# 3D Reconstruction with Handheld Camera Project

· **Hao Ni**
· **Cheng bin Wang**

## Abstract

*Pictures and videos are 2D representation of 3D word, when viewing a collection of pictures or video, the human brain could extract information from each eye and construct 3D shape of the shown object. In theory, a computer can do something similar to what human brain that can do via logical calculations. Due to this, it is possible to reconstruct a 3D world with 2D collection of pictures.*

**Keywords** Structure from motion · 3D scene analysis · Perspective-n-Point problem· triangulation· 3D reconstruction

## 1. Introduction

Given a set of pictures of an unknown object, it possible to recover its 3D shape from information extracted from different views. The task is related with solving problems to do with extracting information from pictures, retrieving camera pose, and locating 3D points.

Reconstruct 3D model techniques is useful in many applications, such as photo tourism, 3D map view and many other applications.

Our project is aim to solve the above-mentioned problems, and reconstruct 3D point cloud with sets of frames with unknown camera poses.

## 2. Previous work

The last two decades have seen a dramatic increase in the capabilities of 3D computer vision algorithms. These include advances in feature correspondence, structure from motion, solving Perspective-n-Point problem and triangulation techniques have been developed in the computer graphics community, and image-browsing techniques have been developed for multimedia applications [16].

### 2.1 Feature Correspondence

Feature detection and matching techniques are important in many applications in computer vision and are core areas in 3D reconstruction. Some early development introduced finding feature points that are corner-like [1]. Interestingness could also be evaluated as high change of intensity using various sliding window functions and this is the backbone of the work done by people like Förstner [2] and then Harris and Stephens [3]. While eigenvalue based detectors like this, such as the Harris detector has rotation invariance, they are non-invariant to image scale.

More modern techniques such as Lowe's Scale Invariant Feature Transform (SIFT) use sampling of scale space to detect points that are invariant to changes in both scale and rotation [4]. This inspired the development of the Speeded Up Robust Feature (SURF) detector and has some gains in speed and robustness [4]. SURF is implemented in OpenCV where it is widely used for various projects and is known to perform reasonably well.

### 2.2 Structure from Motion

Three-dimensional reconstruction from unknown scene structure, camera position and orientation has its roots in the 1980s. It began with methods such as relative orientation [5] and global optimization techniques [6].

The gold standard for performing optimal D reconstruction from correspondences resides in sparse matrix techniques [7]. The math involved

cover necessary highlights such as obtaining the Fundamental matrix from corresponding points, and extracting Camera matrices.

The location of points in multiple images can eventually lead to an estimation of its 3D location. Multiple locations and matches call for the need for fundamental geometry to describe relationships between images. The general taxonomy of this was proposed by Scharstein and Szeliski [8], and this involves the concept of epipolar geometry. Figure 1 show the concept of epipolar geometry where a plane could be bounded at the 3D location of the point and two other locations of where this point appears on two pictures. All possible configurations of this plane will include epipolar line segments that intersect at an epipole on each image. This means knowing where the points are on an epipolar line segment means its partner is on the corresponding epipolar line segment of the second pictures. Find 7 or more good matching points will allow the for estimation of the Fundamental matrix which would describe the necessary epipolar geometry [7].

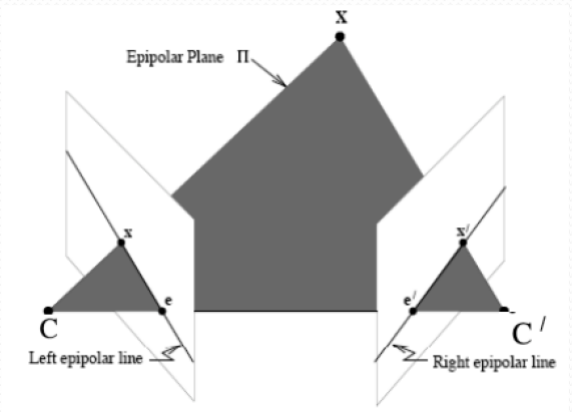



**Figure 1 - Epipolar Geometry**

The use of the Fundamental matrix itself to generate Camera matrices can give rise to projective ambiguity as pairs of camera matrices that differ by projective transformation can result in the same fundamental matrix. Fixing or setting camera parameters gives rise to a specialization of the fundamental matrix called the Essential matrix.

This has fewer degrees of freedom, as now there is only rotation and translation with a scale ambiguity, and this gives rise to fewer solutions when extracting camera matrices. Setting an initial origin point for the first camera matrix (P) allows the second camera matrix (P') to be computed once the Essential matrix is factored [7]. The factored solution gives two possible solutions for rotation and a choice of sign for translation. These solutions could be interpreted geometrically: 1) Two cameras pointing inwards towards object; 2) Accidentally reversing the input for case 1; 3) Two cameras pointing at an object, but facing away from each other; 4) Accidently reversing the input for case 3.

Providing both Camera matrices and a list of calibrated 2D points from both images allow triangulation to take place to estimate 3D locations. Due to the ambiguity in the possible solutions, this may not always work. In the case of cameras that are faced away from each other, rays pointing outwards will never intersect or even come close to matching. However, picking the solution where the rays could point in the other direction, where virtually the rays are heading in the direction behind the image planes, will give a perfectly acceptable 3D location as scale is ambiguous either way when working with the Essential matrix.

### 2.3 Perspective-n-Point problem

The perspective-n-point problem (PnP) is one of the issues in the process of pose recovery. It could be interpreted as deduction of camera pose matrix, given a set of 3D point, and their corresponding 2D projections [9].

Given by Fabrizio and J. Devars's example of PnP problem, the task is to solve following:

Donating 2D point on image plane as $p_i = (u, v)^T$ (and its homogeneous representation as $\tilde{p} = (u, v, 1)^T$ we need to find $\hat{P}_i^c = K^{-1}\tilde{p}$ [10].

The Opencv build in function solvePnP implemented the algorithm to resolve K from provided matching 3D-2D pair by minimizing projection error.

### 2.4 Triangulation

Once the camera Matrix is recovered, we suppose that a point x in R3 is visible in two images. The two known camera matrices are set as P and P'. Let u and u' be projections of the point x in the two images. From this data, the two rays in space

corresponding to the two image points may easily be computed. The triangulation problem is to find the intersection of the two lines in space [11].

In real world cases, it is rare for two rays to intersect in 3D space, as there is tiny rounding problem or other unsolvable issues. Previous researches provide many ways to optimize the intersect point with the two rays.

The method we applied was based on Richard and Hartley's Optimal Method of Triangulation algorithm [11]:

1. Parameterize the pencil of epipolar lines in the first image by a parameter t. Thus an epipolar line in the first image may be written as λ(t).
2. Using the fundamental matrix F, compute the corresponding epipolar line λ'(t) in the second image.
3. Express the distance function d(u, λ(t))^2+ d(u, λ(t))^2 explicitly as a function of t.
4. Find the value of t that minimizes this function.

## 3. Problem Decomposition and Project Scope

The overall problem of constructing 3D points out from multiple images can be broken down into two main steps: Initial 3D model from two images, and Refinement through adding more images. The initial 3D model is important as its accuracy will determine the quality of the model as more points are added. Additional points are added relative to the initial model as it serves as the base where everything is built upon. Every addition is based on the previous result and the process is iterative. The overall problem broken up into steps is outlined below.

Algorithm:

*1.Take first two frames as initial frame. Assume frame1 is taken at origin facing x axis, indicating P1 = I;*

*2.Conduct Feature Matching between frame 1 and frame 2.*

*3. Use matching result to compute Fundamental matrix.*

*4. Deduct Essential Matrix from fundamental Matrix.*

*5. Recover rotation and translation matrix [R|T] of frame 2 from Essential Matrix decomposition.*

*6. Using the matching pair of f1 and f2 to triangulate 3D points, upgrade point cloud, initial 3D model constructed. Keep a 2D-3D Lookup Table with respect to triangulation result.*

*7. Take next frame k from photoset, Do feature matching with previous frame k -1, Look up matching feature point's 3D coordinates from Lookup Table (if exist), pass matching 2D-3D pairs to solvePnP to get camera Pose matrix Pk. Update Lookup Table.*

*8. Triangulate frame k and frame k-1 matching features using Pk and P(k-1). Update Lookup Table and add points to point cloud.*

*9. Repeat 7-9 until finish all frames.*

## 4. Implementation

### 4.1 Improved Feature Matching

The feature detectors used in Part 1 of the project proved to provide plenty of good matches, but also had a significant amount of noise. As these points are used to generate 3D coordinates, which in turn act as a reference for images further along the pipe, it was vital that the feature detector is as robust as possible. Enhancements to the SURF feature detector were made to filter out bad matches and this included process such as KNN ratio test, symmetry test, and RANSAC.

A SURF feature detector was first used to detect interesting points in both images and provided a list of Key Points for both images. Descriptors were then computed for either images or Key Points. For the ratio test, a BruteForceMatcher obtained two nearest neighbors through comparing the descriptors from both discriptor1 to discriptor2, and from descriptor2 to descriptor1. This would be used for the symmetry test later, but in the meantime, features that have a very distinct nearest neighbor, or where the first neighbor is much better than the second neighbor, are desired over with similar neighbors. The ratio test removes matches that are deemed bad, and the rest

move through to the symmetry test. The two sets of DMatches obtained through the BruteForceMatcher that survived the ratio test are compared to see if a match from one image to another also has a pair in the other direction. Similarly, non-symmetrical matches are removed. The resulting matches are passed through a RANSAC test and these returns the Fundamental matrix. Outliers are filtered out.

Finally, we assume that if results of the matches of two images are used to be good references for further images, they must have plenty of matches. We set the threshold to be 60 matches. Otherwise, the user must provide better photos.

**4.2.Structure from Motion**

The steps following matching consists of finding the pose of one camera relative (P') to the camera set at the origin (P = [I|0]), and using this information to perform triangulation to provide a working 3D model for the rest of the algorithm.

As discussed in the background, projective ambiguity can be avoided by converting a fundamental matrix into an essential matrix. This can be done using a camera calibration matrix K, generated from a provided frame. The matrix is constructed to represent reasonable scaling and assumes the focus is at the very center of the image. Given both images are assumed to be as a result of the same cameras, the conversion can be simplified to:

Mat_<double> E = K.t () * F * K;

Decomposing E to obtain a P' is simplified in OpenCV as there is a very useful SVD class that provides U and V matrices as described in equation 9.14 in [7]. Multiplying these with the appropriate orthogonal matrix and skew-symmetric matrix gives us two matrices for rotation, and a translational vector that can have either positive or negative signs. Constructing the P' matrix gives four possible combinations, and it is difficult to tell which solution is valid in the real world unless triangulation is performed. As this operation is only performed once in the entire program run time, and involves only two frames, it was decided the user has to make the best judgment for what solution is required. Two things needs to be considered when deciding the solution for two images: 1) which image is on the left side and which is on the right side? 2) Are these images converging to the same points, or diverging away from each other? Answering these two questions correctly will give the best indication for which solution to choose.

Initial 3D points generated and their links to certain Key Points are important for the additions of further images. The model here is checked to see valid before lookup entries are added to a table to be used in solving PnP problem.

**4.3 Solving Perspective-n-Point problem**

Our Initial implementation was to use SFM to recover camera pose, the result shows that we the point cloud was shifted and rotated after each iteration, which in other words high projection error, and this is the result of accumulated error.

As discussed above, the Structure from motion implementation could recover camera matrix between different frames; however, the method is not applicable to other frames, which are not initial frames. The reason behind is that structure from motion is a rough method to predict camera pose, mismatching features and rounding problem all could leads to errors. Even though RANSAC is applied to optimize the result, the pose estimated still could not be applied to add new frames without further calibration.

In order to predict camera matrix P that not violated to the existing point cloud, we use predicted cloud point to predict new camera pose. There is an openCV function called solvePnP (), which takes a set of 2D-3D matching points, Calibration matrix and Distortion matrix to output Camera Matrix.

Therefore, it is possible to predict camera matrix with existing cloud point. To find the 2D-3D matching pair, we implemented a Lookup Table which stores 2D coordinates of current frame and its triangulated 3D coordinates. We keep track of 2D-3D pair during each iteration of processing new frames.

When new frames come in, feature mapping is performed with the previous frame. Keypoints are then searched up in the Lookup Table to get known 3D coordinates. Now we have a new 2D-3D pair of association.

By passing the 2D-3D pair of matching to solvePnP(), we have the new camera matrix that

minimizes projection error. Triangulation then generates new 3D co-ordinates.

### 4.4 Triangulation

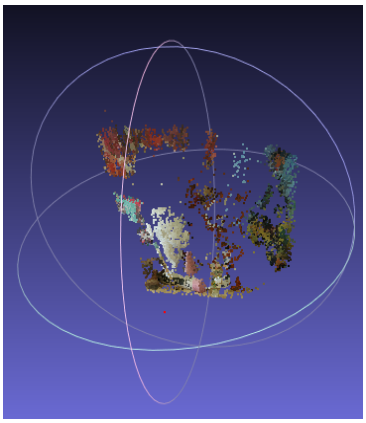
We implemented Linear-LS method described by Richard and Hartley [11]. The result is not as good as expected. We further extended our solution to Iterative-LS method by assigning weight in each iteration to generate more accurate result at cost of computation time.

### 4.5 Scene representation

The representation of point cloud is written in output file in ply format, it consist of the points' coordinates and RGB values. The example illustrate format of stored data:

-1.80715 0.417471 12.1074 37 19 20
-2.04797 -1.08848 12.6908 49 48 42

We use existing software Meshlab [12] to render our ply file.



*The red dot represents first camera location.

## 5.Results and Evaluation

### 5.1 Visual Result

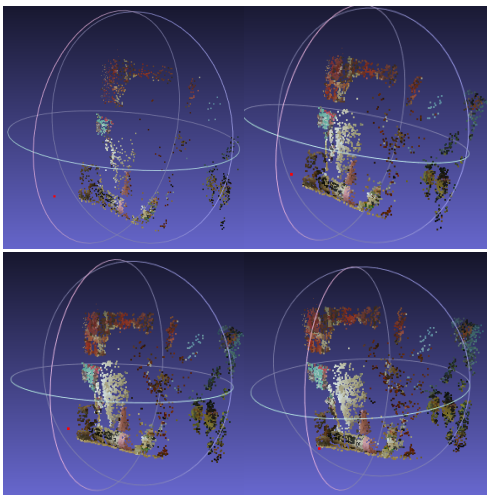
We have applied our application into different data sets to evaluate how well this application construct 3D scene visually.

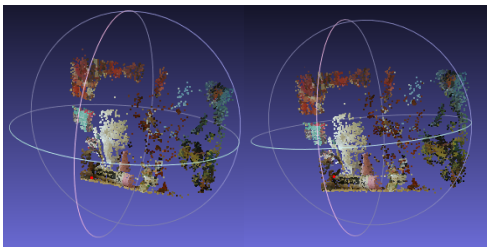
Our algorithm highly depends on the performance of feature matches, and though testing cases, the more matches will build better visual result.

The following pictures shows result of data set art [13], see below graph. The application takes the seven frames from left to right until it finishes seven iterations. The point cloud result after each iteration is shown, see graph below.

Original Data Set Art：

Result：

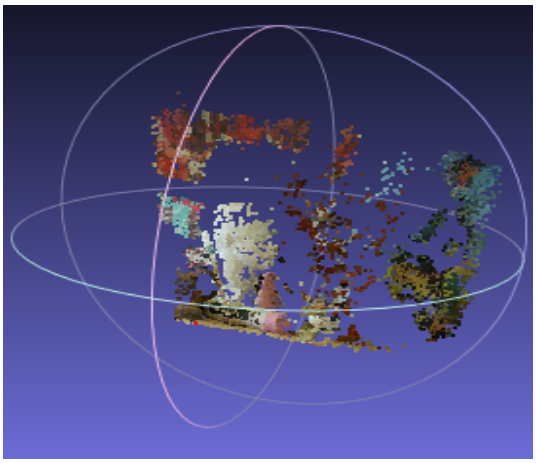




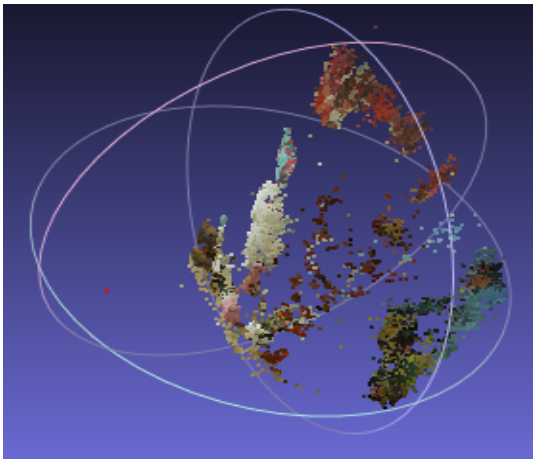
As shown above, the initial model start with 1800 cloud points, extracted from first 2 frames, it is sparse at beginning. As it extracting features from following frames, the cloud grows steadily, and begins to show quite a detailed 3D scene.

The final 3D model with 12083 cloud points is shown below:

Front View:

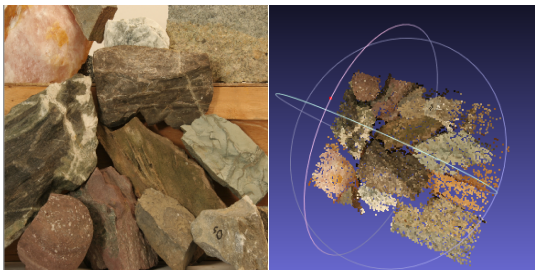


Top View:

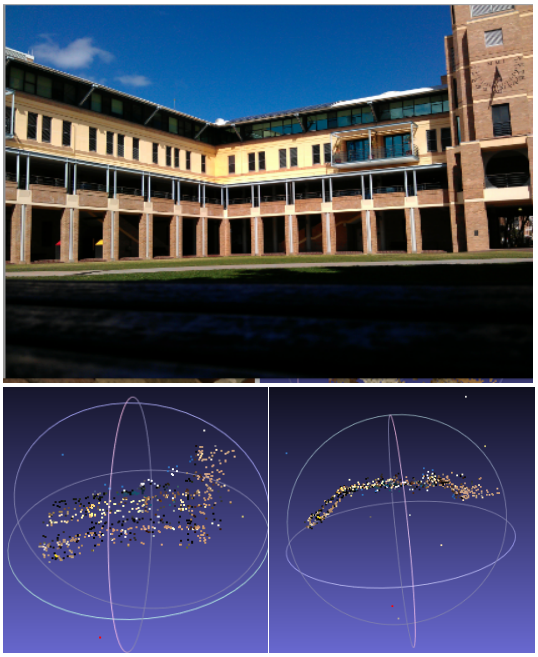


Here are more reconstruction results:

Rocks:



UNSW:



Round rock:

### 5.2 Tests and Evaluation

Apart from visual evaluation, we use statistics evaluation as well. Measuring real distance of object and features of object in real world is not always achievable. We use projection techniques to test and evaluate reconstruction result.

The table below shows, result of projection error of each data set, it is calculated by projecting 3D scene into an existing known frame camera location (ideally origin), and measures the error with the existing frame.

| Dataset | #Frames | #Points | Runtime | Error |
|---|---|---|---|---|
| **Art** | 7 | 12083 | 7.4 mins | 2.812 |
| **Rocks** | 7 | 25648 | 9.2 mins | 2.425 |
| **Round Rock** | 5 | 279 | 4 mins | 7.784 |
| **UNSW** | 4 | 355 | 3 mins | 10.683 |

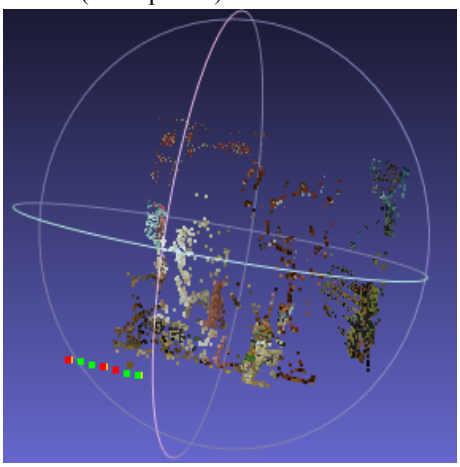
## 6.Discussion

### 6.1 Compared with existing system

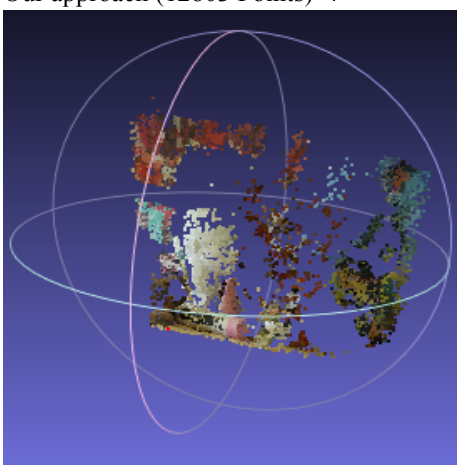
The application generates better visual result than existing well known photo tourism software Bundle [14] on selected datasets. 3D construction has relatively low error rates, and producing more cloud point.

Original Scene:


Bundle(5509 points):



Our approach (12803 Points) :



### 6.2 compare different data sets

As result showed previously, the Art and Rock data sets have better building result both visually and statistically (lower error rate). The reason is that the Art and Rock are taken with calibrated camera, while UNSW, building and round stone are taken with uncelebrated camera; the image is distorted as well.

To conquer the problem mentioned above, further techniques need to be applied such as camera calibration. However, the calibration part was beyond the initial project scope.

## 7.Conclusion

We did pair programming on this project for 6 weeks, and result is visually and statically satisfying. We have successfully improved feature matching

techniques, implemented structure from motion, triangulation, and using openCV function to solve PnP problem.

Further extension direction to the project could be following:

1. Camera calibration: calibrate camera with camera matrix K and distortion matrix T; this could be further done automatically rather than hand setting.

2. Filter images and use video input: videos could generate bad frames when moving fast, apply filtering techniques could keep good frames for reconstruction and ignore bad frames.

## 8. Acknowledgement

Many thanks to the efforts of our mentor Dimitri Semenovich who helped us with valuable advice and our lecturers Arcot Sowmya and Xiongcai Cai.

## 9.References

[1] Moravec, H. (1983). *The Stanford cart and the CMU rover*. Proceedings
of the IEEE, 71(7), 872–884.

[2] Förstner, W. (1986). *A feature-based correspondence algorithm for image matching International Archives Photogrammetry & Remote Sensing*, 26(3), 150–166.

[3] Harris, C., & Stephens, M. J. (1988). *A combined corner and edge detector*. In Alvey vision conference (pp. 147–152).

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool *"SURF: Speeded Up Robust Features",* Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

[5] Fischler, M. A., & Bolles, R. C. (1981). *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartograph*y. Communications of the ACM, 24(6), 381–395.

[6] Spetsakis, M. E., & Aloimonos, J. Y. (1991). *A multiframe approa*ch to visual motion perception. International Journal of Computer
Vision, 6(3), 245–255.

[7] Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry*. Cambridge: Cambridge University Press.

[8] Scharstein, D. and Szeliski, R. *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms* International Journal of Com-puter Vision (2002), 47(1):742.

[9] J. Fabrizio and J. Devars. *An analytical solution to the perspective-n-point
problem for common planar camera and for catadioptric sensor.* International
Journal of Image and Graphics, 8(1):135{155, January 2008. 42

[10] Ievgen http://computer-vision-talks.com/2011/11/pose-estimation-problem/

[11].Richard I. Hartley, Peter Sturm, *Triangulation*, Computer Vision and Image Understanding, Volume 68, Issue 2, November 1997, Pages 146-157, ISSN 1077-3142, 10.1006/cviu.1997.0547. (http://www.sciencedirect.com/science/article/pii/S1077314297905476)

[12] Meshlab http://meshlab.sourceforge.net/

[13] Middlebury stereo vision page [Online]. Available:http://vision.middlebury.edu/stereo/

[14] Washington University http://phototour.cs.washington.edu/bundler/

[15] Salton, G., *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*, Addison-Wesley, 1989.

[16] Snavely N, Seitz, SM , Szeliski, R: *Modeling the World from Internet Photo Collections*, 2007.