*https://www.researchgate.net/figure/Representation-of-the-initial-seed-voxel-and-its-26-neighbors_fig8_255737531*]

nous considérons $\varepsilon \geq 0$. nous définissons le $\varepsilon - embedding$ $\Re_\varepsilon(s) \subset \mathbb{R}^3$ d'un voxel solide s comme :

$$\Re_\varepsilon(s) = \left\{ x \epsilon \mathbb{R}^3 : \|x - s\|_\infty \leq \frac{1}{2} + \varepsilon \right\}$$

Et le $\varepsilon - embedding$ $\Re_\varepsilon(s) \subset \mathbb{R}^3$ de S en tant que l'union

$$\Re_\varepsilon(s) = \bigcup_{s \epsilon S} \Re(s)$$

Intuitivement, cela signifie que nous centrons un cube de longueur d'arête $1 + 2_\varepsilon$ à chaque voxel solide $s \epsilon S$. Cette définition de $\Re_\varepsilon$ correspond bien à la définition de la relation de voisinage $\sim_s$, depuis

$$S \sim_s t \Leftrightarrow \Re_\varepsilon(s) \cap Re_\varepsilon(t) \neq \phi.$$

Cela montre en particulier que $\sim_s$ et $\sim_e$ sont compatibles. Chaque sommet est assigné de manière non ambiguë pour être soit solide ou vide [11-a]. Enfin, nous associons un 2-collecteur $\delta_\varepsilon S$ à chaque ensemble de voxels S en faisant

$$\delta_\varepsilon S := \delta \Re_\varepsilon(S).$$

Si 0<ε<0,5 la surface $\delta_\varepsilon S$ est régulière à 2 collecteurs, en particulier, il n'y a pas de point singulier ni d'arête singulière.

2.3.2 Caractéristique d'Euler

Dans ce qui suit, nous associons une caractéristique d'Euler à un ensemble de voxels S. Tout d'abord, nous notons qu'un seul voxel $\Re(s)$ peut être examiné comme un complexe de cellules finies $C(s)$ composé de 1 cube (3-cell), 6 quadrangles (2-cells), 12 bords (1-cells) et 8 vertices (0-cells). [11-a]

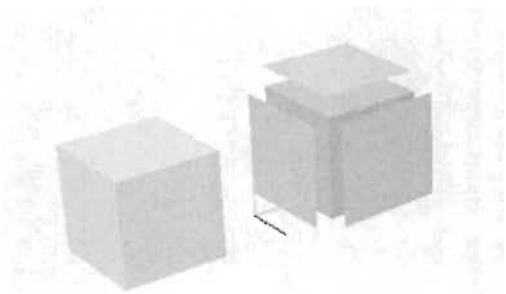


Figure 2.10 Décomposition d'un Voxel en un complexe cellulaire [*cette figure est générée à l'aide du logiciel Sketch*]

De façon analogue, nous pouvons représenter $\Re(s)$ comme un complexe de cellules finies $C(s)$ Défini par

$$C(s) = \bigcup_{s \epsilon S} C(s) .$$

Il est bien connu que la caractéristique d'Euler χ d'un complexe cellulaire C est le nombre de cellules de dimensions paires moins le nombre de cellules de dimensions impaires [11-a]. nous définissons :

$$\chi(s) := n_0 - n_1 + n_2 - n_3 .$$

Où n_i est le nombre de i-cellules (i = 0,1,2,3) de $C(S)$ des exemples de ce calcul sont présentés dans la figure 3.1.

Pour calculer efficacement la caractéristique d'Euler de l'ensemble de voxels S, nous exploitons le fait, que χ est additif

$$\chi(S) = \sum_{a \epsilon \mathbb{Z}^3} \chi(S \cap (a + [0,1]^3)) .$$

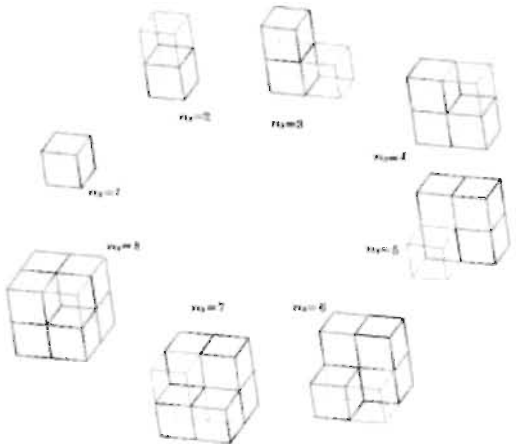


Figure 2.11 Calcul de la caractéristique d'Euler d'un ensemble de voxels. [*https://www.elsevier.es/en-revista-journal-applied-research-technology-jart-81-articulo-the-euler-poincare-formula-through-contact-S1665642313715153*]

Chaque terme $\chi(S \cap (a + [0, 1]^3))$ dans la somme ci-dessus est uniquement déterminé par le statut des 8 Voxels aux positions a, a+(1, 0, 0), ….., a+(1, 1, 1) et peut facilement être calculé dans une table de recherche de taille 256. Si une information importante car elle nous permet de detecter efficacement les changement topologique causés par la progression d'un voxel à une valeur solide [11-a]. Alternativement, la configuration locale autour d'un voxel peut être codée dans un diagramme de décision binaire [11-a].

2.4 Apprentissage automatique dans la reconstruction 3D

L'apprentissage automatique est le présent et l'avenir de tout domaine technologique en ce moment, comme notre problème est très intéressant, nous allons utiliser des méthodes d'apprentissage automatique (ML) pour améliorer la reconstruction 3D de notre système.

Récupérer la dimension perdue à partir d'images 2D uniquement, a été l'objectif des méthodes stéréo multi-vues classiques et des shape-from-X. L'avenue des techniques deep learning avec l'augmentation des grands ensembles de données et datasets, il nous donne une meilleure chance de perfectionner la méthode de reconstruction.

2.4.1 Taxonomie du problème

Pour cela, nous allons utiliser la classification proposée dans le travail de Xian-Feng Han*, Hamid Laga*, Mohammed Bennamoun Senior Member [11-c], qui font une bonne étude des techniques d'apprentissage automatique utilisées dans un tel problème.

Input	Training	1 vs muli RGB, 3D ground truth, Segmentation	One vs multiple objects, Uniform vs cluttered Background.
	Testing	1 vs muli RGB, Segmentation	
Output	Volumetric	High vs Low resolution	
	Surface	Parameterization, temple deformation, Point Cloud	
		Direct vs Intermediating	
Network Architecture	Architecture at training		Architecture at testing
	Encoder - Decoder TL-Net (Conditinal) GAN		Encoder - Decoder
	3D-VAE-GAN		3D-VAE
Training	Degree of Supervision		2D vs 3D supervision. Weak supervision
			Loss Function
	Training Procedure		Adversarial training, Joint 2D-3D embedding Joint training with other tasks.

Figure 2.12 Taxonomie de la reconstruction 3D en utilisant le deep learning. [*Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era by Xian-Feng Han*, Hamid Laga*, Mohammed Bennamoun Senior Member, IEEE*]

La représentation finale de la reconstruction est très cruciale pour le choix de l'architecture du réseau. Il a également un impact sur l'efficacité informatique et la qualité de la reconstruction. En particulier,

- Représentation volumétrique, qui a été largement adoptée dans les premières techniques de reconstruction 3D basées sur l'apprentissage profond, pour permettre le paramétrage des formes 3D à l'aide de réseau de voxels réguliers [11-c]. Ainsi, les convolutions 2D utilisées dans l'analyse d'images peuvent être facilement étendues à la 3D. Ils sont cependant très coûteux en termes d'utilisation de mémoire, et seules quelques techniques peuvent atteindre une précision de niveau (sub-voxels) [11-c].

- Représentation basée sur la surface : d'autres chercheurs ont exploré les représentations basées sur la surface telles que les mailles polygonales et les nuages des points [11-c]. Avec une optimisation de memoire, les représentations ne sont pas des structures régulières et par conséquent, elles ne s'intègrent pas facilement dans les architectures du deep learning.

- Intermédiation : tandis que certains algorithmes de reconstruction 3D prédisent directement la géométrie 3D d'un objet à partir d'images RGB [11-c], d'autres méthodes décomposent le problème en étapes séquentielles, chaque étape prédit une représentation intermédiaire.

Diverses architectures de réseau ont été utilisées pour implémenter le prédicteur f.

L'architecture qui peut être différente lors de la formation et des tests, est composée d'un encodeur h suivi d'un décodeur g, $f = g \circ h$.

L'encodeur mappe l'entrée en une variable x, rappelé vecteur d'entité ou code, utilisant une séquence de convolutions et une opération de mise en commun, suivi de couches de neurones entièrement connectées. Le décodeur également appelé le générateur, décode le vecteur d'entité dans la sortie souhaitée en utilisant des couches entièrement connectées ou un réseau de déconvolution (une séquence d'opérations de convolution et de suréchantillonnage, également appelées upconvolutions) [11-c]. Le premier convient pour la sortie non structurée, en nuage de points, tandis que le second est utilisé pour reconstruire des réseaux volumétriques ou des paramètres de surface [11-c]. Depuis l'introduction de cette architecture, plusieurs extensions ont été proposées en faisant varier l'architecture (ConvNet vs ResNet, Convolutional Neural Networks (CNN) vs. Generative Adversarial Networks (GAN), CNN vs. Variational Auto-

Encoders, et 2D vs. 3D convolutions), et en cascadant plusieurs blocs chacun réalisant une tâche spécifique.

Bien que l'architecture du réseau et ses éléments soient importants, la performance dépend fortement de la façon dont elle est entraînée.

2.5 Conclusion

Il existe de nombreuses techniques de reconstruction 3D. Cependant, seules la lumière structurée, les caméras TOF et RealSense et la forme de la silhouette sont capables d'exécution en temps réel. L'inconvenient des méthodes d'éclairage structure est l'utilisation des motifs lumineux dans le spectre. Les cameras TOF sont très efficaces mais la technologie et encoure immatures en termes de resolution et de précision. Les méthodes Shape-from-silhouette d'autre part, discrète et robuste, ne nécessitant que des images de silhouette qui peuvent être obtenues par soustraction d'arrière-plan même pour un objet non texturé. La technologie RealSense utilise la détection de profondeur stéréoscopique pour déterminer la portée d'un élément. Donc, essentiellement, il a deux caméras et peut en faire la triangulation pour la stéréo. Ce capteur utilise deux caméras infrarouges pour la stéréo et dispose également d'une caméra RGB à bord. Cependant, pour la plupart des systèmes, c'est la bonne méthode pour les systèmes de reconstruction en temps réel actuels. Si une reconstruction plus précise est souhaitée, les modèles d'apprentissage automatique (ML) aideront à donner des résultats beaucoup plus précis.

3

Implémentation

3.1 Notations

Il existe de nombreux algorithmes pour la reconstruction 3D a partir d'une scène avec des points non-organisés. On peut les classer généralement sur la base de géométrie algorithmique, méthodes algébrique et implicites. Les méthodes de géométrie algorithmique utilisent des mécanismes tels que la triangulation de Delaunay [3, 10] et region par croissance [5, 11] pour construire un ensemble de triangles dont les sommets sont les points d'origine. Ces méthodes ont tendance a bien fonctionner sur un ensemble de données denses et propres et capables de reconstruire des surfaces avec des limites. Par contre, elles sont sensibles au bruit et ont tendance a laisser des trous sous les regions échantillonnées. Les méthodes algébriques [19, 21] s'adapter a des surfaces lisses. La plupart de ces méthodes ne peuvent pas gérer une topologie arbitraire, ni reconstruire une géométrie complexe.

L'utilisation de fonctions implicites est l'approche la plus populaire pour la reconstruction de surface a partir d'ensembles de points non organises [9, 13, 17, 18, 23]. Ces méthodes sont basées sur l'utilisation d'une fonction de distance signée, qui permet la representation de différentes topologies. La reconstruction 3D de la surface est assez simple, généralement effectuée a l'aide d'une variation des algorithmes Marching Cubes [16]. La principale limitation de ces méthodes

est leur incapacité a reconstruire des surfaces non multiple. Selon l'article de Boolmenthal et al [7], Les limites sont généralement spécifiées via des fonctions supplémentaires qui devrait avoir le même signe. Boolmenthal et Ferguson [8] décrire un algorithme de polygonisation de surfaces implicites non variées défines par plusieurs régions de l'espace. L'algorithme peut gérer des surfaces avec des limites et des intersections, mais l'existence de plusieurs regions ajoute considérablement a la complexité du polygoniseur.

Ohtake et autres [18] ont démontré une méthode de reconstruction de surface hiérarchique intéressant et flexible basée sur le mélange de fonctions implicites locales. Leur approche gère de très grands ensembles de données, peut reconstruire des entités nettes et fournit un support pour les operations CSG et de morphing.

Cependant, il ne peut pas être utilisé pour reconstruire des surfaces non multiples, ce qui est le cas dans notre système.

Adamson et Alexa [1] présentent un algorithme basé sur des points pour le rendu (non reconstruit) surfaces avec limites et surfaces non orientables. Dans leur approche, un ensemble dense de points définit une surface implicite, qui est identique aux moindres carrés mobiles (MLS) approximation de la surface [2]. La technique ne peut être utilisée que pour le rendu de surface de collecteur et basée sur le lancer de rayons. Une surface avec limite est défini localement comme l'ensemble si de points x (sur la surface implicite), ces distances a une position moyenne pondérée dans un quartier Ω cela contient sachant que x est inférieur à un seuil spécifié par l'utilisateur. Ainsi, les régions peu échantillonnées sont rendues sous forme de trous. contrairement à [1], notre technique reconstruit en fait une représentation polygonale pour ces surfaces.

L'approche Azernikov [4] construit un graphe de connectivité, puis crée des facettes en parcourant le graphe et en trouvant des boucles minimales. Dans le Mesh résultant, chaque voxel produit un sommet, calculé comme le centroïde des points d'entrée à l'intérieur du voxel associé.

L'algorithme ne peut pas garantir que la surface du voxel a une topologie locale bien définie. Par conséquent, un certain nombre d'heuristique sont utilisées pour assurer l'exactitude de la maille polygonale resultant. Il suppose également que

son entrée consiste en un Nuage de points a échantillonnage dense et ne comble pas les lacunes. Il existe trois differences majeures entre notre stratégie de mesh modifiée et l'algorithme Azernikov :

1. Nous ne trouvons qu'un graphe de connectivité pour les voxels de surface et de bordure;
2. Notre approche gère les surfaces non multiples;
3. Parce que certains voxels octree correspondant a des trous doivent être incorporels dans la surface du voxel (afin de les remplir automatiquement), nous utilisons une méthode intuitive pour identifier ces voxels.

3.2 Algorithme de reconstruction

Notre algorithme de reconstruction de surface construit un *Mesh polygonal* à partir d'un *Nuage de points* en utilisant des voxels comme représentation intermédiaire. Les étapes de l'algorithme sont présentées comme suivants:

1. *Voxélisation et remplissage des trous;*
2. *Amincissement topologique;*
3. *Maillage.*

3.2.1 Voxélisation et remplissage des trous

L'algorithme commence par calculer une boite de delimitation étroite pour le *Nuage de points*, qui est ensuite subdivisée en voxels. La taille du voxel doit être suffisamment petite pour éviter de fusionner des caractéristiques de surface distinctes, mais suffisamment grand pour éviter les calculs inutiles. Actuellement la voxélisation s'effectue en fonction d'une toile de voxel spécifiée au début. Les voxels contenant des points originaux sont appelés *p-voxels*.

Quelques voxels vides (ne contenant pas des points) se situant entre les *p-voxels* et représentant des parcelles de surfaces non échantillonnés sont appelés *g-voxels*. *Les g-voxels seront ensuite utilises pour combler les trous*. Nous appelons les voxels de premier plan (Sea) défini par l'union des *p-voxels* et des *g-voxels*. Les voxels de premier plan sont utilisés pour l'éclaircissement topologique. Parce qu'ils n'occupent aucune petite partie de l'espace, nous utilisons une structure de données de décalcomanie [12] pour les stocker. Le complément du voxels de foreground avec la boite de connection est appelé (Background Voxels)

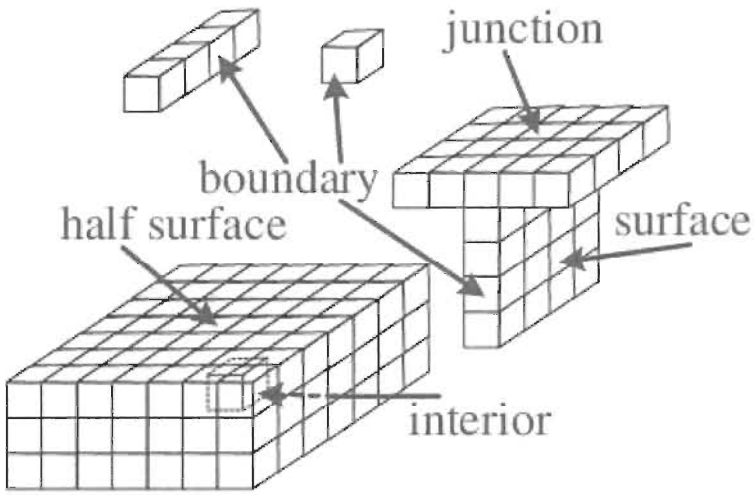


Figure 3.1 *Classification topologique des voxels* [04]

3.2.2 Amincissement topologique

Nous sommes intéressés par la reconstruction des (junctions) du surfaces et de scènes, en préservant bien sûr les limites des surfaces. Le type topologique local d'un voxel est déterminé par le nombre de composants connectés au premier plan et a l'arrière-plan dans son voisinage (comme nous l'avons vu au chapitre 2). Un voxel est appelé voxel de junction s'il se trouve a la junction de deux surfaces ou plus. De même, un voxel est appelé voxel limite s'il est a la limite d'une surface, le long d'une courbe, ou juste un voxel isolé. Les voxels a demi-surface sont un côté d'un voxel épais. Les voxels a demi-surface sont sur un côté d'une surface de voxel épaisse et sont les seuls a être supprimés lors de processus

de amincissement topologique. L'existence de surfaces de voxels épaisses est généralement associée à la présence de bruit dans l'ensemble de données.

En 2007, G. Bertrand et M. Couprie [12-c] ont proposé un amincissement séquentiel (sequential thinning) basé sur les isthmes (definition: Langue de terre qui sépare deux mers et relie deux terres). La strategy utilisée par Bertrand et consiste a detecter dynamiquement les ishmes considérés (1D est 2D) et les accumuler dans un ensemble de contraintes. La suppression d'un point de celui-ci est simple et pas dans l'ensemble de contraintes. De cette façon, un point détecté comme un isthme dans une itération donnée ne peut pas être supprime plus trad. Selon les isthmes considérés, la methods fournit des squelettes curviligne (dans le cas de 1D isthmuses) ou des squelettes de surface (dans le cas de 2D isthmuses).

La prise en compte des isthmus au lieu des points d'extrémités est intéressante pour deux raisons : les isthmes peuvent être facilement définis et détectés localement (ce n'est pas le cas des extrémités de surface), et apparaissent moins souvent que les extrémités pendant le processus d'amincissement, conduisant à des squelettes moins bruités. Cependant, l'approche séquentielle est moins adaptée que les approches parallèles à cet effet, et fournit des squelettes de qualité inférieure.

Nous proposons un algorithme de amincissement a 6 direction produisant des squelettes, chaque iteration de l'algorithme se compose de 6 sous-iterations dans lesquelles des points sont supprimés de la forme si et seulement si la configuration des voisinnage (26-voisinage) est compatible (vu au chapter 02) au moins un masque de l'ensemble de masques M_d pour la direction donnée d.

L'ensemble de masques M_U utilise pour UP les directions sont présentées dans la figure 3.1. l'ensemble de masques M_D,M_N, M_S, M_E, M_W pour les autres directions (EN_BAS, NORD, SUD, EST and OUEST) sont obtenus par des rotations et reflexions appropriées. L'algorithme est le suivant :

Data: $X \subseteq \mathbb{Z}^3$

Result: A skeleton of X

```
Y = X

repeat

      foreach direction d in (U,D,N,S,E,W) do

            Y = Y \ {p∈Y : p matching one mask of M_d}

Until stability ;

Return Y
```

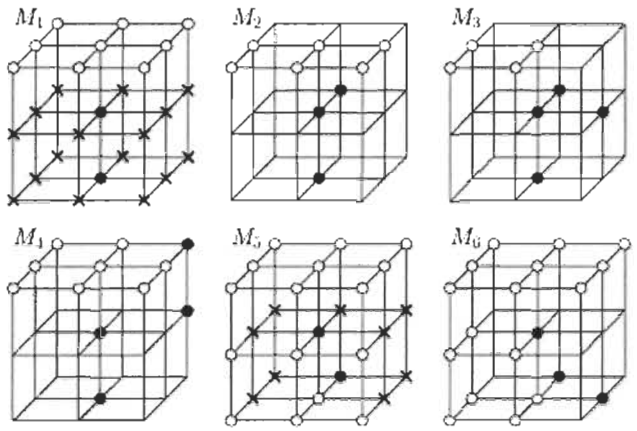


Figure 3.2 Les masques de base M1-M6 et leurs rotations autour de l'axe vertical forment l'ensemble des masques MU pour la direction UP. Les points marqués par un disque noir doivent appartenir à l'objet, les points marqués par un disque blanc doivent être hors de l'objet, et au moins un point marqué par une croix dans les masques M1 et M5 doit appartenir à l'objet. [12-c]

D'autre part, nous pouvons améliorer ces méthodes en séparant l'algorithme en deux étapes, en séparant le processus de d'amincissement (en utilisant l'approche a 6-directions). Chaque iteration se compose de :

1. Mise a jour de l'ensemble de contraintes, en ajoutant des points de l'objet détecté;

2. Suppression de points supprimantes de l'objet qui ne sont pas dans le jeu de contraintes.

Data: $X \subseteq \mathbb{Z}^3$, $K \subseteq X$

Result: A skeleton of X

Y = X

repeat

$K = K \cup \psi_{1D}(X)$

foreach direction d in (U,D,N,S,E,W) do

$X = \tau_d(X, K)$

Until stability ;

Return Y

Cette conception permet de conserver certains points définis à priori par l'ensemble initial K (qui peut être vide). La fonction $\psi_{1D}(X)$ renvoie l'ensemble de toutes les 1D de X. La fonction $\tau_d(X, K)$ renvoie l'ensemble X \ S, S étant l'ensemble de tous les points de X \ K concernant les conditions de suppression X pour la direction d.

Masques de condition de suppression Contrairement à d'autres algorithmes trouvés dans la littérature [10-b, 11-a, 14], ces τ_d permettre la suppression des points d'extrémité de courbes, les points de contrainte étant considérés séparément.

Les conditions de suppression peuvent être représentées à l'aide d'un ensemble de masques: un point respecte la condition de suppression si et seulement il correspond à au moins un des masques (voir figure 3.3).

Pour faire une séparation géométrique et topologique, nous devons autoriser τ_d la suppression des points de fin (pointe avec un seul voisin dans l'objet). Dans ce but, nous ajoutons ce masques M_d, représentant des points de fin qui peuvent être supprimés pour la direction d. Ces masques sont représente dans Figure 3.3 pour le cas d = U.

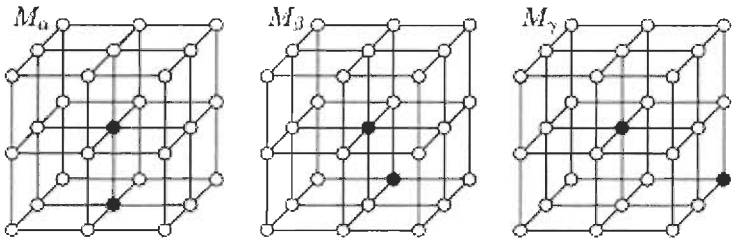


Figure 3.3 Masques Mα-Mγ et leurs rotations autour de l'axe vertical sont ajoutées à l'ensemble de masques MU pour la direction UP afin de supprimer les points de fin.

Réduction de l'ensemble de masques L'ensemble de masques $M_U \cup M_\alpha, M_\beta, M_\gamma$ peut être compacté comme dans la figure 3.4 on peut observer que le masque M_1' qui correspond exactement aux mêmes configurations que celles associées aux masques M_1 est M_α. De la même manière, le masque M_5' correspond exactement a la meme configuration que celles correspondant au masque M_5 est M_β. Les masques M_2', M_3', M_4', M_6' est M_7' sont respectivement les mêmes que M_2, M_3, M_4, M_6 et M_7.

Utilisation des ensembles M_d' avec $d \epsilon U, D, N, S, E, W$, on peut maintenant définir T_d, comme nous l'avons proposé dans l'algorithme :

```
Data: X ⊆ ℤ³, K ⊆ X
Result: A subset of X
R = set of points matching a mask belonging to M'_d
R = R \ K
Return X \ R
```

3.2.3 Maillage

Le maillage est effectué en utilisant une variation de l'algorithme présente par Azemikov et al [13-a]. Étant donné le type topologique local de tous les voxels D_{pg} est connu, les étapes de l'algorithme Azemikov's nécessaires pour éliminer « fausses facettes» (facettes pouvant apparaitre dans le graphique de connectivité mais n'appartenant pas à la surface d'origine). Pour le cas des surfaces, cette version plus simple de l'algorithme peut être utilisée. De plus l'algorithme de Azernikov's ne gère pas tous les surfaces, nous avons modifie l'algorithme d'origine en permettant a une surface d'être composée de plusieurs sous-surfaces connectées au niveau des pixels.

Étant donné deux surfaces Pi et Pj, leur intersection définit un ensemble de voxels de junction. Les voxels de surface touchant les voxels de junction sont appelés voxels de bordure. Un ensemble de voxels de bordure d'une surface définit une courbe de bordure pour la surface. On appelle courbe d'intersection la courbe définie par l'ensemble des voxels de junction.

Étant donné une intersection entre deux surfaces P_i et P_j, il existe trois types de courbe de bordure possible, comme illustré en Figure 3.5.

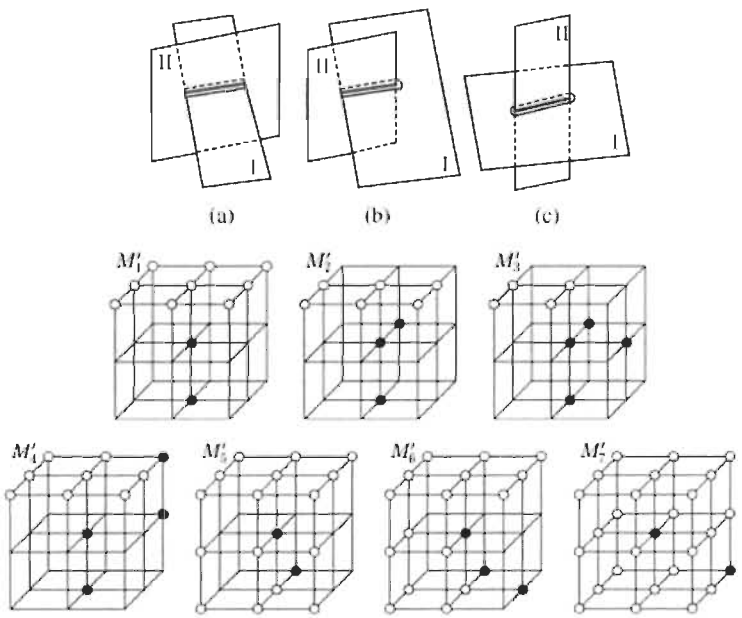


Figure 3.4 Trois types de courbes de bordure dans les intersections de surface. [13-b]

Pour le cas de la surfaceI : deux courbes de bordure déconnectées (Figure 3.5-a), une courbe de bordure ouverte (Figure 3.5-b), et une courbe de bordure fermée ((Figure 3.5-c). Pour les cas (b) et (c), chaque surface se compose toujours d'un seul composant connecté. Le maillage peut ensuite être effectué après que les voxels de junction sont connectés aux voxels de la courbe de bordure, et l'ensemble de voxels resultant est soumis a une operation d'éclaircie [13-b].

Le processus de regroupement commence par l'identification de tous les composants et, pour chacun, création d'une maille à l'aide de la version simplifiée de l'algorithme de Azernikov's.

Une fois ces mailles créées, un vecteur normal est calculé pour chaque voxel de bordure en faisant la moyenne des normales de toutes les faces partageant ce voxel particulier (voir figure 3.6). On commence avec un voxel dans la courbe d'intersection et on continue avec la courbe. Soit v_j le voxel de junction actuel dans la courbe d'intersection.

Aussi, soit v_a v_b deux voxels de bordure adjacents a v_j et appartenant à différentes courbes de bordure C_p et C_q. Nous calculons $m^{a,b} = |n_a . n_b|$, où n_a et n_b sont les normales des voxels frontaliers v_a et v_b, respectivement.

Nous utilisons une matrice triangulaire M stocker dans l'élément $M_{p,q}$ la moyenne de toutes les valeurs $m_{a,b}$ tel que v_a est en courbe de bordure C_p et v_b est en courbe de bordure C_q. Dans ce cas, la moyenne est utilisée pour compenser le fait que différentes courbes de bordure peuvent avoir différents nombres de voxels de bordure.

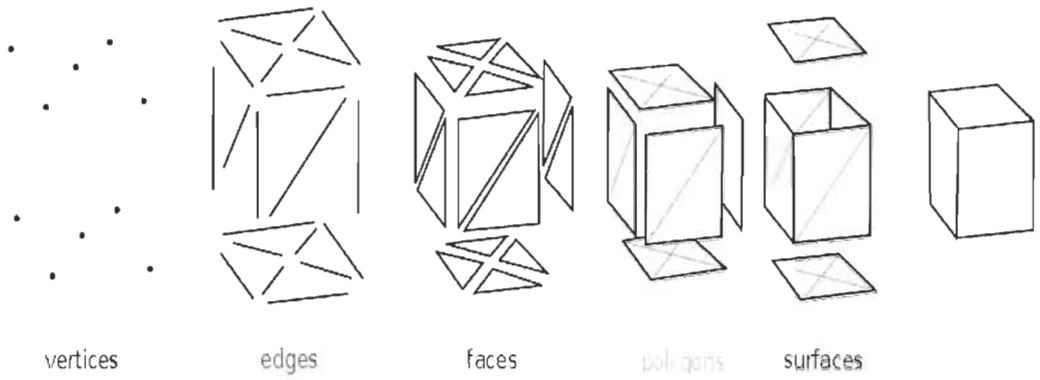


Figure 3.5 étapes de création d'une surface à l'aide de notre méthode. [*https://upload.wikimedia.org/wikipedia/commons/thumb/6/6d/Mesh_overview.svg/720px-Mesh_overview.svg.png*]

Après que la promenade à travers tous les voxels de jonction dans la courbe d'intersection est terminée, soit Mp,q l'élément de la matrice M avec la plus grande valeur, nous connectons les composants associés aux courbes de bordure Cp et Cq et l'ensemble Mp,q à zéro. La justification ici, est qu'il y a une corrélation entre les deux voxels appartenant à ces deux composantes. Cela est interprété comme une forte indication que ces deux composants appartiennent à la même surface. Ensuite, nous identifions l'élément avec la prochaine plus grande valeur dans la matrice M, connectons leurs composants correspondants et mettons la cellule de la matrice à zéro. Ce processus est répété jusqu'à ce que M ne contienne aucun élément non nul.

3.2.4 Coût de l'algorithme

Soit n le nombre d'échantillons dans le nuage de points et soit *m le nombre de voxels utilisés pour* discrétiser *sa boite englobante*. le coût d'attribution d'une entrée donnée à un voxel est $O(1)$, additionnant a $O(n)$ pour l'ensemble des données d'entrée. La classification topologique d'un voxel se fait en analysant un voisinage fini, est donc $O(1)$. Par conséquent, la classification de tous les voxels est effectuée dans $O(m)$. Pendant le remplissage des imperfections, nous avons

au plus $(2L + 1)^3$ segments de ligne voxel et peuvent avoir jusqu'à $L(2L + 1)^4m$ opérations de voxel, ou L la longueur maximale autorisée pour un segment. Le comblement de l'espace est ensuite effectué en $O(m)$. Ainsi, la première étape de l'algorithme de reconstruction de surface a coûté $O(n + m)$. L'amincissement topologique est effectué dans $O(m)$, puisque chaque voxel est visité et supprimé, s'il s'agit d'un voxel à demi-surface, ou conservé, sinon. Enfin, pour L'amincissement, nous devons traiter tous les voxels de premier plan afin de calculer la position moyenne des ensembles de points à l'intérieur de chaque voxel. Étant donné que le coût de création de la maille lui-même est $O(m)$, le coût de toute la phase de maillage est $O(n + m)$. Par conséquent, le coût total de notre algorithme de reconstruction de surface est $O(m + n)$.

3.3 Réseaux de Neurones dans la Reconstruction 3D

L'IA est désormais intégrée à presque toutes les technologies, de la technologie de base à la technologie la plus compliquée au monde. Puisqu'on pense toujours augmenter la precision de notre système nous devons intégrer les derniers concepts de l'apprentissage automatique pour donner à notre algorithme le pouvoir d'apprendre et de s'améliorer avec le temps.

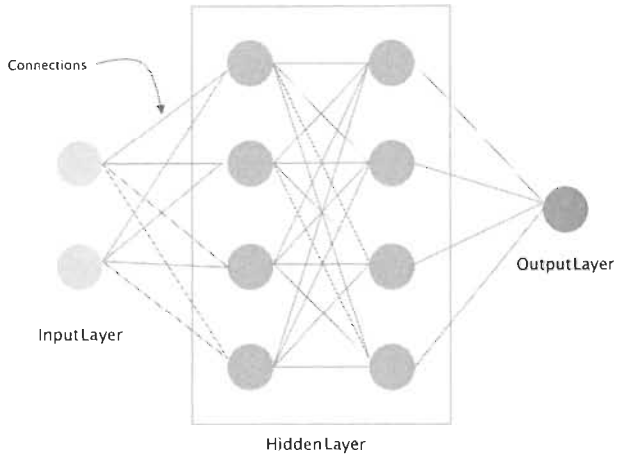
3.3.1 Concepts de réseaux de neurones

La définition la plus simple d'un réseau neuronal, plus correctement appelée réseau neuronal « artificiel » (ANN), est fourni par l'inventeur de l'un des premiers neuro-ordinateurs, Dr. Robert Hecht-Nielsen. Il définit un réseau neuronal comme:

"...un système informatique composé d'un certain nombre de simples, éléments de traitement hautement interconnectés, qui traitent les informations par leur réponse d'état dynamique aux entrées externes. dans *"Neural Network Primer: Part I" par Maureen Caudill, AI Expert, Feb. 1989*

Les réseaux de neurones sont généralement organisés en couches. Les couches sont composées d'un certain nombre de « nœuds » interconnectés qui

contiennent une « fonction d'activation ». Les modèles sont présentés au réseau via la « couche d'entrée »,



qui communique avec une ou plusieurs « couches cachées » où le traitement proprement dit est effectué via un système de « connexions » pondérées. Les couches masquées sont ensuite liées à une « couche de sortie » où la réponse est sortie comme indiquée dans le graphique ci-dessus.

La pluspart des ANN contiennent une certaine avantage qui modifie les points des connexions en fonction des modèles d'entrées qui lui sont présentées. Dans un sens, les ANN apprennent par l'exemple, tout comme leurs homologues biologiques; un enfant apprend à reconnaître les chiens à partir d'exemples de chiens [14].

3.3.2 Architecture du réseau

Dans notre implémentation, le réseau de base est construit sur le réseau VGG à 16 couches [15] avec les modifications suivantes :

- les canaux sont réduits à la moitié du réseau d'origine;
- Pour gérer de petites fonctionnalités, nous utilisons l'approximation des fonctionnalités pour obtenir une carte des fonctionnalités à haute résolution. En particulier, nous insérons une couche de sur-échantillonnage bilinéaire 2x avant d'alimenter la dernière carte de caractéristiques de convolution vers le réseau de

proposition 3D. De même, nous insérons une couche de sur-échantillonnage 4x/ 4x/ 2x avant la couche de regroupement ROI;

- Nous supprimons la 4e opération de mise en commun dans le réseau VGG d'origine, ainsi les parties de convolution de notre réseau procèdent à sous-échantillonnage de 8x.

Nous initialisons les paramètres en échantillonnant les poids du réseau VGG-16 appartenant à imageNet. Bien que notre réseau comporte trois branches, le nombre de paramètres représente environ 75 % du réseau VGG-16.

Le pipeline de la cartographie sémantique RGB-D dense proposé avec notre réseau de neurones. l'entrée d'image RGB et image de profondeur des paires de chaque image clé sont introduites dans le réseau. L'architecture du réseau proposée est illustrée à la figure 3.7. La sortie du réseau et un nuage de points sémantiquement étiqueté. La probabilité d'étiquette de chaque voxel est affinée par une mise à jour bayésienne récursive. Enfin, la carte sémantique 3D dense est générée. Nous ne pensons pas que dans l'architecture actuelle, un voxel se compose d'un seul point 3D.

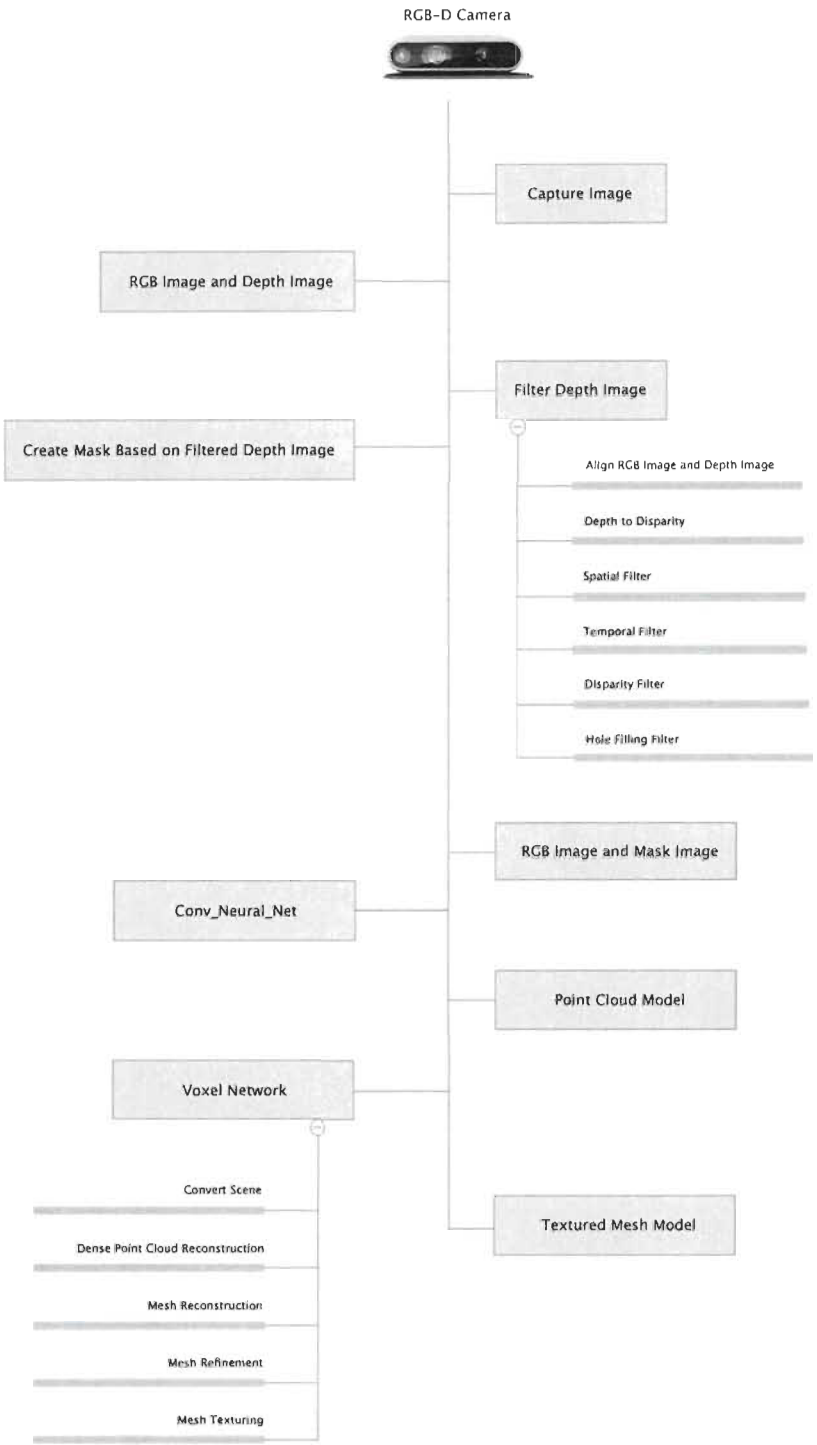


Figure 3.6 Le diagramme de flux [*ce diagramme est généré par le logiciel SketchApp*]

L'architecture du réseau proposé se compose de deux sous-réseaux avec action parallèle: un réseau normal et un VoxelNet. Le premier réseau est composé de trois blocs de construction: CNN. Le VoxelNet est composé des blocs suivants : couche entièrement connectées, pile combinée locale et globale et couche de remodelage. Il obtient des informations de contexte global via le premier réseau tout en préservant des informations de forme locale précises via le VoxelNet.

Couche	Taille du filter	Canal	Filter	Foulée	Taille de sortie	Complexité	# de zéros
Conv1	3 x 3	3	64	1	224 x 224	0.6	0.48
Conv2	3 x 3	64	64	1	224 x 224	12.0	0.32
Pool1	3 x 3			2	112 x 112		
Conv2	3 x 3	64	128	1	112 x 112	6.0	0.35
Conv2	3 x 3	128	128	1	112 x 112	12.0	0.52
Pool2	2 x 2			2	56 x 56		
Conv3	3 x 3	128	256	1	56 x 56	6.0	0.48
Conv3	3 x 3	256	256	1	56 x 56	12.1	0.48
Conv3	3 x 3	256	256	1	56 x 56	12.1	0.70
Pool3	2 x 2			2	28 x 28		
Conv4	3 x 3	256	512	1	28 x 28	6.0	0.65
Conv4	3 x 3	512	512	1	28 x 28	12.1	0.70
Conv4	3 x 3	512	512	1	28 x 28	12.1	0.87
Pool4	2 x 2			2	14 x 14		
Conv5	3 x 3	512	512	1	14 x 14	3.0	0.76
Conv5	3 x 3	512	512	1	14 x 14	3.0	0.80
Conv5	3 x 3	512	512	1	14 x 14	3.0	0.93

Figure 3.7 Architecture du Réseau VGG-16 [*Probabilistic Visibility for Multi-View Stereo, in IEEE Conference on Computer Vision and Pattern Recognition, 2007*]

Le premier sous-réseau est composé : d'un CNN prend une image RVB en entrée, VGG-16 (https://github.com/KaimingHe/deep-residual-networks), le modèle appartient à notre ensemble de données extrait des fichiers vidéo BAG. Après le CNN, la résolution des cartes d'entités est diminuée de 32 fois par rapport à l'image d'entrée; ainsi, il laisse tomber une quantité importante d'informations de forme, qui sont récupérées en utilisant le deuxième sous-réseau VoxelNet.

On note aussi que les champs récepteurs après la couche pool5 de VGG16 sont de dimension 212 x 212, qui n'est pas assez grand pour couvrir l'ensemble de l'image d'entrée 512 x 512. Par conséquent, une pile de contexte, composée d'une chaîne de 6 couches de piles de convolution 5x5x512, est concaténée au sommet d'un réseau VGG-16 pré-formé. Le réseau RVB peut étendre progressivement le champ récepteur, comme le montre la figure 3.7, pour couvrir tous les éléments de la carte des caractéristiques actuelle (L'image originale entière).

Le champ récepteur du réseau RVB peut être décrit comme :

$$\mathcal{RF}_j = \mathcal{RF}_{j-1} + (k_j - 1) \times \prod_{i=0}^{j-1} S_i, j \in [1, n] .$$

Le $\Re F_i$ et Ki sont le champ récepteur et la taille de j, Si fait référence au processus de la i-eme pile de contexte RF_0 et S_0 sont le champ récepteur et le produit de processus avant la première pile de contexte et n=6 est le nombre de piles de contexte. De plus, les cartes de score de toutes les piles de contexte sont fusionnées pour agréger des informations de contexte à plusieurs échelles. Nous remarquons que le dimensionnement spatial des cartes d'entités dans une pile de contexte est inchangée.

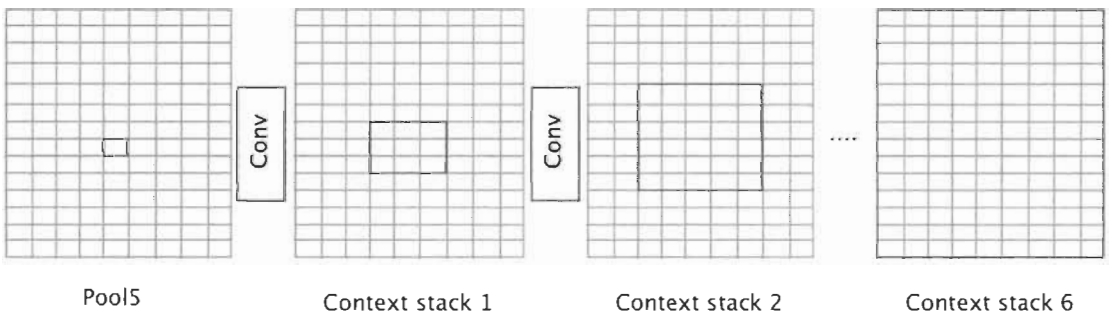


Figure 3.8 Le champ réceptif (La zone de rouge) de la pile de contexte est progressivement étendu pour couvrir tous les éléments de la carte des fonctionnalités.

L'architecture se compose de piles à 3 couches suivant séparément le pool2, le pool3 et le pool4. Afin de prouver la formation du réseau de la divergence, de manière conventionnelle, un taux d'apprentissage plus faible est adopté pour l'architecture pendant la formation. Nous utilisons la normalisation par lots, qui stabilise les signaux d'erreur retransmis; ainsi, un taux d'apprentissage plus élevé (0,01) peut être utilisé pour la formation. Cette architecture conserve le bas niveau de l'image RVB.

Reseau neuronal Voxel

L'entrée de VoxelNet est un nuage de points, qui est représenté comme un ensemble de points 3D $\{p_i | i=1,2,...,n\}$ stocké dans un vecteur de longueur n x 6, ou n est le nombre de points et est un vecteur a 6 dimensions contenant des informations de position $(X, Y, Z)^T$ dans le monde coordonnées et informations sur la couleur des pixels (R, G, B). Inspiré par PointNet [16], nous utilisons également la mise en commun maximale comme une fonction invariante. L'opération de regroupement maximal obtient la fonction globale de tous les points, qui sont concaténés avec les caractéristiques des pixels pour prédire les étiquettes sémantiques ponctuelles. La représentation des caractéristiques de dimension supérieure pour chaque point du sous-réseau peut être résumée par l'équation suivante :

$$[\mathcal{F}^1_{global} \cdots \mathcal{F}^n_{global}] = \mathcal{T}(\mathcal{M}([f^k_{mlp}(p_1) \cdots f^k_{mlp}(p_n)]))$$

f_{mlp} est le réseau de perception multicouche et k est le nombre de réseau de perception multicouche avant le regroupement maximal. Chaque point partage le même ensemble de poids entièrement connectés. M est l'opération de regroupement maximale avec la taille du noyau n x 1, et $\mathcal{T}$ est l'opération de tuile, qui restaure la forme de la carte à partir de 1 x 1 to n x 1.

Le dimensionnement spatial des entités est le même que celui des données d'entrée dans le réseau voxel, de sorte qu'elle peut conserver toutes les informations de forme d'origine. Cependant, si une seule couche de mise en commun maximale est adoptée pour générer la fonctionnalité globale, elle

supprimera des informations contextuelles importantes de l'entrée nuage de points.

3.4 Architecture du Système

Nous avons utilisé une variété de matériel pour pouvoir faire fonctionner le système, pour la caméra de travail, nous avons utilisé une caméra de profondeur RealSense et pour pouvoir former le modèle que nous avons utilisé dans le système. Nous définissons les principaux concepts dans cette section.

3.4.1 Technologie infonuagique pour l'entraînement du modèle

Nous pouvons accéder à la console amazon ML en vous connectant à la AWS Management Console, et en ouvrant la console amazon ML à https:// console.aws.amazon.com/machinelearning/.

Nous pouvons utiliser Amazon Machine Learning pour appliquer l'apprentissage automatique à des problèmes pour lesquels nous avons des exemples existants de réponses réelles. Par exemple, nous utiliserons des serveurs AWS ML pour former notre modèle et utiliser le modèle après dans un système embarqué comme un robot ou des voitures autonomes. nous pouvons utiliser des approches ML supervisées pour ces tâches d'apprentissage automatique spécifiques. Les étapes de l'utilisation d'AWS pour la formation de notre modèle :

1 - Préparer les données

2 - Création d'une dataset pour l'entrainement

3 - Utilisez le modèle pour générer le fichier 'm5'

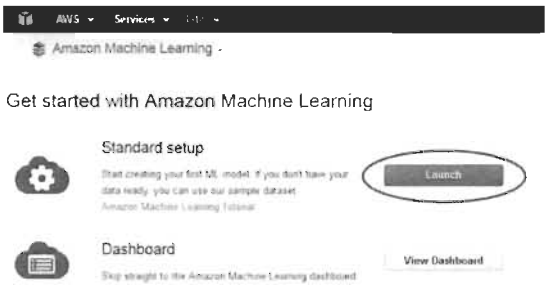
4 - Téléchargez le fichier «m5»

Creation d'un dataset d'entrainement :

1. Ouvrez la console Amazon Machine Learning à l'adresse https:// console.aws.amazon.com/machinelearning/

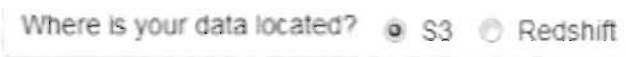
2. Nous choisissons Commencer

3. Sur la page Premiers pas avec Amazon Machine Learning, choisissez Lancer

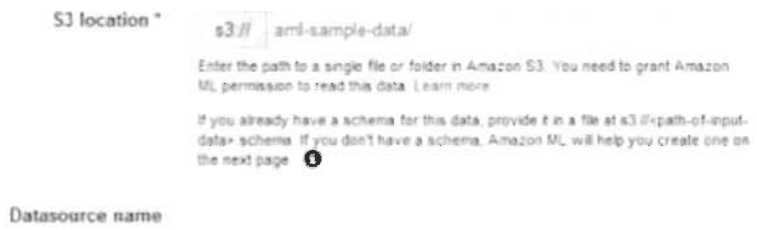


4. Sur la page Input Data, pour Où se trouvent vos données, assurez-vous que S3 est sélectionné

5. Pour Emplacement S3, saisissez l'emplacement complet des fichiers à partir de l'étape 1: Préparez vos données. par example: your-bucket/Kalmia.csv. Amazon ML se prépare s3:// notre_nom_de_bucket

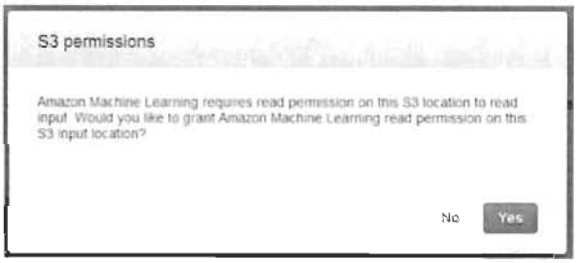


6. Pour le nom de la source de données, nous tapons le nom du projet



7. Nous choisissons de vérifier

8. Dans la boîte de dialogue des autorisations S3, et nous choisissons oui



3.4.2 Caméra RealSense D435

Les caméras Intel RealSense gagnent en popularité depuis quelques années pour être utilisées comme caméra 3D et pour l'odométrie visuelle. Les informations de profondeur sont importantes car elles nous fournissent les informations nécessaires pour comprendre les formes, les tailles et la distance. Cela nous permet (ou à un robot) de savoir à quelle distance il se trouve des objets pour éviter de tomber sur des objets et de planifier un chemin autour des obstacles dans le champ de vision de l'image. Traditionnellement, ces informations proviennent du RADAR ou du LIDAR, mais dans certaines applications, nous pouvons également obtenir cela des caméras. Dans les caméras, nous obtenons souvent de la profondeur en utilisant deux caméras pour une vision stéréo.

La caméra Intel RealSense Depth (série D400) utilise la détection de profondeur stéréoscopique pour déterminer la portée d'un élément. Donc, essentiellement, il a deux caméras et peut en faire la triangulation pour la stéréo. Ce capteur utilise deux caméras infrarouges pour la chaîne stéréo et dispose également d'une caméra RGB à bord. Nous pouvons donc obtenir quatre produits de données du capteur; Image RGB, image en profondeur, image infrarouge gauche et image infrarouge droite. Nous pouvons considérer chaque image comme un instantané 3D de l'environnement, où chaque pixel de couleur (RGB) a également une valeur de plage (profondeur) pour l'élément qui se trouve dans l'image.

Plus les éléments sont éloignés de l'appareil photo, plus l'erreur de portée / profondeur sera grande. Les caméras D400 ont un projecteur infrarouge pour obtenir de meilleures fonctionnalités sur les surfaces avec une texture minimale dans la caméra infrarouge pour calculer la reconstruction stéréo. Ce projecteur

peut être allumé et éteint si nous le voulons. La désactivation du projecteur est souvent utile pour le suivi des applications, car les points projetés ne bougent pas avec les éléments suivis.

Une chose à savoir est que les images infrarouges sont rectifiées (pour que les images se ressemblent dans un plan commun) dans l'appareil photo, mais l'image de la caméra RGB n'est pas rectifiée. Cela signifie que si vous souhaitez que la profondeur et les images RGB soient bien alignées, vous devez rectifier manuellement l'image RGB.

Note: nous utilisons le D435i (la version IMU), utilisons les timestamps des images et INU pour synchroniser les deux capteurs. Lorsque nous utilisons l'heure système.

	D415	D435	D435i
Image Sensor	OV2740	OV9282	OV9282
Active Pixels	1980X1080 COLOR	1280X800 Monochrome	1280X800 Monochrome
Sensor Aspect Ratio	16:9	8:5	8:5
Baseline	55mm	50mm	50mm
F Number	f/2.0	f/2.0	f/2.0
Focal Length	1.88mm	1.93mm	1.93mm
Filter Type	IR Cut - D400, None D410	None	None
Focus	Fixed	Fixed	Fixed
Shutter Type	Rolling Shutter	Global Shutter	Global Shutter
Signal Interface	MIPI CSI-2, 2X Lanes	MIPI CSI-2, 2X Lanes	MIPI CSI-2, 2X Lanes
Horizontal Field of View	69.4	91.2	91.2
Vertical Field of View	42.5	65.5	65.5
Diagonal Field of View	77	100.6	100.6
Distortion	<= 1.5%	<=1.5%	<=1.5%
IMU	N/A	N/A	BMI055

Figure 3.9 Schéma global du fonctionnement du RealSense D435i [https://www.intelrealsense.com/depth-camera-d435/].

L'utilisation d'un capteur RealSense D435i et d'un capteur RealSense T265 peut fournir à la fois les cartes et l'odométrie visuelle de meilleure qualité pour

développer un système SLAM complet. Le D435i utilisé pour la cartographie et le T265 pour le suivi. Intel fournit la bibliothèque RealSense SDK 2.0 pour l'utilisation des caméras RealSense. Il est Open Source et fonctionne sur Mac, Windows, Linux et Android. Il existe également des wrappers ROS et OpenCV.

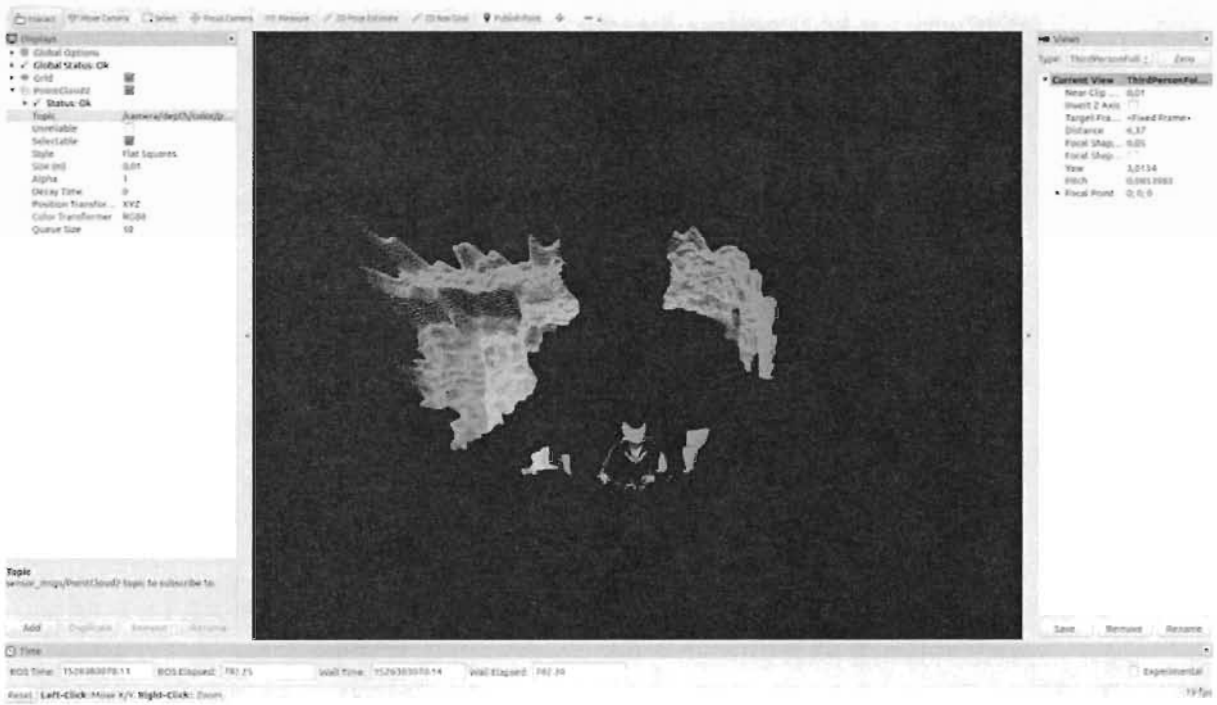
Figure 3.10 Dans le SDK (kit de développement logiciel), il comprend un éditeur pour vous permettre de visualiser des images, d'enregistrer des images, de modifier les paramètres ou de mettre à jour le *firmware*.

3.4.3 Calibration

Lors de l'application de la caméra à certaines applications, des problèmes techniques de précision dans la mesure de profondeur se produisent parfois. Par exemple, lorsqu'une caméra de profondeur est utilisée pour capturer un objet plan, des artefacts visibles sont observés. Pour résoudre ce problème, dans cette section, nous proposerons une méthode simple et efficace pour calibrer la caméra de profondeur :

1. Nous commençons par installer les prérequis pour le script de calibration, nous avons besoin d'OpenCV et du SDK RealSense ;

2. Connectez la caméra RealSense via USB et exécutez le fichier de lancement de la démonstration .



3. Calibrer la caméra à l'aide d'opencv2-python

```
import numpy as np

import cv2

import glob


# critères de résiliation

Criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)


# prepare object points for a 8x6 chess board

objp = np.zeros((6*8,3), np.float32)

objp[:,:2] = np.mgrid[0:8,0:6].T.reshape(-1,2)


# Tableaux pour stocker des points d'objet et des points d'image de toutes
les images.

objpoints = []

imgpoints = []
```

```
images = glob.glob('/path_to_images/*.png')


for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
# Trouver les coins de la carte de calibration
ret, corners = cv2.findChessboardCorners(gray, (8,6),None)

# s il trouve, ajoutez des points d'objet, des points d'image (apres les
avoir affines)
if ret == True:
        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        imgpoints.append(corners2)

        # dessiner et afficher les coins
        img = cv2.drawChessboardCorners(img, (8,6), corners2,ret)
        cv2.imshow('img',img)
        cv2.waitKey(500)

cv2.destroyAllWindows()
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1],None,None)
np.save('/path_to_images/calibration', [mtx, dist, rvecs, tvecs])
```

Avant d'exécuter le script, nous devrons obtenir des images de la caméra avec la capture de la carte d'étalonnage. Nous avons besoin d'au moins 10 images pour

l'étalonnage. Une fois le script d'étalonnage terminé, il enregistrera les données d'étalonnage dans le fichier calibration.npy. pour réutiliser les données d'étalonnage, nous devons exécuter le script suivant :

```
calibration_data = np.load('path_to_images/calibration.npy')
mtx = calibration_data[0]
dist = calibration_data[1]
rvecs = calibration_data[2]
tvecs = calibration_data[3]
```

3.5 Outils logiciels

Notre système est conçu pour que les calculs soient effectués par un robot. Cela rend le facteur d'optimisation très important. Nous avons utilisé Python 3 comme langage principal pour le système et RealSense SDK (wrapper Python) qui nous permet d'interagir avec le système de caméra, quant au rendu et à la reconstruction 3D, nous avons utilisé la bibliothèque Open3D, pour les algorithmes ML, nous avons utilisé le framework Tensorflow pour de tels travaux. Dans cette section, nous parlerons de la raison pour laquelle nous avons utilisé ces technologies.

3.5.1 Python et Tensorflow

Python est un langage de programmation qui permet aux nouveaux programmeurs expérimentés de convertir facilement des idées en code. Il est actuellement le plus répandu, le plus mature et le mieux supporté parmi les langages de programmation dans le domaine de la technologie.

La vision par ordinateur, quant à elle, permet aux ordinateurs d'identifier des objets grâce à des images ou des vidéos numériques. L'implémentation de CV via Python permet aux développeurs d'automatiser les tâches qui impliquent la

visualisation. Alors que d'autres langages de programmation prennent en charge la vision par ordinateur, python domine la concurrence.

Attributs forts de Python : *Alors que nous n'utilisons pas tous les avantages de cette liste, mais cette liste démontrera la domination complète de Python dans ce domaine :*

- *Facilité de codage :* (Codage en anglais simple) est le but du python. Cela permet au programmeur de se concentrer sur la conception et non sur le codage ;
- *Prototypage rapide :* Puisque nous pouvons nous concentrer davantage sur la conception, nous pouvons maintenant expérimenter plus d'idées de conception. Python est bien adapté pour implémenter de nouvelles fonctionnalités. Les bibliothèques comme OpenCV sont écrites en C ++ et rendent Python plus lent car il appellera toujours les bibliothèques C / C ++. Cela signifie que nous aurons l'avantage de développement de python tandis que nous pouvons avoir une optimisation des performances de C ++ .
- *De vastes bibliothèques pour l'apprentissage automatique (comme Tensorflow) :* Python est couramment utilisé pour l'apprentissage automatique. Les scientifiques des données investissent leur temps à contribuer, car il est facile à coder et gratuit. Les développeurs de CV n'ont pas à se soucier beaucoup des projets sur lesquels ils travaillent car la plupart de leurs cas sont déjà couverts par les bibliothèques Python .
- *Open Source :* Python est gratuit contrairement à d'autres technologies dans le domaine comme MATLAB.
- *Web Integration :* Python a des frameworks web matures comme Django. Il vise un temps de développement rapide, des conceptions soignées et réalistes. En outre, Python a des micro-cadres qui sont tout aussi fonctionnels que leurs homologues plus grands .
- *Le plus couramment utilisé :* cela signifie qu'il a une plus grande communauté. Beaucoup de billets de blog et de ressources en ligne concernant python .

Pourquoi nous utilisons Tensorflow ?

Tout d'abord ce qu'est Tensorflow, Tensorflow est une puissante bibliothèque d'apprentissage automatique orientée flux de données créée par l'équipe Brain de Google et rendue open source en 2015. Il est conçu pour être facile à utiliser et largement applicable aux problèmes orientés vers les réseaux numériques et neuronaux ainsi que dans d'autres domaines. Fondamentalement, Tensorflow est une boîte à outils de bas niveau pour faire des mathématiques complexes et il cible les chercheurs qui savent ce qu'ils font pour créer des architectures d'apprentissage expérimentales, pour jouer avec elles et les transformer en logiciels en cours d'exécution. Voici les avantages de l'utilisation de Tensorflow :

- ***Graphs :*** Tensorflow a de meilleures visualisations de graphiques de calcul, qui sont indigènes par rapport à d'autres bibliothèques comme Torch et Theano ;
- ***Gestion de bibliothèque :*** Soutenu par Google, Tensorflow a l'avantage de la performance transparente, des mises à jour rapides et des nouvelles versions fréquentes avec de nouvelles fonctionnalités ;
- ***Débogage :*** Tensorflow vous permet d'exécuter des sous-parties d'un graphique, ce qui lui donne un avantage car vous pouvez introduire et récupérer des données discrètes sur un bord et avoir un excellent débogage ;
- ***Évolutivité (Scalability) :*** Les bibliothèques peuvent être déployées sur une gamme de machines matérielles, à partir de périphériques cellulaires vers un ordinateur avec une configuration complexe ;
- ***parallélisme (Pipelining) :*** Tensorflow est hautement parallèle et conçu pour utiliser divers logiciels d'arrière-plan .

3.5.2 Bibliothèque Open3D

Open3D est une bibliothèque open source qui prend en charge le développement rapide de logiciels traitant des données 3D. Le *frontend* d'Open3D expose un ensemble de structures de données et d'algorithmes soigneusement sélectionnés en C++ et en python. Le *backend* est hautement optimisé et est configuré pour la parallélisation.

La bibliothèque prend en charge une variété de compilateurs comme le GCC 5.x Ou plus récent on Linux, XCode 8.0 Ou plus récent on OS X est le Visual Studio 2017 Ou plus récent on Windows.

Installation depuis PyPI ou Conda

Les packages python Open3D sont distribués via PyPI et Conda, version Python prise en charge (2.7, 3.5, 3.6), nous devons exécuter la commande suivante dans le terminal :

```
pip install open3d .
```

Nous utilisons également un environnement virtuel Conda pour la conteneurisation. Sinon, selon les configurations, pip3 peut être nécessaire pour python 3, ou l'option *user* peut être utilisée pour éviter les problèmes d'autorisation.

3.6 Conclusion

Nous avons présenté la conception de notre système et souligné les choix de conception pris en raison de l'utilisation prévue de ce système. Notre système est basé sur la caméra de profondeur RealSense D435i et nous souhaitons d'atteindre en temps réel le matériel de base. La procédure d'étalonnage est facile à apprendre et ne nécessite qu'un temps d'arrêt système minimal.

4

Résultats

Basé sur le système de reconstruction 3D développé, nous avons examiné plusieurs applications dans le domaine d'imagerie et vision artificielle. Dans cette partie, nous abordons les deux principales applications de ce travail. Nous discutons aussi de l'utilisation de notre système pour divers systèmes autonomes.

Pour développer et tester le système présenté dans cette section, nous utilisons la caméra Real-sense (D435) de Intel. Dans cette partie du travail, nous présentons les résultats que nous avons obtenus, et les différentes expériences impliquées dans la réalisation de ce travail.

4.1 Résultats et discussions :

Toutes ces expériences sont exécutées sur une machine virtuelle AWS EC2 P3 en utilisant une configuration minimale de Deep Learning AMI (Ubuntu) version 12.0 dans un ordinateur GPU avec 4 cœurs de vGPU et 16 Go de RAM avec 256 stockage vSSD. Nous utilisons cette machine virtuelle via une connexion SSH pour accéder à l'instance EC2.

Pour les données d'entraînement, on a utilisé une caméra RealSense D435 et pour le format des données vidéo, c'est un format de fichier BAG (comme mentionné dans le chapitre 03).

Pour le calibrage de la caméra (comme mentionné dans la section Calibration de la caméra du chapitre 3), on a utilisé un échiquier pour effectuer le calibrage géométrique de la caméra et estimer les paramètres intrinsèques de la caméra, y compris la dimension de l'image en pixels, la distance focale et les coordonnées du pixel intersecté. par l'axe optique de la caméra.

Pendant le développement de ce travail, on a capturé plusieurs vidéos de nombreuses plantes de (kalmia, Bleuet, Tomate, Erable). Dans nos expériences, nous choisissons 50 vidéos 3D haute résolution comme vidéos optimales pour l'entraînement et les tests. Le processus de capture est simple et automatisé avec le logiciel de visualisation RealSense. Compte tenu des vidéos acquises, on a pu reconstruire des modèles 3D de la plante avec la méthode de la structure à partir du mouvement.

4.1.1 Collection des images RGB-D

On a utilisé la caméra pour générer plusieurs images RGB-D afin de générer le modèle 3D. On a aussi collecté plus de 500 images RGB-D pour le réseau neuronal.

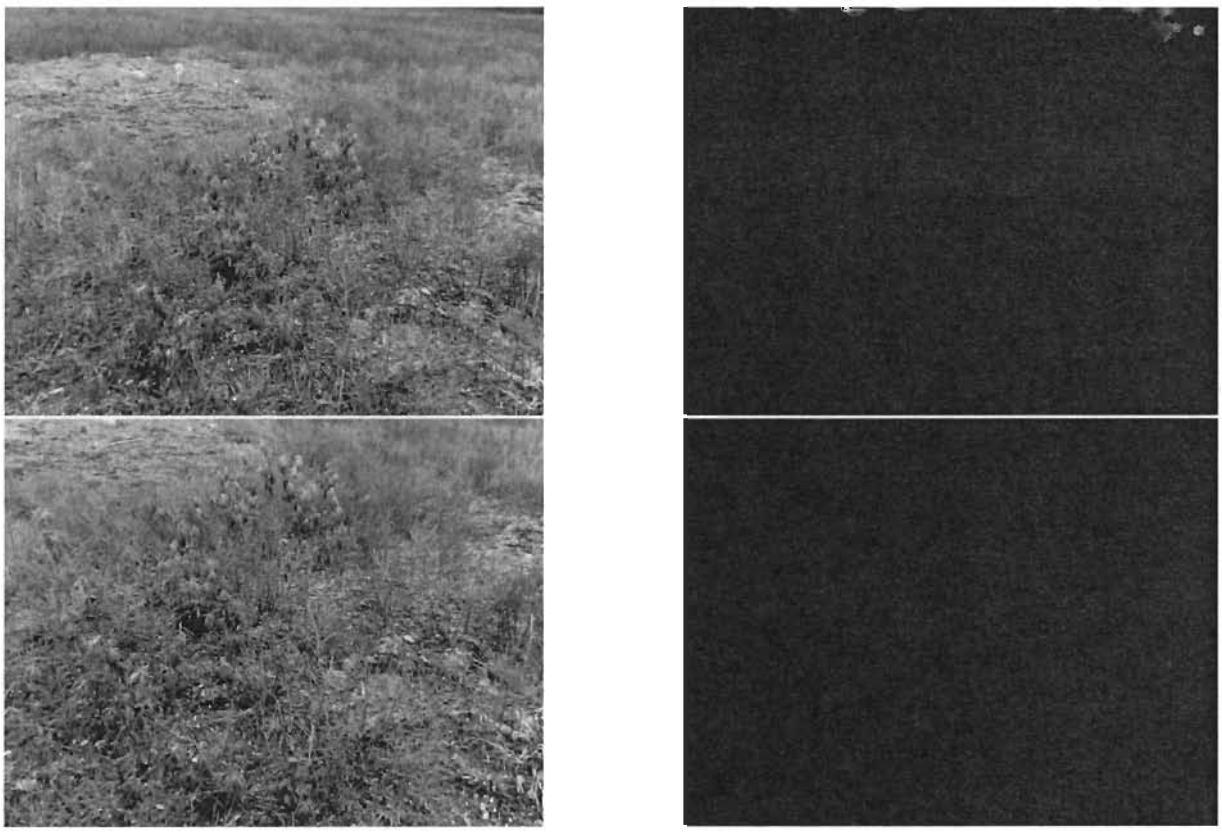


Figure 4.1 Exemples d'images générées à l'aide de la caméra RealSense. les images RGB (a gauche), les images Depth (a droite).

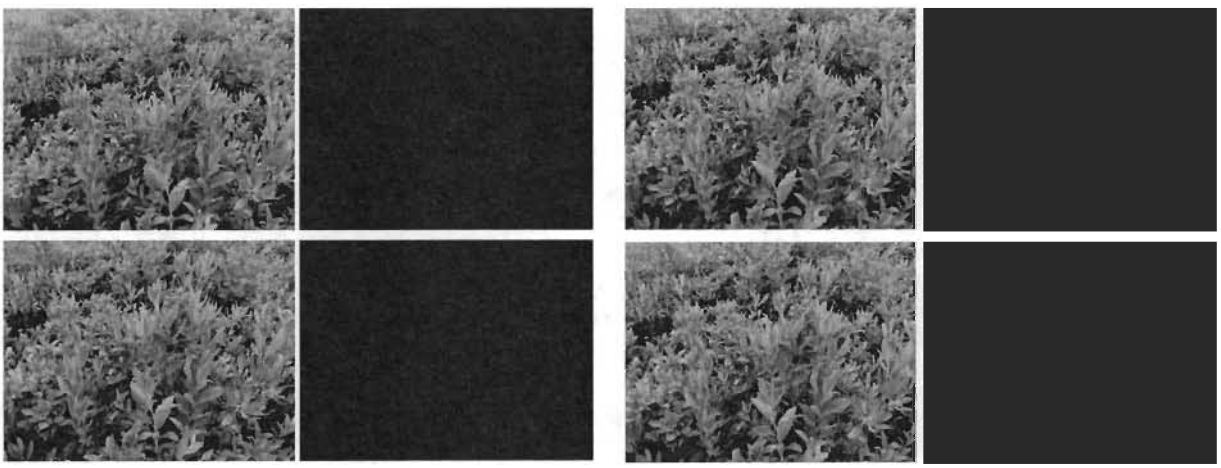


Figure 4.2 Exemples d'images générées à l'aide de la caméra RealSense Pour le Kalmia-LaCroche.

Figure 4.3 Exemples d'images générées à l'aide de la caméra RealSense Pour Bleuet-Duke.

4.1.2 Résultats des Reconstruction 3D

Dans nos expériences initiales (Figure 4.4) où le nombre de vidéos capturées était relativement faible, nous pouvons clairement voir que les modèles 3D dans la figure 4.4 ont moins de détails.

Figure 4.4 Premiers résultats des expériences avant d'avoir suffisamment de vidéo pour entraîner le système.

Pour résoudre les problèmes cités précédemment et améliorer la qualité des modèles 3D, les séquences vidéo ont été segmentées et entraînées au réseau

VGG. Le processus de calcul pour la reconstruction du modèle 3D a été réduit, diminuant ainsi considérablement l'erreur de non-correspondance des images.

Cette méthode était basée sur l'implémentation ImageNet VGG-16, la plante et le fond non vert des images acquises ont ensuite été segmentés en comparant la valeur de chaque pixel à un certain seuil dans l'espace colorimétrique HSV (hue-saturation-value) est LAB (lightness-red/magenta, green- yellow, blue).

Avec les données de la segmentation d'objets végétaux à partir de différentes vues, notre méthode a été appliquée pour transformer une image en une collection des vecteurs et effectuer une correspondance de points clés (comme le montre la figure 4.5). Les points clés entre deux images ont été mis en correspondance en identifiant leurs voisins les plus proches. Dans certains cas, cependant, la deuxième correspondance la plus proche peut être très proche de la première. Cela peut se produire à cause du bruit ou pour d'autres raisons.

Figure 4.5 Affichage de points clés mis en correspondance avec de vecteurs.

Ensuite, sur la base des résultats, la méthode est appliquée pour reconstruire le modèle 3D de la scène. En utilisant notre méthode, nous pouvons calculer les positions de la caméra. La représentation de notre reconstruction 3D était un nuage de points (point-cloud) qui pouvait être utilisé par n'importe quel système et il a été généré dans un format polygone, qui est utilisé dans des objets

graphiques, et nous générons une carte de hauteur de la scène comme indiqué dans la Figure 4.5.

L'inspections visuelles du modèle que nous avons obtenu (Figure 4.4-4.5), et celles rapportées dans la littérature [17], indiquent que nos modèles 3D sont de meilleure qualité car ils ne manquent pas beaucoup de détails sur la surface des feuilles, les pétioles.

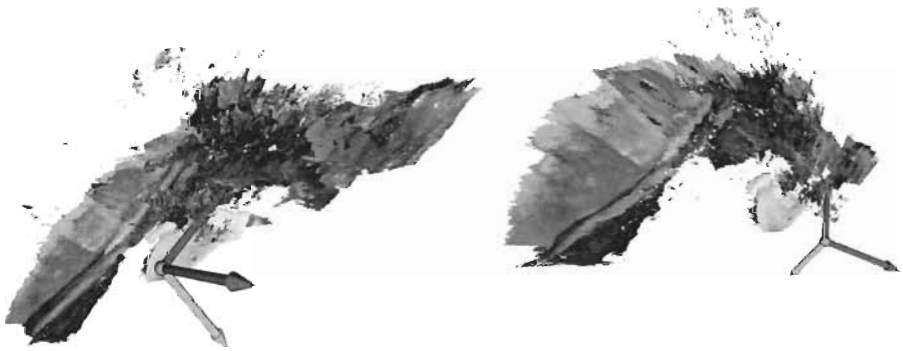


Figure 4.6 Échantillons de la reconstruction 3D d'une scène Kalmia.

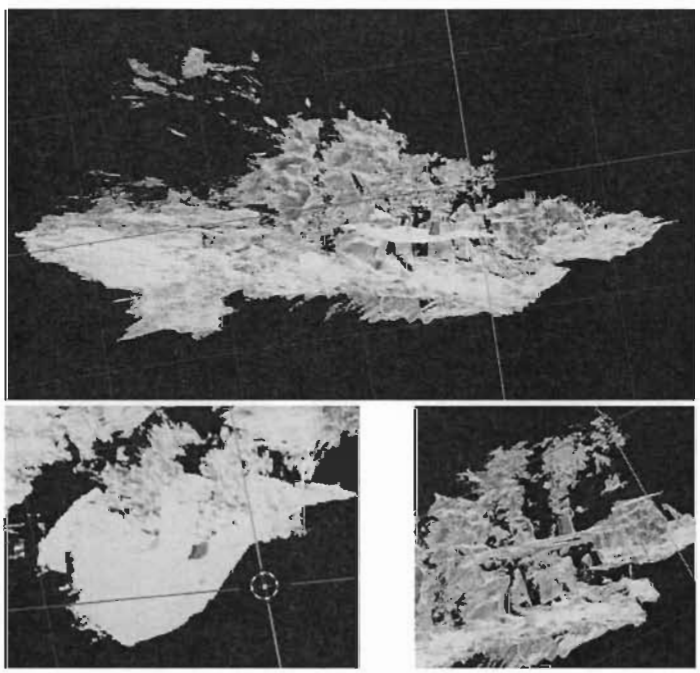


Figure 4.7 Version nuage de points pour le Bleuet-duke.

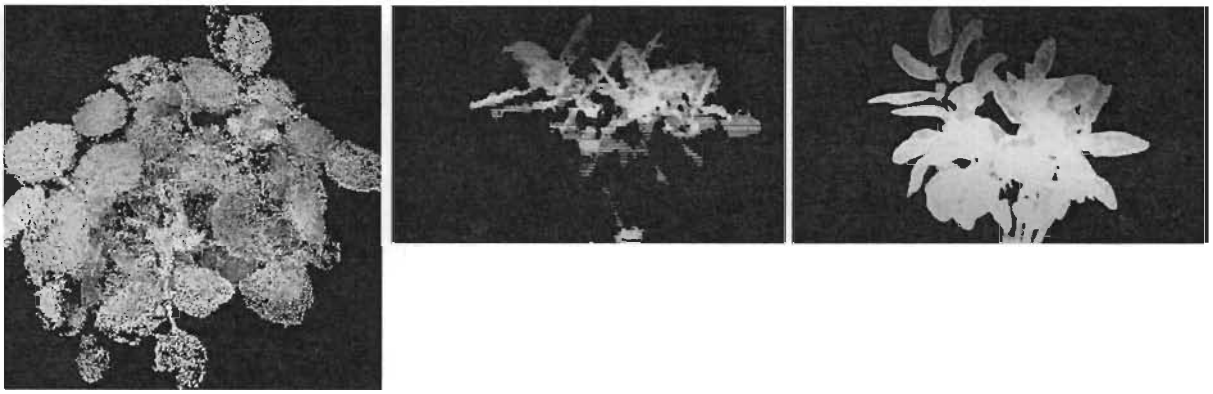


Figure 4.8 Droite : le rendu du nuage de points avec Blender. Gauche: images de profondeur pour aider à la résolution du nuage de points, nous avons utilisé le logiciel Blender.

Dans certaines études, la validation des modèles 3D consiste à extraire des lectures 2D et à les comparer aux mesures obtenues à l'aide du phénotypage destructif manuel. Une autre approche de validation possible consiste a l'utilisation de lectures 2D d'extraction virtuelle unique et de comparaison de celles-ci aux mesures obtenues à l'aide du phénotypage destructif manuel. Une autre approche de validation possible implique l'utilisation d'ensembles de données virtuelles uniques qui ont permis aux auteurs d'évaluer l'exactitude de leurs reconstructions [10,11].

En particulier, les méthodes de pointe permettent de reconstruire des formes 3D représentées par des volumes d'au plus 32^3 voxels en utilisant des configurations informatiques de pointe. Dans ce travail, nous avons présenté un algorithme de reconstruction évolutif. Les performances de notre implémentation surpassent les autres méthodes lors de la reconstruction de volumes 3D (voir chapitre 02), une complexité de mémoire plus petite lors de l'entrainement et les tests, et une durée sous-linéaire marque les ensembles de données que notre méthode peut produire des reconstructions hautes résolutions avec une précision de pointe.

4.2 Comparaison basée sur les modèles 3D

Dans cette section, nous présenterons 4 comparaisons basées sur des modèles 3D, deux du même modèle (Bleuet de même variété), Bleuet avec Bleuet nain, Bleuet avec Tomate et Bleuet avec Érable.

L'algorithme de comparaison est simple, il commence par la lecture des nuage de points (point-clouds-PLY) après l'étape du sous-échantillonnage (downsampling), et la troisième étape consiste à estimer les normales, la dernière partie de l'algorithme est le raffinement ICP (Iterative Closest Point).

Au cours de chaque itération ICP, 1000 points sont échantillonnés au hasard à partir de la source (fichier1.ply), puis le plus proche voisin de chaque point échantillonné de l'ensemble de données cible (fichier2.ply) est calculé. Après, le rejet des valeurs aberrantes est appliqué, et les points restants sont introduits dans le système linéaire. Une fois que cela est résolu, une nouvelle matrice de transformation est appliquée sur le fichier PLY est calculée, et un nouveau cycle d'itération commence si une amélioration significative a été apportée. Sinon, l'algorithme ICP se termine par une matrice de transformation qui aligne les points de (fichier1.ply) avec ceux de (fichier2.ply) [21].

4.2.1 Comparaison des plantes basée sur les caractéristiques

Open3D fournit des implémentations de plusieurs méthodes d'enregistrement, y compris l'enregistrement global par paire, le raffinement local par paire et l'enregistrement multi-voies à l'aide de l'optimisation du graphe de pose et un *workflow* d'enregistrement par paires complet pour les nuages de points. Le *workflow* commence par la lecture de nuages de points bruts, le sous-échantillonnage (downsampling) [16-a], et l'estimation des normales :

```
from py3d import *
source = read_point_cloud('source.pcd')
target = read_point_cloud('target.pcd')
source_down = voxel_down_sample(source, 0.05)
target_down = voxel_down_sample(target, 0.05)
estimate_normals(source_down,KDTreeSearchParamHybrid(radius = 0.1, max_nn= 30))
estimate_normals(target_down,KDTreeSearchParamHybrid(radius = 0.1, max_nn= 30))
```

```
result_icp = registration_icp(source, target, max_correspondence_distance = 0.02,
result_ransac.transformation,TransformationEstimationPointToPlane())
```

Ici, *TransformationEstimationPointToPlane()* invoque un algorithme ICP point à plan. D'autres variantes ICP sont également implémentées.

A. Bleuet avec Bleuet

Figure 4.9 (1) Plante Bleuet. (3) Plante Bleuet même variété. (2) Reconstitution de la première variété Bleuet. (4) Reconstitution de la deuxième variété de Bleuet.

La première étape consiste à appliquer l'algorithme qui effectue une correspondance (*Matching*) entre les deux nuages de points (*point-cloud*), ensuite l'algorithme calculera la distance des nuages, comme on peut le voir sur la figure 4.9 .

Comme nous pouvons voir dans la figure 4.10 d'après les résultats du registre ICP, il y a plus de 90 % de correspondance entre les deux nuages de points (*Matching*). et les résultats du modèle sont comme le montre la figure 4.10. comme nous pouvons le voir, les résultats du modèle sont (Score=1) ce qui signifie que les deux *point-cloud* ont plus de 85 % de compatibilité.

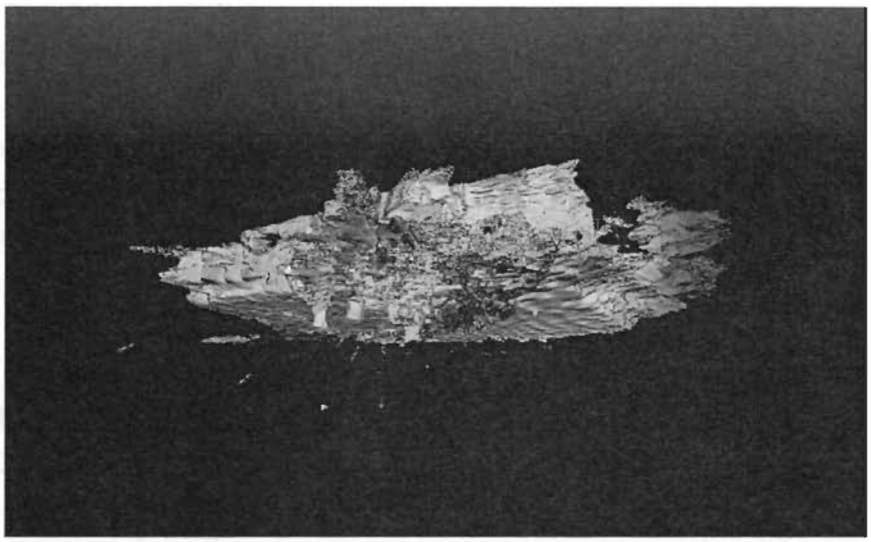


Figure 4.10 Comparer les deux nuages de points (*Matching Point Clouds*) et la distance du nuage (*Distance Clouds*).

```
Format = auto
Extension = ply
Format = auto
Extension = ply
Initial alignment
registration::RegistrationResult with fitness=1.886643e-05, inlier_rmse=1.400607e-02, and correspondence_set size of 37
Access transformation to get result.
Apply point-to-point ICP
registration::RegistrationResult with fitness=8.626549e-03, inlier_rmse=1.253748e-02, and correspondence_set size of 16918
Access transformation to get result.
Transformation is:
[[ 0.54807993 -0.38622631 -0.74230307  1.24586412]
 [ 0.46027077  0.87969367 -0.11808932  0.02654572]
 [ 0.69806389 -0.27785479  0.6599546  -1.09525317]
 [ 0.          0.          0.          1.        ]]

Apply point-to-plane ICP
registration::RegistrationResult with fitness=1.458681e-02, inlier_rmse=1.154695e-02, and correspondence_set size of 28607
Access transformation to get result.
Transformation is:
[[-0.46654784  0.64231726 -0.60871799  2.6430692 ]
 [ 0.31047135  0.76291849  0.5663935  -1.24378583]
 [ 0.82793397  0.07495936 -0.55602141  0.66232881]
 [ 0.          0.          0.          1.        ]]
Score=1
```

Figure 4.11 Résultats de la comparaison des deux nuages de points (Bleuet même variété).

B. Bleuet avec Bleuet-Nain

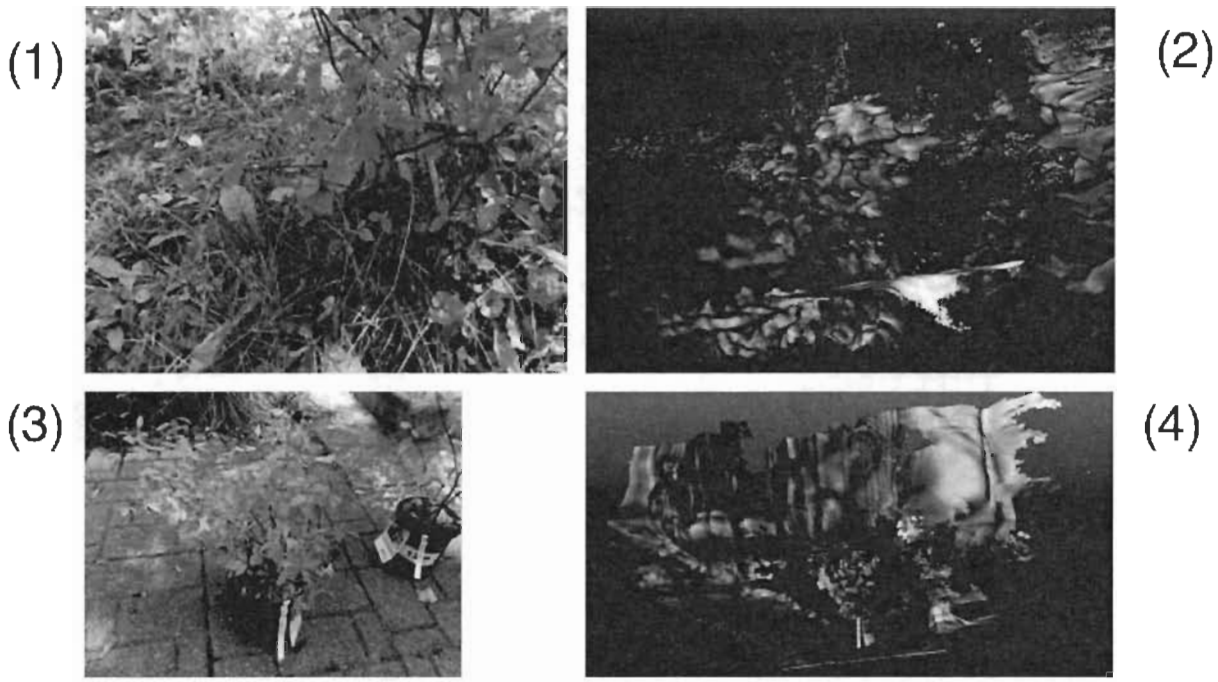


Figure 4.12 (1) Plante Bleuet. (3) Plante Bleuet Nain. (2) Reconstitution de Bleuet. (4) Reconstitution de la plante Bleuet Nain.

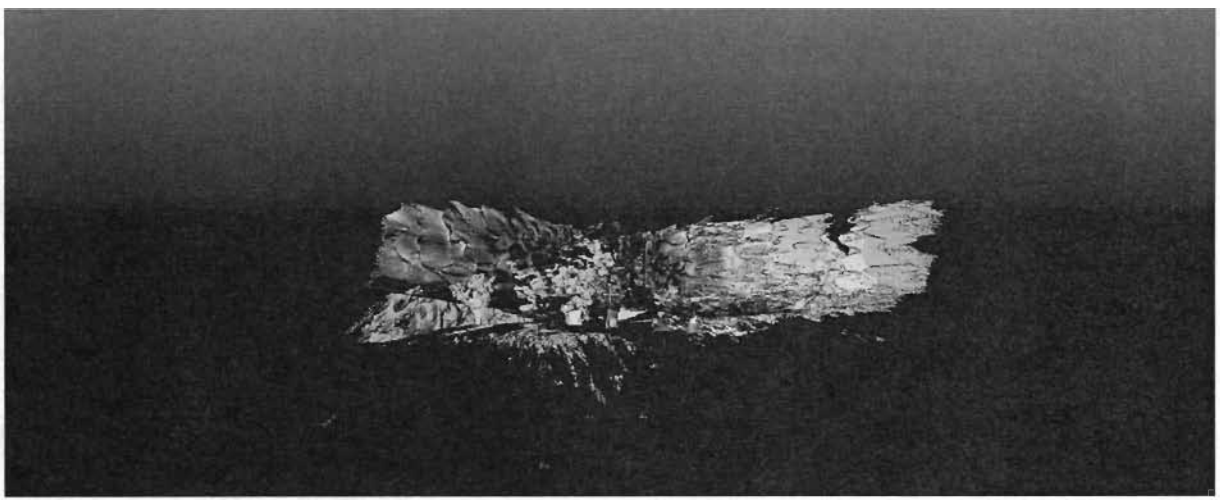


Figure 4.13 Comparer les nuages de points(*Matching Point Clouds*) et la distance du nuage (*Distance Clouds*).

Nous devons calculer la distance des deux nuages de points pour voir la différence entre les deux. Comme on peut le voir sur la figure 4.12 il y a une grande différence quand on calcule la distance entre les deux nuages de points. Et aussi pour générer la comparaison entre les deux plantes, nous devons appliquer la fonction *registration_icp()* comme discuté dans la section *4.1.4*.

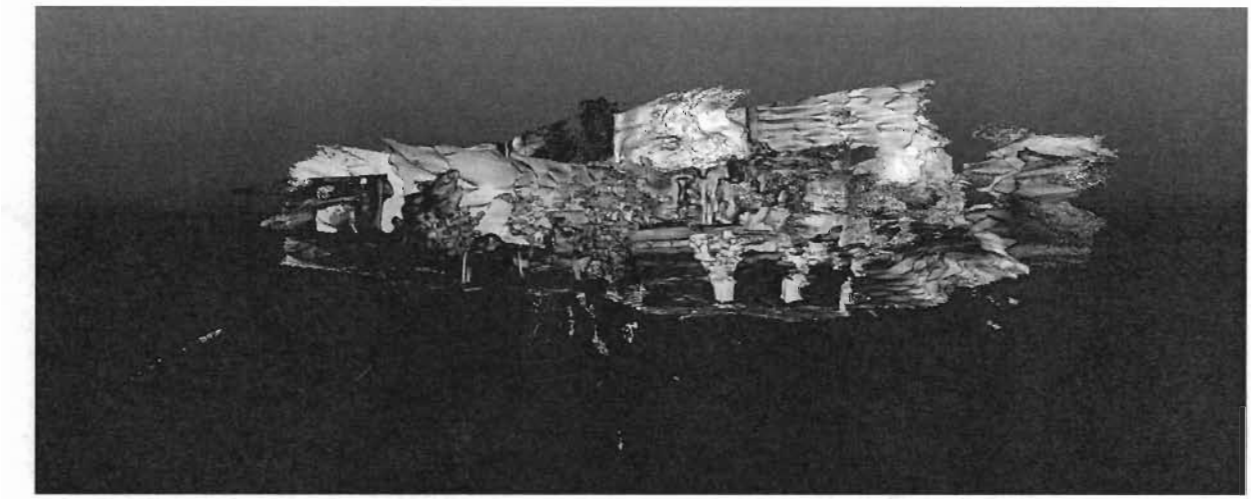


Figure 4.14 Résultats de la fonction *registration_icp()* affichés dans MeshLab.

Comme prévu, la comparaison des deux plantes a été négative comme on peut le voir dans les résultats du modèle.

```
Format = auto
Extension = ply
Format = auto
Extension = ply
Initial alignment
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Apply point-to-point ICP
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Transformation is:
[[ 0.862  0.011 -0.507  0.5  ]
 [-0.139  0.967 -0.215  0.7  ]
 [ 0.487  0.255  0.835 -1.4  ]
 [ 0.     0.     0.     1.   ]]

Apply point-to-plane ICP
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Transformation is:
[[ 0.862  0.011 -0.507  0.5  ]
 [-0.139  0.967 -0.215  0.7  ]
 [ 0.487  0.255  0.835 -1.4  ]
 [ 0.     0.     0.     1.   ]]
Score=0
```

Figure 4.15 Résultats de la comparaison Bleuet vs Bleuet Nain (Score=0).

C. Bleuet avec Tomate

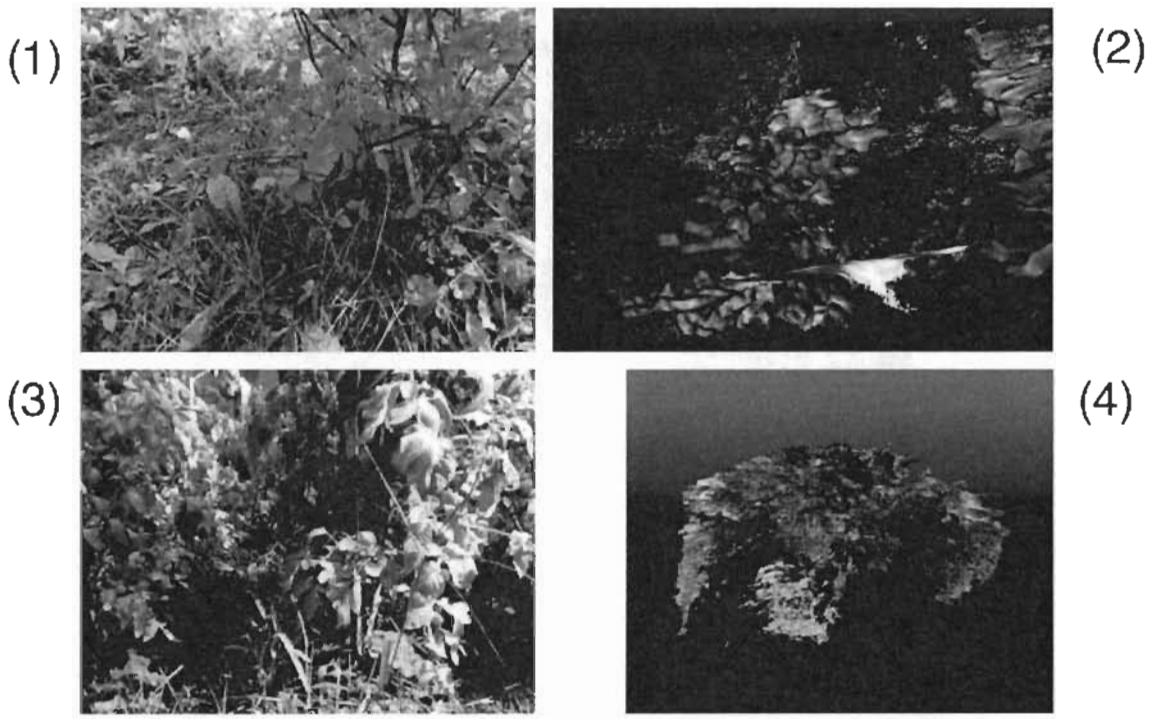


Figure 4.16 (1) Plante Bleuet. (3) Plante plant de tomate. (2) Reconstitution de la plante Bleuet. (4) Reconstitution de la plante tomate.

La deuxième étape de comparaison consiste à calculer la distance entre les nuages de points des deux plantes, on peut voir les résultats de la comparaison et le *ditance_compute()* dans les deux *figures 4.16 et 4.17*

Le score des résultats de comparaison est de 0 (voir la figure 4.17) ce qui signifie qu'il n'y a pas de caractéristiques similaires entre les deux plantes.

Figure 4.17 Résultats de distance_compute() affiché dans MeshLab.

```
Format = auto
Extension = ply
Format = auto
Extension = ply
Initial alignment
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Apply point-to-point ICP
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Transformation is:
[[ 0.862  0.011 -0.507  0.5  ]
 [-0.139  0.967 -0.215  0.7  ]
 [ 0.487  0.255  0.835 -1.4  ]
 [ 0.     0.     0.     1.   ]]

Apply point-to-plane ICP
registration::RegistrationResult with fitness=0.000000e+00, inlier_rmse=0.000000e+00, and correspondence_set size of 0
Access transformation to get result.
Transformation is:
[[ 0.862  0.011 -0.507  0.5  ]
 [-0.139  0.967 -0.215  0.7  ]
 [ 0.487  0.255  0.835 -1.4  ]
 [ 0.     0.     0.     1.   ]]
Score=0
```

Figure 4.18 Résultats de la comparaison Bleuet vs Tomate (Score=0).

D. Tomate avec Érable

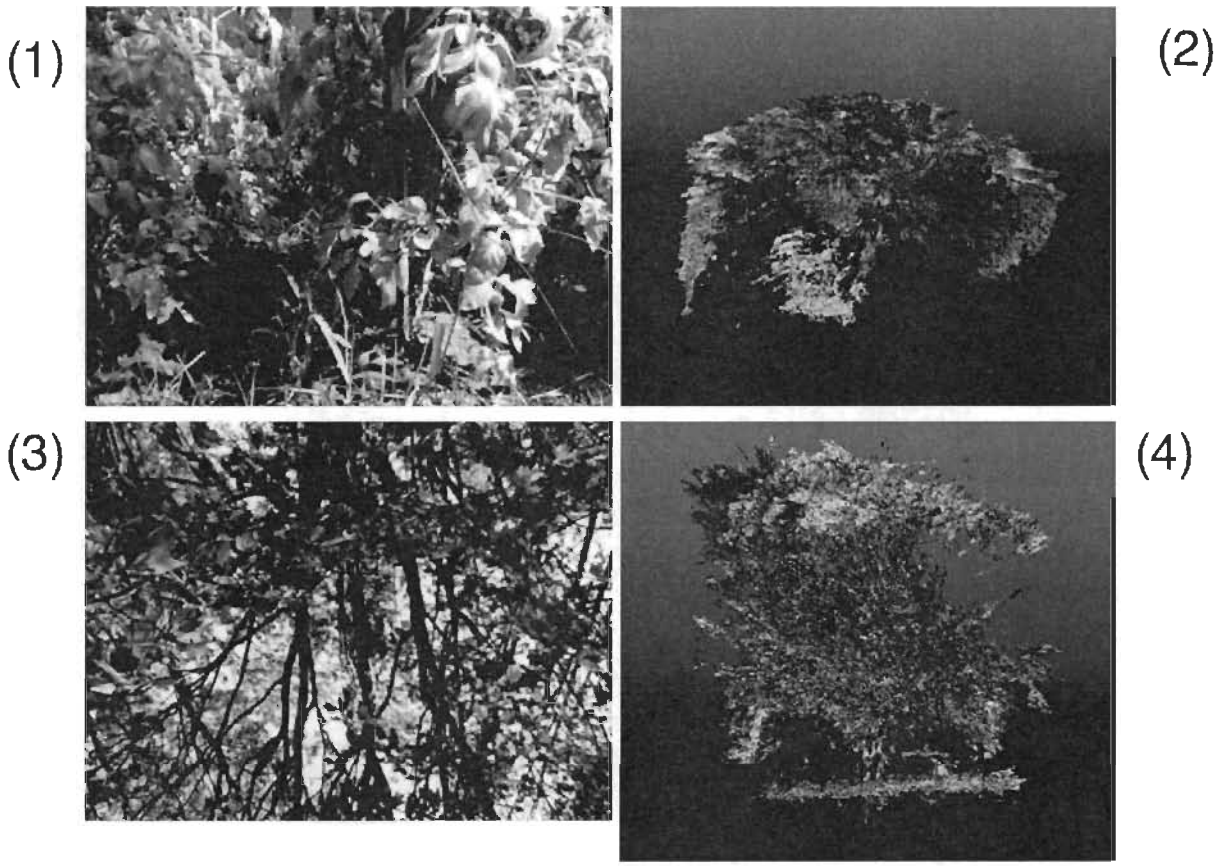


Figure 4.19 (1) Plante Tomate. (3) Plante Érable. (2) Reconstitution de la plante Tomate. (4) Reconstitution de la plante Érable.

le calcul de la distance entre les deux nuages de points aidera à améliorer les résultats de la comparaison comme on peut le voir sur la figure 4.18 un affichage de la distance_compute() entre la plante Tomate avec Érable.

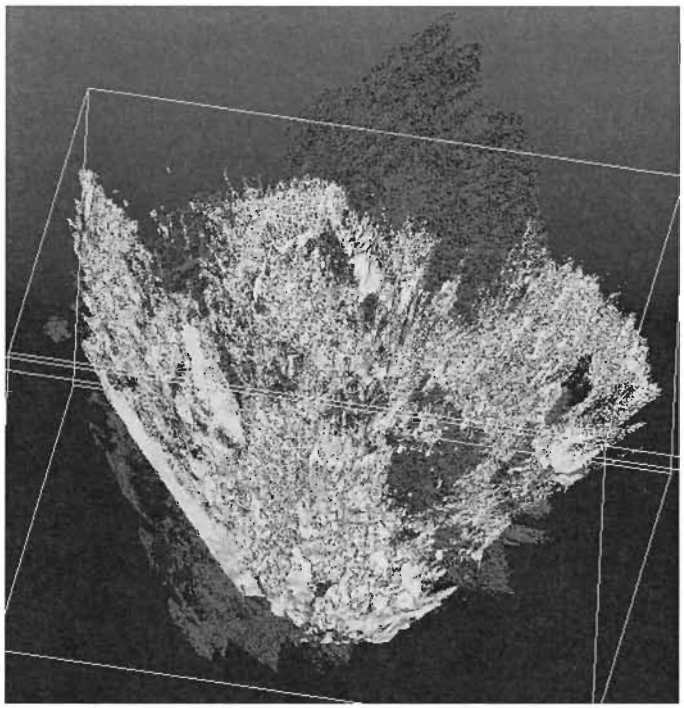


Figure 4.20 Résultats de distance_compute() affiché dans MeshLab.

```
Format = auto
Extension = ply
Format = auto
Extension = ply
Initial alignment
registration::RegistrationResult with fitness=1.566788e-03, inlier_rmse=1.170779e-02, and correspondence_set size of 12136
Access transformation to get result.
Apply point-to-point ICP
registration::RegistrationResult with fitness=1.789878e-03, inlier_rmse=1.155580e-02, and correspondence_set size of 13864
Access transformation to get result.
Transformation is:
[[ 0.86121008  0.02554752 -0.50780856  0.51242726]
 [-0.16758687  0.9576095  -0.23575223  0.7824878 ]
 [ 0.47933895  0.28734363  0.82888501 -1.36353935]
 [ 0.          0.          0.          1.        ]]

Apply point-to-plane ICP
registration::RegistrationResult with fitness=1.905941e-03, inlier_rmse=1.121362e-02, and correspondence_set size of 14763
Access transformation to get result.
Transformation is:
[[ 0.85020618  0.03229202 -0.52567649  0.5784188 ]
 [-0.18487282  0.95331911 -0.24021584  0.81934939]
 [ 0.49244846  0.3006242   0.81636975 -1.31280701]
 [ 0.          0.          0.          1.        ]]
Score=0
```

Figure 4.21 Résultats de la comparaison Tomate vs Erable (Score=0).

comme nous pouvons le voir sur la base des modèles 3D des deux plantes, nous trouvons que le (Score=0), ce qui signifie qu'il n'y a pas de caractéristiques communes entre les deux Plantes.

4.2.2 Comparaison avec les autres méthodes

Pour démontrer clairement l'utilité de notre reconstruction volumétrique basée sur la méthode, nous comparons d'abord le temps d'exécution et les besoins en mémoire de l'architecture déconvolutionnelle et du réseau VGG à deux résolutions différentes (comme l'explique le chapitre 03) de 32^3 est 128^3 . Pour estimer la reconstruction volumétrique avec un réseau déconvolutionnel, nous ajoutons simplement deux blocs de déconvolution supplémentaires à la ligne de base de déconvolution (voir Figure 3.7) a une fonction de base appropriée est remplacée pour générer des volumes 128^3 de 20^3 coefficient de la méthode proposée. Figure 4.4 montre le temps de formation, le temps d'inférence et le pic de la mémoire GPU nécessaire pour entraîner le réseau proposé basé sur VGG pour reconstruire les volumes aux deux résolutions à partir de 127 x 127 images.

Method	Resolution	Batch Size	Forward Time (Hz)	Train Time (Hz)	Memory (GB)
VGG-32	32^3	24	294 (4x)	80.75 (6.3x)	1.7
CNN-32	32^3	24	66.83 (1x)	12.63 (1x)	4.5
VGG-128	128^3	24	30.48 (0.45x)	22.99 (1.8x)	2.2
CNN-128	128^3	2	2.82 (0.04x)	0.19 (0.015x)	10.4

Figure 4.22 Indicateurs de performance utilisant les réseaux CNN et VGG à différentes résolutions.

En raison de la grande taille des volumes vidéo (fichiers BAG), notre réseau proposé est environ trois fois plus rapide pour l'inférence et plus de quatre fois plus rapide pendant l'entraînement, par rapport à notre modèle de base à une résolution plus petite de 32^3 avec une taille de batch de 24. De plus, les besoins en mémoire pendant l'entraînement sont considérablement réduits car les volumes intermédiaires plus grossiers ne sont pas prédits par notre décodeur. Lorsque la résolution est multipliée par quatre (dans chacune des trois dimensions), être 128^3 il devient évident que les réseaux de déconvolution 3D traditionnels deviennent insolubles. Déjà environ sept fois plus lent et trois fois

plus gourmand en mémoire, les réseaux de déconvolution ne peuvent désormais être entraînés qu'avec une taille de batch de 2 sur une carte GPU de 12 Go. L'entrainement par image augmente d'un facteur de plus de 50 par rapport à 32^3 la ligne de base de déconvolution et les performances du temps de test se dégradent de manière tout aussi drastique, rendant cette ligne de base inutilisable. Inversement, un décodeur VGG monocouche n'est que trois fois plus lent à s'entraîner lorsque la résolution est multipliée par quatre (dans chacune des trois dimensions) — Cependant, il reste encore plus rapide à former par rapport au réseau déconvolutionnel reconstruisant des volumes d'une résolution de 32^3.

Pour valider la précision de reconstruction 3D avec la méthode proposée, nous comparons les précisions de reconstruction de vue unique. Nous montrons que l'utilisation de notre couche ne dégrade pas la qualité des prédictions basse résolution mais permet un entraînement nettement plus rapide et donne de meilleures reconstructions haute résolution.

4.3 Conclusions

Nous présentons un système pour générer une reconstruction de scène 3D basée sur l'image en utilisant une caméra 3D Real-sense. Notre contribution comprend : 1) un système simple : avec caméra RGBD *low cost* ; 2) une méthode de segmentation automatique où la segmentation des objets est basée sur une bibliothèque de open-cv ; 3) un modèle 3D de haute qualité : un modèle de reconstruction 3D entièrement visible a été généré à partir de la méthode de mouvement structure-forme ; 4) Une algorithme optimal. Ce système de reconstruction 3D fournit une plate-forme de calcul à faible coût pour les systèmes robotiques qui travaillent dans l'agriculture pour éliminer les mauvaises herbes. Notre logiciel est construit sur une plateforme open source (open3D), ce qui sera bénéfique pour l'ensemble de la communauté.

5

Conclusion

La reconstruction 3D est depuis longtemps l'une des tâches clés dans le domaine de la vision par ordinateur, avec de nombreuses applications scientifiques et industrielles. Ces dernières années, l'apprentissage en profondeur a été appliqué à ce problème fondamental pour tenter de résoudre de nombreux problèmes qui existent avec les méthodes de reconstruction traditionnelles. Cependant, la reconstruction 3D à l'aide de l'apprentissage profond a ses propres défis. Dans ce mémoire, nous avons étudié plusieurs techniques pour relever les défis de l'inefficacité des calculs et de la rareté des données lors de l'entrainement de réseaux de neurones profonds pour la reconstruction 3D. Les premières méthodes de reconstruction 3D utilisant l'apprentissage en profondeur reposaient fortement sur l'utilisation de volumes 3D pour représenter la surface de l'objet ou d'une scène. Ces représentations ont été sélectionnées en raison de la simplicité d'intégration avec les réseaux de neurones convolutifs. Cependant, comparé à un CNN basé sur une image les représentations volumétriques nécessitent beaucoup plus de mémoire et de ressources de calcul.

Dans le chapitre 4, nous avons présenté une méthode pour reconstruire efficacement des volumes 3D haute resolution en forçant le réseau neuronal à apprendre une représentation dans le domaine des fréquences compressées à

l'aide d'une nouvelle couche de transformation cosinus discrète inverse. En remplaçant le réseau convolutif par une couche VGG, nous avons montré qu'il est possible d'augmenter simultanément la résolution volumétrique, réduire les coûts de calcul, et réduire l'utilisation de la mémoire GPU, sans aucune perte de qualité de reconstruction.

Bien que notre couche VGG aboutisse à des reconstructions efficaces et précises, cette méthode présente certaines limites. Premièrement, comme le réseau utilise une couche entièrement connectée pour produire le coefficient DCT (discrete cosine transform), le nombre de paramètres dans le modèle peut augmenter de façon exponentielle avec le nombre de coefficients DCT.

. Deuxièmement, le nombre de coefficients, et donc le taux de compression, doit être sélectionné à l'avance, ce qui donne un modèle qui ne peut reconstruire qu'à des taux fixes. Ce travail pourrait être encore amélioré en modifiant la reconstruction pour utiliser des correctifs ou des blocs « locaux », plutôt qu'un volume globalement compressé. Similaire aux techniques de compression d'image qui utilisent VGG [18], des régions de taille fixe plus petites du volume pourraient être représentées indépendamment.

Une autre préoccupation majeure liée à l'utilisation du l'apprentissage en profondeur pour la reconstruction 3D est la quantité limitée de données disponibles pour l'entraînement. Bien qu'il existe de nombreux ensembles de données d'images, de texte et audio disponibles sur Internet, les ensembles des données de modèles 3D pour les objets et les scènes, sont limités en qualité et en taille (resolution). Ceci est attribué aux difficultés de capture ou de modélisation 3D, nécessitant souvent de nombreuses heures pour acquérir une seule instance. Pour cela nous abordons la question des données en prenant nos données à l'aide de la caméra RealSense D435, il nous a fallu plusieurs mois pour collecter environ 100 vidéos 3D de haute qualité pour l'entraînement de notre réseau.

En pratique, nous avons constaté que la formation du réseau VGG était difficile car le modèle s'interrompait (Crush) et produisait de nouvelles vues qui ne pouvaient pas être utilisées pour la synthèse du nuage de points (voir Figure 4.1). Nous supposons que cela est causé par la méthode d'extraction des coordonnées de la caméra à partir des vidéos qui ont été utilisées pour l'ensemble

de données d'entraînement. Apprendre à reconstruire des scènes plutôt que des objets est une tâche difficile.

En outre, la capture de surfaces 3D précises pour la géométrie au niveau de la scène est beaucoup plus compliquée que l'acquisition de surfaces au niveau de l'objet, principalement en raison des nombreuses occlusions créés par la complexité de la scène. La reconstruction de scène se concentre uniquement sur la récupération des informations de surface 3D pour les pixels visibles et cela était traditionnellement réalisé par vision stéréo. Récemment, il y a eu un regain d'intérêt pour apprendre à prédire la profondeur à partir d'images monoculaires à vue unique [19, 20], grâce à la large gamme d'applications. En règle générale, ces modèles sont formés en utilisant des informations de profondeur de vérité terrain comme supervision. Dans la pratique cependant, l'accès à des ensembles de données RVB-D vastes et diversifiés est limité. Par conséquent, l'approche de supervision aotonome est apparue comme une méthode possible pour former des modèles sans étiquettes.

Dans ce mémoire, nous avons discuté de plusieurs problèmes liés à l'utilisation de l'apprentissage en profondeur pour la reconstruction 3D à vue unique. Notre travail utilise une couche VGG pour remédier aux inefficacités de calcul avec la reconstruction volumétrique. En utilisant cette méthode, nous avons montré une amélioration de l'ordre de grandeur de la résolution volumétrique, consommation de mémoire, temps d'entraînement et vitesse d'inférence. En combinant les progrès récents de la prédiction de vue et de l'estimation de la profondeur, nous avons également abordé les problèmes liés à la disponibilité limitée des données d'entraînement pour la reconstruction 3D. Tous ces résultats ouvrent potentiellement des pistes pour comprendre et interpréter le processus de reconstruction 3D à l'aide de techniques d'apprentissage en profondeur. De plus, pour que les systèmes autonomes prennent des décisions intelligentes, il est important que le système comprenne quelles prédictions sont utiles et fiables.

Bibliographie

[1] V. USENKO, J. ENGEL, J. STU¨CKLER, and D. CREMERS, Reconstructing Street-Scenes in Real-Time From a Driving Car. (2014)

[2] T. ZHOU, M. BROWN, N. SNAVELY, E. BOYER, and D.G. LOWE, Unsupervised Learning of Depth and Ego-Motion from Video, in IEEE International Conference on Computer Vision Systems, 2006.

[3] A. ZENG, S. SONG, K.T YU, and others, Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching, 2016.

[4] J. STARCK and A. HILTON, Correspondence labelling for wide-timeframe free-form surface matching, in International Conference on Computer Vision, 2007.

[5] M. LEVOY, K. PULLI, B. CURLESS, S. RUSINKIEWICZ, D. KOLLER, L. PEREIRA, M. GINZTON, S. ANDERSON, J. DAVIS, J. GINSBERG, J. SHADE, and D. FULK, *The Digital Michelangelo Project: 3D Scanning of Large Statues*, in SIGGRAPH, 2000.

[6-a] J. BATLLE, E. MOUADDIB, and J. SALVI, Recent progress in coded structured light as a technique to solve the correspondence problem: a survey, Pattern Recognition, (1998).

[6-b] D. SCHARSTEIN and R. SZELISKI, High-Accuracy Stereo Depth Maps Using Structured Light, in IEEE Conference on Computer Vision and Pattern Recognition, 2003.

[6-c] K. SATO and S. INOKUCHI, *Three-dimensional surface measurement by space encoding range imaging*, Journal of Robotic Systems, (1985).

[7-a] R. WOODHAM, *Photometric method for determining surface orientation from multiple images*, Optical Engineering, (1980).

[7-b] C. HERNANDEZ, G. VOGIATZIS, and R. CIPOLLA, *Probabilistic Visibility for Multi-View Stereo*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[8-a] D. SCHARSTEIN and R. SZELISKI, *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms*, International Journal of Computer Vision, (2002)

[8-b] *H. HIRSCHMÜLLER and D. SCHARSTEIN, Evaluation of Stereo Matching Costs on Images with Radiometric Differences, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2009)*

[8-c] P. N. BELHUMEUR, A Bayesian approach to binocular stereopsis, International Journal of Computer Vision, (1996)

[8-d] V. KOLMOGOROV and R. ZABIH, *Computing Visual Correspondence with Occlusions via Graph Cuts*, in International Conference on Computer Vision, 2001.

[9-a] S. SEITZ, B. CURLES, J. DIEBEL, D. SCHARSTEIN, and R. SZELISKI, *A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms*, in IEEE Conference on Computer Vision and Pattern Recognition, 2006

[9-b] G. VOGIATZIS, C. H. ESTEBAN, P. H. S. TORR, and R. CIPOLLA, *Multiview Stereo via Volumetric Graph-Cuts and Occlusion Robust Photo-Consistency*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2007)

[9-c] *Y. FURUKAWA and J. PONCE, Accurate, Dense, and Robust Multi-View Stereopsis, in IEEE Conference on Computer Vision and Pattern Recognition, 2007.*

[9-d] D. BRADLEY, T. BOUBEKEUR, and W. HEIDRICH, *Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[10-a] *A. LAURENTINI, The Visual Hull Concept for Silhouette-Based Image Understanding, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1994)*

[10-b] T. KANADE, P. RANDER, and P. J. NARAYANAN, *Virtualized Reality: Constructing Virtual Worlds from Real Scenes*, IEEE Multimedia, Immersive Telepresence, (1997)

[10-c] T. MATSUYAMA, X. WU, T. TAKAI, and S. NOBUHARA, *Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video*, Computer Vision and Image Understanding, (2004)

[10-d] *X. WU and T. MATSUYAMA, Real-time active 3D shape reconstruction for 3D video, in 3rd International Symposium on Image and Signal Processing and Analysis, 2003*

[11-a] S. BISCHOFF L.P. KOBBELT, Isosurface Reconstruction with Topology Control, Lehrstuhl fu¨r Informatik VIII, RWTH Aachen, (2007)

[11-c] X.F. HAN, H. LAGA, M. BENNAMOUN Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era, IEEE, (2010)

[12-b] R. SZELISKI, Rapid Octree Construction from Image Sequences, Computer Vision, Graphics and Image Processing: Image Understanding, (1993)

[12-c] B. RAYNAL, M. COUPRIE, Rapid Octree Construction from Image Sequences, Computer Vision, Graphics and Image Processing: Image Understanding, (1993)

[13-a] *S. AZERNIKOV, A. MIROPOLSKY, and A. FISCHER, Surface reconstruction of freeform objects based on multiresolution volumetric method. the 8th ACM symposium on Solid modeling and applications.*

[13-b] *J. WANG, M. OLIVEIRA, A. KAUFMAN, Reconstructing Manifold and Non-Manifold Surfaces from Point Clouds. (2011)*

[14] K. *AUDRIUS, M. RYTIS, 3D Object Reconstruction from Imperfect Depth Data Using Extended YOLOv3 Network. (2020)*

[14-a] *Pound, M.P.; French, A.P.; Murchie, E.H.; Pridmore, T.P. Automated recovery of three-dimensional models of plant shoots from multiple color images. Plant Physiol. (*2014, *166, 1688–1698).*

[14-b] *Cremers, D.; Kolev, K. Multiview stereo and silhouette consistency via convex functional over convex domains. IEEE Trans. Pattern Anal. Mach. Intell. (*2011, *33, 1161–1174)*

[15] Q. *GUAN, Y. WANG, Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study. (2019)*

[16] He. *KAIMING, X. ZHANG, S. REN, Deep Residual Learning for Image Recognition. Microsoft Research (2020)*

[16-a] Q. *Yi ZHOU, J. PARK, V. VLADLEN, Open3D: A Modern Library for 3D Data Processing.*

[17] *Ni, Z.; Burks, T.F.; Lee, W.S. 3D Reconstruction of plant/tree canopy using monocular and binocular vision. J. Imaging (*2016, *2, 28.)*

[18] *W, JIASONG.; H, REN.; Y, KONG. Compressing Complex Convolutional Neural Network Based On An Improved Deep Compression Algorithm (*2000*)*

[19] *D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In ICCV, 2015.*

[20] *D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In NeurIPS, 2014.*