# STRUCURE AND MOTION RECOVERY USING VISUAL SLAM

*Adarsh Kowdle, Yao-Jen Chang, Tsuhan Chen*

Department of Electrical and Computer Engineering
Carnegie Mellon University
akowdle@andrew.cmu.edu, kevinchang@cmu.edu, tsuhan@cmu.edu

## ABSTRACT

*A lot of applications in the domain of computer vision require recovering structure and motion from a set of 2D images. In order to ease this process, an ongoing effort is to create a structure from motion (SFM) library[1]. This library would package together, feature extraction and tracking algorithms like SIFT and KLT; structure from motion modules like factorization and bundle adjustment, thereby allowing any application to suitably use them. A useful addition to this library is* ***visual simultaneous localization and mapping****. Visual SLAM, allows one to simultaneously obtain a map of an unknown environment and locate an uncontrolled monocular camera within the environment. The aim of this effort is to integrate this visual SLAM algorithm into the SFM library.*

***Keywords*** – SIFT, Structure from motion, Kalman Filter

## 1. INTRODUCTION

There are a lot of ongoing efforts in the domain of SLAM and real-time monocular SLAM in particular. Some significant contributions in this domain are by Davison *et al.* [1]. Our efforts involved understanding the current state of the technology and to suitably incorporate it in the library.

The report has been structured to first give an overview of the basic simultaneous localization and mapping methodology in Section 2. Monocular SLAM which is based on the basic SLAM methodology is then explained followed by a more efficient, unified inverse depth parameterization model for MonoSLAM. This is followed by a summary of experiments and results in Section 3 and Conclusions.

[1] *SFM library is an ongoing project in the Advanced Multimedia Processing(AMP) lab at Carnegie Mellon University*

## 2. OVERVIEW OF CONCEPTS

### 2.1. Simultaneous Localization and Mapping (SLAM)

Let us consider the classic case of a robot placed in a known environment. In this case, a map of landmark locations within the environment is known a priori. The robot can thus proceed to take measurements of the known landmark locations, and use these measurements to determine its own location within the environment. This is known as the *localization* problem. A counter case would be one wherein the motion of the robot within the environment is known a priori. The robot could then take measurements of the environment and build up a map of the environment from these measurements. This is known as the *mapping* problem [2].

SLAM aims to solve the localization and mapping problems together, i.e. we could place a robot in an unknown location in an unknown environment and have the robot incrementally build a map of this environment while simultaneously using this map to compute its own location. This is achieved by application of a landmark extraction algorithm and a Kalman filter loop.

A significant part of SLAM is the ability of the robot to recognize and initialize key features from the environment [3]. These features should satisfy certain important conditions: they should be stationary, easily re-observable, and easily distinguishable from each other. A feature extraction algorithm such as Harris corner detection or the Shi & Tomasi algorithm is used to locate features in the environment which satisfy these conditions.

The heart of the SLAM process lies in the Kalman filter loop, which maintains the position and uncertainty of the robot and the landmarks at any given time The estimated position of the robot as well as the estimated positions of the landmarks is put together into a state vector, and in parallel, a covariance matrix representing the ambiguity or uncertainty of each of these positions is also maintained. The Kalman filter works in three distinct steps: predict,

measure and update. In the *prediction stage*, the state vector and covariance matrix for the next time instant are predicted using the current state and the motion model. In the *measurement stage*, the actual measurement of range and bearing is made; usually using an on-board device like an odometer. In the *update stage*, based on the error between the prediction and the measurement, the state vector and the covariance matrix are updated using the kalman gain [4].

In addition to the uncertainty in the position of the robot and feature points, the covariance matrix also represents the covariance between the feature points. Due to the nature of the kalman loop, the error and thus the kalman gain reflects on the entire covariance matrix. Thus, the uncertainty of the system about the locations of each feature point keep decreasing as the system observes more of the environment.

### 2.2. Monocular SLAM

A successful application of the SLAM methodology has been introduced by Davison *et al.* [1] where a real-time algorithm which can recover the 3D trajectory of a monocular camera has been described.

The key in this application is modeling the uncontrolled motion of the camera. As this is a freely moving camera, a linear motion model would not work. In order to account for this, a constant velocity, constant angular velocity model is assumed. This imposes some smoothness into the motion and assumes that large abrupt accelerations are unlikely. In order to account for this non-linear motion model, the extended kalman filter (EKF) takes the place of the kalman loop. EKF functions along the same lines as the kalman filter with the non-linear motion model being the only difference.

The state of the system as explained before is described by the state vector and covariance matrix. The state vector in this case however has to describe the state of the camera in 3D and the positions of the feature points. The state of the camera contains 13 parameters – 3 parameters describing the position of the camera, 4 parameters representing the quaternion describing the orientation of the camera and 6 parameters describing the linear velocity vector and angular velocity vector. The feature points are defined by their 3D locations with respect to the world. The whole system is described as a probabilistic 3D map modeled by a multi-variate gaussian with the mean as the state vector and variance as the covariance matrix. The state vector and covariance matrix are maintained over the whole motion of the camera as the main intent of the application is closed loop mapping, i.e. being able to recognize a feature which was initialized a long time back.

MonoSLAM is derived from the SLAM methodology and thus works along the same lines. The feature initialization however, is different. The drawback of this approach is that it requires a template at the system initialization step to describe the scale of measurement and initialize the EKF process. When the system is short of a minimum number of feature points it decides to initialize new features. The features points are detected using Shi and Tomasi's feature detection algorithm. The features at this stage are however only partially initialized and thus are not added to the EKF map. At this stage, the system has an estimate of the epipolar line along which the feature lies, with low uncertainty but, does not have any information about the depth. An assumption made at this stage is to assume that the figure lies at a distance of 0.5m to 5m from the camera. Over the next few frames, depth hypotheses evenly distributed along the epipolar line from 0.5m to 5m are tested to finally narrow down on a depth value which gives highest amount of correlation for that feature. The feature is then assumed to be fully initialized and becomes part of the EKF map.

Once the feature is initialized, the rest of the process is standard EKF procedure. The prediction stage involves the prediction of the next state of the camera and feature points. The measurement stage involves performing a correlation search around the predicted region to find the actual position of the features. The update stage is updating the state vector and covariance matrix based on the error in prediction.

### 2.3. Unified inverse depth parameterization

A monocular camera measures the bearing of image features. As explained in the previous sections, to infer the depth of a feature the camera must observe it repeatedly as it translates through the scene, each time capturing a ray of light from the feature to its optic center. The angle between the captured rays is the feature's *parallax* – this is what allows its depth to be estimated. For instance, a star in the sky exhibits very low parallax even in the presence of significant motion due to its extreme depth. This parallax has been studied and modeled to obtain an estimate of the depth, using inverse depth parameterization [5].

A limitation of Davison's MonoSLAM approach as highlighted by Montiel *et al.* [5] is that, it makes use of features which are close to the camera, which exhibit significant parallax and would therefore work in room-scale scenes. Another limitation put forth, refers to the feature initialization required in Davison's approach [1]. The delayed feature initialization which requires the feature to be visible for a few frames before it is considered as a fully initialized feature means that,

observing these features would not help update the pose of the camera till it is fully initialized.

The unified inverse depth parameterization overcomes these limitations by defining a way of representing a point at infinity. The idea is that, no observable feature is truly infinitely far from the camera. A point at infinity is simply far enough away relative to the camera motion since it has been observed that no parallax has been observed. This method requires no feature initialization, the features are added to the map right from the point it is first viewed and the standard EKF loop takes care of updating its position and the pose of the camera.

The fundamental operation of the inverse depth algorithm is similar to other MonoSLAM implementations, in that, the standard EKF loop is used to converge to the location of the feature. The main difference however, is the representation of the feature points which allows for modeling points at extreme depth.

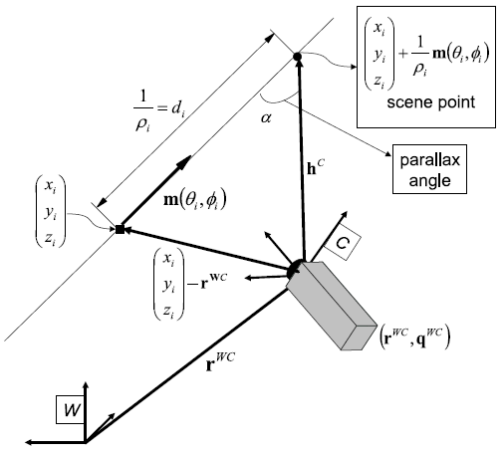


Fig. 1: Feature parameterization and measurement equation

The position of each new 3D feature *i* added to the map is parameterized by a six dimension state vector (Fig. 1).

$$Y_i = (x_i, y_i, z_i, \theta_i, \varphi_i, \rho_i)^T$$

The state codes the ray for the first point observation as: $(x_i, y_i, z_i)$, the camera optical center where the 3D point was first observed; $\theta_i$ and $\varphi_i$, the azimuth and elevation for the ray directional vector $\mathbf{m}(\theta_i, \varphi_i)$. The point depth along the ray $d_i$ is coded by its inverse $\rho_i = 1/d_i$. In other words, every feature point is modeled as follows.

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i)$$

Using the above representation, a point even at extreme depth can be modeled when the inverse depth ρi tends to zero.

The working of this algorithm is very similar to Davison's implementation. Consider the above representation as analogous to a fully initialized feature in the camera state vector. The inverse depth for the feature is initially set to a low value. The EKF process suitably corrects it to finally converge to an appropriate depth and position, as the feature is re-observed over the sequence of images.

### 2.4. Integration with the structure from motion library

Simultaneous localization and mapping, MonoSLAM in particular, provides for a significant addition to our structure from motion library. MonoSLAM allows one to obtain the trajectory of a *freely moving* monocular camera. It allows for drift free performance which outperforms structure from motion (SFM). Integrating MonoSLAM with our SFM library would thus greatly benefit for certain applications.

The integration of MonoSLAM with the rest of the library is achieved by using the SIFT feature trajectory obtained from the SFM library. The library returns the trajectory of all the SIFT features as its output. The features found in every frame can replace the feature initialization step in the MonoSLAM system. Further, the entire matching process which is a key step of the EKF loop, can be removed by using the SIFT feature trajectory. The results from this integration are described in the next section.

## 3. EXPERIMENTS AND RESULTS

### 3.1. Structure and motion viewer

In order to obtain a good visualization of the camera trajectory and the 3D locations of the feature points, a Structure and motion (SAM) viewer, has been developed by Tzung-Han Lin[2] from ITRI.

The viewer accepts the intrinsic matrix of the camera; the extrinsic matrix which gives the position of the camera with respect to the world for every frame. The viewer also accepts the set of 3D feature point locations and the color of these feature points. Using these parameters the viewer gives a 3D scatter of the feature points along with the camera positions and trajectory. This viewer has been used in all our experiments.

[2] *Tzung-Han Lin from Industrial technology and research institute (ITRI), Taiwan worked with the AMP lab at CMU and developed the SAM viewer*

Fig. 2

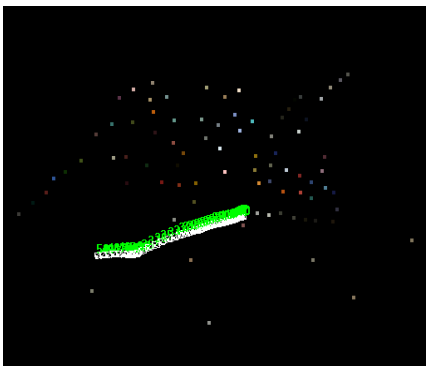

Fig.3(a)

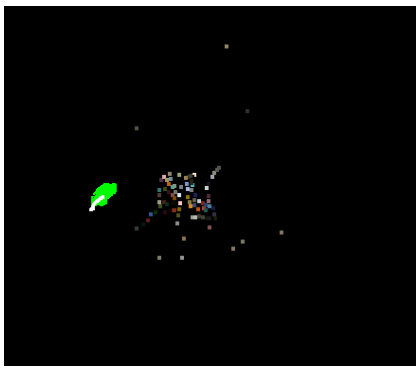

Fig.3(b)

Fig.2: Image from the test video sequence; Fig.3: (a) Camera trajectory (where each camera position has been labeled in green) and feature point scatter as seen in the viewer. (b) Another view of the scatter for better 3D visualization.

### 3.2. Inverse depth parameterization

The motivation behind this is to study and understand the impact of inverse depth parameterization on an outdoor scene with feature points even at considerable depth. The experiments were carried out using a matlab code developed by Montiel *et al*. obtained from the SLAM summer school (2006) database [6]. The system was incomplete in a sense, as it lacked any form of feature extraction. The system required manual initialization of features. However, in order to complete the system, we would require some form of feature initialization in the system.

The integration of the system with the SFM library was found to fill in this gap. As explained in the earlier section on the SFM library, the SIFT feature trajectory can be used to integrate the two systems together. The SIFT feature trajectories obtained from the library, gives the frames between which a particular feature is observed, along with the positions of these key-points in each frame. The SIFT feature trajectories were used to initialize new features in the system. The trajectory of these features, across the frames it is visible was used to replace the matching process of the EKF loop, thus integrating the two systems together.

The matlab code-base was modified to accept any video sequence from the user. The output parameters of the system were then modified as required by the viewer. One of the images used in our experiment is shown in Fig. 2, the some snapshots of the output as seen in the viewer is shown in Fig. 3a and Fig. 3b. (Note: In order to provide a better visualization of the object in view a specific set of feature points have been selected here)

## 4. CONCLUSION

The results from the experiments show that, the SIFT features obtained from the SFM library can be used with the EKF loop of a SLAM methodology to achieve efficient real time localization and mapping. The integration was done offline, by using the SIFT trajectories in the matlab code-base of monocular SLAM. Integrating the MonoSLAM codebase directly into the SFM library can result in a unified solution for simultaneously recovering the structure of the environment and the motion of the camera. The next phase of work would be a real-time implementation of the process described in the paper.

## 5. REFERENCES


[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse "MonoSLAM: Real-Time Single Camera SLAM," IEEE transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 6, 2007.

[2] H. Durrant-Whyte, D. Rye, E. Nebot, "Localisation of Automatic Guided Vehicles", ISRR 1995

[3] S. Riisgaard, and M. R. Blas, "SLAM for dummies".

[4] Drexel University Lectures, "Simultaneous Localization and Mapping (SLAM)"

[5] J. M. M. Montiel, J. Civera and A. J. Davison, "Inverse Depth Parametrization for Monocular SLAM," RSS 2006.

[6] http://www.robots.ox.ac.uk/~SSS06/, "SLAM using monocular vision", practical session at the SLAM summer school, 2006.