3D Reconstruction - SfM with Multiple Images

Shangru Yi
Course Project for Digital Image Processing
Southern University of Science and Technology
11510099@mail.sustc.edu.cn

***Abstract*—Within this project, we implement 3D reconstruction algorithm based on SfM. Multiple images are used to restore the 3D structure of objects. We illustrate the principles of SfM based 3D reconstruction and run our implementation on two public datasets. Public library vlfeat and Matlab vision toolbox are utilized to finish this project.**

***Index Terms*—3D reconstruction, SfM**

I. Introduction

3D reconstruction from multiple image aims to estimate three-dimensional structures from two-dimensional image sequences with coupling local motion signal, such as rotation or other movement. This subject is studied in the fields of computer vision. The structure for normal reconstruction algorithm takes a set of images as input and produce a series point with estimated space position in three dimensions. Since 3D spatial position of a point can be reconstructed from two two-dimensional images once the position of same object points in images and camera intrinsic parameters are given, as illustrated in Section 2, several images for the same object taken by a camera placing at different positions, which can be done by moving camera around the object, can be used to reconstruct 3D structure. This method is termed as Structure from Motion (**SfM**). There are some important application for **SfM**. The first is in geosciences, which constructing the geological structures using images acquired from a range of digital cameras, especially in rugged environments when terrestrial laser scanning is limited. The other is in cultural heritage preservation and documentation. Since **SfM** provides a non-invasive approach for structure detection i.e. without direct interaction between object structure and operators, it replace traditional surveying when there are serious constraints in terms of site accessibility or feasibility of installing invasive surveying detectors.

Within this project, we realize the 3D reconstruction from multiple images based on **SfM**. We first introduce the camera model and some principle behinds 3D reconstruction in Section 2 and then illustrate the main step of structure reconstruction process, given a set of images in Section 3. The experimental result is demonstrated in Section 4. Finally, we present the conclusion and discuss about future works in Section 5.

II. Camera Model Review

We introduce the camera model utilized for projection in this section. In real world, an object can be represented by a group of points in three-dimensional coordinate if we have defined the origin. However, as for points in image, all points in real world is projected to an image and there are only two dimension information. When placing this image in the real world, the points in the image along with the camera are equipped with three-dimensional coordinate. We can thus use the rotation and translation in coordinate for the images taken by camera(s) placing in different spatial positions. If we use the camera center as the origin, the image can be represented in three dimensions with one axis as its focal length, shown as **Figure 1**. Moreover, since we always represent the positions of points in images with its coordinate in unit of pixel, we want to transform the real world coordinate in the same manner, i.e., representing points in unit of pixel. In order to accomplish this goal, the parameter of camera is required, since the coordination distance from a point in image to a spatial point is linked through the focal length. Then we finish construct a spatial coordinate for both points in image and actual spatial points.

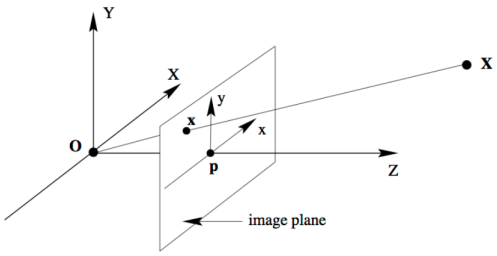

Fig. 1. Camera model with camera center

However, the origin is defined as the center of a camera, given some parameters relating with camera. We want to extend this some predefined origin, instead of only the camera-based coordination. Thus we need to transform the coordinate to world coordinate, i.e. actual spatial coordinate in three dimension, as shown in **Figure 2**. In order to model this, an extra matrix recording the camera project parameter is added. Within this matrix, there are two subpart: one is the rotation matrix, denoted as **R** and the other is the translation vector, denoted as **t**. This matrix is called as extrinsic matrix, which is estimated from the image points.

We have illustrated the camera model and the image formation model for one image and now we turn to 3D structure reconstruction from multiple images. As for this project, we implement our 3D reconstruction algorithm based on **SfM**. Since we use **SfM**, which utilizes the same camera for all

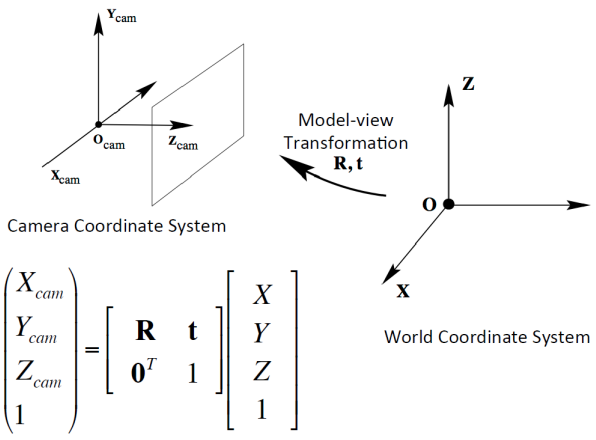

Fig. 2. Camera coordinate transform

images, the intrinsic matrix is the same for all images. And it is clear that once we are given both the intrinsic and extrinsic matrix, a point in image can be recovered to world coordinates built for that camera. When there are multiple cameras, multiple world coordinates are built. In order to reconstruct the 3D information for objects, we want to establish the relationship between multiple worlds coordinates and unify them to a uniform one. Now our problem becomes how to find the relationship between these world coordinates. When looking at images taking from slightly different angle, we can find some points presenting in both images but only encountering some translation or rotation. Thus, if we can use these points to construct a track to link all images, we can decide the relative position for each image and change its world coordinate to a relative one. The changed world coordinates are ready to be unified as a global one. Through placing the detected points from multiple images on the global world coordinate, the depth information and the shape changes of objects can be recovered and we finish our task of 3D reconstruction. The general principles for **SfM** 3D reconstruction are illustrated as above and there are still some detailed questions remaining, which are explained in running step for 3D reconstruction in following section:

1. How to calculate extrinsic matrix from images?
2. How to find and track those feature points in multiple images?
3. How to establish the relationship between multiple images using those feature points?

III. Main Steps for 3D Reconstruction

A. Feature Points Extraction with *SIFT* Algorithm

The first step for 3D-reconstruction based on **SfM** is to extract feature points from images. The features are also regraded as the image descriptors, since it carry important information including edge position. Features are described as points with particular kinds of surface texture surrounding them, e.g. corner points, which make them stand out [1]. It is important same points are extracted from images of the same scene, but taken from different viewpoints. The surrounding of points are usually expressed in vector, which is ready for some similarity search algorithm to conduct matching between them. After matching, the extracted points are further used for following parameter estimation for 3D reconstruction. Since there are illumination variance and possible degradation due to blur, it is expected that the extracted features are invariant to those degradation or basically unaffected.

There are abundant feature extraction algorithms for this purpose, including scale-invariant feature transform (**SIFT**) and Harris corner detection. These algorithms utilize pixel densities in square regions around the detected feature points and squares are rotated and scaled in a covariant way with the surrounding intensity pattern. In this manner, it is expected that same surface area changed by a similarity transform will be covered. Though Harris corner detection is rotation-invariant, it is not scaling-invariant, since a corner in fixed window size after scaling would be no longer a corner fitting in that window. **SIFT** can deal with scaling image by dynamically adjusting the window size. Given this, we employ **SIFT** to extract the image descriptors for this project and we briefly illustrate the procedures for feature extraction in **SIFT** as follows.

SIFT is a feature detection algorithm to detect and describe local features in images, proposed by D. Lowe in 2004. Interesting points on the object can be extracted to provide a feature description of the object. It is expected that the features extracted from images should be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as edges and corners. Also, the relative positions between them in the original scene shouldnt change from one image to another. In practice, SIFT detects and uses a much larger number of features from the images. The extracted features are invariant to uniform scaling, orientation, illumination changes and partial invariant to affine transformation, including translation, rotation and scaling. As result, **SIFT** transforms an image to a large collection of features in terms of points and vector descriptors for surrounding. There are several steps to run **SIFT** for descriptor generation [2]. Since we use merely **SIFT** to extract candidate points in images and ignore the extracted vector descriptors, we illustrate the starting two steps. Further details about **SIFT** are available in [2].

1. Scale-space Extrema Detection
   This step is to use a dynamic window size and then a local maximum in that window. The local maximum is treated as a candidate for feature points. The window size is found by using Gaussian filter with different variance. With small variance, smaller corner can be detected and with large variance, larger corner can be detected. The variance can be utilized as scaling factor to dynamically adjust window size.
2. Key-point Localization
   It is clear that edge is the position where encountering great changes in one dimension, leading big ratio between the first and second eigenvalues. We dont want edge

points but merely the corners. However, edges are also detected in the first steps, we want to eliminate them from our candidate points. For each candidate points, if its ratio exceeds a preset value, it is regarded as edge point.

B. Features Matching

The images are matched based on the extracted feature points in pairing manner. Since the extracted feature points presents in adjacent images, the pairing relationships between features in two images are found by matching algorithm. The pairing relationship can thus be used to estimate the transform matrix of images. Since the **SIFT** can find the feature points in two images and express those points in vector, some distance-based features matching algorithm is ready for this aim, including K-Nearest-Neighbor (KNN) with KD trees. Matching between two views or images using **SIFT** and Harris Corner detection are shown in **Figure 3** and **Figure 4**. Within our projects, we use Kanade-Lucas-Tomasi (**KLT**) feature tracking algorithm to tract the matching relation between extracted points in two images. **KLT** focus on tracking for some selected points in images. The extracted points are usually the result of feature extraction algorithms, including **SIFT** and Harris corner detection. There are two parts in original **KLT**: one is good feature selection and other one is the feature tracking from image to image. Typically, the ratio of Eigen-value within a window is used as selection criteria for feature points in images, which is the same as Harris corner feature selection. Since we use the features extracted from **SIFT**, the feature selection part is eliminated. Before we illustrate the feature tracking step in **KLT**, there are some important assumptions:

1. The intensity value maintains constant
2. Motion is relatively small and the time is consistent
3. Close points encounter similar motion, remaining close to each other.

With above assumptions, the pixels in a local area within window are assumed to move a same distance or encounter same transformation from one image to the other and we just want to minimize the difference between the intensity values for pixels in window to find the disparity vector, considering translation in two dimensions. Moreover, in order to deal with scaling issue, pyramidal feature tracking is introduced. For each level of pyramid, the image is scaled to a factor and the disparity vector estimation for image is implemented. Now we illustrate the running steps of **KLT**. **KLT** takes two images and the extracted feature points in one image as input. The motion, i.e. disparity vector, estimation is then implemented for each feature points in that image and its correspondent in the other image can be found. There are some feature points absent from the other image. The output of **KLT** is the feature points matching between two images. This manner is extended to multiple images in a similar way. The established match can be thus used for multi-image relationship estimation and 3D reconstruction. The detailed illustration for **KLT** is available in [3].

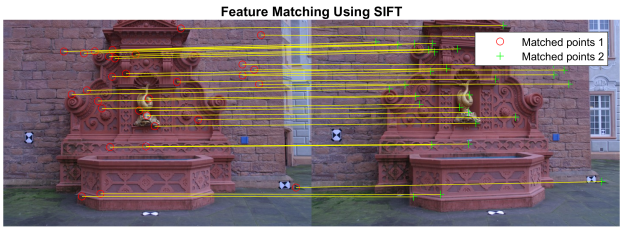

Fig. 3. Feature matching using **SIFT**

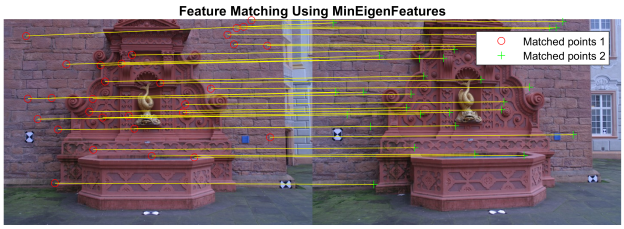

Fig. 4. Feature matching using Harris corner

C. Fundamental Matrix Estimation

After the descriptors or feature points matching, the extrinsic matrix is required to be estimated. Instead of directly estimating **R** and **T**, we start from estimating the fundamental matrix. Since there is mapping between points in related two images, we want to establish their relationships. As stated in Section 2, the spatial point can be expressed in terms of intrinsic matrix and extrinsic matrix, which is composed of rotation matrix and translation vector. Assume a spatial point presents in two images. We can thus express one of them in terms of the other along with the two extrinsic matrix, assuming we use same intrinsic matrix for images. According to linear algebra, the expression is equivalent to the calculation of determinant [1]. The determinant calculation can be further expressed in terms of **Equation 1**, where x_1, x_2 are the position of point in two images and $\boldsymbol{F}$ is a 3×3 matrix, called as *fundamental matrix* of the image pair. Detailed deduction and principle illustrations are available in [1]. The **Equation 1** can be extended in terms of **Equation 2**. It is clear that we only need 8 point-pairs to calculation all the components in $\boldsymbol{F}$.

$$x_1^T \boldsymbol{F} x_2 = 0 \tag{1}$$

$$[x_1x_2, x_1y_2, x_1, y_1x_2, y_1y_2, y_1, x_2, y_2, 1] \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{2}$$

Since we can find lots of point-pairs in the above feature matching step, the problem now becomes the selection of

8 point-pairs to calculate $\boldsymbol{F}$. Assume we have calculated $\boldsymbol{F}$ from a selected set of 8 point-pairs. According to **Equation 1**, two matching points are expected to be related through fundamental matrix. We can thus define the error function given the fundamental matrix as **Equation 3**. Furthermore, we define inlier points as the points matching with error smaller than a pre-defined threshold. Then it is expected that a good fundamental matrix can produce more inlier points. For implementing **RANSAC**, we sample 8 points matching from feature matching and calculate fundamental matrix using sampled 8 points matching. We then use the calculated fundamental matrix to count the number of inlier points. We repeat above steps for some preset iteration number and select the fundamental matrix with max number of inlier points as the output. After the retrieval of fundamental matrix, we now turn to our ultimate goal, i.e. finding world coordinates for feature points.

$$e = ((\boldsymbol{F}x_1) \cdot x_2)^2 + (x_1 \cdot (\boldsymbol{F}x_2))^2 \tag{3}$$

D. SfM for Two Images

After we found a good fundamental matrix from two images, we want to find the extrinsic matrix and thus rotation matrix and translation vector. The fundamental matrix can be rewritten to essential matrix according to **Equation 4**, where $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are the intrinsic matrix. According to [4], the essential matrix can be rewritten using SVD as **Equation 5**. If we take the first rotation matrix as identity matrix and the first translation vector as $\mathbf{0}$, we can get four possible solutions for the second extrinsic matrix. For the four solutions, we can take the solution with maximal number of inlier points lying in front of both cameras. Once the extrinsic matrix for both images are decided, we can get the position of spatial points in world coordinate. We might want more feature points in images, in order to reconstruct a dense and relatively completed structure for objects. Thus, after retrieving the extrinsic matrix, we run a dense version of feature point extraction in one image and find their matching in the other image. For this project, we utilize a dense version of **SIFT**, which extract features in small local areas. Now we complete the procedures of reconstructing 3D structure using two images.

$$\boldsymbol{E} = \boldsymbol{K_1}^T \boldsymbol{F} \boldsymbol{K_2} \tag{4}$$

$$\boldsymbol{E} = \boldsymbol{U} diag(1, 1, 0) \boldsymbol{V}^T \tag{5}$$

E. Reconstruction with multiple images with Incremental SfM

We have already completed the 3D reconstruction based on **SfM** using two images and we now extend it to multiple images. This can be done in incremental manner. Assume we have three images. We first use image 1 and image 2 to perform our 3D reconstruction according to above steps, using image 1 as origin by taking its rotation matrix as identity matrix and translation vector as $\mathbf{0}$. We can get the extrinsic matrix for image 2 and thus the relative position information for image 1 and image 2. Then we use image 2 and image 3 to perform our 3D reconstruction according to above steps, using image 2 as origin by taking its rotation matrix as identity matrix and translation vector as $\mathbf{0}$. We can get the extrinsic matrix for image 3 and thus the relative position information for image 2 and image 3. Since there are one origin in the uniform world coordinate, we want to establish the connection between these two relative representations. Since image 2 presents in both the first and second coordinate, its extrinsic matrix is clearly the bridge. There are only one extrinsic matrix for image 2 in a uniform coordinate. If we take the image 1 as origin, there is rotation and translation for image 2, leading change in its extrinsic matrix used in **SfM** for image 2 and image 3. As for rotation matrix, the new rotation matrix for image 3 is not only the calculated one but the rotation matrix of image 2 multiplying with the calculated one. As for translation vector, the new translation vector for image 2 is not only the calculated one but the translation vector of image 2 adding with the calculated one. For the case of multiple images, we just extend this manner until all extrinsic matrix for images are adjusted.

IV. Experiment and Running Step

Instead of estimating the camera intrinsic parameters from several standard images, i.e. chess board, we directly use the available information about intrinsic matrix for images. The images are grouped in sets. The datasets are originally from [5]. We test our implementation on two datasets: one is the **handprint**, having two images; the other is the **fountain**, having 8 images. We run two times for **fountain**, respectively selecting 5 and 8 images to see the results. For images in the same dataset, it is taken by the same camera by moving around the target object, leading the same intrinsic matrix for calculation.

The running results for **handprint** is shown in **Figure 5** and **Figure 6**.

Fig. 5. Handprint (2 images) - Front

The running results for **fountain** with 5 images is shown in **Figure 7** and **Figure 8** and the running results for **fountain** with 8 images is shown in **Figure 9** and **Figure 10**.

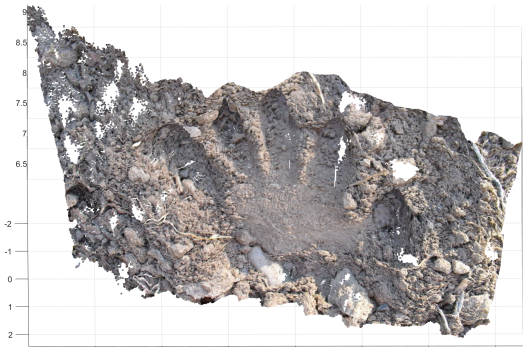

Fig. 6. Handprint (2 images) - Back

Fig. 7. Fountain structure reconstruction (5 images) - Front

As for **handprint**, it is clear that we see the depth information from reconstructed point cloud figures and the points extracted by the dense version of **SIFT** fully fill the figure. As for **fountain**, the reconstruction result is not that good when the figure number is large and there are slices for the structures, which is assumed to be the result of inconsistent extrinsic matrix estimation for incremental **SfM**. Moreover, according to the assumption of **KLT**, the motion is required to be relatively small. As the number of images increasing, this assumption is more or less weaken. However, we can still see the recovered depth information.

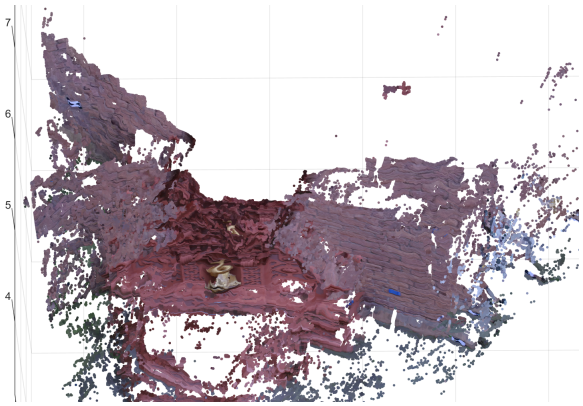

Fig. 8. Fountain structure reconstruction (5 images) - Top

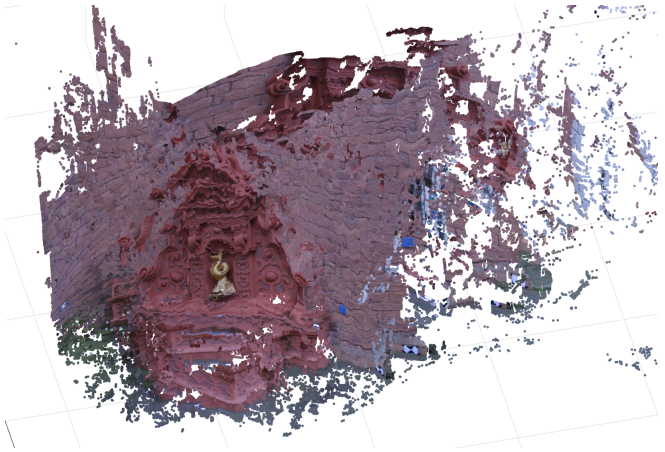

Fig. 9. Fountain structure reconstruction (8 images) - Front

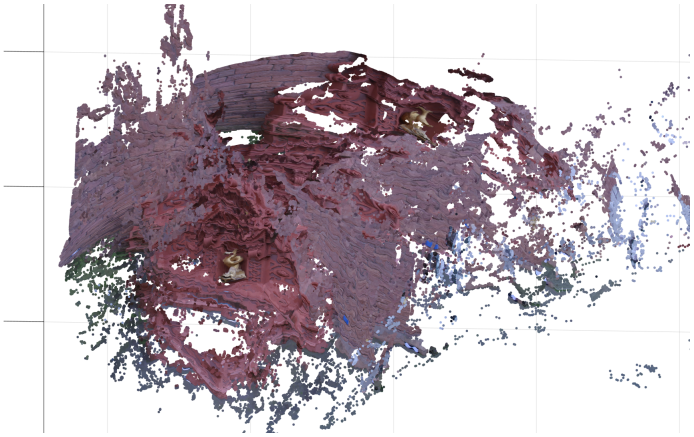

Fig. 10. Fountain structure reconstruction (8 images) - Top

There are several basic **.m** files to conduct the steps illustrated in Section 3. We illustrate their functions according to the main steps for 3D reconstruction.

1. **main.m**
   Take image path and some output setting as input.
2. **SfMConstruction.m**
   Contain main steps illustrated in Section 3 and show reconstruction result.
3. **getSIFTFeature.m**
   Call function in **vlfeat** to perform **SIFT** or its dense version.
4. **getMatches.m**
   Get feature point matching using vision.P
5. **RANSAC.m**
   Retrieve fundamental matrix by iterative sampling.
6. **RotationAndTranslation.m**
   Estimate rotation matrix and translation vector.
7. **TriangulationImage.m**
   Restore the spatial information for 3D points, given extrinsic matrix and feature points.

Within this project, some present **Matlab** functions used for computer vision, including **vision.pointTracker**, and C-based extra libraries **vlfeat** are used for both the original and dense version of **SIFT**. ***A zip package for vlfeat is in the submission folder. Please unzip it and enter command like "run('yourPath/vlfeat/toolbox/vl_setup')" to set up vlfeat before running the main function.***

V. Conclusion and Future Works

Within this project, we implement a system to perform 3D reconstruction from multiple images. We use incremental **SfM** with feature extracting using **SIFT**. By estimating fundamental matrix of two images, we can establish their relative position relationship and recover the spatial information for matching in images in incremental manner.

Due to the time limit and loss of expertise knowledges, present implementation in this project only consider the relationship between two images and extend the projection relation by adding one image at the same time. There are information loss for points presenting in multiple images, resulting the inconsistent estimation of rotation matrix and translation matrix. A more accurate method is to establish tracks for feature points presenting in multiple images and use **Bundle adjustment** to minimize the re-projection error using different extrinsic matrix. Though there are **Bundle Adjustment** in **Matlab**, we did not find any good reference about it and we may leave this task as our future works.

References

[1] T. Moons, L. Van Gool, M. Vergauwen *et al.*, "3d reconstruction from multiple images part 1: Principles," *Foundations and Trends® in Computer Graphics and Vision*, vol. 4, no. 4, pp. 287–404, 2010.

[2] A. Mordvintsev and A. K. Revision, "Introduction to sift (scale-invariant feature transform)," https://opencv-python-tutroals.readthedocs.io/en/latest.

[3] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[4] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[5] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2008, pp. 1–8.