3D Modeling Using Multi-View Images

by

Jinjin Li

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved August 2010 by the
Graduate Supervisory Committee:

Lina J. Karam, Chair
Chaitali Chakrabarti
Tolga M. Duman

ARIZONA STATE UNIVERSITY

December 2010

ABSTRACT

There is a growing interest in the creation of three-dimensional (3D) images and videos due to the growing demand for 3D visual media in commercial markets. A possible solution to produce 3D media files is to convert existing 2D images and videos to 3D. The 2D to 3D conversion methods that estimate the depth map from 2D scenes for 3D reconstruction present an efficient approach to save on the cost of the coding, transmission and storage of 3D visual media in practical applications. Various 2D to 3D conversion methods based on depth maps have been developed using existing image and video processing techniques. The depth maps can be estimated either from a single 2D view or from multiple 2D views.

This thesis presents a MATLAB-based 2D to 3D conversion system from multiple views based on the computation of a sparse depth map. The 2D to 3D conversion system is able to deal with the multiple views obtained from uncalibrated hand-held cameras without knowledge of the prior camera parameters or scene geometry. The implemented system consists of techniques for image feature detection and registration, two-view geometry estimation, projective 3D scene reconstruction and metric upgrade to reconstruct the 3D structures by means of a metric transformation. The implemented 2D to 3D conversion system is tested using different multi-view image sets. The obtained experimental results of reconstructed sparse depth maps of feature points in 3D scenes provide relative depth information of the objects. Sample ground-truth depth data points are used to calculate a scale factor in order to estimate the true depth by scaling the obtained relative depth information using the estimated scale factor. It was found out that the obtained reconstructed depth map is consistent with the ground-truth depth data.

ACKNOWLEDGEMENTS

I would like to acknowledge the patience and strong support of my thesis advisor Dr. Lina J. Karam in enabling me to complete my thesis. I would like to thank my committee members Dr. Chaitali Chakrabarti and Dr. Tolga M. Duman for agreeing to be on my committee. I would also like to acknowledge the support provided to me by the members of the Image, Video and Usability (IVU) lab. I would like to acknowledge the valuable assistance provided to me by Srenivas Varadarajan, Qian Xu, Milind Gide, Adithya Vishnu Murthy and Nabil Sadaka. I would like to thank my parents and my family for their strong love and contribution.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This chapter presents the motivations behind the work in this thesis and briefly summarizes the contributions and organization of the thesis.

1.1 Motivation

In recent years, with the giant leap in image and video processing technologies, the introduction of three-dimensional televisions (3D TVs) [1] into the commercial market is becoming a reality. Nowadays, there are many commercial companies, such as Samsung, Sony, Panasonic and LG, producing 3D TVs. The 3D TVs can be more attractive to viewers because they produce stereo scenes, which create a sense of physical real space. 3D vision for humans is caused by the fact that the projected points of the same point in space on the two human eyes are located at different distances from the center of focus (center of fovea). The difference between the distances of the two projected points, one on each eye, is called disparity. Disparity information is processed by high levels of the human brain to produce a feeling of the distance of objects in 3D space. A 3D television employs some techniques of 3D presentation, such as stereoscopic capture, 3D display and 2D plus depth map technologies.

Due to the success of introducing 3D visual technologies, including 3D games and 3D TVs, to the commercial market, the demand for a wide variety of 3D content such as 3D images, 3D videos and 3D games is increasing significantly. To satisfy this demand, there is an increasing need to create new 3D video content as well as converting existing 2D videos to 3D format. Converting 2D content into 3D depends on different 2D to 3D conversion tools.

Among various kinds of 2D to 3D conversion systems, the method that converts the 2D content to 3D by generating a depth map is a popular one as this method is efficient in the coding, transmission and storage of the 3D content. Using multi-view 2D images to estimate the 3D depth information is a typical method in this category. Multiple views of the same scene provide enough information of the 2D scene, and the mathematical computer vision methods can be used to estimate the 3D structure. The multi-view based 2D to 3D conversion borrows concepts from various applications including image registration, feature tracking, object localization and structure reconstruction. From the multi-view image sequences, the 2D to 3D conversion system performs the object matching between different views in order to estimate the depth map of the 2D scene. Another advantage of the multi-view 2D to 3D conversion method is that the parameters of the camera can be estimated. In addition, the multi-view 2D to 3D conversion method is applicable to various acquisition methods of images, especially for image sources captured using hand-held and uncalibrated cameras. It requires less prior information about the camera and the scene for the 3D reconstruction as compared to other 3D reconstruction models.

1.2 Contributions

In this thesis, a 2D to 3D image conversion system is presented. The implemented 3D modeling system is able to process image sets that are obtained using an uncalibrated camera, without knowing the information about the ground-truth of the scene and the camera settings. The techniques involved in the implemented 2D to 3D conversion system consist of feature extracting and tracking, image registration, three dimensional geometry estimation and refinement, camera calibration and scene structure reconstruction. The 2D to 3D conversion system in this thesis uses the scale invariant feature transform (SIFT) to extract the features of objects and register the feature points in different views. The Random Sample Consensus (RANSAC) is implemented to remove the outliers in the correspondences, so that the inliers for the corresponding feature points and two-view geometries can be estimated. Triangulation and bundle adjustment are employed later to estimate and refine the projective reconstruction of the 3D scene. Finally, an auto-calibration technique is used to upgrade the projective reconstruction of structures to the metric coordinates. Through these combined techniques, the relative depth information is estimated for feature points among multiple views of the scene. Different multi-view image sets are used to test the 2D to 3D conversion system, and experimental results are presented and analyzed. The 3D sparse depth map produced by the 3D modeling system is compared with ground-truth data, and it is shown how a scale factor can be estimated to recover the ground-truth depth.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 introduces the existing methods for 2D to 3D conversion. Chapter 3 presents the background material that is related to the work in this thesis. Chapter 4 describes the main components of the implemented 2D to 3D conversion system, and implementation details are also discussed. Chapter 5 presents the experimental results of the 2D to 3D conversion system based on different multi-view image sets. Chapter 6 summarizes the contributions of this thesis and proposes future directions of research.

2. RELATED WORK

This chapter summarizes the existing work that is related to 2D to 3D conversion techniques. Section 2.1 describes several major methods that make use of a single image for depth estimation. Section 2.2 summarizes methods that use multiple views to reconstruct the 3D scene.

Many methods were proposed for 2D to 3D conversion in recent years. One such method is to estimate the depth image from monocular videos or images, and then the original 2D images or videos and the computed depth images are used to obtain the 3D content, through a process known as depth image based rendering (DIBR) [2]. Methods that produce stereo image pairs are presented in [3] [4]. Advantages of these methods, which generate directly stereo image pairs rather than 2D images and their corresponding depth maps (2D+depth), are that they are suitable for many display equipments as they produce an output that can be readily displayed for 3D viewing humans. The shortcoming of the methods of [3] [4] is that they have a lot of constraints on the camera motion and image sets, limiting their uses in practical implementations. Compared to the methods in [3] and [4], by estimating the depth map for 2D to 3D conversion, since the depth map can be highly compressed, the 2D+depth method can save a large portion of the transmission bandwidth. The advantages of using depth maps for 2D to 3D conversion are discussed in [5]. These advantages are among the reasons why the 2D+depth method became a popular direction in current 2D to 3D conversion researches.

2.1 Depth Map from a Single Image

Some proposed 2D to 3D conversion methods are based on a single image to estimate the depth map. Several typical methods are summarized as below.

Studies of depth values obtained from focus cues are done in [6] and [7]. In [6], using the relationship between the image blur and the focus degree of edge pixels, a relative pixel-resolution depth map can be estimated. In the first step, a macroblock depth map is calculated by dividing the image into macroblocks (16×16) and computing the wavelet transform of each macroblock. Then, a 256-level depth map is created by thresholding the local spatial frequencies in a macroblock. This depth map reflects the spatial frequency content in each macroblock. The second step is to create a pixel-based depth map by estimating the defocus degree of the edge pixels, based on the macroblock depth map. The edge pixels are detected using a multi-scale wavelet transform by finding the local maxima of the scaled wavelet spectrum. To differentiate the type of edge pixels, the Lipschitz regularity [8] of an edge is computed using the decay of wavelet transform coefficients from a coarser to a finer scale in the neighborhood of edge pixels. The edge pixels with a Lipschitz regularity between 0 and 1 are defocused edges while the edge pixels with Lipschitz regularity between -1 and 0 are focused ones. Through combining the edge pixels and blur degree represented by the Lipschitz regularity in the rows of an image, the sections between the nearby two edges can be categorized to different image structures, thus assigned with the depth values according to the depth values in the macroblocks to which the edge pixels belong.

Due to blocking and stripe artifacts being introduced by the method of [6], the work done in [7] provides several new techniques to enhance the depth map. To get the initial depth map, the method of [7] is implemented using overlapping windows of size 16×16 to analyze the frequency energy at each point. The resulting depth map has less block artifacts and is more smoothed. In detecting the edge pixels, a 2D Gaussian function is used as the smoothing function and among the maxima points, those less than a given threshold are discarded to reduce the noise effect. For the detected edge pixels, the discontinuities in edges are corrected by searching possible edge points in the neighborhoods of every edge pixels. Additionally, to correct the depth values for the focused foreground objects with uniform color and texture, color-based segmentation are used to modify the depth values.

The image structure is used for the depth estimation in [9]. In [9], the image structure is described using two amplitude spectrum descriptions, the global spectral signature and the local spectral signature. The global spectral signature of an image is the mean magnitude of the global Fourier transform over a set of known images. The local spectral signature is the mean amplitude of the local macroblock wavelet transform over a set of known images. The spectral signatures reflect the general structures shared by the image set. The global spectral signature reveals the dominant orientations and textural patterns and is strongly related to the spatial structure of the scene for real-world images. A local spectral signature which is computed using Gabor filters, gives information about the dominant orientations and scales in a local region in the image and their mean spatial distribution. Torralba and Oliva in [9] have illustrated that changes in mean depth of a scene not only affects the slope of the global magnitude spectrum but also changes the local orientation and scales. In [9], the first step of depth estimation is to separate the man-made structure and natural structure in the considered image by observing the difference in energy distribution across the spatial frequencies of an image. To estimate the mean depth of the image scene, the mean depth is represented as a conditional expectation of the image feature vector, which is composed of the set of statistical measurements that are derived from the spectral signatures. For a training data set, the joint probability density distribution between the mean depth and the image feature vector can be described using a cluster of Gaussian functions, and the parameters for the Gaussian functions are estimated using the EM algorithm [10]. After this, the mean depth for any new image is estimated based on the image feature statistics.

Some studies use scene classification [11] and stage classification [12] to get the depth value according to the relative depth information in these categories. Contrary to the method used in [9] which uses the mean depth information for structure classification, in [12], the image is classified into one of a limited number of typical 3D scene geometries called stages and then the depth information is obtained from the stages. Each stage has a unique depth pattern and provides the characteristics of the scene objects such as location and scales. The initial stages are obtained from a training stage using a large database of images, and 19 categories of stages are derived. A further stage categorization is performed to each stage by analyzing the distribution of gradients in the images. The Gaussian scale-space model is applied to extract the feature information, and the Weibull distribution is used to represent the histograms of the Gaussian derivative filter responses. The parameters for the integral Weibull distribution are estimated using the Maximum Likelihood Estimator (MLE), which fits well the histograms of various types of images. The parameters for 15 stages are trained using the Support Vector Machine (SVM). For a given image, the image is analyzed using the Gaussian derivative filter and Weibull distribution to get the feature vectors and is then fitted into a collection of best matching stages which have unique depth profiles.

2.2 Depth Map from Multiple Image Views

Besides the depth estimation from a single view, the depth information can be obtained from multiple views of the scene. The projection reconstruction of a 3D model using the factorization method to get the camera motion and object structure is discussed in [13] [14] [15] [16] and [17].

The work done in [13] builds a measurement matrix from the 2D feature points in multiple views using the 2D points' coordinates, and uses the SVD algorithm to decompose the measurement matrix into a product of two matrices, one representing the camera rotation and the other one giving information about the depth of feature points under the projective transformation. The translation vectors of the cameras in multiple views are computed from the average of the rows of the measurement matrix. The metric transformation is computed by enforcing some constraints on the parameters of the camera matrices.

To estimate the projection matrix and the 3D points from 2D feature points in multiple views, the work done in [14] and [15] represents the projection relationship among the 2D points, the projection matrix and the 3D points using an arbitrary scale factor, such as

$$\lambda_{mn} x_{mn} = P_m X_n \tag{1}$$

where m is the view number, n is the index of 3D point, λ_{mn} is the scale factor, X_n is the n^{th} 3D point, x_{mn} is the n^{th} projected 2D point in the m^{th} view, and P_m is the projection matrix of the m^{th} view. In [14], the scale factor λ_{mn} is calculated using the fundamental matrices and the epipoles that are estimated from the 2D feature points. Then the 2D feature points in multiple views are weighted by the scale factor λ_{mn} to form the measurement matrix, which is decomposed further using SVD to produce the projection matrices and the 3D points. In [15], the scale factor λ_{mn}, the projection matrix and the 3D points are estimated recursively by minimizing the 2D reprojection error using the SVD factorization and the weighted least squares (WLS) algorithm [18]. Some iterative factorization methods to minimize the reprojection error, which do not require the knowledge of the scene geometry, are illustrated in [16] and [17].

Other than using the factorization method for 3D modeling, the depth information can be generated by searching correspondences among multi-view images and by applying triangulation on the feature points [19]. A complete system to build visual models from multi-view camera images is presented in [20]. The system can deal with uncalibrated image sequences acquired with a hand-held camera. Additionally, no prior information about the camera calibration and motion is required for this system.

The implemented multi-view 2D to 3D conversion system in this thesis is based on the system of [20]. While the authors of [20] provide a general framework without providing details about how to implement the individual components of the 2D to 3D system, this thesis presents detailed description and analysis of all the implemented components of the system. The feature detection and matching procedure in this thesis is different from that in [20].

3. BACKGROUND

This chapter gives some background knowledge on 3D modeling in computer vision. In Section 3.1, the camera geometry and the pinhole camera model, which are basic for the analysis of 3D systems, are illustrated. In Section 3.2, the epipolar geometry between multiple views and the fundamental matrix are introduced for further use. The dual absolute quadric and its basic properties are described in Section 3.3.

3.1 Camera Geometry

In computer vision, homogeneous representations of lines and points are described as follows.

A line passing through the point $(x, y)^T$ can be described as:

$$ax + by + c = 0 \tag{2}$$

So, the vector $l = (a, b, c)^T$ is the homogeneous representation of the line in the projective space. Alternatively, the line can be described using the vector $(a, b, c)^T$. Therefore, equation (2) can be written in the form of two inner products:

$$l^T \cdot x = (a, b, c) \cdot (x, y, 1)^T = 0 \tag{3}$$

According to (3), a 2D point can be expressed using a three-dimensional vector $(x, y, 1)^T$, whose third element serves as a scale factor. For a more general case, the homogeneous representation $(x, y, z)^T$ of a point denotes the point $(x/z, y/z)^T$ in 2D vector form. Similarly, the three-dimensional point $X = (x, y, z)^T$ can be represented using the homogeneous notation as

$X = (x, y, z, 1)^T$, and the plane π on which X lies is represented in homogeneous form as

$$\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T \tag{4}$$

A 3D point X lying on the plane π satisfies:

$$\pi^T \cdot X = (\pi_1, \pi_2, \pi_3, \pi_4) \cdot (x, y, z, 1)^T = 0 \tag{5}$$

A basic camera model is the projective pinhole camera geometry as shown in Fig. 1. It is assumed that the camera center is the origin of a Euclidean coordinate system. The camera center O_C is also called the optical center. The image captured by the camera is typically projected on the camera plane (also called the focal plane) behind the camera center, with a negative focal length $-f$ on the z-axis. In addition, according to the imaging mechanism of cameras, the image on the camera image plane is upside-down with respect to the real scene. In the model in Fig. 1, the image plane is placed to be in front of the camera center, and the distance from the image plane to the center point is the focal length f. In this latter case, the image does not have to be inverted. The plane, which passes through the camera center and is parallel to the image plane, is denoted as the principal plane. The line, passing through the camera center and perpendicular to the image plane, is called the principal axis. The intersection of the principal axis with the image plane is a point called the principal point.

In this camera model (Fig. 1), a 3D point in space at a position $X = (x, y, z)^T$, can be mapped to the image plane by forming a line starting at the

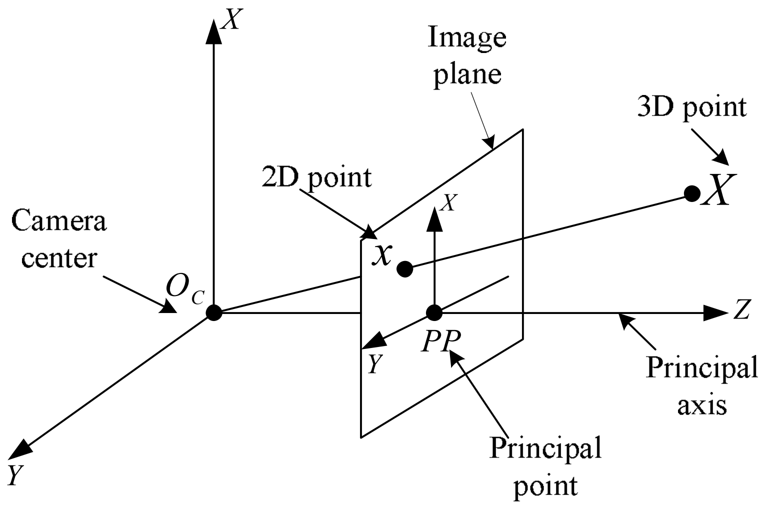


Fig. 1. Pinhole camera geometry. The image plane is in front of the camera center O_C.

camera center to the point X, and the intersection of this line with the image plane is a 2D point x lying on the image plane. Using the similar triangles, the position of x with respect to the camera center can be represented in homogeneous coordinates as $\left(\frac{f \cdot x}{z}, \frac{f \cdot y}{z}, f, 1\right)^T$ in the 3D space. The homogeneous representation of the 2D point x on the image plane is $(f \cdot x, f \cdot y, z)^T$, while the origin of the image plane is the same as the principal point PP.

A projective camera [21] is modeled though the projection equation as

$$x = PX \tag{6}$$

where x represents the 2D point in homogeneous representation, that is, it is a

3×1 dimensional vector. P is a 3×4 projection matrix. X stands for the 3D point in homogeneous representation, and it is a 4×1 dimensional vector.

The Projection matrix P can be represented as

$$P = K[R \mid t] \tag{7}$$

where R is a 3×3 rotation matrix representing the orientation of the camera coordinates with respect to the world coordinates, t is a 3×1 translation vector which shifts the camera center O_C with respect to the world coordinate system, and t is given by

$$t = -R\cdot O_C \tag{8}$$

The transformation (including rotation and translation) between different coordinates is shown in Fig. 2. In (7), K is the intrinsic camera matrix, called the camera calibration matrix and is given by

$$K = \begin{bmatrix} f & s & u \\ 0 & \alpha f & v \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

where f is the focal length of the camera, α is the aspect ratio of the pixel size on the image plane in the x and y directions, (u,v) represents the coordinates of the principal point with respect to the left bottom corner of the image plane, and s is the skew factor which is non-zero if the x and y axes of the image coordinates are not perpendicular.

3.2 Epipolar Geometry and Fundamental Matrix

The geometry between two views of the same scene can be represented using the

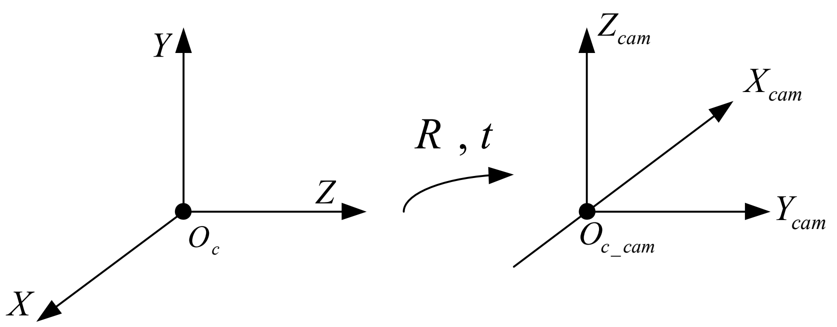


Fig. 2 . An example of a rotation and translation between different projective coordinates.

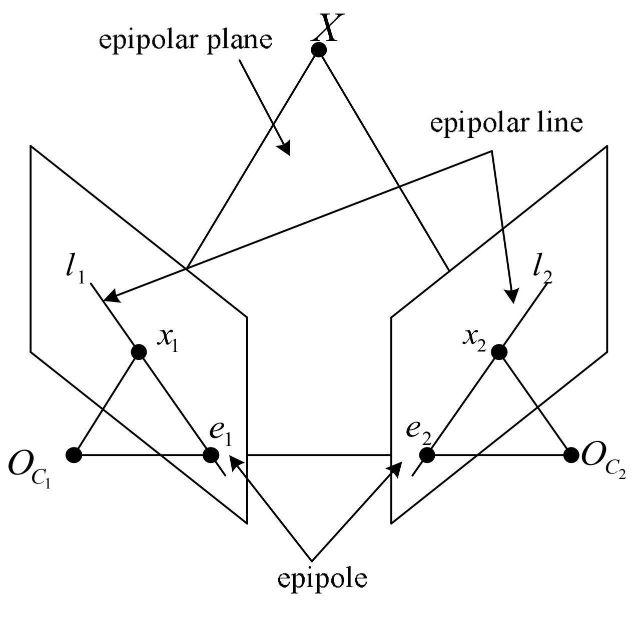


Fig. 3. Point correspondence geometry between two views.

epipolar geometry. The epipolar geometry is illustrated in Fig. 3. Suppose a 3D point X in space is projected into two views to generate 2D points x_1 and x_2, respectively. As three points form a plane, x_1, x_2 and X would lie on a common plane π_E. The plane π_E is denoted as the epipolar plane. The line connecting the two camera center is the baseline between two views, and it also lies on the epipolar plane. The intersection points of the baseline with the two views are the epipoles denoted by e_1 and e_2, one in each view. The line, which connects the 2D point and the corresponding epipole on the same plane, is called the epipolar line. The epipolar line l_2 in the second view is parallel to the ray through x_1 and the camera center O_{C_1}, and it is the projected image in the second view of that ray. Since the 3D point X lies on the ray through x_1 and camera center O_{C_1}, the projected 2D point x_2 of the 3D point X in the second view must be lying on the epipolar line l_2.

From the above discussion, any point x_2 in the second image that matches the point x_1, must lie on the epipolar line l_2, and the epipolar line l_2 in the second view is the mapped image of the ray through x_1 and camera center O_{C_1}. So, there is a mapping between the 2D point in one view and the epipolar line in the other view. The fundamental matrix F_{12} is defined to represent this mapping relationship $x_1 \xrightarrow{F_{12}} l_2$. Similarly, the fundamental matrix F_{21} represents the mapping between x_2 and l_1. The fundamental matrix is the algebraic representation of the epipolar geometry, and it is a 3×3 matrix with a rank of 2.

The epipolar line l_2 corresponding to the 2D point x_1 is represented by

$$l_2 = F_{12} x_1 \tag{10}$$

The fundamental matrix relates to the corresponding epipoles e_1 and e_2 as follows:

$$e_2^T F_{12} = 0 \tag{11}$$

$$F_{12} e_1 = 0 \tag{12}$$

From (11) and (12), the epipole e_1 in the first view is the right null-space of F_{12}, and the epipole in the second view e_2 is the left null space of F_{12}. Epipoles for two views can be computed from the fundamental matrix using the singular value decomposition (SVD). Suppose M is a $m \times n$ matrix; the singular value decomposition of M is in the form of

$$M = U \Sigma V^T \tag{13}$$

where U is a $m \times m$ unitary matrix, Σ is a $m \times n$ diagonal matrix and V is a $n \times n$ unitary matrix. The diagonal entries of Σ are the singular values of M. The column vectors of U are the left-singular vectors of M, and the column vectors of V are the right-singular vectors of M. That is, the relationship of the corresponding left singular vector u, right singular vector v, and the singular value σ can be represented as

$$u^T M = \sigma v^T \tag{14}$$

$$Mv = \sigma u \tag{15}$$

Since the rank of the fundamental matrix is 2, in the SVD of F, the third singular value in the diagonal matrix is zero. According to (14) and (15), if U and V are, respectively, the left singular and right singular matrices in the SVD of the fundamental matrix F, the third column of the left-singular matrix U and the third column of the right-singular matrix V would correspond to the left null vector and the right null vector of the fundamental matrix F, respectively, and would satisfy (11) and (12). Thus, the epipole in the second view is computed from the third column of the left-singular matrix U in the SVD of the fundamental matrix F, and the epipole in the first view is given by the third column of the right-singular matrix V in the SVD of the fundamental matrix F.

As stated in [22], the two 2D points x_1 and x_2, corresponding each to the projection of the 3D point X into two different views, are related as follows:

$$x_2^T F_{12} x_1 = 0 \tag{16}$$

and

$$x_1^T F_{21} x_2 = 0 \tag{17}$$

From (16) and (17), the fundamental matrices, F_{12} and F_{21}, can be related as

$$F_{21} = F_{12}^T \tag{18}$$

In this thesis, the fundamental matrix F_{12} is denoted as F for simplicity.

The fundamental matrix F can be computed from the projection matrices P_1 and P_2 for two views as follows:

$$F = [e_2]_\times P_2 P_1^+ \tag{19}$$

where P_1^+ is the pseudo-inverse of P_1, described as

$$P_1^+ = P_1^T \left(P_1 P_1^T\right)^{-1} \tag{20}$$

and $[e_2]_\times$ is the skew-symmetric matrix of e_2. That is, suppose $e_2 = (a,b,c)^T$,

$$[e_2]_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{21}$$

According to (19), in the special case when $P_1 = K[I \mid 0]$ and $P_2 = K[M \mid m]$, the fundamental matrix is derived as [21]

$$F = [m]_\times M \tag{22}$$

It is also possible to compute F, without information about the camera projection matrices, only from the corresponding image points. F can be derived up to a scale factor from a minimum of 7 point correspondences [21] or using the 8-point algorithm [21].

The 8-point algorithm that is used to calculate the fundamental matrix F is implemented using the Direct Linear Transformation (DLT) algorithm [21]. Given a set of corresponding image points, which contains more than 8 correspondences, $x_i^1 \leftrightarrow x_i^2$, the first step is to normalize the corresponding points to a new set of points such that the centroid of the new points is the coordinate origin $(0,0)^T$, and the average distance of the points from the origin is $\sqrt{2}$. This can be represented using two homogeneous transformations T_1 and T_2 as follows:

$$\hat{x}_i^1 = T_1 x_i^1 \tag{23}$$

$$\hat{x}_i^2 = T_2 x_i^2 \tag{24}$$

The second step is to use the new set of corresponding points to calculate the fundamental matrix F. By substituting $\hat{x}_i^1 = (a_i^1, b_i^1, 1)^T$ and $\hat{x}_i^2 = (a_i^2, b_i^2, 1)^T$ in (16), an expansion of (16) can be expressed as [21],

$$a_i^2 a_i^1 f_{11} + a_i^2 b_i^1 f_{12} + a_i^2 f_{13} + b_i^2 a_i^1 f_{21} + b_i^2 b_i^1 f_{22} + b_i^2 f_{23} + a_i^1 f_{31} + b_i^1 f_{32} + f_{33} = 0 \quad (25)$$

and $\hat{F}$ is a 3×3 matrix with 9 unknown entries as follows

$$\hat{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \quad (26)$$

Using n sets of correspondences, a set of linear equations are formed giving an overdetermined system of equations as follows:

$$A \cdot \hat{f} = \begin{bmatrix} a_1^2 a_1^1 & a_1^2 b_1^1 & a_1^2 & b_1^2 a_1^1 & b_1^2 b_1^1 & b_1^2 & a_1^1 & b_1^1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_n^2 a_n^1 & a_n^2 b_n^1 & a_n^2 & b_n^2 a_n^1 & b_n^2 b_n^1 & b_n^2 & a_n^1 & b_n^1 & 1 \end{bmatrix} \hat{f} = 0 \quad (27)$$

where $\hat{f}$ is a column vector and is formed by unwrapping the fundamental matrix $\hat{F}$ row-wise as follows:

$$\hat{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^T \quad (28)$$

The least square solution for $\hat{f}$ can be computed using the SVD. In the SVD of A, A can be factorized as

$$A = U \Sigma V^T \quad (29)$$

The right-singular column vector in V that corresponds to a singular value of zero is equal to $\hat{f}$. And by wrapping $\hat{f}$ back to a 3×3 matrix, the fundamental matrix $\hat{F}$ can be computed. Note that, in practice, the smallest singular value of A can be non-zero but is very small.

Since the fundamental matrix should have a rank of 2, this constraint should be enforced to ensure this property. Once the fundamental matrix $\hat{F}$ is computed, the SVD of $\hat{F}$ is computed again, and the smallest singular value in the diagonal matrix is set to zero, so that the rank of the fundamental matrix becomes 2.

Finally, by multiplying the normalization matrices T_1 and T_2 with $\hat{F}$, the fundamental matrix F is obtained as

$$F = T_2^T \hat{F} T_1 \tag{30}$$

3.3 Properties of Conics and Quadrics

3.3.1 General conics and quadrics

The conic C, also called point conic, is a curve in the 2D plane and can be described using a second-degree equation. The hyperbola, ellipse and parabola are the main types of conics. In the homogenous representation, the conic is represented as a 3×3 symmetric matrix with 5 degrees of freedom. The point conic C can be represented in matrix form as

$$x^T C x = 0 \tag{31}$$

where x is a 2D point on the conic.

The dual conic C^*, also called line conic, is a conic defined by the lines that are tangent to the point conic C as

$$l^T C^* l = 0 \tag{32}$$

The lines l that satisfy (32) are tangent to the point conic C. Thus, the dual conic C^* is the adjoint matrix of the conic C. Here, the adjoint matrix of an invertible matrix M is given by

$$M^* = \det\left(M\right)\cdot M^{-1} \tag{33}$$

C^* is called the dual conic because it is defined from lines, while C is defined from 2D points, and there is a duality between the 2D points and the lines. For example, in (3), the role of the 2D point x and the line l can be interchanged since $l^T x = 0$ implies $x^T l = 0$. In addition, the cross product of two lines is a 2D point, while the cross product of two 2D points produces a line, represented as

$$l_1 \times l_2 = x \tag{34}$$

$$x_1 \times x_2 = l \tag{35}$$

In the same sense, there is also a duality between a 3D point and a plane.

Similarly, in 3D space, the quadric Q is a surface defined from the 3D point on the quadric. Q can be represented as

$$X^T Q X = 0 \tag{36}$$

where the quadric Q is a symmetric 4×4 matrix and the 3D point X is a 4×1 vector. The dual quadric Q^* is a quadric defined from the planes π that are tangent to the quadric Q as

$$\pi^T Q^* \pi = 0 \tag{37}$$

So, a plane that is tangent to the quadric Q satisfies (37). The dual quadric is the adjoint matrix of the quadric. The intersection of a plane α with a quadric Q

is a conic C.

3.3.2 The absolute conic and the dual absolute quadric

The absolute conic Ω_∞ is a point conic on the plane at infinity. As a special case in the homogeneous representation of 3D points, if the fourth entry of the point vector is zero, the 3D point $(X_1, X_2, X_3, 0)^T$ is not a real point in space and it is called an ideal point. The ideal points all lie on an imaginary plane called the plane at infinity. The homogeneous representation of the plane at infinity is

$$\pi_\infty = (0,0,0,1)^T \tag{38}$$

So that

$$\pi_\infty^T X_{ideal} = 0 \tag{39}$$

where X_{ideal} is an ideal 3D point on the plane at infinity.

The absolute conic Ω_∞ can be represented in matrix form as

$$X^T \Omega_\infty X = 0 \tag{40}$$

where X is a 3D point lying on the absolute conic Ω_∞.

The dual absolute quadric Q_∞^* is the dual of the absolute conic Ω_∞. The dual absolute quadric is a surface formed of all planes tangent to the absolute conic Ω_∞, and is represented in the homogeneous form as a 4×4 matrix of rank 3.

Here are some properties of Q_∞^* [23]:

- Q_∞^* is a degenerate quadric. It is singular and its rank is 3. It has 8 degrees of freedom.

- Q_∞^* is symmetric and positive semi-definite (PSD).

- The plane at infinity π_∞ is a null vector of Q_∞^*. That is,

$$Q_\infty^* \pi_\infty = 0 \tag{41}$$

- The dual absolute quadric Q_∞^* has a canonical form under the metric transformation as

$$Q_\infty^* = \hat{I}_{4\times4} = \begin{bmatrix} I_{3\times3} & 0 \\ 0_{1\times3} & 0 \end{bmatrix} \tag{42}$$

so that the projective transformation between the Q_∞^* in projective space and that in metric transformation is represented as

$$\hat{I}_{4\times4} = diag(1,1,1,0) = HQ_\infty^* H^T \tag{43}$$

where H is a homography matrix which transforms Q_∞^* from the projective frame to the metric one.

3.3.3 The dual image of the absolute conic (DIAC)

As stated in [21], the image of the absolute conic (IAC), denoted as ω, is used to represent the mapping between the points of the absolute conic on the plane at infinity and the points on the camera image plane. The IAC is the projected image of the absolute conic Ω_∞ on a 2D image plane. The IAC is a point conic represented as:

$$\omega = K^{-T} K^{-1} \tag{44}$$

where K is the intrinsic camera matrix.

The dual image of the absolute conic (DIAC), denoted as ω^*, is the dual of the IAC ω. The DIAC is the projected image of the dual absolute quadric Q_∞^* on the image plane by a projection matrix P. This is described as:

$$\omega^* = PQ_\infty^* P^T \tag{45}$$

As the DIAC is the adjoint matrix of the IAC, the DIAC ω^* can be represented using only the intrinsic camera matrix K as [21]

$$\omega^* = KK^T \tag{46}$$

From (46), if the DIAC is calculated, the intrinsic camera parameters can be estimated.

3.4 The Hierarchy of Transformations

In 3D space, the points, lines and planes can be transformed using a homography matrix H. Due to different geometric properties of the transformations, there is a hierarchy of transformations starting from the projective transformations to the affine transformations, the metric transformations, and finally generating the most specialized Euclidean transformations.

The projective transformation is the least strict transformation among these four types of transformations as it produces the most distortion to the original shapes of the objects in space. The projective transformation can be represented in homogeneous form as

$$H_P = \begin{bmatrix} A & t \\ V^T & v \end{bmatrix} \tag{47}$$

where A is a 3×3 matrix, t is a 3×1 translation vector, V is a 3×1 vector and v is a scalar. H_P has 15 degrees of freedom (dof). The projective transformation does not preserve the orientations or similarities with respect to the original shapes.

The homogeneous representation of an affine transformation is given by

$$H_A = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \tag{48}$$

where A is a 3×3 matrix and t is a 3×1 translation vector. H_A has 12 degrees of freedom. The affine matrix A consists of two fundamental transformations, rotations and non-isotropic scaling in X, Y and Z directions. Thus, the similarity of area ratios and angles between planes are not preserved in an affine transformation. But the parallelism of planes, the ratio of areas on parallel planes and the ratio of volumes are preserved.

The metric transformation is a transformation that consists of rotation, isotropic scaling and translation. It can be represented in homogeneous form as

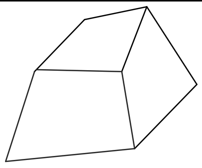
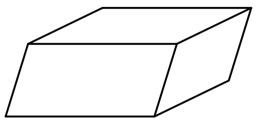
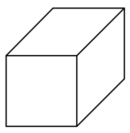
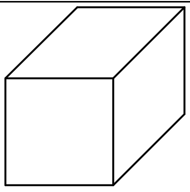
$$H_{\mathrm{M}} = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \tag{49}$$

where t is a 3×1 translation vector, s is a scalar, and R is a 3×3 rotation matrix and is an orthogonal matrix such that

$$R^T R = RR^T = I \tag{50}$$

The metric transformation matrix H_{M} has 7 degrees of freedom. The metric transformation is stricter than the affine transformation because it also preserves the angle between different planes. The scalar s has the effect of scaling the object so that the volume is changed.

Table 1. Geometry properties of different types of transformations.

Name	Matrix Definition	Distortion	Invariant Properties
Projective (15 dof)	$H_P = \begin{bmatrix} A & t \\ V^T & v \end{bmatrix}$		Intersection and tangency of surfaces in contact.
Affine (12 dof)	$H_A = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Parallelism of planes, volume ratios, centroids. The plane at infinity.
Metric (7 dof)	$H_{\mathrm{M}} = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Volume ratios, angle ratios. The absolute conic.
Euclidean (6 dof)	$H_{\mathrm{E}} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Volume, angle.

The Euclidean transformation is the strictest transformation because it only rotates and translates the objects in the 3D space, without changing the ratio and shape of the objects. The homogeneous representation of the Euclidean transformation is

$$H_{\mathrm{E}} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \tag{51}$$

where t is a 3×1 translation vector and R is a 3×3 rotation matrix and is an orthogonal matrix. H_{E} has 6 degrees of freedom.

The definitions and properties of these transformations are summarized in Table 1.

4. IMPLEMENTED MULTI-VIEW 2D TO 3D CONVERSION SYSTEM

This chapter describes the implemented procedures for the multi-view 2D to 3D conversion system. There are five major steps in the 3D modeling process: image feature detection and registration using the scale invariant feature transform (SIFT), removing the outliers by exploiting the two-view geometry using the random sample consensus (RANSAC), estimating the projective 3D structures through triangulation, projective transformation refinement using bundle adjustment, and upgrading to metric reconstruction through auto-calibration. The implementation details are discussed after the description of each stage. Section 4.1 presents an overview of the implemented 2D to 3D conversion system. The five main components of the 3D modeling system are described in Sections 4.2 to 4.6. Section 4.7 describes how to estimate the sparse depth map relative to the middle camera center. Additional implementation notes are given in Section 4.8.

4.1 Overview of the Implemented System

The performance of the 3D depth estimation improves with the number of available multiple 2D views. In this work, 8 views from different viewpoints are processed and they are found to be sufficient for proper depth reconstruction as shown in Chapter 5.

The components of the implemented 2D to 3D conversion system are shown in a flowchart in Fig. 4. After reading in the multiple view images, the first step for relating multiple views to each other is to extract features in each view and match the extracted features between different views. For this purpose, an algorithm called the scale invariant feature transform (SIFT) is used for the extraction and matching of feature points in multiple views. Feature detection and matching is implemented between the middle view and one of the other views at each time, and the two-view geometry is estimated for each pair of views using the corresponding feature points in the two considered views. To remove the outliers in the feature matching, a robust algorithm called the random sample consensus (RANSAC) is applied to the matching feature points. In this step, the projection matrices between the middle view and the other views are estimated from the inliers of feature points. Besides, the common feature points between all 8 views are determined from the set of inliers. The next step is to retrieve the structure of the 3D scene and the positions of the multiple cameras using the common feature points and geometries of all views. For this purpose, triangulation is implemented to produce the projective reconstruction of the 3D scene, and the projection matrices of multiple views are refined through bundle adjustment. After bundle adjustment, the refined common 3D feature points are back-projected to multiple 2D views and the average reprojection errors of the 2D points in all views are calculated. If the average reprojection error is smaller than a threshold, the projective reconstruction of the 3D scene is upgraded to a metric one. If the reprojection error is large, the procedure is repeated starting from the RANSAC step to re-estimate the geometry and structure from new sample sets of feature points. After the metric upgrade, sparse depth maps, consisting of relative depth values at the locations of the extracted feature points that are common

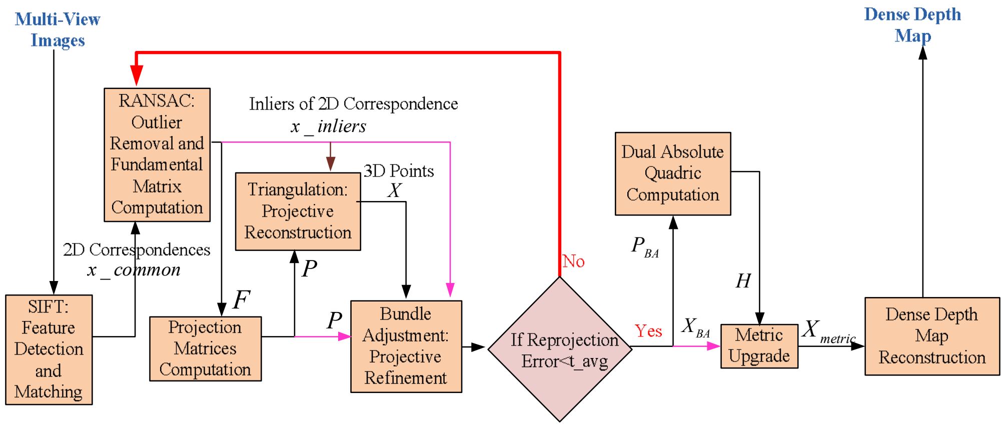

Multi-View Images
SIFT: Feature Detection and Matching
2D Correspondences
x _ common
RANSAC: Outlier Removal and Fundamental Matrix Computation
Inliers of 2D Correspondence
x _ inliers
F
Projection Matrices Computation
P
Triangulation: Projective Reconstruction
3D Points
X
Bundle Adjustment: Projective Refinement
If Reprojection Error<t_avg
No
Yes
Dual Absolute Quadric Computation
P_BA
H
X_BA
Metric Upgrade
X_metric
Dense Depth Map Reconstruction
Dense Depth Map

Fig. 4. Flowchart of the implemented 2D to 3D conversion system. F represents the fundamental matrix between the middle view and the other views, P represents the projection matrices for all views, X_BA and P_BA are the refined 3D points and projection matrices, respectively.

among all views, can be estimated. Details about the SIFT, RANSAC, triangulation, bundle adjustment and metric upgrade are discussed in Sections 4.2 to 4.6, respectively.

4.2 Scale Invariant Feature Transform (SIFT)

For a given multi-view image or video set, the first step is to relate different views to each other by finding, in the multiple views, the relevant feature points that correspond to the same 3D point in space. A restricted number of corresponding points, which spread over most regions of a scene, is sufficient to determine the geometric model. Thus, the first step is to detect the suitable features points in the 2D multiple views and to match the selected feature points among different views.

In the implemented system, a feature detection and tracking method called the scale invariant feature transform (SIFT) [24] is applied to detect the feature points and generate the invariant feature descriptors for each feature point. The generated feature descriptors are further used to match the feature points in different views. The SIFT consists of five major steps: scale-space extrema detection, keypoint localization, orientation assignment, keypoint descriptor and keypoint matching. This algorithm is able to generate a large number of feature points that are densely distributed over a wide range of scales and most locations in the image, while being robust to scaling and rotation in 3D viewpoints, and to changes in illumination.

4.2.1 Scale-space extrema detection

The first stage of the feature detection and tracking is to find the candidate locations that are invariant to scale changes in multiple views. This is done by searching for the extrema in the Gaussian scale-space.

The Gaussian scale-space $L(x, y, \sigma)$ for the input image $I(x, y)$ is defined as [25]

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{52}$$

where $G(x, y, \sigma)$ is the Gaussian function given by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{53}$$

The difference between two different scales is calculated to generate the difference-of-Gaussian (DOG) function, which is represented as

$$D(x, y, \sigma) = \left(G(x, y, s\sigma) - G(x, y, (s-1)\sigma)\right) * I(x, y) = L(x, y, s\sigma) - L(x, y, (s-1)\sigma) \tag{54}$$

The detailed procedures for the extrema detections are illustrated in Fig. 5. There are several octaves which help to reduce the computations of the scale-space representation. In each octave of scale-space, the scale σ increases from the initial scale σ_0 (at the begging of each octave) to twice of it, and each octave of scale is divided into an integer number of intervals S. So the constant factor k separating nearby scales is $k = 2^{1/S}$. At the s^{th} stage in an octave, the scale factor is $k^s\sigma_0$. The input image is convolved with Gaussian functions to produce different scale-space images. The adjacent image scale functions are subtracted to produce the Difference of Gaussian (DOG) images. In the next octave, the Gaussian images with twice the initial value σ_0 are down-sampled by 2, and the same operations as in the previous octave, are performed without losing accuracy with respect to σ.

As illustrated in Fig. 6, in all scale levels, the maxima and minima points in the DOG images are detected by comparing a pixel (marked with X in Fig. 6) to its 26 neighbors in 3×3 regions at the current and adjacent scales in the DOG. These extrema points are also called keypoints and they are invariant to scale differences in different images. [24]

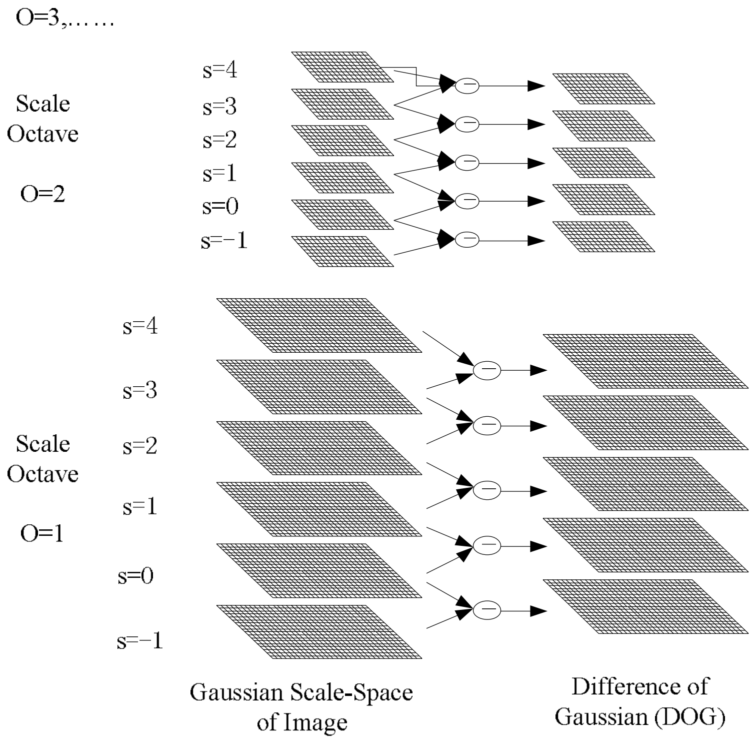


Fig. 5. Computing the Difference of Gaussian functions at different octaves and scales.

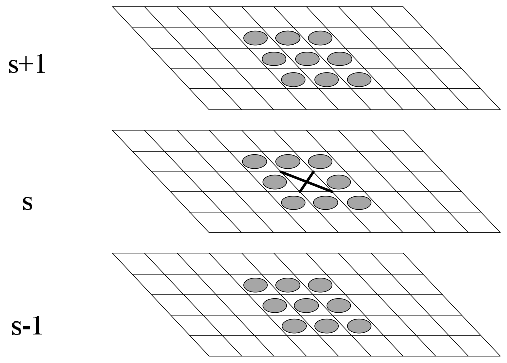


Fig. 6. Detecting maxima and minima in the DOG functions. The point marked with ‘X’ is the evaluation point. The points marked with circles are the surrounding points used to determine whether the evaluation point is an extrema or not.

4.2.2 Keypoint localization

Keypoint localization performs a more accurate localization of the keypoint according to the nearby data, and eliminates points with low contrast or that are poorly localized along an edge. By setting the derivative of the Taylor series expansion of $D(x, y, \sigma)$ [26] to be zero, the offset from the original detected point is calculated and accurate locations of the keypoints are determined. Through thresholding the second-order Taylor expansion of the DOG function at the offset, the keypoints with low contrast are removed. On the edges, there are some unstable keypoints which have large curvatures across the edge but small curvatures in the perpendicular direction. The parameters of a 2×2 Hessian matrix are used to calculate the ratio of the curvature across the edge and the curvature in the perpendicular direction. If the ratio is larger than a threshold, the keypoint is discarded.

4.2.3 Orientation assignment

By adding the orientation information of keypoints to the content of the descriptor, the matching process will be invariant to the rotation of objects in different views. Orientation assignment is performed at the detected keypoints by creating a gradient histogram multiplied by the gradient magnitude and a circular Gaussian window. The gradient magnitude and orientation are computed for a detected feature point at location (x, y) in the scale image $L(x, y, \sigma)$ as

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (55)$$

$$\theta(x,y) = \tan^{-1}\left({(L(x,y+1) - L(x,y-1))^2} \Big/ {(L(x+1,y) - L(x-1,y))^2} \right) \quad (56)$$

The orientation histogram includes the gradient orientations of the sample points surrounding the keypoint (in a 4×4 region around the keypoint), weighted by the gradient magnitude and a Gaussian circular window with a standard deviation equal to 1.5 times the scale of the considered keypoint. The histogram has 36 bins representing the 360 degrees orientation range. Fig. 7 illustrates an example of the orientation histogram with only 8 bins. The dominant direction of the keypoint corresponds to the peak in the orientation histogram.

4.2.4 Keypoint descriptor

From the previous steps, the keypoints are properly localized and refined, and their scale and dominant orientations are determined. The keypoint descriptors are formed to describe the features of the keypoints so that the corresponding keypoints can be tracked with respect to similar features in their descriptors. For a feature point, the 16×16 surrounding area of the keypoint is divided into 4×4 sub-regions and is used to calculate the descriptor. After Gaussian smoothing, the gradient magnitudes and orientations of the samples in the 4×4 sub-regions are calculated and an 8 bin histogram for each 4×4 sub-region is generated. The values of the orientation histogram entries are stored as a column vector to represent the descriptor of a keypoint. As the descriptor for a keypoint contains a 4×4 array of histograms, which each contains 8 direction bins, the dimension of the keypoint descriptor is 4×4×8=128, as illustrated in Fig. 8. In order to ensure the invariance of the descriptors to rotation, the descriptors, which consist of the

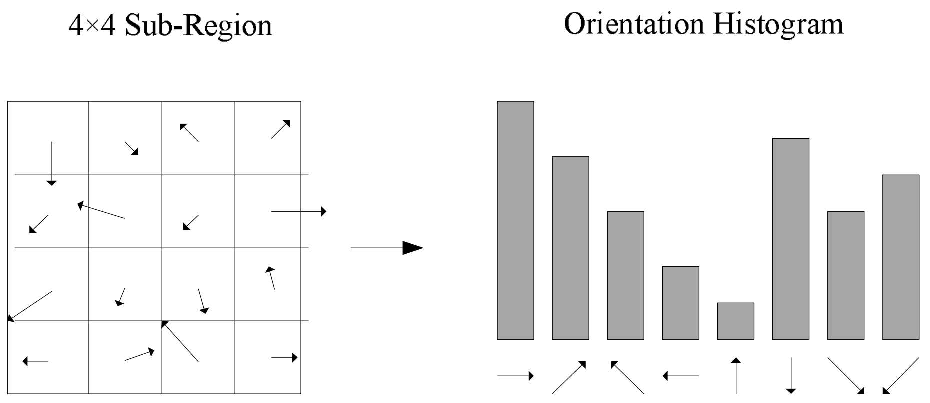


Fig. 7. Orientation histogram of pixels in a 4×4 sub-region. The histogram has 8 bins.

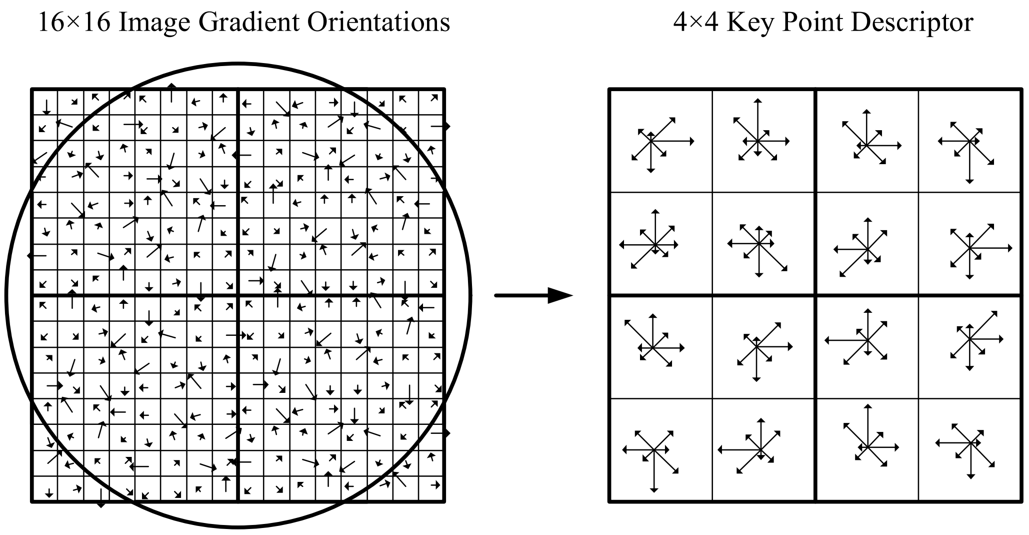


Fig. 8. Creating a keypoint descriptor using a 16×16 surrounding area of a keypoint. The 16×16 region is firstly smoothed by a Gaussian filter, illustrated using the circle. The 8 bin orientation histograms of the 4×4 sub-regions are calculated; thus, the descriptor consists of 128 entries.

orientations of gradients, are rotated by an angle $-\theta$, where θ is the angle of the dominant direction of the keypoints.

4.2.5 Keypoint matching

Using the above steps, the keypoints and their descriptors in each view can be determined. The next step is to relate the keypoints in different views and to find the matching feature points between two different views. A modified K-D tree algorithm called Best-Bin-First method [27] is used to find the descriptor of the keypoint in one view with minimum Euclidean distance from the descriptor in the other view.

4.2.6 Implementation notes for the SIFT

In the implemented system, the corresponding features are tracked between the middle view (View 4) and other views. The camera coordinates of the middle view are assigned to be the same as the world coordinates, so that the projection matrix for the middle view P_4 is assumed to be $[I_{3\times3} \mid 0_{3\times1}]$.

The SIFT algorithm is implemented between the middle view and one of the other views at each time; so, there are 7 loops totally among 8 views. In the function of ‘runsift’, it calls two functions to do the SIFT, ‘sift’ and ‘siftmatch’. First, the ‘sift’ function is used to return the Gaussian scale-spaces and Difference-of-Gaussian scale-spaces. In addition, the scale, the dominant orientation and the descriptor of the keypoints are stored in the output of 'sift'. Using the descriptors of the feature points, coordinates of the matching points between the middle view and one of the other views are calculated using the function 'siftmatch'.

In a single iteration, the outputs of ‘runsift’ are two $2 \times N$ matrices, where N is the number of matching points. Each matrix corresponds to a view, and each column of a matrix contains the coordinates of a 2D keypoint in a view. The columns with the same index in the two matrices correspond to matching 2D keypoints. As the number of matching points between different pairs of views may not be the same, a cell array consisting of matrix elements is used to store the coordinates of the matching 2D points in all iterations. After the matching is performed between the middle view and the other 7 views, the resulting cell array consists of 14 cells.

The MATLAB SIFT toolbox that is used here, can be downloaded from [28], and is provided by Andrea Vedaldi.

4.3 Random Sample Consensus (RANSAC)

4.3.1 Description of the RANSAC algorithm

In the feature detection and matching step, since the descriptors of feature points, which are used for feature matching in multiple views, are determined with respect to the local sub-region around the feature points, there may be some inaccurate matching between feature points in multiple views. These mis-matching pairs of feature points are also called the outliers. An outlier removal and model estimation method called the random sample consensus (RANSAC) [29] is used to remove the outliers and to estimate the two-view geometry using the inlier feature points. RANSAC is a robust estimation algorithm which operates in an opposite manner as compared to the conventional smoothing techniques, such as Least Square Optimization [30]. Instead of using a large data set to obtain an initial solution, followed by removing the invalid data, RANSAC uses only a small initial data set to calculate the solution model and, then, it determines the solution accuracy according to the applicability of the solution to other data points with respect to certain prerequisites.

The brief procedure of the RANSAC algorithm is described as follows. To robustly fit a data set S into a model and remove the outliers in the data set, first, a relatively small set consisting of s samples is randomly selected from the input data set S, and the initial solution model is calculated using this small sample set. Second, the whole data set of S is fitted into this solution to calculate the distance from the original data value. Those data samples whose distance is within a threshold t, form the inlier data set S_i. S_i is the consensus data set and only includes the inliers. Third, if the proportion of data in S_i to the data in S is larger than previous trials, the sample trial number N is re-estimated using this probability of inliers. Furthermore, if the current trial count is larger than the estimated N, the solution model is re-estimated using all the inliers in S_i and the procedure terminates. On the other hand, if the current trial count is less than the estimated sample trial number N, a new set of s samples is selected and the calculation is repeated from the first step. Finally, the largest consensus set S_i is selected after a number of N trials, and the model is re-estimated using all the sample points in S_i.

In the above RANSAC algorithm, it is not necessary to compute every possible sample set s from all points. The number of iterations N can be determined using the following estimation method. With a probability p (it is usually chosen to be 0.99), at least one of the selections is a sample set of s data points that are free from outliers. The probability that a selected sample point is an inlier is denoted as ω, and thus the probability for a sample to be an outlier is

$$\varepsilon = 1 - \omega \tag{57}$$

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

$$\left(1 - \omega^s\right)^N = 1 - p \tag{58}$$

where $1 - \omega^s$ 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:

$$N = {\log\left(1 - p\right)} \big/ {\log\left(1 - \omega^s\right)} \tag{59}$$

The interest here is in applying RANSAC to estimate the two-view geometry by finding the fundamental matrix F and removing the outliers in the corresponding image feature points.

The sum of Sampson distances [21] [31] of the eight pairs of corresponding feature points is used to represent the error of the fundamental matrix estimation.

The sum of Sampson distances of the eight pairs of feature points F_error is computed as

$$F\_error = \sum_{i=1}^{8} d_{sampson}\left(x_1^i, x_2^i\right) \tag{60}$$

where x_j^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 the Sampson Distance between x_1 and x_2 and is given by

$$d_{sampson}(x_1, x_2) = \frac{\left((x_2)^T F x_1\right)^2}{(Fx_1)_1^2 + (Fx_1)_2^2 + (F^T x_2)_1^2 + (F^T x_2)_2^2} \tag{61}$$

where $(Fx_j)_n$ is the n^{th} element in the product of F and x_j. The fundamental matrix estimation error F_error in (60) needs to be small to ensure the properness of the estimated fundamental matrix.

In addition, in the calculation of the fundamental matrix F using the 8-point algorithm, the eight 3D points corresponding to the randomly selected set of 2D feature points should not be lying on the same plane in the 3D space. If they 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. In order to ensure that the selected 8 correspondences do not correspond to 3D points lying on the same plane, the homography matrix H between the eight pairs of corresponding 2D feature points in two views, $x_2^i = Hx_1^i$ $(i = 1,2,...,8)$, is computed using the SVD method as follows. Since $x_2^i = Hx_1^i$, the cross product of x_2^i and Hx_1^i is zero.

Suppose $x_1^i = \left(a_1^i, b_1^i, c_1^i\right)^T$ and $x_2^i = \left(a_2^i, b_2^i, c_2^i\right)^T$, the cross product of x_2^i and Hx_1^i can be expressed as [21]

$$A \cdot h = \begin{bmatrix} 0^T & -c_2^i {x_1^i}^T & b_2^i {x_1^i}^T \\ c_2^i {x_1^i}^T & 0^T & -a_2^i {x_1^i}^T \\ -b_2^i {x_1^i}^T & a_2^i {x_1^i}^T & 0^T \end{bmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0 \tag{62}$$

where h_j ($j = 1,2,3$) is the j^{th} column of H. In (62), the matrix A has a rank of 2 and only two of the three equations are linearly independent. For the eight pair of correspondences, by taking the first two equations in (62), a 16×9 matrix can be formed as

$$B \cdot h = \begin{bmatrix} 0^T & -c_2^1 {x_1^1}^T & b_2^1 {x_1^1}^T \\ c_2^1 {x_1^1}^T & 0^T & -a_2^1 {x_1^1}^T \\ \ldots & \ldots & \ldots \\ 0^T & -c_2^8 {x_1^8}^T & b_2^8 {x_1^8}^T \\ c_2^8 {x_1^8}^T & 0^T & -a_2^8 {x_1^8}^T \end{bmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0 \tag{63}$$

After computing the SVD of *B*, the right-singular column vector that corresponds to the smallest singular value is the solution for h. By wrapping h back to a 3×3 matrix, the homography matrix H can be generated. Then, 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 calculated as follows

$$H\_error = \sum_{i=1}^{8} \left(x_2^i - Hx_1^i\right)^2 \tag{64}$$

If the 3D points corresponding to the eight pairs of 2D feature points do not lie on the same plane, the reprojection error H_error is large.

In this context, the RANSAC procedure [21] can be summarized as follows.

Step 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 the random sample set, ω is the probability that a sample is an inlier, p is the probability that all selected sample sets are inliers, t_{HF} is the threshold for the ratio of F_error and H_error and t is the threshold to select the inliers.

Step 2: The value of N for repetition is calculated according to (59).

Step 3: A random sample set of 8 correspondences ($s = 8$) is selected and the fundamental matrix F is calculated using the 8-point algorithm as described in Section 3.2.

Step 4: The ratio $R = \dfrac{H\_error}{F\_error}$ is used to evaluate the accuracy of the obtained solution of the fundamental matrix. If R is greater than a specified threshold t_{HF}, the solution is deemed to be satisfactory and the procedure proceeds to the next step (Step 5); otherwise, the procedure is repeated from Step 3 by randomly selecting another eight pairs of corresponding feature points.

Step 5: For all pairs of corresponding feature points x_1 and x_2, the Sampson distance $d_{sampson}(x_1, x_2)$ in (61) 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.

Step 6: The ratio of the number of inliers to the number of the whole set of feature points is calculated and is denoted as the 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 (59). Furthermore, if the current trial count exceeds the estimated N, the procedure goes to Step 7. Otherwise, if the current trial count is less than N, the procedure repeats from Step 3.

Step 7: After N trials, F is re-estimated using the largest set of inliers.

Based on the computed fundamental matrix F, the projection matrices P_1 and P_2 for two views can be calculated. Suppose F is the fundamental matrix between Views 1 and 2 and, thus, it satisfies (16). The projection matrices P_1 and P_2 are determined as described below.

According to [21], given the fundamental matrix F, the pair of camera projection matrices can be defined as

$$P_1 = K[I \mid 0] \tag{65}$$

$$P_2 = K[SF \mid e_2] \tag{66}$$

where S is any skew-symmetric matrix, and e_2 is the epipole in the second view and satisfies (11). The skew-symmetric matrix is a square matrix whose transpose is equal to its negative, represented as

$$S = -S^T \tag{67}$$

The projection matrices $P_1' = K[I \mid 0]$ and $P_2' = K[A + a \cdot v^T \mid \lambda a]$, where λ is a scalar, a is a 3×1 vector, and v is a 3×1 vector, have the same fundamental matrix as the canonical pair $P_1 = K[I \mid 0]$ and $P_2 = K[A \mid a]$. By assigning S in (67) to be $[e_2]_x$ as defined in (21), the general form of the camera projection matrices can be expressed as follows [21]:

$$P_1 = K[I_{3\times 3} \mid 0_{3\times 1}] \tag{68}$$

$$P_2 = K[[e_2]_x F + e_2 \cdot v^T \mid \lambda e_2] \tag{69}$$

where v is any 3×1 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.

4.3.2 Implementation notes for RANSAC

The RANSAC algorithm is implemented to remove the outliers from the matching feature points that are calculated using SIFT, and to estimate the fundamental matrix F between the middle view and other views. Since the projection matrix for the middle view P_4 is assigned to be $K[I_{3\times 3} \mid 0_{3\times 1}]$, the other one is assigned as $P_i = K[[e_i]_x F_{4i} \mid e_i]$, where F_{4i} denotes the fundamental matrix between the 4th view (middle view) and the i^{th} view $(x_i^T F_{4i} x_4 = 0)$. The epipole in the i^{th} view, e_i, is the third column of the left basis matrix of the SVD of F_{4i}, corresponding to the zero singular value.

The 'ransacfitfundmatrix' function is used for RANSAC. In each loop, two cells of the 2D corresponding feature points, with the first cell corresponding to the middle view, are taken out of the SIFT cell array to be used as the input to this function. As the projection matrix for the first input is assigned to be $K[I_{3\times 3} \mid 0_{3\times 1}]$ in this function, the first input element to this function is the set of 2D feature points in the middle view after SIFT, and the second input element is the set of 2D feature points in the i^{th} view after SIFT. The output of this function is the fundamental matrix F_{4i} and the 2D feature point indices of the inliers between the middle view and the i^{th} view.

In each run of RANSAC, the inliers between two views are calculated. This procedure is implemented between the middle view and each of the other views for seven times. Usually, the number of the common inliers among multiple views will become smaller as the number of views increases.

After RANSAC, the inliers of the common 2D feature points that are common in all the considered eight views are output, and the projection matrices for all views are calculated.

The RANSAC toolbox is provided by Peter Kovesi from The University of Western Australia. It can be obtained from [32]. This software is modified in our implementation by adding Step 4 in the RANSAC procedure as discussed in Section 4.3.1.

4.4 Triangulation

4.4.1 Introduction of the triangulation method

From the RANSAC algorithm, the inliers among the corresponding 2D feature points, and the projection matrices are calculated for multiple views. The triangulation algorithm 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 centers, can be easily computed to get the corresponding 3D point in space. But due to the presence of noise and digitization errors, it is possible that the intersection of these two rays does not exist in the 3D space. That is why triangulation is needed for the 3D point estimation.

In the implemented system, the 3D points are reconstructed using a simple SVD-based algorithm similar to the one described in Section 3.2.

For each 2D feature point, (6) 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 ($i = 1,2,..,8$), as

$$x_i \times P_i X = 0 \tag{70}$$

Suppose $x_i = (a_i, b_i, c_i)^T$ ($i = 1,2,...,8$). By selecting the first two equations from (70) for the 8 corresponding points, 16 equations are represented as

$$\begin{cases} a_1\left(p_1^{3^T} X\right) - p_1^{1^T} X = 0 \\ b_1\left(p_1^{3^T} X\right) - p_1^{2^T} X = 0 \\ \ldots \\ a_8\left(p_8^{3^T} X\right) - p_8^{1^T} X = 0 \\ b_8\left(p_8^{3^T} X\right) - p_8^{2^T} X = 0 \end{cases} \tag{71}$$

where p_j^i represents the i^{th} column of P_j ($i = 1,2,..,8$). This equation set can be expressed in the form of

$$AX = 0 \tag{72}$$

where A is a 16×4 matrix represented as

$$A = \begin{bmatrix} a_1 p_1^{3^T} - p_1^{1^T} \\ b_1 p_1^{3^T} - p_1^{2^T} \\ \ldots \\ a_8 p_8^{3^T} - p_8^{1^T} \\ b_8 p_8^{3^T} - p_8^{2^T} \end{bmatrix} \tag{73}$$

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

$$A = U \sum V^T \tag{74}$$

If the singular values in $\sum$ are arranged in descending order, the solution for X is the last column of V.

The above triangulation procedure assumes no noise in the estimated 2D points. Suppose there is a noisy matching pair of 2D feature points, $x_1 \leftrightarrow x_2$, which do not actually match each other and do not satisfy the epipolar constraint in (16). According to the work done by Hartley and Sturm [33], a constrained MSE-based triangulation method is used to find the corresponding coordinates of the 2D feature points. The relevant points x_1' and x_2' should be lying close to the noisy points, and should also satisfy the epipolar constraint in (16). These correct matching feature points are localized by minimizing the Euclidean distance function

$$[d(x_1, x_1')]^2 + [d(x_2, x_2')]^2 \tag{75}$$

subject to the epipolar constraint

$$x_2'^T F x_1' = 0 \tag{76}$$

With the knowledge of the epipolar geometry as discussed in Section 3.2, any pair of 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 constraint. The optimal 2D points x_1' and x_2', which are closest to the original matching points, would lie on a pair of epipolar lines l_1 and l_2, respectively. The distance equation (75) can be represented using the distance of the noisy points to the epipolar lines:

$$[d(x_1, l_1')]^2 + [d(x_2, l_2')]^2 \tag{77}$$

where $d(x_i, l_i')$ represents the perpendicular distance from point x_i to line l' $(i = 1,2)$. As indicated before, the 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_1' in the first image by a parameter t as $l_1'(t)$. Using the fundamental matrix F, the other epipolar line l_2' is related to l_1'. Thus, the distance function in (77) can be represented as a polynomial function of t. The parameter $t_{\min}$ that minimizes the polynomial distance function is computed by finding the real roots of the nominator in the polynomial function and by evaluating the distance function at each of the real roots. Then, the two epipolar lines at $t_{\min}$ and the corrected 2D points x_1' and x_2' on these lines can be calculated.

The corrected 2D points x_1' and x_2' are used to compute the corresponding 3D point X using the SVD as discussed before.

At the end of triangulation, the projective 3D structure is reconstructed, providing the 3D point X, in addition to the already computed set of projection matrices P and the 2D feature points x for all views.

4.4.2 Implementation notes for triangulation

Using the common matching 2D feature points x and projection matrices P for all views, the 3D feature points can be calculated through triangulation. The function 'vgg_X_from_xP_lin' implements the triangulation assuming no noise. The input of this function consists of 2 matrices. One input matrix stores the coordinates of 8 matching points, one in each view, corresponding to the same 3D point. The second matrix is a three-dimensional matrix storing the projection matrices P of all views. This function uses the SVD method to calculate the 3D points as described in Section 4.4.1, and assumes there is no error in the feature matching.

The triangulation toolbox is provided by Tomas Werner from the University of Oxford. It can be downloaded from [34].

4.5 Bundle Adjustment

4.5.1 Description of the bundle adjustment algorithm

Once the 3D points and projection matrices have been obtained for all views, there is a need to refine the 3D structure through a global minimization step, due to the fact that both the 3D points and the projection matrices which are derived from the fundamental matrices are susceptible to noise. This can be solved using a maximum likelihood estimation produced by the bundle adjustment algorithm [35]. The goal is 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 the reprojected 2D points $P(X)$ are minimized. Bundle adjustment is illustrated in Fig. 9.

The rays emanating from a 3D point and reprojected to the image planes of multiple views form a bundle. Through bundle adjustment, given m 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 the reprojected 2D points $x_{ij}^{BA} = P_i^{BA} X_j^{BA}$ become stable. The reprojected 2D points x_{ij}^{BA} need to minimize the following Euclidean distances from the initial 2D feature points x_{ij}:

$$\min_{P_i, X_j} \sum_{i=1}^{m} \sum_{j=1}^{n} d\left(x_{ij}, x_{ij}^{BA}\right) \tag{78}$$

A typical method of sparse bundle adjustment uses the Levenberg-Marquardt (LM) algorithm [36] [37] to do the non-linear minimization of the reprojection error. The LM algorithm is an iterative procedure that calculates the minimum of a non-linear least square problem. Given an initial measured vector w, an initial parameter vector v and a functional relation f which maps the parameter vector v to an estimated measurement vector as $\hat{w} = f(v)$, the objective is to find iteratively the parameter vector v^+ that minimizes the squared $\sum_w^{-1}$-norm, $\varepsilon^T \sum_w^{-1} \varepsilon$, where $\varepsilon = w - \hat{w}$ and $\sum_w$ is the covariance matrix of the uncertainty of the measure vector w. This is done by solving the following equation iteratively to get the difference δ_v:

$$\left(J^T \sum_w^{-1} J + \mu I\right)\delta_v = J^T \sum_w^{-1} \varepsilon \tag{79}$$

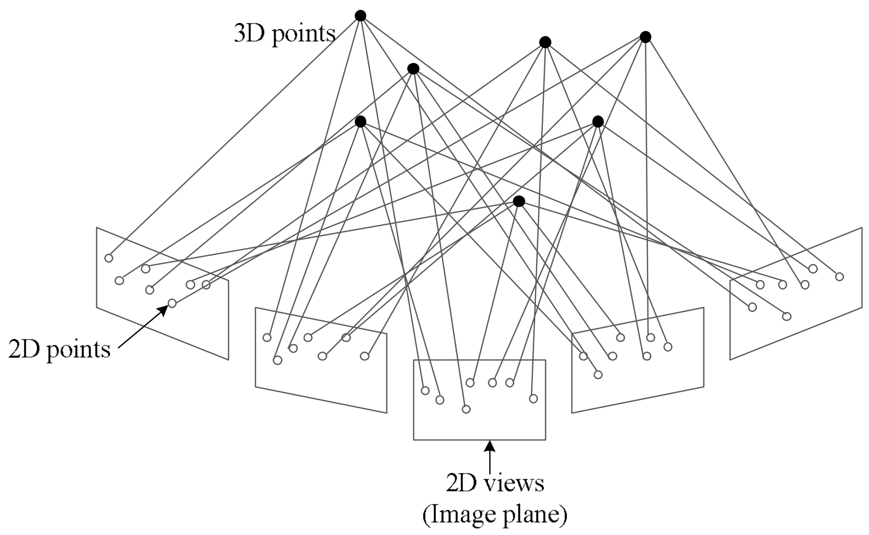


Fig. 9. Illustration of bundle adjustment. The rays back-projected from the corresponding 2D feature points in different views to a single 3D point, constitute a bundle.

where J is the Jacobian matrix of f, μ is the damping term that assures a reduction in the error in iterations, I is the identity matrix, and δ_v is the difference of the estimated parameter vector from the previous parameter vector v. After solving for δ_v, the optimal parameter vector is $v^+ = v + \delta_v$.

According to [35], in order to use the LM algorithm for bundle adjustment, the initial measurement vector w consists of the observed common 2D feature points in all views, the initial parameter vector v is defined by all the parameters of the projection matrices in all views and by the 3D points, and the functional relation f can be calculated using the projection relationship between the corresponding 2D and 3D points. The measurement vector w can be represented as

$$w = \left(x_{11}^T, \ldots, x_{1m}^T, x_{21}^T, \ldots, x_{2m}^T, \ldots, x_{n1}^T, \ldots, x_{nm}^T\right)^T \tag{80}$$

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 represented as

$$v = \left(p_1^T, \ldots, p_m^T, X_1^T, \ldots, X_n^T\right)^T \tag{81}$$

where 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.

The difference parameter vector δ_v is solved iteratively according to (79). The modified parameters of the projection matrices in all views and the 3D points in space can be calculated by adding δ_v to the original v.

4.5.2 Implementation notes for bundle adjustment

The function 'bundleadjustment' is used to implement the bundle adjustment for the refinement of the projective reconstruction so that the reprojection distance function (78) is minimized using the 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 provided by 'bundleadjustment' consisting of the refined projection matrices of all views and the coordinates of the 3D points.

The projection matrix for the middle camera P_4 is set to $\left[I_{3\times3} \mid 0_{3\times1}\right]$ in the RANSAC procedure. In the bundle adjustment procedure, the projection matrix of the middle view needs also to be fixed so that the coordinates of the middle camera will remain the same as the world coordinates. In our MATLAB implementation, the function ‘bundleadjustment’ provides an option to fix a certain number of the projection matrices starting from the first input projection matrices. Thus, instead of using the projection matrix and 3D point sets ordered from the first view to the eighth view, it is better to reorder P_4, X_4 and x_4 of the middle view to the first place and use the reordered projection matrices, 2D points and 3D points as the input of the bundle adjustment function. After implementing the bundle adjustment, the output projection matrices P_{BA} and 3D points X_{BA} are reordered back to the original order.

With P_{BA} and X_{BA}, in order to check the difference between x_{BA} and the original 2D point x, the 3D points X_{BA} are reprojected to the 2D multiple views as follows:

$$x_{BA} = P_{BA} X_{BA} \tag{82}$$

After bundle adjustment, the average deviation of the resulting x_{BA} from the original x in all views can be calculated. If the average deviation is larger than a threshold, the reconstructed 3D scene is not accurate enough and contains a lot of noise. The program needs to go back to RANSAC and implement the triangulation and bundle adjustment again. This is because in RANSAC, the 8 points are randomly chosen to compute the fundamental matrix F. If the average deviation of all views is less than a threshold, the next step called metric upgrade is implemented. The threshold was adjusted for different image sets based on the

Table 2. Average reprojection error threshold for different image sets.

Image set	Microsoft	Building	Temple
t_avg	0.3	0.24	0.1

number of iterations needed to run the algorithm and to enforce convergence within a relative range. The values of the threshold for the average deviation are different for various image sets. They are listed in Table 2 for three sample image sets (refer to Chapter 5, Section 5.1, for a description of these image sets).

The bundle adjustment toolbox is called Vincent toolbox, provided by Vincent Rabaud from UCSD. It can be downloaded from [38].

4.6 Metric Upgrade

4.6.1 Description of metric upgrade

After the bundle adjustment, the 3D model can be reconstructed but, at this point, the reconstruction is done using a projective transformation, which is not sufficient to represent the proper structure of the scene. Therefore, a method to upgrade the projective reconstruction to a metric one is implemented. Auto-calibration is a process of determining internal camera parameters and metric reconstruction directly from multiple uncalibrated scenes. Unlike other calibration methods which either depend on knowing the image of a calibration grid or on the known properties of the scene such as vanishing points, auto-calibration can be implemented directly by imposing constraints on the internal camera parameters and the external projection parameters.

In the implementation of metric upgrade, the goal is to find a rectifying homography H that transforms the projective reconstruction $\{P_{proj}, X_{proj}\}$ to the metric reconstruction as $\{P_{metric} = P_{proj}H^{-1}, X_{metric} = HX_{proj}\}$. From [21], the homography H for the metric transformation is described as,

$$H = \begin{bmatrix} K & 0 \\ -p^T K & 1 \end{bmatrix}^{-1} \tag{83}$$

where K is the intrinsic camera matrix, and the coordinates of the plane at infinity in the projective reconstruction are represented as $\pi_\infty = (p^T, 1)^T$.

From Section 3.3, in order to find H, there is a need to find the homography matrix H that would transform Q_∞^* to its canonical form as discussed in Section 3.3.2. The dual image of the absolute conic (DIAC) is the projected image of the dual absolute quadric Q_∞^* under a certain projection P, as presented in (45). The dual absolute quadric Q_∞^* in projective space can be transformed to its metric canonical form $\hat{I}_{4\times4}$ by the homography H, as in (43).

Furthermore, the DIAC ω^* is an entity in the image plane of the camera that depends only on the intrinsic camera parameters as in (46). By substituting the camera matrix K into (46), the DIAC ω^* can be represented as

$$\omega^* = \begin{bmatrix} f^2 + s^2 + u^2 & s\alpha f + uv & u \\ s\alpha f + uv & (\alpha f)^2 + v^2 & v \\ u & v & 1 \end{bmatrix} \tag{84}$$

The idea of auto-calibration is to use (84) to transfer the constraint on the intrinsic camera matrix to a constraint on the dual absolute quadric Q_∞^* under the projection matrix P, and solve the homography matrix H after the estimation of Q_∞^*. The dual absolute quadric Q_∞^* is calculated through a polynomial minimization using the Linear Matrix Inequality (LMI) relaxations [39]. The constrained optimization problem can be stated as follows,

$$\min \quad f(x) \tag{85}$$

$$\text{subject to} \quad g_i(x) \geq 0, \quad i = 1,2,\ldots,M \tag{86}$$

The LMI relaxations are computed by adding lifting variables and constraints to linearize the monomials in (85) and (86) up to a degree. In the LMI relaxation, if the degree of the monomials are up to 2δ, the order of the LMI relaxation is referred to be δ.

The constraints on the camera matrix can be expressed using the coefficients of Q_∞^* and serve as the objective function $f(x)$. The constraints $g_i(x)$ are composed of several constraints on the Q_∞^*. [23]

The objective function for minimization is represented as

$$f(Q_\infty^*) = \sum_i \left(\omega_{11}^{*i} - \omega_{22}^{*i}\right)^2 + \left(\omega_{12}^{*i}\right)^2 + \left(\omega_{13}^{*i}\right)^2 + \left(\omega_{23}^{*i}\right)^2 \tag{87}$$

where $\omega_{jk}^{*i} = p_j^i Q_\infty^* p_k^{i\,T}$ and p_k^i is the k^{th} row of the i^{th} camera projection matrix. For determining the objective function, it is assumed that the skew factor is zero ($s = 0$), the focal lengths in the x and y direction are equal ($\alpha = 1$), and the principal point lies at the left-bottom origin of the image plane ($u = v = 0$), so that the DIAC ω^* can be represented by $diag(f^2, f^2, 1)$. The objective function in (87) enforces these conditions on the DIAC ω^*.

The polynomial minimization is subject to the following constraints:

(i) Q_∞^* has to be positive semi-definite (PSD) so that ω^* is PSD and, hence, it can be decomposed into KK^T. Positive semi-definite means that all the elements in the matrix Q_∞^* are greater than or equal to zero. This constraint is fulfilled if all of the principal minors of Q_∞^* are positive.

(ii) To ensure that Q_∞^* is rank deficient, the determinant of Q_∞^* is set to zero.

(iii) To fix the scale of Q_∞^*, the Frobenius norm of Q_∞^* is set to 1.

An equivalent mathematical representation of the constrained polynomial minimization is as follows:

Objective function: min $f(Q_\infty^*) = \sum_i (\omega_{11}^{*i} - \omega_{22}^{*i})^2 + (\omega_{12}^{*i})^2 + (\omega_{13}^{*i})^2 + (\omega_{23}^{*i})^2$ (88)

Subjected to: $\det(Q_\infty^*) = 0$ (89)

$$[Q_\infty^*]_{jk} \geq 0, \quad j = 1,2,3 \text{ and } k = 1,2,\ldots,\binom{4}{j} \qquad (90)$$

$$\|Q_\infty^*\|_F^2 = 1 \qquad (91)$$

where $\omega_{jk}^{*i} = p_j^{iT} Q_\infty^* p_k^i$, p_k^i is the k^{th} column of the i^{th} camera projection matrix, $\det(Q_\infty^*)$ is the determinant of Q_∞^*, $[Q_\infty^*]_{jk}$ is a principal minor of Q_∞^*, and $\|Q_\infty^*\|_F^2$ is the Frobenius norm of Q_∞^*. The principal minor $[Q_\infty^*]_{jk}$ is the determinant of the k^{th} $(k = 1,2,\ldots,\binom{4}{j})$ sub-matrix of Q_∞^*. This sub-matrix is obtained by removing a number of j rows and j columns with the same index numbers from Q_∞^*. For the 4×4 matrix Q_∞^*, there are $\begin{pmatrix} 4 \\ j \end{pmatrix}$ possible sub-matrices that can be formed in this way. The Frobenius norm of a matrix is the square root of the square sum of all entries in the matrix.

After solving for Q_∞^* through the LMI relaxation, the SVD of Q_∞^* is computed to derive the homography H as explained in details later in Section 4.6.2. The 3D feature points X_metric and projection matrices P_metric under metric transformation are represented as:

$$X_{metric} = HX_{proj} \tag{92}$$

$$P_{metric} = P_{proj}H^{-1} \tag{93}$$

The relevant sparse depth information of the scene can be obtained from the 3D points X_{metric} in (92).

4.6.2 Implementation notes for metric upgrade

The first step in metric upgrade is to calculate the dual absolute quadric Q_∞^*. Q_∞^* is represented using the symbolic parameters in the MATLAB symbolic toolbox. As Q_∞^* is a 4×4 symmetric matrix, 10 symbolic parameters are used to represent Q_∞^*. In the LMI relaxation, the objective function $f\left(Q_\infty^*\right)$ and the constraint equations can be expressed in terms of the symbolic parameters of Q_∞^*. The GloptiPoly toolbox and SeDuMi toolbox are used here to calculate the optimal solution of Q_∞^* that minimizes the objective function $f\left(Q_\infty^*\right)$.

The GloptiPoly is a MATLAB toolbox that helps in solving the linear matrix inequality (LMI) relaxations of global optimization problems. It also makes use of the SeDuMi toolbox for LMI relaxations of non-convex optimization problems. Using the function 'defipoly', a cell array is obtained and used to store the minimization function and the constraint equations, as described in Section 4.6.1. The first element of the cell array is the objective function to minimize. The function 'gloptipoly' is used for the global optimization. The cell array containing the objective function and constraint equations serves as the input to this function, and the order of the LMI relaxation is set to 2 [23]. If an optimum solution exists for minimizing the objective function $f(Q_\infty^*)$, the obtained values of the symbolic parameters of Q_∞^* are stored in the output.

After solving for the dual absolute quadric Q_∞^* through global optimization, the homography H is calculated using the SVD according to (43). The detailed procedure for solving H is described below.

Using the SVD, Q_∞^* can be decomposed as

$$Q_\infty^* = UDV^T \tag{94}$$

where U and V are unitary matrices and D is a diagonal matrix with nonnegative real numbers (singular values) on the diagonal. Since Q_∞^* is a symmetric matrix, in the SVD of Q_∞^* given by (94),

$$U = V \tag{95}$$

The diagonal entries d_{ii} of D are arranged in a descending order (d_{44} is nearly 0). D can be represented as follows:

$$D = \begin{bmatrix} d_{11} & 0 & 0 & 0 \\ 0 & d_{22} & 0 & 0 \\ 0 & 0 & d_{33} & 0 \\ 0 & 0 & 0 & d_{44} \end{bmatrix} \tag{96}$$

where $d_{44} \approx 0$. As D is a diagonal matrix, it can be further decomposed as

$$D = D_{sqrt} D_{sqrt} = \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & \sqrt{d_{44}} \end{bmatrix} \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & \sqrt{d_{44}} \end{bmatrix} \tag{97}$$

and (94) can be expressed as

$$Q_{\infty}^{*} = UD_{sqrt} D_{sqrt} V = UD_{sqrt} I_{4\times4} D_{sqrt} V \tag{98}$$

where $I_{4\times4}$ is a 4×4 identity matrix. Since $d_{44} \approx 0$, $D_{sqrt} I_{4\times4} D_{sqrt}$ can be approximated as

$$D_{sqrt} I_{4\times4} D_{sqrt} \approx \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{99}$$

and (99) can be rewritten as

$$D_{sqrt} I_{4\times4} D_{sqrt} \approx \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{100}$$

Let

$$\hat{D}_{sqrt} = \begin{bmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{22}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{101}$$

and

$$\hat{I}_{4\times4} = diag(1,\ 1,\ 1,\ 0) \tag{102}$$

it follows that

$$D_{sqrt}\ I_{4\times4} D_{sqrt} \quad = \hat{D}_{sqrt}\ \hat{I}_{4\times4} \hat{D}_{sqrt} \tag{103}$$

Using (103), Q_∞^* in (98) can be written as

$$Q_\infty^* = U\hat{D}_{sqrt}\ \hat{I}_{4\times4} \hat{D}_{sqrt}\ V \tag{104}$$

By multiplying $\left(U\hat{D}_{sqrt}\right)^{-1}$ to the left of Q_∞^* in (104) and $\left(\hat{D}_{sqrt}V\right)^{-1}$ to the right, it follows that

$$\hat{I}_{4\times4} = \left(U\hat{D}_{sqrt}\ \right)^{-1} Q_\infty^* \left(\hat{D}_{sqrt}\ V\right)^{-1} \tag{105}$$

From (43), (95) and (105), the homography matrix H is expressed as

$$H = \left(U\hat{D}_{sqrt}\ \right)^{-1} \tag{106}$$

Finally, knowing the homography H, the coordinates of the 3D points and projection matrices in metric reconstruction X_{metric} and P_{metric} are transformed according to (92) and (93), respectively.

The GloptiPoly toolbox is provided by D. Henrion. It can be downloaded from [40]. The SeDuMi toolbox is provided by Lehigh University at the link [41].

4.7 Transformation of 3D Points in Metric Reconstruction

After transforming the projective reconstruction by the homography H, the projection matrix of the middle camera is no longer $[I_{3\times 3} \mid 0_{3\times 1}]$, as it has been transformed by a translation vector t_H and rotated by a rotation matrix R_H. The coordinates of the 3D points are represented with respect to the world coordinates, but they are not consistent with the transformed coordinates of the middle camera under the metric reconstruction.

An example of the transformation geometry between the world coordinates and the middle camera coordinates is illustrated in Fig. 10. Due to this, the coordinates of the 3D points may not be lying on the same side with respect to the middle camera center in the metric transformation. It results in depth values, for the 3D feature points, that are either positive or negative with respect to the coordinates of the middle camera.

To solve this problem, the coordinates of the 3D points in the metric frame need to be transformed by the rotation matrix R_H and translation vector t_H in order to be consistent with respect to the middle camera. R_H and t_H can be computed from the decomposition of the projection matrix P_{4_metric} for the middle view in metric transformation as follows. From (7) and (8), the homogenous representation of the projection matrix P can be expressed as

$$P_{4\_metric} = K[R \mid t] = [M \mid Kt] \tag{107}$$

where $M = KR$ is a 3×3 matrix, and K is the intrinsic camera matrix and is

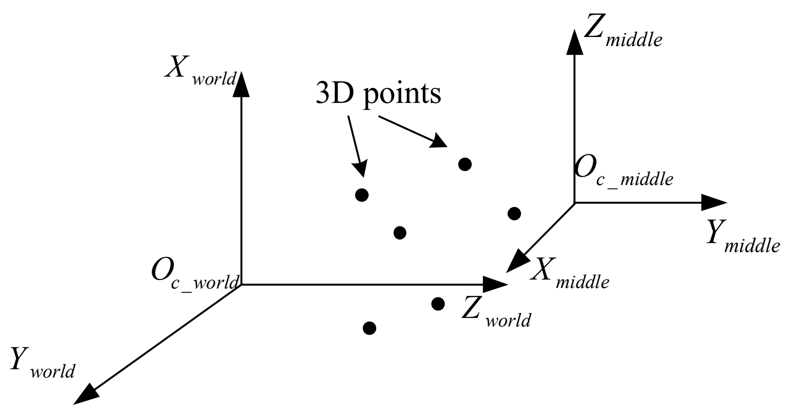


Fig. 10. Illustration of 3D points with respect to the world coordinates and the middle camera coordinates under metric transformation frame.

an upper-triangular matrix, R is the rotation matrix and t is the translation vector. As $M = KR$ and K is an upper-triangular matrix, the RQ decomposition is used to calculate K and R. The RQ decomposition uses the Givens rotation matrices to calculate the upper-triangular matrix K. The Givens rotations have three types as

$$G_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \tag{108}$$

$$G_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{109}$$

$$G_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{110}$$

Through multiplying M by G_x on the right of M, the first column of M will remain the same. Similarly, the second and third column of M will remain the same when M is multiplied with G_y and G_z on the right of M.

The first step of the RQ decomposition is to multiply M on the right by G_x to generate M^x and to set m_{32}^x to zero, where m_{32}^x is the element of M^x located on the third row and the second column of M^x. The rotation angle θ_x in G_x is calculated using the following equations as

$$\begin{cases} m_{32}\cos\theta_x + m_{33}\sin\theta_x = 0 \\ \cos^2\theta_x + \sin^2\theta_x = 1 \end{cases} \tag{111}$$

where m_{ij} is the element of M at the i^{th} row and j^{th} column of M.

so that

$$\cos\theta_x = \frac{-m_{33}}{\sqrt{m_{32}^2 + m_{33}^2}} \tag{112}$$

$$\sin\theta_x = \frac{-m_{32}}{\sqrt{m_{32}^2 + m_{33}^2}} \tag{113}$$

and then G_x is computed using (108), (112) and (113).

After setting m_{32}^x to zero, M^x is multiplied with G_y on the right to generate $M^{x,y}$ and $m_{31}^{x,y}$ is set to zero. G_y is computed in a similar way to G_x. Finally, $M^{x,y}$ is multiplied with G_z on the right to generate $M^{x,y,z}$ and $m_{21}^{x,y,z}$ is set to zero. G_z is the computed similar to G_x and G_y. After these operations, an upper-triangular matrix K is formed as

$$K = M^{x,y,z} = MG_x G_y G_z \tag{114}$$

Since G_x, G_y and G_z are unitary matrices, the rotation matrix R can be represented as

$$R = G_z^T G_y^T G_x^T \tag{115}$$

The translation vector t is calculated as

$$t = K^{-1} p_4 \tag{116}$$

where p_4 is the fourth column of the projection matrix P_{4_metric}.

In this work, the rotation matrix and translation vector were obtained from the homography matrix H using the function 'vgg_KR_from_P' that is included in the triangulation package introduced in Section 4.4.2. The resulting transformation matrix T is expressed as

$$T = \begin{bmatrix} R_H & t_H \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{117}$$

Multiplying the 3D point after metric upgrade by T will transform the coordinates of the 3D points to make them consistent with the middle camera coordinates in the metric frame, that is

$$X_{metric\_trans} = TX_{metric} \tag{118}$$

The sparse depth values for common feature points among all the views are given by the third elements of the 3D point vectors X_{metric_trans}.

4.8 Important Notes

In the implementation of metric upgrade, due to the limit of the GloptiPoly toolbox, using the original projection matrices P for the optimization computation are not proper because the entries of P are too large for computations in the GloptiPoly toolbox. A pre-processing is implemented to normalize the P matrices by an estimated camera matrix K_{estm} at the beginning of the 3D modeling as

$$P_{norm} = K_{estm}^{-1} P \tag{119}$$

In our implementation, K_{estm}^{-1} is set as:

$$K_{estm}^{-1} = \begin{bmatrix} 1020 & 0 & 0.5\times w \\ 0 & 1020 & 0.5\times h \\ 0 & 0 & 1 \end{bmatrix} \tag{120}$$

where w refers to the width of input images and h is the height of the input images. Due to the fact that $x = PX$ normalization of the projection matrix is equivalent to normalizing the 2D points x at the beginning of the implementation by multiplying them with the inverse of K_{estm} on the left after the SIFT, as follows:

$$x_{norm} = K_{estm}^{-1}\, x \tag{121}$$

In the following steps normalization, the projection matrices that are calculated from RANSAC will be in the normalized form P_{norm}.

5. EXPERIMENTAL RESULTS

In this chapter, the experimental results of the proposed 3D reconstruction model using different images sets are presented and analyzed. Section 5.1 introduces the image sets used to evaluate the performance of the 2D to 3D conversion system. Section 5.2 illustrates the sparse depth maps for different image sets. Section 5.3 presents a performance analysis of the scale factor in the implemented system.

5.1 Data Set Description

In the implementation of the multi-view 3D reconstruction, four different image sets for 3D reconstruction are used here: Egypt_Temple, Tempe_Building, Microsoft_Ballet and Table. The Egypt_Temple image set is taken by a hand-held Canon digital camera in Egypt. The Tempe_Building image set is taken from a Remote Control Airplane (RCA), and it is the birds-eye view of the buildings in downtown Tempe, Arizona. The Microsoft_Ballet image set is downloaded from Microsoft's website [42]. The Table image set was taken using a hand-held Canon digital camera for objects set up on a table. In addition, the ground-truth depth data for the Table image set was collected by measuring the distance of each object from the camera. This ground-truth data is used to further test the performance of the system. The 8 views of the four image sets are shown in Fig. 11, Fig. 12, Fig. 13 and Fig. 14, respectively. The top four views are View 1 to View 4 from left to right. The bottom four views are View 5 to View 8 from left to right. Although multiple views can be obtained using multiple cameras fixed at different viewpoints, the 8 views of the above image sets are taken with the same

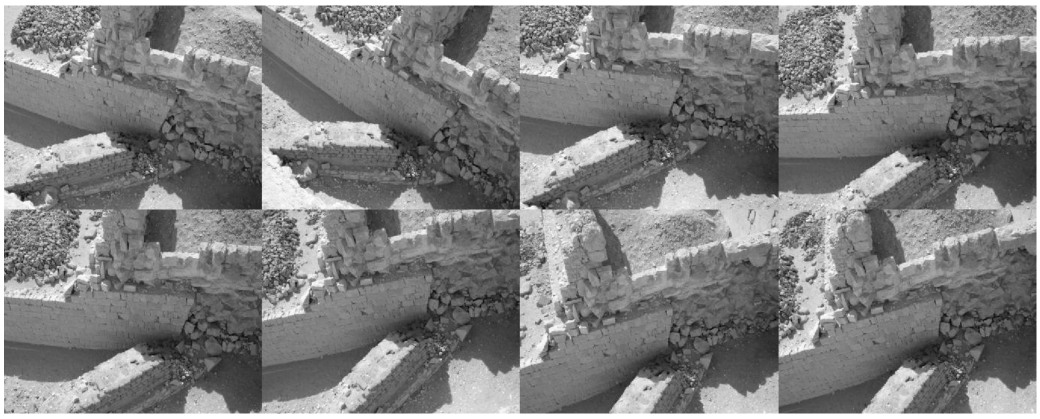


Fig. 11. Eight views of Egypt_Temple (size of each view is 512×384). The top four views are View 1 to View 4 from left to right, and the bottom four views are View 5 to View 8 from left to right.

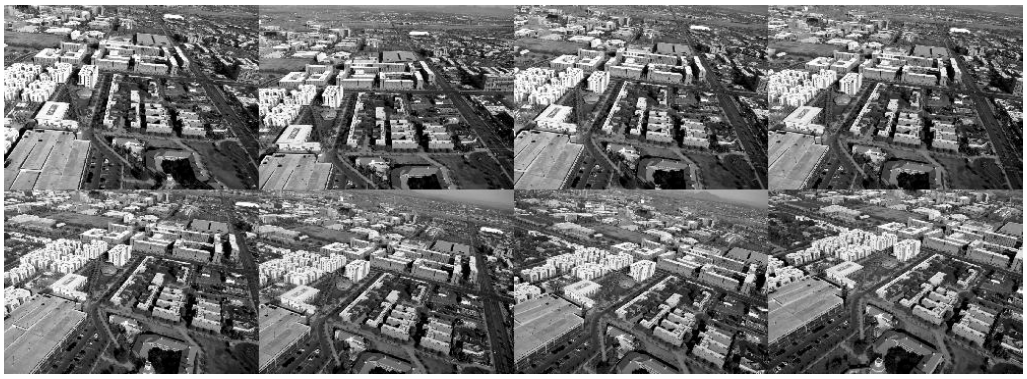


Fig. 12. Eight views of Tempe_Building (size of each view is 912×684). The top four views are View 1 to View 4 from left to right, and the bottom four views are View 5 to View 8 from left to right.

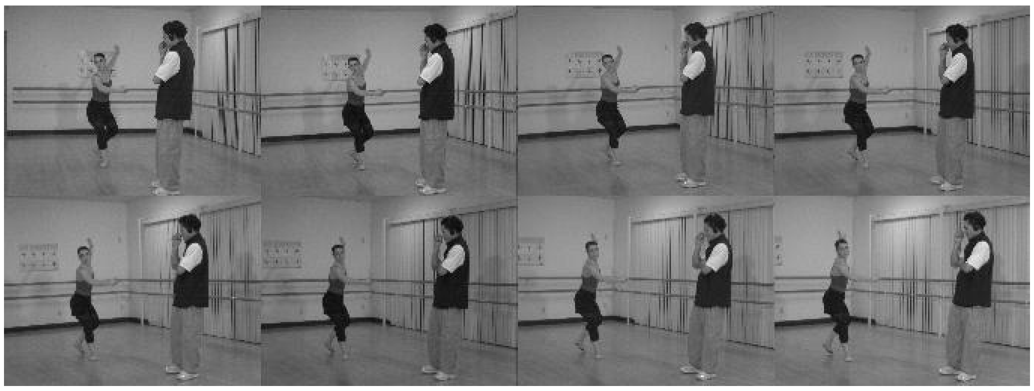


Fig. 13. Eight views of Microsoft_Ballet (size of each view is 256×192). The top four views are View 1 to View 4 from left to right, and the bottom four views are View 5 to View 8 from left to right.

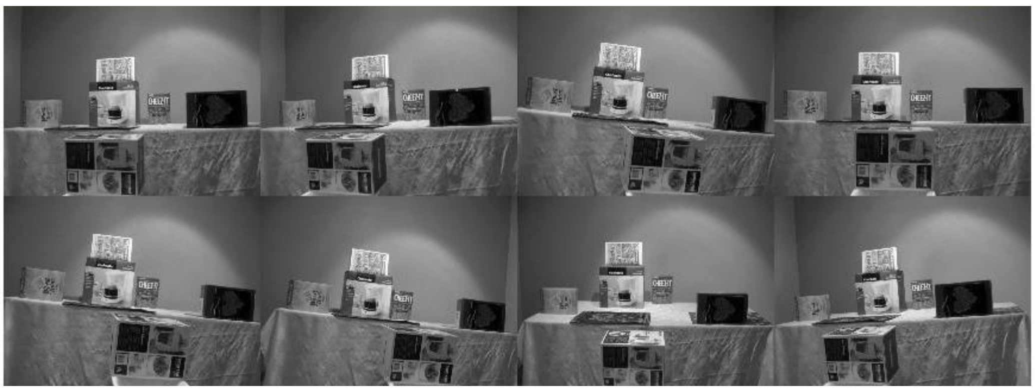


Fig. 14. Eight views of Table (size of each view is 648×486). The top four views are View 1 to View 4 from left to right, and the bottom four views are View 5 to View 8 from left to right.

camera from different angles with respect to the same scene. The MATLAB environment is used to produce the program for the implementation.

The multi-view image set is read in and the intensities for the image pixels are converted to the double data type for further computation. For the Tempe_Building, Egypt_Temple and Table image sets, as the image sizes are too large for MATLAB to process the image data, bilinear down-sampling is used to resize the image to 25% of their original widths and heights. The multi-view frames are stored in a three-dimensional matrix, whose third dimension indicates the view number.

5.2 Sparse Depth Map Results

5.2.1 Results for the Egypt_Temple image set

For the Egypt_Temple image set, the matching feature points between the middle view (View 4) and View 1 after SIFT is shown in Fig. 15. There are 1807 matching points. Due to the large number of matching points, only 1/40 of all the matching points are shown in Fig. 15 for illustration, and are marked and connected with each other to show the matching clearly. The stars in the images indicate the locations of the 2D feature points in each view, and the lines connecting two corresponding 2D points illustrate the matching between two views. As the matching points that are calculated by SIFT only depend on the descriptors of the keypoints in the local image region, and are not related by the epipolar geometry between the considered views, there may be some mis-matched feature points between the two views. This is illustrated in Fig. 15. There are two mis-matched connections that can be seen in Fig. 15.

The matching inliers between the middle view (View 4) and View 1 after RANSAC are shown in Fig. 16. Here, only 1/40 of all the inlier matching points are marked and connected in Fig. 16 to illustrate the matching clearly. Comparing Fig. 15 and Fig. 16, the two mis-matched crossing lines are removed. RANSAC helps to remove the outliers, and the corresponding feature points are related by both the feature descriptors and the geometry. The number of inliers is 1771. It is 98% of the corresponding feature points after SIFT.

After feature matching between the middle view and all the other views, the common feature points among all 8 views are searched and are shown in Fig. 17. The five views in the upper row are View 1 to View 5 from left to right, and the four views in bottom row are View 5 to View 8 from left to right. Here, only 1/40 of all the matching points are marked and connected to illustrate the matching clearly. We can see that the feature points in different views match with each other correctly. There are totally 400 common feature points among the 8 views in the Egypt_Temple image set.

Finally, the depth values of sample common feature points are plotted on the middle view as shown in Fig. 18. Note that the calculated depth values in the image are actually fractional values and are rounded down to the nearest integer for illustration purpose. A larger depth value means that the point is farther from the camera and a smaller depth value indicates that the point is closer to the camera. As the picture is captured from the right side of the scene, the depth value

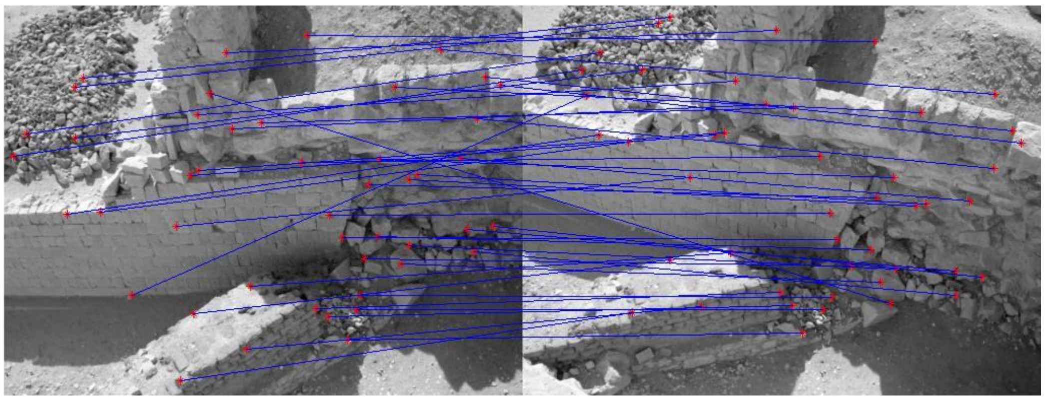


Fig. 15. Plot of 1/40 of the total number of matching feature points after the SIFT between View 4 (left) and View 1 (right) in Egypt_Temple.

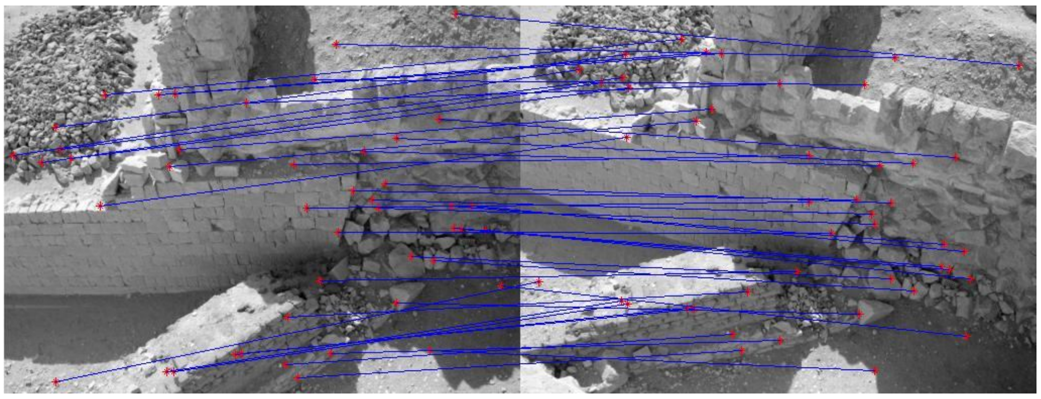


Fig. 16. Plot of 1/40 of the total number of matching inliers after RANSAC between View 4 (left) and View 1 (right) in Egypt_Temple.

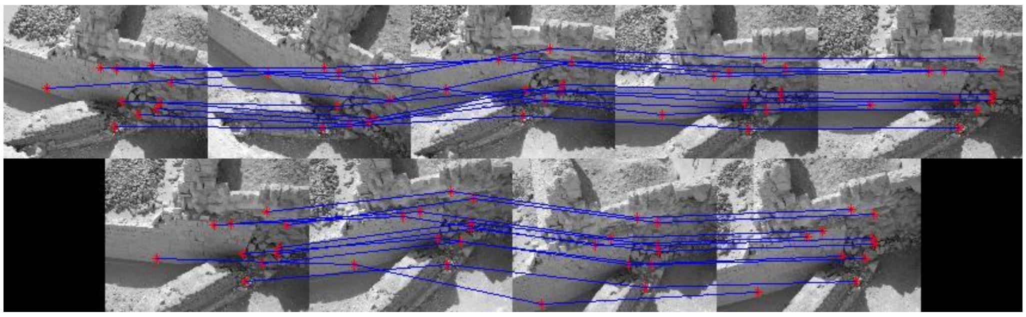


Fig. 17. Plot of 1/40 of the total number of common feature points in all views of Egypt_Temple. Five views on the top are View 1 to View 5 from left to right. Four views on the bottom are View 5 to View 8 from left to right.

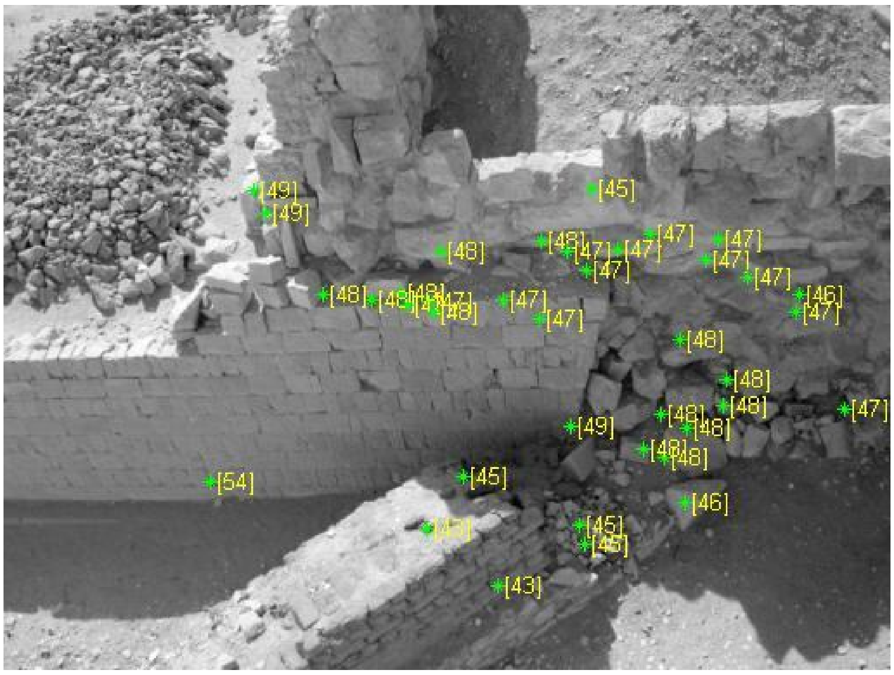


Fig. 18. Plot of 1/40 of the total number of depth values of feature points on the middle view (View 4) of Egypt_Temple. The depth values are rounded down to the nearest integers.

gets larger by observing the feature points from right to the left in the horizontal direction.

5.2.2 Results for the Microsoft_Ballet image set

After implementing the system on the Microsoft_Ballet image set, the matching points between the middle view (View 4) and View 1 after SIFT are shown in Fig. 19. There are 414 matching points, but only 1/10 of all the matching points are marked and connected in the figure to illustrate the matching clearly. As the results show, there are some mismatching points.

The matching inliers between the middle view and View 1 after RANSAC are shown in Fig. 20. Compared to the matching using SIFT in Fig. 19, it is clear that RANSAC helps in removing the outliers. The number of inliers is 216, which is 52% of the corresponding points after SIFT.

The common feature points among all eight views are shown in Fig. 21. There are only 11 common feature points among 8 views in Microsoft_Ballet. This is because there are not many different structures in the multiple views of this image set. The plot of depth values for feature points on the middle view is shown in Fig. 22. As before, larger depth values indicate being farther from the camera and vice versa. As the number of common feature points is not large enough, the estimated depth value can be inaccurate in different trials. One example of inaccurate depth calculation that is obtained from one run is shown in Fig. 23.

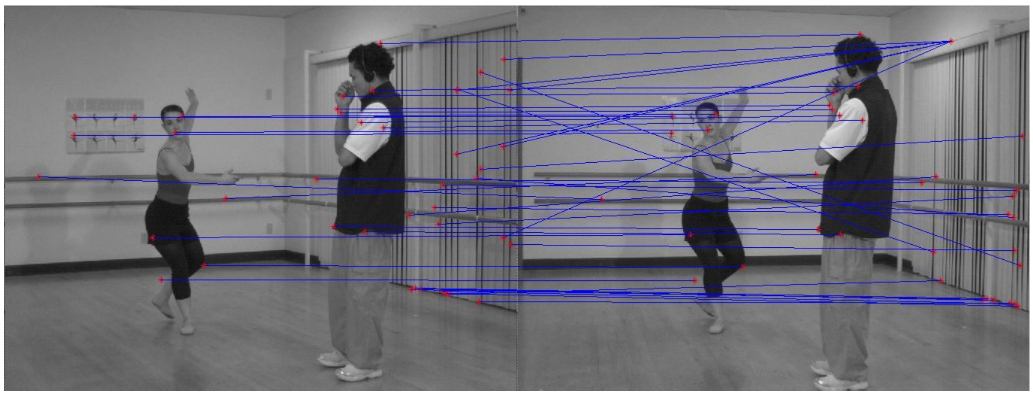


Fig. 19. Plot of 1/10 of the total number of matching feature points after the SIFT between View 4 (left) and View 1 (right) in Microsoft_Ballet.

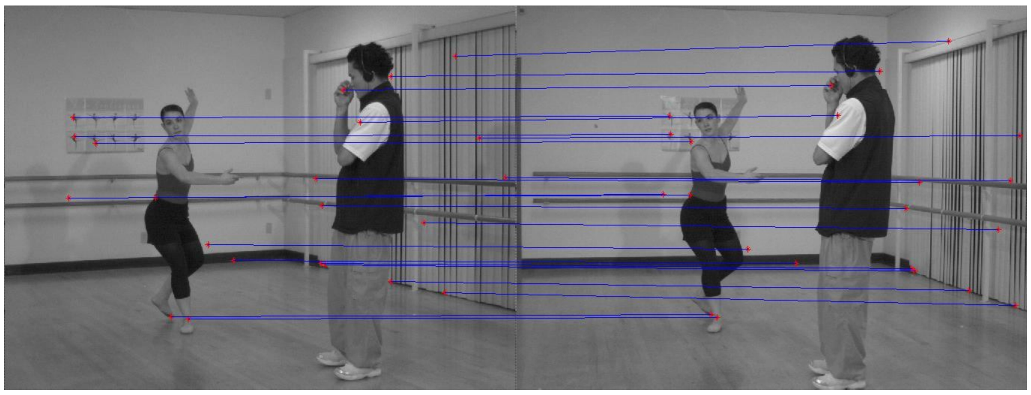


Fig. 20. Plot of 1/10 of the total number of matching feature points after RANSAC between View 4 (left) and View 1 (right) in Microsoft_Ballet.

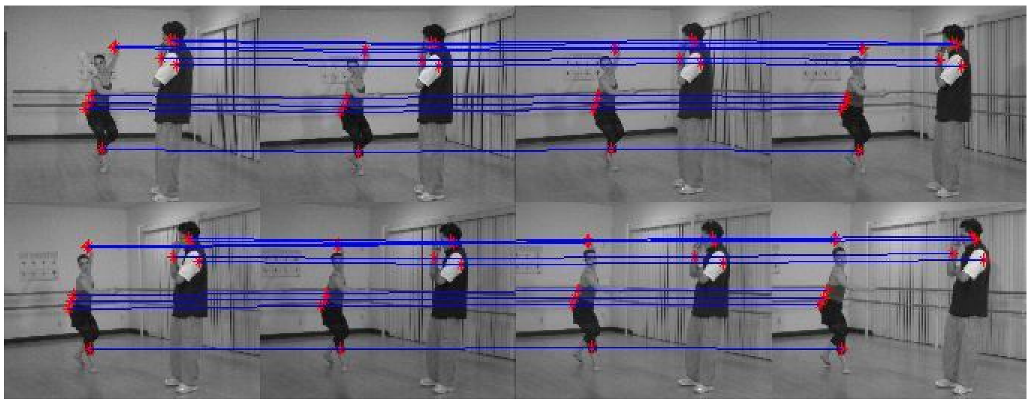


Fig. 21. Plot of common feature points in all views of Microsoft_Ballet. Four views on the top are View 1 to View 4 from left to right. Four views on the bottom are View 5 to View 8 from left to right.

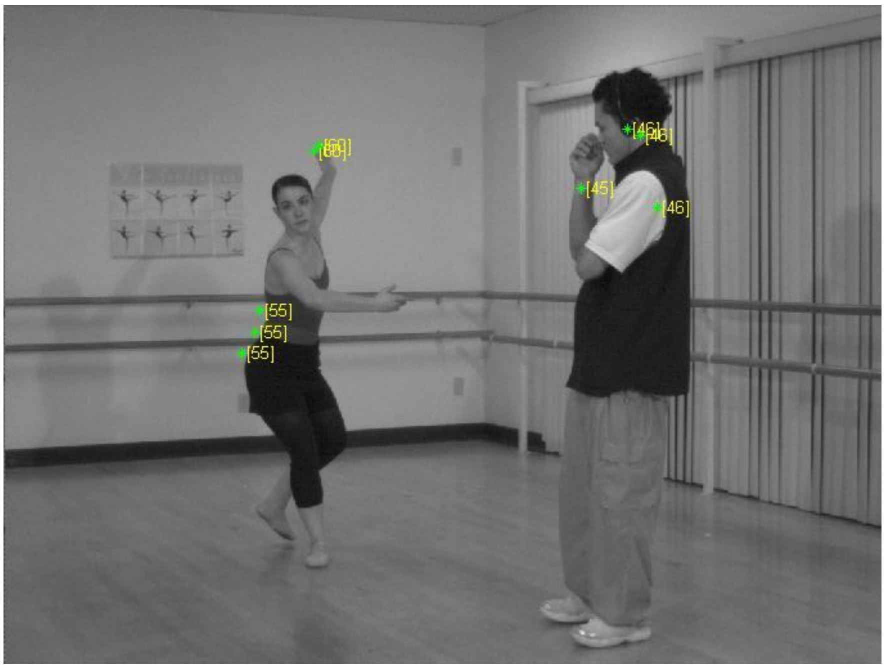


Fig. 22. Plot of 1/40 of the total number of depth values of feature points in the middle view (View 4) of Microsoft_Ballet.

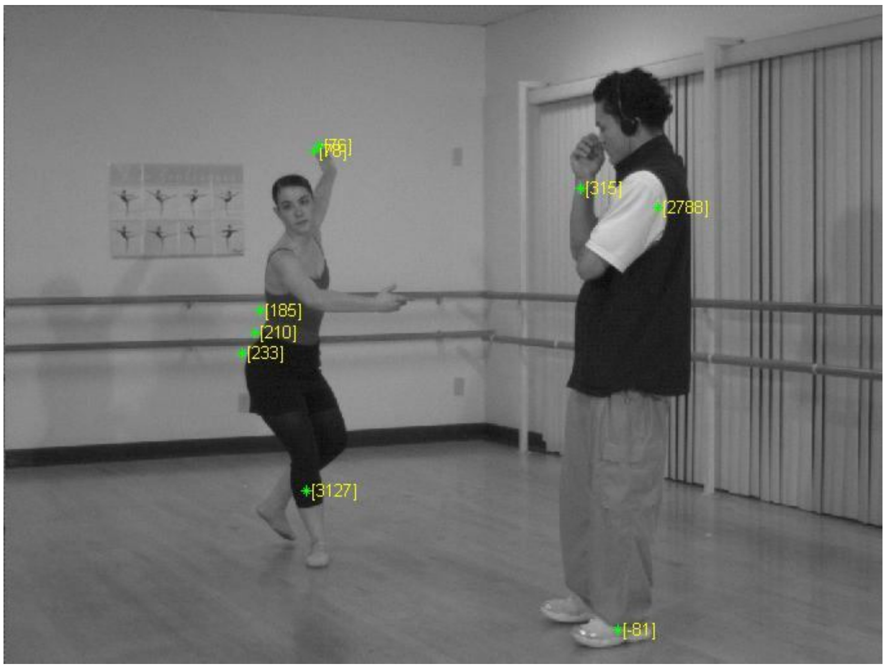


Fig. 23. An example of wrong calculated depth values plotted in the middle view (View 4) of Microsoft_Ballet.

5.2.3 Results for the Tempe_Building image set

Using the Tempe_Building image set as the input to the 3D reconstruction system, the matching points between the middle view (View 4) and View 1 after SIFT are shown in Fig. 24. There are 3219 matching points, but only 1/40 of all the matching points are shown for clarity in Fig. 24 to show the matching clearly. Also, some mis-matched feature points can be seen in Fig. 24.

The matching inliers between the middle view (View 4) and View 1 after RANSAC are shown in Fig. 25. Only 1/40 of all the inlier feature points are shown in this figure for clarity. The number of inliers is 3122, 97% of which correspond to the matching feature points after SIFT.

The common feature points among all 8 views are shown in Fig. 26. Only 1/40 of all the matching points are shown in this figure. There are totally 362 corresponding feature points among the eight views in Tempe_Building.

Finally, the depth values on the middle view are plotted as shown in Fig. 27. As before, a larger depth value indicates being farther from the camera and a smaller depth value means that the feature point is closer to the camera center.

5.2.4 Reprojection error results

Table 3 shows the number of common feature points and the average mean square reprojection error in all eight views after bundle adjustment for the three image sets. The number of common feature points in Egypt_Temple and Tempe_Building is large enough to reconstruct the 3D scene correctly, while that in Microsoft_Ballet is not sufficient to estimate the correct depth values all the time.

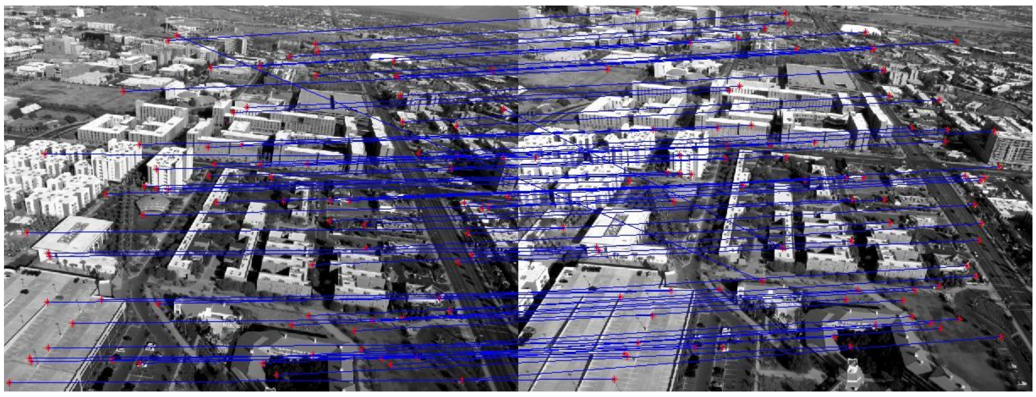


Fig. 24. Plot of 1/40 of the total number of matching feature points after the SIFT between View 4 (left) and View 1 (right) in Tempe_Building.

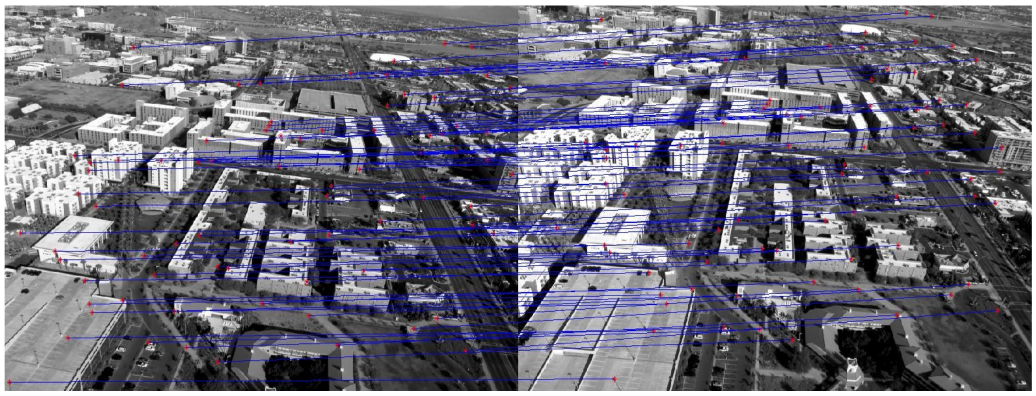


Fig. 25. Plot of 1/40 of the total number of matching inliers after RANSAC between View 4 (left) and View 1 (right) in Tempe_Building.

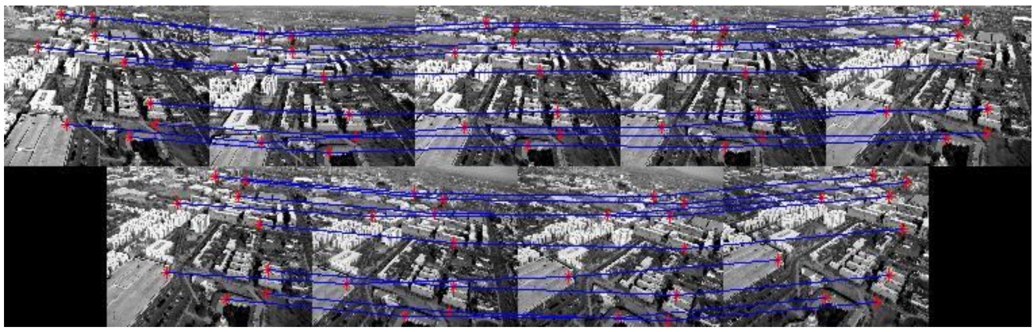


Fig. 26. Plot of 1/40 of the total number of common feature points in all views of Tempe_Building. The five views on the top are View 1 to View 5 from left to right. The four views on the bottom are View 5 to View 8 from left to right.

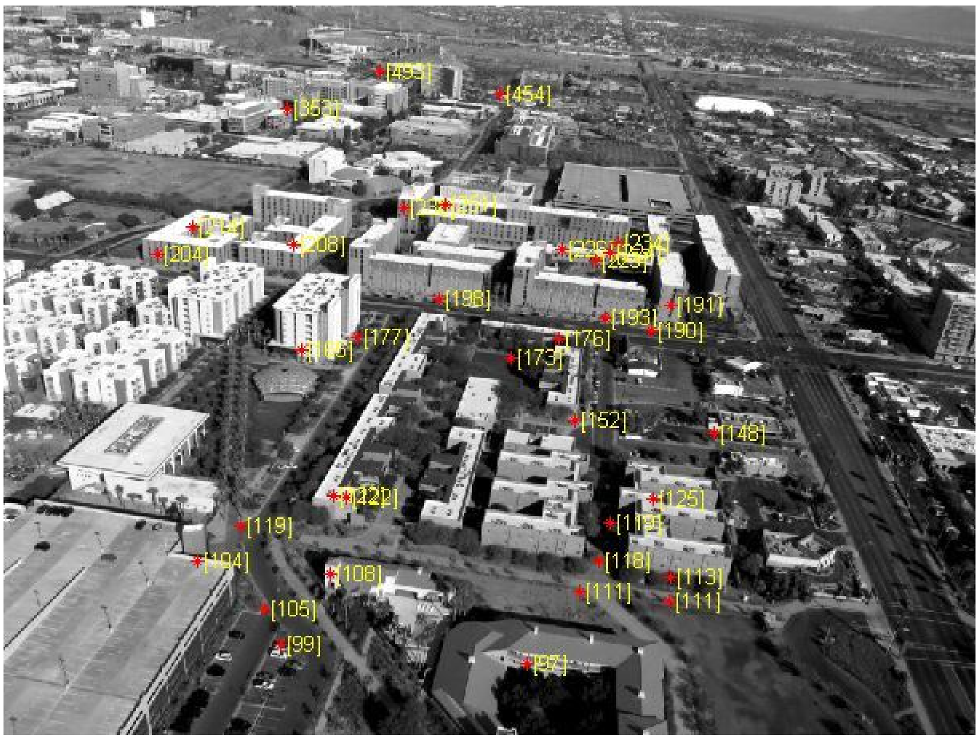


Fig. 27. Plot of 1/40 of the total number of depth values of feature points on the middle view (View 4) of Tempe_Building.

Table 3. Number of common feature points and reprojection errors for three image sets.

Image Set	No. of feature points	Average Reprojection
Egypt_Temple	400	0.0985
Microsoft_Ballet	11	0.2951
Tempe_Building	362	0.2377

5.3 Analysis of the System Performance

5.3.1 Analysis of the scale factor stability

The 3D scene modeled in this system is a metric reconstruction of the scene. Compared to the 3D scene under the Euclidean frame, the metric transformation differs from the Euclidean one by a scale factor s. Under the metric reconstruction, the distance of the objects to the camera center is scaled by s from the ground-truth data. The scale factor and the depth values of the feature points, thus, can vary with different runs of the system. The depth values generated by two different runs are shown in Fig. 28 for the Egypt_Temple image set, as an example.

Although the depth values can be different due to the randomness of the scale factor s, for all 3D points in the same run, the scale factor would be the same. That is, suppose the ground-truth depth values for two feature points are D_1 and D_2, the depth values calculated by the 2D to 3D conversion system should be sD_1 and sD_2. For any given pair of feature points, the ratio of their calculated depth values should remain constant in different runs.

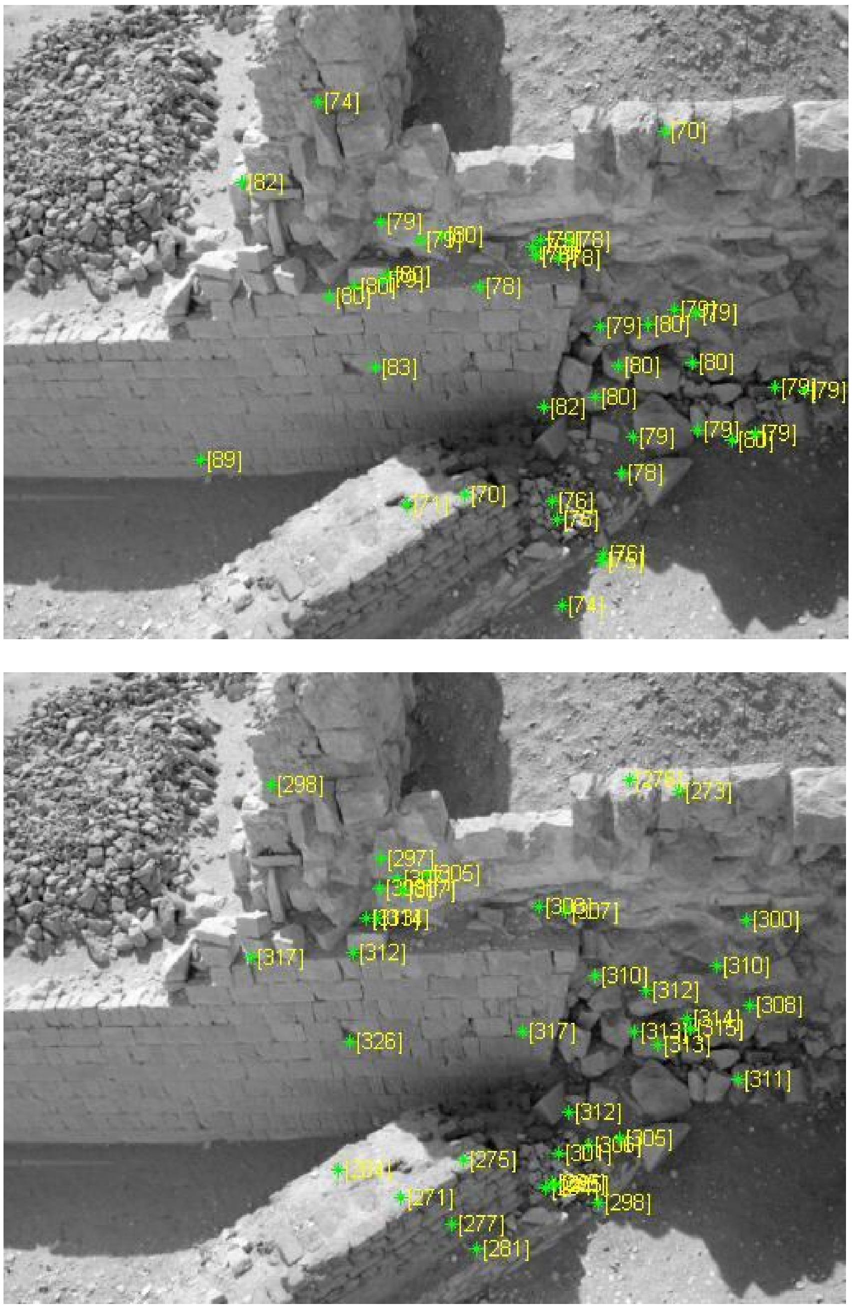


Fig. 28. Plot of the depth values for feature points on the middle view (View 4) at two different iterations for the Egypt_Temple image set.

Table 4. 2D points with minimum depth and maximum depth, and ratio of the maximum and minimum depth values in 5 iterations based on Egypt_Temple image set.

Iteration	**Max depth**	**Max depth position**	**Min depth**	**Min depth position**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$
1	41.1188	(310.47, 66.01)	31.7423	(229.636, 336.160)	1.2954
2	125.2106	(310.47, 66.01)	94.7702	(229.637, 336.159)	1.3212
3	131.8619	(310.47, 66.01)	101.5992	(229.636, 336.160)	1.2979
4	27.4511	(310.47, 66.01)	21.1687	(229.636, 336.160)	1.2968
5	61.6124	(310.47, 66.01)	46.4913	(229.644, 336.161)	1.3252

The feature points corresponding to the maximum and the minimum depth values are chosen to perform the analysis of the scale factor. Using the Egypt_Temple image set, the coordinates of the maximum-depth feature point and minimum-depth feature point in the middle view (View 4) and the ratio of their depth values for five different iterations are shown in Table 4. From the results in Table 4, it can be seen that the ratio of the maximum depth value and the minimum depth value remains stable in different runs. The standard deviation of the max depth and min depth ratio is 0.014. It can be concluded that the randomness of the scale factor in the metric transformation is the major cause of different depth values in different runs.

5.3.2 Effect of RANSAC on the scale factor stability

In the implementation of RANSAC, as the eight feature points that are used to calculate the fundamental matrix are randomly chosen from all the feature points, the two-view geometry will not remain the same at different runs. The threshold

t that is used to select the inliers of the feature points, as discussed in Section 4.3.1, may affect the calculated depth values of feature points. Using the 3D modeling system implemented in this thesis, a total of 15 runs (five for each value of t) are implemented for three different values of threshold t (t=0.01, 0.015 and 0.02). The results for the Egypt_Temple image set are similar to those in Table 4. The results based on the Tempe_Building image set are shown in Table 5. From the results in Table 5, the ratio of the maximum and the minimum depth values become less stable if the value of the threshold t increases. This is because the constraint to choose the inliers in RANSAC will become less strict when the threshold increases, and this produces more noise in the estimated depth values.

5.3.3 Effect of the reprojection error on the scale factor stability

Comparing the standard deviation of the max depth and min depth ratio in the Egypt_Temple and Tempe_Building image sets, it can be seen that the standard deviation for the Egypt_Temple image set is 0.014, and that for the Tempe_Building image set is much larger than 0.014. The threshold of the average reprojection error t_avg after bundle adjustment, as discussed in Section 4.5.2, also has an effect on the stability of the scale factor s. The threshold of the average reprojection error t_avg in Tempe_Building is much larger than that in Egypt_Temple. This is because the minimum reprojection error of the Tempe_Building image set converges to a larger value than the minimum reprojection error of the Egypt_Temple image set, due to the fact that the structure

Table 5. Statistical analysis of the depth values of the feature points in the Tempe_Building image set for 15 runs based on the average mean square reprojection error across all views.

t=0.01							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
1	58.88	294.47	121.64	47.58	5.00	(354.37,62.55)	(114.00,590.69)
2	57.42	343.58	125.64	55.18	5.98	(354.37,62.54)	(113.98,590.72)
3	22.89	128.44	49.00	20.68	5.61	(354.37,62.55)	(113.98,590.72)
4	33.82	154.84	67.50	25.00	4.58	(354.40,62.58)	(113.95,590.73)
5	40.39	168.96	78.17	27.21	4.18	(354.37,62.55)	(113.98,590.72)
ST	**15.45**	**94.91**	**33.86**	**15.26**	**0.73**		
t=0.015							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
6	16.85	88.34	35.37	14.19	5.24	(496.70,74.06)	(113.98,590.72)
7	35.13	148.55	68.25	23.92	4.23	(354.37,62.55)	(113.98,590.71)
8	25.01	150.41	54.88	24.10	6.01	(354.35,62.55)	(113.97,590.71)
9	38.72	230.77	84.76	36.66	5.96	(496.70,74.06)	(113.98,590.72)
10	37.60	215.40	81.22	34.63	5.73	(354.37,62.55)	(113.98,590.72)
STD	**9.43**	**57.47**	**20.28**	**9.12**	**0.74**		
t=0.02							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
11	97.79	629.68	219.27	100.08	6.44	(354.37,62.55)	(113.98,590.72)
12	44.35	151.44	79.65	23.21	3.41	(496.70,74.06)	(113.98,590.72)
13	180.90	668.16	332.69	106.40	3.69	(354.37,62.55)	(113.98,590.72)
14	80.78	240.58	137.06	36.22	2.98	(496.70,74.06)	(113.98,590.72)
15	50.83	209.84	98.39	33.61	4.13	(496.70,74.06)	(113.98,590.72)
STD	**54.82**	**247.99**	**103.94**	**39.92**	**1.36**		

of the Tempe_Building image set is more complicated. As the threshold of the reprojection error gets larger, the average reprojection errors in all the 2D views are larger and, thus, the accuracy of the detected 2D feature points is worse. This will produce more noise in the estimated structure of the 3D scene.

Different methods to compute the reprojection error in the 2D to 3D conversion system also affect the stability of the scale factor in the metric reconstruction. For example, if another norm other than the average MSE reprojection error among the 2D feature points in all views is used, the estimated depth values can be affected as shown in Table 6. For the results in Table 6, the reprojection error is calculated as follows. First, the average mean square reprojection error of 2D feature points is calculated for each view. Then, the final reprojection error is taken to be the maximum average mean square reprojection error among all views. It can be seen that using the latter method for computing the reprojection error, results in a maximum to minimum depth ratio that is less stable than that in Table 5.

5.3.4 Evaluation of the system performance with respect to the ground-truth depth

Another important method to analyze the performance of the implemented 3D reconstruction system is to evaluate the calculated depth values with respect to ground-truth data. As discussed in Section 5.3.1, the calculated depth values differ from the ground-truth depth values by a random scale factor s. In order to compare the results with the ground-truth depth, the computed depth values must be scaled to be within the same range as the ground-truth data. The scale factor is

Table 6. Statistical analysis of the depth values of the feature points in the Tempe_Building image set for 15 runs based on the maximum reprojection error across all views.

t=0.01							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
1	50.80	2302.73	175.69	210.53	45.33	(354.41,62.53)	(113.95,590.69)
2	48.21	1137.24	149.47	132.02	23.59	(354.45,62.52)	(113.95,590.70)
3	104.28	406.13	195.75	65.09	3.899	(354.37,62.55)	(113.98,590.72)
4	56.32	370.38	127.25	58.71	6.58	(496.70,74.06)	(113.98,590.72)
5	40.10	387.04	101.11	57.57	9.650	(496.08,74.06)	(113.95,590.70)
ST	**25.46**	**838.05**	**37.62**	**66.81**	**17.15**		
t=0.015							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
6	17.37	88.44	36.10	14.29	5.09	(354.37,62.55)	(113.98,590.72)
7	31.45	271.98	76.67	41.33	8.65	(354.41,62.53)	(113.95,590.69)
8	96.49	1044.05	250.03	150.42	10.82	(354.37,62.55)	(113.98,590.72)
9	55.34	291.46	116.30	46.95	5.27	(496.70,74.06)	(113.98,590.72)
10	33.65	159.77	68.20	25.95	4.75	(496.75,74.05)	(113.95,590.69)
ST	**30.89**	**385.22**	**83.61**	**54.44**	**2.69**		
t=0.02							
Run	**Min depth**	**Max depth**	**Mean depth**	**STD of depth**	$\frac{\textbf{Max depth}}{\textbf{Min depth}}$	**Max depth position**	**Min depth position**
11	37.39	166.46	73.95	26.90	4.45	(354.37,62.55)	(113.98,590.72)
12	48.23	239.10	98.88	38.81	4.96	(354.45,62.52)	(113.95,590.70)
13	132.11	492.89	244.61	78.44	3.73	(354.37,62.55)	(113.98,590.72)
14	41.41	233.46	89.01	37.57	5.64	(354.37,62.55)	(113.98,590.72)
15	52.00	487.72	130.01	72.75	9.38	(354.37,62.55)	(113.98,590.72)
ST	**39.48**	**154.56**	**68.72**	**23.11**	**2.21**		

calculated using the following method. Given the calculated depth values for the common feature points and the corresponding ground-truth depth values, the first step is to convert both depth data sets to zero-mean data sets by subtracting the average depth value from each data set. Second, for both the zero-mean ground-truth depth data set and the zero-mean calculated depth data set, the average of the positive depth values and the average of the negative depth values are computed separately, and these are denoted by avg_truth_pos, avg_truth_neg, avg_calc_pos and avg_calc_neg, respectively. The estimated scale factor s is computed as

$$s = \frac{|avg\_truth\_pos - avg\_truth\_neg|}{|avg\_calc\_pos - avg\_calc\_neg|} \tag{122}$$

Finally, by multiplying the zero-mean calculated depth values with the scale factor s, and adding the average of the ground-truth depth values to the scaled zero-mean calculated depth values, the scaled calculated depth values will be in the same scale range as the ground-truth depth values.

To study the implemented system performance as discussed above, the Table image set, whose multiple views are taken indoor with the manual measurement of the distances from objects to the camera center, is used for analysis. Fig. 29 shows 1/5 of the estimated depth values for the middle view (View 4). The corresponding ground-truth depth values are plotted on the middle view in Fig. 30.

The average of the ground-truth depth values of all feature points is 58.10, and it is 41.03 for the calculated depth values. Note that the unit of the depth value is inch. After subtracting the average from each depth value data set, the zero-mean ground-truth depth values and zero-mean calculated depth values are shown in Fig. 31. The combination of these two plots is shown in Fig. 32. In the zero-mean ground-truth depth data set and zero-mean calculated depth data set, $avg\_truth\_pos = 12.85$, $avg\_truth\_neg = -11.51$, $avg\_calc\_pos = 10.22$ and $avg\_calc\_neg = -9.16$. The scale factor is calculated according to (122) using these parameter, $s = 1.2569$. After scaling the zero-mean calculated depth values, the zero-mean ground-truth depth values and scaled calculated depth values are plotted in ascending order in Fig. 33. By adding the average of ground-truth depth values to the zero-mean scaled calculated depth values, the combined plot of the ground-truth depth values and scaled calculated depth values is shown in Fig. 34. The mean square error (MSE) between the ground-truth and the scaled calculated depth values of feature points in the middle view is 0.0869.

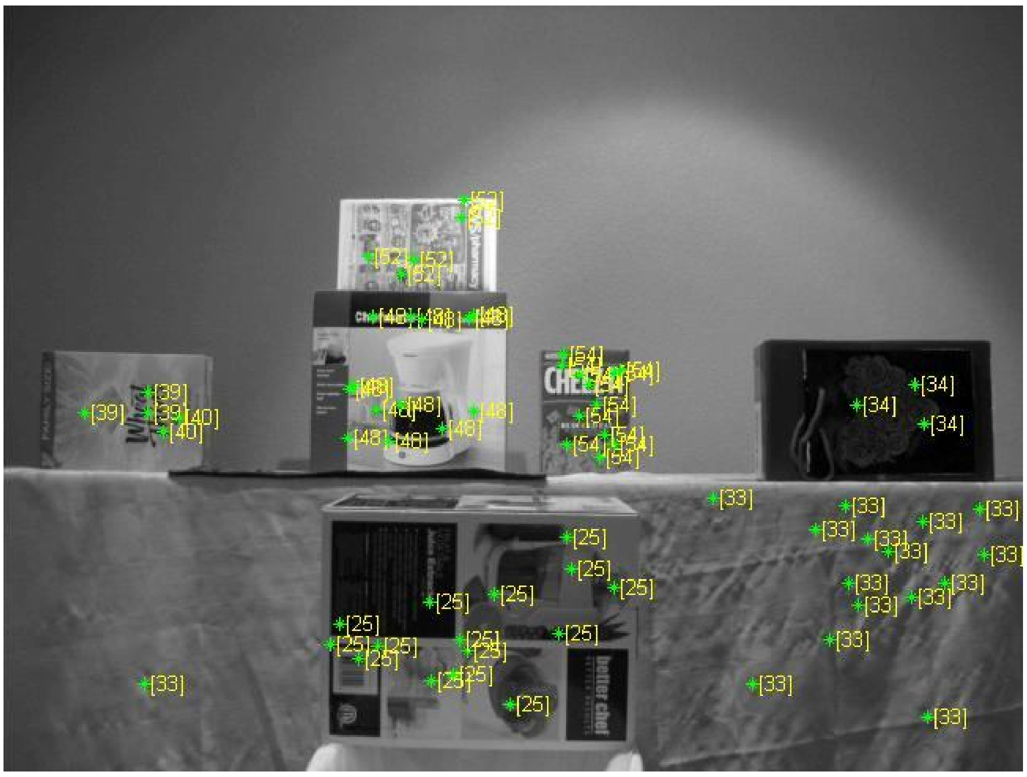


Fig. 29. Plot of 1/5 of the total number of the calculated depth values for feature points in the middle view (View 4) of the Table image set.

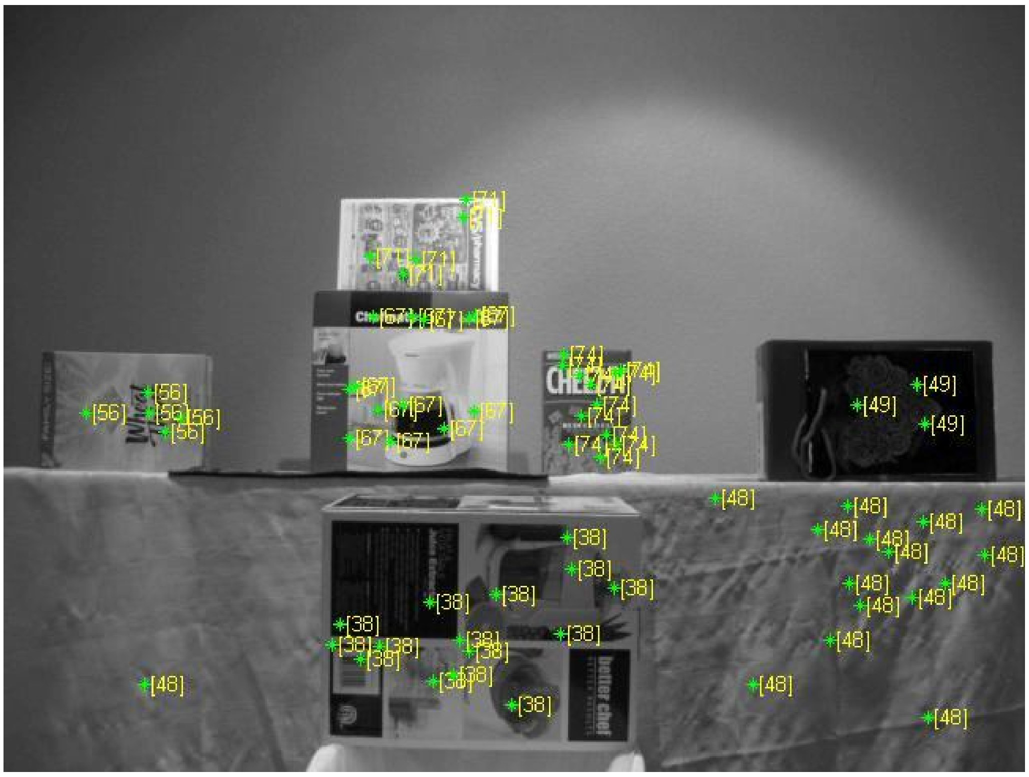


Fig. 30. Plot of 1/5 of the total number of ground-truth depth values for feature points in the middle view (View 4) of the Table image set.

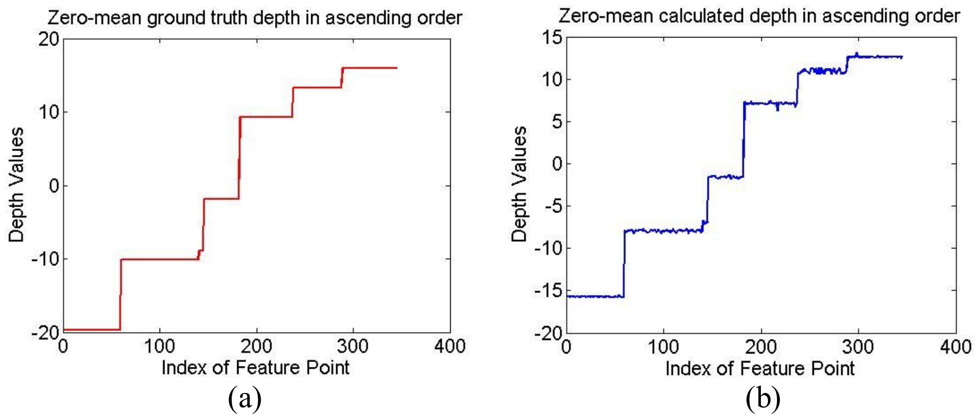


Fig. 31. (a) Zero-mean ground-truth depth values, and (b) zero-mean calculated depth values of the feature points detected in the Table image set.

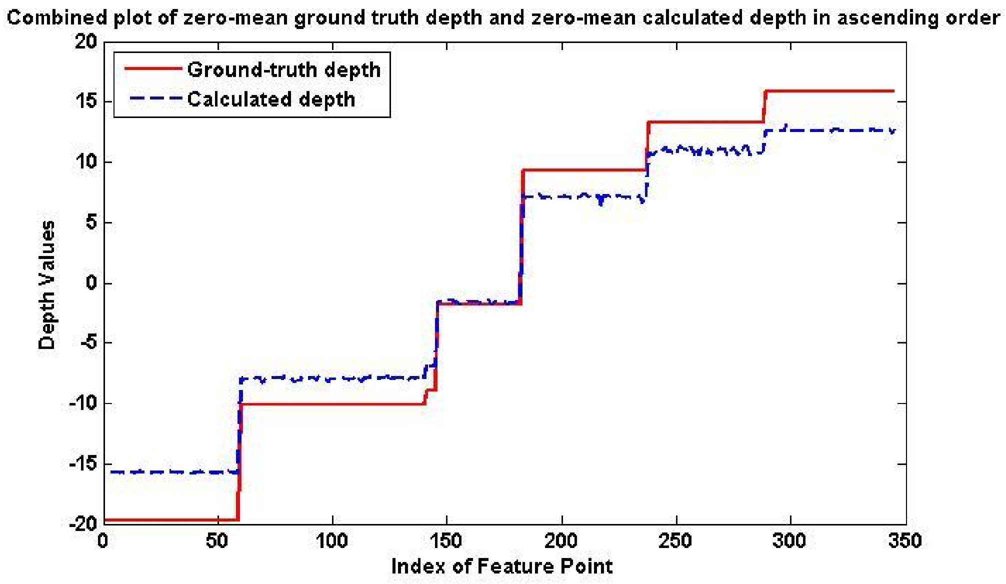
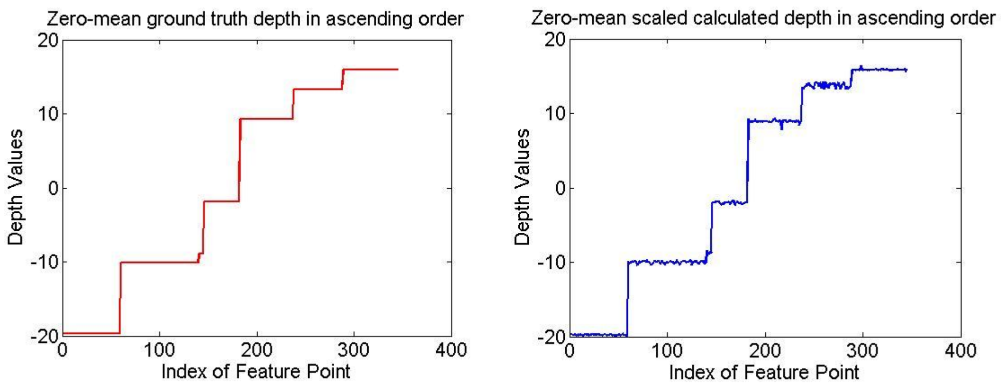


Fig. 32. Combined plot of the zero-mean ground-truth depth values and the zero-mean calculated depth values of feature points in the Table image set.



(a) (b)

Fig. 33. (a) Zero-mean ground-truth depth values, and (b) scaled zero-mean calculated depth values of the feature points detected in the Table image set.

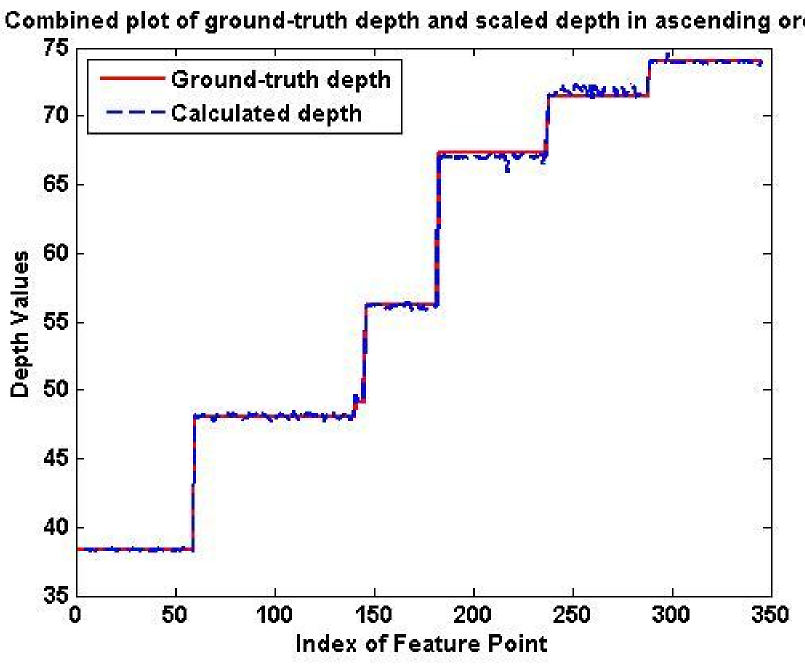


Fig. 34. Combined plot of the zero-mean ground-truth depth values and the scaled zero-mean calculated depth values of feature points in the Table image set.

5.3.5 Depth reconstruction with a limited number of known ground-truth depth

From the discussion in Section 5.3.4, by using the ground-truth depth information for all the feature points, it was found that the reconstructed calculated depth is similar to the ground-truth with a small MSE of 0.0869. In practical applications, usually it is not possible to know the depth values for all feature points; so, the question is: at least how many points with ground-truth depth value are required to reconstruct a reliable 3D model? And, given the ground-truth depth values of several feature points, how to scale the calculated depth and reconstruct the ground-truth depth values for all the feature points?

In the special case where only one ground-truth depth value of a feature point is known, the scale factor is calculated by dividing the ground-truth depth value by the calculated depth value corresponding to the same feature point. Then, the calculated depth values of all feature points are multiplied by the obtained scale factor. Using one point may produce incorrect depth values with a high MSE, because the chosen point may contain noise and, thus, it is not general enough to compute the scale factor of all feature points. Using the Table image set, if the ground-truth depth value at (377.36, 254.37) in the middle view is given, the scalar is computed to be 1.3802. The ground-truth and scaled calculated depth values of feature points are plotted in Fig. 35 in ascending order of depth values. The MSE between the ground-truth depth values and the calculated depth values is 3.9721, which is much larger than the MSE calculated knowing the ground-truth information of all feature points (0.0869).

With more than one known ground-truth depth values of feature points, a similar method to that presented in Section 5.3.4 can be used to reconstruct the depth values of all feature points. With the prior information of ground-truth depth values of N feature points, the scaled calculated depth values for all feature points are computed as follows. The first step is to shift each N-point depth data set (the ground-truth depth data set and the corresponding calculated depth data set) by its average value to form a zero-mean data set. The average of the positive values and the negative values for both zero-mean N-point data sets are calculated, and the scale factor is computed similar as in (122). Then, the zero-mean calculated depth values of all feature points are multiplied by the

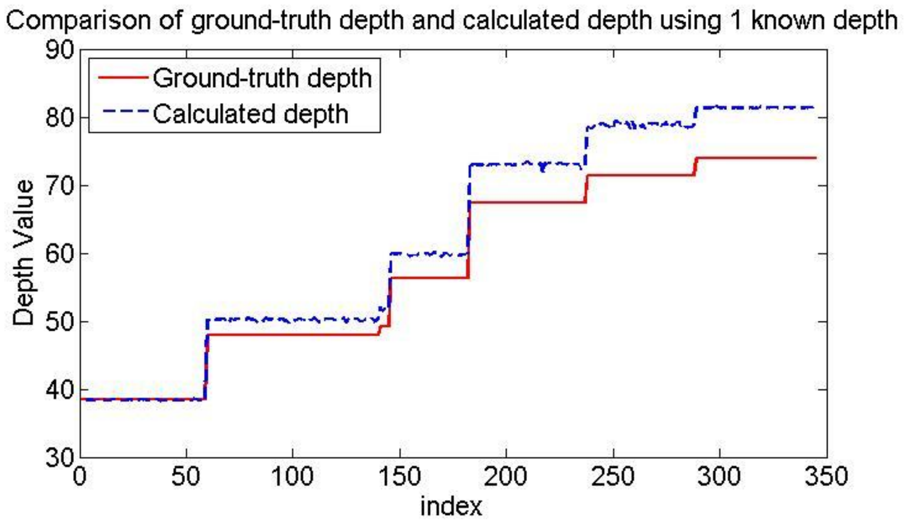


Fig. 35. Combined plot of the ground-truth depth values and the scaled calculated depth values for feature points in the Table image set using only 1 known ground-truth depth value.

obtained scale factor.

The next step is to shift the zero-mean scaled calculated depth values of all feature points by adding a shift factor. The shift factor is calculated as follows. Using the mean values of both the ground-truth and calculated depth values of N feature points, denoted as avg_calc_N and avg_truth_N, the scalar of the mean values is calculated as

$$scalar\_mean = \frac{avg\_truth\_N}{avg\_calc\_N} \tag{123}$$

Then the average for the calculated depth values of all feature points, denoted as avg_calc_all, is calculated. The shift factor is generated by scaling avg_calc_all by $scalar\_mean$ as

$$shift = (avg\_calc\_all) \cdot (scalar\_mean) \tag{124}$$

The shift factor calculated in (124) is added to the zero-mean scaled calculated depth values of all feature points, giving the final result for the scaled calculated depth values of all feature points.

Experiments are performed by choosing, randomly, N depth values out of the total set of ground-truth depth values for different values of N, in order to see the effect of different numbers of known ground-truth depth feature points on the reconstructed depth values using the Table image set. N is chosen to be 1, 3, 5, 10, 20, 40, 80, 150, 200, 250, 300 and 345, respectively, where 345 is the total number of common feature points in the considered image set. The MSE is calculated for different values of N, as shown in Table 7 and Fig. 36.

From Table 7 and Fig. 36, it can be seen that, as the number N of known ground-truth depth values increases, the MSE becomes smaller. When N increases from 1 to 40, there is a steep drop of the mean square error. If N exceeds 40, the mean square error has only a slight difference with that calculated using all the ground-truth depth values. The combined plot of the ground-truth and reconstructed depth values of all feature points with $N = 1$, $N = 10$, $N = 40$ and $N = 345$ is shown in Fig. 37. According to the results in Fig. 37, the reconstructed calculated depth values using 40 ground-truth depth values are similar to those using all ground-truth depth values. Thus, for the considered image set, it is sufficient to reconstruct the depth values for all feature points using at least 40 points.

Table 7. MSE between ground-truth and scaled calculated depth values of feature points in the Table image set using different numbers of known ground-truth depth values.

No. of known ground-truth depth values	**MSE**	**No. of known ground-truth depth values**	**MSE**	**No. of known ground-truth depth values**	**MSE**
1	3.9721	20	0.2359	200	0.0913
3	0.7865	40	0.0971	250	0.0894
5	0.5841	80	0.1028	300	0.0872
10	0.2611	150	0.0978	345	0.0869

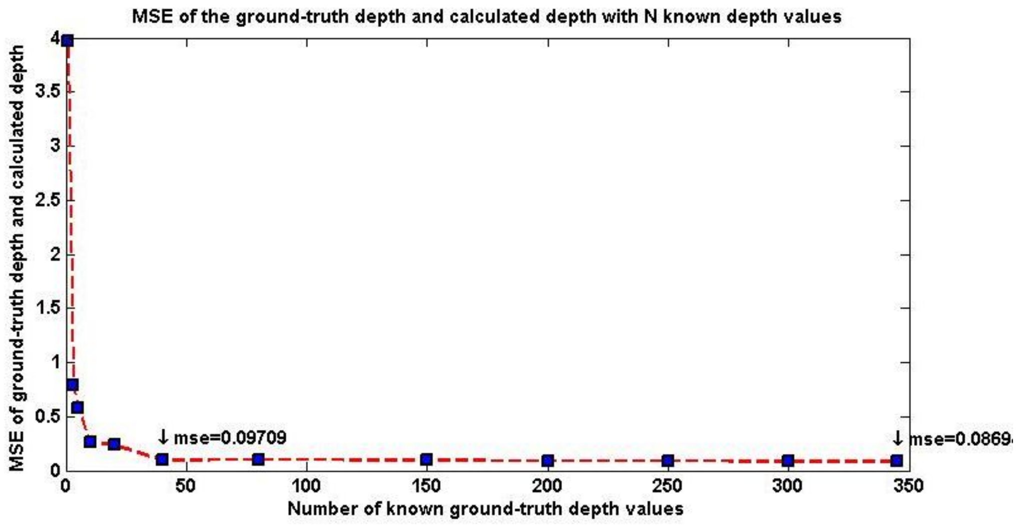
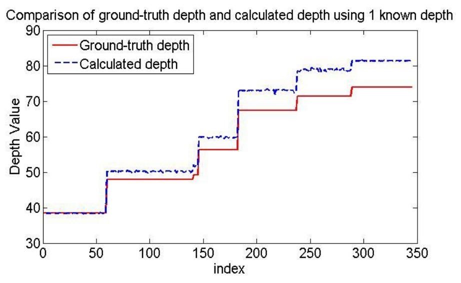
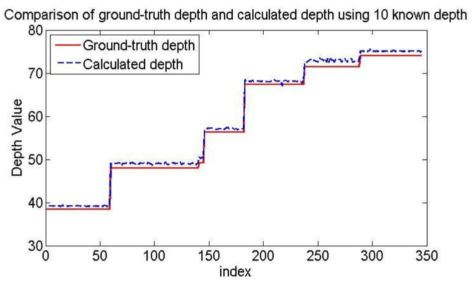


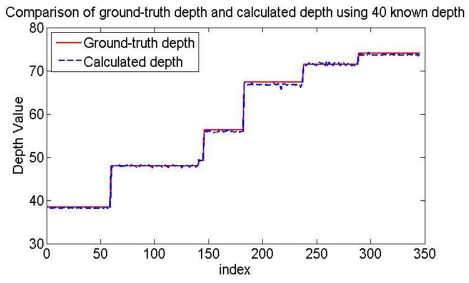
Fig. 36. MSE between the ground-truth and scaled calculated depth values of feature points in the Table image set using a different number of known ground-truth depth values.



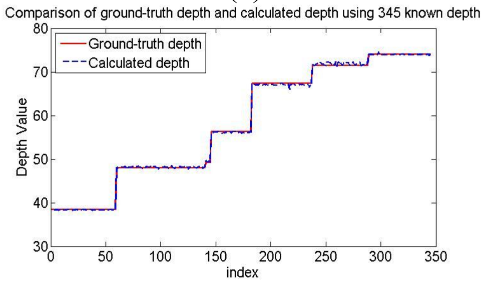
(a)



(b)



(c)



(d)

Fig. 37. Combined plot of the ground-truth and reconstructed depth values of feature points in the Table image set using different numbers of known ground-truth depth values. (a) Comparison using 1 known ground-truth depth value; (b) Comparison using 10 known ground-truth depth values; (c) Comparison using 40 known ground-truth depth values; (d) Comparison when all ground-truth depth values are known.

6. CONCLUSION

This thesis implements a 3D modeling system that utilizes multiple views of a scene to reconstruct the 3D depth information. This work contributes to the field of 3D imaging and 2D to 3D video conversion. This chapter summarizes the contributions of this thesis and proposes several directions for future research. Section 6.1 summarizes the contributions of the thesis, and Section 6.2 discusses directions of future work.

6.1 Contributions

In this thesis, a complete system for 3D modeling and depth reconstruction is implemented. The contributions of the thesis can be summarized as follows:

- The 3D modeling system in this thesis implements feature detection and image registration techniques to relate multiple views. The algorithm of the scale invariant feature transform (SIFT) is used to detect the features and to match the features among different views.
- Outlier removal and two-view geometry computation are implemented in this thesis. A robust estimation algorithm called the random sample consensus (RANSAC) is used to refine the feature matching between two views by removing the outliers of the feature points. The geometry between two views is estimated using the inliers of feature points.
- The projective reconstruction of the 3D scene is implemented in this thesis. The two-view geometry and correspondences of the feature points are used in conjunction with triangulation to reconstruct the projective structure of the

3D scene. Bundle adjustment is used to refine the projective reconstruction of the 3D scene.

- The projective reconstruction of the 3D scene is upgraded to a metric transformation through auto-calibration. Besides, the intrinsic camera parameters can be estimated using auto-calibration.
- Different image sets with eight multiple views are used to test the 2D to 3D conversion system. The results show correct relative calculated depth information of the feature points in multiple views. The scale factor in the metric transformation frame is estimated with the prior information of ground-truth depth values of feature points. The scaled metric reconstruction of the 3D scene only has a slight difference with the ground-truth data. The accuracy of the scale factor computation was analyzed in terms of the number of known ground-truth depth values.

6.2 Future Work

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

- In the step of image feature localization and feature matching, the scale invariant feature transform (SIFT) used in the 3D modeling system is sensitive to the noise in the image sequences and can produce some mis-matched 2D feature points between multiple views. More robust algorithms for feature detection and matching need to be studied.

- Triangulation can be improved by implementing the method that corrects the positions of 2D points under the existence of noise.
- The implementation time for the system can be further reduced in order to make this system suitable for processing the image and video sequences in real time.
- It is possible to implement the 3D modeling based on a pre-designed model with enough prior information about the 3D scene. This will help to reduce the complexity and computation.
- The 2D to 3D conversion system in this thesis can be further extended for video sequences instead of images. This will involve a further study of object and camera motion estimation.
- The 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.

7. REFERENCES

[1] C. Fehn, R. D. L Barre, and S. Pastoor, "Interactive 3-DTV--Concepts and Key Technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 524-538, March 2006.

[2] C. Fehn, "Depth-image-based Rendering (DIBR), Compression and Transmission for a New Approach on 3D-TV," *Proceedings of the SPIE: Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, pp. 93-104, May 2004.

[3] T. Okino, H. Murata, K. Taima, T. Linuma and K. Oketani, "New Television with 2D/3D Image Conversion Technologies," *Proceedings of the SPIE: Stereoscopic Displays and Virtual Reality Systems III*, vol. 2653, pp. 96-103, January 1996.

[4] E. Rotem, K. Wolowelsky and D. Pelz, "Automatic Video to Stereoscopic Video Conversion," *Proceedings of the SPIE: Stereoscopic Displays and Virtual Reality Systems XII*, vol. 5664, no. 1, pp. 198-206, January 2005.

[5] P. V. Harman, "Home-based 3D Entertainment – An Overview," *Proceedings of the IEEE International Conference on Image Processing*, pp. 1-4, September 2000.

[6] S. A. Valencia and R. M. R. Dagnino, "Synthesizing Stereo 3D Views from Focus Cues in Monoscopic 2D Images," *Proceedings of the SPIE: Stereo Displays and Virtual Reality Systems X*, vol. 5006, pp. 377-388, May 2003.

[7] G. Guo, N. Zhang, L. S. Huo and W. Gao, "2D to 3D Conversion Based on Edge Defocus and Segmentation," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2181-2184, March 2008.

[8] S. Mallat and S. Zhong, "Characterization of Signals from Multiscale Edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no.7, pp. 710-732, July 1992.

[9] A. Torralba and A. Oliva, "Depth Estimation from Image Structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1226-1238, September 2002.

[10] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm," *Neural Computation*, vol. 6, no.2, pp. 181-214, March 1994.

[11] Y. S. Huang, F. H. Cheng and Y. H. Liang, “Creating Depth Map from 2D Scene Classification,” *Proceedings of the International Conference on Innovative Computing Information and Control*, pp. 69, December 2008.

[12] V. Nedovic, A. W. M. Smeulders, A. Redert and J. M. Geosebroek, “Depth Information by Stage Classification,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1-8, October 2007.

[13] C. Tomasi and T. Kanade, “Shape and Motion from Image Streams: A Factorization Method,” *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137-154, November 1992.

[14] P. Strum and B. Triggs, “A Factorization Based Algorithm for Multi-Image Projective Structure and Motion,” *Proceedings of* the *European Conference on Computer Vision*, vol. 2, pp. 709-720, April 1996.

[15] B. Triggs, “Factorization Methods for Projective Structure and Motion,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 845-851, June 1996.

[16] S. Maham and M. Hebert, “Iterative Projective Reconstruction from Multiple views,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 430-437, June 2000.

[17] A. Heyden, R. Berthilsson and G. Sparr, “An Iterative Factorization Method for Projective Structure and Motion from Image Sequences,” *Proceedings of the International Conference on Image and Vision Computing*, vol. 17, no. 13, pp. 981-991, November 1999.

[18] N. Cressie, “Fitting variogram models by weighted least squares,” Mathematical Geology, vol. 17, no. 5, pp. 563-586, July 1985.

[19] M. Pollefeys and L. Van Gool, “Stratified Self-calibration with the Modulus Constraint,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 707-724, August 1999.

[20] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and R. Koch, “Visual Modeling with a Hand-held Camera,” *International Journal of Computer Vision*, vol. 59, no.3, pp. 207-232, September 2004.

[21] R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision,” Cambridge University Press, 2002.

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

[23] M. Chandraker, S. Agarwal, F. Kahl, D. Nister and D. Kriegman, "Autocalibration via Rank-Constrained Estimation of the Absolute Quadric," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, June 2007.

[24] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[25] A. P. Witkin, "Scale-space Filtering," *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1019–1022, August 1983.

[26] M. Brown and D. G. Lowe, "Invariant Features from Interest Point Groups," *Proceedings of the British Machine Vision Conference*, pp. 253-262, September 2002.

[27] J. Beis and D. G. Lowe, "Shape Indexing using Approximate Nearest-Neighbor Search in High-Dimensional Spaces," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1000–1006, June 1997.

[28] SIFT code available online at: http://www.vlfeat.org/~vedaldi/code/sift.html.

[29] 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.

[30] 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.

[31] P. D. Sampson, "Fitting Conic Sections to "Very Scattered" Data: An Iterative Refinement of the Bookstein Algorithm," *Computer Graphics and Image Processing*, vol.18, no. 1, pp. 97-108, January 1982.

[32] RANSAC code available online at:

http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/.

[33] R. I. Hartley and P. Sturm, "Triangulation," *Proceedings of the ARPA Image Understanding Workshop*, pp. 957-966, November 1994.

[34]Triangulation code available online at: http://www.robots.ox.ac.uk/~vgg/.

[35] 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.

[36] K, Levenberg, "A Method for the Solution of Certain Non-linear Problems in Least Squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164-168, January 1944.

[37] 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.

[38] Bundle Adjustment toolbox available online at:
http://vision.ucsd.edu/~vrabaud/toolbox/doc/.

[39] J. B. Lasserre, "Global Optimization with Polynomials and the Problem of Moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796-817, February 2001.

[40] GloptiPoly toolbox code available online at:
http://homepages.laas.fr/henrion/software/gloptipoly/.

[41] SeDuMi code available online at, http://sedumi.ie.lehigh.edu/.

[42] Microsoft_Ballet image sets available online at:
http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/.