Design Flow of Motion Based Single Camera 3D Mapping

Guojun Yang, Zhen Zhou, Thomas Gonnot and Jafar Saniie
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, Illinois, USA

***Abstract*—With the improvements in sensor technologies and image processing algorithms, computer vision has become a major tool for robots to recognize and gauge their surroundings. For instance, the Kinect sensor can be used as an excellent depth camera for indoor navigation. However, there exist situations that need recognition and spatial interpretation of the environment using limited hardware resources. The Kinect is not suitable for outdoor use while LIDAR is too large and expensive to be installed on an autonomous miniature surveillance drone. Therefore, the use of a single camera is the only feasible option for many embedded applications. Hence, this paper examines different Structure from Motion (SfM) implementations, and focus on methods that require only one camera to perform an efficient 3D mapping of a scene.**

I. INTRODUCTION

3D information is extremely important for applications related to artificial intelligence, especially when robots are used to understand its surrounding environments. For the robots to navigate in a certain area, collision avoiding and path planning abilities are necessary. The navigation is successful, only if obstacles and empty spaces can be accurately detected by the robot. This can be made possible by extracting the 3D information from sensors. Humans always perceive the world in 3D by using a deliberate stereo vision system. However, for computer, it becomes a very difficult task. During the recent years, researchers have made significant progress and obtained comprehensive outcomes, such as depth maps and 3D point cloud [1][2][3].

Using stereovision camera or depth camera to create depth images are the most widely used methods for 3D mapping. Stereo vision cameras, depth cameras and LIDAR sensors are all able to reconstruct the 3D information effectively, but there are limitations as well. Typical depth camera such as Kinect is not able to operate outdoor, especially when the sunlight is strong. LIDAR is often too large to be mounted on small devices, and it is also relatively expensive. Stereo vision cameras need more hardware resources compared to using single camera. Moreover, most of the existing videos were recorded using single cameras. Consequently these videos cannot be directly used for 3D mapping. Therefore, the environment needs to be reconstructed from videos taken by single cameras.

This paper focuses only on mapping the environment using video clips taken by single regular camera. There are a number of similar research works that use single camera to extract 3D information. Noah Snavely [2] presented an application that is able to build a point cloud representation based on huge amount of photos of the same area, taken by different cameras. Researchers from Microsoft [4] came up with an application that can stabilize time lapse video by estimating the locations and poses of camera, and further virtually rerouting the camera and stabilize the video. Bao's team [1] successfully extracted the semantic information by recognizing certain key objects and estimating their relative location.

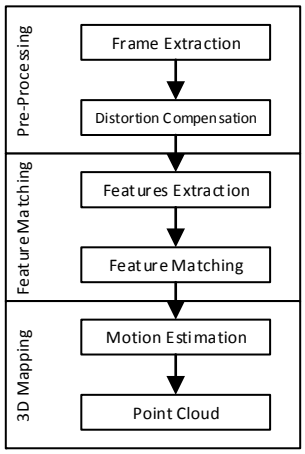


Figure 1. Work flow of 3D mapping using a single camera

To achieve the goal of 3D reconstruction using single camera, we used an AR drone from Parrot as a reference. The camera on the drone is used to perform 3D reconstruction by recording video while the drone is flying around our research laboratory. Figure 2 shows the major steps involved in reconstruction of 3D mapping. The first step is pre-processing of the acquired video which includes extraction of frames from the video and rectifying the image to compensate the lens distortion. This process is discussed in Section II. Once a

978-1-4799-8802-0/15/$31.00 ©2015 IEEE

new frame is rectified and cropped, it proceeds to perform feature match process (Section III). Feature points are located, and matched with previously processed frames. Finally, the motion of camera is estimated according to matched feature points, and a point cloud representation is created as well.

II. IMAGE DISTORTION COMPENSATION

The first step in 3D mapping is to place the camera on the drone for properly generating images. In this paper, a wide-angle camera is available and used as the only sensor on the drone. Wide-angle camera provides wider visual area, which is very helpful for navigation and 3D reconstruction. However, wide-angle also brings significant distortions on images. Such distortion will affect the accuracy of feature matching and 3D mapping. To compensate the distortion, the wide angle camera must be calibrated.

A typical camera can be modeled as pinhole [5]. As described in pinhole model, an ideal camera should have no distortion: the projection from real world coordinate (shown as M in equation 1) to camera plane coordinate (shown as m). Such process can be interpreted by equation 1. As shown in equation 1, s is an arbitrary scale factor that helps to balance the equation. A represents the intrinsic camera matrices, and [R|t] is the extrinsic matrix, which represents the rotation and translation of the camera. Equation 2 illustrates the information carried by the intrinsic matrix A and extrinsic matrix [R|t]. This is the ideal form of pinhole projection, which is without distortion. Hence in intrinsic matrix A, there are only f_x and f_y, which represent the focal length in horizontal and vertical directions respectively, c_x and c_y (principal points of the image) represent the center of the images, since the images are not cropped.

$$sm = A[R|t]M \tag{1}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

Equation 1 and equation 2 represent the ideal case of projection from real world 3D coordinate to 2D camera plane. Such projection can actually be broken into 2 sub-projecting processes; one is the projection from 3D real world coordinate to 3D camera coordinate, as shown in equation 3, and the other is from 3D camera coordinate to 2D camera plane. Distortion happens during the second projection.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{3}$$

In ideal pinhole model, the projection from 3D camera coordinate to the camera plane is free of distortion. This projection can be interpreted as equation 4.

$$\begin{aligned} u &= f_x \times x' + c_x \\ v &= f_y \times y' + c_y \end{aligned} \tag{4}$$

where

$$\begin{aligned} x' &= \frac{x}{z} \\ y' &= \frac{y}{z} \end{aligned} \tag{5}$$

When radial distortion and tangential distortion are considered, the projection from 3D camera coordinate to camera plane becomes more complicated. As shown in equation 6, additional coefficients (k_1 to k_6 and p_1 to p_2 which define the radial and tangential distortion respectively) are added to the projection due to the distortion. The calibration process aims to find these coefficients. By re-projecting each one of the captured images, distortion can be compensated, which allows generation of an accurate 3D mapping.

$$\begin{aligned} x'' &= x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2\left(r^2 + 2x'^2\right) \\ y'' &= y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1\left(r^2 + 2y'^2\right) \end{aligned} \tag{6}$$

where

$$r^2 = x'^2 + y'^2 \tag{7}$$

The intrinsic matrix is determined by the optical features of the camera lenses, which means that when the technical specification of a camera is known, the distortion coefficients can be calculated. Certain applications [2] use the Exif (Exchangeable image file format) tag, containing the information about the lens of the camera, to rectify the distorted images. The drone used for our experiment had unknown camera technical specification. Therefore, we had to determine the intrinsic matrix A using chessboard calibration. In this calibration process, a chessboard was used as a reference. Corners that connect squares with the same color are used as feature points. Once the feature points are located in the different images, maximum likelihood estimation, as introduced by Z.Zhang [5], is performed to determine the intrinsic matrix.

III. FEATURE POINT DETECTOR

One of the most challenging aspects of performing 3D mapping using single camera is to find out matched points through images. Once matched points are identified in different images, two-view or multi-view geometry calculation can be conducted. Furthermore, the 3D location of each feature points can be determined. With the location of feature points determined, the point of cloud can be created.

It is easy for humans to identify the same point or object from different images, however, it is difficult for robots to perform such operations. The challenge in performing multi-view geometry computation is to identify and match feature

points across images as mentioned earlier. In multi-view geometry, the movement of the camera, which is shown in the form of the camera extrinsic matrix, can only be determined once enough matching feature points are identified. To overcome the misperception created by translation and rotation of camera, which will result in failure of matching, scale and rotation invariant feature points algorithm need to be adopted. One of these methods is introduced by David G. Lowe [6], which is called the SIFT algorithm, for Scale-Invariant Feature Transform.

Compared to Harris corner detection which is only translation and rotation invariant, SIFT is also scale invariant. Each feature point detected by the SIFT algorithm stores information including the location where the feature point is detected, as well as the scale information. The orientation and descriptor are determined or assigned after the feature point is located. Scale invariant detector is pivotal to 3D mapping via flying drone, since in most cases, camera on drone are aiming the same direction, which enlarges or scale up objects in front of the drone.

When detecting feature points using SIFT, the first step is using difference of Gaussian to approximately locate potential feature points. To be more specific, each image will be convolved with a Gaussian kernel multiple times. During each convolution, a Gaussian kernel with different deviation will be applied. Difference of Gaussian or DoG will be then performed between each of two consecutive convolved images. By down sampling the image, a new octave is created. The new octave will then be convolved with the same Gaussian kernels and the DoG is generated as for the first octave. This process is shown in Figure 2, where the DoG processes are conducted on two different octaves (Original photo is shown as first octave and resized photo is shown as next octave).

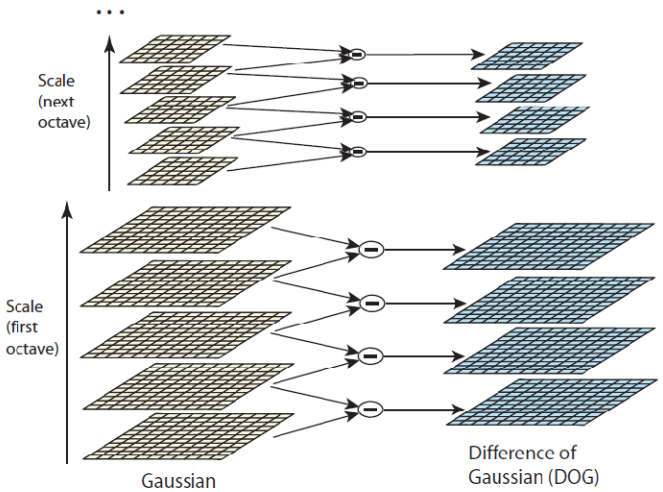


Figure 3. Illustration of DoG[6] between different scales in each octave

The DoG process is a close approximation to the scale-normalized Laplacian of Gaussian, which can detect edges correctly. Edges detected using different Gaussian kernels will have edges and noise with different strength. The candidates for feature point will be found among the DoG images that have largest value compared to its8 neighbor pixels from same scale, as well as 18 pixels from its consecutive scales. The orientation of each feature point is assigned relative to its neighbor pixels. The gradient magnitude, m(x, y) of each feature point is calculated as shown in equation 6.

The orientation is calculated using equation 7. These orientations and magnitudes are initial values, since the orientation is still relative to the image which is not rotation invariant. After each of the key points has been assigned a magnitude and orientation, its neighbor will be split into 64 regions. These 8 by 8 regions are combined together to form a 2 by 2 region. Each region has its magnitude and orientation weighted by a Gaussian kernel. The descriptor of each feature point is a vector containing magnitude and orientation information of all 4 neighboring regions. Such process is illustrated in Figure 3. The matching process is based on the descriptor using the nearest neighborhood.

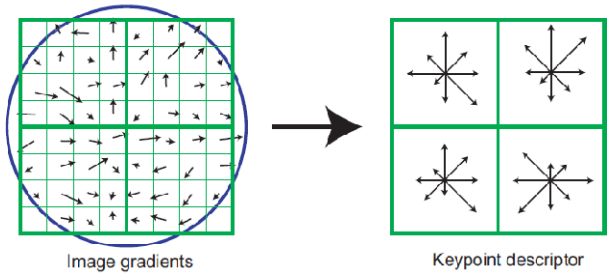


Figure 4. Illustration of descriptor of SIFT feature point [6]

IV. 3D MAPPING

Motion estimation or mapping is the final stage of the overall process. This process highly relies on multi-view geometry. Multi-view geometry helps to calculate the rotation and translation, which is represented by the extrinsic matrix of the camera for each frame. As shown in Figure 4, there are 3 different photos taken by the same camera at different locations t_1, t_2 and t_3 with different poses R_1, R_2 and R_3. X is a keypoint in real world, which is projected on photos taken at locations 1 to 3. Each projection is shown on camera planes as X_1, X_2 and X_3.

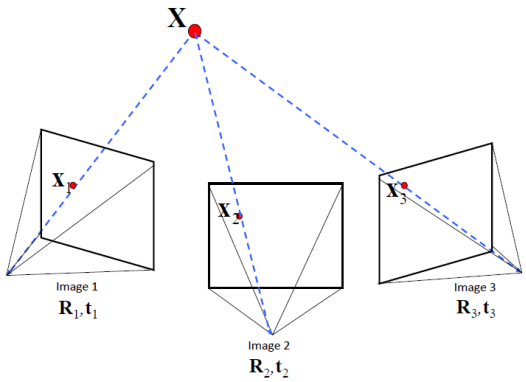


Figure 5. Illustration of multi-view geometry

Once a pair of projections from a keypoint is known, the fundamental matrix can be estimated using equation 8. To achieve an estimation, multiple points should be chosen and the matrix that gives smallest geometric error is determined, as shown in equation 9 [7].

$$x_1^T F x_2 = 0 \tag{8}$$

$$\min \sum_j ||x_1^j - F x_2^j||^2 \tag{9}$$

Once the fundamental matrix is found, essential matrix E can be calculated as:

$$E = K_1^T F K_2 \tag{10}$$

At this point, it is possible to do the reverse projection using equation 1. Since the intrinsic and extrinsic matrices of the camera are known, M location of points in real world can be calculated by using m as input.

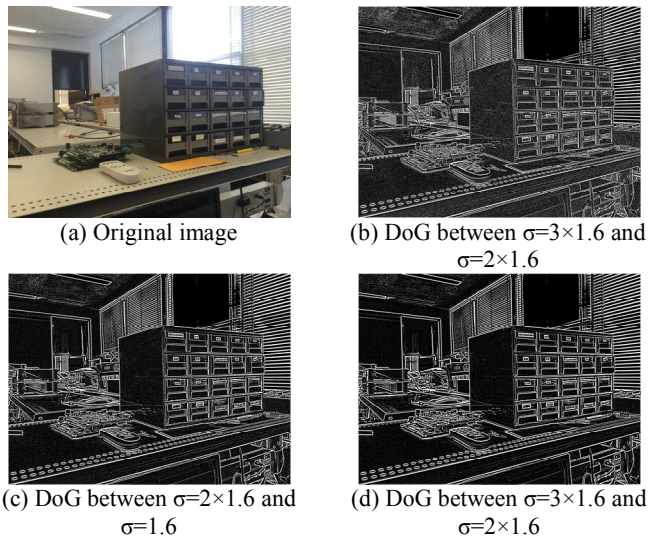
V. RESULTS

In this study, we have implemented and tested some of the algorithms mentioned in previous sections of this paper. The first experiment conducted is image distortion compensation. A set of photos were taken by the drone in front of a chessboard pattern with multiple orientations. During such experiments, we discovered that the chess board must be held perfectly flat on a surface; otherwise it will generate an incorrect camera intrinsic matrix. To overcome this, we used a tablet displaying the image of a chessboard. Figure 5a shows one frame from the original video, compared with one frame from the calibrated video as shown in Figure 5b. The difference is obvious: in the case of the non-calibrated video, all doors and walls are appearing with straight lines. However, the calibrated video appears to be stretched toward the four corners, which create blank areas (shown as black). Such blank areas contain no information, therefore must be removed to improve computational efficiency, while performing feature point detection.

(a) Original video (b) Calibrated video

Figure 6: Camera calibration

Figure 6 shows the result of edge detection using the Difference of Gaussian method. The brightness of the white lines indicates the strength of edges. Once the edge detection is completed, the local extremas will be found as candidates of feature points, as shown in Figure 7.

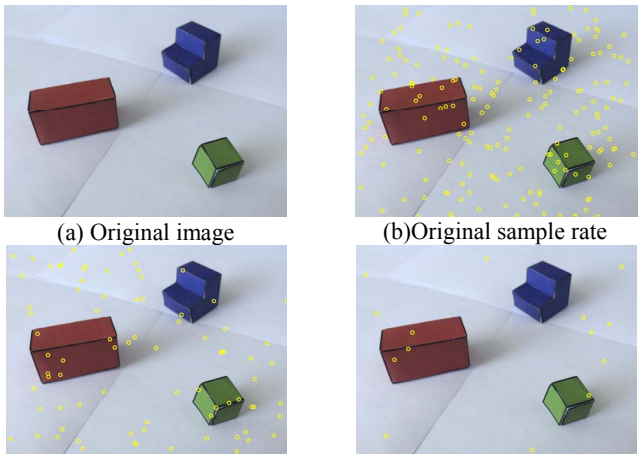


(a) Original image (b) DoG between σ=3×1.6 and σ=2×1.6

(c) DoG between σ=2×1.6 and σ=1.6 (d) DoG between σ=3×1.6 and σ=2×1.6

Figure 7: Edge detection using Difference of Gaussian

The feature point candidates are found in Figure 7a. Figures 7b, 7c and 7d show local extremas found at different scales. By down sampling the image, keypoints at different scale can be located.



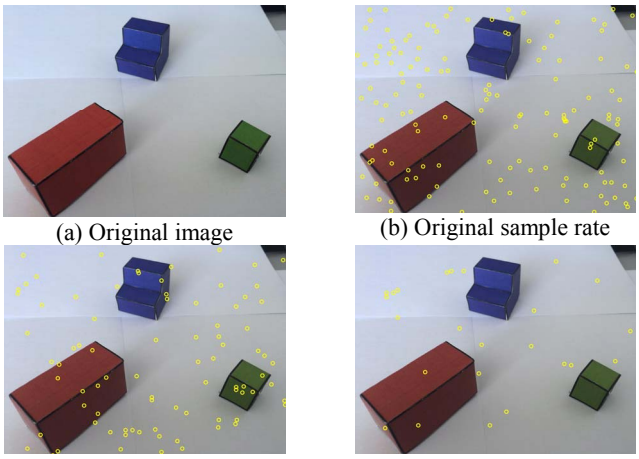
(a) Original image (b)Original sample rate

(c) 50% down sample (d) 25% down sample

Figure 8: Feature point candidates from first view point

Figure 8a is a photo taken with different angle. Similar to Figure 7, feature point candidates are found in Figure 8a. Figures 8b, 8c and 8d show local extremas found at different scales.

Using the feature points located by the algorithm we implemented, the reconstruction of the component drawer was conducted using Matlab script and code provided by Jianxiong Xiao [6], which can map the feature points in 3D space and further render these points with correct color.



(a) Original image (b) Original sample rate

(c) 50% down sample (d) 25% down sample

Figure 8: Feature point candidates from second view point

VI. CONCLUSION

This study demonstrates the ability of 3D mapping for motion estimation. The proposed method is capable of extracting 3D information from videos taken by single cameras accurately. The process of 3D mapping is computationally heavy. This can be resolved by implementing the 3D mapping algorithm on FPGA platforms. With better computational performance, the drone can perform 3D mapping at real-time rate or achieve path planning. Hence the drone can capture video of objects and reconstruct their 3D model automatically.

REFERENCES

[1] S. Y. Bao, M. Bagra and S. Savarese, "Semantic structure from motion with object and point interactions," *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops),* pp. 982-989, 2011.

[2] A. Saxena, M. Sun and A. Ng, "Make3D: Learning 3D Scene Structure from a Single Still Image," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 5, pp. 824-840, 2009.

[3] N. Snavely, S. M. Seitz and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *ACM Trans. Graph.,* vol. 25(3), pp. 835-846, 2006.

[4] J. Kopf, M. F. Cohen and R. Szeliski, "First-person Hyper-lapse Videos," *Proceedings of ACM SIGGRAPH 2014 ACM Transactions on Graphics (TOG),* vol. 33, no. 4, pp. 78:1-10, 2014.

[5] J. Xiao and L. Quan, "Multiple view semantic segmentation for street view images," *2009 IEEE 12th International Conference on Computer Vision,* pp. 686-693, 2009.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," vol. 60, no. 2, pp. 91-110, 2004.

[7] R. Hartley and A. Zisserman, Richard Hartley and Andrew Zisserman, Cambridge University Press, ISBN: 0521540518, 2004.

[8] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22(11), pp. 1330-1334, 2000.