



3D image-based reconstruction for patient position detection in percutaneous surgery

TER report submitted by

ARNAUD GOETZ

arnaud.goetz@etu.unistra.fr

MASTER INFORMATIQUE I3D

PARCOURS CURSUS MASTER EN INGÉNIERIE, MENTION IIRVIJ

IMAGE, REALITÉ VIRTUELLE, INTERACTION ET JEUX

Supervised by

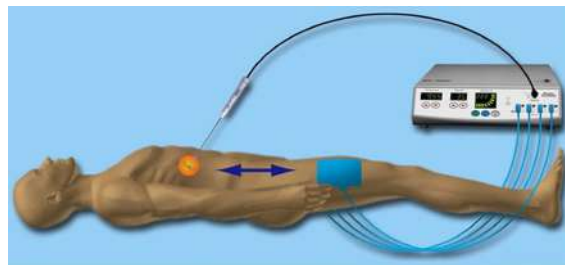
CAROLINE ESSERT

essert@unistra.fr

Contents	1
1 Introduction	2
1.1 Context	2
1.2 Work objectives	2
2 Literature review	3
2.1 Traditional Methods : Photogrammetry	3
2.1.1 Structure from Motion	4
2.1.2 From Sparse to Dense: Multi-View Stereo Reconstruction . . .	5
2.2 AI in Modern 3D Representations	7
2.2.1 NeRF	7
2.2.2 Gaussian Splatting	8
2.2.3 Depth Maps	9
2.3 Medical Use Case	11
2.3.1 CAScination : Volumetric reconstruction	11
3 Implementation	12
3.1 Set up installation	12
3.2 Development	13
3.2.1 Data Preparation	13
3.2.2 Camera Calibration	13
3.2.3 Data Segmentation	14
3.2.4 Depth inference	14
3.2.5 Point cloud generation	14
3.2.6 Point cloud registration	15
4 Conclusion	18
5 Bibliography	19

1.1 Context

This work focuses on the reconstruction of 3D shapes from 2D images. The aim is to generate a point cloud from a photo or video by transforming these images into depth maps, in order to reconstruct the shape of the patient’s skin. This pointcloud will be the basis for a future work of recalibrating the surface created from the pointcloud with the shape of the patient’s actual skin.



1.2 Work objectives

This thesis begins with a literature review of state-of-the-art methods in the field. It covers the technologies and work already carried out in the field of 2D image based 3D reconstruction. This part consists of a literature search leading to a summary of the different approaches.

3D image-based reconstruction for patient position detection in percutaneous surgery 2

Introduction The reconstruction of 3D geometry from photos is a challenge in the field of Computer Vision. This area of study has diverse applications such as 3D mapping in archeology, navigation with autonomous cars, computational photography, video games and medical operations. Modeling the 3D geometry of objects or environments poses challenges and has led to the development of many approaches, including active techniques with sensors, and passive image-based approaches.

This literature review focuses on **passive** image-based methods, which offer a rapid way of capturing 3D structure compared to other techniques. Passive techniques require no interaction with the scene and rely on cameras to capture images, where all the information is contained. In contrast, active techniques involve the use of sensors or signals, such as Lidar, radar, ultrasound, IR, or laser, to interact with the scene and extract data.

Traditional methods offers reliable techniques for generating 3D models. Among these traditional approaches, photogrammetry is one of the most widely studied and applied reconstruction method.

Image-based 3D reconstruction estimates the 3D shape of an object or scene from multiple photos, assuming known materials, viewpoints, and lighting. Without these assumptions, the problem becomes complex, as different 3D shapes can create the same images. However, with certain conditions, modern techniques can achieve highly detailed results. Multi-view stereo (MVS) is a popular method that uses multiple images and relies on camera parameters, often calculated through Structure from Motion (SfM). A typical MVS process includes these steps:

1. Collect images
2. Estimate camera parameters (if not known)
3. Reconstruct 3D geometry
4. Optionally reconstruct materials.

Camera parameters MVS algorithms requires that every image match a camera model that describes how a 2D point from an image is projected in a 3D world. The common model is called the pinhole camera model.

$$C = \begin{pmatrix} fx & s & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \end{pmatrix}$$

Where C can be decomposed in two matrices, the matrix with the intrinsic

parameters such as vertical and horizontal focal lengths (f_x, f_y) and principal point (c_x, c_y). The second one is the extrinsic matrix that gives translations and rotations parameters. In MVS, we suppose these parameters to be known and the scene to be static at any time. A problem arises if the camera parameters are not known, which is the case in the vast majority of datasets.

2.1.1 Structure from Motion

Structure from Motion is a technique used to compute camera parameters of an ordered or unordered set of images. In that case, the process is slightly different (Fig 2).

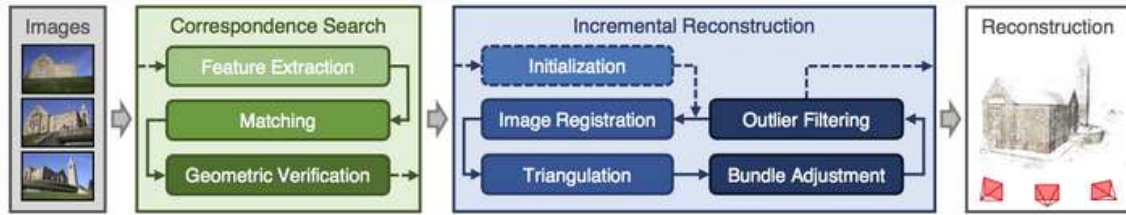


Figure 2: Incremental SfM pipeline

The process begins by detecting and matching key points across all images, then selects an initial image pair with overlap to estimate their camera positions and triangulate some 3D points. From there, it adds one image at a time, estimating each new camera's position using matches to existing 3D points and triangulating new points to expand the model. After each step, it refines the reconstruction using bundle adjustment, which is used to minimize errors in camera positions and 3D points. This process repeats until all images are included.

Early Structure from Motion research focused on two and three-view geometry under the assumption of rigid scenes [12]. A key advancement was the use of RANSAC [2] to handle the outliers in initial matches, which are common due to factors like occlusions and distortions. It helps excluding false matches, improving accuracy.

Initially, SfM methods were designed for structured image sequences, such as video frames, where sequences helped establish matches efficiently. Some modern Multi-View Stereo (MVS) applications, such as Google StreetView [1], still use structured image sequences. However, many applications now rely on unordered image sets, which pose new challenges for feature matching and scene reconstruction.

The introduction of robust feature detection techniques, such as corner detection [11] and blob detection [6], significantly improved SfM performance on unstructured datasets. These detectors ensure consistent feature matching across images despite variations in lighting, distortions and occlusions. However, pairwise matching in unstructured datasets is computationally expensive, necessitating more efficient matching strategies.

To address this, modern SfM pipelines employ advanced feature-matching algorithms reducing computational complexity while maintaining accuracy.

Two feature matching methods that reflect recent improvements in algorithms are:

1. **Vocabulary Tree Matching:** This method quickly finds similar images in large datasets by using a pre-trained vocabulary tree. It groups images based on visual features and matches each one to its closest neighbors. This makes the process faster and more efficient, especially for thousands of images [10].
2. **Transitive Matching:** This method extends existing feature matches using transitive relations. If an image A matches image B and image B matches image C, the algorithm attempts to match A to C directly. This enhances the feature matching graph.

2.1.2 From Sparse to Dense: Multi-View Stereo Reconstruction

Once the camera poses and a sparse 3D point cloud have been estimated using SfM, the next step is Multi-View Stereo (MVS). While SfM is used for estimating camera poses and sparse geometry, it is not enough to get a representation of the scene. To generate a dense 3D model, we need to recover more information, which is where MVS comes in. MVS methods can be categorized based on the nature of the scene representation:

Depth Map Reconstruction : The MVS based on depth maps has emerged as the main approach due to its flexibility [15]. Depth map-based 3D reconstruction is popular because it is simple and works like a "divide and conquer" strategy. Instead of solving the whole 3D scene at once, it focuses on one image at a time. For each photo, it guesses the depth by comparing it with a few neighboring views, as illustrates in Fig. 3. These images depth guesses are later merged into a single, detailed 3D model with a technique called depth map fusion. While patch-based methods build surfaces piece-by-piece, depth maps act like stacking "layers" from individual images, making them faster for large datasets but sometimes noisier in overlapping areas.

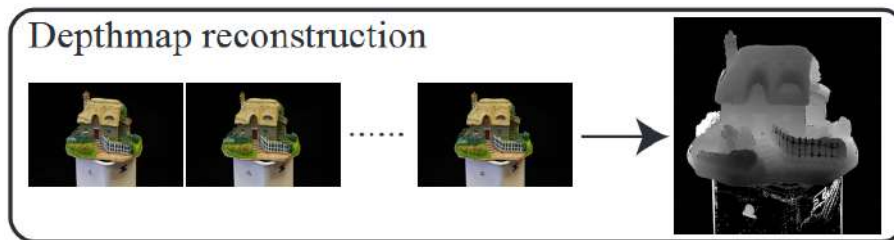


Figure 3: Depth Map representation

Point Cloud Densification : Patch-based 3D reconstruction turns basic 3D points from SfM data into a detailed surface by building small, flat "patches" that act like puzzle pieces. Instead of guessing depth per image (depth maps, which can leave gaps) or splitting space into rigid 3D grids (like volumetric methods, which need huge memory), patches grow smartly. They start at key points, adjust their position and angle to match colors and patterns across images, and spread to fill empty areas while ignoring mismatches. This makes them great for complex scenes (such as messy rooms or curved objects) because they adapt to shapes naturally, avoid noise, and work for both tiny objects and huge spaces. They're especially useful for tasks like historical sites or mapping cities, where precision and flexibility matter most but they are computationally expensive.

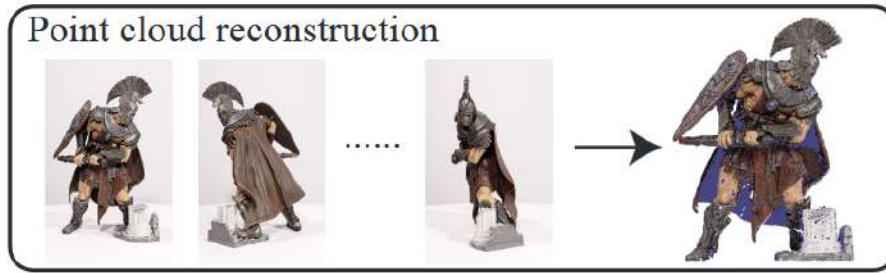


Figure 4: Point Cloud representation

Volumetric Reconstruction : Volumetric surface extraction is a versatile method that can utilize 3D data from various sources such as depth maps, MVS point clouds, laser-scanned 3D points, or a combination of these inputs.

Generally when using depth maps, they are transformed into a 3D volumetric representation using a signed distance function (SDF). This function assigns each voxel in a 3D grid a distance value to the nearest surface, with a sign indicating whether the voxel is inside or outside the object. Large memory is needed, which is often handled with data structures like octrees. Compared to other methods, volumetric approaches give a consistent result for small scenes, though they usually need more computing power.

In the end, choosing a method depends on the trade-offs between resolution, memory, and how complex the scene is. In recent years, various SfM and MVS pipelines have been developed, each offering distinct features. Among the most widely used are ColMap [9] and AliceVision [8].

Summary Traditional 2D image-based reconstruction methods, such as Structure from Motion and Multi-View Stereo, are still widely used in areas like urban modeling and medical applications. They can produce 3D models without requiring training data, making them practical and fast. However, these methods rely heavily on rich textures and distinct visual features across images; they often struggle with

flat, uniform, or reflective surfaces where feature matching becomes unreliable. The next section highlights how AI tools can replace or merge with traditional techniques.

2.2 AI in Modern 3D Representations

Recent progress in Machine Learning has changed how we reconstruct 3D models from 2D images. Older methods seen in 2.1 relied heavily on geometric constraints and iterative steps, but AI techniques are now improving and even replacing these approaches. Deep learning can better solves challenges like ambiguity in an image, objects blocking each other (occlusion), lights variations and can adapt to many types of scenes.

In this section are presented AI-based techniques such as neural radiance fields (NeRF), depth prediction networks, and Gaussian Splatting.

2.2.1 NeRF

Neural Radiance Fields [7] is a technique in 3D reconstruction, offering a new approach compared to traditional methods seen in 2.1. While traditional techniques use an explicit representation such as voxel grids or point clouds to represent the geometry of a scene, NeRF instead encodes the scene implicitly with a continuous function.

This function is parameterized by a neural network, which maps 3D spatial coordinates (x, y, z) and viewing directions (θ, ϕ) to color (RGB values) and a volume density (σ) . This means that, rather than storing geometry directly, NeRF learns how light and views can influence the scene. The Figure 5 illustrates such model.

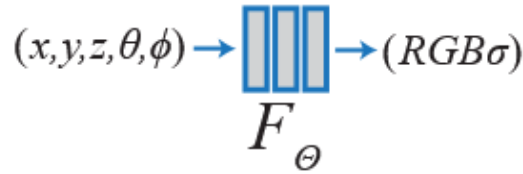


Figure 5: Architecture of NeRF [7]

Like traditional methods, the reconstruction process begins with capturing multiple images of the scene from different viewpoints. These images must be associated with camera poses, which can be obtained using SfM if they are not provided. Once the data is prepared, rays are projected through each pixel of these images into 3D space. Along each ray, the model samples a number of 3D points and feeds them into the neural network.

For every sampled 3D point along a ray, the network predicts a color and a density value. These outputs are then combined using a volume rendering equation, which simulates how light behave through the scene. This process generates a pixel color and density for each ray. During training, these rendered pixels are compared to the

actual pixels from the original images to minimize a cost function. Over time, the model learns to reproduce the entire scene from any viewpoint.

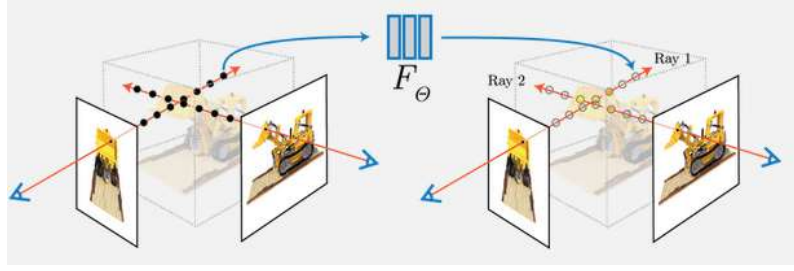


Figure 6: Architecture of NeRF [7]

Then, the Marching Cubes algorithm can be used to turn NeRF data into a visible 3D mesh use (Figure 7). First, the space is divided into a grid of voxels. Each corner of these cubes have a density value from the trained NeRF model. A density threshold can be picked to decide what counts as solid (above the threshold) and what's empty. For each cube, the algorithm checks its corners against the threshold and create triangle shapes where the density changes, this is where the surface is. As it goes through the whole grid, it builds up a full 3D mesh made of these triangles. The resolution of the grid affects how detailed the mesh is, higher resolution gives more detail but takes more computing time. Choosing the right threshold helps reduce noise while keeping important features.

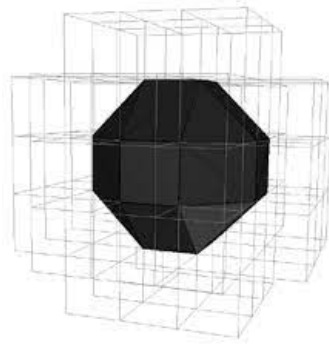


Figure 7: Polygonising a neural radiance field

2.2.2 Gaussian Splatting

3D Gaussian Splatting is a radiance field method for 3D scene rendering. Unlike Neural Radiance Fields (NeRFs), it replaces neural networks with an explicit representation using optimized 3D Gaussians, enabling real-time rendering. It is a rendering technique and is not used for mesh extraction or for a dense point cloud extraction unlike NeRFs where Marching Cubes can be used to obtain a mesh.

- Explicit representation: Scene structure is defined by anisotropic 3D Gaussian primitives with learnable parameters (position, covariance, opacity, and spherical harmonics coefficients).

- Real-time performance: Rasterization-based rendering avoids costly volumetric neural network inference.
- Direct editing: Supports manipulation of Gaussians (position, scale, rotation) without retraining.

This approach achieves NeRF-like view synthesis quality while enabling interactive applications due to its differentiable splatting pipeline. The process detailed in the paper [4] and simplified in Figure 8 can be summarize as follow :

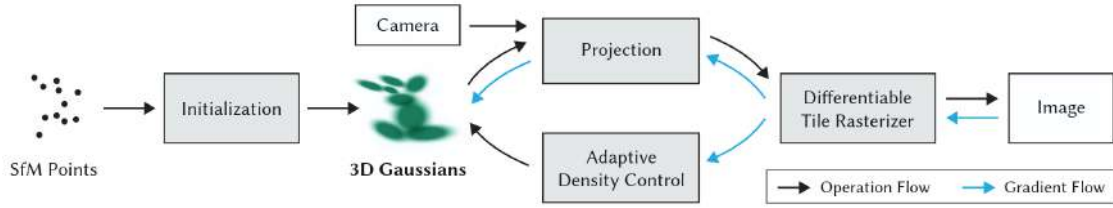


Figure 8: Gaussian Splatting process [4]

First, like NeRFs or photogrammetry, the scene is captured with photos from multiple viewpoints. Then, SfM is used to compute camera poses and get a sparse point cloud. For each point of the scene, a 3D Gaussian "splat" is created. 3D Gaussians are a generalization of 1D Gaussians to 3D space. They are basically ellipsoids in 3D space, characterized by a center, a scale and a rotation.

The idea is now to optimize those Gaussians from the camera views and compare to the ground truth image. Like Neural Networks, a gradient descent can be used to optimize positions, size, orientation to match the images. The idea to obtain a dense representation is to add or remove Gaussians to better represent the scene, this process is called Density control. Each 3D Gaussian is optimized with a view-dependent color and opacity. When blended together, they create a full model that can be rendered from any angle.

Summary NeRFs and Gaussian Splatting show how AI is changing 3D reconstruction. NeRFs can be used for tasks like novel view synthesis and mesh reconstruction, while Gaussian Splatting is focused on fast, high-quality rendering but it doesn't give a mesh or point cloud we can edit. These methods prove that AI can either replace or support traditional 3D techniques, depending on the goal. The next section is about how AI can also improve older pipelines for example, by generating depth maps to speed up processing without changing the whole system.

2.2.3 Depth Maps

Depth maps plays a critical role by representing the distance between the camera and objects in an image, basically with grayscale image where pixel intensity encodes depth. Depth maps are essential in 3D reconstruction pipelines as they provide geometric information from 2D images, to understand scene structure and object

However, creating large-scale depth datasets is expensive and impractical with those traditional methods. To address this, Monocular Depth Estimation (MDE) uses deep learning to predict depth from a single image. However, MDE faces challenges due to the high variability of scenes and large, labeled datasets.

It means creating a general model capable of predicting depth from a single image in any situation. Unlike traditional models that need labeled training data, this model learns from unlabeled monocular images with millions of everyday images taken with a single camera, without any depth annotations. The model is designed to do strong zero-shot generalization, meaning it can estimate depth even for images and scenes it has never encountered during training.

- The architecture uses Transformers instead of classical convolutional neural network
- Data Augmentation : Unlabeled images are labeled using a pretrained "Teacher" model
- Knowledge distillation: A "Student" is trained on both labeled and pseudo-labeled data.
- Semantic enrichment: Semantic features are information that describe what is in the image like identifying objects and their relationships.

Summary We have seen that new AI techniques can be used in 3D reconstruction pipelines, such as generating depth maps for dense reconstruction, because depth maps provide semantic information and depth values to infer the 3D structure of a scene. After exploring traditional photogrammetry (2.1) and the new AI techniques in 3D reconstruction (2.2), the third part explores concrete 3D reconstructions in medical imaging, diagnostics, and surgical planning.

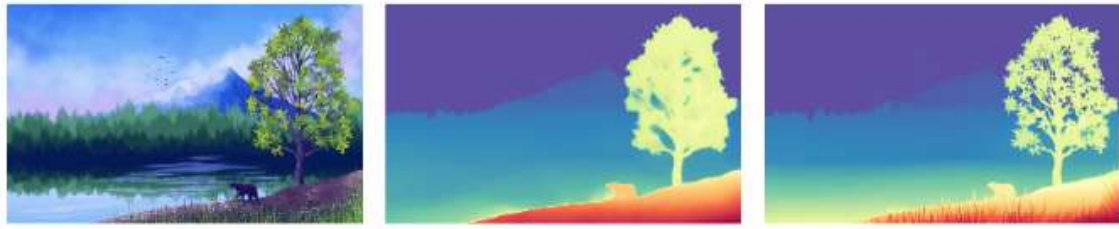


Figure 9: Predictions of models trained on pseudo-labeled data (middle) and synthetic images (right) [14]

2.3 Medical Use Case

2.3.1 CAsCination : Volumetric reconstruction

In the context of hepatocellular carcinoma (HCC) treatment, the paper "New horizons in ablation therapy for hepatocellular carcinoma" [3] provides insights into Cascination technology used for image-guided tumor ablation that integrates 3D reconstruction, virtual targeting, and needle placement. The technology is private (CAsCination AG), so full details of the method are not available. This description relies on limited information and images and may include errors or assumptions. The process include steps.

Image acquisition Image acquisition uses CT or CBCT scans to capture detailed images of the liver, tumors, blood vessels, and surrounding areas. These scans are chosen for their clarity and compatibility with operating rooms. During the scan, the patient briefly holds their breath to minimize blurring caused by breathing, ensuring sharp images for precise surgical planning.



Figure 10: A CT scan of a hepatocellular carcinoma

Segmentation Images are segmented using AI or manual annotations because segmentation isolates relevant structures for planning needle paths and ensures safety by avoiding critical organs.

3D model The 3D reconstruction involves volumetric reconstruction using triangulation (marching cubes or voxel-based surface rendering). It converts 2D scans images into a usable 3D virtual model. The segmented regions for patient skin are

basically transformed into a mesh surface model. Volume rendering may also be used for internal structure visualization.

Virtual Planning The virtual planning is set after the surgeon defines a target point (tumor center or edge) and a entry point on the skin. An algorithm computes optimal needle paths while displaying all structures in 3D. Surgeons can adjust and verify angles interactively.

Calibration and Registration The system uses a binocular camera and reflective markers attached to the patient’s skin and surgical tools. This setup allows the virtual 3D model (created from scans) to align with the patient’s actual anatomy during surgery. The system could potentially only use markers to determine 3D positions. By placing markers on the patient’s skin and tools, the binocular camera tracks their precise locations in real time. Each marker acts as a reference point, allowing the system to calculate the exact 3D coordinates of every tracked point.

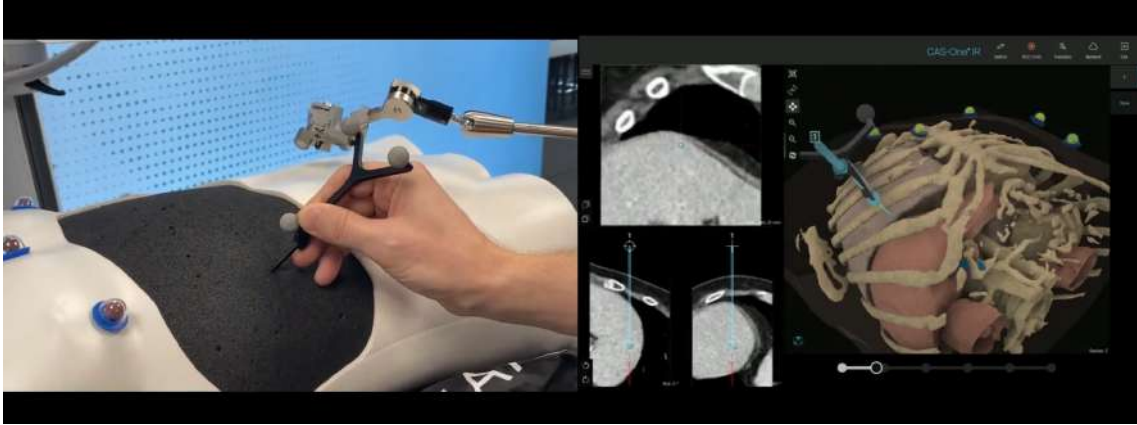


Figure 11: Interactive 3D model with markers for registration

3 Implementation

This part consists of implementing the DepthAnything [13] and Open3D [16] libraries to create a point cloud from photos.

3.1 Set up installation

I used a Conda environment to manage all dependencies. The main libraries installed were torch and torchvision for deep learning, opencv-python for image processing, and matplotlib for visualization. Other necessary packages were also added to support data loading, training, and evaluation. This setup helped keep the environment stable and consistent.

3.2.1 Data Preparation

To create the dataset, I used a teapot or a pillow as the main object and captured images from multiple angles. The images were taken under good lighting conditions, but that may not reproduce reality in an operating room with artificial lights. Such images are displayed in Figure 12

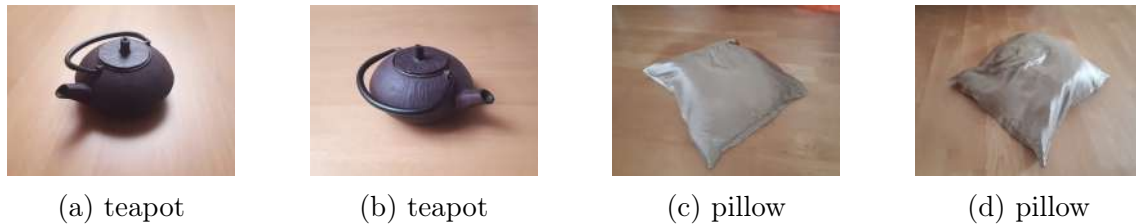


Figure 12: Dataset examples

Open3d provides multiples functions useful for creating point cloud from images. The main constraint that I encountered was the need of camara intrinsic and extrinsic parameters. As my phone was used to take pictures, I did not know those parameters and need to approximate them with a camera calibration.

Camera calibration is a process determining the internal parameters of a camera. These parameters include the focal length, optical center, and lens distortion coefficients. The idea is to establish a model that maps 3D points in the real world to 2D points in the image. Calibration is typically done by capturing images of a known pattern (such as a checkerboard) from different angles and using these images to estimate the camera's parameters.

A script using Python and OpenCV was used. Such images are displayed in Figure 13.

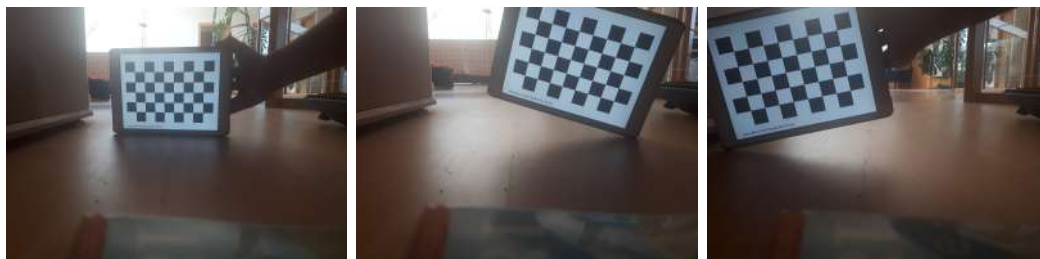


Figure 13: Calibration examples

The estimated parameters were $fx = 3739$, $fy = 3743$, $cx = 2328$ and $cy = 1823$.

$$C = \begin{pmatrix} 3739 & s & 2328 \\ 0 & 3743 & 1823 \\ 0 & 0 & 1 \end{pmatrix}$$

3.2.3 Data Segmentation

To isolate the object and improve the quality of the point cloud, I used Meta's Segment Anything model [5] for data segmentation. My script allowed me to separate the teapot or the pillow from the background in each image. By removing irrelevant background pixels, I thought it will reduce noise and focused the reconstruction process on the object. I encountered difficulties with segmenting images that contain too many distinct features, such as the teapot. My script is based on the central pixel of the image to extract the main object. The problem is what if the central pixel belong himself to an subpart of the main object ? Results are shown in Figure 14.

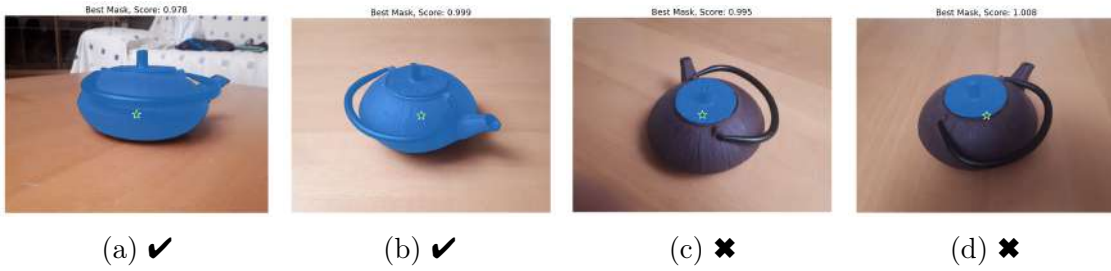


Figure 14: Segmentation examples

3.2.4 Depth inference

To create pointclouds, open3D needs depth information for each image. In this case, we need monocular depth. As seen in 2.2.3, DepthAnything is a recent AI model that can infer depth on single images. They provide multiple models, I choose the large model because the computation cost during inference is not too high (1 sec/image). Figure 15 represents the outputs of my script using DepthAnything V2. The output is a grayscale image but DepthAnything returns by default a rgb image. Open3D documentation is unclear about what type of data is needed for depth images, some code exemples provides a single channel image, that is why I decided to go with grayscale images.

3.2.5 Point cloud generation

The previous sections are the tools leading to the creation of a pointcloud. Figure 17 shows examples of pointclouds. There are issues I did not fixed. First, I observe a distortion on the ground. I suppose it either come from wrong camera parameters or the depth information is not precise enough or may not be interpreted correctly.







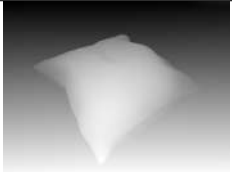
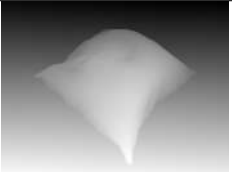
Type	Théière 1	Théière 2	Coussin 1	Coussin 2
Image				
Depth				

Figure 15: Depth Inference from images using DepthAnything V2

I tried to manually adjust camera parameters but no solution seems to appear "realistic". I also tried to use the default camera parameters given by Open3D. The pointclouds are still very similar. I could not get a pointcloud where the ground is flat. Figure 16 illustrates the key issue with my implementation that creates distortion.

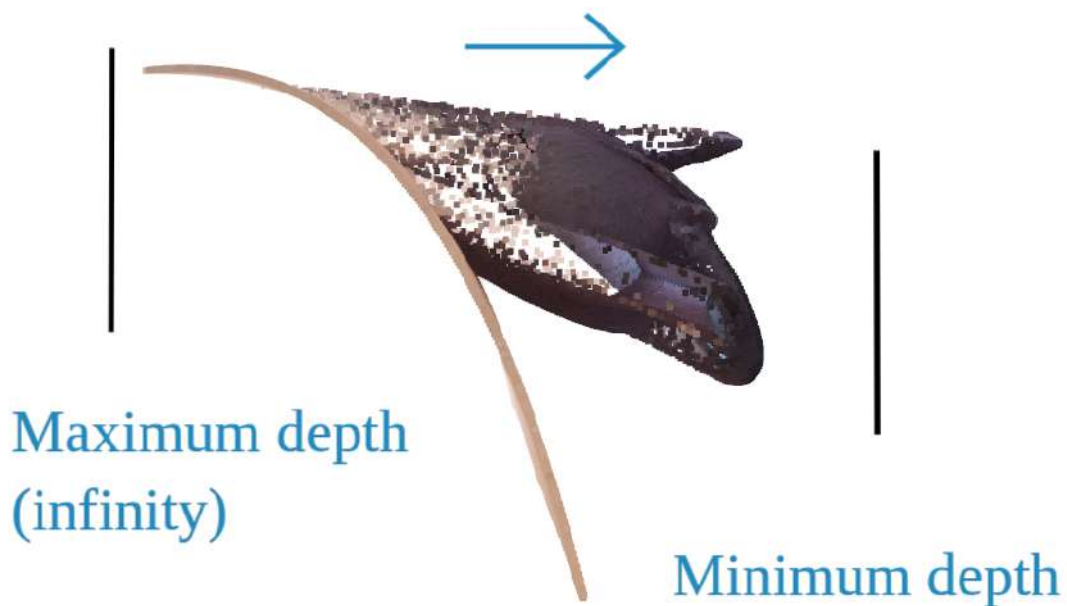


Figure 16: Side view from teapot pointcloud

3.2.6 Point cloud registration

When a pointcloud is created for each image, the idea is to merge those pointclouds into one. Each pointcloud is created in a local space, the global coordinates are not





















Type	Théière 1	Théière 2	Coussin 1	Coussin 2
Image				
Ground				
Ground v2				
Segmented				
Segmented v2				

Figure 17: Pointclouds

known. To solve the problem open3d suggest to do a Iterative Closest Point (ICP) to estimate camera poses. First I tried with just two cloudpoint. Open3d suggest also to do a global registration on two pointcloud before doing a local registration (ICP). Figure 18 shows the two pointcloud before global registration with a random initialization and after the transformation.

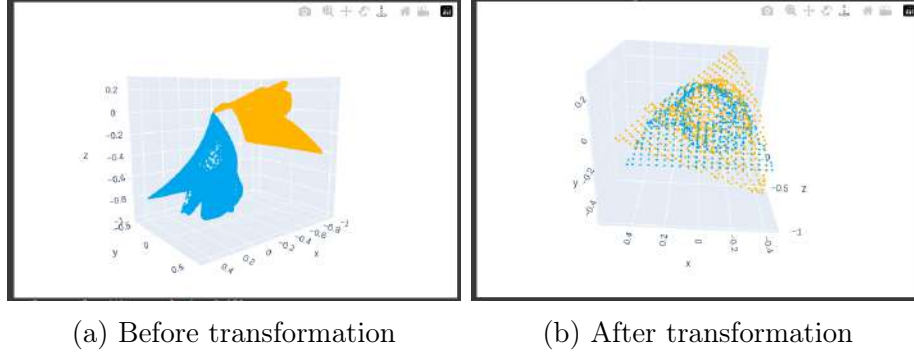


Figure 18: Global registration

Then I tried to do a local registration to get only one merged pointcloud with two different methods, a registration through the use of Iterative Closest Point "Point-to-Point" and "Point-to-Plane". Results are displayed in Figure 19. In both cases I get a pretty poor result. My implementation stops at this point because I have not been able to fix this problem. Ideally, these process should be carried out iteratively by performing an ICP image after image to complete the final point cloud.

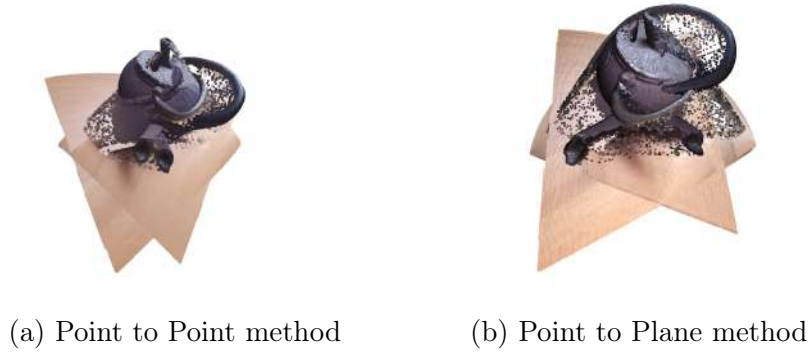


Figure 19: Iterative Closest Point registration

4 Conclusion

Summary of the work My TER was spent halfway through the bibliography and the other half on implementing a solution. The state-of-the-art required a lot of investment and time, but I believe it's the most accomplished part of my work. The implementation with Open3D is not yet complete, in the sense that my work is not directly usable in practice and requires improvements.

Personal conclusion For the bibliography, I found it very easy to understand the papers that dealt with AI methods, unlike the other subjects which were completely new to me, especially photogrammetry, which seemed to me to be a very vast field and difficult to deal with. For the implementation, I found it very difficult to find the information I needed, especially on Open3D, which does not explain its methods. I ran out of time at the end of the semester to get a working solution. I don't think I've focused enough on the medical field. In the end, it only appears in one part of the bibliography, and in my implementation I didn't use medical images to test my solution.

Perspective For the future, I might try other frameworks a little more guided and automated than Open3D, such as COLMAP or AliceVision, which offer automatic pipelines. More generally, this work is part of a more comprehensive reconstruction process involving the registration of the final point cloud on a real patient surface.

5 Bibliography

- [1] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.
- [2] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [3] Jacob Freedman, Henrik Nilsson, and Eduard Jonas and. New horizons in ablation therapy for hepatocellular carcinoma. *Hepatic Oncology*, 2(4):349–358, 2015. PMID: 30191017.
- [4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.
- [6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.
- [7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.
- [8] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Computer Vision—ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part IV 11*, pages 257–270. Springer, 2013.
- [9] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [10] Sivic and Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings ninth IEEE international conference on computer vision*, pages 1470–1477. IEEE, 2003.
- [11] Mike Stephens and C Harris. A combined corner and edge detector. In *Alvey vision conference*, volume 15, 1988.
- [12] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. *Proceedings of the National Academy of Sciences*, 90(21):9795–9802, 1993.

- [13] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.
- [14] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.
- [15] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo, 2018.
- [16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.