Visual Simultaneous Localisation and Mapping for a tree climbing robot

by

Benzun Pious Wisely Babu

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering

by

October 2013

APPROVED:

Professor Michael A. Gennert, Major Thesis Advisor, Head of Department

COMMITTEE MEMBERS:

Professor David Cyganski, Electrical and Computer Engineering, WPI

Assistant Professor Taskin Padir, Robotics Engineering, WPI

Abstract

This work addresses the problem of generating a 3D mesh grid model of a tree by a climbing robot for tree inspection. In order to generate a consistent model of the tree while climbing, the robot needs to be able to track its location while generating the model. Hence we explored this problem as a subset of Simultaneous Localization and Mapping problem. The monocular camera based Visual Simultaneous Localization and Mapping(VSLAM) algorithm was adopted to map the features on the tree. Multi-scale grid based FAST feature detector combined with Lucas Kande Optical flow was used to extract features from the tree. Inverse depth representation of feature was selected to seamlessly handle newly initialized features. The camera and the feature states along with their co-variances are managed in an Extended Kalman filter. In our VSLAM implementation we have attempted to track a large number of features. From the sparse spatial distribution of features we get using Extended Kalman filter we attempt to generate a 3D mesh grid model with the help of an unordered triangle fitting algorithm. We explored the implementation in C++ using Eigen, OpenCV and Point Cloud Library. A multi-threaded software design of the VSLAM algorithm was implemented. The algorithm was evaluated with image sets from trees susceptible to Asian Long Horn Beetle.

Acknowledgments

I would like to express my gratitude to Prof. Michael A. Gennert for his patience with me and providing valuable insight in helping me complete this thesis. I would also like to thank my sincere gratitude Prof. David Cyganski and Prof.Taskin Padir for being a part of my advisory committee. I would also like to thank Dr Mc Clint Farald of US Department of Agriculture for giving us better understanding of the problems they face in eradication of Asian Longhorn beetle. The numerous visits to USDA helped me better understand the requirements for this project better. I would like to thank Justin Gostanian, Erick Read, Ian Campbell, Eric Cobane, Ryan Giovacchini and Thomas Murray on their effort in developing the tree climbing robot as a part of their MQP. I would like to thank Velin Dimitrov and R.J Linton from RiVeR lab, WPI for the motivation and support. I would like to thank my friends for the providing me with mental support and confidence to complete this project. Last but not the least I would like to thank my family and my loving girl friend for listening to all my frantic need for help and assisting me in my endeavor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Map creation is an active field of research in robotics. Automation of map creation not only saves time and effort but also helps in model generation of environments that are manually inaccessible for hand-based drawings. Model generation from images and video has been a core problem of research in computer vision and has been studied under the Structure from Motion(SfM) problem. In this project we have applied the concepts of the SfM to help aid in combating the Asian Longhorn Beetle, one of the most destructive and invasive species that has caused the damage and death of several trees in the United States, particularly in the New England area.

1.1 The Motivation

Invasive pests when introduced in an uncontrolled environment pose threat to forestry and agriculture. Asian Long Horned beetle (ALB) is one such pest that is a serious threat to the hardwood industry in the New England area. ALB is native to eastern Asian countries and has been introduced into US forests accidentally through import packaging. The ALB infestation in central Massachusetts is the largest in

USA. Approximately 30,000 trees have been infested in Worcester, MA alone. ALB usually affects Maple, Willow, Elm and similar trees. ALB infestation destroys the cambium of the trees and eventually the internal structure of the tree trunk thus making them weak and vulnerable to disease. This reduces the economic value of the trees.

The US department of Agriculture (USDA) is the chief government agency responsible for the eradication of the ALB in USA. The current approach for the eradication of ALB requires visual inspection of the trees near the vicinity of affected region. An infested tree is identified by[show image]

- 3/4th inch exit holes
- oviposition holes - scar marks from female laying eggs
- frass that is generated by the larvae eating the trunk
- specific leaf feeding pattern
- presence of the beetle

Figure 1.1: The Asian Long Horn Beetle with the exit hole on a tree.

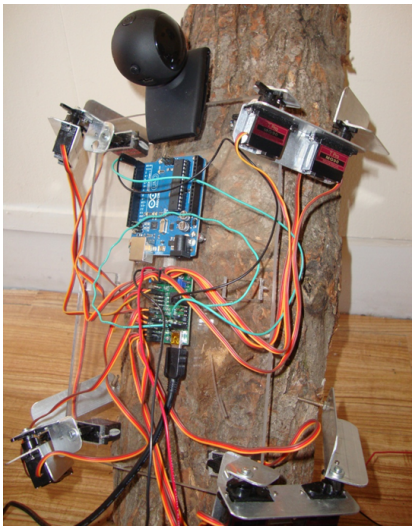
In order to eradicate the ALB infestation the USDA sends ground survey and aerial survey teams to determine the extend of the ALB infestation. Nearly 62% of New England is forested. Survey of trees is a time consuming process. So far nearly 2 million trees have been surveyed in Worcester, MA alone. Additionally, climbing trees poses a safety hazard for humans.

A robotic inspection of such trees will greatly help in speeding up the process and provide USDA with a tool for remote monitoring of trees.

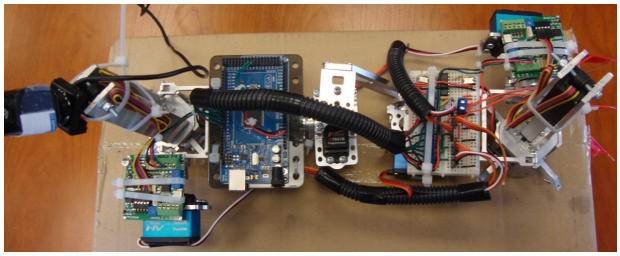
1.2 Robotic Mapping

With the progress in robotics remote surveillance of building, tunnels, bridges and disaster zones has become a reality. Remote surveillance is used when it is not possible to access the environment directly or when very detailed information is required. In forestry, monitoring of trees can help determine details about the condition of the tree such pests, diseases and other harmful stress affecting the tree.

Visual information about the tree bark and structure plays an important role in detection of ALB. Converting the image of a tree to a 3D model provides a spatial information about the structure and features of the tree. This can play an important role in the detection of infestation by insects. A similar problem is solved in robot navigation in terms of environment mapping. SLAM (short for Simultaneous Localization and Mapping) is a map generation algorithm popularly used by robotics community in this regard. SLAM[6] uses sensor information as a feedback for feature estimation. It gives the robot the ability to visualize and see the environment with respect to itself thus helping it make more suitable actions on the environment.



(a) Tree Climbing robot 2012



(b) Tree Climbing robot 2013

Figure 1.2: The tree climbing robot developed by WPI MQP teams

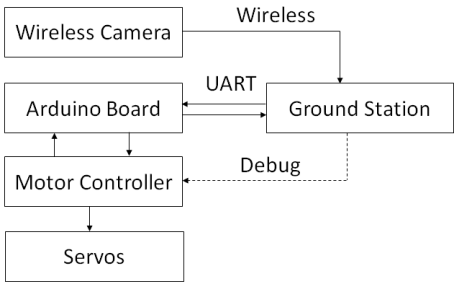


Figure 1.3: The Overall system architecture for the Tree climbing robot

At WPI, MQP teams have been working on developing a tree climbing robot (figure 1.2). The current robots have limited payload and use minimal sensors. Due to the constrained payload capacity, it is feasible to use only a monocular camera for mapping. Since the robot has limited processing capabilities, it is necessary to process the images from the camera on a workstation computer. Images are expected to be transferred wirelessly to the workstation. Figure(1.3) describes the system architecture. The idea behind this work is to make use of the images captured by the MQP robot to generate the 3D models of the tree using suitable mapping algorithm.

Chapter 2

Preliminaries

Some mathematical notations have been used for this work and the same have been reported in this document. The first section in this Chapter summarizes a list of these notations and quaternions. The following sections deal with the camera model and radial distortion model that have been used to convert features from image coordinate to world coordinate.

2.1 Notations

2.1.1 Vectors

Vectors are represented with boldface font. The frame of reference is indicated as a super script. For example $\mathbf{v^A}$ represents a vector $\mathbf{v}$ in the reference frame $\mathbf{A}$.

2.1.2 Matrices

A matrix is also represented using boldface font. In the equations we frequently make use of the Identity matrix, $\mathbf{I}$ and Zero matrix $\mathbf{0}$. The dimensions of the matrix is represented using subscripts. $\mathbf{X_m}$ denotes a matrix of dimension $\mathbf{m\ x\ m}$ and $\mathbf{X_{mn}}$

denotes a matrix of dimension **m x n**.

2.1.3 Frame conventions

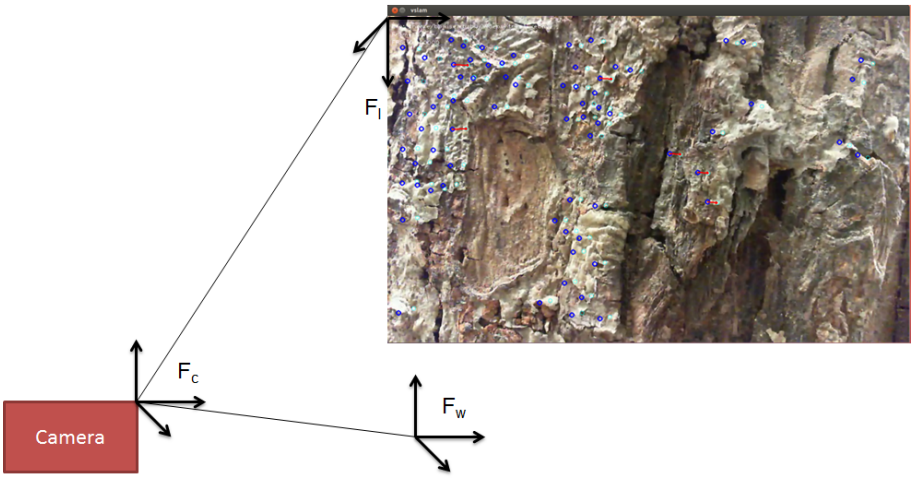


Figure 2.1: Various frame of references in the system.

Camera Frame: The camera frame is the frame the camera is in. This is usually denoted by with a subscript **C** eg:- $\mathbf{F_c}$

Image Frame: This is the optical frame on which the image is projected. The **z** axis is usually assumed to be perpendicular to the image plane. This is represented with a subscript **I**. eg:- $\mathbf{F_I}$

World Frame: This is the frame in which the camera and the features exist. This is assumed to be the root frame. This frame is represented with the subscript **W**. eg:- $\mathbf{F_w}$

To represent a transformations between frames we use the following notation $\mathbf{F_B^A}$ which represents a transformation from frame **B** to frame **A** (the reference frame is *A*).

The transformation from the Camera frame to the Image frame $\mathbf{F_I^C}$ is fixed based on the camera configuration.

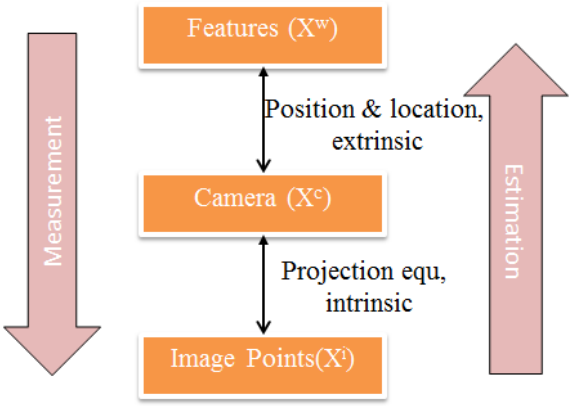


Figure 2.2: The frame conversions for a point in image coordinate to a point in world coordinate.

2.2 Quaternions

Quaternions are a number system that extends the complex numbers. It can be used to represent orientation effectively.

A quaternion is generally defined as,

$$q = (q_0, \mathbf{q}) = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{2.1}$$

where q_0 represents the scalar part and $\mathbf{q}$ represents the vector part. The vectors $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ satisfy,

$$\mathbf{i^2} = \mathbf{j^2} = \mathbf{k^2} = \mathbf{ijk} = -1 \tag{2.2}$$

2.2.1 Angle-axis form

A unit quaternion represents any rotation in the SO(3) i.e. a rotation by an angle θ about a fixed axis defined by an unit vector $r = (r_x, r_y, r_z)$. The angle-axis

representation of an unit quaternion is

$$q = \begin{bmatrix} \cos(\theta/2) & r^T \sin(\theta/2) \end{bmatrix}^T \tag{2.3}$$

2.2.2 Conversion to Rotation Matrix

A quaternion rotation can be converted into a rotation matrix, by simplifying the quaternion multiplications qpq^*. If q represents a quaternion in angle axis form.

$$\begin{bmatrix} \cos\theta + r_x^2(1-\cos(\theta)) & r_x r_y(1-\cos(\theta) - r_z\sin(\theta) & r_x r_z(1-\cos(\theta)) + r_y\sin(\theta) \\ r_y r_x(1-\cos(\theta)) + r_z\sin(\theta) & \cos(\theta) + r_y^2(1-\cos(\theta)) & r_y r_z(1-\cos(\theta)) - r_x\sin(\theta) \\ r_z r_x(1-\cos(\theta)) - r_y\sin(\theta) & r_z r_y(1-\cos(\theta)) + r_x\sin(\theta) & \cos(\theta) + r_z^2(1-\cos(\theta)) \end{bmatrix} \tag{2.4}$$

2.3 Camera Model

The configuration of the camera is described mathematically using the camera model. The Pin hole camera model is a simple model commonly used in computer vision. In the Pin hole camera model, an inverted projection of the scene is assumed to be formed behind the camera aperture. This model assumes a small aperture with no lenses. For a point $X^c = (x_1, x_2, x_3)$ observed using a camera with intrinsic parameter,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

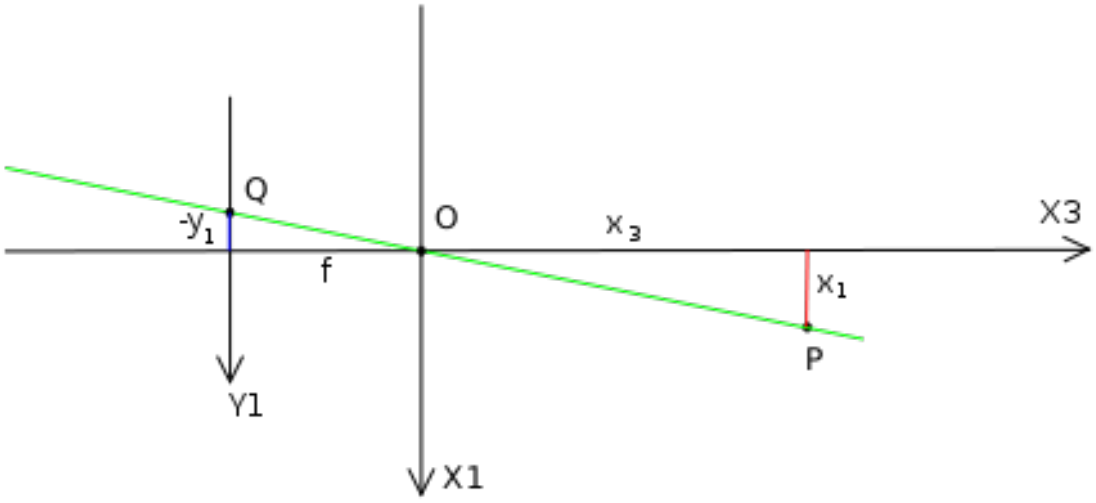


Figure 2.3: The Pinhole camera model.

if represents a feature in the camera coordinate frame, its projection $Y^I = (y_1, y_2)$ is defined as,

$$\begin{aligned} y_1 &= -f_x * \frac{x_1}{x_3} + c_x \\ y_2 &= -f_y * \frac{x_2}{x_3} + c_y \end{aligned} \tag{2.6}$$

2.4 Radial Distortion Model

Distortion occurs in camera due to defects/aberrations in the lenses used. The Distortion due to spherical aberrations can be removed by modelling the camera distortion as Radial distortion. Radial distortion, whilst primarily dominated by low order radial components, can be corrected using Brown's distortion model.

$$\begin{aligned} x_u &= (x_d - x_c)(1 + K_1 r^2 + K_2 r^4) \\ y_u &= (y_d - y_c)(1 + K_1 r^2 + K_2 r^4) \\ r &= \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2} \end{aligned} \tag{2.7}$$

Chapter 3

Background

In Chapter 1 we have discussed the motivation behind this project and my interest in solving the challenge by using robotic mapping as a tool. Chapter 2 summarizes the notations used in this work. In the current chapter, the concepts of SLAM and its subset VSLAM will be introduced as the robotic mapping techniques in focus. For the purpose of determining the pose of the camera and locating features on trees, a non-linear optimal state estimator, the extended Kalman Filter is used. A theoretical background on Kalman filter and extended Kalman Filter is reviewed in section 2 and 3 of this Chapter. A brief overview of the notions, conventions and mathematical tools in SLAM is presented, along with a literature review on the current state of the art in robotic mapping.

3.1 SLAM

Simultaneous Localization and Mapping(SLAM) is a technique used in robotic community to localize the robot and at the same time generate a map of the environment around it. This can be considered as a chicken and egg problem as the localization depends on how accurate the robot knows the map of the environment and the

mapping depends on how accurate the robot knows its location.

Simultaneous Localization and Mapping(SLAM) is a technique used in robotics community for generating map of an unknown environment. SLAM has been formulated and applied in multiple mobile and aerial robots. [7][1] give a survey of the current state of the art SLAM algorithms. SLAM aggregates a number of approaches for automated map generation without any additional pose knowledge apart from sensor information. Information from the map is required to correctly localize yourself ; at the same time, localization is going to affect how accurate you can build the map[15]. It is commonly referred to as a chicken and egg problem.
The underlying methods used to solve SLAM are dominated by the type of sensors used, the time available for processing, the representation of the environment and the dimensionality of the resulting map. Similar to SLAM we intend to generate a model of the tree with unknown structure while climbing it. This model will help us identify the existence of the invasive pests.

3.2 VSLAM

Visual Simultaneous Localization and Mapping(VSLAM) is a technique that uses only the input from a monocular camera to generate a map of the environment around the robot. In VSLAM, the visual features in the environment as landmarks to generate the map of the environment. VSLAM uses a monocular camera which is a light weight, small, cheap and passive sensor that is readily available.
In our problem a three dimensional model of tree is to be generated using only visual information from the camera on the robot. We decided to implement an on-line technique instead of batch technique as quicker the model is available to the foresters in the field, the more useful the tool will be. Also the processing is limited by the

field computer that is carried. The end model is expected to be represented as a mesh grid that can be used to inspect the tree surface. The most suitable solution based on the requirements is an EKF based MonoSLAM.

MonoSLAM was first introduced by Andrew J. Davison in [4]. Davison demonstrated real time processing for Simultaneous Localization and Mapping using a single hand-held camera. He presented a Bayesian Framework for single camera Localization via active tracking of a sparse set of features. An extended Kalman Filter was used to propagate the uncertainty in the system[5]. He introduced an undelayed automatic initialization of the features unlike previous work where a batch process was involved for initialization[]. Davison et al introduced the Inverse Depth representation of the features which neatly deals with the nonlinearities of the Euclidean space. Inverse Depth representation has caught on with the Vision based SLAM community for its simplicity in design and power in representation.

3.3 Kalman Filter

Kalman filter[9] is a discrete filtering technique that uses a series of measurements observed over time to produce the best estimate. It is an optimal estimator that minimizes the error between the current estimate of the observation and the actual observation. In Kalman filter the system is observed to be linear with gaussian white noise. Assuming the states of the system at time step k is represented as X_k and the covariance of the system is represented by P_k.

The first step in Kalman filter is to propagate the states of the system using the system matrix A to the next state using 3.1

$$
\begin{aligned}
X_{k|k-1} &= AX_{k|k-1} + W_k \\
P_{k|k-1} &= AP_{k-1}A^T + Q \\
z_{k|k-1} &= HX_{k|k-1}
\end{aligned}
\tag{3.1}
$$

The next step is to calculate the Kalman gain at time step k using 3.2

$$
K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{3.2}
$$

Once observations for a time step k are available as $z_{k|k}$, a Kalman update is performed using 3.3

$$
\begin{aligned}
X_{k|k} &= X_{k|k-1} + K_k(z_{k|k} - z_{k|k-1}) \\
P_{k|k} &= (I - K_kH)P_{k|k-1}
\end{aligned}
\tag{3.3}
$$

3.4 Extended Kalman Filter

Since most real world problems are non linear the Extended Kalman Filter[10](EKF) provides a tool for such conditions. In EKF the system model and the measurement model is assumed to be non linear functions $f(X)$ and $h(X)$. The Jacobian of the system and the measurement model is calculated at each time step3.4.

$$
\begin{aligned}
A_{k-1} &= \frac{\partial f}{\partial X}|_{X_{k-1|k-1}} \\
H_k &= \frac{\partial h}{\partial X}|_{X_{k|k-1}}
\end{aligned}
\tag{3.4}
$$

The EKF prediction is performed using 3.5

$$
\begin{aligned}
X_{k|k-1} &= f(X_{k|k-1}, W_k) \\
P_{k|k-1} &= A_{k-1} P_{k-1} {A_{k-1}}^T + Q \\
z_{k|k-1} &= h(X_{k|k-1})
\end{aligned}
\tag{3.5}
$$

followed by kalman gain calculated using 3.6

$$
K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R)^{-1} \tag{3.6}
$$

finally the EKF update is performed using equation 3.7

$$
\begin{aligned}
X_{k|k} &= X_{k|k-1} + K_k(z_{k|k} - z_{k|k-1}) \\
P_{k|k} &= (I - K_k H_k) P_{k|k-1}
\end{aligned}
\tag{3.7}
$$

Chapter 4

Approach

There are three main layers required to solve Visual SLAM algorithm (figure: 4.1):

1. Image Processing Layer
2. Filtering Layer
3. Model Management Layer

All steps pertaining to images such as feature extraction and feature matching is done in the image processing layer. Filtering layer implements the Extend Kalman Filter which is used to estimate the states of the system. By analyzing VSLAM in layers it makes it easy to evaluate the different implementation for the individual components without having to worry about the interactions between them. The model management layer ensures synchronization between the layers.

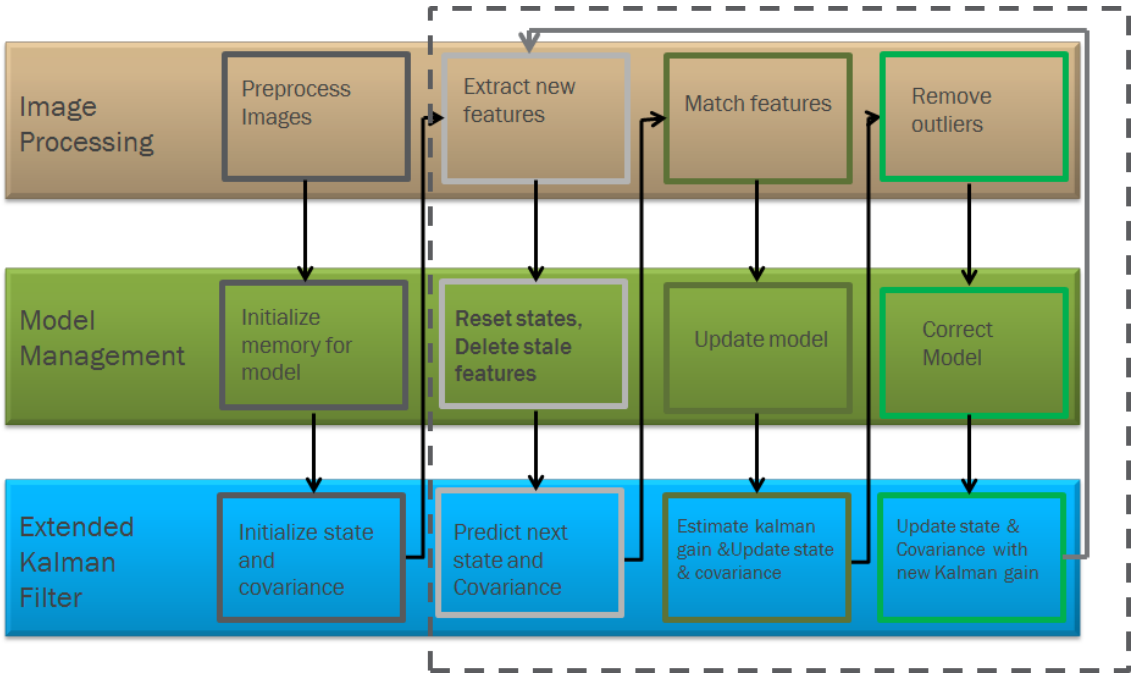


Figure 4.1: A block diagram of the processes in the approach

4.1 Image Processing Layer

Image Processing in general is used to extract information from images. Information from images are usually in the pixel coordinate frame. Using the projection model and the distortion model of the camera it is possible to convert these features to world coordinate frame.

4.1.1 Feature Detection

In computer vision there are many types of features extracted from the images such as lines, edges, corner points, ridges etc. Corner points are easy to detect and manage, and these are what we have used in VSLAM. For good solutions in VSLAM it is necessary to have well distributed corner points. A good corner detector as scale, rotation and lighting invariant. This provides the detected corners with robustness. Corners are mathematically well defined points in the image which can be repeatably

found.

4.1.2 Interest Point detectors

The terms corners and interest points are used interchangeably and refer to features in an image, which are represented in the image coordinate system.

FAST

Features from Accelerated Segment Test is a feature detector that is computationally fast[13]. A feature is extracted by comparing 16 pixels around it in the Bresenham circle of radius 3 (figure 4.2). A pixel declared interest point if it satisfies either of the following conditions:

- Condition 1: A set of 12 contiguous pixels is greater than the intensity of the pixel plus the threshold.
- Condition 2: A set of 12 contiguous pixels is greater than the intensity of the pixel is minus the threshold.

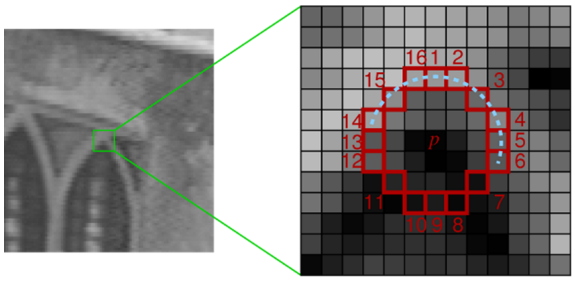


Figure 4.2: The FAST detector comparison operator.

A High Speed test can be applied to reject candidates by examining 4 example pixel locations in 1, 9, 5 and 13.
FAST can be further modified to use more than 16 pixels but in such cases a machine learning algorithm is used to speed up the detection.

Good Features to Track

Good Features to Track is also known as Shi-Thomshi feature detector[14]. Shi-Thomsai interest point detector is based upon the harris corner[8] detector where the weighted sum of squared distances S between a patch and the patch shifted by (x, y) is calculated as

$$S(x,y) = \sum_u \sum_v w(u,v)\ \left(I(u+x, v+y) - I(u,v)\right)^2 \tag{4.1}$$

which can be reduced into the form,

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} H \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.2}$$

H is the Harris matrix and is represented as,

$$H = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{4.3}$$

The angle brackets denote averaging. An interest point is characterized by a large variation of S in all directions of the vector (x, y). Shi Thomasi claim the minimum $(min(\lambda_1, \lambda_2))$ of the eigen value of H is more stable to determine if the pixel is an interest point.

SURF

Speed Up Robust Feature[2] is another interest point detector and descriptor. SURF has very good robustness and repeat-ability.

The SURF detector is based on the determinant of the Hessian Matrix of Gaussians. The Hessian detector is dened as $\det(H(u, v, \sigma))$ with

$$H(u,v,\sigma) = \begin{bmatrix} L_{uu}(u,v,\sigma) & L_{uv}(u,v,\sigma) \\ L_{uv}(u,v,\sigma) & L_{vv}(u,v,\sigma) \end{bmatrix} \tag{4.4}$$

where $L_{uu}(u,v,\sigma) = \frac{\partial^2 L(u,v,\sigma)}{\partial u^2}$ denotes the second order derivative of the Gaussian convolution of the image.L_{uv} and L_{vv} are defined analogously.In L_{uu}, L_{uv} and L_{vv} are roughly approximated by box filters D_{uu}, D_{uv}, D_{vv}.

Integral images are used to improve the performance speed, so the search in σ can be carried out in scale space.

4.1.3 Matching

There are two general methodologies for matching.

Descriptor Based

When the features are detected, in addition to the feature location, a descriptor based on the neighborhood of the features is also generated. A match is found by calculating the norm between the different pairs of descriptors. The pair of descriptors with the with the minimum distance between them is usually defined as a match.

Local Consistency Based

Between frames there is usually a well defined local consistency in small neighborhood window around the point of interest. By applying these constrains it is possible to track points between two frames.

In Optical Flow algorithm, For a $2D + t$ dimensional case (x, y, t) with intensity $I(x, y, t)$ will have moved by Δx,Δy and Δt between the two image frames, and the following brightness constancy constraint can be given:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{4.5}$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \text{H.O.T.} \tag{4.6}$$

The LucasKanade method[11] makes the following assumptions the intensity displacement

- between two nearby instants (frames) is small
- approximately constant within a neighborhood of the point p under consideration.

Within a small window, the equation 4.6 is assumed to be true. By using least square for all the windows in the image, the optical flow and the corresponding matches is estimated.

4.2 Extended Kalman Filter

The pose of the camera and the location of the features are estimated using the Extended Kalman Filter in Visual SLAM. In monocular VSLAM, the camera is assumed to move with a constant velocity while the features are stationary.

4.2.1 State Representation

The state vector X_t in a Kalman filter is not directly observed. It is indirectly estimated using measurements from the camera. The entire VSLAM system is described both by the camera and the position of the features.

Camera States

The camera state vector x_s is parameterized as 13 states as described below,

$$x_s = \begin{bmatrix} r_c^w \\ q_c^w \\ v_c^w \\ \omega_c^w \end{bmatrix} \tag{4.7}$$

Where, $r_c^w = (x_c, y_c, z_c)^T$ represents the $3D$ position of the optical center of the camera in the world coordinate system. It is represented by a $3x1$ vector.
q_c^w represents the unit quaternion specifying the orientation of the camera in the world coordinate system. It is represented using a $4x1$ vector
v_c^w represents the linear velocity of the camera in the world coordinate system. ω_c^w represents the angular velocity of the camera in the world coordinate system.

Initially the camera is assumed to start from the origin of the world coordinate system $r_w^c = (0, 0, 0)$ looking in the direction of positive z axis $q_w^c = (1, 0, 0, 0)^T$. It is assumed to be stationary with $v_w^c = (0, 0, 0)$ and $\omega_w^c = (0, 0, 0)$. The uncertainty in the camera state is described using P_s. P_s is a symmetric positive definite matrix of dimension $13X13$.

Features States

The features are represented using the inverse depth encoding.A feature y_i in inverse depth encoding[3] is comprised by the following 6 dimensional vector:

$$y_i^w = \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix} \tag{4.8}$$

where,

$\begin{bmatrix} x_i^c & y_i^c & z_i^c \end{bmatrix}$ specifies the 3D position of the cameras optical center at the rst observation of feature y_i^w

θ_i and ϕ_i are azimuth and elevation of the feature in reference to the camera coordinate system

ρ_i is the inverse depth of y_i^w

The conversion of inverse depth feature to 3D representation is given by

$$X_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix} + \frac{1}{\phi} m(\theta_i, \phi_i) \tag{4.9}$$

$$m(\theta_i, \phi_i) = \begin{bmatrix} \sin(\theta_i)\cos(\phi_i) & \sin(\phi_i) & \cos(\theta_i)\cos(phi_i) \end{bmatrix}^{\mathrm{T}} \tag{4.10}$$

The full state vector x_t for a map composed of n features in inverse depth is

therefore composed of

$$X_t = \begin{pmatrix} x_s^T & y_1^T & y_2^T & y_3^T & \dots & y_n^T \end{pmatrix}^T \tag{4.11}$$

4.2.2 Process Model

The Process Model describes the state transition function from the previous camera state x_{k-1} to the current camera state x_k. Since we assume a continuous velocity model in visual SLAM. The equation 4.12 defines it.

$$f_k(x_{k-1}) = \begin{bmatrix} r_{wk}^c \\ q_{wk}^c \\ v_{wk}^c \\ \omega_{wk}^c \end{bmatrix} = \begin{bmatrix} r_{wk-1}^c + v_{wk-1}^c \Delta t \\ q_{wk-1}^c \times quat(\omega_{wk-1}^c \Delta t) \\ v_{wk-1}^c \\ \omega_{wk-1}^c \end{bmatrix} \tag{4.12}$$

In Extended Kalman filter, Jacobian of the transition function $f_k(x_{k1})$ defined in equation (4.12)at the current state is used to update the covariance $\sum_k$ in the system. Jacobian F_k is of $dim(F_k) = 13x13$ of the following form

$$F_k = \begin{bmatrix} \frac{\partial r_{wk}^c}{\partial r_{wk-1}^c} & 0 & \frac{\partial r_{wk}^c}{\partial v_{wk-1}^c} & 0 \\ 0 & \frac{\partial q_{wk}^c}{\partial q_{wk-1}^c} & 0 & \frac{\partial q_{wk}^c}{\partial \omega_{wk-1}^c} \\ 0 & 0 & \frac{\partial v_{wk}^c}{\partial v_{wk-1}^c} & 0 \\ 0 & 0 & 0 & \frac{\partial \omega_{wk}^c}{\partial \omega_{wk-1}^c} \end{bmatrix} \tag{4.13}$$

The process noise that is added to the covariance $\sum_k$ is defined as follows

$$R_k = \tilde{F}_k V_{max,k} \tilde{F_k}^T = \begin{bmatrix} \frac{\partial r^c_{wk}}{\partial v^c_{wk-1}} & 0 \\ 0 & \frac{\partial q^c_{wk}}{\partial \omega^c_{wk-1}} \\ \frac{\partial v^c_{wk}}{\partial v^c_{wk-1}} & 0 \\ 0 & \frac{\partial \omega^c_{wk}}{\partial \omega^c_{wk-1}} \end{bmatrix} V_{max,k} \begin{bmatrix} \frac{\partial r^c_{wk}}{\partial v^c_{wk-1}} & 0 \\ 0 & \frac{\partial q^c_{wk}}{\partial \omega^c_{wk-1}} \\ \frac{\partial v^c_{wk}}{\partial v^c_{wk-1}} & 0 \\ 0 & \frac{\partial \omega^c_{wk}}{\partial \omega^c_{wk-1}} \end{bmatrix}^T \tag{4.14}$$

4.2.3 Measurement Model

In Extended Kalman filter the measurement function is used to convert state estimates to estimated measurement $\hat{z_k}$. According to the pinhole camera model (see chapter 2) an observed point on the image sensor is defined by $h^c = (h_x h_y h_z)^T$ in the camera coordinate system. For a point in inverse depth encoding y_i the measurement is given by,

$$h^c_i = R^c_w \left(\rho_i \left(\begin{pmatrix} x_{c,i} \\ y_{c,i} \\ z_{c,i} \end{pmatrix} - r^w_c \right) + m(\theta_i, \phi_i) \right) \tag{4.15}$$

R^c_w is the rotation matrix between the world coordinate frame and the camera coordinate frame.

The jacobian of the measurement equation (4.15) with respect to the camera states is defined by $\frac{\partial h(y_i)}{\partial x_s}$. It is required to update the covariance $\sum_k$ of the system.

$$\frac{\partial h(y_i)}{\partial x_s} = \begin{pmatrix} \frac{\partial h(y_i)}{\partial r^w_c} & \frac{\partial h(y_i)}{\partial q^w_c} & 0 \end{pmatrix} \tag{4.16}$$

where $dim(\frac{\partial h(y_i)}{\partial x_s}) = 2 \times 13$.

4.3 Model Management Layer

Model management layer is responsible for

- synchronize the features with the state model in the Extended Kalman Filter
- provide debug and display information
- convert the features in inverse depth to 3D representation
- provide interface to generate the mesh grid

Synchronization

Often the feature detector might find an invalid feature that might go out of field of view or be obscured before the features depth has been well estimated. In such situations it is necessary to delete the feature from the system.

Debug & Display

All the debug outputs and the display of features in images can be implemented in the Model Management layer and thus avoid overheads on the other layers.

Convert to 3D

The features need to be represented in 3D representation inorder to visualize the points. This operation can be performed using the equation (4.9).

Generation of mesh grid

Since the points converted into 3D need not be ordered in time or space, the mesh grid algorithm need to be flexible. The Fast triangulation of unordered points algorithm[12] from the PCL library is used to generate the 3D model of the tree. It is

a greedy triangulation algorithm. It works by maintaining a list of fringe points.By extending it until all possible points are connected the mesh grid is created. It can deal with unorganized points, coming from one or multiple scans, and having multiple connected parts. It is also suitable for point clouds with non uniform density.

Chapter 5

Implementation

The algorithm was finally implemented using threads. 5.1 shows different threads and how they communicate between each other.

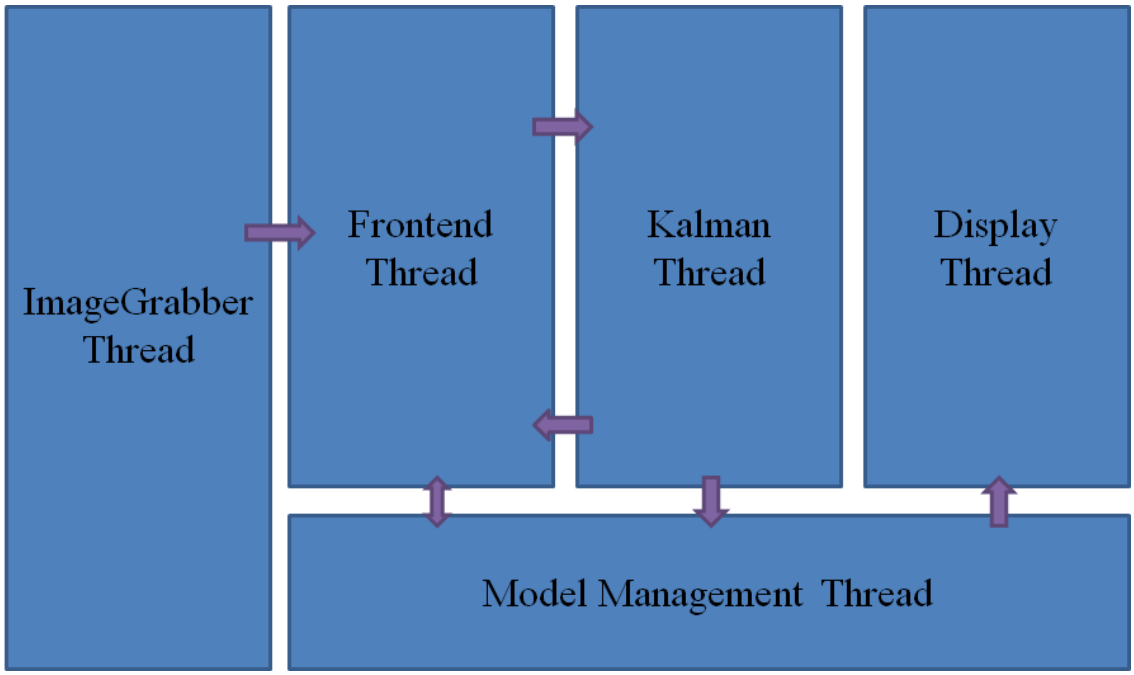


Figure 5.1: A block diagrams of the threads in the implementation

5.1 Image Grabber Thread

The image grabber thread loads the images into the program memory as a queue. An image pyramid is created by from the single original image by successively down-sampling until some desired stopping point is reached. The number of levels in the pyramid are specified in the config file. Higher the level, smaller the size of the image.

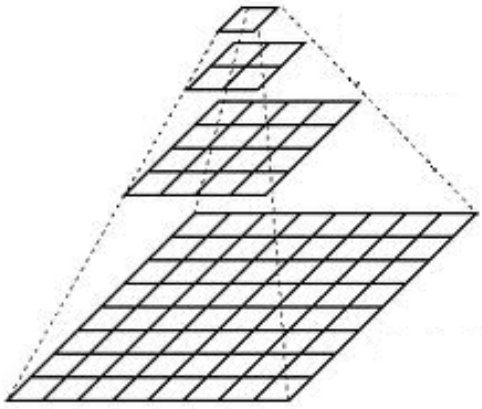


Figure 5.2: A visualization of image pyramid with 4 level

5.2 Frontend Thread

The frontend is responsible all the features in the model. The frontend implements a multi-scaled grid based FAST feature detector for generating new features. It uses optical flow based matching algorithm to perform feature matches.

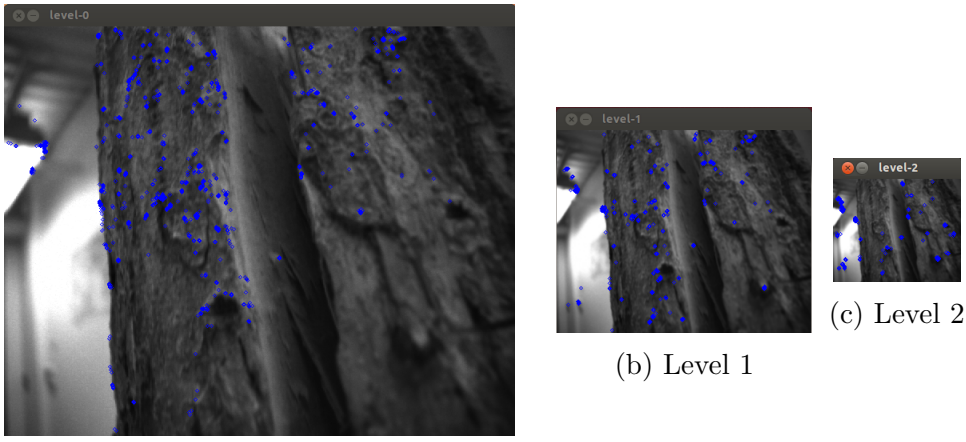
5.2.1 Multi-Scaled FAST grid

Since the tree need not be uniformly illuminated, a single threshold for the entire tree might not be a good choice. Hence the image is divided into grid depending on the maximum number of features to be extracted. The fast algorithm discussed

in Chapter 4 is implemented repeatedly to these sub grids of image with increasing threshold. The threshold that provides significant number of features within the grid is selected. The interest point with the maximum response is declared as the feature of interest with the grid. This algorithm is repeated to all the levels in the image pyramid.

Once the features are detected they are stored in Quadtree that facilitates faster search of these features. Quadtree is a tree data structure where each node has 4 children. Each feature is inserted into the quadtree based on the quadrant in which they occur.

In order to search for a quadtree, all the nodes that intersect an area of interest is found. Following this, the elements in it are checked. This helps in avoiding the hassle of searching through the entire list of features.



(a) Level 0

(b) Level 1

(c) Level 2

Figure 5.3: The FAST features detected in all the three levels. In Level 2 we find features that were not detected in the blurred part in Level 0

5.2.2 OpenCV

FAST and other detectors are implemented in features2d module of OpenCV. OpenCV is implemented under BSD license. OpenCV was implemented with the aim to improve computational efficiency and with a strong focus on real-time applications in image processing.The library is primarily written in C/C++. It has a strong active development community.

5.3 Kalman Thread

The Kalman Filter is implemented based on the equations (3.7). When there are n features in the system, the size of the mean vector is $13 + 6n$ and the size of the co-variance matrix is $13 + 6n \times 13 + 6n$. Table (5.1) describes the EKF SLAM procedure.

Event	**SLAM**	**EKF**
New Feature point detected	Landmark Initialization	Append the mean and covariance of the states
Camera obtains new image	Robot Motion	perform prediction on the states
Feature matches between two frames	Map correction	Perform EKF update
Feature goes out of view	Landmark deletion	remove corresponding states

Table 5.1: Different steps in VSLAM and the corresponding action within the EKF system

5.3.1 Sparse Matrix

Since the cross correlation between one feature to another is zero, most of the elements in the Jacobian Matrix and the Covariance matrix is 0. To prevent wastage

of space and computation power on these 0 elements, the matrix is stored in the compressed column order Storage format.

In Compressed Column Order (CCO) Storage format three smaller arrays are used to represent the matrix. For instance, let m be a column-major sparse matrix. Then its non-zero coefficients are sequentially stored in memory in a column-major order (values array). A second array of integer stores the respective row index of each coefficient (inner indices array). Finally, a third array of integer, having the same length than the number of columns, stores the index in the previous arrays(i.e. either values or inner indices array) of the first element of each column (outer indices).

Here is an example, with the matrix:

$$m = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 17 \\ 7 & 5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 0 & 8 \end{bmatrix} \tag{5.1}$$

Its sparse matrix representation using the Compressed Column Storage format:

Values:	22	7	3	5	14	1	17	8
Inner indices:	1	2	0	2	4	2	1	4
Outer indices:	0	2	4	5	6	7		

5.3.2 Eigen

The Eigen Library was used to implement all the matrix multiplication. It supports sparse Matrices and is released under LGPL3+. Being a template library, it

has less overhead. It makes use of the SSE3/SSE2 vectorizations to achieve faster computation times.

5.4 Display Thread

The PCL's vizualizer is implemented in this thread. It displays

- the path traversed by the camera.
- the mesh grid generated by the Fast Triangulation of unordered points.

5.4.1 Point Cloud Library

Point Cloud Library is a opensource project for point cloud processing. PCL is distributed under the BSD license. The Fast triangulation of unordered points is implemented[12] in PCL as a part of the image surface reconstruction module.

5.5 Model Management Thread

As described in chapter 4, the model management synchronizes the different threads. Threading was implemented using boost. for synchronization between the image thread and the frontend threads, a producer consumer paradigm was used. Mutex was used when resources were used to prevent race conditions and data corruptions. Memory for all the resources was declared using shared pointer thus ensuring no memory leaks.

Chapter 6

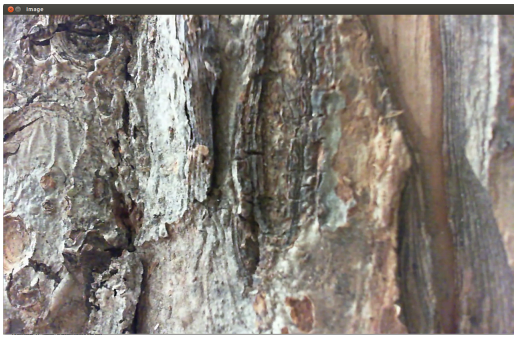
Evaluation

6.1 Data Sets

A total of 5 data sets were collected for testing the algorithms. The trees (data set) that were sampled for this purpose, belong to the categories that are susceptible to infestation by the Asian Long Horn Beetle. A list of trees used for testing are summarized in the Table below.

Name	**Description**
Tree 1	Tree with dry bark in indoor lighting
Tree 2	Maple Tree big ridge bark (outdoor)
Tree 3	Pine Tree with thick scaly bark (outdoor)
Tree 4	Tree with smooth bark (outdoor)
Tree 5	Maple Tree with small ridge bark (outdoor)

Table 6.1: The data-sets that were collected with brief description



(a) Tree 1

(b) Tree 2

(c) Tree 3

(d) Tree 4

(e) Tree 5

Figure 6.1: Sample image from all the date sets

6.2 Features vs Bark type

For each of the above data set, the Multi-scale grid based FAST algorithm was used to obtain an average number of features. The average number of features in short, was the mean of the features counted by the algorithm for 20 frames. This is summarized in the following Table.

#	**Data Set**	**Type**	**Average no of features in a frames**
1	Data Set 1	Dry Bark	175
2	Data Set 2	Smooth Bark	258
3	Data Set 3	Big Ridge	281
4	Data Set 4	Small Ridge	250
5	Data Set 5	Scaly Bark	239

Table 6.2: Average no of features is calculated in batches of 20 frames

It has been demonstrated that more number of features in the VSLAM system will yield in better solution[16]. Since We are getting more 150 features per frame reliably, the algorithm is suitable for the different tree types.

6.3 Comparison of Feature Detectors

A comparative study of the average number of features obtained for a single frame using different feature detector algorithms was done. Data set 1 was used for this study. The results indicated that Grid based FAST detector was able to detect more number of features in a single frame compared to the other algorithms. The grid based FAST detector algorithm was thus chosen in order to aid in better estimation of the tree features[16].

#	Data Set	Type of detector	average no of features in a frames
1	Data Set 1	FAST	43
2	Data Set 1	Grid based FAST	87
3	Data Set 1	Good Features to track	50
4	Data Set 1	SURF	47

Table 6.3: Comparison between different types of feature detector used for the dry bark dataset with indoor lighting

6.4 Matching vs Bark type

Once the grid based FAST was finalized as the feature detection algorithm, an optical flow matching algorithm was used to compare the matching percentages in all the data sets. A higher matching percentage indicated better ability to keep track of the features in different frames. The matching percentage is defined by the following equation.

$$matching\% = \frac{matchesinconsecutiveimages}{Featuresdetectedinn1stimage} \tag{6.1}$$

#	Data Set	Average matching %
1	Data Set 1	58.75
2	Data Set 2	85.45
3	Data Set 3	80.32
4	Data Set 4	78.43
5	Data Set 5	79.89

Table 6.4: Average no of matches using the formula in 6.1

The average matching percentage for all the data sets lie in the range 58 - 85, indicating that the optical flow matching algorithm is a suitable tool for feature matching.

6.5 Comparison of Matching techniques

Similar to the study carried out for comparing feature detection algorithms, a comparison of matching algorithms was also performed. Optical flow and descriptor based matching algorithms were compared for Data set 1. In this study, the optical flow algorithm emerged as a better option, yielding a matching percentage of 58.78.

#	**Data Set**	**Matching Technique**	**% of Matches**
1	Data Set 1	Optical Flow	58.78
2	Data Set 1	Descriptor Based	40.87

Table 6.5: Comparison between the matching techniques

6.6 Sparse vs dense

The extended Kalman filter was implemented using Sparse matrix and dense matrix. A comparison between the performance of both is presented in the following table. It was concluded that the sparse matrix based implementation was faster than the dense matrix.

Operation	**Sparse(Avg) in msec**	**Dense(Avg) in msec**
Add a new feature	0.10	0.36
Delete a feature	0.10 2	0.20
Kalman Measurement	2.0	10.0
Prediction	5.00	9.00
Kalman Update	120.00	300.00

Table 6.6: Average time for the different operations in the EKF filter. The processor used is Intel i7-3630QM CPU 2.40GHz

Chapter 7

Conclusion

Visual Simultaneous Localization and Mapping is a suitable algorithm for generating 3D mesh grid of the tree. The multi-scaled grid based FAST feature extractor is successful in generating large number of features in trees usually affected by ALB. In addition to finding the features, matches need to be determined in order to update the Extend Kalman Filter. We see that by using Quad tree combined with optical flow we can get reasonable number of matches to correct the system estimates.

Large Matrix sizes implemented in dense memory format can lead to slow computation in system. Hence by using Sparse memory format for the matrices, we can achieve near real time processing. Finally implementing Visual SLAM for mesh grid generation needs effective use of the computing resources such as multi-threading, shared pointers, corner detection etc available in tools such as OpenCV, Eigen, PCL.

Chapter 8

Future Work

The current implementation produces a model that is of arbitrary scale to the actual tree. It is possible to introduce scale into monocular system by using external observations. One such observation about the scale can be introduced by counting the number of steps the robot takes and the amount of distance traveled in each step.

In Image processing front, the matching algorithm currently only produces an average of 80% matches. This is mainly due to the local inconsistencies that occur when using optical flow. By improving upon the prediction of features in the next image, we can get better matches and thus provide a more robust estimate of the system.

From a computation point of view, the Extended Kalman filter computation speed can be further increased by exploiting properties of symmetric matrix. Also further parallelism can be introduced in computation of measurements by using GPU to get faster speeds.

Radius estimation and other characteristics of the tree can be extracted from the mesh grid by performing addition estimation, which will be useful for tree surveying.

Loop closure in SLAM is a constantly researched upon topic, incorporating loop closure techniques with VSLAM can extended this project to give better results over a long time. Finally by incorporating potential ALB related features in the 3D model can further facilitate the USDA authorities in identifying probable tree candidates.

Bibliography

[1] Tim Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[3] J. Civera, A.J. Davison, and J. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, October 2008.

[4] Javier Civera, Oscar G. Grasa, Andrew J. Davison, and J. M. M. Montiel. 1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609631, 2010.

[5] 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*, 29(6):1052–1067, 2007.

[6] M. W M G Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[7] H. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[8] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, page 147151, 1988.

[9] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, March 1960.

[10] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Kalman filters for non-linear systems: a comparison of performance. *International Journal of Control*, 77(7):639–653, 2004.

[11] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[12] Z.C. Marton, R.B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*, pages 3218–3223, 2009.

[13] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005*, volume 2, pages 1508–1515 Vol. 2, 2005.

[14] J. Shi and C. Tomasi. Good features to track. In *, 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94*, pages 593–600, 1994.

[15] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, December 1986.

[16] H. Strasdat, J. M M Montiel, and A.J. Davison. Real-time monocular SLAM: why filter? In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657–2664, 2010.