3D Reconstruction using Stereo Vision and Structure From Motion

Ainharan Subramaniam, Ryan Cao, Quoc Anh Hoang

***Abstract*— Artificial intelligence systems that observe a scene must be able to obtain three dimensional (3D) information. With an ever growing dependence on computer vision based artificial intelligence such as autonomous cars it is important such systems obtain accurate 3D data. To address this problem in this paper we discuss current models of reconstructing a scene given two dimensional (2D) data, such as 3D reconstruction from stereo vision as well as structure from motion. The paper outlines our experimentation with both methods and explains the advantages and disadvantages of both techniques.**

I. INTRODUCTION

Image analysis from two dimensions can restrict our world vision, 3D images provide more information. The challenge of 3D scene reconstruction from 2D images through methods in computer vision such as stereo vision and structure from motion (SfM), are used to bring insight to be able to better analyze the world. These ideas involve using multiple images in different perspectives and properties of a common camera to estimate depth perception. By matching key points in this set of images and determining its depth with one of these techniques, a set of 3D points can be extrapolated to generate a point cloud. This point cloud is essentially a 3D representation of the subject in the 2D images, and hence provide a better understanding of ones world view.

II. PROJECT GOALS

At its core, our project goal is to generate point cloud models of 3D object or scenes, based on photos taken from around the subject material in order to advance our knowledge on the 3D reconstruction pipeline. Our initial goal for a minimum viable product is to achieve reconstruction of a sparse model point cloud, with input data including both the images and projection camera matrices. This process covers the core steps of the 3D reconstruction pipeline, and allows us to gain experience in the libraries, functions and data structures required for more complex refinements. Part of this goal involves creating our own implementation of 3D point triangulation based on input camera matrices rather than using existing library functions.

After we complete the core functionality, we intend on advancing the completeness of the 3D model by implementing reconstruction using structure from motion methods, which – unlike stereo reconstruction – creates point cloud models based on more than two images for input. The primary challenge in this stage will likely be the accurate calculation and matching of key points across different triangulations, which will require additional calculations based on input projection matrices.

Once this is complete, we intend on branching out our efforts into several concurrent goals with which we could improve the quality, functionality and/or applicability of our program. These goals represent extension objectives which we may complete:

- Mesh reconstruction based on point cloud data, which aids in visualisation and will be useful in further applications that require importation of a 3D model.
- Image pre-processing to achieve higher accuracy of key point detection, which in turn will make the sparse point cloud points more precise and reduce outliers due to noise.
- Dense model reconstruction which can be achieved by integrating colour information, increasing density of keypoints in point clouds, extracting adjacent pixels of points and plotting as a point-square mesh.

With all of this we hope to learn more about how we can then take the knowledge of 3D reconstruction, and extend it and apply it in order to create clearly models; as well as create a springboard for future variations and implementations of additional functions for scene reconstruction.

III. LITERATURE SURVEY

There have been many previous works in 3D reconstruction based on images, utilised in various projects. Of these, two of the largest are the Photo Tourism project by the University of Washington, and the Building Rome in a Day project by Microsoft Research which drew heavily upon the former. Both of these projects followed the 3D reconstruction pipeline to create first sparse, then dense model point cloud reconstructions of tourist attractions; utilising multi-view SfM techniques and large picture datasets from many angles around objects in order to build accurate and realistic recreations of their targets. The Building Rome in a Day project took this process further, drawing upon public photo repositories and mapping out an entire city across photos which are from vastly different angles, camera types and views, and qualities. Both of these projects served to show us what is capable with 3D reconstruction in creating scenes of other places that can be appreciated by anyone. [1][2] In our project we primarily referred to several papers and reports which helped guide us in recreating each element of the 3D construction pipeline; helping us work out which algorithms work best at each stage. This included lecture slides by the German Centre for Artificial Intelligence (DFKI) [3], which provided more detailed information on the way algorithms such as Structure from Motion operate and what challenges we would have to overcome in implementing it; as well as a report by (Macann et. al 2015) providing an approximate example of the minimum viable product we would have to make and potential data sources. [4]

IV. Preliminary Definitions

- *Structure in Motion* - Structure from motion is a method of determining the positions of points in a scene as well as the motion [R, t] of the calibrated camera

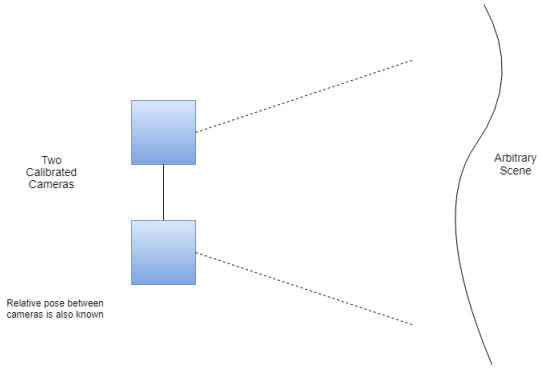


Fig. 1. Simplistic case: co-planar stereo cameras

- *Point Cloud* - A series of points in 3D space, which describe the geometry of a object/scene
- *Camera Matrix*: The camera matrix or projection matrix is a 3 x 4 matrix which describes the mapping of a pin-hole camera. We can decompose this projection matrix into the multiplication of of the following matrices:
  - *Intrinsic* - transforms 3D camera coordinates to 2D homogeneous image coordinates
  - *Extrinsic* - describes the cameras location in the world as well as the rotation/direction the camera is facing

V. Problem Decomposition

Problem Decomposition The method of sparse model point cloud reconstruction follows a basic pipeline presented below. Our goal is to provide our own implementations of the camera matrix calculations and the triangulation in both stereo and structure from motion (SfM) implementations rather than relying on existing libraries. To first part of the 3D reconstruction problem is the correspondence problem which refers to the task of finding a set of points which can be identified in the opposing image, i.e. for every point in the first image there are many possible matches in the 2nd image. However matches are often ambiguous and locally points look similar. Using known geometry of the camera to help limit the search for matches we can find correspondences between two images via correlation-based or feature based methods. Correlation based methods simply refers to checking if an image shares the same location. Feature based requires to find matching/similar features in the images then checking if the layout a subset is similar. The matching can be limited to searching for a match to be along a certain line in the other image. If the cameras are aligned, the search for corresponding point is 1D along the corresponding row of the other camera. If cameras are not aligned a 1D search can still be determined for the corresponding point. P_1, C_1, C_2 determine a plane that cuts image I_2 in a line: P_2 will be on that line Sometimes it is useful to rectify the images when

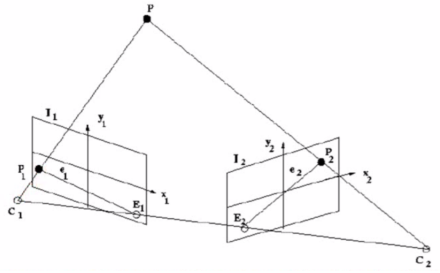


Fig. 2. Unaligned stereo cameras

the camera planes are not aligned. This means both cameras are rotated so that they are looking perpendicular to the line joining the camera centers. Effectively the epipolar lines will be horizontal simplifying the problem to a 1D search problem. For this project after normalizing i.e. rectifying our images we used the SIFT algorithm as a feature based correspondence method. This included the extraction of image keypoints as descriptors with invariance to scale and rotation and uniqueness/differentiability from each other. As well as matching keypoints and descriptors between pairs of images using FLANN based matching, and elimination of outliers using RANSAC. After calculating the inlier correspondence we then triangulate these points using the 8 point algorithm as discussed later, to compute the point cloud.

VI. Design and Algorithms

A. Stereo vision - Hartleys normalized eight-point algorithm

The fundamental matrix is a basic tool in the analysis of scenes taken with two uncalibrated cameras, and the 8-point algorithm estimates the essential matrix or the fundamental matrix from a set of eight (or more) corresponding image points. It has the advantage of simplicity of implementation. Richard Hartley [7] proposed that the coordinate system of each of the two images should be transformed, independently, into a new coordinate system according to the following principle.

- The origin of the new coordinate system should be centered (have its origin) at the centroid (center of gravity) of the image points. This is accomplished by a translation of the original origin to the new one.
- After the translation the coordinates are uniformly scaled so that the mean distance from the origin to a point equals $\sqrt{2}$

This principle results, normally, in a distinct coordinate transformation for each of the two images. As a result, new homogeneous image coordinates are given by

$$\bar{y} = Ty$$

$$\bar{y}' = T'y'$$

where T, T' are the transformations (translation and scaling) from the old to the new normalized image coordinates. This normalization is only dependent on the image points which are used in a single image and is, in general, distinct from

normalized image coordinates produced by a normalized camera. The epipolar constraint based on the fundamental matrix can now be rewritten as

$$0 = (\bar{y}')^T((T')^T)^{-1}FT^{-1}\bar{y} = (\bar{y}')^T\bar{F}\bar{y}$$

Where $\bar{F} = ((T')^T)^{-1}FT^{-1}$

This means that it is possible to use the normalized homogeneous image coordinates y, y' to estimate the transformed fundamental matrix F using the basic eight-point algorithm described above. The purpose of the normalization transformations is that the matrix Y, constructed from the normalized image coordinates, in general has a better condition number than Yhas. This means that the solution f is more well-defined as a solution of the homogeneous equation Y f than f is relative to Y.Once f has been determined and reshaped into Fthe latter can be de-normalized to give F according to:

$$F = (T')^T\bar{F}T$$

1) Mesh Generation: Mesh generation relied on the use of Delaunays Algorithm, which iterates over every pair of points in the the input and draws a circumcircle through them, then checking whether any other points lie within the drawn circle. If a point does lie within the circumcircle, the pair is discarded as non-adjacent. Eventually a final list of pairs of points is found, each of which then forms an edge in the Delaunay triangulation (not to be confused with 3D position triangulation).[5]

The basic version of the algorithm does not take into account outlier points or larger triangles, and aims to form a closed convex hull across all input points. In our usage of the algorithm we implemented an additional condition which would remove Delaunay triangles that exceeded a certain size.

Delaunays algorithm can be extended by marking a set of edges as boundaries, which will mean that edges beyond the boundary will not generate additional vertices.[6] This is one of the few ways of dealing with concave curves in surface reconstruction. Due to time constraints we are unlikely to implement bounded Delaunays for mesh surface reconstruction.

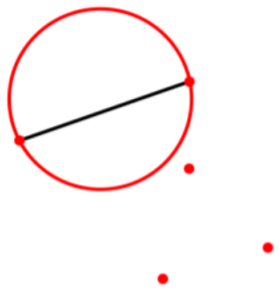


Fig. 3. A delaunay edge

2) Results: Our results in the stereo reconstruction of a point cloud were largely successful, with clearly recognisable point clouds being generated with very low numbers of

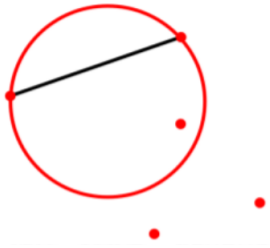


Fig. 4. Not a delaunay edge

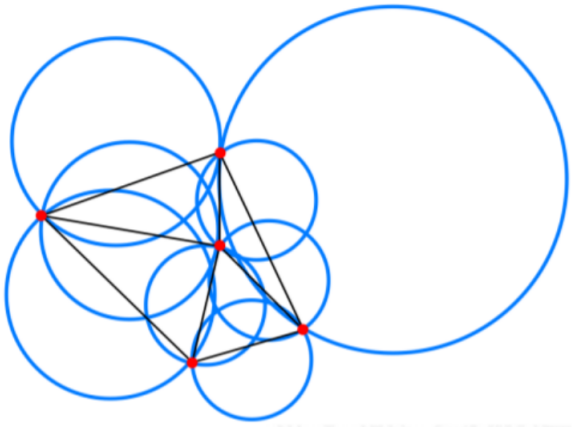


Fig. 5. A delaunay triangulation

output points. This worked across multiple pairs of image in the dataset and could return visually accurate depth in the point cloud. In our mesh implementation using scipy with custom thresholding based on triangle area, we were able to remove many outliers and return a recognisable dinosaur convex hull. However, further improvements could be made to the mesh by implementing an additional edge threshold to remove thin triangles.

Fig. 6. Sample image from Dinosaur dataset

B. Structure from Motion

Finding structure from motion (SfM) presents a similar problem to finding to structure from stereo vision. It is a collection of techniques in order to reconstruct 3D scene, camera pose from a set of correspondence points.

1) *Feature detection*

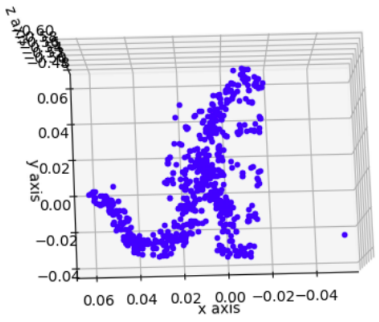


Fig. 7. 3D point cloud using 2 images

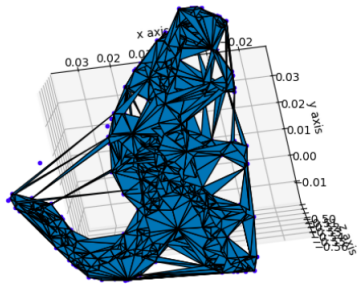


Fig. 8. Mesh model

- used SIFT algorithm from OpenCV library

2) *Feature matching*
   - used Brute-Force algorithm [9] to calculate the best matches
3) *Estimation of fundamental matrix*
   - Relates corresponding points in a stereo image pair based on the equation:

$$x^T * F * x' = 0$$

   where x, x' are stereo images and F is the fundamental matrix
   - 8 point algorithm [7] applied along with RANSAC to reduce outliers and mismatches
4) *Essential matrix*
   - Similar to fundamental matrix that relates point correspondences in two stereo images and can be computed with the formula:

$$E = K^T * F * K$$

5) *Decomposition in rotation and translation matrix*
   - The essential matrix can be decomposed into rotation and translation matrix using Singular Vector Decomposition [10] using homogenized keypoint:

$$E = U * \sum * V'$$

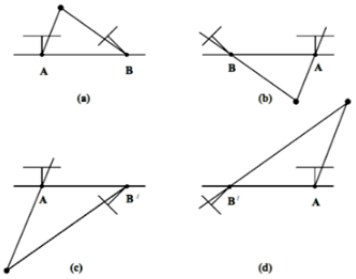
   Where U , V are unitary matrix, and $\sum$ is a diagonal matrix.

$$Rotation - Matrix = U * W * V$$

Where

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

This results in four possibilities of rotation and translation matrix of which one projects the key points in front of the camera.



6) Triangulation For every image pair, the first image lies at origin Thus the rotation matrix is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

and the translation vector is [0 0 0]. The coordinates of a point in world coordinate space is calculated based on the equation

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}^{-1} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where (X, Y, Z) are coordinates of a 3D point in world coordinate space, (u, v) are coordinate of pixel in images, s is the scale coefficient of an image

7) Triangulation across multiple image
   - Triangulation [8] using essential matrix suffers from scale ambiguity, thus reconstructed points computed across multiple images cannot be combined directly. This is formulated as Perspective-n-point problem, where camera pose must be estimated based on the 3D world coordinates and corresponding 2D images coordinates. The keypoints from the first image pair are used to start triangulation. Every image is added using keypoints that have already been used in reconstruction to compute the new projection matrix.
   - This projection matrix is used to triangulate new three dimensional points, thereby incrementally adding new 3D - 2D point correspondences.

1) Results: For testing purposes, we used 2 datasets, Fountain-P11 (8 images) and Dinosaur (36 images), both provided with camera matrices. While successfully managed to reconstruct 3D object using 2 images, we were not able to reconstruct good 3D models from multiple images. With an increasing size of dataset, we experienced higher rate of

outliers and mismatches. Due to a time constraint, we did not implement Bundle adjustment to optimize 3D structure and image parameters to improve our results. Nevertheless, the point cloud was still recognisable as the target object from certain angles.

Fig. 9. Sample image from Fountain-P11 dataset

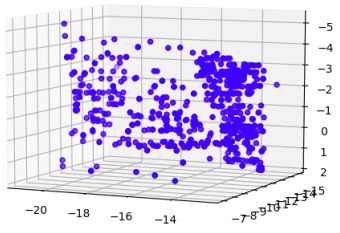


Fig. 10. 3D reconstruction using 8 images from Fountain-P11

VII. Packages Used

The main packages used in the implementation of this project were:

- OpenCV - Used for reading images, and the feature selection and matching stages of the 3D reconstruction pipeline
- Numpy - Used extensively for matrix manipulation of arrays and mathematical calculations
- Scipy - Used in the implementation of mesh generation using Delaunays with triangle size thresholding in order to remove outliers.
- Matplotlib - Primary package used in visualisation of the point cloud, as well as providing an alternative method of generating a mesh face from the point cloud without any outlier or concave face filtering.

Of note is OpenGL, which was investigated and implemented as a testing viewer platform for visualisation of generated point clouds. Whilst in our project we only generated sparse point clouds, which made us switch to Matplotlib due to its suitability for use and robust display functions of Matplotlib, OpenGL wouldve been a better choice if dense model reconstruction was implemented.

VIII. Conclusion

Throughout the project, we have gained much deeper understanding of various techniques to approach 3D reconstruction based on either 2 or multiple images. We successfully managed to implement 3D reconstruction using Stereo vision algorithm. Results from our Dinosaur dataset were very positive and it was clear to recognise the object. Next step was to implement a sparse 3D reconstruction from multiple 2D images, provided with camera matrices. We approached this problem with SfM algorithm using OpenCV library. As it was more challenging than expected, we experienced issues with matching keypoints from multiple images and reducing outliers. Also documentation of OpenCV was poor and lacked of working examples. Thus, we did not have enough time to work on our proposed extensions, which are described in the following section.

IX. Future Extension

- Image preprocessing
- Point cloud refinement using ICP or Bundle adjustment
- Dense 3D reconstruction for higher quality visualization
- Testing with larger datasets
- More user-friendly interface

X. Team Member Contributions

Member	Contributions
Ainharan Subramaniam	Stereo vision reconstruction coding, SfM reconstruction coding, matlab viewer implementation, general contribution to presentation and report
Ryan Cao	Feature matching and selection, viewer testing with OpenGL, mesh generation, general contribution to presentation and report
Quoc Anh Hoang	SfM implementation, framework implementation dataset testing, general contribution to presentation and report

Task	Approx. Time Taken
Dataset collection, General Research and Feature Matching & Selection Implementation	collection,
OpenGL viewer testing	0.5 weeks
3D point triangulation with stereo, matplotlib viewer usage implementation	1.5 weeks
SfM basic implementation	0.5 weeks
Mesh Generation	0.5 weeks

A. Datasets

- Vgg Dinosaur
- Fountain-P11

References

[1] Phototour.cs.washington.edu. (n.d.). Photo Tourism. [online] Available at: http://phototour.cs.washington.edu/ [Accessed 28 May 2018].

[2] Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., M. Seitz, S. and Szeliski, R. (2009). Building Rome in a Day. [online] Twelfth IEEE International Conference on Computer Vision (ICCV 2009): IEEE. Available at: http://grail.cs.washington.edu/rome/rome˙paper.pdf [Accessed 29 May 2018].

[3] Gava, C. (2011). Dense 3D Reconstruction. [pdf] DFKI. Available at: https://ags.cs.uni-kl.de/fileadmin/inf˙ags/3dcv-ws11-12/3DCV˙WS11-12˙lec10.pdf [Accessed 30 May 2018].

[4] Mcann, S. (2015). 3D Reconstruction from Multiple Images. [online] Available at: http://cvgl.stanford.edu/teaching/cs231a˙winter1415/prev/projects/CS231a-FinalReport-sgmccann.pdf [Accessed 4 May 2018].

[5] Salman, N. and Yvinec, M. (2009). Surface Reconstruction from Multi-View Stereo. [online] Sophia Antipolis, France: INIRIA. Available at: https://hal.inria.fr/inria-00422344/document [Accessed 26 May 2018].

[6] Schlnsen, R. (2017). Surface Reconstruction Iterative Closest Point, Delaunay-Triangulation. [online] Hamburg: University of Hamburg. Available at: https://tams.informatik.uni-hamburg.de/lehre/2017ws/seminar/ir/doc/slides/RobinSchluensen-Surface˙Reconstruction.pdf [Accessed 24 May 2018].

[7] R. Hartley, In defense of the eight-point algorithm, IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, no. 6, pp. 580593, 1997.

[8] R. I. Hartley and P. Sturm, Triangulation, in Proceedings of the ARPA Image Understanding Workshop 1994, Monterey, CA, 1994,pp. 957966.

[9] Docs.opencv.org. (2018). Feature Matching OpenCV 3.0.0-dev documentation. [online] Available at: https://docs.opencv.org/3.0-beta/doc/py˙tutorials/py˙feature2d/py˙matcher/py˙matcher.html [Accessed 31 May 2018].

[10] Mills, P. (2018). Singular Value Decomposition (SVD) Tutorial: Applications, Examples, Exercises — Statsbot Blog. [online] Statsbot Blog. Available at: https://statsbot.co/blog/singular-value-decomposition-tutorial/ [Accessed 31 May 2018].