3D OBJECT RECONSTRUCTION USING MULTIPLE VIEW GEOMETRY: CONSTRUCT MODEL WITH ALL THE GIVEN POINTS

LEOW TZYY SHYUAN

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor (Hons.) of Mechatronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2010

DECLARATION

Signature	:	___________________________
Name	:	Leow Tzyy Shyuan
ID No.	:	07UEB06711
Date	:	15 April 2011

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“3D OBJECT RECONSTRUCTION USING MULTIPLE VIEW GEOMETRY: CONSTRUCT MODEL WITH ALL THE GIVEN POINTS”** was prepared by **LEOW TZYY SHYUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	_________________________
Supervisor	:	Dr. Tay Yong Haur
Date	:	_________________________

Specially dedicated to

my beloved grandmother, mother and father

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Tay Yong Haur for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

3D OBJECT RECONSTRUCTION USING MULTIPLE VIEW GEOMETRY: CONSTRUCT MODEL WITH ALL THE GIVEN POINTS

ABSTRACT

In this project, the main objective is to construct 3D model with all the given point obtained from SIFT detection. The proposed reconstruction of the 3D model starts from detection and matching of the interest point based on the similarity of the appearance. Combination of Hessian-Affine detector is used for wide baseline and 3D scene. Next, epipolar geometry is being computed. RANSAC is used for robust estimation to deal with outliers caused by occlusion. Derivation of extended Kruppa's equation which are responsible for describing the epipolar constrain of two projections of a general algebraic curves is used. Later on, the two-view reconstruction can be done from two selected view or key frames obtained from initial camera frames and reconstruct it. Two canonical camera matrices are obtained. The two-view matches are being triangulate to obtain initial 3D points. Optimal triangulation is chosen. From two key frames, we increasingly add another frame to the original to form key frame set. It is then being imposed with the previous key frame set by using camera pose from 3D to 2D points. Again, RANSAC is used to eliminate false features. The intrinsic parameters of camera are obtained to construct a final metric reconstruction of 3D model. Auto-calibration of the camera can also be used. In this case it will bypass the need of finding intrinsic parameter of the camera. For additional steps, dense stereo and texture mapping can be done by backprojecting the image intensities onto the 3D model for better appearance.

TABLE OF CONTENTS

DECLARATION	**ii**
APPROVAL FOR SUBMISSION	**iii**
ACKNOWLEDGEMENTS	**vi**
ABSTRACT	**vii**
TABLE OF CONTENTS	**viii**
LIST OF TABLES	**x**
LIST OF FIGURES	**xi**
LIST OF SYMBOLS / ABBREVIATIONS	**xiii**
LIST OF APPENDICES	**xiv**

CHAPTER

1	**INTRODUCTION**		**1**
	1.1	Aim	1
	1.2	Background and Motivation	1
	1.3	Objectives	2
	1.4	Scope Of Work	3
2	**LITERATURE REVIEW**		**4**
	2.1	Review of Journals and Articles on the Existing Method	4
3	**METHODOLOGY AND SYSTEM DESCRIPTION**		**9**
	3.1	System Overview	9
	3.2	Random Sample Consensus (RANSAC)	13
		3.2.1 Overview of RANSAC algorithm	13

		3.2.2 Implementation of RANSAC algorithm	19
	3.3	Triangulation	19
		3.3.1 Overview of Triangulation method	19
		3.3.2 Implementation of Triangulation method	22
	3.4	Bundle Adjustment	23
		3.4.1 Overview of Bundle Adjustment algorithm	23
		3.4.2 Implementation of Bundle Adjustment algorithm	25
	3.5	Building 3Dmap	26
4	**RESULTS AND DISCUSSION**		**27**
	4.1	Overview of the Experiment conducted in the system	27
	4.2	Experiments on the system	28
		4.2.1 Experiment 1: Different Texture	28
		4.2.2 Experiment 2: Number of frames	32
		4.2.3 Experiment 3: Number of feature points	35
		4.2.4 Experiment 4: RANSAC iteration	38
		4.2.5 Experiment 5: Number of plane	41
5	**CONCLUSION AND RECOMMENDATIONS**		**45**
	5.1	Conclusion	45
	5.2	Contributions	45
	5.3	Future Work	46
REFERENCES			**48**
APPENDICES			**50**

LIST OF TABLES

TABLE	TITLE	PAGE
Table 4.1:	Parameters involved in the 3D model reconstruction	30
Table 4.2:	Parameters involved in the 3D model reconstruction	34
Table 4.3:	Parameters involved in the 3D model reconstruction	37
Table 4.4:	Parameters involved in the 3D model reconstruction	40
Table 4.5:	Parameters involved in the 3D model reconstruction	43

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1: Examples of 3D reconstruction of image		2
Figure 2.1: Procedure of the proposed method from (Ebrahimnezhad1 et al. 2008)		8
Figure 3.1: Five major steps in 3D modeling process		10
Figure 3.2: Brief Flowchart of the implemented 2D to 3D conversion system		12
Figure 3.3: Point correspondence geometry between two views (R. I. Hartley 2003)		20
Figure 3.4: Illustration of bundle ajustement (R. I. Hartley 2003)		24
Figure 4.1: Image data set1		28
Figure 4.2: Image data set2		29
Figure 4.3: Image data set3		29
Figure 4.4: 3D model for data set1		29
Figure 4.5: 3D model for data set2		30
Figure 4.6: 3D model for data set3		30
Figure 4.7: Image data set1		32
Figure 4.8: Image data set2		32
Figure 4.9: 3D model for data se1		33
Figure 4.10: 3D model for data set2		33
Figure 4.11: Image of data set of interest		35

Figure 4.12: 3D model 1 with smaller number of feature points 36

Figure 4.13: 3D model 2 with larger number of feature points 36

Figure 4.14: Image of data set of interest 38

Figure 4.15: 3D model 1 with 150 iteration 39

Figure 4.16: 3D model 2 with 200 iteration 39

Figure 4.17: Image of data set of interest 41

Figure 4.18: 3D model 1 with number of plane = 200 42

Figure 4.19: 3D model 2 with number of plane = 500 42

LIST OF SYMBOLS / ABBREVIATIONS

c_p	specific heat capacity, J/(kg·K)
h	height, m
K_d	discharge coefficient
M	mass flow rate, kg/s
P	pressure, kPa
P_b	back pressure, kPa
R	mass flow rate ratio
T	temperature, K
v	specific volume, m^3
α	homogeneous void fraction
η	pressure ratio
ρ	density, kg/m^3
ω	compressible flow parameter
ID	inner diameter, m
MAP	maximum allowable pressure, kPa
MAWP	maximum allowable working pressure, kPa
OD	outer diameter, m
RV	relief valve

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A:	Computer Programming Listing	50

CHAPTER 1

INTRODUCTION

1.1 Aim

The aim of this project is to learn the necessary programming skills as well as familiarize with some existing software toolbox in order to achieve the objective of the project. Besides that, there is also a need to sharpen analyzing skill and also management skill to get this project done.

1.2 Background and Motivation

The word reconstruction means 'rebuilding' of something that has been torn apart. In some situation, for example in machine inspection context, it is necessary to acquire data one piece at a time in order to be able to view what is inside the body of a machine where the machine is not convenient for a person to reach for it. One of the big advantages of reconstruction of 3D object is that it allows a person to view or visualize all the data once it is been put back together again.

The creation of 3D models from multiple view geometry is one of the fundamental problems in computer vision and image –based modeling. A large body of research has been devoted to the problem of analyzing the 3D structure of a scene from multiple views. It is the process of capturing the shape and appearance of real objects. The multi-view theory is well understood when the scene consists of point and line features. This project is more about the combination of Mechatronics system as well as Computer Graphic Vision system. The programming algorithm of the 3D reconstruction by multiple view geometry is learned from computer vision system whereas the application of it can be related to Mechatronics filed.

The most important factor for the completion of this project is our interest in this topic. Without any interest in this topic, there is no way we can manage to achieve the results today. Besides, with some programming background learned from previous subjects, we are able to catch up with the software and programming introduced by others. This is also one of the factor to motivate us in the completion of this project Furthermore, knowing the powerful application of this topic in future is also one of the motivations.

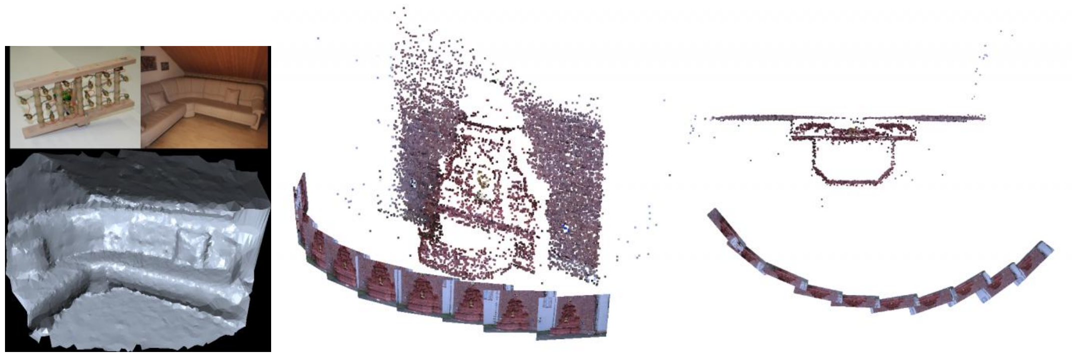
Figure 1.1: Examples of 3D reconstruction of image

1.3 Objectives

- To reconstruct 3D model using multiple view geometry
- To reconstruct 3D model in a more robust way
- To achieve the reconstruction results with higher percentage of similarity as possible

1.4 Scope Of Work

In order to achieve the objectives of this project, the following scope of work have been identified:

- Identify the elements involve in 3D reconstruction
- Identify the process flow of 3D reconstruction
- Identify the available software toolbox and algorithm developed, understand, analyzed and try to make improvement on the existing

CHAPTER 2

LITERATURE REVIEW

2.1 Review of Journals and Articles on the Existing Method

Reconstruction of surface model for a rigid scene using a moving camera through a sequence of photo images is a challenging problem. Up until today, it is still an active research topic especially in computer vision. In recent years, there has been extensive focus in the literature to recover the 3D structure from image sequence. The multi-view theory is by now well understood when the scene consists of points and line features.

In (S.D. Ma 1996), recovery of the 3D position of a conic section from two and three views is done given that we know the projection matrices. However, the method introduced is not so effective. In (C. Schmid 1998), they introduce a new method to recover the homographic matrix of the conic plane. Besides, the reconstruction of higher-order curves was introduced in the paper from (R. Berthilson 1999). In this paper, the matching curves are represented parametrically where the objective is to find a parameterization of each matching curves so that in the new parameterization, the points traced on each curve are the matching points. However, because the optimization is over a discrete parameterization, thus, for a planar algebraic curve with x degree, we will need $x(x+3)$ minimal number of parameters to solve the non-linear bundle adjustment machinery with some initial guess. Furthermore, the problems of recovering the camera projection matrices from matching projections of algebraic curves are growing. In order to solve this problem,

(J. Y. Kaminski 2000) show how to recover the fundamental matrix from matching conics with the result that at least 4 matching conics are required for a unique solution. Although the method used in this paper was able to generalize the results to higher order curves, it only considers planar curves.

Based on the research from (Kaminski et al. 2001), they introduced a number of new results in the context of multi-view geometry from the general algebraic curves without the restriction to the planar curves. The multi-view theory mentioned above will be fragmented when it comes to curve features, especially in the case of non-polar algebraic curves of general degree. Derivation of the extended Kruppa's equations for the recovery of epipolar geometry from two projections of algebraic curves is one of the methods introduced. The advantage of this system is that it required only minimal number of algebraic curves for a solution of the epipolar geometry as a function of their degree and genus. New results on the reconstruction of general algebraic curves from multiple views are being established. There are three different representation of curves proposed:

i. Regular point representation where the reconstruction from two views of a curve of degree *d* has two solutions. One with degree *d* and another with degree *d(d-1)*
ii. The dual space representation (tangents) for which a lower bound for the number of views necessary for reconstruction is derived as a function of curve degree *d* and genus
iii. New representation based on the set of lines meeting the curve which does not require any curve fitting in image space. A lower bound for the number of views necessary for reconstruction is derived but this time, is just as a function of degree alone, without the genus

In the paper of (Heuel and Forstner 2001) they introduced a geometric method for matching 2D line segments from multiple oriented images, optimally reconstructing 3D line segments and grouping the 3D line segments to corners. They present a calculus of projective geometric entities containing:

i. Representation of projective entities with its uncertainties

ii. Rules of constructing entities include both redundant and non-redundant projective entities

iii. Hypothesis test of relations between projective entities

This uses two developments in combining projective geometry and statistics. The geometric entities lines, points, and planes in 2D and 3D together with their uncertainties were represented in homogeneous coordinates. New entities including their propagated uncertainty can be constructed either directly or as estimation. Relations like incidence, parallelism, equality and orthogonality between lines, points and planes can be tested statistically based on a given significant level. An algorithm which exploits only the geometric constrain in multiple view is introduced. It is used to solve:

i. Matching 2D line segments under multiple views

ii. Optimally reconstruct 3D line from matched 2D lines

iii. Group the line segments to higher aggregates (for example, the 3D corners that consists of a corner point and two 3D line segments)

Using this method, matching of 2D line segments in multiple views are possible and 3D line segments were optimally reconstructed using ML-type estimation scheme. One of the advantages of the proposed method is that it gives straight forward and reasonable results. Besides, the result does not use any image intensity. It is based on geometrical constrains with error propagation and statistical test, driven by estimated precision of the image features. There is no need of threshold other than a significant value for the hypotheses test when we want to match the 3D lines.

Improvements can be done where the estimation method using GauB-Helmert model to estimate unknown entities can be extended to a more robust estimation such that a feature within a match can be identified as an outlier. Besides, for the matching and grouping of 3D lines, adding other cues will help stabilize the solution. Furthermore, the addition of the cues can improve the speed of the program since a good match hypothesis is found earlier that with only geometric cues. One of the suggestion is that we can make use of the 2D neighborhood relationship of points, lines and image regions to infer a 3D neighborhood relationship of 3D line segments.

Similar method can also be found in (Kanatani cf 1996) where they used monograph to presents techniques for statistical geometric reasoning. The homogeneous representations were used but they do not fully make use of the elegant projective formulations and thus, lead to more complex expressions. Besides Kanatani, (Criminisi 2001) uses covariance matrices of homogeneous entities and analyze the neglible effects of second order terms for mean and variance. However, these two methods do not perform as powerful as the method proposed by Heuel and Forstne.

According to (Ebrahimnezhad1 et al. 2008), different algorithm are used because of the wide range of options, for examples, the image projection model, number of cameras and available views availability of camera calibration, feature types and model of the scene. For a static, fixed object with moving camera, the shape and motion recovery problem can be formulated by finding out the six motion parameters of the object. For example, the orientation displacement and position together with the accurate 3D world coordinates for each point. Based on (Hartley 2003) the standard method of rigid motion recovery has been developed in the last decade based on sparse feature points. This method typically assumes that correspondences between scene features like corners or surface creases have been established by tracking technique. The disadvantage of this method is that it only reconstructs sparsely distributed 3D point. Generally, motion estimation method will suffer from instability due to quantization noise, measurement errors and outliers in input data. In order to overcome this problem, a different robust estimation technique known as RANSAC (RANdom SAmple Consensus) is introduced. This is a successful method to deal with outliers.

In (Ebrahimnezhad1 et al. 2008), a constructive method is proposed to moderate the bias problem using curve based stereo matching and robust motion estimation by tracking the projection of the space curves in perpendicular double stereo images. The advantage of this method is it not only can increase the motion estimation accuracy but also can reduce the problem of statistical bias. Besides, this method is more robust against the perturbation of the edge points. Any large error in depth direction of stereo rig 1 is restricted by minimizing the error in parallel direction of stereo rig 2 and vice versa. The curve matching scheme proposed is also robust against color maladjustment of cameras and shading problem during object motion. Although this method is robust in the curve reconstruction and matching, the method used needed calibrated camera to perform the function.

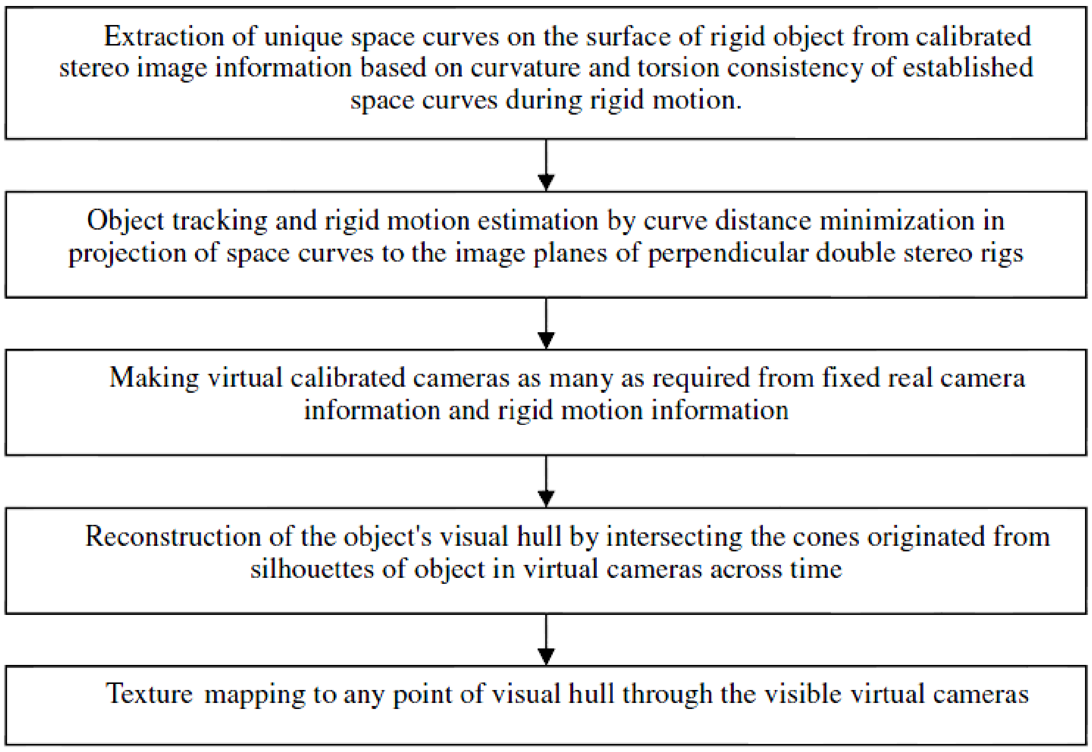
Extraction of unique space curves on the surface of rigid object from calibrated stereo image information based on curvature and torsion consistency of established space curves during rigid motion.
Object tracking and rigid motion estimation by curve distance minimization in projection of space curves to the image planes of perpendicular double stereo rigs
Making virtual calibrated cameras as many as required from fixed real camera information and rigid motion information
Reconstruction of the object's visual hull by intersecting the cones originated from silhouettes of object in virtual cameras across time
Texture mapping to any point of visual hull through the visible virtual cameras

Figure 2.1: Procedure of the proposed method from (Ebrahimnezhad1 et al. 2008)

As the conclusion in this section, all method proposed by the papers and journals have their own advantage and disadvantage. Some of the technique are powerful, for example method used in (Ebrahimnezhad1 et al. 2008) but it is performed with calibrated camera, which is not what we want in our case. The extended Kruppa's Equations used in (Kaminski et al. 2001) can be considered for describing the epipolar constraint of two geometry projections. In order to minimize occlusion, RANSAC can be included into the algorithm. Method from (Heuel and Forstner 2001) can be used for matching 2D line segments from multiple oriented images. It also optimally reconstructs 3D line segments and grouping the 3D line segments to corners.

CHAPTER 3

METHODOLOGY AND SYSTEM DESCRIPTION

3.1 System Overview

This chapter mainly describes the implemented steps for the multiple-view reconstruction of 2D image to 3D image. There are total of five major steps in the 3D modeling process that have been implemented: image feature detection and registration using scale invariant feature transform (SIFT), removing the outlier by exploiting the two-view geometry using the random sample consensus (RANSAC), estimating the projective 3D structures through a process called triangulation, bundle adjustment, constructing a 3D map and finally converts into file that can be view in OpenSceneGraph. Choosing OpenSceneGraph as a toolkit to view the built 3D model is because it is an open source high performance 3D graphics toolkit especially useful for visual simulation, games, virtual reality, scientific visualization and modeling. Figure 3.1 is shown to briefly describe the process flow of the steps to create a 3D model.

In each section later, a more detailed graph will be shown. In this work, 4 views, 6 views and 8 views from different viewpoints are processed. They are found that using more views will have a better 3D reconstruction results. Results will be discussed in Chapter 4.

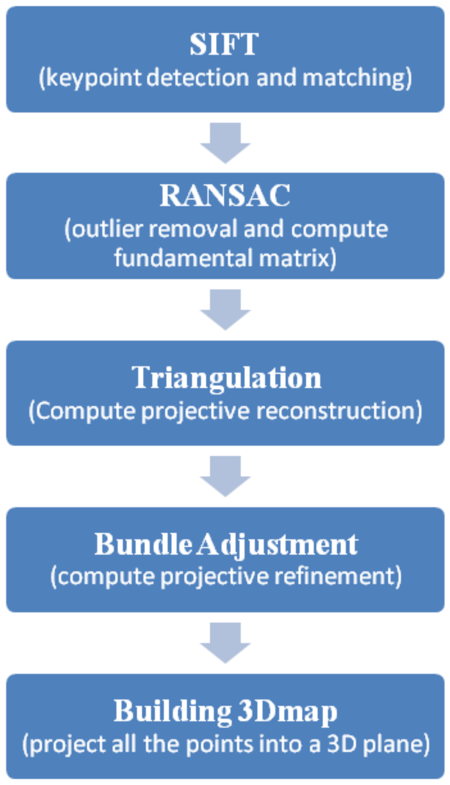
SIFT
(keypoint detection and matching)
RANSAC
(outlier removal and compute fundamental matrix)
Triangulation
(Compute projective reconstruction)
Bundle Adjustment
(compute projective refinement)
Building 3Dmap
(project all the points into a 3D plane)

Figure 3.1: Five major steps in 3D modeling process

Refer to Figure 3.2, it shows the implementation of 2D image to 3D model overall process flowchart. Firstly, the multiple view images were being captured from the video. All the images will then being process through an algorithm called scale invariant feature transform (SIFT) for key point detection, extraction, and matching purposes. At this point, feature detection and matching is implemented between the reference view and one of the other views at each time. The chosen image to be the reference of the system is the front view of the object. Later on, two-view geometry is estimated for each pair of views using the corresponding feature points in the two views used. The detailed of this step were discussed and done by the author’s partner.

For removing the outliers in the feature matching, a robust algorithm is being implemented. The implemented algorithm is called random sample consensus (RANSAC) for matching features. In this step, the projection matrices between the reference views are estimated from the inliers of the feature points. All the common points between all the views were determined from the set of inliers. Later on, the structure of the 3D object and the positions of the multiple cameras were retrieved using the common feature points and geometries of all views. Triangulation is being implemented to produce the projective reconstruction of the 3D object. Besides, bundle adjustment is being applied for refinement of the projection matrices. After that, the refined common 3D feature points were being back-projected to multiple 2D views in order to calculate the average reprojection errors of the 2D points. From this point, if the average reprojection error is found to be smaller than a threshold value preset by the author, the projective reconstruction is then being computed into a matrix form. On the other hand, if the error is larger than the threshold value, the procedure is being repeated starting from the RANSAC step to re-estimate the geometry and structure from new sample sets of feature points. The process is being repeated until all the points of interest were being evaluated. Followed by this, sparse depth maps were estimated and all the points were being plotted onto the 3D map plane. A 3D model was then being constructed.

The details of RANSAC, triangulation, bundle adjustment and metric upgrade, building the 3D map plane and 3D model will be discussed in the following sections.

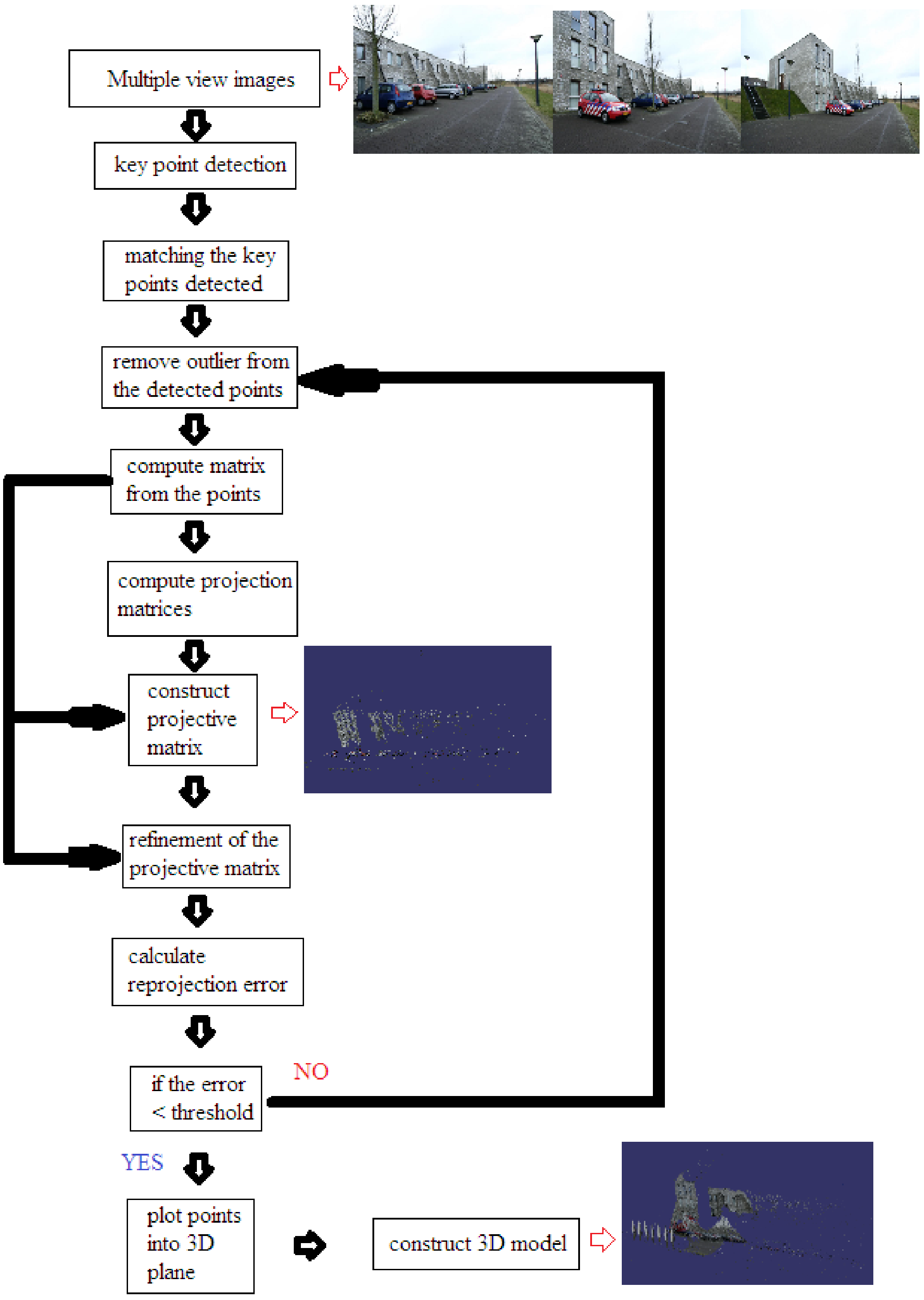
Multiple view images
key point detection
matching the key points detected
remove outlier from the detected points
compute matrix from the points
compute projection matrices
construct projective matrix
refinement of the projective matrix
calculate reprojection error
if the error < threshold
NO
YES
plot points into 3D plane
construct 3D model

Figure 3.2: Brief Flowchart of the implemented 2D to 3D conversion system

3.2 Random Sample Consensus (RANSAC)

3.2.1 Overview of RANSAC algorithm

Since the descriptor of feature points were used for feature matching in multiple views, the feature detection and matching steps were determined with respect to the local sub-region around the feature points. However, there might be cases where the matching points between the feature points were inaccurate. All these mismatch points were called outliers. So, RANSAC was being introduced in this step for outlier elimination and model estimation (M. A. Fischler 1981). This method was being introduced long time ago but it is still widely used until now due to the effectiveness of this method. It is popular for the effectiveness of eliminating the outliers and to estimate the two-view geometry using the inlier feature points. RANSAC is being chosen over conventional smoothing techniques such as Least Square Optimization (P. Torr 1997) because it is a robust estimation algorithm which operates in an opposite manner. Instead of using a large data set to obtain an initial solution followed by removing the invalid data, RANSAC uses small initial data set to calculate the solution model. The accuracy of the solution was being determined according to the applicability of the solution to other data points with respect to certain prerequisites.

Here, a brief procedure of RANSAC algorithm was being described. For example, a data set S is being used. In order to robustly fit data set S into a model and remove the outliers, a relatively small set of s samples is randomly chosen from input data set S. The initial solution model is calculated using this small sample set. Later on, the whole data set of S is being fitted into this solution to calculate the distance from the original data value. Data samples which the distance is within the threshold value t are the inliers data set S_i. S_i is also called the consensus data set which contains only inliers. If the ratio of data in S_i over S is larger than previous trials, the sample trial number N is re-estimated using this ratio of inliers. Besides, if the current trial count is larger than N, the solution model is re-estimated using all the inliers in S_i and the procedure is being terminated. On the contrary, if the current trial count is less than the estimated trial number N, a new set of s samples is selected and the calculation is repeated from the first steps. In the end, the largest consensus set S_i is selected after a number of N trials and the model is re-estimated using all sample points in S_i.

It is not necessary to compute every possible sample set s from all points. The number of iteration N can be determined using the following estimation method. With probability p (taking p as 0.99), at least one of the selection is a sample set of s data points that are free from outliers. Let ω to be the probability that a selected sample point is an inlier. It gives the equation

$$\varepsilon = 1 - \omega \tag{3.1}$$

$1 - p$ denotes the probability that all selected sample sets are not free from outliers, and each set contains at least one outlier. The probability can also be represented as

$$(1 - \omega^s)^N = 1 - p \tag{3.2}$$

where the $1 - \omega^s$ is the probability that, in one sample set, there is at least one outlier. The number of sample sets N can be determined as follows:

$$N = {\log(1 - p)}/{log(1 - \omega^s)} \tag{3.3}$$

The interest here is in applying RANSAC to estimate the two-view geometry by finding the fundamental matrix F and removing outliers. According to (R. I. Hartley 2003) the sum of Sampson distances of each pairs of the corresponding points is used to represent the error for the fundamental matrix estimation. The error is denoted as F_{error}

$$F_{error} = \sum_{i=1}^{8} d_{sampson}(x_1^i, x_2^i) \tag{3.4}$$

where x_1^i is the feature point of the *i*th pair of correspondences in the *j*th view (j = 1,2) and $d_{sampson}(x_1,x_2)$ is the Sampson Distance between x_1 and x_2 and is given by the formula below

$$d_{sampson}(x_1,x_2) = \frac{((x_2)^T F x_1)^2}{(Fx_1)_1^2 + (Fx_1)_2^2 + (F^T x_2)_1^2 + (F^T x_2)_2^2} \tag{3.5}$$

Where $(Fx_j)_n$ is the *n*th element in the product of F and x_j. The fundamental matrix estimation error F_{error} needs to be small to ensure the reliability of the estimated fundamental matrix.

Another thing that needed to be take note was that the 3D points corresponding to the randomly selected set of 2D feature points should not be lying in the same plane in the 3D space. This is because if all lie on the same plane in 3D space, the estimated geometry model is not general to estimate the depth information for all the 3D points. To avoid that, a homogeneous matrix H between each pairs of corresponding 2D feature points in two views,

$$x_2^i = Hx_1^i \ (i = 1,2, \dots n) \tag{3.6}$$

was computed using Single Value Decomposition (SVD) method. SVD is a very powerful set of technique dealing with sets of equations or matrices that are either singular or numerically very close to singular. From the equation, we know that the cross product of x_2^i and Hx_1^i is zero. According to (R. I. Hartley 2003), suppose

$$x_2^i = (a_2^i, b_2^i, c_2^i)^T \qquad x_1^i = (a_1^i, b_1^i, c_1^i)^T \tag{3.7}$$

and Hx_1^i can be expressed as

$$A \cdot h = \begin{bmatrix} 0^T & -c_2^i {x_1^i}^T & -b_2^i {x_1^i}^T \\ c_2^i {x_1^i}^T & 0^T & -a_2^i {x_1^i}^T \\ b_2^i {x_1^i}^T & a_2^i {x_1^i}^T & 0^T \end{bmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \tag{3.8}$$

where $h_j(j = 1,2,3)$ is the j^{th} column of H. In (3.8), the matrix A has a rank of 2 and only two of the three equations are linearly independent. For eight pair of correspondences, by taking the first two equations in (3.8), a 16 x 9 matrix can be formed as

$$B \cdot h = \begin{bmatrix} 0^T & -c_2^1 {x_1^1}^T & -b_2^1 {x_1^1}^T \\ c_2^1 {x_1^1}^T & 0^T & -a_2^1 {x_1^1}^T \\ \cdots & \cdots & \cdots \\ 0^T & -c_2^8 {x_1^8}^T & b_2^8 {x_1^8}^T \\ c_2^8 {x_1^8}^T & 0^T & -a_2^8 {x_1^8}^T \end{bmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0 \tag{3.9}$$

After the computation of *SVD* of *B*, the right-singular column vector that corresponds to the smallest singular value is the solution for *h*. By making *h* back to a 3 x 3 matrix, the homography matrix *H* can be generated. In order to compute the error in the estimation of the homography matrix *H*, the sum of the reprojection errors between the corresponding 2D feature points is

$$H_{error} = \sum_{i=1}^{8} (x_2^i - H x_1^i)^2 \tag{3.10}$$

The H_{error} will be large if the 3D points corresponding to the pairs of the 2D feature points do not lie on the same plane.

The RANSAC procedure introduced by (R. I. Hartley 2003) can be summarized as follows,

1. Initial values are set as $s = 8$, $\omega = 0.01$, $p = 0.99$, $t_{HF} = 4000$ and $t = 0.002$. Where s is the size of random sample set, ω is the probability that a sample is inlier, p is the probability that all selected sample sets are inliers, t_{HF} is the threshold for the ratio of the F_{error} and H_{error} and t is the threshold to select the inliers.

2. The N value is calculated using (3.3)
3. A random sample set of 8 correspondences ($s = 8$) is chosen and fundamental matrix F is calculated using the 8-point algorithm.
4. The ratio $R = H_{error} / F_{error}$ is used to evaluate the accuracy of the obtained solution of the fundamental matrix. If $R > t_{HF}$, the solution is satisfactory and the procedure proceeds to next step. On the other hand, if $R < t_{HF}$, the procedure is repeated from step 3 by randomly selecting another eight pairs of corresponding feature points.
5. For all the corresponding pairs of points x_1 and x_2, the Sampson distance $d_{sampson}(x_1, x_2)$ in (3.5) is calculated. The inliers that are consistent with F are determined to be the feature points whose Sampson distance is smaller than the selected threshold t.
6. The ratio of the number of inliers to the number of the whole set of feature points is calculated and is denoted as probability ω that a data point is an inlier. If ω is larger than the previous computed ω value, the sample trial number N is re-estimated using (3.3). If the current trial count exceeds the estimated N, the procedure will continue to next step. Otherwise, step 3 is repeated.
7. After N trials, F is re-estimated using the largest set of inliers.

Based on the fundamental matrix F, the projection matrices P1 and P2 for two pairs of view can be calculated. As stated in (R. I. Hartley 1992), the two 2D points x1 and x2 corresponding each to the projection of the 3D point X into two different views, are related as:

$$x_2^T F_{21} x_2 = 0 \tag{3.11}$$

and

$$x_1^T F_{21} x_2 = 0 \tag{3.12}$$

From (3.11) and (3.12), the fundamental matrices, F_{12} and F_{21} can be related as

$$F_{21} = F_{12}{}^T \tag{3.13}$$

For simplicity, the fundamental matrix F_{12} is denotes as F. Supposing F is the fundamental matrix between view 1 and view 2, it satisfy (3.11). The projection matrices P_1 and P_2 are determined as describe below. According to (R. I. Hartley 2003), given fundamental matrix *F*, the pair of camera projection matrices can be defined as

$$P_1 = K[I|0] \tag{3.14}$$

$$P_2 = K[SF|e_2] \tag{3.15}$$

where *S* is and skew-symmetric matrix, and e_2 is the epipole in the second view and satisfy

$$e_2^T F_{12} = 0 \tag{3.16}$$

The skew-symmetric matrix is a square matrix where its transpose is equal to its negative, represented as

$$S = -S^T \tag{3.17}$$

The projection matrices $P'_1 = K\ [I|0]$ and $P'_2 = K\ [A + a\ \ v\ T \mid \lambda\ a]$, where λ is a scalar , *a* is a 3x 1 vector, and *v* is a 3 x 1 vector, have the same fundamental matrix as the canonical pair $P_1 = K\ [I|0]$ and $P_2 = K\ [A|a]$. By assigning S in (3.17) to be $[e_2]_x$ where $[e_2]_x$ is being defined as

$$[e_2]_x = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{3.18}$$

The general form of the camera projection matrices can be expressed as follows (R. I. Hartley 2003):

$$P_1 = K[I_{3x3}|0_{3x1}] \tag{3.19}$$

$$P_2 = K[[e_2]_x F + e_2 \cdot v^T | \lambda e_2] \tag{3.20}$$

Where *v* is 3 x 1 vector, λ can be any scalar value and the epipole e_2 is the left singular vector corresponding to the smallest singular value of the SVD of the fundamental matrix *F*.

3.2.2 Implementation of RANSAC algorithm

Referring to the discussion above, RANSAC algorithm is implemented to remove the outliers from the matching points that are calculated from SIFT. It can also used to estimate the fundamental matrix *F* between the reference view and other view. The 'ransacfitplane.m' function is used for RANSAC. In each loop, two cells of the 2D corresponding feature points, with the first cell corresponding to the reference view, are taken out of the SIFT cell array to be used as the input for this function. The first input element to this function is the set of 2D feature points in the reference view after SIFT and the second input element is the set of 2D feature points in the i^{th} view after SIFT. Output of this is the fundamental matrix *F* as well as the 2D feature point indices of the inliers between the reference view and the i^{th} view. For each RANSAC operation, the inliers between two views are calculated. This procedure is implemented between the reference view and each of other views for seven times (taking 8 views for evaluation). In the system, usually the number of inliers will decrease as the number of views increases.

After RANSAC, the common 2D inliers feature points which are common in all the eight views are the output. The projection matrices for all the views are being calculated. The RANSAC toolbox is provided by Peter Koversi from The University of Western Australia (RANSAC code). The code is being modified by adding step4 in the RANSAC procedure mention in section 3.2.1.

3.3 Triangulation

3.3.1 Overview of Triangulation method

From the RANSAC algorithm in the previous step, the inliers among the corresponding 2D feature points as well as the projection matrices are calculated for the multiple views. Triangulation method is implemented to estimate the 3D geometry with respect to the common 2D feature points in all views.

Ideally, the intersection of the two lines that are formed by connecting each of the matching 2D points and their corresponding camera centres can be easily computed to get the corresponding 3D point in space. However, due to noise and digitization errors, it is possible that the intersection of these two rays does not exist in 3D space. So, triangulation is being used for 3D point estimation.

The geometry between two views of the same scene can be represented by using the epipolar geometry. The epipolar geometry is illustrated in Figure 3.3

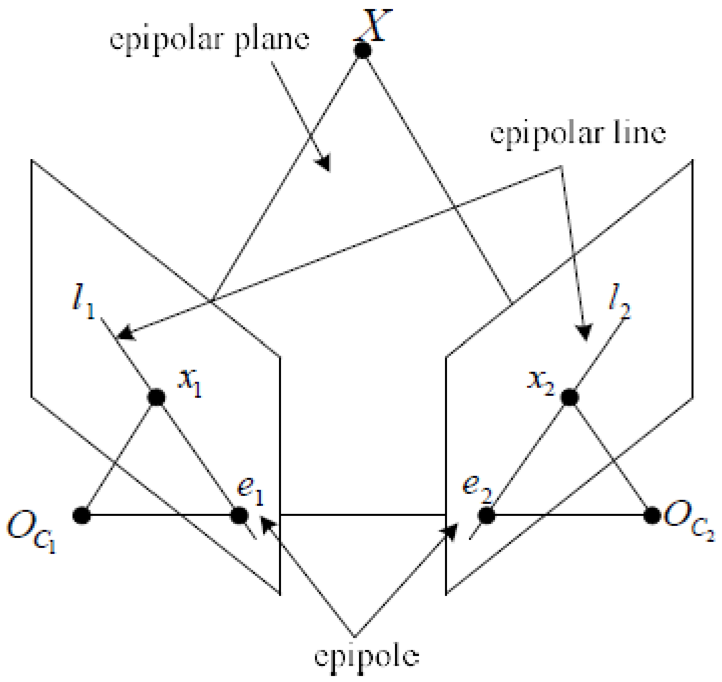

Figure 3.3: Point correspondence geometry between two views (R. I. Hartley 2003)

The 3D points are reconstructed using a simple SVD-based algorithm similar to the one shown in Figure 3.3. For each 2D feature point,

$$x = PX \tag{3.21}$$

is used to relate the 2D point *x* and the corresponding 3D point *X*. The cross product of the 2D point *x* and *PX* is calculated for the corresponding 2D points in 8 views, x_i ($i = 1,2,\ldots,8$), as

$$x_i \times P_i X = 0 \tag{3.22}$$

Suppose $x_i = (a_i, b_i, c_i)^T$ (i = 1,2, …, 8). Choosing the first two equations from (3.22) for the corresponding 8 points, 16 equations are computed as shown below

$$\begin{cases} a_1 \left(p_1^{3^T} X\right) - p_1^{1^T} X = 0 \\ b_1 \left(p_1^{3^T} X\right) - p_1^{2^T} X = 0 \\ \cdots \\ a_8 \left(p_8^{3^T} X\right) - p_8^{1^T} X = 0 \\ b_8 \left(p_8^{3^T} X\right) - p_8^{2^T} X = 0 \end{cases} \tag{3.23}$$

Where p_j^i represent the ith column of P_j (i =1,2,…,8). This can also be expressed in the form of

$$AX = 0 \tag{3.24}$$

Where A is a 16 x 4 matrix represented as

$$A = \begin{bmatrix} a_1 p_1^{3^T} - p_1^{1^T} \\ b_1 p_1^{3^T} - p_1^{2^T} \\ \cdots \\ a_8 p_8^{3^T} - p_8^{1^T} \\ b_8 p_8^{3^T} - p_8^{2^T} \end{bmatrix} \tag{3.25}$$

To solve for the 3D coordinates of X, the SVD of A is computed as

$$A = U \sum V^T \tag{3.26}$$

If the singular values in Σ are arranged in descending order, the solution for X is the last column of V. All the triangulation procedure above assumes no noise in estimated 2D points. If there are noise occurs, according to work done by (Hartley and Sturm 1994), a constrained MSE-based triangulation method is used to find the corresponding coordinates of 2D feature points. The correct matching feature points are localized by minimizing the Euclidean distance algorithm

$$[d(x_1,x'_1)]^2 + [d(x_2,x'_2)]^2 \tag{3.27}$$

with the epipolar constraint

$$x'^T_2 F x'_1 = 0 \tag{3.28}$$

Any pair of the corresponding points must lie on a pair of corresponding epipolar lines l_1 and l_2 in two views and any pair of matching points lying on these two lines will satisfy the epipolar constrains. The optimal 2D points x'_1 and x'_2, which are the points that are closest to the original matching points, will lie on a pair of epipolar lines l_1 and l_2 respectively. So, the equation in (3.27) can be represented using the distance of the noisy points to the epipolar lines:

$$[d(x_1,l'_1)]^2 + [d(x_2,l'_2)]^2 \tag{3.29}$$

Where the perpendicular distance from point x_i to line l' (i = 1.2) are represented by the term $d(x_i, l'_i)$. As mentioned before, correct matching 2D points x'_1 and x'_2 lie on these two epipolar lines and can be found by representing the epipolar line l'_1in the first image by a parameter t become $l'_1(t)$, With the fundamental matrix F, the other epipolar line l'_2 is related to l'_1. Thus, equation (3.29) can be represented as polynomial function of t. Another parameter t_{min} is introduced to minimize the polynomial function. By evaluating the distance function at each of the real roots, the corrected 2D points x'_1 and x'_2 on these lines at t_{min} can be calculated. The corrected 2D points x'_1 and x'_2 are used to compute the corresponding 3D point X using SVD. At the end of the triangulation, the projective 3D structure is reconstructed.

3.3.2 Implementation of Triangulation method

By using the common matching 2D feature points x and projection matrices P for all views, the 3D feature points can be calculated through triangulation. Triangulation function is performed using 'vgg_X from xP_lin' by making the assumption that no noise occurs in the system. The inputs of the system include 2 matrices. One input matrix stores the matching points coordinates in each view corresponding to the same 3D point. The second matrix is a 3D matrix storing the projection matrices *P* of all views. This function uses SVD method to calculate the 3D points as describe in section 3.3.1. The toolbox for triangulation is provided by Tomas Werver from the University of Oxford (Triangulation toolbox).

3.4 Bundle Adjustment

3.4.1 Overview of Bundle Adjustment algorithm

After obtaining the 3D points and projection matrices from all views, refinement of the 3D structure is needed through a global minimization step. This is because both the 3D points and the projection matrices which are derived from fundamental matrices might be affected by noise. Bundle adjustment algorithm introduced by (Lourakis and Argyros 2009) was used to solve this problem. The main objective of using the algorithm was to find the projective structures *P* of multiple views and the 3D points *X* so that the mean square distances between the observed 2D image feature points *x* and reprojected 2D points *P*(*X*) are minimized. Figure 3.4 illustrate the bundle adjustment.

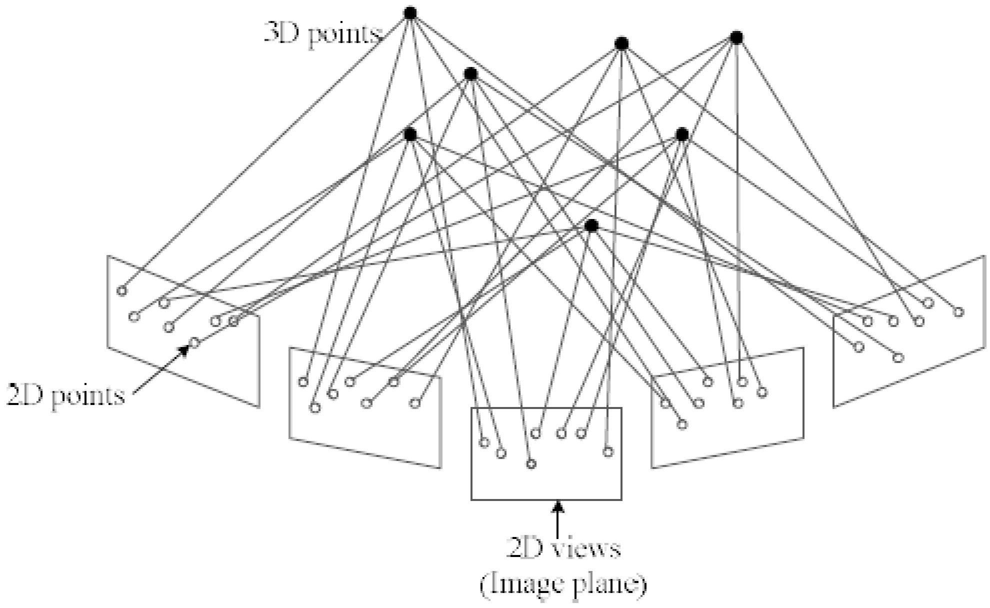

Figure 3.4: Illustration of bundle adjustment (R. I. Hartley 2003)

The rays back-projected from the corresponding 2D feature points in different views to a single 3D point, constitute a bundle. Through bundle adjustment, given *m* number of views, a new set of projection matrix P_i^{BA} (i = 1,…,*m*) and 3D space points X_j^{BA} (j = 1,…,*n*) will be calculated so that reprojection 2D points

$$x_{ij}^{BA} = P_i^{BA} X_j^{BA} \tag{3.30}$$

become stable. The reprojected 2D points x_{ij}^{BA} need to minimize the following Euclidean distances from the initial 2D feature points x_{ij} :

$$\min_{P_i X_j} \sum_{i=1}^{m} \sum_{j=1}^{n} d(x_{ij}, x_{ij}^{BA}) \tag{3.31}$$

The typical method of sparse bundle adjustment uses Levenberg-Marquardt (LM) algorithm (D. Marquardt 1963) to do the non-linear minimization of the reprojection error. The LM algorithm is an iterative procedure that calculates the minimum of non-linear least square problem. According to (D. Marquardt 1963), in order to use the LM algorithm for bundle adjustment, the initial measurement vector *w* consists of observed common 2D feature points in all views and by the 3D points, and the functional relation *f* can be calculated using the projection relationship between corresponding 2D and 3D points. *w* can be represented as

$$w = (x_{11}^T, \dots, x_{1m}^T, x_{21}^T, \dots, x_{2m}^T, \dots, x_{n1}^T, \dots, x_{nm}^T)^T \tag{3.32}$$

where *m* is the number of views, *n* is the number of common feature points in each view, x_{ij} is the i^{th} 2D point in the j^{th} view. The measurement vector *v* can be represent as

$$v = (p_1^T, \dots, p_m^T, X_1^T, \dots, X_n^T)^T \tag{3.33}$$

where the p_k is the unwrapped vector representation of the projection matrix corresponding to the k^{th} view, and X_s is the s^{th} 3D point.

3.4.2 Implementation of Bundle Adjustment algorithm

The function ‘findtriangulationLM’ is used to implement the bundle adjustment for the refinement of the projective reconstruction so that the reprojection distance function (3.31) is minimized using Levenberg-Marquardt algorithm. The 2D feature points in all views, the projection matrices of all views and the 3D points estimated from triangulation are used as the input for this function. The output of the ‘findtriangulationLM’ consists of the refined projection matrices of all views and the coordinates of the 3D points. The projection matrix of the reference view was fixed so that the coordinate of the reference camera will remain the same as the world coordinates. If the average deviation is larger than a threshold, the reconstructed 3D scene is not accurate enough and contains a lot of noise. So, the program needs to go back to RANSAC and implement the triangulation and bundle adjustment again. The toolbox is called Vincent toolbox provides by Vincent Rabaud from UCSD (Bundle Adjustment toolbox).

3.5 Building 3Dmap

After the 3D points was being computed, all the points are being transform onto a 3D plane to construct a 3D model. All the points and matrices were being saved in a .3dc file format. The 3D model built is then being view in another open source program called OpenSceneGraph. It is an open source 3D graphical application programming interface used by many developers in fields like visual simulation, computer games, virtual reality, and scientific visualization and modelling.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview of the Experiment conducted in the system

Based on the system that has been implemented, some experiments were being carried out in order to test the performance of the system. The author has come out with the list of the area of focus on the performance of the system. The objectives of conducting all these experiments were as listed below:

1. To evaluate the effectiveness of the system when constructing the 3D model
2. To evaluate the similarity of the constructed model compared to the desired model
3. To evaluate the robustness of the system react to the feature points obtained from the author's partner

In order to achieve the objectives above, there are total of 5 experiments that have been carried out:

1. **Experiment 1**: Effect of the using different texture of the object on the computed 3D model
2. **Experiment 2**: Effect of the number of frame used on the computed 3D model
3. **Experiment 3**: Effect of the number of point detected on the computed 3D model

4. **Experiment 4**: Effect of number of RANSAC iteration on the computed 3D model
5. **Experiment 5**: Effect of the number of plane obtained on the computed 3D model

For each of the experiment carried out, the author was evaluating different type of parameters that might affect the robustness of the 3D reconstruction system. All the results obtained were valid results from the system. For further details on the programming code can refer to the appendix behind.

4.2 Experiments on the system

4.2.1 Experiment 1: Different Texture

Objectives:

- To evaluate the effect of using different object with different type of texture on the computed 3D model
- To observe the similarity of the reconstructed model compared to the expected model

Setup and procedure:

1. There are total of 3 different data that is being evaluated. The three sets of images chosen for evaluation are shown in Figure 4.1, 4.2 and 4.3
2. The number of frame is set to 4, and all the other parameter is constant, the 3D model constructed is being captured and being evaluated.

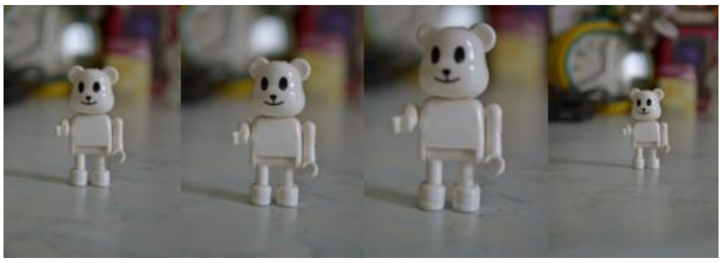
Figure 4.1: Image data set1

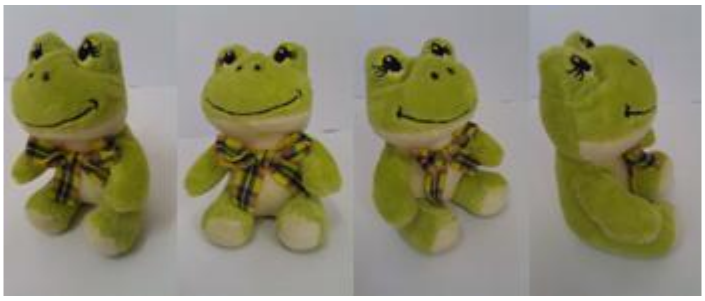
Figure 4.2: Image data set2

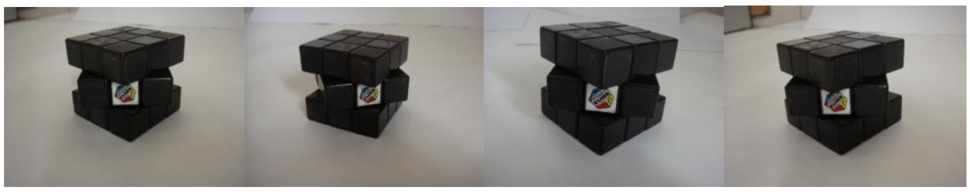
Figure 4.3: Image data set3

Results:

After computing the experiment, the reconstructed 3D models for each of the different object were shown in Figure 4.4, 4.5 and 4.6.

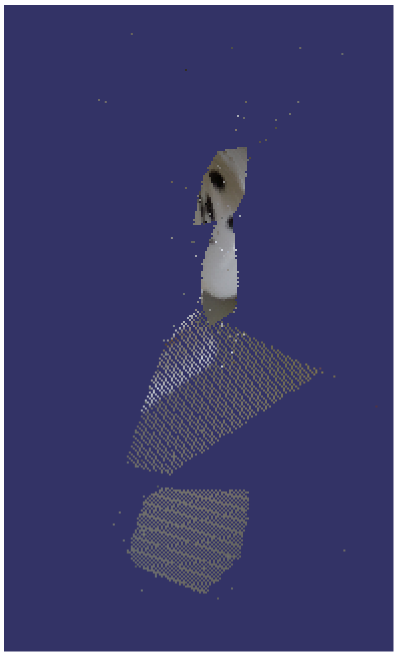
Figure 4.4: 3D model for data set1

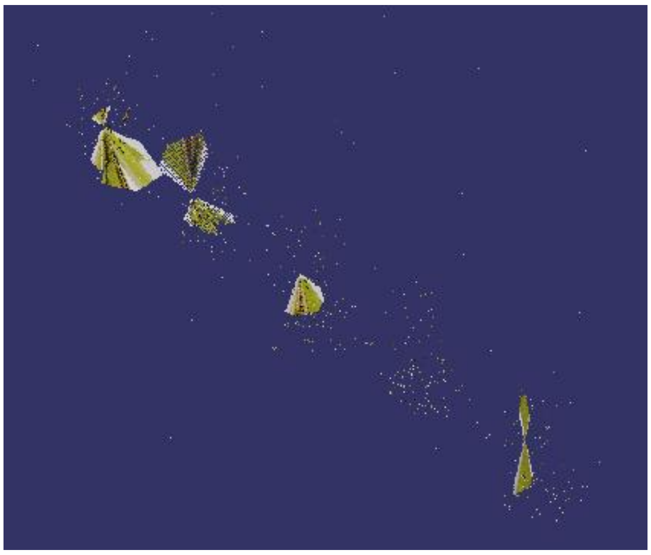
Figure 4.5: 3D model for data set2

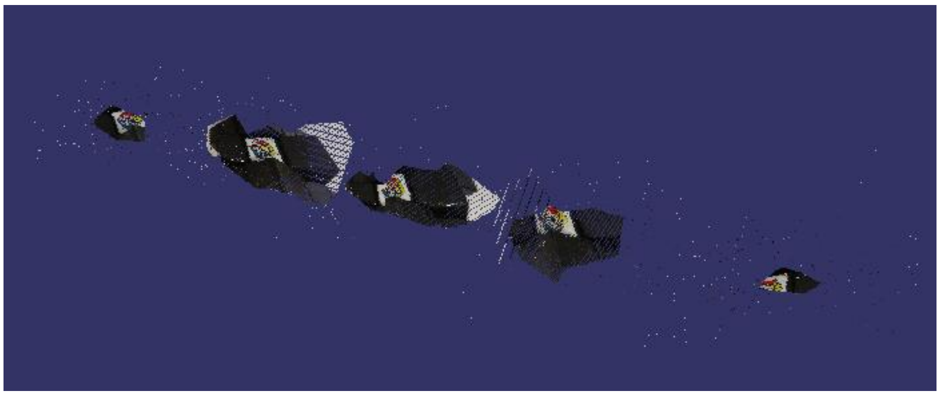
Figure 4.6: 3D model for data set3

Table 4.1: Parameters involved in the 3D model reconstruction

Image set	**Set 1**	**Set 2**	**Set3**
Type of texture	smooth	rough	geometry
Total time used (s)	36.25	294.76	137.35
Number of planes	100	100	100
RANSAC iteration	200	200	200
Distance from camera	1	1	1
Maximum distance	0.5	0.5	0.5

Discussion:

Based on the 3D model constructed above, we can see that all three reconstructed model gives only portion of the original expected model. By comparing the three different texture models, we can see that object with rough surface, which is the data set2 gives the worse result compare to the two objects. The object with geometry shape gives a clearer result. The reconstruction 3D object was much closer to the expected shape in data set3. And for the data set1, the texture of the model is not really visible compared to data set3 but it at least gives a rough 3D structure on the object compared to data set2. Besides, by comparing the processing time for the three different texture data, we can see that the object with rough surface takes the longest time for the processing followed by the geometrical object and the smooth surface. This is because a object with rough surface have more feature points detected, so, it needs longer time for processing. On the contrary, the object with smooth surface will give the least point detection, which explains why the processing time is the fastest.

Conclusion:

Based on the results above, it shows that object with rough surface give the worse reconstructing results whereas the object with the geometry type surface will gives the best results in the 3D reconstruction. So, different type of the surface of the object will affect the similarity of the 3D reconstruction.

4.2.2 Experiment 2: Number of frames

Objectives:

- To evaluate the effect of using changing the number of frame used to build the 3D model
- To observe the similarity of the reconstructed model compared to the expected model

Setup and procedure:

1. Two sets of data are being prepared. The two data sets are shown in Figure 4.7 and 4.8.
2. For data set1, the number of frames used is set to 4 frames whereas for data set2, the number of frames used is set to 8frames.
3. Other parameters remained constant.
4. The built 3D model is being created and evaluated

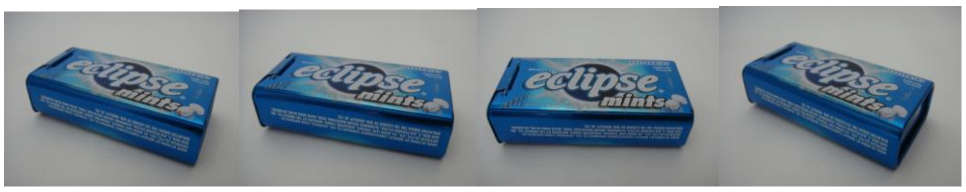
Figure 4.7: Image data set1

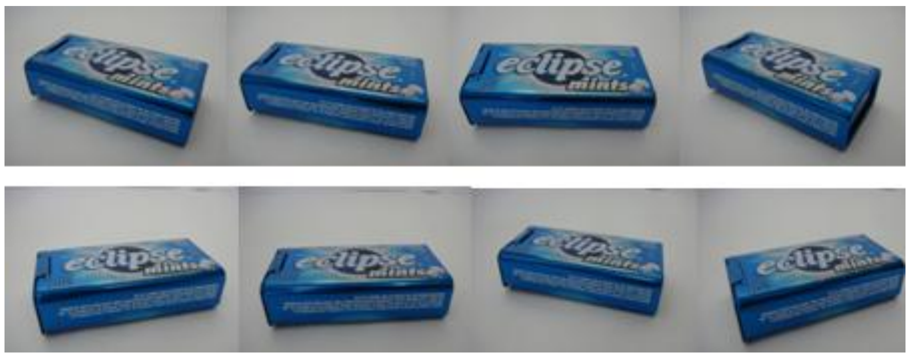
Figure 4.8: Image data set2

Results:

After computing the experiment, the reconstructed 3D models for each of the case were shown in Figure 4.9 and 4.10.

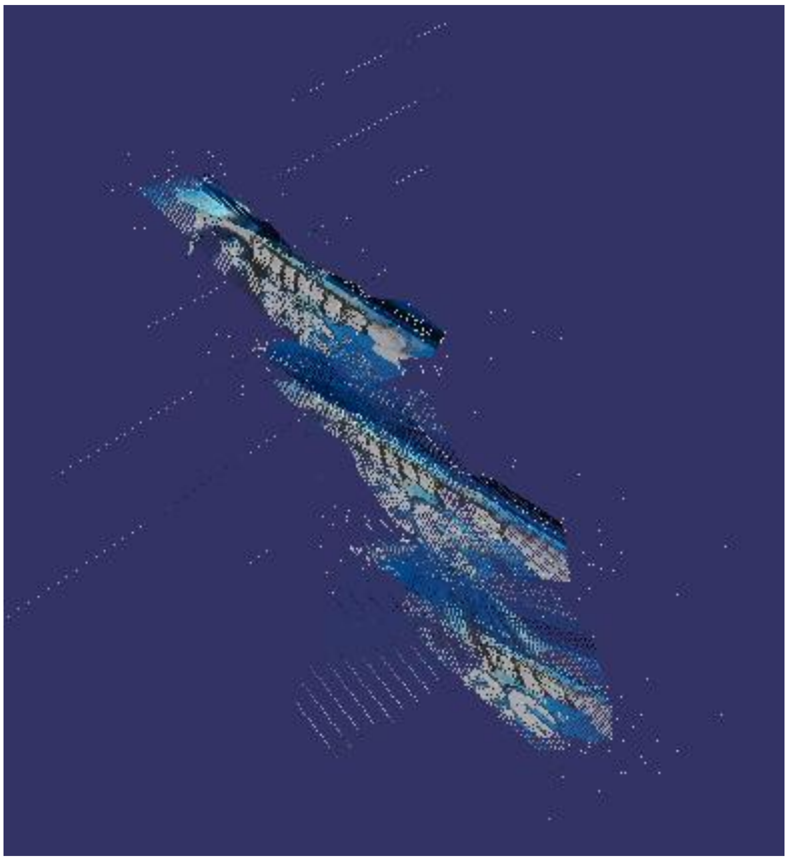

Figure 4.9: 3D model for data se1

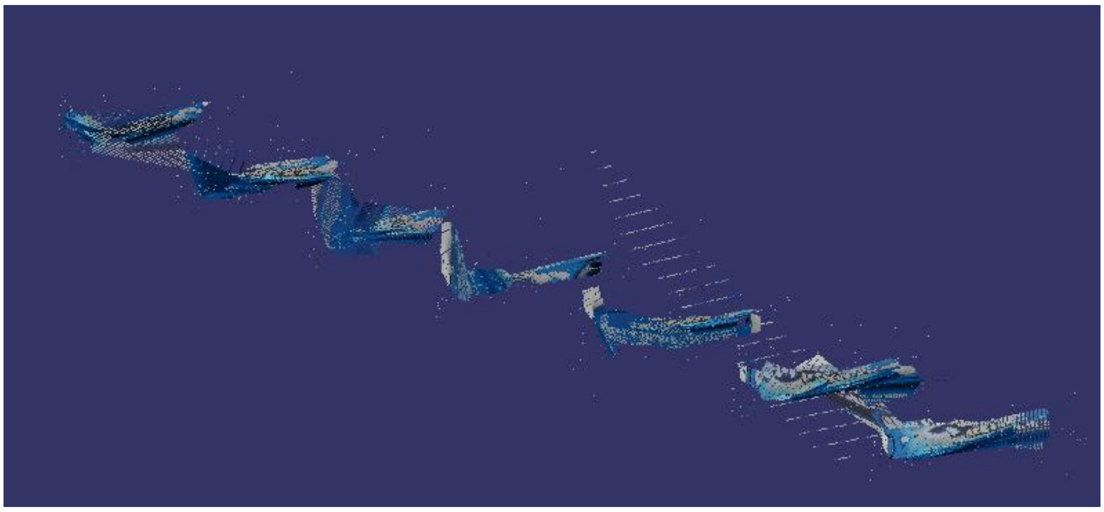

Figure 4.10: 3D model for data set2

Table 4.2: Parameters involved in the 3D model reconstruction

Image set	**Set 1**	**Set 2**
Number of frames	4	8
Total time used (s)	158.97	357.13
Number of planes	100	100
RANSAC iteration	200	200
Distance from camera	2	2
Maximum distance	0.75	0.75
PointCloud	1919	5104

Discussion:

From Experiment1, the author and her partner have made a conclusion that object with geometry shape gives better results compared to the smooth and rough surface object for 3D reconstruction. So, in the second experiment, the author decided to take a rectangular object for evaluation. Refer to the 3D model constructed above, we notice that for data set1, the constructed image have 3section whereas for data set2, the constructed image have 7 section. The number of section shown in the results represents the number of pairs taken. For each pairs chosen, matching, RANSAC, triangulation followed by 3D model reconstruction is being done for better reprojection matrices. Based on the results we notice that with the increasing of the number of frames, the reconstructed 3D models are relatively more detailed. More point clouds are being plotted with the increasing number of frame. The more point cloud plotted, the better 3D model the system can produced and thus, the similarity of the reconstructed 3D model compared to the expected model increased. By referring to the processing time, it is obvious that the time taken to process the data with more frames will be longer if compared to the case for smaller number of frames. However, in the ideal case, the expected results from the system are supposed to be showing only one frame in the system. Instead of that, the results obtained are having multiple frames in the final 3D model. This might due to the mismatch between the point clouds of each plane during the triangulation steps. Refer to equation (3.27) as well as (3.29) these are the Euclidean distance algorithm used for correct matching feature. Minimizing the distance between the feature points might help to solve this problem.

Conclusion:

In conclusion, in order to get a better 3D reconstruction model, using more frames are the better choice. This is because from the results obtained, data set2 with 8 frames gives a better 3D model compared to data set1 with 4 frames.

4.2.3 Experiment 3: Number of feature points

Objectives:

- To evaluate the effect of using changing the number of feature points detected to build the 3D model
- To observe the similarity of the reconstructed model compared to the expected model

Setup and procedure:

1. Refer to Figure 4.11, one set of image data set is being prepared. The number of frames to be observed is set to 6.
2. Using the same data set, the number of feature points obtained is being altered by controlling the threshold of the SIFT detection and matching.
3. Other parameters remained constant.
4. The built 3D model is being created and evaluated

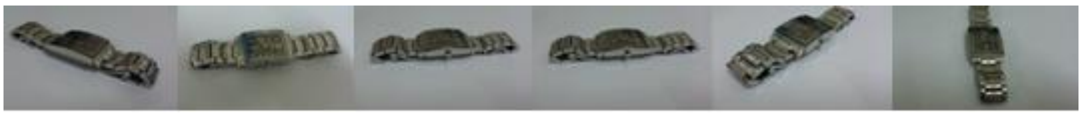
Figure 4.11: Image of data set of interest

Results:

After computing the experiment, the reconstructed 3D models for each of the case were shown in Figure 4.12 and 4.13.

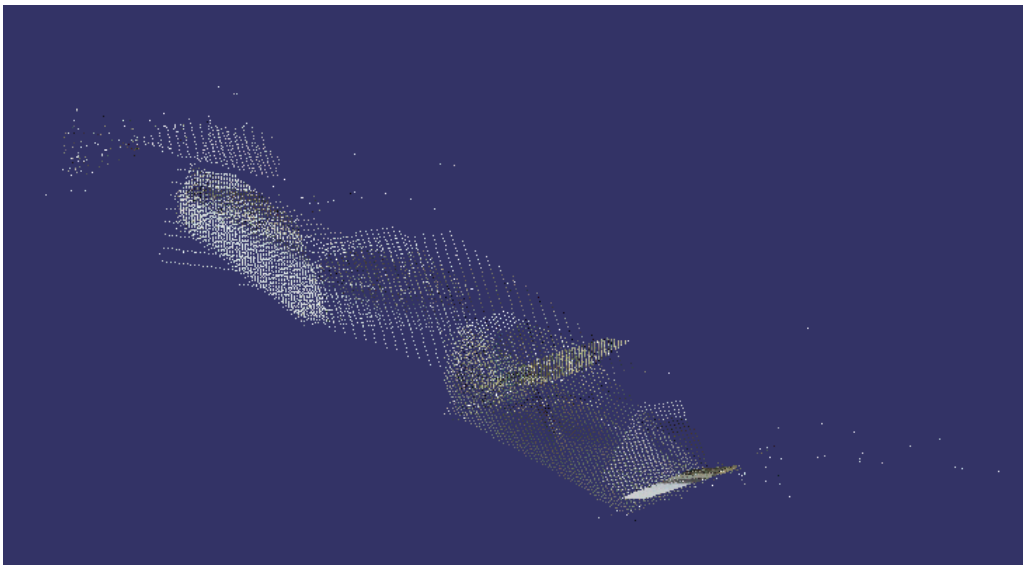

Figure 4.12: 3D model 1 with smaller number of feature points

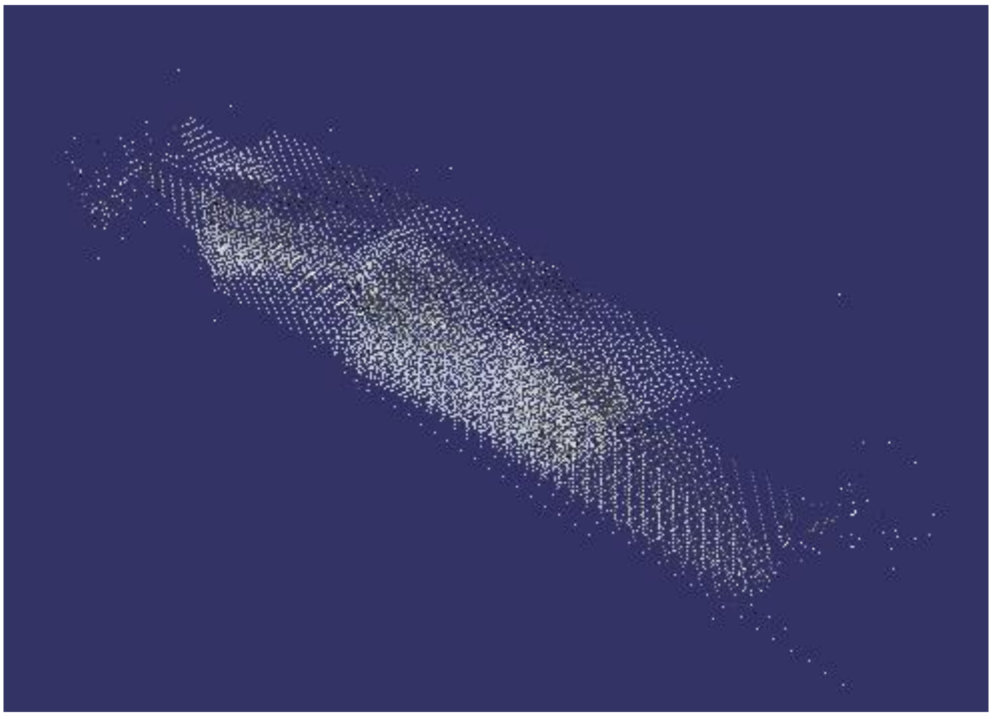

Figure 4.13: 3D model 2 with larger number of feature points

Table 4.3: Parameters involved in the 3D model reconstruction

Case	**1**	**2**
Number of frames	6	6
Total time used (s)	1472.91	3189.53
Number of planes	100	100
RANSAC iteration	200	200
Distance from camera	10	2
Maximum distance	2	0.75
Feature points	588	1255

Discussion:

In the third experiment, the author is trying to compare the difference in the results if we are using different amount of feature points. In case 1, the number of feature points is 588 whereas for case 2, the number of feature points is 1255. Here, the object of interest in our experiment is a watch. Refer to the results obtained, we can clearly see that the 3D model constructed in case 1 have less concentrated points compared to the situation in case 2. The reconstruction model of the watch in case 1 is significantly not very obvious if compared to case 2. So, we can see that case 2 will have higher percentage of similarity compared to the expected image. In this case, we also noticed that both the results obtained do not carried the same colour texture as what we expected. Supposing the model constructed should have a metallic texture but from our results, we get is a white colour model of the object. This might due to the mismatch between the retrieval of the colour pixels when performing the matching algorithm.

Conclusion:

In conclusion, the higher number of the feature points taken will gives a better results in 3D model reconstruction.

4.2.4 Experiment 4: RANSAC iteration

Objectives:

- To evaluate the effect of variation in RANSAC iteration on the 3D object reconstruction
- To observe the similarity of the reconstructed model compared to the expected model

Setup and procedure:

1. Refer to Figure 4.14, a set of images on a building is being used for evaluation. Total of 8 frames of the building is being captured.
2. Using the same data set, the number of RANSAC iteration is being altered by controlling the threshold of the RANSAC inliers and outliers parameters.
3. For case 1, the number of iteration is set to 150 and for case 2, the number of iteration is set to 200
4. Other parameters remained constant.
5. The built 3D model is being created and evaluated

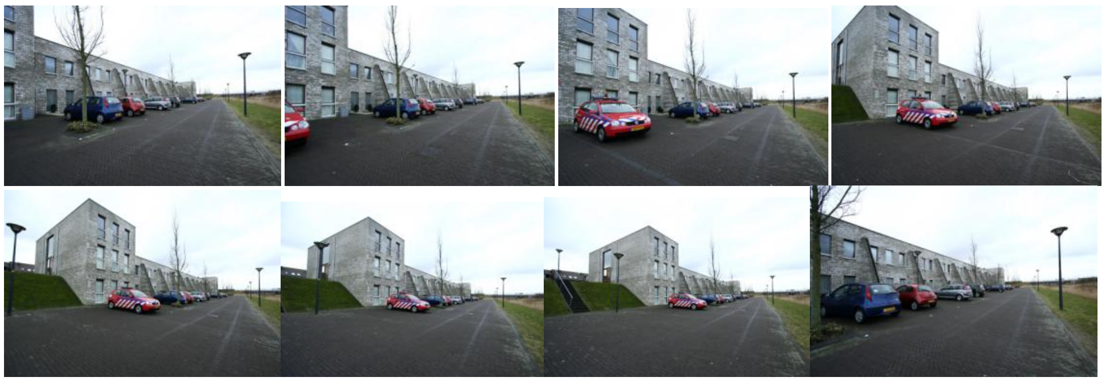

Figure 4.14: Image of data set of interest

Results:

After computing the experiment, the reconstructed 3D models for each of the case were shown in Figure 4.15 and 4.16.

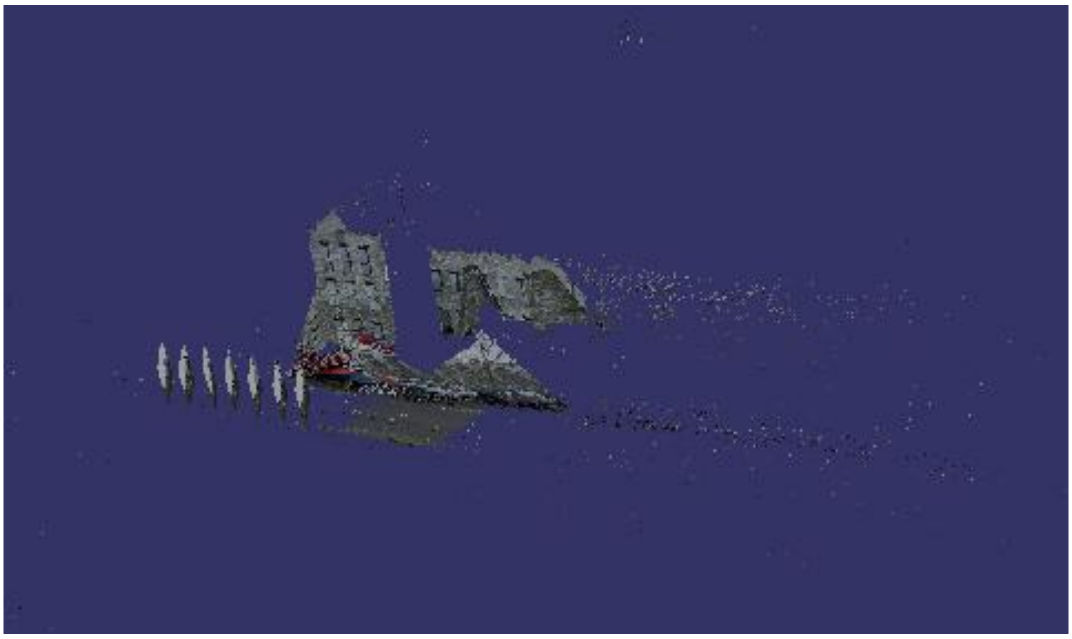

Figure 4.15: 3D model 1 with 150 iteration

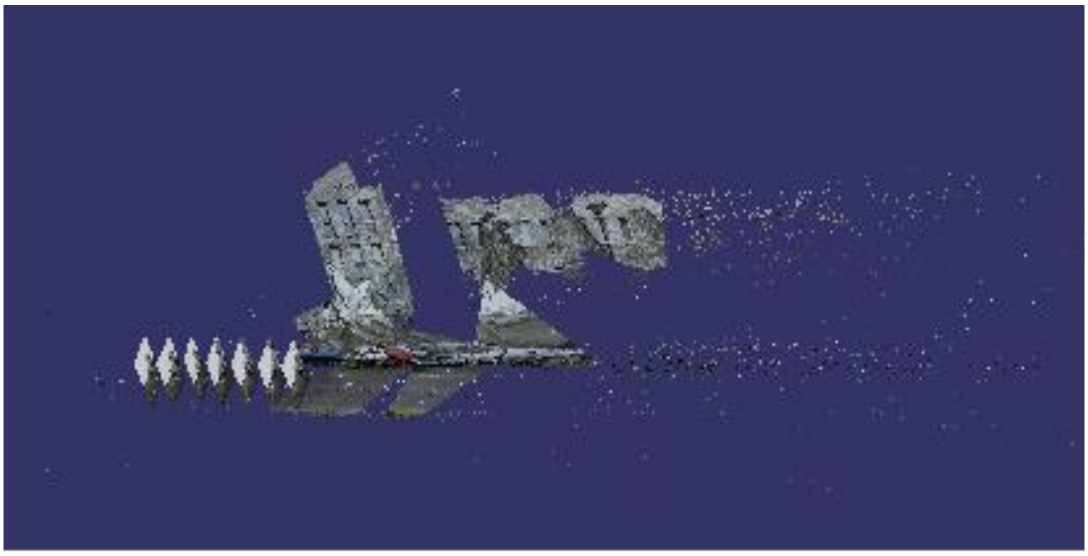

Figure 4.16: 3D model 2 with 200 iteration

Table 4.4: Parameters involved in the 3D model reconstruction

Case	**1**	**2**
Number of frames	8	8
Total time used (s)	771.421	965.22
Number of planes	100	100
RANSAC iteration	150	250
Distance from camera	30	30
Maximum distance	10	10
MAP points	10597	13646

Discussion:

In the fourth experiment, the author is trying to compare the difference in the results if we are altering the number of RANSAC iteration when performing the RANSAC algorithm to eliminate the inliers. In case 1, the number of MAP points is 10597 whereas for case 2, the number of MAP points is 13646. Here, the object of interest in our experiment is a building. The picture of building is obtained from the internet source (FIT3D.com). Refer to the results obtained, we can clearly see that the 3D model constructed in case 1 have less MAP points compared to the situation in case 2. Although for both case, the 3D building model constructed have significantly higher percentage of similarity compared to the results obtain in the previous experiment, there are difference between the two cases. The reconstruction model of the building in case 1 is less accurate compared to the model created in case 2. As we can see from the results obtained, Figure 4.16 has plotted more points and it thus created better 3D model. For the case 2, there are less reprojection error compared to case 1 as the increasing number of iteration eliminate the probability of the mismatch in the system. However, there are some drawbacks to obtain a better 3D model. By comparing the number of time needed to compute the 3D model, case 2 takes longer processing time is compared to case 1 with less iteration. In order to obtain better results in the 3D reconstruction, scarification of the time for computation is needed.

Conclusion:

In conclusion, the higher number of the RANSAC iteration give higher accuracy of the results. Case 2 give the higher similarity of 3D model compared to case 1.

4.2.5 Experiment 5: Number of plane

Objectives:

- To evaluate the effect of variation in number of plane on the 3D object reconstruction
- To observe the similarity of the reconstructed model compared to the expected model

Setup and procedure:

1. Refer to Figure 4.17, a set of images on a building is being used for evaluation. Total of 8 frames of the building is being captured.
2. Using the same data set, the number of plotting image plane is being altered to evaluate the differences between the results.
3. For case 1, the number of plane is set to 200 and for case 2, the number of plane is set to 500
4. Other parameters remained constant.
5. The built 3D model is being created and evaluated

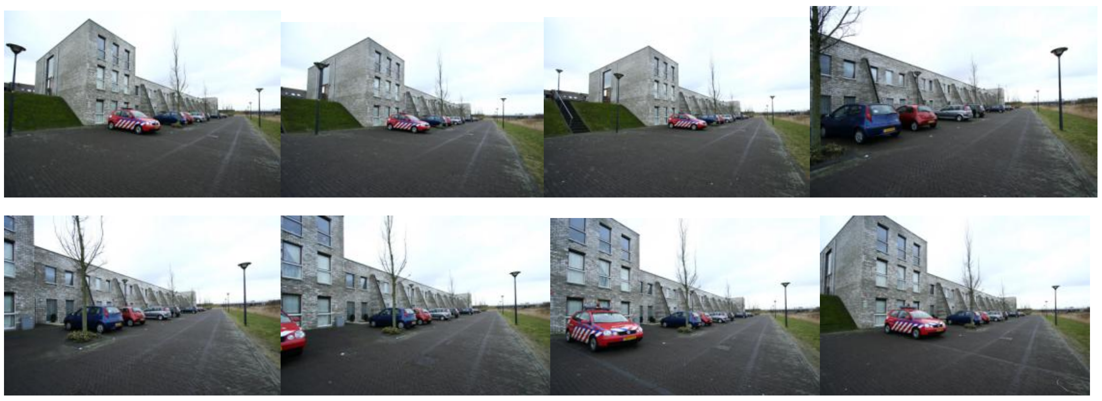

Figure 4.17: Image of data set of interest

Results:

After computing the experiment, the reconstructed 3D models for each of the case were shown in Figure 4.18 and 4.19.

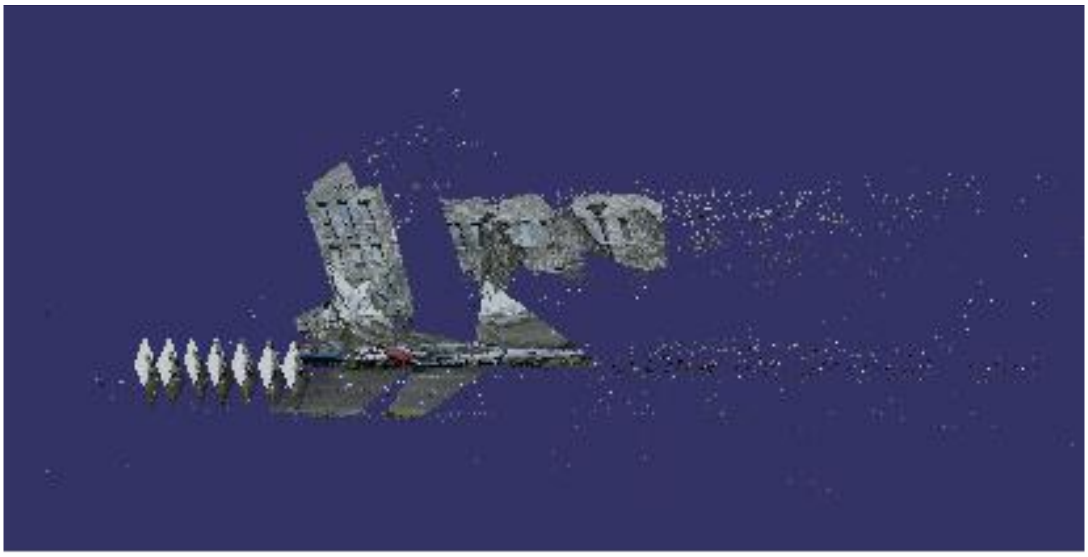

Figure 4.18: 3D model 1 with number of plane = 200

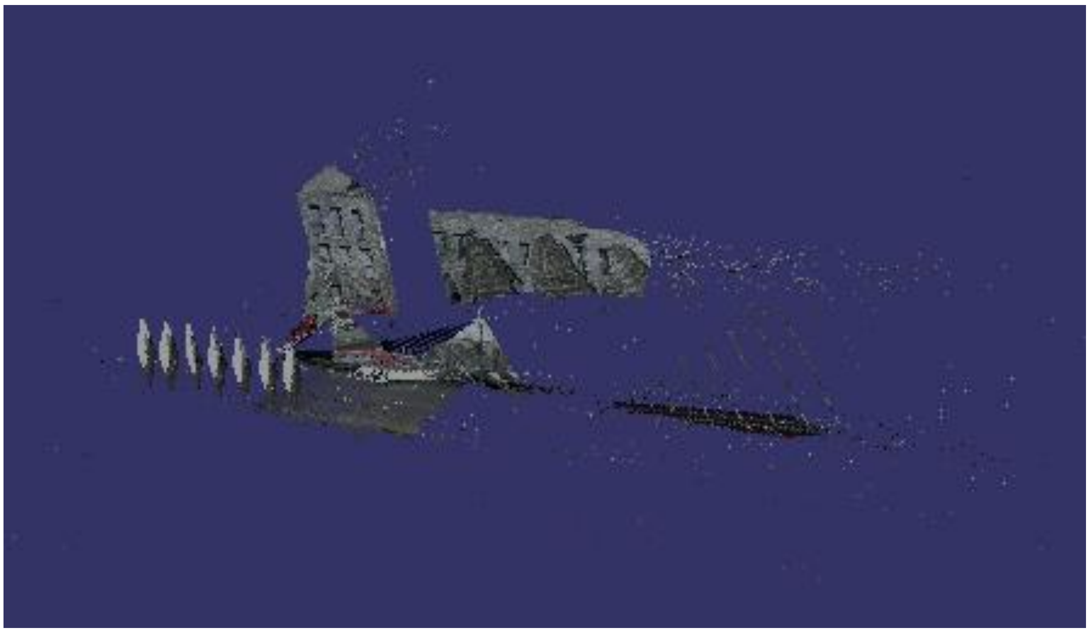

Figure 4.19: 3D model 2 with number of plane = 500

Table 4.5: Parameters involved in the 3D model reconstruction

Case	**1**	**2**
Number of frames	8	8
Total time used (s)	923.76	2559.44
Number of planes	200	500
RANSAC iteration	150	150
Distance from camera	30	30
Maximum distance	10	10
MAP points	14402	22376

Discussion:

In the fifth experiment, the author is trying to compare the difference in the results if we are altering the number of plane needed to plotting and estimating the point cloud in the 3D reconstruction process. In case 1, the number of MAP points is 14402 whereas for case 2, the number of MAP points is 22376. Here, the author used back the image data set in experiment4, which are the pictures of building obtained from the internet source (FIT3D.com). Refer to the results obtained, we can clearly see that the 3D model constructed in case 1 with fewer number of plane create less detailed, less concentrated points in the constructed model. Thus, we can say that the reconstruction model of the building in case 1 has lower similarity with the expected 3D building model compared to the model created in case 2. As we can see from the results obtained, Figure 4.19 has plotted more points due to the additional plane created for estimating the reprojection matrices and it thus created better 3D model. However, same as the problem in experiment 4 above, there are some drawbacks to obtain a better 3D model. By comparing the number of time needed to compute the 3D model, case 2 takes longer processing time is compared to case 1 with fewer number of planes. The amount of time needed for computing the 3D model with 500 number of plane is almost triple the time of computing a 3D model with number of plane equals to 200. In order to obtain better results in the 3D reconstruction, scarification of the time for computation is needed.

Conclusion:

In conclusion, the higher number of plane needed to compute a 3D model give higher accuracy of the results. Case 2 give the higher similarity of 3D model compared to case 1. However, the drawback is that longer time is needed for computation.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This report implements a 3D modelling system that utilizes multiple views of a scene to reconstruct the 3D model from the given feature points obtained from the author's partner. This works contributes to the field of 3D imaging and 2D to 3D video conversion. In this chapter, the author summarizes the contribution of the thesis in section 6.2 and also discuss about the future works in section 6.3.

5.2 Contributions

In this thesis, the objectives of this project were being covered. A system to reconstruct a 3D modelling using multiple view geometry method is being created. Although this work does not really perform in a more robust way, the basic requirement for the 3D reconstruction is met. A further improvement on this works will be mentioned in the following section for a more robust system. Besides, the similarities between the constructed 3D models are made to be as close as possible to the expected model. The contribution of the thesis can be summarized as follows:

- Outlier removal and two-view geometry computation are implemented in this thesis. A robust algorithm called random sample consensus or widely known as RANSAC is used for the refinement of the feature matching between two views. The refinement is done by removing the outliers of the feature points. The geometry between two views is estimated using the inliers of feature points.

- Projective reconstruction of 3D scene is implemented in this thesis. Two-view geometry and correspondences of the feature points are used in conjunction with triangulation to reconstruct the projective structure of the 3D object. Besides, Bundle adjustment is included in the triangulation to refine the projective reconstruction of the 3D model.
- Different image sets with variety of four, six and eight multiple views are used to test the 2D to 3D conversion system. The result shows that the optimum number of views for reconstructing a model is eight. Objects with different texture will gives different results. From the results obtained above, object with a geometry shape and texture gives the best results. Altering the number of feature points, number of planes and the RANSAC iteration will also gives different results on the 3D model.

5.3 Future Work

There are quite a lot of issues left to be further studied and implemented in this thesis in order to make the 3D modelling system more robust and stable for various applications.

- The triangulation can be improved by implementing the method that corrects the positions of 2D points under the existence of noise.
- Ideally, this system is planned to rebuild the 3D model from pictures to ease the engineers in constructing 3D model of machine and parts. However, the time taken for the processing image as well as the accuracy and the detailed of the 3D model constructed is not so optimum. The implementation time for the system can be further reduced as well as increasing the accuracy and detailing of the 3D model is needed in order to make this system suitable for image processing and video sequences in real time.

- It is also possible to implement the 3D modelling based on a pre-designed model with enough prior information about the 3D scene. This is greatly reduced the complexity and computation of the system.
- The conversion between the 2D to 3D in this thesis can be further extended for video sequences instead of images. Multiple images will be capture from the given video sequences. This will involve a further study of object and camera motion estimation.
- The projective reconstruction of the 3D scene can be upgraded to a metric transformation through auto-calibration. The intrinsic camera parameters can be estimated using auto-calibration.
- Introduction to sparse depth map for feature points can be used to generate a dense depth map for all the pixels in the image. The techniques of image rectification and image segmentation need to be integrated in the system to generate dense depth map.

REFERENCES

S.D. Ma and L.Li. 1996 Ellipsoid Reconstruction from Three Perspective Views. In Proceedings International Conferences of Pattern Recognition.

C. Schmid and A. Zisserman. 1998 The Geometry and Matching of Curves in Multiple Views. In Proceedings European Conference on Computer Vision

R. I. Hartley and A. Zisserman. 2003 Multiple View Geometry, second ed. Cambridge University Press.

R. Berthilson K. Astrom and A. Heyden. February 1999 Reconstruction of curves in R^3, using Factorization and Bundle Adjustment. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(2)

J. Y. Kaminski and A. Shashua. 2000. On Calibration and Reconstruction from Planar Curves. In Proceedings European Conference on Computer Vision

K. Kanatani. 1996 Statistical Optimization for Geometric Computation: Theory and Practice. Elsevier Science

A. Criminisi. August 2001 Accurate Visual Metrology from Single and Multiple Uncalibrated Images. Springer-Verlag London Ltd.

Ebrahimnezhad1, H., a, H. Ghassemian & a (2008) Robust motion from space curves and 3D reconstruction from multiviews using perpendicular double stereo rigs. Image and Vision Computing, 26.

Heuel, S. & W. Forstner. 2001. Matching, reconstructing and grouping 3D lines from multiple views using uncertain projective geometry. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, II-517-II-524 vol.2.

Kaminski, J. Y., M. Fryers, A. Shashua & M. Teicher. 2001. Multiple view geometry of non-planar algebraic curves. In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, 181-186 vol.2.

M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications of the ACM*, vol. 24, No. 6, pp. 381–395, June 1981.

P. Torr and D. Murray, “The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix,” *International Journal of Computer Vision*, vol. 24, no. 3, pp. 271-300, September 1997.

R. I. Hartley, “Estimation of Relative Camera Positions for Uncalibrated Cameras,” *Proceedings of the European Conference on Computer Vision*, pp. 579-587, May 1992.

RANSAC code available online at:
http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/

R. I. Hartley and P. Sturm, “Triangulation,” *Proceedings of the ARPA Image Understanding Workshop*, pp. 957-966, November 1994.

Triangulation toolbox code available at:
http://www.robots.ox.ac.uk/~vgg/

M. I. A. Lourakis and A. A. Argyros, “SBA: A Software Package for Generic Sparse Bundle Adjustment,” *ACM Transactions on Mathematical Software*, vol. 36, no. 1, pp. 1-30, March 2009.

D. Marquardt, “An Algorithm for the Least-squares Estimation of Nonlinear Parameters,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431-441, June 1963.

Bundle Adjustment toolbox code available at:
http://vision.usdc.edu/~vrabaud/toolbox/doc/

Issac Esteban, FIT3D toolbox, IAS, University of Amsterdam, TNO Defense, Security and Safety, April 2010, http://www.fit3d.info/

OpenSceneGraph 3D graphics programming interface available at:
http://www.openscenegraph.org

M.A. Fishler and R.C. Boles. "Random sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography". Comm. Assoc. Comp, Mach., Vol 24, No 6, pp 381-395, 1981

APPENDICES

APPENDIX A: Computer Programming Listing

Build3Dmodel.m

```
%% SCRIPT TO OBTAIN 3D STRUCTURE
%clear workspace and command window
clear;
clc;

%% PRODUCE 3D MODEL USING RECONSTRUCTED STRUCTURE
% The images is plotted in space
% The images is approximated using plannar patches

%% LOAD DATA
% run the FIT_setup.mat file from fit3d toolbox

run('C:\Users\SunGoku\Documents\MATLAB\FIT3D\FIT3D_setup.m');

% Load camera motion data PcamScaled.mat
load(strcat(install_path,'/Tryout/reconstruction/.mat  file/original/PcamScaled.mat'));

% Load image features FplusExtra.mat
load(strcat(install_path,'/Tryout/reconstruction/.mat  file/original/FplusExtra.mat'));

% Load camera calibration matrix K.mat
load(strcat(install_path,'/Tryout/reconstruction/.mat file/original/K.mat'));

% Load Files structure for retrieving the images Files.mat
load(strcat(install_path,'/Tryout/reconstruction/.mat file/original/Files.mat'));

% Fix image location
Files.dir = strcat(install_path,Files.dir);

%% SETTING PARAMETERS FOR THE SYSTEM

PcamI = PcamScaled; % camera matrices
Km = Kcanon10GOOD; % camera calibration matrix
dist1 = 30; % distance from camera
plotFramese = true; % plot images in space
plotPoints = true; % plot point clouds
fitPlanes = true;
nplanes = 100; % number of planes
fitthreshold = 0.01; % plane threshold
density = 50;
points = 500; % closest N points
minpoints = 20; % minimum points
maxDist = 10; % maximum distance
planeransac = 200; % RANSAC iterations
startFrame = 1;

%% DETAILS ABOUT THE PARAMETERS INVOLVED

% FplusExtra   -> (nx9) n features x [frame1,frame2,X1,Y1,X2,Y2,R,G,B]
% Pcam         -> (3x4xk) camera matrices in relative frame for k camera poses
% Km           -> (3x3) camera calibration matrix
% Files        -> (structxk) struct containing the name of the image files for
%                            plotting in the point cloud
% dist         -> (1x1) show points within distance
% plotFrames   -> (true/false) Plot the images positioned in space
% plotPoints   -> (true/false) Plot the point clouds
% fitPlanes    -> (true/false) Use the plane fitting algorithm
% nplanes      -> (1x1) Number of iterations to run the plane
%                       fitting algorithm
% fitthreshold -> (1x1) Fitting threshold for the planes
% density      -> (1x1) Density of the displayed planes
% points       -> (1x1) Maximum number of points to select at every
%                       iterations
% minpoints    -> (1x1) Minimum number of points requiered for a
%                       plane
% planeransac  -> (1x1) Ransac iterations for plane fitting
% maxDist      -> (1x1) Maximum distance from points to mean
% startFrame   -> (1x1) Starting frame

%% WITH THE MATCHING FEATURE POINTS AND CAMERA MOTIONS, A 3D MODEL IS BUILT

% 3 GENERAL STEPS INVOLVED:
% 1. function 'plotFrames' created 3D map with image displayed in space
% 2. the 3D model is created containing point cloud.
% 3. function 'fitPlanes' will represent the point cloud with planar
%    patches
%% START THE 3D RECONSTRUCTION STEPS
% taking PcamScaled filed as the original camera matrix

tic; % counting the total time used for 3D reconstruction

%1. Normalize the camera matrix from 3x4x8 into 4x4x8
%2. Inverse the matrix Pcam

Pcam = invertMotion(normalizePcam(PcamI));

    % Number of frames (number of frame is either 4,6 or 8 in our system)
    frames = size(Pcam,3);

%3. Absolute Pcam in 3x4x8 matrix
    [PcamABS] = getTrajectory3DNorm(Pcam);

%4. making b into 3x3x8 matrix (value follow the 1st 3 row and 3 column of PcamABS)
    b = PcamABS(:,1:3,:);

%5. centers is the 8x3 matrix (value for each row is the last colum value
%   from PcamABS)
    centers = reshape(PcamABS(:,4,:),3,size(PcamABS,3))';

%6. Normalized the camera matricex and invert back the Pcam and save in PcamInv
PcamInv = normalizePcam(Pcam);

    for(i=1:size(Pcam,3))
        PcamInv(:,:,i) = inv(PcamInv(:,:,i));
    end;

%7. Making the Pcam from 4x4x8 back into 3x4x8
    PcamInv = PcamInv(1:3,:,:);

%8. Getting PcamABSInv in 3x4x8 matrix
    [PcamABSInv] = getTrajectory3DNorm(PcamInv);

%9. Creating blank matrix for this three variables
    MAP = [];
    POINTCLOUD = [];
    POINTCLOUDidx = [];
    MAPc = [];

%10. determine the matrix coeffiecient
    for(i=startFrame:frames-1)

        if(size(Km,3)==1)
          K = Km;
        else
          K = Km(:,:,i);
        end;

%11. Find all matches between frames
        % find out the matches between frame i and i+1 and put it in nx2 matrix
        P = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,3:4);
        % on the 3rd column of the coding, add in '1' for every row
        P = [P,ones(size(P,1),1)];

        % find out the matches between frame i and i+1 and put it in nx2 matrix
        Q = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,5:6);

        % on the 3rd column of the coding, add in '1' for every row
        Q = [Q,ones(size(Q,1),1)];

        % Get image indexes
        indxAllFeatures = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,1);

        % Get RGB values for each point of the matched value
        R = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,7);
        G = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,8);
        B = FplusExtra(FplusExtra(:,1)==i & FplusExtra(:,2)==i+1,9);

        % Triangulate points
        x3d = findTriangulationLM(P,Q,[eye(3),[0;0;0]],PcamInv(:,:,i+1),K,K)';

        % Find points which distance to the camera center is below the
        % threshold value

        d = sqrt(x3d(:,1).^2+x3d(:,2).^2+x3d(:,3).^2);
        x3d = x3d(d<dist1,:);
        indxFeatures = indxAllFeatures(d<dist1);

        R = R(d<dist1,:);
        G = G(d<dist1,:);
        B = B(d<dist1,:);

        % Put the points in absolute coordinates
        for(f=1:size(x3d,1))
            x3d(f,1:3) = (PcamABS(:,1:3,i)*x3d(f,1:3)'+PcamABS(:,4,i))';
        end;

         % Create localMAP
        MAPlocal =
[x3d(:,1:3),floor(R*255),floor(G*255),floor(B*255),ones(size(x3d,1),3)];

        POINTCLOUD = [POINTCLOUD;MAPlocal];
        POINTCLOUDidx = [POINTCLOUDidx;indxFeatures];
    end;

%12. Plot images
    MAPframes = [];
    if(plotFrames)

        for(i=1:frames-1)

            PcamInv = inv([PcamABS(:,:,i);0,0,0,1]);
            PcamInv = PcamInv(1:3,:);

            if(size(Km,3)==1)
                K = Km;
            else
                K = Km(:,:,i);
            end;

            % Reading the image
            strcat(Files.dir,Files.files(i).name)
            img = imread(strcat(Files.dir,Files.files(i).name));

            % subsample rate
            subsample = 6;

            % Prepare image map
            imgMAP =
zeros(length(1:subsample:size(img,2))*length(1:subsample:size(img,1)),9);
            counter = 1;

            % Get 3D points of the image
            for m=1:subsample:size(img,2)
                for n=1:subsample:size(img,1)

                    % Put in camera center coordinates
                    aa = inv(K)*[m;n;1];
                    % Move to the camera 1 reference frame
                    aa = PcamABS(:,1:3,i)*aa+PcamABS(:,4,i);
                    if(size(img,3)==1)
                        R = floor(img(n,m));
                        G = R;
                        B = R;
                    else
                        R = floor(img(n,m,1));
                        G = floor(img(n,m,2));
                        B = floor(img(n,m,3));
                    end;
                    imgMAP(counter,:)  = [aa',double(R),double(G),double(B),ones(1,3)];
                    counter = counter+1;
                end;
            end;
            size(imgMAP);
            MAPframes = [MAPframes;imgMAP];
        end;
    end;

    %% FIT ALL THE n PLANES TO THE WHOLE POINT CLOUD

    foundPlanes = 0;
    MAPplanes = [];

    if(fitPlanes)

        % Get all points
        totalX3D = POINTCLOUD(:,1:3);

        x3dToFit = totalX3D;

        % Get number of closest points
        npts = points;

        minpointsLocal = minpoints;

        subsetx3dToFit = x3dToFit;

        % Obtain as many planes as requested
        for(np=1:nplanes)

            % display the size(x3dToFit)
            fprintf('Number of Iteration: %d\n',np);
            fprintf('Size: %d\n',size(x3dToFit,1));
            fprintf('Number of points: %d\n',npts);


            % If the points obtain is enough
            if(size(x3dToFit,1)>minpointsLocal)

                % Choose a random point
                n = getNRandom(1,size(x3dToFit,1));

                % Get the closest points
                closest = 
ipdm(x3dToFit,x3dToFit(n,:),'Subset','smallestfew','limit',min(npts,size(x3dToFit,1))
);

                % Get those points
                subsetx3dToFit = x3dToFit(full(closest)~=0 & full(closest)<maxDist,:);
                subsetIdx = POINTCLOUDidx(full(closest)~=0 & full(closest)<maxDist,:);

                % Eliminate poitns which are too far away from the pointcloud
                % centroid
                centroid = mean(subsetx3dToFit);

                % calculate distances and mean distance
                D = ipdm(subsetx3dToFit,centroid);
                meanD = median(D);

                % Eliminate points that are larger than mean
                subsetx3dToFit = subsetx3dToFit(D<meanD,:);
                subsetIdx = subsetIdx(D<meanD,:);

                % If the set size is enough
                if(length(subsetx3dToFit)>minpointsLocal)

                    % Fit plane
                    [B,P,inliers, warning] =
ransacfitplane(subsetx3dToFit',fitthreshold,0,minpoints,planeransac);

                    if(length(inliers)>minpointsLocal)
                        foundPlanes = foundPlanes+1;
                        fprintf('A Plane has been found, determining texture\n');

                        % Select the picture
                        indx = floor(max(subsetIdx(inliers,:)));

                        % Find proyection for texture
                        [MAPplane,MAPplaneColor] =
makePlanePlot3dcHull(B,density,subsetx3dToFit(inliers,1:3),Files,[floor(median(subset
Idx(inliers,:))),floor(median(subsetIdx(inliers,:)))],PcamABS,K,centers);

                        % Add map
                        MAPplanes = [MAPplanes;MAPplane];
                        MAPc = [MAPc;MAPplaneColor];

                        % Remove inliers
                        [x3dToFit,I] =
setdiff(x3dToFit(:,:),subsetx3dToFit(inliers,:),'rows');
                        POINTCLOUDidx = POINTCLOUDidx(I,:);

                    end;

                end;

            end;

        end;
    end;


    if(plotPoints)

        if(size(MAP,1)==1)
            MAP = POINTCLOUD;
        else
            MAP = [MAP;POINTCLOUD];
        end;
    end;
    fprintf('Total time used %.3f s\n', toc) ;
    %% WRITING 3DC FILE

    fprintf('Writing 3DC file');

    if(size(POINTCLOUD,1)>0)
        dlmwrite('3d.3dc',POINTCLOUD,' ');
    end;
    if(size(MAPplanes,1)>0)
        dlmwrite('3dPlane.3dc',MAPplanes,' ');
    end;
    if(size(MAPc,1)>0)
        dlmwrite('3dPlaneC.3dc',MAPc,' ');
    end;
    if(size(MAPframes,1)>0)
        dlmwrite('3dFrames.3dc',MAPframes,' ');
    end;
```

findTriangulationLM.m

```
%% FIND TRIANGULATION LM
% With the fundamental matrix ans a set of correcpoding points, the 3D points
% where the rays intercept are found
% This can be done by minimizing a cost functionas described by Zisserman
% in p.314 using the LM algorithm.
%
% Input  - X1   -> (nx3) set of homogeneous points in first image
%        - X2   -> (nx3) set of homogeneous points in second image
%        - P1   -> (3x4) First camera matrix
%        - P2   -> (3x4) Second camera matrix
%        - K1   -> (3x3) First camera calibration
%        - K2   -> (3x3) Second camera calibration
%
% Output - X3D  -> (3xn) A matrix of triangulated 3D points

%% FUNCTION

function [X3D] = findTriangulationLM(X1,X2,P1,P2,K1,K2)

    % Get only the interested image points
    x1 = X1(:,1:2);
    x2 = X2(:,1:2);


    % Perform triangulation on the points
    for i=1:size(x1,1)

        % Get 3D image points
        xphat = K1\X1(i,1:3)';
        xqhat = K2\X2(i,1:3)';

        % Build matrix A for first image
        A = [xphat(1)*P1(3,:)-P1(1,:);
             xphat(2)*P1(3,:)-P1(2,:);
             xqhat(1)*P2(3,:)-P2(1,:);
             xqhat(2)*P2(3,:)-P2(2,:)];

        % Normalize matrix A
        A1n = sqrt(sum(A(1,:).*A(1,:)));
        A2n = sqrt(sum(A(2,:).*A(2,:)));
        A3n = sqrt(sum(A(3,:).*A(3,:)));
        A4n = sqrt(sum(A(4,:).*A(4,:)));
        Anorm = [A(1,:)/norm(A(1,:));
                 A(2,:)/norm(A(2,:));
                 A(3,:)/norm(A(3,:));
                 A(4,:)/norm(A(4,:))];

        % Get the 3D point
        [Uan,San,Van] = svd(Anorm);
        X3D(:,i) = Van(:,end);

        % Normalize 3D point to ensure the points not in infinity
        X3D(:,i) = X3D(:,i)./X3D(4,i);

    end;
```

getNRandom.m

```
%% GET RANDOM POINTS (INTEGERS)
% Get n different random numbers out of a set
%
% Input  - n    -> (1x1) Number of random numbers
%        - K    -> (1x1) Maximum number
%
% Output - R    -> (1xn) List of different random numbers
%% FUNCTION

function R = getNRandom(n,K)

    % Get all posible numbers
    N = randperm(K);

    % Get n of them
    R = N(1:n)';
```

getTrajectory3DNorm.m

```
%% GET TRAJECTORY 3D NORMINAL
% To get the absolute trajectory of a camera given the set of relative
% camera poses.
% It is done by normalizing the camera matrices and chain
% multiplication.
% Input  - Pcam         -> (3x4xn) Relative camera matrices for n poses
% Output - PcamABS      -> (3x4xn) Absolute camera matrices for n poses

%% FUNCTION

function [PcamABS] = getTrajectory3DNorm(Pcam)

    PcamABS = zeros(4,4,size(Pcam,3));

    % Normalize the camera matrices
    PcamN = normalizePcam(Pcam);

    PcamABS(:,:,1) = PcamN(:,:,1);

    % Accumulate the camera matrices to get the absolute value
    for(i=2:size(Pcam,3))

        PcamABS(:,:,i) = PcamABS(:,:,i-1)*PcamN(:,:,i);

    end;

    PcamABS = PcamABS(1:3,:,:);
```

invertMotion.m

```
%% INVERT MOTION
% Invert the camera motion
% Either from camera 1 to camera 2 or camera 2 to camera 1
% Input  - Pcam    -> (4x4xn) Camera motion (normalized)
% Output - PcamInv -> (4x4xn) Inverted camera motion (normalized)

%% FUNCTION

function PcamInv = invertMotion(Pcam)

PcamInv = Pcam;
for(i=1:size(Pcam,3))
    PcamInv(:,:,i) = inv(PcamInv(:,:,i));
end;
```

normalizePcam.m

```
%% NORMALIZED CAMERA MATRICES
% Normalizes a 3x4 camera matrices to 4x4
% Input  - Pcam      -> (3x4xn) n camera matrices
% Output - PcamN     -> (4x4xn7) n normalized camera matrices
%% FUNCTION

function [PcamN] = normalizePcam(Pcam)


    PcamN = Pcam;

    for i=1:size(Pcam,3)

        PcamN(4,:,i) = [0,0,0,1]; % make the matrices into 4 x 4 matrix

    end;
```

randomsample.m

```
%% RANDOM SAMPLE
% Selects n random items from an array
%
% Inputs:    a - Either an array of values from which the items are to
%                be selected, or an integer in which case the items
%                are values selected from the array [1:a]
%            n - The number of items to be selected.
%
%Output:     item - a array of ramdom permutation od the intergers 1:n
%% FUNCTION

function item = randomsample(a, n)

    npts = length(a);

    if npts == 1   % scalar argument for a
    npts = a;
    a = [1:a]; % Construct an array 1:a
    end

    if npts < n
    error(...
    sprintf('Trying to select %d items from a list of length %d',n, npts));
    end

    item = zeros(1,n);

    for i = 1:n
    % Generate random value in the appropriate range
    r = ceil((npts-i+1).*rand);
    item(i) = a(r);       % Select the rth element from the list
    a(r)    = a(end-i+1); % Overwrite selected element
    end
```

ransac.m

```
%% RANSAC
% Fits model to data robustly with RANSAC algorithm
% Input:
%     x        -> Data sets to fit a model M
%                 Assumed that x is of size [d x Npts]
%                 where d is the dimensionality of the data
%                 Npts is the number of data points.
%
%     fittingfn-> Handle to a function that fits a model to s
%                 data from x.
%                    M = fittingfn(x)
%
%     distfn   -> Handle to a function that evaluates the
%                 distances from the model to data x.
%                    [inliers, M] = distfn(M, x, t)
%                 This function must evaluate the distances between points
%                 and the model returning the indices of elements in x that
%                 are inliers.
%                 The points that are within distance't' of the model.
%                 Additionally, if M is a cell array of possible models
%                 'distfn' will return the model that has the most inliers.
%                 After this, M will be a non-cell object representing only
%                 one model.
%
%     degenfn  -> Handle to a function that determines whether a
%                 set of datapoints will produce a degenerate model.
%                 This is used to discard random samples that do not
%                 result in useful models.
%                    r = degenfn(x)
%
%     s        -> The minimum number of samples from x required by
%                 fittingfn to fit a model.
%
%     t        -> The distance threshold between a data point and the model
%                 used to decide whether the point is an inlier or not.
%
%     feedback -> An optional flag 0/1 Defaults = 0.
%
%     maxDataTrials -> Maximum number of attempts to select a non-degenerate
%                      data set. Defaults = 100.
%
%     maxTrials -> Maximum number of iterations.
%                  Defaults = 1000.
%
% Output:
%     M         -> The model having the greatest number of inliers.
%     inliers   -> An array of indices of the elements of x that were
%                  the inliers for the best model.

%% FUNCTION

function [M, inliers, war] = ransac(x, fittingfn, distfn, degenfn, s, t, feedback, ...
                               maxDataTrials, maxTrials)



    % Test number of parameters
    error ( nargchk ( 6, 9, nargin ) );
    error ( nargoutchk ( 3, 3, nargout ) );

    if nargin < 9; maxTrials = 1000;    end;
    if nargin < 8; maxDataTrials = 100; end;
    if nargin < 7; feedback = 0;        end;

    [rows, npts] = size(x);

    p = 0.99;         % Optimum probability of choosing at least one sample
                      % free from outliers

    bestM = NaN;      % Sentinel value allowing detection of solution failure.
    trialcount = 0;
    bestscore =  0;
    N = 1;            % Initialisation for number of trials.
```

```
war = false;

while N > trialcount

    % Select at random s datapoints to form a trial model, M.
    % Make sure these points are not in a degenerate configuration.
    degenerate = 1;
    count = 1;
    while degenerate
        % Generate s random indicies in the range 1 until npts
        ind = randsample(npts, s);

        % Test that these points are not a degenerate configuration.
        degenerate = feval(degenfn, x(:,ind));

        if ~degenerate
            % Fit model to this random selection of data points.
            % M may represent a set of models that fit the data
            % M will be a cell array of models in this case
            M = feval(fittingfn, x(:,ind));

            if isempty(M)
                degenerate = 1;
            end
        end

        % Safeguard against being stuck in this loop forever
        count = count + 1;
        if count > maxDataTrials
            warning('Unable to select a nondegenerate data set');
            break
        end
    end

    %% EVALUATE DISTANCE BETWEEN POINTS

    % Model returning the indices of elements in x that are inliers.
    % Additionally, if M is a cell array of possible models 'distfn'
    % will return the model that has the most inliers.
    % After this call M will be a non-cell object representing 1 model

    [inliers, M] = feval(distfn, M, x, t);

    % Find the number of inliers to this model.
    ninliers = length(inliers);

    if ninliers > bestscore    % latest largest set of inliers
        bestscore = ninliers;  % Record data for this model
        bestinliers = inliers;
        bestM = M;

        % Update estimate of N, the number of trials to ensure the data
        % we choose with probability p is a data set with no outliers

        fracinliers =  ninliers/npts;
        pNoOutliers = 1 -  fracinliers^s;
        pNoOutliers = max(eps, pNoOutliers);  % Avoid division by -Inf
        pNoOutliers = min(1-eps, pNoOutliers);% Avoid division by 0.
        N = log(1-p)/log(pNoOutliers);
    end

    trialcount = trialcount+1;
    if feedback
        fprintf('trial %d out of %d         \r',trialcount, ceil(N));
    end

    % Safeguard against being stuck in this loop forever
    if trialcount > maxTrials
        warning( ...
        sprintf('ransac reached the maximum number of %d trials',...
                maxTrials));
        war = true;
        break
    end
end
fprintf('\n');
```

```
    if ~isnan(bestM)   % A solution is found
        M = bestM;
        inliers = bestinliers;
    else
        M = [];
        inliers = [];
    end
```

ransacfitplane.m

```
%% RANSAC FIT PLANE
% Using RANSAC method to fits plane to 3D array of points
% Uses RANSAC algorithm to robustly fit a plane to a set of 3D data points
% to eliminate the outliers
%
% Input - XYZ      -> 3xNpts array of xyz coordinates to fit plane
%       - t        -> The distance threshold between data point and the plane
%                     used to decide whether a point is an inlier or not.
%       - feedback -> Optional flag 0 or 1 to turn on RANSAC feedback
%                     information.
%
% Output - B       -> 4x1 array of plane coefficients in the form
%                     b(1)*X + b(2)*Y +b(3)*Z + b(4) = 0
%                  -> The magnitude of B is 1.
%                  -> This plane is obtained by a least squares fit to all the
%                     points that were considered to be inliers, hence this
%                     plane will be slightly different to that defined by P below.
%        - P       -> The three points in the data set that were found to
%                     define a plane having the most number of inliers.
%                  -> The three columns of P defining the three points.
%        - inliers -> The indices of the points that were considered
%                     inliers to the fitted plane.
%
%% FUNCTION

function [B, P, inliers, warning] = ransacfitplane(XYZ, t, feedback,dataIt,ransacIt)

    if nargin == 2
    feedback = 0;
    end

    [rows, npts] = size(XYZ);

    if rows ~=3
        error('data is not 3D');
    end

    if npts < 3
        error('too few points to fit plane');
    end

    s = 3;  % Minimum number of points needed to fit a plane.

    fittingfn = @defineplane;
    distfn    = @planeptdist;
    degenfn   = @isdegenerate;

    %performing RANSAC
    [P, inliers, warning] = ransac(XYZ, fittingfn, distfn, degenfn, s, t,
feedback,dataIt,ransacIt);

    % Perform least squares fit to the inlying points
    if(length(inliers>3))
        B = fitplane(XYZ(:,inliers));
    else
        B = 0;
    end;
%% DEFINE A PLANE WITH GIVEN DATA POINTS
% Define a plane given 3 data points as required by RANSAC
% In this case,the 3 points were used directly to define the plane.

function P = defineplane(X);
    P = X;

%% CALCULATE DISTANCE BETWEEN A PLANE AND ARRAY OF POINTS
% The plane is defined by a 3x3 matrix, P.  The three columns of P defining
% three points that are within the plane.

function [inliers, P] = planeptdist(P, X, t)

    % Find mean of points
    m = mean(X,2);

    n = cross(P(:,2)-P(:,1), P(:,3)-P(:,1));  % Plane normal.
    n = n/norm(n);                             % Make it a unit vector.

    npts = length(X);
    d = zeros(npts,1);
    dm = zeros(npts,1);
    for i = 1:npts
        d(i) = abs(dot(X(:,i)-P(:,1), n));    % Distance
        dm(i) = abs(dot(X(:,i)-m, n));
    end

    tm = (max(dm)-mean(dm))/2;

    % Locate nd all points which distance to plane is below threshold and
    % distance to the mean of the points is small

    inliers = find(abs(d) < t);


%% DETERMINE A SET OF POINTS ARE IN DEGENERATE CONFIGURATION OR NOT
% A function to determine whether a set of 3 points are in a degenerate
% configuration for fitting a plane as required by RANSAC.
% If they are colinear, they are degenerate.

function r = isdegenerate(X)

    % The three columns of X are the coordinates of the 3 points.
    r = iscolinear(X(:,1),X(:,2),X(:,3));
```