3D Reconstruction from Camera Images

Cheng, J., Walter, K. & Yang, J.

1 Introduction / Problem definition

THE reconstruction of 3D models is an active area of research in computer vision and is highly relevant to industry and society at large. The aim of 3D re-construction is to capture the shape and appearance of real objects. Some examples of useful applications of 3D reconstruction include use in object identification, animation, digital archiving and augmented reality [1]. In this project we will explore and implement a system capable of creating a 3D model of an object using 2D photos of the object as input.

At the most basic level, this report will focus on the problem of reconstructing a 3D model from a sequence of 2D images. More specifically, we will be constructing a 3D model from a set of points obtained from the 2D images (point cloud).

2 Literature survey

This section will be divided up into two sections. The first section is the literature survey done for the interim report before we selected a method and the second section will survey the literature of the chosen method.

2.1 Broad survey of 3D reconstruction methods

There are many ways to infer 3D shape from still images. This literature survey aims to focus on the selected outline several of the more important ways this inference can be accomplished that have been documented in the research literature.

Shape from object attribute

One of the classic ways to obtain a 3D model from a still image is to inspect the shading of an object. This method was proposed by Horn [2] in 1975 and works due to the intensity of a region of the object dropping off as the shape moves into the plane of the picture. Essentially, as a shaded region becomes 'darker' we gain information about the change in direction of the surface normal and hence the orientation of the point. Orientation information about each point in the image gives us a description of the overall 3D shape of the object.

Object texture can also provide information about object shape. The idea behind this method is that textures are made up of instances of pattern elements arranged in some way. Either the distribution of the elements or the elements themselves can be distorted and the new state of the system can be analyzed. For example, Lobay used clusters of key points (obtained by SIFT) to identify pattern elements on clothes and used several instances of these elements to construct a model of the piece of clothing [3].

Like the analysis of shading on an object surface, we can analyze blurred regions of an object to infer shape. Nayar & Nakagowa took successive images of an object at different levels of focus. These focus levels were then interpolated to get accurate depth estimation [4].

Active Range-finding

Several methods that measure the distance from the camera to the object have been developed to infer shape.

Curless and Levoy [5] used a laser to sweep across a scene. As the laser passed over objects, the beam was deformed. The deformation of straight beam created by the laser was used with triangulation to estimate the 3D location of each point the beam touched.

A variation on this technique was proposed by Bouget and Perona [6]. Instead of a laser, a stick was used in front of a light that was shining on the object. The shadow produced by the stick was swept over the object and the deformation of the shadow was used in the same way that Curless and Levoy used the laser.

Point based representations

These representations involve describing the surface of an object with surface points, which can be thought of as physical particles with attractive forces and an orientation [7]. Having the points behave in this manner allows for interpolation of partial 3D data to account for 'holes' in the representation while retaining the ability to render the particle system as a continuous surface.

Model Based Representations

If we know something about the object that we are modelling, we can use this information to obtain a more accurate and detailed 3D model. For example, architecture is generally made of planar shapes that are oriented at 90-degree angles to each other. The system of Zhu and Kanade [8] is a good example of this, as it used several geometric primitives such as planes, cylinders and cones to describe various architectural features of the scene such as roofs or doors and windows.

2.2 Structure from motion

Structure from motion (SfM) is a well-studied technique to solve the 3D reconstruction problem. Indeed, this technique has been used to construct large 3D reconstructions of complex landscapes such as the city of Rome using pictures found on the internet [9].

The first step of SfM is matching image features between images in the dataset. The simplest approach is to match features in images in a pairwise manner (i.e. all images matched with all over images) [10]. But recent work has shown that this brute-force method can be significant improved by representing the images as high dimensional vectors and determining the most similar using dot products [9].

Once we have matched feature points we can triangulate feature points in two given images into 3D coordinates. The standard mathematical technique of triangulation is enough for this step of the SfM pipeline [11].

3 Definition of project goals

Our goal is to produce 3D reconstructions using multiple camera images of several different objects/scenes.

We will focus on qualitative measures for evaluation. We made this decision because the application of 3D reconstructions in industry generally requires that the 3D model looks to be an accurate reproduction of the actual object (i.e. in movie animation, heritage landmark archiving, medical imaging) so we consider this evaluation method to be adequate for our purposes.

To achieve the overall goal of the project, the following high-level objectives will be

pursued:

1. Implementation of image preprocessing pipeline
2. Creation of 3D point clouds
3. Surface reconstruction using point cloud
4. System testing

See section 4 of report for details of each objective.

4 Scope of project

The 3D reconstruction field is vast with many different approaches to the problem and any number of objects that could be reconstructed.

In this project we will limit our experiments to datasets that have camera calibration information provided.

We will limit our reconstructions to the following datasets:

- Middlebury College Multi-View Stereo dataset
    - 312 images of a temple
- University of Oxford Visual Geometry Group
    - 36 images of a toy dinosaur

5 Task decomposition

5.1 Image dataset

We will source our data from the Multi-View Stereo datasets available from Middlebury college and the University of Oxford Visual Geometry Group. These datasets contain a complete set of images along with camera matrices and ground truth data.

5.2 Image preprocessing

Most images taken by camera suffer from distortions (i.e. straight lines become curved) and may not be in a suitable state for effective feature detection. We will use various techniques (i.e. camera calibration) to counteract camera distortions and apply image processing steps learnt in class (i.e. thresholding, histogram equalization) to get the images ready for feature detection and extraction in the next step.

5.3 Feature extraction and matching

In this step we will take adjacent images from our dataset (i.e. one image and the one directly after it in the sequence) and apply the Scale Invariant Feature Transform algorithm to each one. We will match features between the two images using K-nearest neighbour and then use the RANSAC algorithm to fit a model to these feature points.

5.4 Build point cloud

We will calculate projection matrices for the current images and obtain 3D points by using these matrices along with the matched key points from the previous step.

5.5 Textured mesh creation

Finally, we will visualize the 3D model by inputting the point cloud obtained in the previous step into Point Cloud Library or MeshLab.

5.6 System testing

We will evaluate the results of our algorithm by comparing our models with models created by VisualSFM (using same input images).

6 Design

Our final design was based on the Structure from Motion concept.

6.1 Algorithms used

The main algorithms used in our reconstruction pipeline included Scale Invariant Feature Transform (SIFT), Random Sample Consensus (RANSAC) and triangulation. After these steps, software was used to create a mesh from the resulting

3D point cloud. In this section we provide a general description of these algorithms and then outline how they were used in this experiment.

SIFT

SIFT feature detection and description is used to find points within an image that are scale invariant (SIFT features are also somewhat invariant to other transforms as well) [2]. The invariance of these points makes them good for matching the feature between images.

The SIFT algorithm works on 8 x 8 'octaves' of a given image. Each of these octaves are blurred using gaussian filters (at varying sigma levels). Differences are taken between each layer to create a 'stack' of Differences of Gaussians (DoG). This stack of DoG forms a 3D scale space that can be searched for potential feature points (the extrema of this space are considered potential feature points). After identifying the extrema of the scale space, a SIFT descriptor is computed for each one. The SIFT descriptor is normalized histogram of gradients and magnitudes vector containing 128 elements.

Triangulation

In general, triangulation is the process of determining the position of a point given some other known points **[16]**. The name refers to the fact that triangles are used to locate the point of interest.

In our project, the points of interest are 3D points of the object we are reconstructing. The known points are SIFT feature points that we extract and match (in the 2D image dataset) in the earlier part of our algorithm pipeline.

To accomplish triangulation, we need more than just the feature points. The method is based on knowing the parameters of the camera matrices associated with a given image [11]. These matrices are obtained by decomposing the fundamental matrix (which itself is obtained by solving the coplanarity equations for our feature points [12]). Once we have the camera matrices we can solve the following equation for X_i which are the 3D points that correspond to matched 2D feature points:

$$x_i = PX_i$$

where x_i is a 2D feature point and P is the camera matrix [13].

RANSAC

RANSAC is used to fit a model between several matched points in two images (in computer vision) [14].

The RANSAC procedure proceeds by randomly selecting several matched feature points and fitting a model between them. After this fitting, the goodness of the fit is determined by looking at how many inliers (points within a given threshold of the line of fit) there are. This process is carried out N times and the model with the most inliers is selected.

K-Nearest Neighbour (KNN)

In general, KNN works by selecting a feature from one set and comparing it with every feature of another set (i.e. in features in two images) [15]. Some metric, such as Euclidean distance, is used to see how 'close' the two features are. In KNN we find the K closet elements and we assign the majority class label out of these K elements to the originally selected feature.

6.2 Pipeline

The pipeline employed in this experiment consisted of the previously described algorithms applied to our dataset.

1. Obtain dataset consisting of a sequence of images $[x_i, x_{i+1}, \dots, x_n]$
2. For images $[x_i, x_{i+1}]$ over $i = 0 \,..\, n-1$:
    a. Extract SIFT features
    b. Match features with KNN
    c. Calculate camera matrices
    d. Apply RANSAC to matched features
    e. Perform triangulation with camera matrices and matched feature points
    f. Convert triangulated points to 3D (triangulation returns 4D homogenous coordinates)
    g. Append triangulated 3D points to PLY file
3. Render PLY file in MeshLab.

6.3 Implementation

Preprocessing

Smoothing of images was carried out in OpenCV prior to feature extraction.

SIFT

The OpenCV *'xfeatures2d.SIFT_create()'* function was used to extract 2D features.

Triangulation

The triangulation algorithm was hand coded using the Python linear algebra package.

Mesh Generation

Meshes were created from point clouds using Mesh lab.

7.4 Justification of algorithms used

Triangulation

This is a well-known method that has proven to be useful when creating 3D reconstructions [16]. Since this is the goal of our project, the use of this algorithm is justified.

SIFT

3D reconstruction relies on the ability to match features across images that were taken at different positions/views. For this reason, the feature points need to be scale and rotation invariant. Since SIFT features have these properties, the use of the SIFT algorithm is justified.

RANSAC

There are two justifications for using RANSAC. Firstly, the RANSAC algorithm is known to produce robust models when given a set of 2D correspondences as input [14]. Secondly, SIFT detection can lead to 'outlier' features being identified. RANSAC can find an accurate model in the presence of outliers so is an excellent choice to fit models to correspondences of SIFT features between images [14].

KNN

In this experiment we used the *Fast Approximation to Nearest Neighbour (FLANN)* version of KNN. This is justified because the running time of our pipeline would have been significantly increased if we used the standard KNN algorithm. Also, FLANN-based KNN produces an adequate approximation to KNN [17].

8 Testing

Our testing phase involved producing 3D models from our datasets and then inputting these same datasets into VisualSFM software to obtain a benchmark image to compare against.

Example images of our datasets can be seen in *Figure 1* and *Figure 4*.

Test results for the dinosaur dataset can be seen in *Figure 2* (benchmark) and *Figure 3* (our result).

Figure 1: One of the 36 dinosaur images in the dataset. Each of the 36 pictures were taken from a different view of the toy.

Figure 2: Benchmark dinosaur created by VisualSFM software.

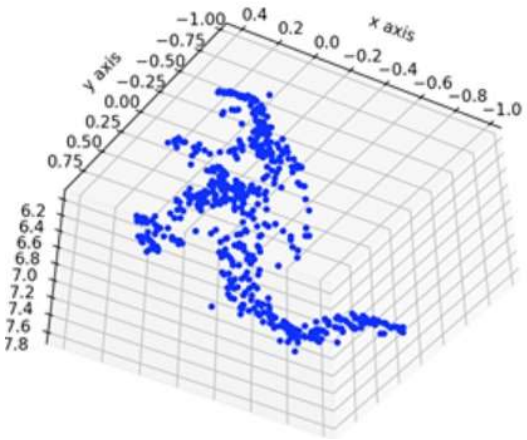

Figure 3: Dinosaur model produced by our algorithm pipeline.

Our algorithm produced a somewhat reasonable approximation to the toy dinosaur. Clearly, we have lost some fine detail and there appears to be more 'outlier' points with respect to the benchmark. This can be explained by extra steps that VisualSFM uses in its pipeline relative to our pipeline. For example, VisualSFM carries out a bundle adjustment to reduce reprojection errors.

Figure 4: One of the 312 temple images in the Middlebury dataset. Each of the 312 pictures were taken from a different view of the temple.

Figure 5: Benchmark temple created by VisualSFM software.

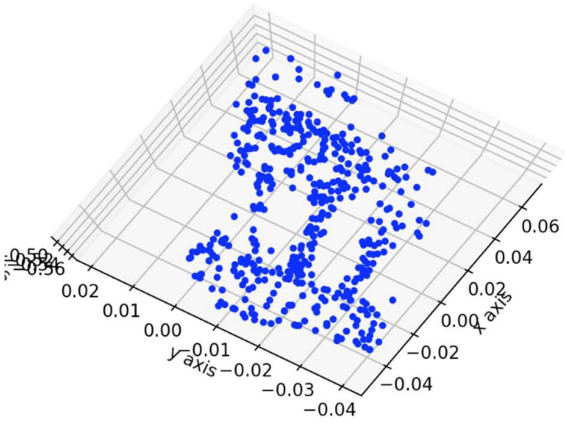

Figure 6: Temple model produced by our algorithm pipeline.

The loss of detail mentioned for the dinosaur in more pronounced in the temple. Even though we can make out the three pillars in the middle, its not clear that a non-biased observer would be able to identify this as the 3D structure of a temple. Clearly the benchmark is far superior in this case.

9 Conclusion and future extensions

In conclusion, our method of using triangulation with 2D SIFT feature points reproduced a decent approximation to the shape of the original object in a qualitative sense.

With optimization, this method would be useful in the industrial applications previously mentioned (movie or gaming 3D modelling, augmented reality).

Some optimizations/extensions that could be made to our method to make it more useful include:

- Adding an algorithm to compute surface normals of the point cloud (can interpolate these to create a meshed surface)
- Write an algorithm to reconstruct the object surface given the point cloud and surface normals
- Improve the efficiency of our pipeline (312 temple image to minutes when using more than 30 images).

10 Group member contributions

All three group members were tasked with researching/implementing our chosen algorithm pipeline in a different way. We did this because before we had started we had no way of knowing which implementation would work or produce the best results.

During the first week of the project, all groups members searched the literature for a method to use and find adequate datasets that we could work with (Week 9). Some small experiments were also done during this weak with image pre-processing and feature extraction of the chosen datasets (Jingran Cheng and Jingshi Yang). An attempt to implement a Patch-Bash Multi-View stereo algorithm was also made this week (Kane Walter).

Due to the class exam, work on the project was delayed until week 12. During this week, all three team members worked on a slightly different version of our chosen pipeline so that we could pick the best one and iterate on that. The implementations included Python + Matplotlib + OpenCV

(Jingran Cheng), OpenCV + MeshLab (Kane Walter) and OpenCV + Point Cloud Library (Jingshi Yang).

During week 13 we found that the Python + matplotlib implementation was giving the best results so further work was done on this implementation to improve results and clean up the code (Jingran Cheng). The final report and demo presentation were also created during this week (Kane Walter). A final experiment with a 'structured light' implementation was also carried out (Jingshi Yang) but we decided to not pursue this further.

Group members

Jingran Cheng	(z5103331)
Jingshi Yang	(z5110579)
Kane Walter	(z5033544)

11 References

[1] F. Remondino and S. El-Hakim, "Image-based 3D Modelling: A Review", *The Photogrammetric Record*, vol. 21, no. 115, pp. 269-291, 2006.

[2] R. Szeliski, *Computer vision*. London: Springer, 2011, p. 580.

[3] A. Lobay and D. Forsyth, "Shape from Texture without Boundaries", *International Journal of Computer Vision*, vol. 67, no. 1, pp. 71-91, 2006.

[4] S. Nayar and Y. Nakagawa, "Shape from focus", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, pp. 824-831, 1994.

[5] B. Curless and M. Levoy, "Better optical triangulation through spacetime analysis", In *Fifth International Conference on Computer Vision*, pp. 987-994, Cambridge, Massachusetts, 1995.

[6] J. Bouguet and P. Perona, " 3D Photography Using Shadows in Dual-Space Geometry", *International Journal of Computer Vision*, vol. 35, no. 2, pp. 129-149, 1999.

[7] R. Szeliski and D. Tonnesen, "Surface modeling with oriented particle systems", *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 185-194, 1992.

[8] Z. Zhu and T. Kanade, "Modeling and Representations of Large-Scale 3D Scenes", *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 119-120, 2008.

[9] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a day", *ICCV*, 2009.

[10] C. Wu, "Towards Linear-Time Incremental Structure from Motion", *2013 International Conference on 3D Vision*, 2013.

[11] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2003, p. 353.

[12] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2003, p. 12.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2003, p. 11.

[14] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2003, p. 117-122.

[15] I. Witten and E. Frank, *Data mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, Calif.: Morgan Kaufmann, 2005, pp. 129-136.

[16] R. I. Hartley and P. Sturm, "Triangulation", *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146-157, 1997.

[17] M. Muja and D. Lowe, "Scalable Nearest Neighbor Algorithms for High Dimensional Data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227-2240, 2014.