Springer manuscript No.
(ENPM667 Project 1)

Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM

ENPM667 Midterm Project Report

Harsh Kakashaniya · Rohitkrishna Nambiar

Submitted: 19th November 2018

Abstract The fusion of inertial and visual data is widely used to improve an objects pose estimation. However, this type of fusion is rarely used to estimate further unknowns in the visual framework. In this paper we present and compare two different approaches to estimate the unknown scale parameter in a monocular SLAM framework. Directly linked to the scale is the estimation of the objects absolute velocity and position in 3D. The first approach is a spline fitting task adapted from Jung and Taylor and the second is an extended Kalman filter. Both methods have been simulated offline on arbitrary camera paths to analyze their behavior and the quality of the resulting scale estimation. We then embedded an online multi rate extended Kalman filter in the Parallel Tracking and Mapping (PTAM) algorithm of Klein and Murray together with an inertial sensor. In this inertial/monocular SLAM framework, we show a real time, robust and fast converging scale estimation. Our approach does not depend on known patterns in the vision part nor a complex temporal synchronization between the visual and inertial sensor.

Keywords IMU vision fusion · Absolute scale · Monocular SLAM · Kalman filter

Harsh Kakashaniya
UID:
E-mail: harshbk@umd.edu

Rohitkrishna Nambiar
UID: 115507944
E-mail: rohit517@umd.edu

Contents

List of Figures

1 Introduction

On-board pose estimation is very useful for a variety of applications ranging from autonomous robotics to augmented reality. Pose estimation can be done using different sensors such as cameras and inertial measurement units (IMU). Simultaneous localization and mapping (SLAM) using a monocular camera can be used to get the trajectory of the camera and a map of the environment. But monocular cameras suffer from scale ambiguity causing the overall trajectory to drift making it unusable in real time. Using a stereo camera helps solve the scale ambiguity. IMUs on the other hand can be used to get the trajectory traveled by integrating the acceleration measurements over time. But this leads to highly inaccurate trajectory estimates. The estimate of the scale factor is essential to fuse both these measurements. This fusion helps us determine the unknown scale factor λ.

In this paper, two methods are presented for scale estimation. The first one is a spline fitting method by Jung and Taylor [1]. The second is a multi rate Extended Kalman Filter(EKF). Both the approach have been simulated in MATLAB.

This report is organized as follows. Section 2 goes over the camera and imu setup, Section 3 covers the camera and image formation process, Section 4 goes over the scale estimation problem with monocular camera, Section 5 outlines the spline fitting method with results.

2 Hardware Setup

This section goes through the setup used in the research paper. USB uEye UI-122xLE fisheye camera was used as the vision input. The camera has a resolution of 752 480 and a frame rate up to 87 fps. Motion blur was minimized due to the high dynamic range and global shutter of the camera. VG400CC-200 solid state gyro which includes a tri-axial gyroscope and tri-axial accelerometer was used. It has an output frequency of 75 Hz with an input range of $\pm$10g and ¡1.25 mg resolution. The IMU outputs acceleration around its 3-axis along with yaw, pitch and roll.

3 Camera and image formation process

To understand how a camera perceives the environment, it is necessary we understand the image formation process. For us humans, we see things when a light originating from a light source is reflected on an object and enters our eyes. A camera acts very much similar to the human eye. The earliest and first model of an optical camera is the pinhole camera which is a simple and highly accurate representation of our eye model. This is the simplest device to form an image of a 3D scene on a 2D surface. As seen in figure 1 rays of light enter the pinhole and forms an inverted image of the object. This is called

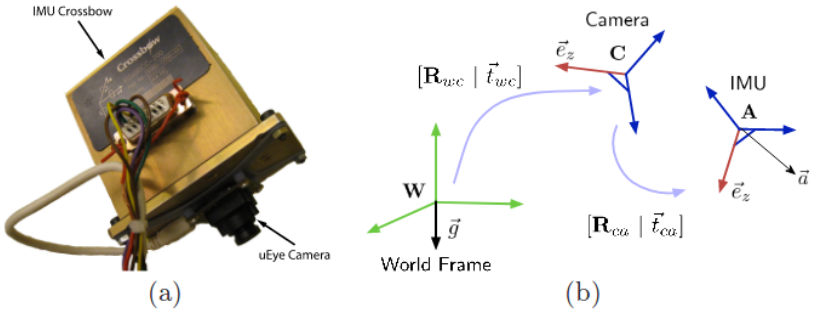

Fig. 1 **a.** Camera/IMU setup. **b**. Frame transformations

perspective projection. As the image formed in the image place is inverted, we consider a virtual plane in front of the pinhole that acts as the image plane.

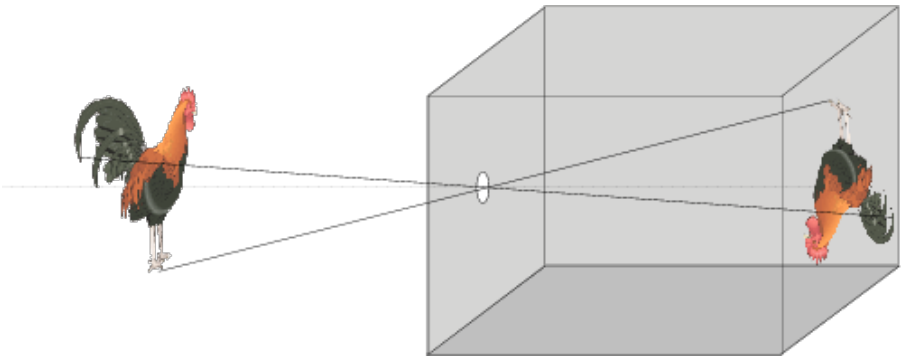

Fig. 2 Please write your figure caption here

Although the pinhole model is quite accurate, modern cameras have lenses. They help gather more light from a source leading to higher quality sharper images. Ignoring the internal diffraction, we assume thin lens equations for the imaging process. Every camera has a region of depths over which the scene is sharp. Modern cameras have variable apertures which help in focusing on objects at varying distances in the scene.

Every camera has an intrinsic, extrinsic and distortion parameters that are important to understand for every application. These parameters are used to correct for lens distortion, measure objects in physical world and to determine the location of the camera in the real world. Camera calibration is the process of estimating these parameters specific to a given camera and application setup. Figure 2 demonstrates how the a real world 3D coordinate is related to a 2D pixel coordinate using the parameters mentioned above.

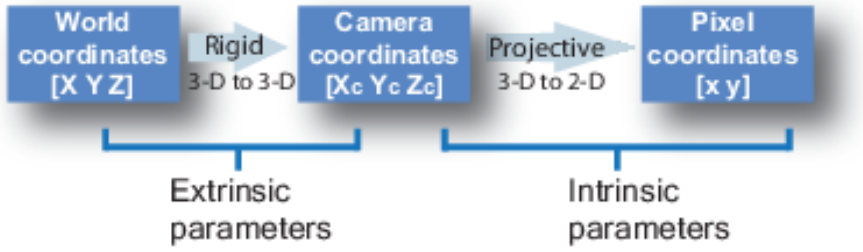

Fig. 3 Please write your figure caption here

The extrinsic parameters relate the rotation R and translation t of the camera origin at the optical center to the world frame. The intrinsic parameters consists of the focal length given by fx and fy, the optical center given by cx and cy and the skew coefficient s. The distortion parameters consists of radial and tangential distortion along the x and y direction respectively. The number of coefficients are decided based on the lense in consideration. Camera calibration techniques presented in [4] [5] and [6] are widely used in commercial and open source camera calibration tool boxes.

4 What's wrong with monocular camera?

As the name suggests a monocular camera is a camera with a single imaging sensor. In section X. we say that given a perfectly calibrated camera ie. with the intrinsic and extrinsic parameters known, we can measure size of objects in the real world. This however is not true for a single image taken from a monocular camera. When 3D objects are projected onto the image plane (2D), the depth information stored in the z-axis is lost. This can be seen in the example below.

As seen in the above diagram, irrespective of the location of the object in the real world, the size of the object in the image plane remains the same. Mathematically, this can be shown as

$$\frac{y}{f} = \frac{y_1}{z_1 + f} = \frac{y_2}{z_2 + f} \tag{1}$$

Where f is the focal length of the camera. The title of the paper mentions scale estimation. In the context of SLAM which will be covered in the next section, scale is a term used to denote the factor by which the computed trajectory by a SLAM algorithm needs to be multiplied/scaled to make it equal to the ground truth. For example, if the measured distance is 2m, whereas the ground truth is 4m, the scale value would be 2. This is an area of active research and the community has come up with different techniques to solve

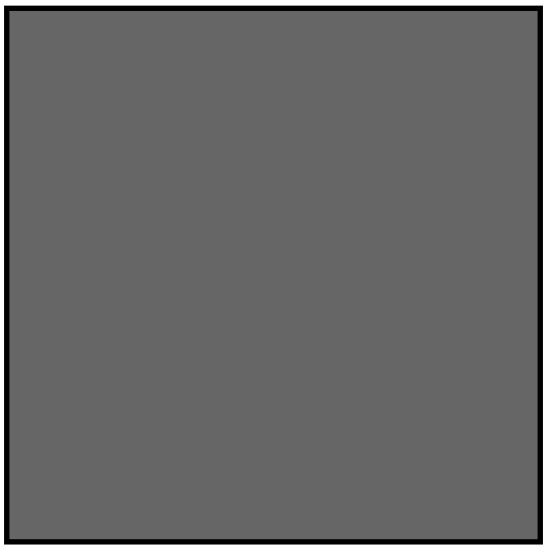

Fig. 4 Monocular camera image coordinates

this problem. As the title suggests, we would be talking about the fusion of IMU and Vision for solving this challenge.

5 Spline Fitting Method

In this paper there are 2 methods of fusing IMU data and Monocular camera. First of which is spline fitting using scale factor with the data of IMU and Monocular camera. Depending on the reliability of the data from both sources to estimate distance.So here we did simulation of whole system with arbitrary value of data considering linear moment of robot with noisy data. These simulations can be seen in result section. With the matlab code. So initially theory behind curve fitting is explained to have a algebraic background of types of curve fitting and their applications.We use spline in curve fitting because we get discrete data from sensors and camera but tracing spline helps for interpolation and extrapolation to have continuous high probability data of path.

Spline is fitted on the data points which is output of sensor in our case. Spline fitting also helps in reducing error due to noise. For Improving accuracy we can increase number of spline but this will also result into increase in computation. So we have to trade off between accuracy and computation. Spline fitting is useful to give smooth curve .N degree spline may has N-1 curves in its shape. So if spline has X3 term that means it can have 2 curves in its spline.Ideal method to cover n points in a plan is by tracing nth order spline which will result into zero error. So spline will pass through all the points and will have a general equation as

$$A_0 + A_1X + A_2X^2 + A_3X^3 + ... + A_nX^n = Y \tag{2}$$

This problem then converts in simple Ax = B form. we can get values of n terms by plugging in n points values.

Where

$$A = \begin{pmatrix} 1 & X_1 & X_1^2 & \cdots & X_1^n \\ 1 & X_2 & X_2^2 & \cdots & X_2^n \\ 1 & X_3 & X_3^2 & \cdots & X_3^n \\ : & : & : & \cdots & : \\ 1 & X_n & X_n^2 & \cdots & X_n^n \end{pmatrix} B = \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ : \\ A_n \end{pmatrix} C = \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ : \\ Y_n \end{pmatrix} \tag{3}$$

But for practical applications we use other methods and do not plot n degree spline as it is computationally expensive .Secondly if there is more noise in system it will give wrong results. For practical application there are 3 kinds of spline fitting according to the application:-

5.1 Types of curve fitting

5.1.1 Maximum error

This is the method where spline is fit in a way where the point with maximum error is reduced and curve move towards the outlier point. If there exist in order to reduce maximum error of a point. This on the other hand results into

$$E(f) = |max(f(xi) - yi)|whereigoesfrom1toleftn \tag{4}$$

Actually due to one outlier here line shifted in order to reduce maximum error. So this is not reliable process but is easy to compute.

5.1.2 Average error

This method is used to minimize average and fit the spline with minimum error conditions.

$$E(f) = \frac{1}{n} * \Sigma|f(x_k) - Y_k| \tag{5}$$

Actually due to one outlier here line shifted in order to reduce maximum error. So this is not reliable process but is easy to compute.

5.1.3 RMS (Least square method)

This is the most efficient method to minimize the error this is also called as least square method. Hence this is mostly used in curve fitting. We have also created an algorithm with the help of same type of curve fitting.

$$E(f) = \{\frac{1}{n} * \Sigma|f(x_k) - Y_k|\}^{\frac{1}{2}} \tag{6}$$

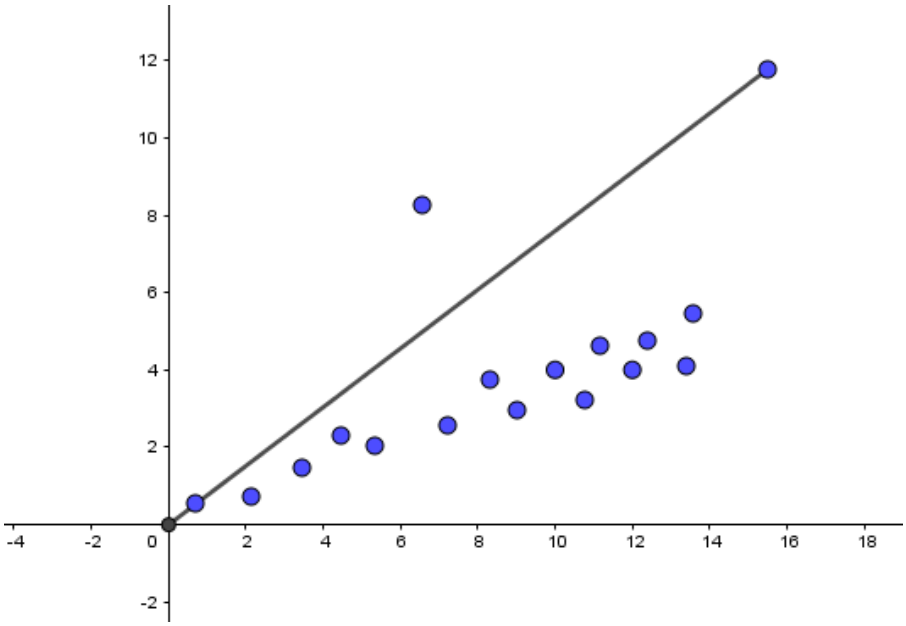

Fig. 5 Curve Fitting with Maximum error

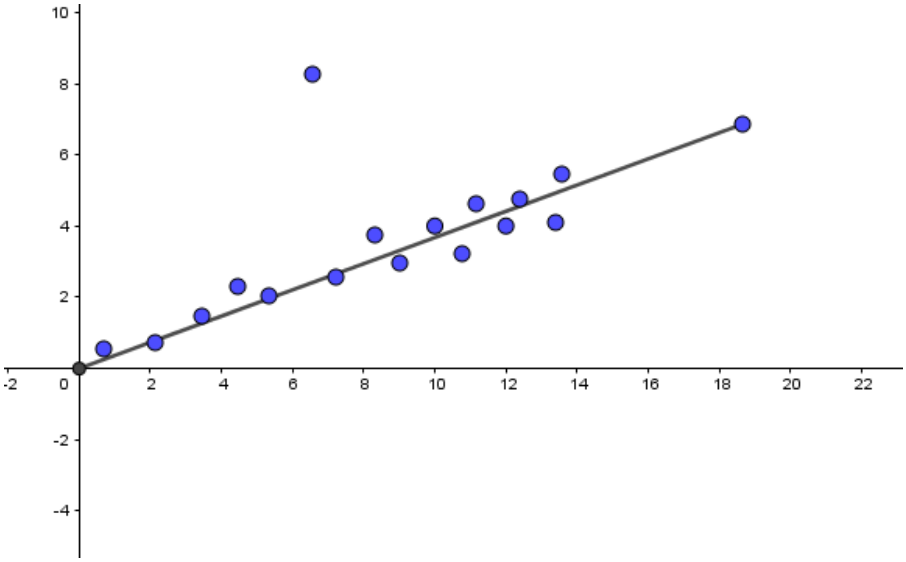

Fig. 6 Curve Fitting with Average error

5.2 IMU Reading

As we dont have real IMU lets simulate virtual readings for this we took X, Y, Z as a function on time

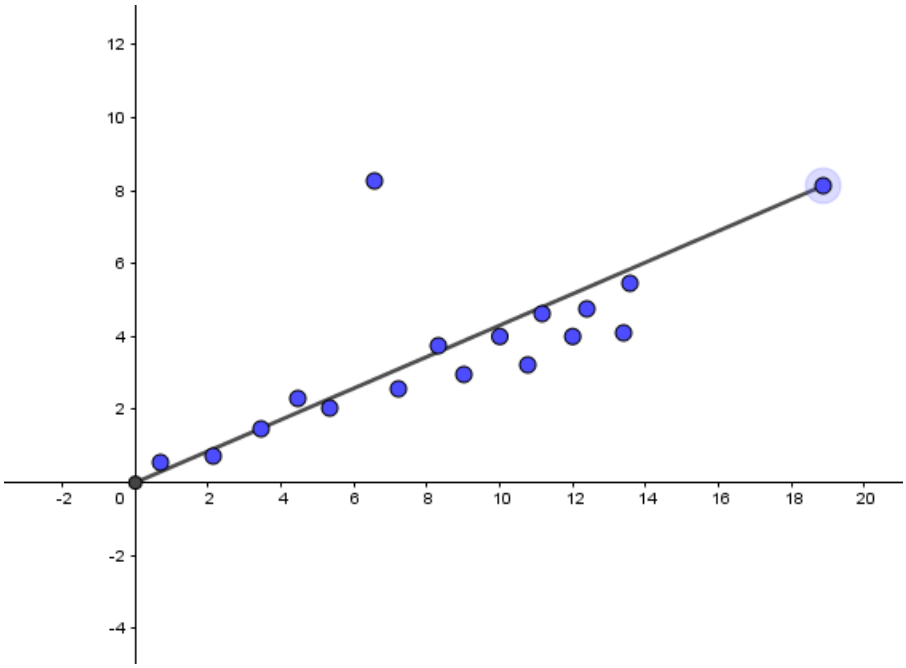

Fig. 7 Curve Fitting with RMS error

$$X_1 = A_y t^2 + B_y t + C_x \tag{7}$$

$$Y_1 = A_y t^2 + B_y t + C_y \tag{8}$$

$$Z_1 = A_z t^2 + B_z t + C_z \tag{9}$$

With this treatment we will add some random noise to all the terms so that the data can be similar to what we obtain form IMU.

$$noise = A * (rand(1, n) - 0.5) \tag{10}$$

So our noise will range from -A/2 to A/2. As function is rand() so it will change for every point.

$$X = X_1 + noise \tag{11}$$

$$Y = Y_1 + noise \tag{12}$$

$$Z = Z_1 + noise \tag{13}$$

We also took some random points of camera pose. And provided our system with data set. Proof of Least square method in our case: Lets consider optimization of X first.

From equation (7) we have

Here A, B, C are known We have data set of X with respect to t To calculate error in RMS we have

$$Error_x = \{\sum_{i=1}^{n}(|X_i - (A_x t_i^2 + B_x t_i + C_x)|)^2\}^{\frac{1}{2}} \tag{14}$$

When we differentiate some quantity and equate it to zero it goes to either minima or maxima And if double derivative is positive it goes to minima.

So in this case if we take partial derivatives of error with respect to Ax , Bx , Cx

$$\frac{\partial Error_x}{\partial A_x} = \{\sum_{i=1}^{n} 2(|X_i - (A_x t_i^2 + B_x t_i + C_x)|)\}(-t_i^2) \tag{15}$$

$$\sum_{i=1}^{n}(|-t_i^2 X_i + (A_x t_i^4 + B_x t_i^3 + C_x t_i^2)|) = 0 \tag{16}$$

$$\frac{\partial Error_x}{\partial B_x} = \{\sum_{i=1}^{n} 2(|X_i - (A_x t_i^2 + B_x t_i + C_x)|)\}(-t_i) \tag{17}$$

$$\sum_{i=1}^{n}(|-t_i X_i + (A_x t_i^3 + B_x t_i^2 + C_x t_i^1)|) = 0 \tag{18}$$

$$\frac{\partial Error_x}{\partial C_x} = \{\sum_{i=1}^{n} 2(|X_i - (A_x t_i^2 + B_x t_i + C_x)|)\}(-1) \tag{19}$$

$$\sum_{i=1}^{n}(|-1 X_i + (A_x t_i^2 + B_x t_i + C_x)|) = 0 \tag{20}$$

So we can compute matrix in such a way that the problem changes to

Ax=B

Where,

$$A = \begin{pmatrix} \Sigma t^4 & \Sigma t^3 & \Sigma t^2 \\ \Sigma t^3 & \Sigma t^2 & \Sigma t \\ \Sigma t^2 & \Sigma t & \Sigma 1 \end{pmatrix} B = \begin{pmatrix} A_x \\ B_x \\ C_x \end{pmatrix} C = \begin{pmatrix} \Sigma X_i t^2 \\ \Sigma X_i t \\ \Sigma X_i \end{pmatrix} \tag{21}$$

From this equation we can calculate X matrix.Similarly,We can get X matrix of Y and Z axis.

$$A = \begin{pmatrix} \Sigma t^4 & \Sigma t^3 & \Sigma t^2 \\ \Sigma t^3 & \Sigma t^2 & \Sigma t \\ \Sigma t^2 & \Sigma t & \Sigma 1 \end{pmatrix} B = \begin{pmatrix} A_y \\ B_y \\ C_y \end{pmatrix} C = \begin{pmatrix} \Sigma Y_i t^2 \\ \Sigma Y_i t \\ \Sigma Y_i \end{pmatrix} \tag{22}$$

$$A = \begin{pmatrix} \Sigma t^4 & \Sigma t^3 & \Sigma t^2 \\ \Sigma t^3 & \Sigma t^2 & \Sigma t \\ \Sigma t^2 & \Sigma t & \Sigma 1 \end{pmatrix} B = \begin{pmatrix} A_z \\ B_z \\ C_z \end{pmatrix} C = \begin{pmatrix} \Sigma Z_i t^2 \\ \Sigma Z_i t \\ \Sigma Z_i \end{pmatrix} \tag{23}$$

By this treatment we converted given points into equation with least square method here we used condition of

$$min \begin{pmatrix} A_x t^2 + B_x t + C_x - \lambda_i X_c \\ A_y t^2 + B_y t + C_y - \lambda_i Y_c \\ A_z t^2 + B_z t + C_z - \lambda_i Z_c \end{pmatrix}^2 \tag{24}$$

Here in this formula we have different value of for different spline according to accuracy of camera data and IMU reading.

5.3 Results

In our case we simulated the results and found following outputs. Plotting the second order curve for given data of IMU with just least square method. We get

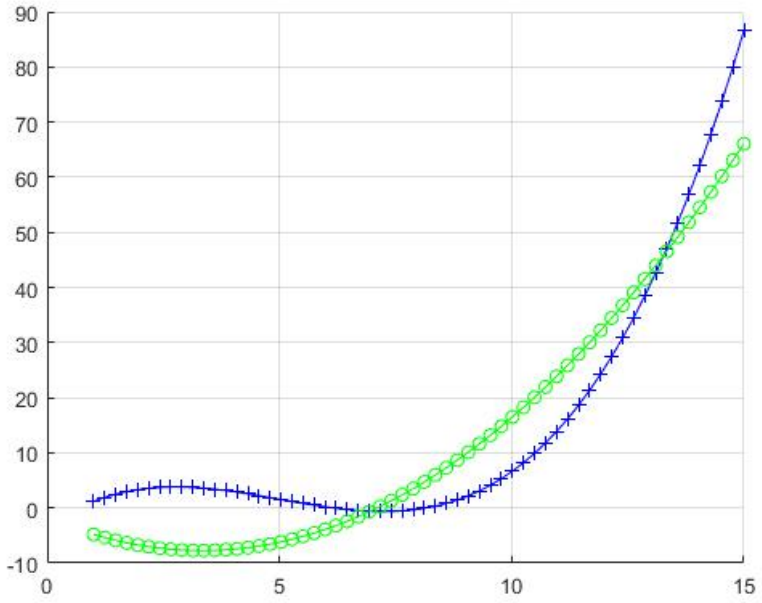

Fig. 8 In this case Blue is actual curve traced by IMU this are points with noise.And green is plotted curve. With least square method.

Then according to the paper we fused data of camera and IMU so we get following graph.

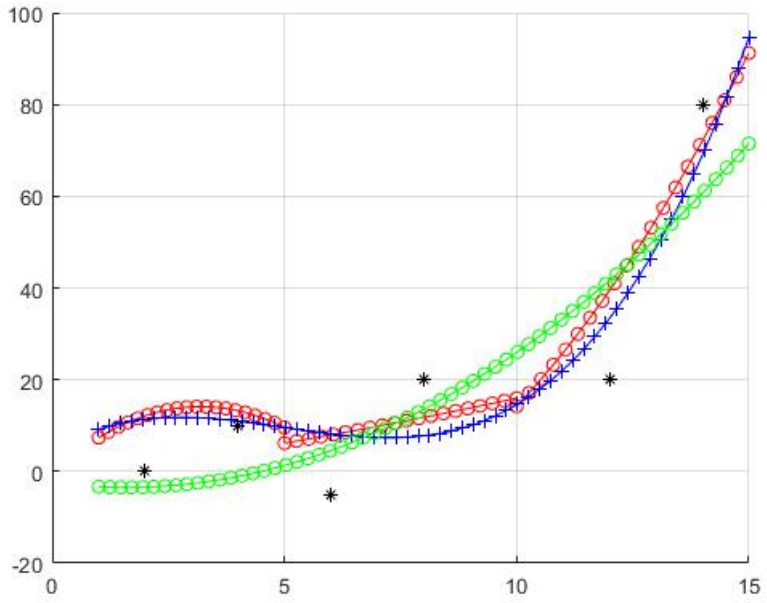

Fig. 9 Multiple spline fit

In Figure 9 we have 3 curved fitted as shown as red which gives better results and it also uses data of camera shown with black dots. So here according to the above formula scale factor of different section is different it is according to the quality of reading by camera. For example, in our case

$$\lambda_1 = 0.5, \lambda_2 = 0.5, \lambda_3 = 0.1 \tag{25}$$

In 3rd section camera readings were not accurate so taken into smaller scalar value. This camera data further improved the result. And we get minimum error and good fit. With the given spline fitting method.

So if we look into different of error with traditional Least square method with Jung and Taylor method. It gives good results. Almost we get the same curve.

The graph in Figure 10 has error comparison with and without Jung and Taylor method of split curve fitting and scaling camera input. So it is better to follow the given first method in the paper. It results into considerable reduction in error which is clearly seen in graph.

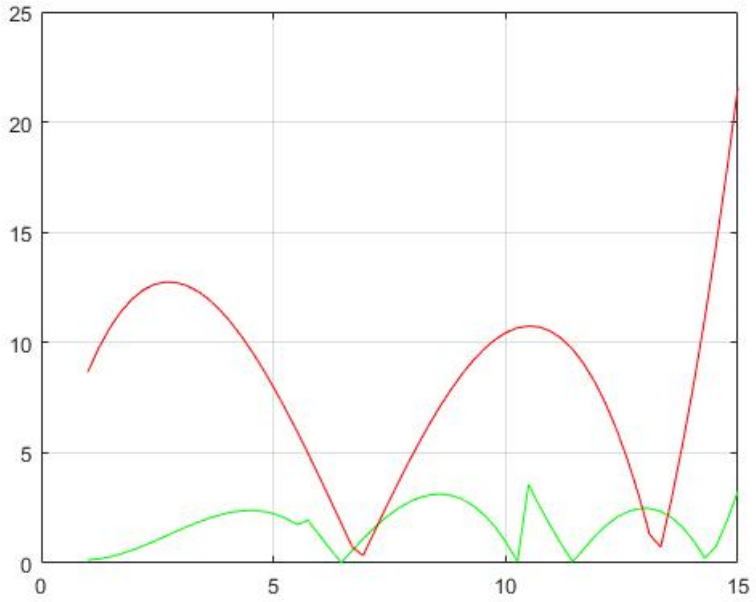

Fig. 10 A. Red curve for method with one curve and least square method. B. Green curve is using the given method by Jung and Taylor.

6 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a technique for estimating the motion of the robot and reconstructing the map/structure of the unknown environment. SLAM using only visual information only is specifically referred to as visual SLAM (vSLAM). The SLAM problem can be stated as follows:

How can a body navigate in a previously unknown environment while constantly building and updating a map of its workspace using onboard sensors only? [**7**]

From the problem statement above, we notice that the robot has no a priori knowledge of the workspace or environment that it is in. This makes SLAM a very challenging problem in probabilistic robotics. Accurate pose estimation is critical for many applications in computer vision, autonomous robotics and augmented reality.

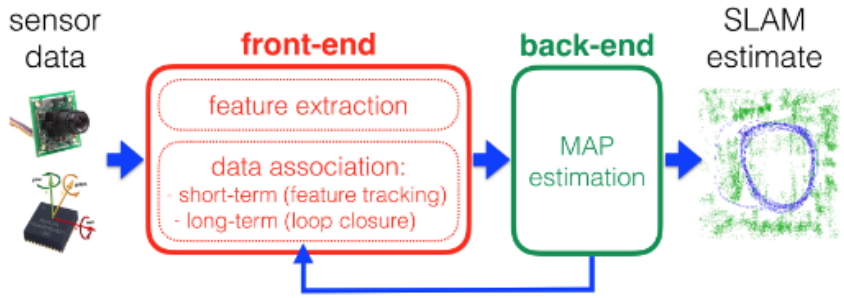

Fig. 11 SLAM architecture

6.1 Visual Odometry

As described in the above section, odometry is the technique of estimating the position of a robot over time using sensors such as cameras, wheel encoders or any sensor measuring relative movement. Compared to vSLAM which maintains a global consistent map, VO maintains a local consistent map optimized over the last n frames. A generalized VO pipeline can be summarized as follows

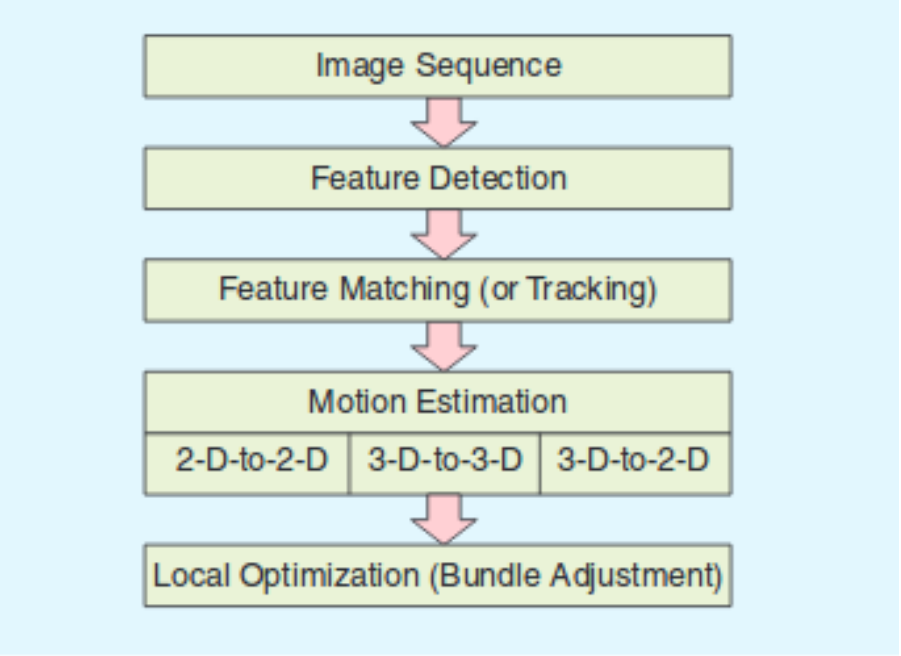

Fig. 12 Visual Odometry Pipeline

Different algorithms exist due to the different type of cameras available such as Stereo, Mono, RGB-D. As we use a monocular setup in our research paper, we would be going through the visual odometry algorithm from 2D to 2D correspondences for a monocular sensor. The algorithm is summarized as follows.

Algorithm 1 Monocular Visual Odometry

1: Capture a new image frame I_k
2: Extract and match features between frames I_k and I_{k-1}
3: Compute the essential matrix E from the matched feature points
4: Decompose E into R_k and t_k to form T_k
5: Choose the correct T_k matrix and scale t_k accordingly from scale factor
6: Compute $C_k = C_{k-1} * T_k$
7: Goto step 1.

Since we are using a monocular sensor, we always compare the image Ik captured at time instant k with the image I_{k-1} capture at time instant k-1. In a stereo setup, we would compare images from the left and right part of the stereo. Once an image is captured, we compute the features for image Ik and Ik-1. In the feature detection step, the image is searched for keypoints called features which are likely to be present in successive images. Features can be defined as image patterns that differ from its immediate neighbourhood in terms of intensity, color and texture. There exist a variety of feature detectors in literature which vary in performance and properties. Some of the properties of feature detectors are rotation, scale and affine invariant, repeatability, localization accuracy, robustness and efficiency. Heitanen et al. present a comparison of feature detectors and descriptors for object class matching in [12]. Once we detect features, we then need to compute a descriptor such that features detected in two images can be compared with each other. The easiest approach would be to form a path of pixels and compare using a sum of squared distances (SSD) or normalized cross correlation (NCC) metric. Such descriptors are not invariant to any of the above mentioned properties and perform poorly in practical applications. One of the popular descriptors called SIFT [13] uses gradient orientations as its descriptors. This forms a 128-element descriptor that is invariant to most of the above mentioned properties such as rotation, scale and illumination which makes it widely applicable in practical real-time applications. For extracting features from two images there are two paths with one being to detect and match features independently in two frames and the other being to track features in subsequent frames an example of which is a KLT Tracker [14]. There are various methods employed to match features accurately such a RANSAC which stands for Random Sample Consensus and is used to remove outliers among feature matches. Once we detect and match features in Ik and Ik-1, we calculate the essential matrix E which describes the geometric relationship between two images. We can use Nisters five point algorithm [15] or Longuet-Higgins eight point algorithm [16] to get the essential matrix E. Once we get the essential matrix E, we decompose it to extract the

rotation and translation parts. Four different solutions are obtained of which the correct pair can be found out by triangulation. The solutions are given as

6.2 EKFSLAM

6.3 PTAM

References

1. S-H Jung and Camillo J Taylor. Camera trajectory estimation using inertial sensor measurements and structure from motion results. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2001.
2. Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.