Multi-View 3D Reconstruction of Objects

Dhanush Dinesh, Akash Sonth

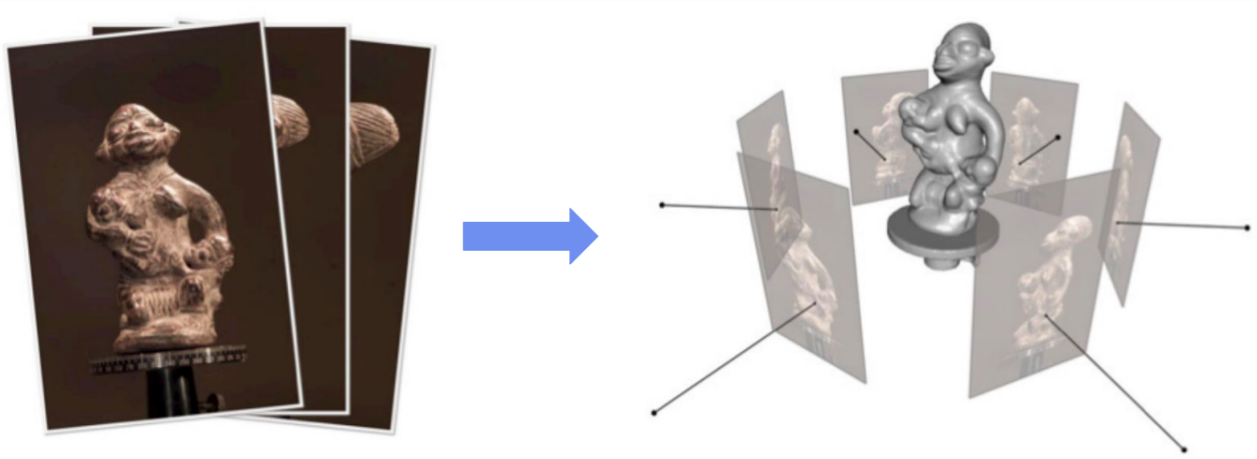

Figure 1: Figure 1: Example of a multi-view 3D surface reconstruction [1]

Abstract

The acquisition of 3D models is quite essential to research in computer graphics and computer vision. Constructing 3D models manually even with the help of modeling software is laborious and time consuming. In this project, we implement a lightweight algorithm which can produce a 3D model by analysing multiple images of an object. The output 3D points are compared to the ground-truth 3D model for a qualitative analysis.

1 Introduction

3D reconstruction has become essential in the field of computer vision. Although we have developed sophisticated software to construct 3D perspective, we require specific hardware such as radar, LiDAR, or structured light scenario which are not usually present in the devises we use.

3D modeling of objects is a technique used extensively in applications such as modeling, realistic rendering, gaming, robot navigation, 3D scene creation, target recognition, architectural design, illustration, commercial advertising and much more. Research in this domain has a lot of practical and theoretical significance.

With the advent of deep learning, people have worked on 3D reconstruction using Recurrent Neural Networks (RNNs) [2] as well as Convolutional Neural Networks (CNNs) [3] which require high computation. To overcome computation problems, we propose the following algorithm which uses feature matching, triangulation of points, making bundle adjustments, and finally merging point clouds by the iterative closest point algorithm.

1.1 Objectives

- The main aim of this project is to construct a 3D point cloud model of an object using multiple images from different viewpoints.

- Another objective of this project is to verify the feasibility and effectiveness of the proposed model through experimental analysis.

2 Approach

In this section we detail our algorithm and the required background knowledge for multi-view 3D reconstruction. For a given dataset (X) of n images arranged serially, pairwise images x_i and x_{i+1} are chosen to obtain the corresponding 3D points z_i. Thus, for $(n-1)$ pairs of images, we obtain $(n-1)$ sets of 3D points which are concatenated together to obtain the predicted 3D points (Z) corresponding to an object.

SIFT keypoints and descriptors are obtained for each image (x_i). To find feature matches between two images, the FLANN (Fast Library for Approximate Nearest Neighbors) matching algorithm is adopted. This gives a speed-up when compared to the BF (Brute Force) matching algorithm. A Euclidean distance measure of the matched features between images is computed. Matched features are then determined by thresholding the distance. RANSAC (RANdom Sample Consensus) is then applied to find the inliers of all the matched feature points.

Given camera intrinsics ($K_{3\times3}$), the extrinsic parameters $R_{3\times3}$ and $t_{3\times1}$ are to be obtained. In some cases, both the extrinsic and intrinsic parameters are given. The camera projection matrix ($P_{3\times4}$) is then obtained as $K \times [R|t]_{3\times4}$. If camera extrinsic parameters are not given, R and t for the first camera are assumed to be $I_{3\times3}$ and $[0,0,0]^T$ respectively (see Fig. 2). Based on this assumption, the parameters of other camera orientations are computed, to finally estimate the projection matrix P.

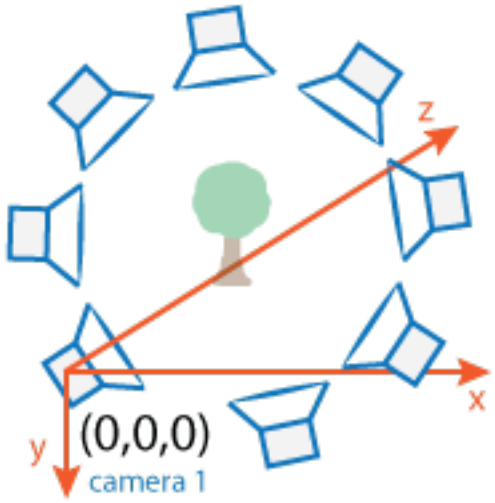

Figure 2: Figure 2: Example of a multi-view camera setup [1]

From the obtained camera projection matrices and matched points, the 3D points in world space can now be predicted by triangulation. To minimize the error in these predicted points, bundle adjustment is performed. This gives us our final point cloud of 3D points. In some cases when there is misalignment between point clouds (that can be resolved by rotation and translation), ICP (Iterative Closest Point) is performed to merge the point clouds together.

In general, the problem of multi-view 3D reconstruction can be split into three major steps [4, 5]:

- **Calibration Problem:** Any algorithm that aims to produce a 3D reconstruction of a scene must be able to determine how the points of the scene are projected on the image plane. As such, it is necessary to have prior knowledge of intrinsic camera properties (focal length, image center, etc.). Further, it is necessary to determine the rotation and translation matrix that projects the 3D scene point to the 2D image plane. Typically, this matrix is said to contain the camera's extrinsic properties.
- **Matching Problem:** The ability to recognize and associate points that appear in multiple stereometric views of an image is crucial to the performance of any 3D reconstruction method. The feature matching technique must be robust to outliers and accurately capture keypoints.

- **Reconstruction Problem:** Obtaining the 3D equivalents of the points from the associations made and calibration parameters forms the crux of this problem. Additionally, the 3D points must be dense enough to visually define the shape and texture of an object.

2.1 Background Knowledge

Before we delve into the algorithms we plan to use in this project, it is important to have a clear understanding of the geometric and algebraic principles that form the mathematical basis of this project.

2.1.1 Epipolar Geometry

In the case of multi-view images, the epipolar geometry of a stereo pair defines the relations between the multiple cameras, a 3D scene point, and that point's projections in each of the camera's image plane. Fig. 3 illustrates the

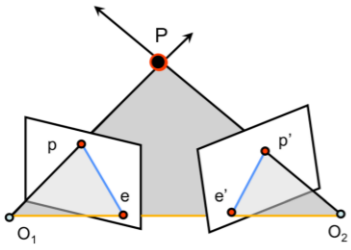

Figure 3: Figure 3: Epipolar geometry [6]

conventional epipolar geometry setup wherein two cameras observe the same 3D scene point P, whose projection in each of the image planes is located at p and p' respectively. The camera centers are located at O_1 and O_2, and line joining them is referred to as the baseline. The plane defined by O_1, O_2, and P is the **epipolar plane**. The locations of the point at which the baseline intersects the image planes are called the **epipoles**, e and e'. Finally, the lines defined by the intersection of the epipolar plane and the two image planes are known as **epipolar lines** [7]. As seen in Fig. 3, the epipolar lines pass through the epipoles.

Given the above description of epipolar geometry, we deduce the important fact that for each 3D point's projection onto one image the same point's projection must be observed in the other image on a known epipolar line. Thus, it is possible to test if two points correspond to the same 3D point. Epipolar constraints are algebraically described by the fundamental matrix between the two cameras. In Fig. 4 below, we can see that corresponding points in the first image lie on epipolar lines in the second image.

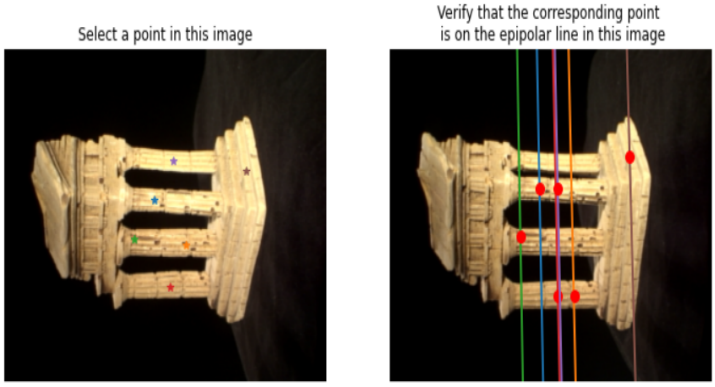

Figure 4: Figure 4: Matching Points Detected Using Epipolar Lines

2.1.2 Fundamental Matrix

The fundamental matrix may be thought of as the algebraic equivalent of the epipolar geometry of a stereo pair. Homogeneous image points in one image are transformed by the fundamental matrix to epipolar lines in the other image. In other words, the fundamental matrix allows for a mathematical expression of the epipolar constraint. Thus, without knowing the actual position of 3D scene point P, we can establish a relationship between any p and p'. [6]

The epipolar constraint for a point pair (p, p') expressed in terms of the fundamental matrix F is:

$$p^T F p' = 0$$

In the following subsections, we describe the methods we use in this project to tackle the aforementioned problems.

2.2 Calibration Problem

For this project, the image datasets (Middlebury [8] and EPFL [9]) already contain all the necessary information pertinent to intrinsic camera properties. Thus, we are only required to compute the extrinsic camera matrices. This is done by estimating the **essential matrix**, a 3×3 matrix that is useful for determining both the relative position and orientation between the cameras and the 3D position of corresponding image points. For a pair of stereo images, the essential matrix E is computed from the inner camera matrices (K and K') and the fundamental matrix F.

$$E = (K')^T F K$$

In the following sections, we describe the procedures involved in fundamental matrix estimation and 3D reconstruction.

2.3 Matching Problem

For the purposes of feature detection and matching, a combination of the SIFT (Scale Invariant Feature Transform) and RANSAC (Random Sample Consensus) algorithms is used to obtain a set of points that allow us to estimate the fundamental matrix, which in may in turn be used to compute the transformation matrices that allow us to acquire a 3D metric reconstruction from 2D correspondences.

We now elaborate on SIFT, RANSAC, and the eight-point algorithm for fundamental matrix estimation.

2.3.1 SIFT Algorithm

SIFT (Scale Invariant Feature Transform) is a feature detection algorithm. The main idea involved in the SIFT algorithm is that image content can be transformed into local feature coordinates that are invariant to translation, rotation, and scaling. Thus, SIFT allows for the identification and localization of local image features, commonly known as 'keypoints' of the image. Crucially, these keypoints are scale and rotation invariant and are thus unaffected by size or orientation of an image.

Broadly speaking, the SIFT algorithm comprises four parts:

- **Scale Space Construction:** Ensure that features are scale invariant.
- **Keypoint Localization:** Identify and locate keypoints.
- **Orientation Assignment:** Ensure that keypoints are rotation invariant.
- **Keypoint Descriptor Assignment:** Individual keypoints are assigned a unique descriptor or fingerprint.

A complete description of the SIFT algorithm is provided in [10]. For the purposes of this project, SIFT is used to extract keypoints from each image in a set of stereo images. The keypoints are then matched across the set of stereo images. Keypoints between images are matched by identifying their nearest neighbors. However, the nearest neighbor approach for feature matching lacks robustness because it is not able to filter outliers. This is where the RANSAC algorithm comes into play.

2.3.2 RANSAC

RANSAC (Random Sample Consensus) is an iterative algorithm that estimates a mathematical model for a set of data that contains outliers. As such, at its core, RANSAC is an outlier detection method. In this project, RANSAC will be used to eliminate faulty correspondences between images to obtain an accurate set of matched points that can be used to estimate the *fundamental matrix*. While the details of the fundamental matrix estimation procedure are provided in later sections, for now it suffices to say that we require 8 point pairs to estimate the fundamental matrix. Given this information, we may outline the RANSAC algorithm as follows [11]:

1. **Sample** (randomly) the number of point pairs required to fit the model (8, in our case)
2. **Solve** for model parameters using sample.
3. **Score** by the fraction of inliers within a preset threshold of the model.
4. **Repeat** 1-3 until the best model is found.

In general, as the number of iterations increases, the more likely it is that an accurate estimation of the fundamental matrix will be found. The following equation provides an estimate of the number of iterations k needed to obtain a solution where some iteration of RANSAC selects only inliers with a probability p, given a prior assumption of the proportion of corresponding points being inliers (w)

$$k = \frac{log(1-p)}{log(1-w^n)}$$

Finally, we plan to use RANSAC in tandem with the fundamental matrix estimation algorithm– for each iteration, a random samples of eight point pairs are used to estimate the fundamental matrix F, and this fundamental matrix is then used to determine the fraction of inliers across the entire set of point pairs within a preset threshold. A point pair (x', x) satisfying the following equation is considered an inlier:

$$|x'^T F x| < T$$

where T is a preset threshold. In simpler terms, the epipolar constraint (with a threshold) is leveraged in RANSAC to ensure accurate estimation of the fundamental matrix.

Therefore, to summarize, RANSAC allows us to determine the optimal fundamental matrix by ensuring that the point pairs used in its calculation are good matches.

2.3.3 Normalized Eight-Point Algorithm for Fundamental Matrix Estimation

While we have already alluded to the eight-point algorithm in previous sections, in this subsection we provide a detailed overview of the algorithm and its application in estimating the fundamental matrix.

The normalized eight point algorithm is used to estimate the fundamental matrix for a stereo pair given a set of point correspondences (or point pairs). Each point correspondence, say (x_i, x_i') where $x_i = (u_1, v_i)$ and $x_i' = (u_1', v_1')$ produces one constraint on the fundamental matrix, as it must satisfy the epipolar constraint [6].

$$x_i'^T F x_i = 0$$

On expanding this equation, we obtain:

$$\begin{pmatrix} u_i u_i' & v_i' u_i & u_i & u_i' v_i & v_i v_i' & v_i & u_i' & v_i' 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f13 \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

Thus, for n point correspondences, we obtain the following matrix equation:

$$\begin{pmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f13 \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \mathbf{0}$$

$$\Leftrightarrow \mathbf{Af} = 0$$

where $\mathbf{A}$ is the $n \times 9$ equation matrix, and $\mathbf{f}$ is a 9-element column vector containing the entries of $F_{3\times3}$.

The fundamental matrix has 7 degrees of freedom so we require at least 8 point correspondences to compute it. [6]

The least-squares solution to $\mathbf{f}$ is obtained via singular value decomposition (SVD) on the matrix $\mathbf{A} = UDV^T$. In fact, the vector $\mathbf{f}$ that minimizes $||Af||$ such that $\mathbf{f}$ is a unit vector can be found along the column of V corresponding to the least singular value.

The true fundamental matrix has rank 2, therefore we seek a solution that is the best rank 2 estimate of F. This requires that we solve the optimization problem defined below [6]:

$$\min_F ||F - \hat{F}||_F$$
$$\text{s.t. } \det F = 0$$

This is solved by SVD, where $\hat{F} = UDV^T$. Then, the best rank-2 approximation is given by:

$$U \begin{pmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} V^T = 0$$

The above process describes the unnormalized or "vanilla" eight point algorithm. In practice, this approach leads to inaccuracies because the distance between a point x and its corresponding epipolar line $l = Fx$ may be very large. To reduce this error, we use the **normalized eight point algorithm**. [6]

2.4 Reconstruction Problem

Having described our methods to tackle the matching problem, we now consider the reconstruction problem.

Our approach to solving the reconstruction problem can be broken down as follows:

2.4.1 Camera Projection Matrix Estimation

To extract the camera matrix we need to first compute the **essential matrix** E from the fundamental matrix F and intrinsic camera matrices, K and K'.

$$E = K'^T F K$$

Then, given the SVD of the essential matrix $E = UDV^T$, and first camera matrix $P = [\mathbf{I}|0]$, there are four possible choices for the second camera matrix P' given by [12]:

$$P' = [UWV^T|\mathbf{u_3}]$$

$$P' = [UWV^T|-\mathbf{u_3}]$$

$$P' = [UW^TV^T|\mathbf{u_3}]$$

$$P' = [UW^T V^T | -\mathbf{u_3}]$$

where,

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

It can be shown that a reconstructed point P will be in front of both cameras for only one choice of P'. Thus, testing on a single point is sufficient to determine the correct P'. [12]

2.4.2 Triangulation

Having estimated the camera matrices (P, P'), and knowing the points (x, x') in the image planes, a projective-invariant triangulation method is used to obtain the point (X) in 3D space.

The equations $x = PX$ and $x' = PX$ can be further combined into $AX = 0$, which is linear in X. The solution to this is the (unit) eigenvector of $A^T A$ with least eigenvalue, thus giving us the 3D world point. This is based on the assumption that there is no little or no error (noise) in the image points (x, x').

2.4.3 Bundle Adjustment

In the case that the image points (x, x') are noisy, the equation $x = PX$ will not be satisfied. Assuming that the measurement noise is Gaussian, we solve using Maximum Likelihood estimate. That is, we attempt to estimate camera projection matrices $\hat{P}^i$ and 3D points $\hat{X}_j$ that minimize reprojection error i.e. allow for recovery of exact image points $\hat{x}^i_j$ as $\hat{x}^i_j = \hat{P}^i \hat{X}_j$, and also minimize the geometric image distance between the reprojected point and observed (ground truth) image points x^i_j. Thus, we solve the following optimization problem [12]:

$$\min_{\hat{P}^i, \hat{X}_j} \sum_{ij} d(\hat{P}^i \hat{X}_j, x^i_j)^2$$

where $d(x, y)$ is the geometric image distance between homogeneous points x and y. This joint estimation of the camera matrices and 3D projections is known as *bundle adjustment* [12].

2.4.4 Iterative closest point (ICP)

Iterative closest point (ICP) is a optimization algorithm used to minimize the difference between two point clouds of point. In the ICP algorithm, one point cloud which is the reference is kept fixed, while the other one is transformed to match the reference.

The steps in ICP algorithm are as follows

- For each point in the probe point cloud (source), find the closest point in the reference point cloud (destination).
- Compute the transformation between probe point cloud and reference point cloud (SVD).
- Update the probe point cloud
- The error function can be used for aligning the input point cloud to that of reference.

Next calculate the final transformation, transform the points in the probe image using the obtained transformation. Once the pose correction is done using ICP, both reference and probe point clouds are converted to a single point cloud that best captures the object's true structure. Thus, the purpose of ICP here is to resolve the point clouds obtained from multiple stereo pairs.

3 Experiments and Results

3.1 Data Set

Experiments are conducted on the widely-used multi-view reconstruction datasets- Middlebury multi view [13], EPFL dense multi-view stereo [9].

Middlebury multi view [13] consists of images of two objects- "Temple" and "Dino", captured by a Stanford Spherical Gantry [14]. Fig. 5 shows the camera setup used to capture the dataset. Each image is of resolution 640×480. Each object has a dense view (images sampled on a hemisphere), ring (images sampled on a ring around the object), and a sparse ring counterpart. Along with these images, the dataset also includes intrinsic and extrinsic camera matrices. Reconstruction is performed on all three counterparts for this project. A sample of the data-set is shown in Fig. 6.

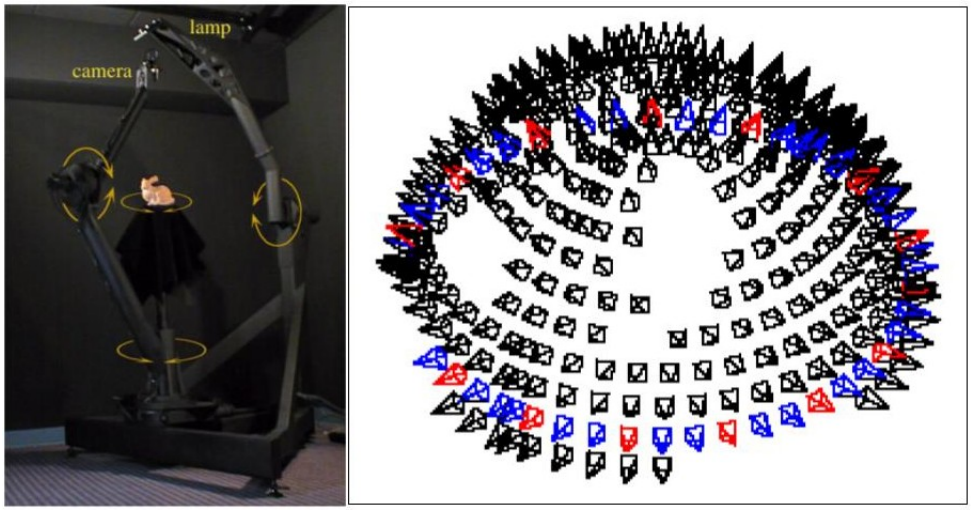

Figure 5: Figure 5 Left- The Stanford Spherical Gantry arrangement. Right- Orientations at which the camera was positioned to capture the dataset [14]

Figure 6: Figure 6: A sample of the Middlebury "temple" data-set [8]

EPFL dense multi view stereo [9] is a collection of high-resolution (3072×2028) images of various structures (fountain, Herz-Jesu, castle). Along with these images, the dataset also includes projection matrices. A sample of the data-set is shown in Fig. 7.

Figure 7: Figure 7: A sample of the EPFL "fountain-P11" data-set [9]

3.2 Results

In this subsection we show our estimated 3D models for the different datasets, and provide a qualitative analysis of the same. The dataset authors do not open-source their metric computation code and corresponding requirements. The authors accept dense point-cloud files and compute metrics themselves to report on their website. Moreover, those datasets that do provide the ground-truth do so for triangulated mesh models with the ground-truth consisting of millions of points. This is not feasible with our existing compute power and is highly impractical for our case.

3.2.1 Middlebury "temple" dataset

In the below 3D model, we can see that the side view of the temple and the pillars are clearly separated. This model was created from 16 input images.

Figure 8: Figure 8: Left- One of the input dataset images. Right- Predicted 3D model from 16 views

The following figure shows the difference in predicted 3D model when the number of input images is increased. While we obtain a much more dense model with more images, it also brings in more noise. In addition, the time taken to process also increases substantially.

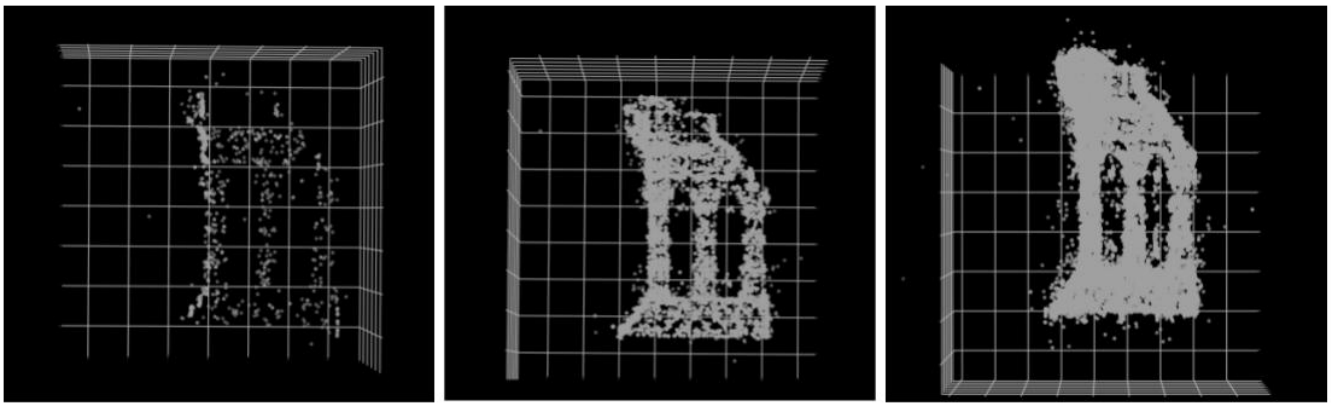

Figure 9: Figure 9: Left- Reconstruction from 16 views sampled on a ring. Center- Reconstruction from 47 views sampled on a ring. Right- Reconstruction from 312 views sampled on a hemisphere

The next plot show the top view of our estimated 3D model. It correctly depicts the object, i.e., temple when seen from the top. It should be noted here that input image corresponding to top view specifically was not provided in the dataset. The algorithm estimated points for the roof of the temple from partial views of the roof from different views.

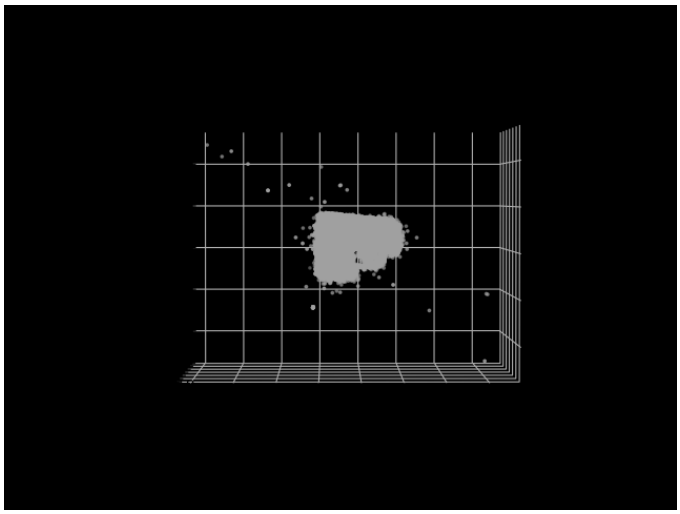

Figure 10: Figure 10: Top view (roof) of the estimated 3D model from 312 views

In the following figure, we can clearly see the depth created by the pillars of the temple sample. We can also make out the steps of the foot of the temple as well as the roof of the temple to some extent.

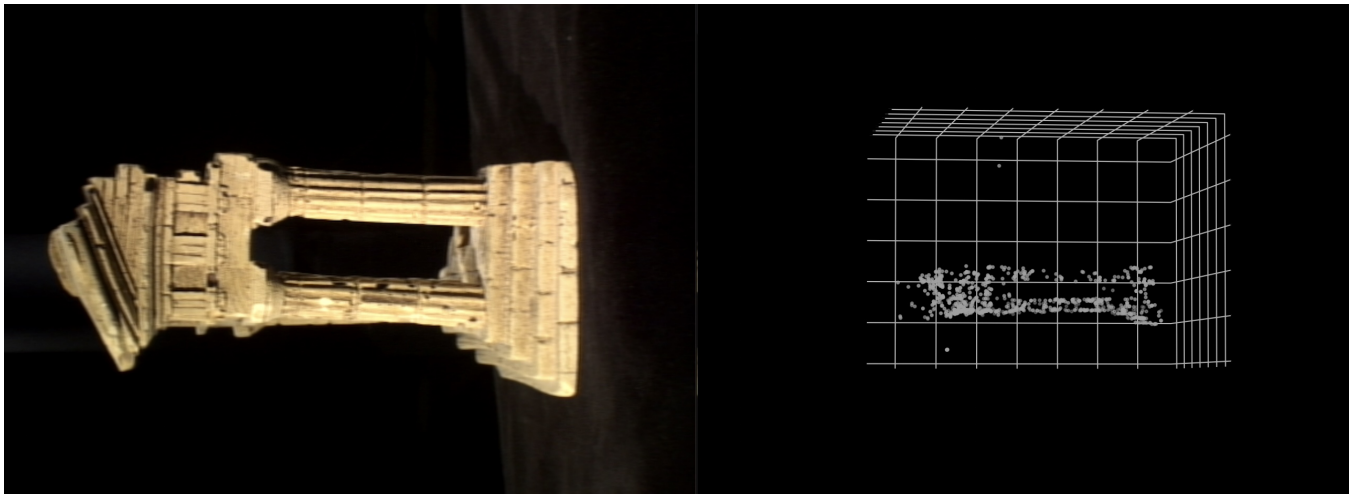

Figure 11: Figure 11: Left- One of the input dataset images. Right- Predicted 3D model from 16 views

3.2.2 EPFL "castle-P19" dataset

The EPFL dataset consists of images taken with large difference in angles between consecutive images. Therefore, the number of matched points between images are lower than usual. This results in an incomplete and sometimes incomprehensible 3D reconstruction. This can be observed in the following reconstructed models for "Castle-P19" and "Fountain-P11".

In the below figure we see an image of one side of the castle and the 3D reconstructed model. We can observe that there is a tree in the vicinity due to which a lot of feature points were identified at that location.To make the corresponding points between the image and model clearer to perceive, we have labelled some of the areas. For a better perspective, we have taken the screenshots of the model by tilting it up.

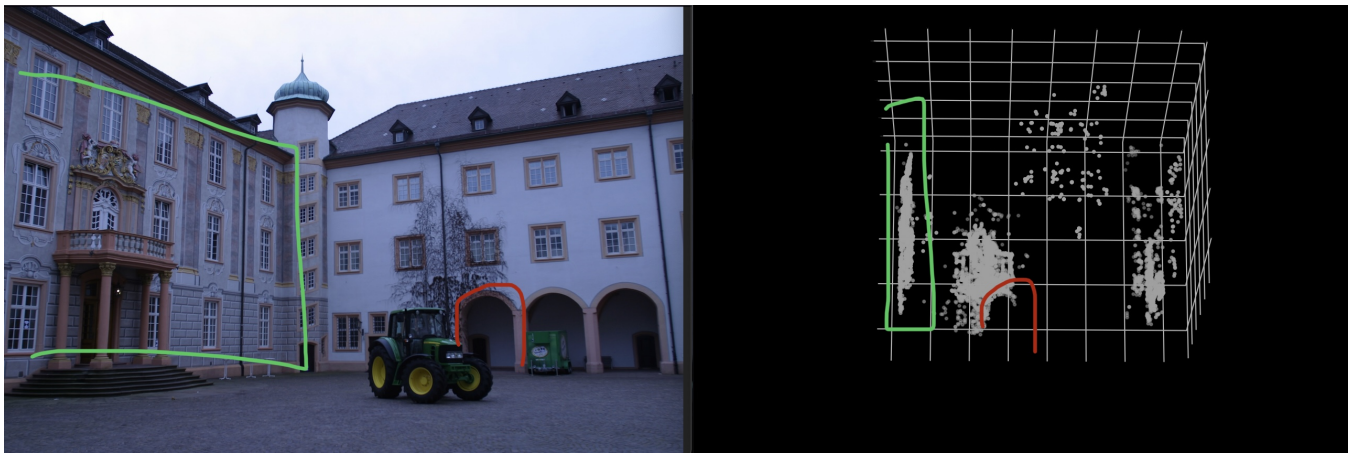

Figure 12: Figure 12: 3D model along with a image taken from the same angle.

In the following figure we see other side of the castle. As shown in the predicted output, there are a lot of feature points identified at the periphery of the entrance where there is a lot of intricate work. We can also observe that there are a lot of points in the corner of the building where there are a lot of edges formed due numerous windows.

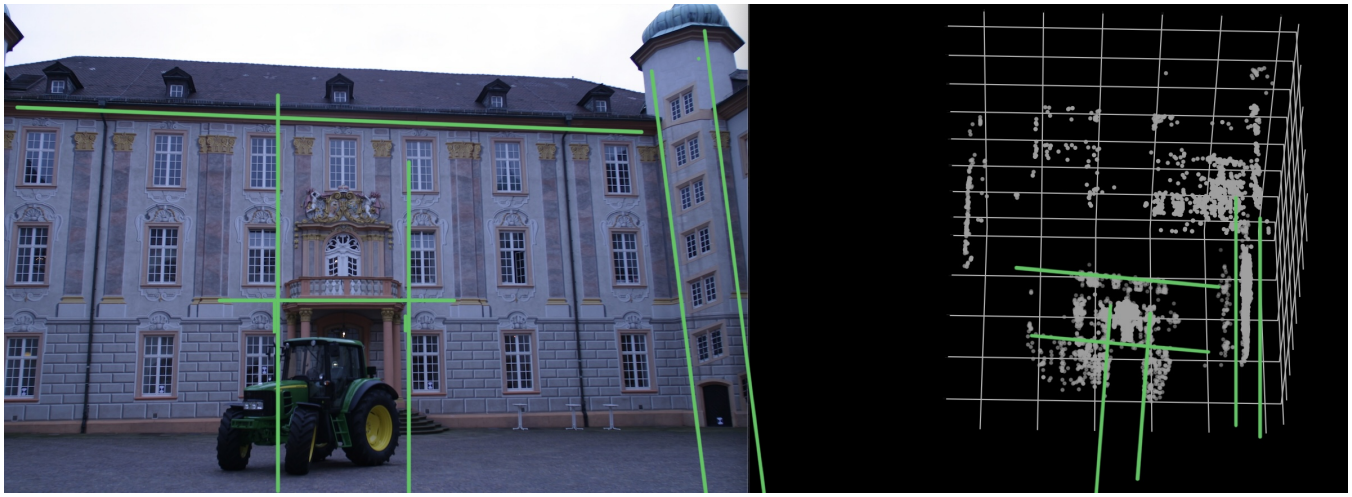

Figure 13: Figure 13: Side of the castle

In the next figure we see that the algorithm found a lot of feature points in the left corner of the castle due to the several branches of the tree.

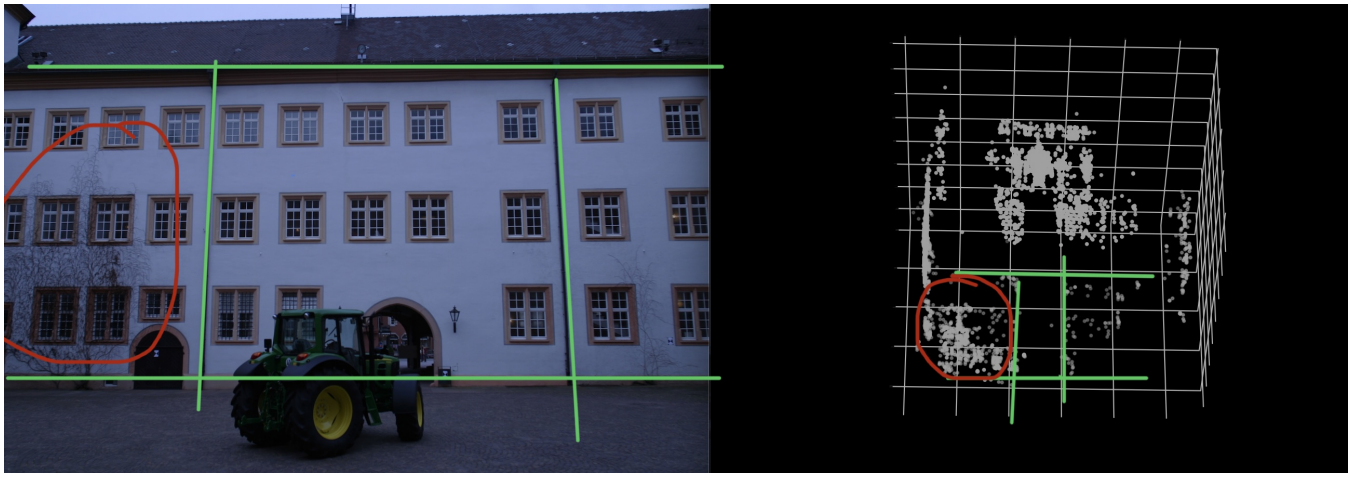

Figure 14: Figure 14: Side of the castle

In the next figure image we can identify the tree as well as the arch beneath the tree and all the feature points of the corner windows.

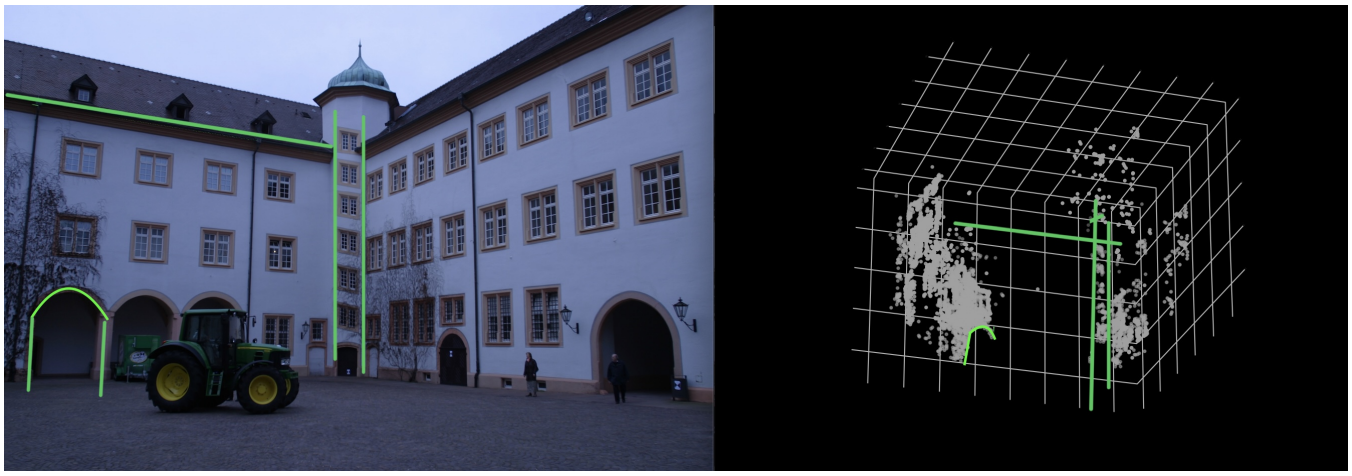

Figure 15: Figure 15: Side of the castle.

The following top view of the 3D model gives us a better perspective of all the 4 sides of the castle. The right edge of the output shows very few points. This is because it mainly consists of a plain wall with no windows or design, and no features to detect.

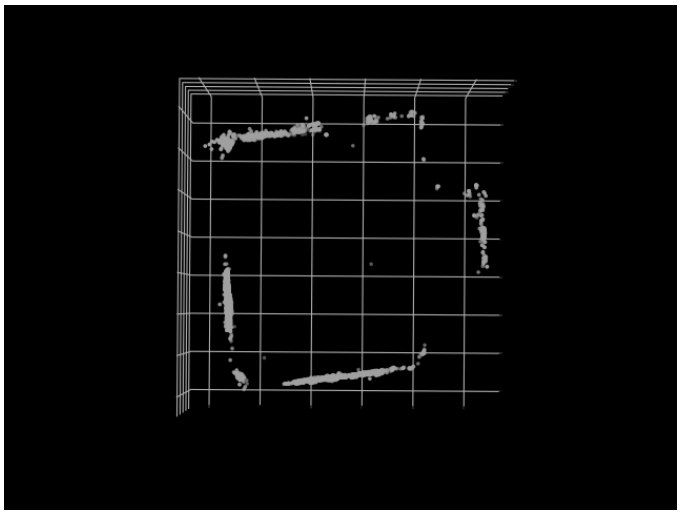

Figure 16: Figure 16: Top view of the castle

3.2.3 EPFL “fountain-P11” dataset

Below we can see a side view of the fountain data-set. We can observe that there are 2 planes in the output which is the result of the two planes present in the image. Here we can observe the fountain wall is on the right side of the output and the castle building with the pillared entrance can be see on the left side of the image.

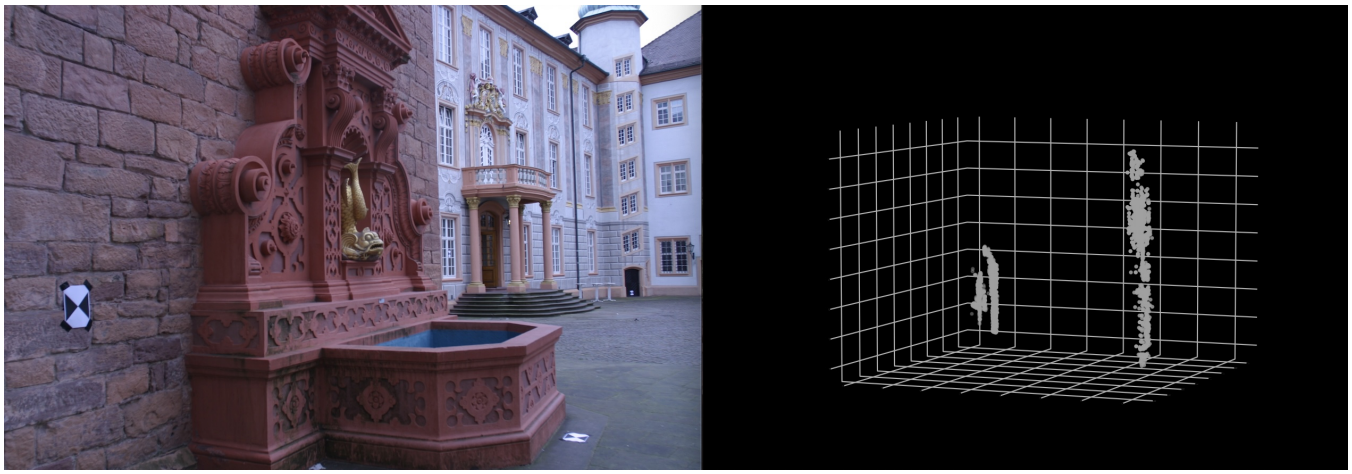

Figure 17: Figure 17: 3D model along with an image taken from the same angle.

In this figure we can observe that the wall of the fountain having the most texture has the most number of feature points.The entrance also has the lot of points due to the edges and windows. The algorithm could only reconstruct the two planes and not the fountain.

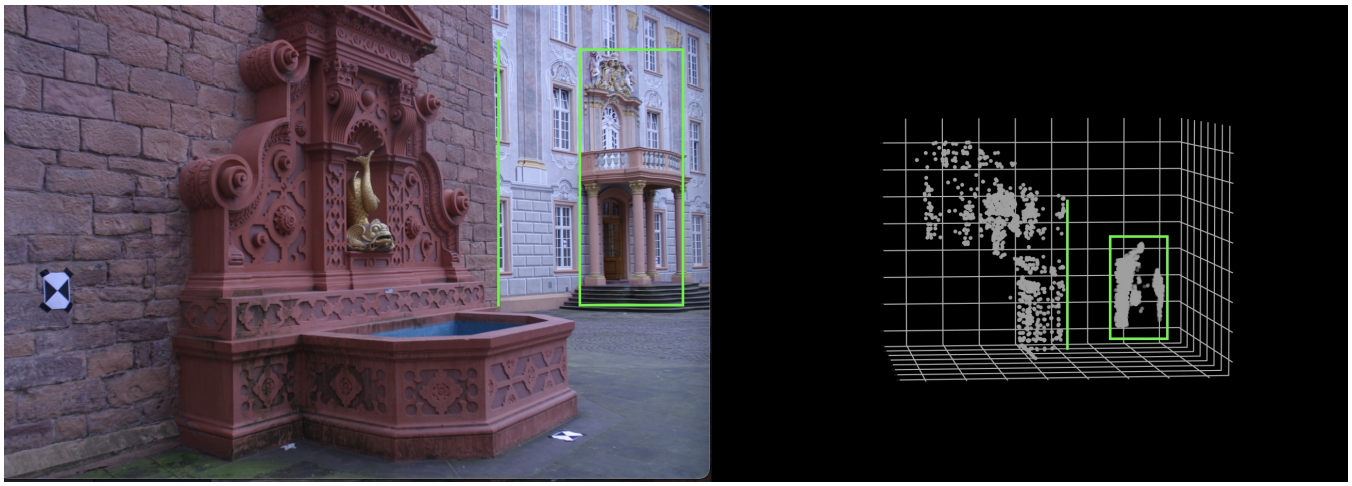

Figure 18: Figure 18: 3D model along with a image taken from the same angle.

3.2.4 Middlebury "dino" dataset

The "dino" object consists of multiple spikes where most of the features are captured, and a uniform body surface where little to no features are detected.

In this figure we can observe that the spikes of the Dino was captured. We can clearly see the feature points created by the back of the Dino.

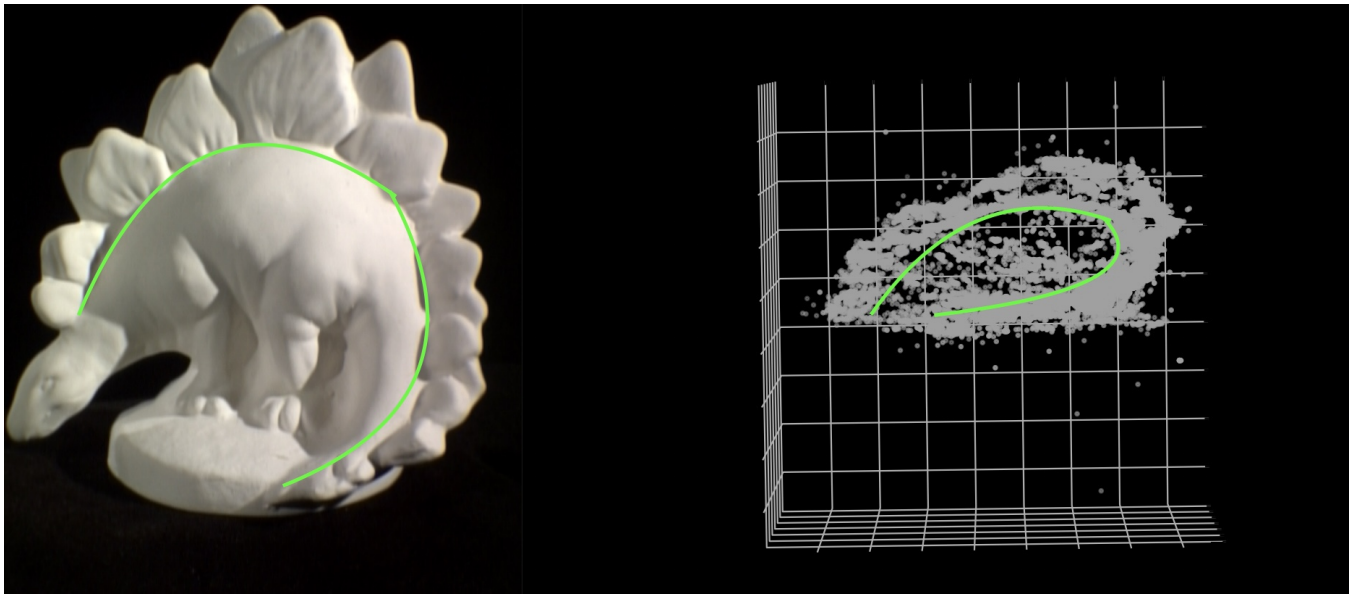

Figure 19: Figure 19: 3D model along with a image taken from the same angle of the Dino object.

In the following figure we can see that the top view of the model clearly shows the orientation and the depth of the Dino. There are very few points outside the structure of the Dino, i.e. noise, which shows how robust the algorithm is.

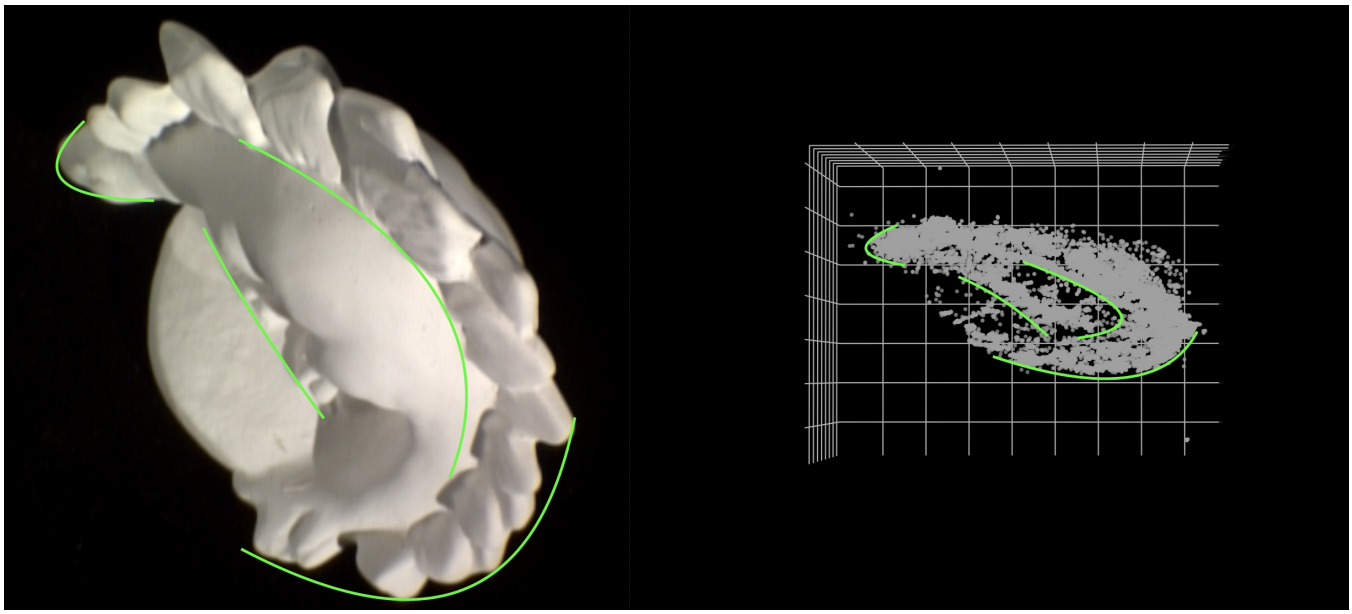

Figure 20: Figure 20: Top view of the Dino

3.3 SIFT feature matching: Brute Force vs. FLANN

In this section we compare the BF (Brute Force) Matching algorithm with FLANN. KNN (k-Nearest Neighbors) followed by Lowe's ratio test is usually used after using either method for matching the SIFT descriptors. Several parameters in the feature matching methods are standard and recommended as per the documentation.

The BF matching algorithm gives 80 matching feature points corresponding to the given input images as shown in the below figure.

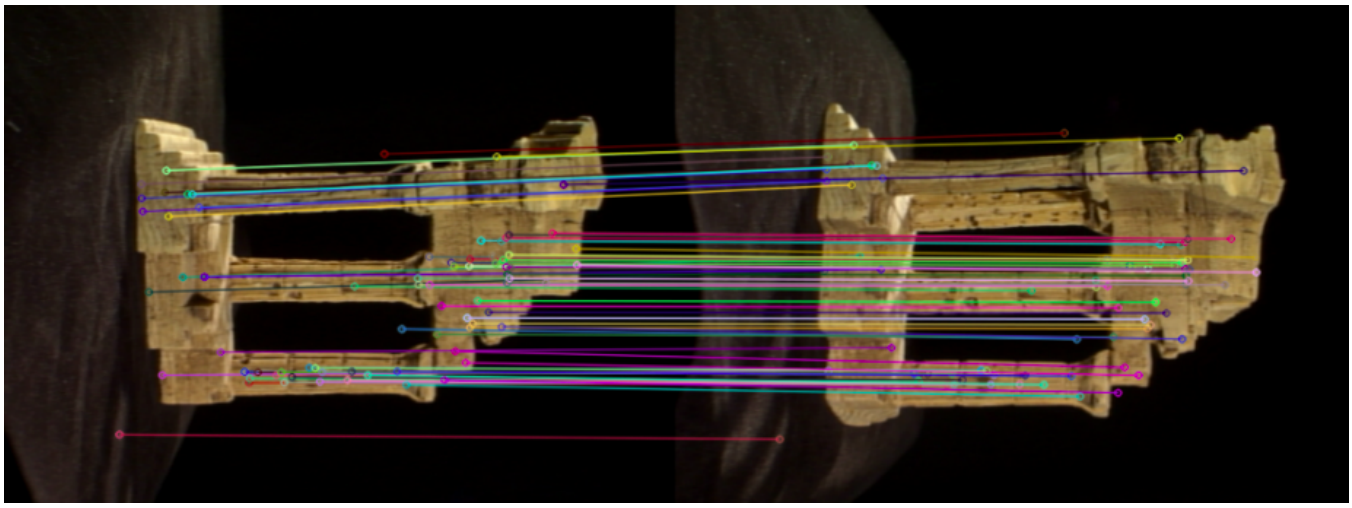

Figure 21: Figure 21: Output of the BF Matching algorithm

The FLANN matching algorithm gives 82 matching feature points corresponding to the given input images as shown in the below figure. These results are obtained when the search parameter (the number of times the trees in the index should be recursively traversed) is set to 50. Using larger values for this parameter gives marginally better results but at the cost of higher execution time.

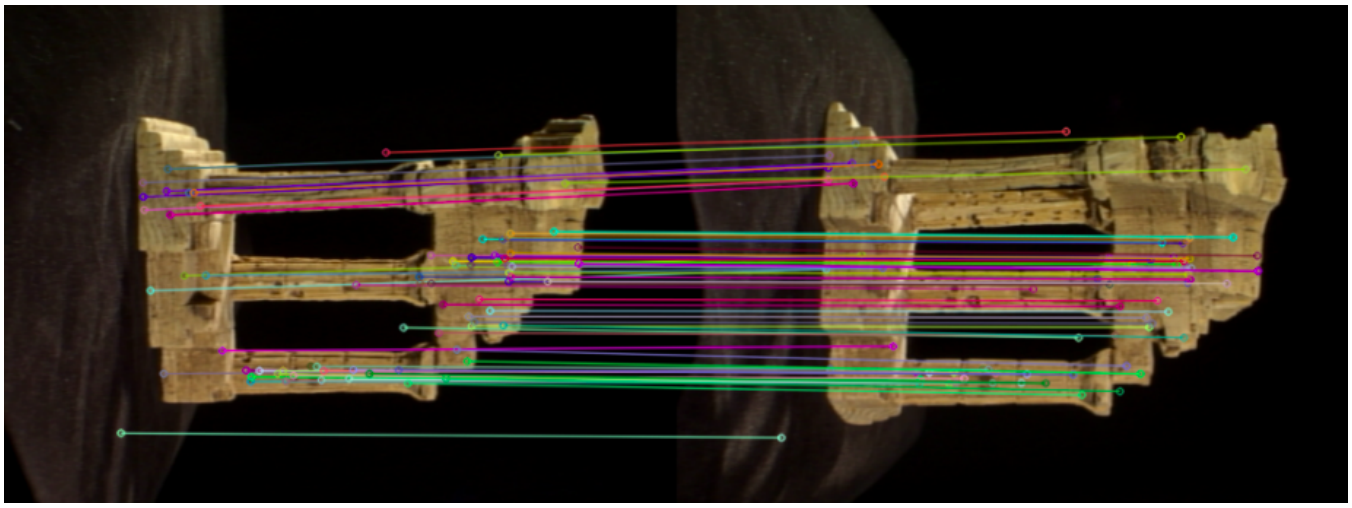

Figure 22: Figure 22: Output of the FLANN Matching algorithm

The results by either BF or FLANN matcher are very close. But BF takes much more time to run as compared to FLANN, thus impractical in large datasets.

4 Conclusion and Future Work

In this project, we implemented a 3D reconstruction algorithm based on multiple input images. We have created a robust algorithm which can develop a 3D model by taking in multiple images as the input. Since this method depends on classical computer vision algorithms, it is very lightweight and inexpensive in terms of computation.

As future work, 3D models can be constructed in Blender, and images can be captured by positioning the camera at different positions to generate a synthetic dataset. Surfaces can also rendered from the plotted 3D points to obtain texture of the object.

References

[1] Y. Furukawa and C. Hernández, *Multi-View Stereo: A Tutorial*, 2015.

[2] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 628–644.

[3] L. Fangmin, C. Ke, and L. Xinhua, "3d face reconstruction based on convolutional neural network," in *2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2017, pp. 71–74.

[4] G. Vogiatzis and C. Hernández, *Practical 3D Reconstruction Based on Photometric Stereo*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 313–345. [Online]. Available: https://doi.org/10.1007/978-3-642-12848-6_12

[5] Calle Olsson, "The structure from motion pipeline," https://www.youtube.com/watch?v=i7ierVkXYa8, 2012.

[6] S. S. Kenji Hata, "Cs231a course notes 3: Epipolar geometry." [Online]. Available: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf

[7] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[8] Daniel Scharstein, "Middlebury data-set," https://vision.middlebury.edu/mview/, 2012.

[9] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *2008 IEEE conference on computer vision and pattern recognition*. Ieee, 2008, pp. 1–8.

[10] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 11 2004.

[11] A. Bobick, "Cs4495 computer vision: Random sample consensus." [Online]. Available: https://www.cc.gatech.edu/~afb/classes/CS4495-Fall2014/slides/CS4495-Ransac.pdf

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[13] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 1. IEEE, 2006, pp. 519–528.

[14] Marc Levoy, "Stanford spherical gantry," http://graphics.stanford.edu/projects/gantry/, 2004.