TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

REAL TIME VISUAL LOCALIZATION AND MAPPING OF MOBILE ROBOT IN DYNAMIC ENVIRONMENT

Submitted By:
Nischal Maharjan (073 BEX 421)
Rashik Shrestha (073 BEX 432)
Sajil Awale (073 BEX 436)
Shrey Niraula (073 BEX 443)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN ELECTRONICS & COMMUNICATION / COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

April, 2021

PAGE OF APPROVAL

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled **"Real Time Visual Localization and Mapping of Mobile Robot in Dynamic Environment"** submitted by Nischal Maharjan (073 BEX 421), Rashik Shrestha (073 BEX 432), Sajil Awale (073 BEX 436), Shrey Niraula (073 BEX 443) in partial fulfilment of the requirements for the Bachelor's degree in Electronics Communication / Computer Engineering.

Supervisor,
Mr. Jitendra Kumar Manandhar
Lecturer
Department of Electronics and Computer Engineering

Internal Examiner,
Mr. Suman Sharma
Lecturer
Department of Electronics and Computer Engineering

External Examiner,
Manoj Ghimire
Co-Founder/CEO
Rara Labs

DATE OF APPROVAL

COPYRIGHT

ACKNOWLEDGEMENT

We would like to express our gratitude to our department, Department of Electronics and Computer Engineering as well as our major project supervisor **Jitendra Kumar Manandhar** for providing us the opportunity to carry out the major project entitled 'Real Time Visual Localization and Mapping of Mobile Robot in Dynamic Environment'.

We would also like to appreciate Robotics Club, IOE Pulchowk Campus for providing the necessary hardware equipment required in our projects.We are equally thankful to all the feedbacks and review from our teachers and supervisor which motivated us to further enhance our project.

At last, we would like to thank our friends, seniors and families who assisted and driven us for the completion of this project within the limited time frame.

ABSTRACT

Robot localization is an integral part in mobile robotics. It is the base for path planning and navigation tasks for robot and for AR/VR applications as well. SLAM has been a well known method for mapping the unknown environment and localizing yourself in the map. Visual SLAM, the category of SLAM, makes the use of visual sensors such as camera to perform SLAM. Such visual sensors are available at cheap cost nowadays and hence it is one of the most researched and popular topics in mobile robotics.

This project uses camera as its only sensor to build 3d map of entire room and localize itself in the built map. The map can then be used for navigation purposes within the mapped environment. Problems such as dynamically changing environment, varying lightening conditions, lack of textured environment are the hindrances for visual SLAM. Some of these problems has been well tackled in this project. Dynamic objects in the environment have been masked to minimize its effect. Light invariant feature extraction has been used to tackle with variations in lightening conditions.

Robot Operating System (ROS) has been used to communicate between various parallel processes and between robot and PC. Most of the high computational tasks have been done in PC. The robot then responses on the basis of commands given by processes which are running on PC.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

BA	Bundle Adjustment
BF	Brute Force
BoW	Bag of Words
BRIEF	Binary Robust Independent Elementary Features
CFF	Cascade Feature Fusion
CLG	Cascade Label Guidance
FAST	Features from Accelerated Segment Test
ICNet	Image Cascade Network
mIOU	Mean Intersection Over Union
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective N Point
PSPNet	Pyramid Parsing Network
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SIFT	Scale Invariant Feature Transform
SLAM	Simulataneous Localization And Mapping
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition

1 INTRODUCTION

1.1 Background

Localizing yourself correctly is the primary and the most important task in many fields like robotics and AR/VR applications. If a device/robot is aware about its location and the 3D map of the environment, it can perform other tasks like navigating in the environment, obstacle avoidance, placing augmented reality objects in the environment and many more.

Humans are gifted with beautiful and extremely powerful brain which can process the images taken from eyes and localize itself in the environment. Moreover, the brain has ability to memorize the entire 3D environment which helps us to navigate to our destination. Lots of researches have been carried out to mimic this ability of humans to an artificial bots. Yet, today's state of art system is still not able to match the level of human brain's ability to localize and navigate in 3D world.

But, robots has the freedom to use as many sensors of various types and as much memory as they want while humans are limited by their five senses and memory. Researchers have been using this advantage to match the localization ability of robots to that of humans. Expensive sensors like 3D lidars can help to increase the accuracy significantly.

Still, use of advance sensors are too costly and not practical to use in small robots and mobile phones. Hence, accurate localization using low cost sensor, less memory and less processing power has been today's big research problem.

1.1.1 Mapping, Localization and Path planning

Mapping refers to making a map of the surrounding environment which you can use later for localization and navigation purposes.

Localization refers to finding your position with respect to the prebuilt map of the surrounding.

Once the current pose and prebuilt map is known, one can use various algorithms to navigate from the current location to the desired destination. Its like using google maps. Google has already created maps for you and GPS can provide absolute localization in the map. Then when we search for way to any destination, it uses path planning algorithms to find best and fastest path to the destination.

1.1.2 Simultaneous Localization and Mapping

Imagine you lost your way (well, you lost your smart phone too). How will you navigate your way back to home? You will not know the way to return back as you are unaware of the new environment i.e there is no prebuild map in your mind. Also, you are unaware of where you are because you do not have any reference point in environment with respect to which you can localize yourself.

In such situation, person tries to roam around and gather information about local map, localizing in that map. For global localization propose, we need to find a specific standard landmark (like coffee shop we know, the known road junction) with respect to which absolute localization is possible.

This is very well known problem in robotics. An algorithm known as Simultaneous Localization and Mapping (SLAM) is used to tackle this problem. More about this algorithm in theory.

1.1.3 Problem with dynamic environment

Well, machine are not as smart as humans. It cannot figure out which part of the surrounding are just temporary and which part are permanently there. For example, a dumb machine might learn to turn right from the point where a dog was standing in order to reach its destination. Well, it was technically correct for that moment, but why will that dog stay in the same spot every time just to show path to the dumb machine.

This is a huge problem in the world of robotic navigation. A robust algorithm is needed to distinct between static and dynamic portion of the environment and store only those information that comes from permanent portions of environment in its map database.

1.1.4 Problem with change in lightning

As explained earlier, machine is dumb. It cannot recognize the same room at night which it saw earlier in sunlight. Basically the information about a picture of a room at day and night are stored using completely different binary patterns by the computer. It makes the navigation task much harder. The map which was made during day cannot be used for localization and path planning at night or in low light conditions. Many light invariant algorithms have been developed to tackle this problem, but still the efficiency of localization is drastically reduced when the lightening condition during the localization and mapping are completely different even in the same environment.

1.2 Objectives

The main objective of this project is to navigate in dynamic indoor environment using Visual sensors only. The objectives of this project are as follows:

1. To perform Visual SLAM in indoor environment
2. To localize robot in the presence of moving people
3. To perform the task with minimal use of computational resources
4. To deal with absolute re-localization problem, also known as *The kidnapped robot* problem
5. To deal with change in lighting conditions of the environment

1.3 Scope of the Project

Robotic navigation is widely used in different field robotics and computer vision. Self driving cars, autonomous drones, package delivering robot by Amazon, food delivering robots in restaurants and many more robotics applications uses SLAM for mapping and navigation purposes.

In the current situation of COVID-19, these types of autonomous robots can be used for reducing human interaction specially in hospitals. Moreover, in country like Nepal, using expensive sensors is not feasible. So, Visual SLAM would be the best option for navigation of autonomous bots. Use of cheap parts can help for the mass production of these robots in low cost which can be used for betterment and development of the country.

2 LITERATURE REVIEW

2.1 Visual SLAM

OpenVSLAM [17] is a graph based SLAM that uses orb features of each frame and track the position of camera on the basis of movement of the feature points in the corresponding frames.

ORB–SLAM [11], ORB-SLAM2 [12] use the similar techniques like openvslam which are a kind of indirect SLAM that carries out visual SLAM processing using local feature matching among frames at different time instants. ORB-SLAM3 [2] incorporates inertial data into the pose graph with landmarks as in ORB-SLAM2, which gives much more accuracy in localization specially in the cases like fast change in camera direction, dynamic environment and vibrations.

Some methods like RTAB-Map (Real-Time Appearance-Based Mapping) [9] uses graph based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location.

Meanwhile, LSD–SLAM [5] is a different approach of direct SLAM, which realizes visual SLAM processing directly exploiting brightness information of each pixel in images. It should be noted that the direct method does not have to explicitly extract any keypoints from images. Unlike the indirect method, the direct method can be correctly operated in more texture-less environments because it utilizes whole information from images. However, the direct method presents more susceptibility to changes in lighting conditions. Additionally, direct method achieves lower performance than the indirect one when using rolling shutter cameras.

2.2 Dynamic Object Detection

Each and every objects in the mapping gives some keypoints. This is fine until there are some dynamic objects in the environment that hinder the mapping process. Dynamic objects in the scene can be considered to be the target object of CNN based approach to filter them out. Mask-RCNN [8] is instance segmentation model to generate the mask for specified classes of objects. However it is far beyond the real time. Some SLAM algorithm such as DynaSLAM [1] makes use of MRCNN approach to filter out the objects while performing SLAM. Detectron2 [18], the model zoo provided by facebook AI research team provides the pretrained models based on Mask-RCNN providing good quality prediction on segmentation output but slow for inferencing on CPU environment limiting our target. CenterMask [10], the model zoo built upon the detectron2, introduces the anchor-free segmentation that can perform faster segmentation than detectron2. However, the performance of CenterMask is only good enough at highcore GPU Titan Xp, and is unable to achieve high speed inference on regular CPU processing.

Moreover, these methods [8, 14, 18] are all based on instance segmentation that tries to instantiate the pixels of even same class that is slower than semantic segmentation that performs classwise segmentation. On moving the focus to semantic methods, several methods exists such as [19, 15, 13, 20]. ICNet [20] accepts three different resolutions images in each branch and combine the coarse prediction map from low resolution branch image with finer details obtained from the high-resolution branch.

The actual implementation of removal of keypoints features from the dynamic objects are available in DynaSLAM [1]. It makes the use of MaskRCNN technique to mask out the dynamic objects. However the real time is not possible in this approach with CPU, hence more real time approach trading off the quality of segmentation is to be referred which we have tried to carry out.

3 THEORETICAL BACKGROUND

3.1 Image Formation

3.1.1 Camera Model

In the Computer Vision tasks, the basic starts with defining the model of the camera. On the basis of camera model the tasks are performed. The camera model refers to the mathematical algorithms that are used to convert the position of 3D points of the object into the 2D points in the image plane. If the camera parameters are known and we have 3D points then the camera model can successfully calculate where that point is positioned in the image. There are number of types of camera models such as equirectangular model, fisheye model, pinhole camera model etc. Pinhole camera model is explained below in detail.

3.1.2 Pinhole Camera Model

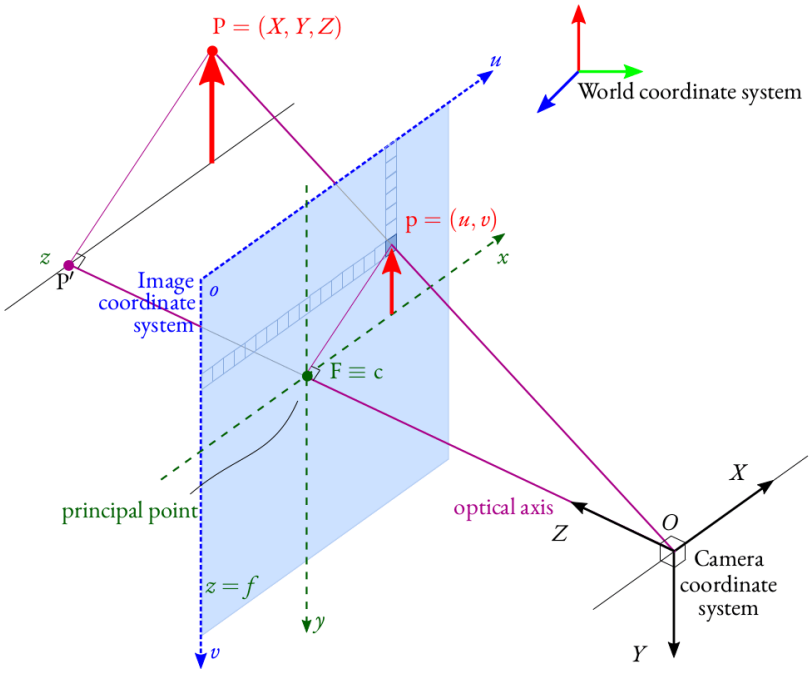


Figure 3.1: Pinhole Camera Model

Source: `https://lhoangan.github.io/assets/images/posts/2018-07-30/camera_model.png`

It is the simplest camera model which describes the mathematical relationship of the projection of points in 3D-space onto a 2D image plane. It is basically the perspective projection followed by the rotation of the image plane by 180 degrees.

Let (X, Y, Z) be the 3D point in the space which is projected into the image plane at (u, v) as shown in figure 3.1.Let f be the focal length of the camera i.e. the distance of image plane from the optical centre. Hence using properties of similar triangles,

$$\frac{u}{f} = \frac{X}{Z} \quad \text{and} \quad \frac{v}{f} = \frac{Y}{Z} \tag{3.1}$$

3.1.3 Camera Projection

The projection of 3D point into the image plane by the means of perspective projection can be represented by the following equation,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = L(K \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}) \tag{3.2}$$

where,
L is Lens configuration of camera(intrinsic parameter)
K is Spatial relationship between sensor and pinhole known as **Camera Matrix**(intrinsic parameter).
$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$ is the camera body configuration w.r.t. to world coordinate system(extrinsic parameters)
$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ in combination is termed as projection matrix.

For Camera Matrix, using equation (3.1) we have,

$$Zu = fX \quad \text{and} \quad Zv = fY \quad \text{and} \quad z = Z$$

In matrix form we get,

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.3}$$

This equation (3.3) projects 3D point in camera coordinate system into the image coordinate system. For further conversion into pixel coordinates we need to know the optical centre of the the camera, suppose (cx, cy). Let α_x and α_y be the pixel scaling factors and s be the slanting factor, when image is not normal to the optical axis. Then we get,

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & cx \\ 0 & \alpha_y & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

or,

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s' & cx \\ 0 & f_y & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.5}$$

The 3×3 matrix in equation (3.5) is known as camera matrix K.
For Ideal perspective projection, we have,

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s' & cx \\ 0 & f_y & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.6}$$

In case of distortion due to Lens configuration, we have two types of distortions; radial and tangential distortion which can be represented respectively as below,

$$u_{distorted} = u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad , \quad v_{distorted} = v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), and$$

$$u_{distorted} = u + [2p_1 uv + p_2(r^2 + 2u^2)] \quad , \quad v_{distorted} = v + [p_1(r^2 + 2v^2) + 2p_2 uv]$$

Hence distortion parameters are $[k_1\ ,\ k_2\ ,\ p_1\ ,\ p_2,\ k_3]$

3.2 Visual Features

Visual Features also known as feature points or corner points are the points in the images that are invariant under change in view, different illumination and change in scale. The visual features must be detection invariant as well as descriptor invariant. Detection invariant refers to the characteristic of being detected i.e. when a point is detected in one image it is to be detected in another one from same scene even if image differ in scale and orientation.The features must also be descriptor invariant i.e. the descriptor of the keypoints should not change significantly under view point changes and hence the descriptor can be used for matching the features.

There are various kind of visual features on the basis of the way they are computed. Some of them are SIFT(Scale Invariant Feature Transform), SURF(Speeded Up Robust Features), ORB(Oriented FAST and rotated BRIEF), KAZE features etc. SIFT and SURF are patented and slow. ORB being faster is most preferred for real time operations which has been used in our project. The brief explanation of ORB features is as below.

3.2.1 ORB Features

ORB (Oriented FAST and Rotated BRIEF) as name suggests, ORB extractor uses **FAST** algorithm to estimate the keypoints and uses **BRIEF**(Binary Robust Independent Elementary Features) to compute descriptor. Given a pixel of the image ORB compares its intensity with 16 surrounding pixels in a Bresenham circle of radius 3 as shown in figure 3.2.

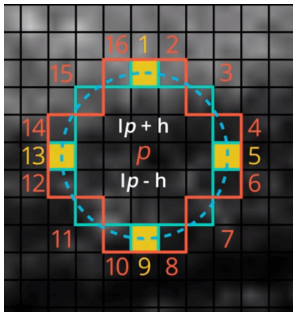


Figure 3.2: ORB Keypoint

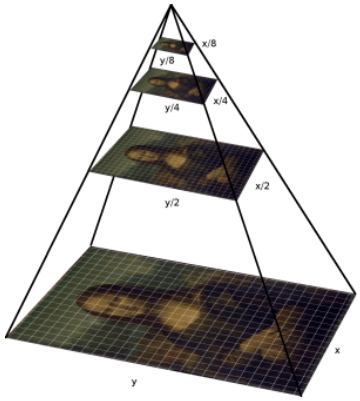


Figure 3.3: Image pyramid

Source: `https://miro.medium.com/max/358/1*sMlXGwLPQU60UldEiY-Mow.png`
`https://pyimagesearch.com/wp-content/uploads/2015/03/pyramid_example.png`

If I_p be the intensity of the pixel and h be the threshold the surrounding pixel is said to be more brighter if its intensity is greater then $(I_p + h)$ and less brighter if less than $(I_p - h)$ else consider of same level of brightness. The pixel is considered as the keypoint if at least 12 points either have greater intensity, or lower intensity than the pixel. Instead, by comparing only pixels at position 1,5,9 and 13 as show in figure 3.2 we can reduce the computational time. This portion is FAST algorithm which doesn't account orientation component and multiscale feature. Hence ORB uses multiscale pyraid in addition to FAST algorithm. The image pyramid is the multiscale representation of image which has sequence of downscaled images as shown in figure 3.3. The FAST algorithm detects points in each of scale level of pyramid hence keypoints will be located effectively at different scales. Thus property of scale invariance of the keypoints will be attained. For the rotation invariant property to be satisfied, the orientation attribute has to be assigned to the keypoints which is done on the basis of intensity centroid. The intensity weighted centroid of the patch with keypoint located at centre is computed. The orientation is the direction of the vector from the corner point to the so computed centroid. The moments of patch is given by

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y)$$

where, I(x,y) is the intensity of the pixel at position (x,y) and the centroid is computed on the basis of moment as

$$C = (m_{10}/m_{00}, m_{01}/m_{00})$$

Thus orientation is given by

$$\theta = atan2(m_{10}/m_{10}) \tag{3.7}$$

After the detection invariant keypoint has been selected now the descriptor for those points are to be specified. For which ORB uses BRIEF(Binary Robust Independent Elementary Feature) descriptor. BRIEF converts the image patch around the keypoint

into binary feature vectors hence they could be used to describe the keypoints. It describes a keypoint by the bit strings of length 256. Due to pixel level analysis, Gaussian smoothing is applied to eliminate high frequency noises. Then from predefined neighbourhood of the keypoint termed as patch pair of pixels are selected at random.The first pixel is selected using normal distribution with the standard deviation of sigma with keypoint as centre, whereas second pixel is selected similarly using normal distribution with spread of half of sigma centered at first pixel. The binary value is assigned to pair as below

$$\tau(p; x, y) = \begin{cases} 1: & \text{if } p(x) < p(y) \\ 0: & \text{otherwise} \end{cases}$$

The BRIEF feature vector is obtained as

$$f_n(p) = \sum_{1<i<n} 2^{i-1} \tau(p; x_i, y_i) \tag{3.8}$$

ORB introduces slight change into above mentioned method resulting in rBRIEF by steering according to the orientation of the keypoint computed using equation (3.7). For any binary feature at point (x_i, y_i) there is a $2 \times n$ matrix as

$$S = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{pmatrix}$$

Steered version of S is computed using the rotation matrix computed by θ.

$$S_\theta = R_\theta S$$

Hence using equation (3.8) steered BRIEF operator is computed as

$$g(p, \theta) = f_n(p) | (x_i, y_i) \, \epsilon \, S_\theta \tag{3.9}$$

ORB thus discretizes angles into 12 degree increments and lookup table of precomputed BRIEF features is created. The orientation θ must be consistent across views and appropriate set of points from S_θ will be used to compute descriptor of the keypoint. Like this the visual features are detected and describes from the image by the ORB algorithm.

3.3 Multiview Geometry

The Multiview geometry refers to intrinsic projective geometry between multiple views. If the number of views is to be 2 then it is termed as epipolar geometry. Epipolar Geometry between two camera views is the geometry of the intersection of image planes with the pencil of planes having baseline as axis, the line joining camera centres. It is independent of the scene structure but only depends upon the camera intrinsic parameters and the relative poses. This concept of epipolar geometry can be used to address following points

- **Correspondence Geometry**: Given the point p in first view how the p' in second view is constrained.

- **Camera Geometry**: Given the corresponding image points in two views what is the relative pose between the cameras.
- **Scene Geometry**: Given the correspondence and the camera projection matrix the position of point in 3D space.

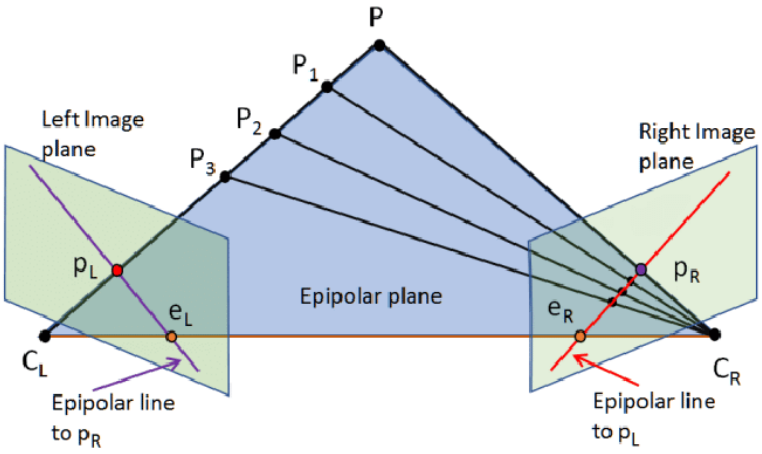


Figure 3.4: Epipolar Geometry [4]

Let us suppose a point P in 3D-space being viewed by two cameras positioned at C_L and C_R as shown in figure 3.4. These three points spans a plane π called ***Epipolar Plane***. It is the plane which consist of the baseline. The corresponding projected 2D point projected using equation (3.6) are p_L and p_R respectively. For point p_L in first view having same point as p_R in next view the epipolar geometry defines the relationship between them. For every point p_L in left view there exist a corresponding line in the right view known as ***Epipolar Line*** and vice versa. Epipolar line is the intersection of the an epipolar plane with the image plane. Here 3D points P, P_1, P_2, P_3 are projected into the epipolar line in the right view whereas they all are projected at a single point in the left view. Similarly the optical centre of one camera may be projected on the view of next camera. The point thus obtained in figure shown by e_L and e_R is termed as ***Epipole***. All epipolar lines intersect at the epipoles. Epipole is basically the vanishing point of the baseline direction. For further detail explanations refer [6].The mathematical representation of this relationship between two views can be understood with help of *Fundamental Matrix* and *Essential Matrix* explained below

3.3.1 Fundamental Matrix

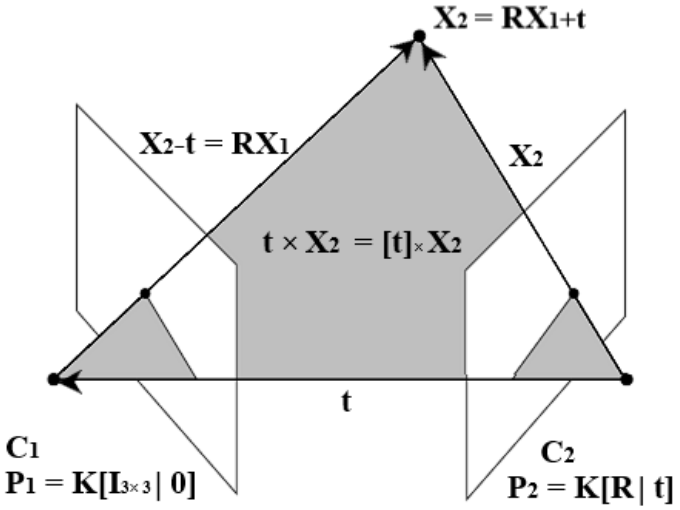


Figure 3.5: Epipolar Constraint

Fundamental Matrix is the algebraic representation of the epipolar geometry. Suppose a two cameras with centre at C_1 and C_2 have relative rotation and translation of **R and t** as shown in figure 3.5. 3D point have a coordinate X_1 with respect to camera C_1 and X_2 with respect to second camera. The vector **t** is the translation vector shown by the arrow in the figure. The ray back projected from 3D point to the second camera is defined by vectorX_2. Hence from vector addition the ray back projected on first camera is $X_2 - t$.
As relative pose between cameras is known ,

$$X_2 = RX_1 + t$$

$$or, X_2 - t = RX_1$$

Hence these three vector span a epipolar plane as show in figure 3.5, whose surface normal is given by cross product of any two vectors in the plane as,

$$t \times X_2 = [t]_\times X_2$$

where $[t]_\times$ is the skew symmetric matrix for the vector t i.e. cross product is represented as the matrix multiplication.
Since surface normal is perpendicular with the epipolar plane hence,

$$(X_2 - t)^T [t]_\times X_2 = 0$$

$$or, (RX_1)^T [t]_\times X_2 = 0$$

$$or, X_1^T R^T [t]_\times X_2 = 0$$

$$or, -X_2^T [t]_\times R X_1 = 0$$

$$or, X_2^T [t]_\times R X_1 = 0$$

$$or, X_2^T E X_1 = 0 \tag{3.10}$$

where $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ is the essential matrix which will be discussed further below.
Let K be the camera matrix as defined in equation (3.5) and x_1 and x_2 be the corresponding projected point of same 3D point into image planes of camera C_1 and C_2 respectively then,

$$x_1 = KX_1 \quad \text{and} \quad x_2 = KX_2$$

Hence, equation (3.10) becomes

$$x_2^T K^{-T} E K^{-1} x_1 = 0$$

$$or, x_2^T F x_1 = 0 \tag{3.11}$$

where $\mathbf{F} = \mathbf{K^{-T} E K^{-1}}$ is the fundamental matrix. and equation (3.11) is the epipolar constraint that the corresponding points in two views must satisfy.

Properties of Fundamental Matrix

- Fundamental Matrix is the 3×3 matrix of rank 2
- Given corresponding points in two views the fundamental matrix satisfies equation (3.11)
- For any point x_1 in first image the corresponding epipolar line in second image is given by $l_2 = Fx_1$ and similarly $l_1 = F^T x_2$ is epipolar line in first image for points x_2 in second image.
- If e_1 and e_2 be the epipoles then $Fe_1 = 0$ and $e_2^T F = 0$. i.e, e_1 is right null vector of F and e_2 is left null vector of F.
- If $\mathbf{F}$ is fundamental matrix for camera pair (C_1, C_2) then $\mathbf{F^T}$ is fundamental matrix for opposite pair (C_2, C_1)

Estimation of Fundamental Matrix from 2D correspondences
Given 2D correspondence between two views fundamental matrix can be estimated. Suppose $x_1 = (u_i, v_i, 1)$ and $x_2 = (u_i', v_i', 1)$ be the corresponding i^{th} homogeneous 2D points in the images from the two views. Then from equation (3.11) we have,

$$x_2^T F x_1 = 0$$

$$or, \begin{bmatrix} u_i' & v_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f23 \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = 0$$

$$or, \begin{bmatrix} u_1'u_1 & u_1'v_1 & u_1' & v_1'u_1 & v_1'v_1 & v_1' & u_1 & v_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ u_n'u_n & u_n'v_n & u_n' & v_n'u_n & v_n'v_n & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Above equation is in the form of

$$Ax = 0$$

This is least square problem(See Appendix A.2) which can be solved by calculating the pseudo inverse of the matrix A. This can be done simply by the use of technique known as *Singular Value Decomposition.*(See Appendix A.1). Like this given the correspondence points between two images the fundamental matrix representing the relation between those two views can be estimated. We require to estimate 9 unknowns and each correspondence provides a single constraint. But due to homogeneous representation of image points, the degree of freedom is reduced to 8 because of scale factor. Hence we require only 8 constraint to solve the equation i.e, if 8 correspondence is present we are able to estimate the fundamental matrix. But due to noise in the points, we use number of points and solve them using least square problem and is optimized based on RANSAC algorithm (See Appendix A.3)

3.3.2 Essential Matrix

Essential Matrix defines the algebraic representation of epipolar geometry similar to the fundamental matrix. It is the specialization of the fundamental matrix in case of normalised image coordinates when camera matrix is known. When camera matrix K is given, the image points can be multiplied by the inverse of the camera matrix to obtain the normalised coordinates. Hence in case of normalised coordinates, the fundamental matrix converts into essential matrix. The property of essential matrix is similar to that of fundamental matrix.
As determined in equation (3.10) we have

$$E = [t]_{\times} R \tag{3.12}$$

Essential matrix can be estimated from the fundamental matrix given the camera calibration matrix **K** as

$$E = K^T F K \tag{3.13}$$

3.3.3 Triangulation

Given 2D-2D corresponding point between two images of same scene and relative position of second camera with respect to the first one, 3D coordinate of that point can be calculated. This is known as *Triangulation.*
Suppose points x_1 and x_2 represent same 3D point X in two images captured by two different camera having relative rotation and translation represented by matrix R and t
Let

$$P_1 = K_1[I_{3*3}|0_3]$$

then

$$P_2 = K_2[R_{3*3}|t_{3*1}]$$

From projective transformation we have,

$$\lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \tag{3.14}$$

and,

$$\lambda \begin{bmatrix} x_2 \\ 1 \end{bmatrix} = P_2 \begin{bmatrix} X \\ 1 \end{bmatrix} \tag{3.15}$$

Taking cross product on both sides we have,

$$\begin{bmatrix} x_1 \\ 1 \end{bmatrix} \times P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0 \quad \text{and} \quad \begin{bmatrix} x_2 \\ 1 \end{bmatrix} \times P_2 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0$$

Combining above equations we have,

$$\begin{bmatrix} \begin{bmatrix} x_1 \\ 1 \end{bmatrix} \times P_1 \\ \begin{bmatrix} x_2 \\ 1 \end{bmatrix} \times P_2 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = 0 \tag{3.16}$$

Equation 3.16 is in the form of

$$Ax = 0$$

This equation is least square problem(See Appendix A.2). This can be simply solved by the use of technique known as *Singular Value Decompostion*(See Appendix A.1).

3.4 Pose Estimation

The concept of multiview geometry or epipolar geometry can be used to estimate the pose of the camera with respect to the initial pose of camera. The goal of estimating pose(Position and orientation) can be achieved using various methods using the correspondence between image points as well as 3D scene points explained below.

3.4.1 Pose from 2D correspondence

In this method we use the concept that Essential Matrix incorporates the translation and rotation matrix as shown in equation (3.12). Given the 2D-2D correspondence between two images of the same scene we are able to compute the relative pose between two camera positions. This method can be broken down into two steps, first estimating the essential matrix from the correspondence and then decomposing the essential matrix into rotation and translation matrices.

As shown in 3.3.1 we can compute fundamental matrix from 2D correspondence and then use equation (3.13) to estimate essential matrix from the fundamental matrix. Or we can directly estimate essential matrix by normalising the image points using camera

matrix K. The next step is to recover Rotation and Translation from the the estimated Essential Matrix.

If we consider Initial camera position to be at the origin of world coordinate system as shown in figure 3.5 then the image of origin in the second camera will be *Epipole*. Since the translation vector is along the baseline, the epipole correspond to translation. we know,

$$\begin{bmatrix} x \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Considering normalised points, then t will be epipole in second image because,

$$or, \begin{bmatrix} x \\ 1 \end{bmatrix} = [R\ |t] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = t$$

Hence using the property of essential matrix and epipole,

$$t^T E = 0$$

Hence translation being epipole in second image is the left nullspace of essential matrix. Thus from SVD (See Appendix A.1) if $E = UDV^T$ where U in term of column vetors be $U = [u_1, u_2, u_3]$ then $t = u_3$ *or* $-u_3$ i.e., $U = [u_1, u_2, t]$

$$U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T = [t]_\times R$$

The $[t]_\times$ operation is converted in terms of U. This cross product result transformation of arbitrary vector into the space perpendicular to t itself. For any vector a new orientation is defined by space u_1, u_2 *and* t since U is orthogonal matrix. Hence any vector is transformed into this space using U^T and then we remove the t elements so that vector will remain perpendicular to the t. Then we rotate by 90 degree in the space of u_1 and u_2, which is finally transformed back to original space using U matrix. Therefore,

$$U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T = \left(U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T \right) R$$

Using SVD for rotation matrix R we have,

$$U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T = \left(U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T \right) (UYV^T)$$

$$or, U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} YV^T$$

$$or, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} Y$$

Hence possible solutions for Y are

$$or, Y = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} or \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$$

Therefore, we have

$$R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T \quad or \quad R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T V^T$$

So there are four possible results of the decomposition as

$$R = UYV^T \quad t = u_3$$

$$R = UY^TV^T \quad t = u_3$$

$$R = UYV^T \quad t = -u_3$$

$$R = UY^TV^T \quad t = -u_3$$

If det(R) = -1 then t = -t and R = -R. Here we have four configuration of cameras. This ambiguity is removed by carrying out the triangulation using computed pose and the configuration which gives point in front of camera is the optimal choice. Like this we are able to compute pose of camera from 2D correspondence using the concept of epipolar geometry and essential matrix.

3.4.2 Pose from 2D-3D Correspondence(Linear PnP)

Given the corresponding 2D-3D correspondence the pose of camera can be estimated. This method is simple form of perspective n point algorithm also known as **Linear PnP**. For 3D points in world coordinate system obtained after triangulation if we have corresponding 2D keypoints in image present in the camera coordinate system, then using this relation between 2D and 3D points and the concept of camera projection we are able to estimate the pose of the camera in the world coordinate system.

Let $x_i \rightarrow X_i$ be the 2d-3D corresponding points and $P_1 = K_1[I_{3*3}|0_3]$ be the projection matrix of the camera. Then from projective transformation we have,

$$\lambda \begin{bmatrix} x \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Taking cross on both sides we have,

$$\begin{bmatrix} x \\ 1 \end{bmatrix} \times P \begin{bmatrix} X \\ 1 \end{bmatrix} = 0$$

$$or, \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} \tilde{X} = 0$$

where $\tilde{X}$ is the 3D homogeneous point in four dimension. and P_1, P_2, P_3 are the 3 rows of projection matrix respectively.

$$or, \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \begin{bmatrix} P_1\tilde{X} \\ P_2\tilde{X} \\ P_3\tilde{X} \end{bmatrix} = 0$$

$$or, \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} P_1\tilde{X} \\ P_2\tilde{X} \\ P_3\tilde{X} \end{bmatrix} = 0$$

$$or, \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix}_{3\times3} \begin{bmatrix} \tilde{X}^T & 0_{1\times4} & 0_{1\times4} \\ 0_{1\times4} & \tilde{X}^T & 0_{1\times4} \\ 0_{1\times4} & 0_{1\times4} & \tilde{X}^T \end{bmatrix}_{3\times12} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix}_{12\times1} = 0$$

$$or, \begin{bmatrix} 0 & -\tilde{X}^T & v\tilde{X}^T \\ \tilde{X}^T & 0 & -u\tilde{X}^T \\ -v\tilde{X}^T & u\tilde{X}^T & 0 \end{bmatrix}_{3\times12} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix}_{12\times1} = 0 \tag{3.17}$$

This equation takes form of **least square problem** as $Ax = 0$ which is solved using **Singular Value Decomposition** (See Appendix A.2). Each correspondence gives 2 constraints therefore to compute 12 unknown we require at least 6 point correspondence. Hence we have computed the elements of projection matrix. Now we need to extract the Rotation and Translation from the Projection matrix given the camera matrix K.
We know,

$$P = K[R\ |t]$$
$$or, K^{-1}P = [R\ |t]$$

Hence

$$R = K^{-1}P_{1:3} \quad \text{and} \quad t = K^{-1}P_4$$

Since R must be orthogonal matrix with determinant 1 it must be cleaned up and translation vector must be scaled.

$$R = UDV^T \quad \text{Using SVD}$$
$$R_c = UV^T, \quad t_c = t/D_{1,1} \quad if det(UV^T) = 1$$
$$R_c = -UV^T, \quad t_c = -t/D_{1,1} \quad if det(UV^T) = -1$$

Like this we can estimate the pose of camera given 2D-3D point correspondence. RANSAC algorithm is used to reject the outliers and compute accurate pose of the camera.

3.4.3 Pose from 3D-3D Correspondence(Procrustes Problem)

Given two set of corresponding 3D points

$$A = \{a1, a2, ..., an\}$$

$$B = \{b1, b2, ..., bn\}$$

we find the scaling (s), rotation(R) and translation(T) transformation called similitude transformation that satisfies

$$A_i = sRB_i + T \tag{3.18}$$

This method is also known as Procrustes Problem. Given the no of correspondence $N > 3$, we can compute the required R and T by solving the following minimization problem.

$$\min_{R,T} \sum_{i=1}^{N} ||A_i - RB_i - T||^2 \tag{3.19}$$

Hence differentiating w.r.t T, we obtain translation as the difference between the centroids as,

$$T = \frac{1}{N}\sum_{i}^{N} A_i - R\frac{1}{N}\sum i^N B_i = \bar{A} - R\bar{B}$$

Hence the objective function can be written as

$$\min_{R} ||A - RB||_F^2$$

where

$$A = (A_1 - \bar{A}, A_2 - \bar{A}, ..., A_n - \bar{A})$$
$$B = (B_1 - \bar{B}, B_2 - \bar{B}, ..., B_n - \bar{B})$$

This the Frobenius norm which can be represented in term of trace as,

$$||A - RB||_F^2 = tr(AA^T) + tr(BB^T) - tr(RBA^T) - tr(AB^TR^T)$$

The last two negative terms are equal , hence the minimization problem can be converted to maximization problem.

$$\max_{R} tr(RBA^T)$$

Here, RBA^T is a 3×3 matrix. If SVD of BA^T is USV^T and $Z = V^TRU$ then,

$$tr(RBA^T) = tr(RUSV^T) = tr(ZS) = \sum_{1}^{3} Z_{ii}S_i <= \sum_{1}^{3} S_i$$

Since Z is orthogonal matrix its diagonal elements are less than or equal to 1. Hence the $tr(RBA^T)$ is maximized when Z is identity matrix.

$$Z = V^T RU = I$$

$$R = U^T V$$

Like this we can estimate the translation and rotation between two coordinate system given the point corresponding points.

3.5 Graph Optimization

3.5.1 Introduction

A graph is basically collection of numbers of nodes connected to each other by edges. Each node of the graph represents a **state variable** to optimize, each edge between two variables represents a pairwise **observation** of the two nodes it connects. The state variables and observations can be scalar quantity or high dimensional quantity. For example, for a robot moving in floor, $[x, y, yaw]$ can be state variable and $[linear\_velocity, angular\_velocity]$ can be measurement. Measurement in an edge depends only on two state variables in nodes it connects.

The overall goal in these problems is to find the configuration of parameters or state variables that maximally explain a set of measurements affected by Gaussian noise

3.5.2 Maximum Likelihood Estimation

For simplicity, let us take single edge of graph having measurement z_{ij}, connecting two state variables (i.e parameters) x_i and x_j as shown in figure 3.6.

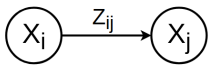


Figure 3.6: Single edge graph

For any initial guess of parameters xi and xj, we can find the estimated value of zij using measurement model. The difference between this estimated zij and measured zij gives the error term, which we aim to minimize. In case of visual slam this error term is actually the re-projection error.

Estimated zij is the function of xi and xj represented by P,

$$\hat{z}_{ij} = P(x_i, x_j) \tag{3.20}$$

Since, the measurement is affected by Gaussian noise, the error distribution can be represented as,

$$L(z_{ij}, \hat{z}_{ij}) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp \frac{-(z_{ij} - \hat{z}_{ij})^2}{2\sigma_{ij}^2} \tag{3.21}$$

where, σ_{ij} = Standard Deviation of measurement noise

Now, likelihood of all the measurement terms can be obtained as the product of likelihood of each individual measurements,

$$L = \prod_{i,j} L(z_{ij}, \hat{z}_{ij})$$

$$L = \prod_{i,j} L(z_{ij}, P(x_i, x_j)) \tag{3.22}$$

The problem turns to be maximizing the above likelihood function to get best estimate of $\mathbf{x}$. Where, $\mathbf{x} = [x_0, x_1, x_2, ...x_n]$ represents array of all the unknown parameters.

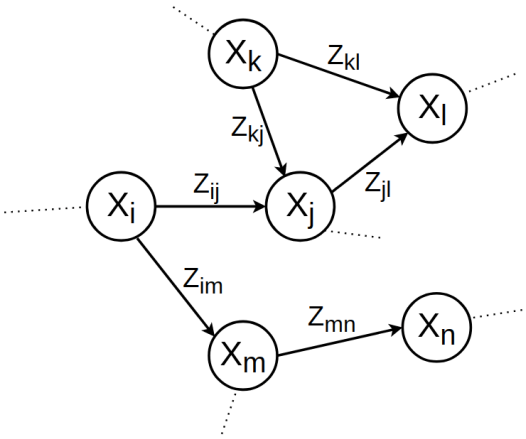


Figure 3.7: Graph representation

Best estimate of $\mathbf{x}$ is given by,

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \prod_{i,j} L(z_{ij}, P(x_i, x_j))$$

In order to get rid of the exponential root e and turn this maximization problem to be a minimization problem, we minimize the $-log$ likelihood instead:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} -log(\prod_{i,j} L(z_{ij}, P(x_i, x_j)))$$

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} -log(\prod_{i,j} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp \frac{-(z_{ij} - P(x_i, x_j))^2}{2\sigma_{ij}^2})$$

Here, $e_{ij} = z_{ij} - P(x_i, x_j)$ is the error term.
e_{ij} is the function of x_i, x_j and z_{ij} i.e $e_{ij} = e_{ij}(x_i, x_j, z_{ij})$
Since, x_i and x_j are only varying quantities and z_{ij} will always be constant for particular e_{ij}, we will write $e_{ij} = e_{ij}(x_i, x_j, z_{ij})$ to simplify the notation.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} -log(\prod_{i,j} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp \frac{-e_{ij}^2}{2\sigma_{ij}^2})$$

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} -\sum_{i,j} log(\frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp \frac{-e_{ij}^2}{2\sigma_{ij}^2})$$

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} -\sum_{i,j} log(\frac{1}{\sqrt{2\pi\sigma_{ij}^2}}) + \frac{-e_{ij}^2}{2\sigma_{ij}^2}$$

Here, the first term and coefficient of second term (1/2) are constant terms (i.e independent of $\mathbf{x}$), hence can be removed.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i,j} \frac{e_{ij}^2}{\sigma_{ij}^2}$$

Here,

$$\Omega_{ij} = \frac{1}{\sigma_{ij}^2}$$

Ω_{ij} is called information matrix, which is just the inverse of covariance matrix σ_{ij}^2

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i,j} e_{ij}^T \Omega_{ij} e_{ij} \tag{3.23}$$

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} F(\mathbf{x}) \tag{3.24}$$

3.5.3 Optimization

Let $\check{\mathbf{x}}$ be the initial guess value of $\mathbf{x}$. i.e $\check{\mathbf{x}} = [\check{x_0}, \check{x_1}, \check{x_2}, ...\check{x_n}]$

Then we need to find the best estimate of change in $\mathbf{x}$ i.e $\Delta\hat{\mathbf{x}}$, which gives minimum value of $F(\check{\mathbf{x}} + \Delta\mathbf{x})$.

$$\Delta\hat{\mathbf{x}} = \arg\min_{\Delta\mathbf{x}} F(\check{\mathbf{x}} + \Delta\mathbf{x})$$

We have,

$$F(\check{\mathbf{x}} + \Delta\mathbf{x}) = \sum_{i,j} e_{ij}(\check{x_i} + \Delta x_i, \check{x_j} + \Delta x_j)^T \Omega_{ij} e_{ij}(\check{x_i} + \Delta x_i, \check{x_j} + \Delta x_j)$$

Here, e_{ij} is non linear function. So, we approximate it at $x = x_0$ using first order taylor expansion.

$$e_{ij}(\check{x_i} + \Delta x_i, \check{x_j} + \Delta x_j) \approx e_{ij}(\check{x_i}, \check{x_j}) + \left.\frac{de_{ij}}{dx_i}\right|_{x_i=\check{x_i}, x_j=\check{x_j}} \Delta x_i + \left.\frac{de_{ij}}{dx_j}\right|_{x_i=\check{x_i}, x_j=\check{x_j}} \Delta x_j$$

$$e_{ij}(\check{x_i} + \Delta x_i, \check{x_j} + \Delta x_j) \approx e_{ij}(\check{\mathbf{x}}) + J_{ij}\Delta\mathbf{x}$$

where,

$$\check{\mathbf{x}} = [\check{x_0}, \check{x_1}, \check{x_2}, ...\check{x_n}]$$
$$\Delta\mathbf{x} = [\Delta x_0, \Delta x_1, \Delta x_2, ...\Delta x_n]$$

$J_{ij} = \left[\frac{de_{ij}}{dx_0}, \frac{de_{ij}}{dx_1}, \frac{de_{ij}}{dx_2}, ...\frac{de_{ij}}{dx_n}\right]$ is the Jacobian of e_{ij} computed at $\check{\mathbf{x}}$

Note that, e_{ij} still depends upon only x_i and x_j, not the whole set of $\mathbf{x}$. And, J_{ij} is very sparse array as e_{ij} changes with respect to x_i and x_j only.

So, we have

$$F(\check{\mathbf{x}} + \Delta\mathbf{x}) \approx \sum_{i,j} (e_{ij}(\check{\mathbf{x}}) + J_{ij}\Delta\mathbf{x})^T \Omega_{ij} (e_{ij}(\check{\mathbf{x}}) + J_{ij}\Delta\mathbf{x})$$

$$F(\check{\mathbf{x}}+\Delta\mathbf{x}) \approx \sum_{ij} \{e_{ij}(\check{\mathbf{x}})^T \Omega_{ij} e_{ij}(\check{\mathbf{x}}) + [2e_{ij}(\check{\mathbf{x}})^T \Omega_{ij} J_{ij}(\check{\mathbf{x}})]\Delta\mathbf{x} + \Delta\mathbf{x}^T [J_{ij}(\check{\mathbf{x}})^T \Omega_{ij} J_{ij}(\check{\mathbf{x}})]\Delta\mathbf{x}\}$$

Put,

$$c_{ij} = e_{ij}(\check{\mathbf{x}})^T \Omega_{ij} e_{ij}(\check{\mathbf{x}})$$
$$b_{ij} = e_{ij}(\check{\mathbf{x}})^T \Omega_{ij} J_{ij}(\check{\mathbf{x}})$$

$$H_{ij} = J_{ij}(\check{\mathbf{x}})^T \Omega_{ij} J_{ij}(\check{\mathbf{x}})$$

We get,

$$F(\check{\mathbf{x}} + \Delta\mathbf{x}) \approx \sum_{i,j} c_{ij} + 2b_{ij}\Delta\mathbf{x} + \Delta\mathbf{x}^T H_{ij} \Delta\mathbf{x}$$

Put,

$$c = \sum_{i,j} c_{ij} \qquad b = \sum_{i,j} b_{ij} \qquad H = \sum_{i,j} H_{ij}$$

We get,

$$F(\check{\mathbf{x}} + \Delta\mathbf{x}) \approx c + 2b\Delta\mathbf{x} + \Delta\mathbf{x}^T H \Delta\mathbf{x} \tag{3.25}$$

Equation 3.25 is in quadratic form (analogous to $(c + bx + ax^2)$). So, its minimum value exists when its derivative term equals to zero.

$$\left.\frac{\delta F(\check{\mathbf{x}} + \Delta\mathbf{x})}{\delta \Delta\mathbf{x}}\right|_{\Delta\mathbf{x}=\Delta\hat{\mathbf{x}}} = 0$$

$$\left.\frac{\delta (c + 2b\Delta\mathbf{x} + \Delta\mathbf{x}^T H \Delta\mathbf{x})}{\delta \Delta\mathbf{x}}\right|_{\Delta\mathbf{x}=\Delta\hat{\mathbf{x}}} = 0$$

$$2b + 2H\Delta\hat{\mathbf{x}} = 0$$

$$H\Delta\hat{\mathbf{x}} = -b \tag{3.26}$$

Equation 3.26 can be solved to find $\Delta\hat{\mathbf{x}}$. Now, the previous guess value can be updated by adding the estimated change.

$$\boxed{\hat{\mathbf{x}} = \check{\mathbf{x}} + \Delta\hat{\mathbf{x}}} \tag{3.27}$$

This process can be repeated again and again iteratively to get best estimate of $\mathbf{x}$.

3.5.4 Application

Graph optimization can be used in wide range of problems that can be modeled as graph. It can be used in fields like Simultaneous Localization and Mapping (SLAM), Bundle Adjustment (BA).

3.6 Graph based SLAM with Landmarks

3.6.1 Modelling Graph

The SLAM problem can be modeled as graph with number of edges and vertices, which can then be solved by using graph optimization algorithms. Visual SLAM has two types of unknowns and two types of knowns.

Two types of unknowns are:

- Pose of robot (6 dim)
- Position of landmark (3 dim)

Two type of known values are:

- 2D projection of landmark to robot pose (2 dim)
- Odometry between robot poses (6 dim)

The unknown terms are represented by the vertices in graph and the known terms are represented by edges in the graph. The known terms are actually the measurement values i.e the image of landmark taken by camera positioned at robot pose (in case of Visual SLAM) and relative transformation between two robot poses given by any type of sensor that gives odometry information like wheel encoders and IMU. The unknown terms are the state variables which needs to be optimized based on known measurement values.

A measurement depends only on the relative location of two state variables, e.g., an odometry measurement between two consecutive poses depends only on the connected poses. Similarly, a measurement of a 3D point or landmark depends only on the location of the observed point in the world and the position of the sensor.

The key difference between SLAM and BA is, BA lacks second type of known quantity i.e odometry information between robot poses. So, BA solely depends upon bunch of images without any information of how camera has moved while taking those images.

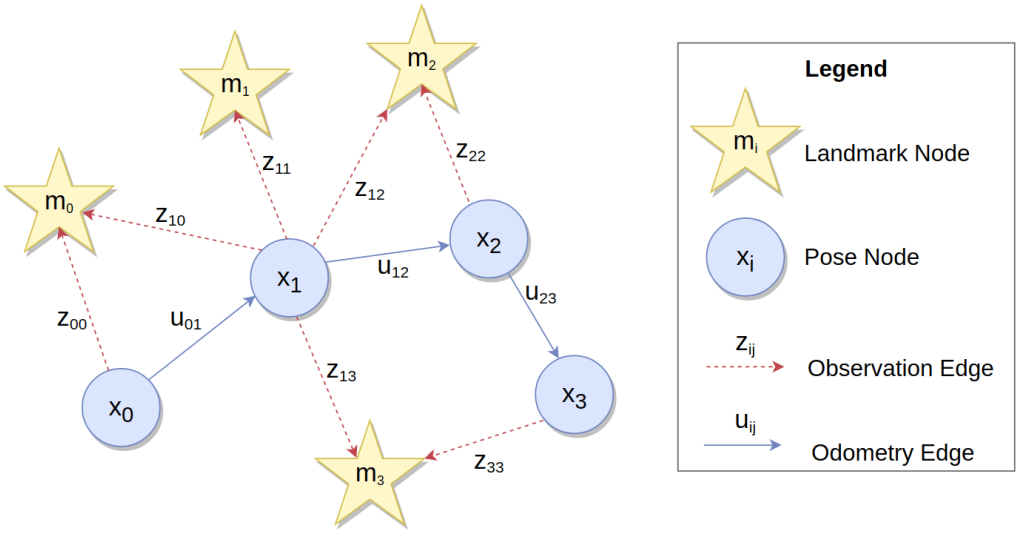


Figure 3.8: Graph based representation of Visual SLAM

Graph based SLAM using landmarks uses same technique of graph optimization as explained earlier. Just a difference is, now we have different varieties of nodes and edges.

3.7 Image Segmentation

Image segmentation is process to segment the parts of the image into categories. It classifies each pixel to corresponding classes. This is different than the classification and detection problem where we either needs to find whether particular object is in that image and anchor them with bounding box. In this, we need the pixel wise classification.

3.7.1 Types of Image Segmentation

Image segmentation methods are simply categorized into Semantic segmentation Instance Segmentation methods. The semantic segmentation is class level segmentation methodology in which the objects that belongs to the same class are identified with same mask. When each instances (objects) of that same class are again distinguished with different mask, it is termed as Instance Segmentation.

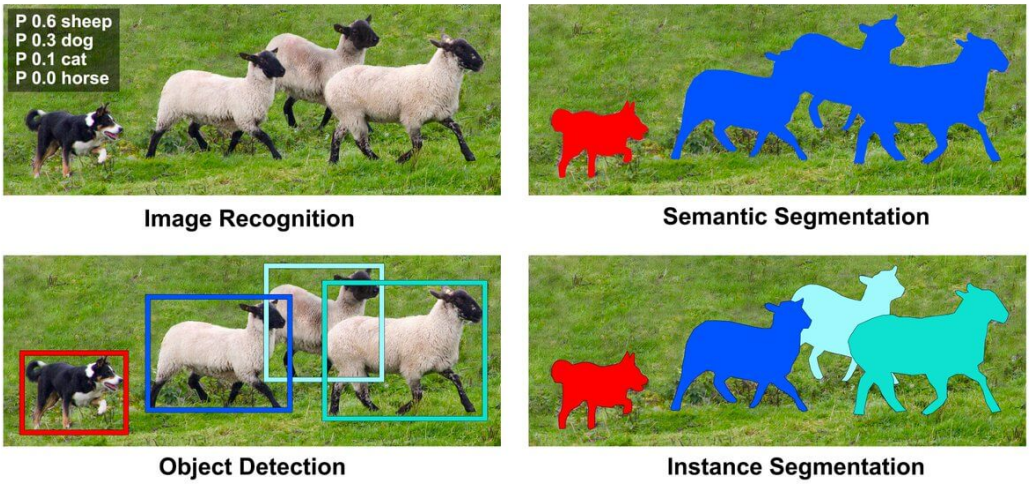


Figure 3.9: Differences between Detection, Classification, Segmentation

Source: https://files.ai-pool.com/d/DV8TLgkWsAEGsEs.jpg

Figure 3.9 clearly explains the concept between semantic segmentation and instance segmentation. In the semantic segmentation, all the sheep are given the same mask (blue colored mask), but in the instance segmentation, each sheep is assigned the different masks(different colored masks). The figure 3.9 also highlights the differences between the object recognition and object detection and segmentation methods.

3.8 Metrics used for Image Segmentation

Several metrics exists for the image segmentation task. The simplest one is pixel accuracy. There exists advanced metrics such as Jaccard Index, Dice Coefficients etc. Jaccard index, also called mIOU is common benchmark metric to represent the segmentation precision. It has been used in segmentation and classification task based on ResNet, PSPNet and ICNet as well.

3.8.1 Pixel Accuracy

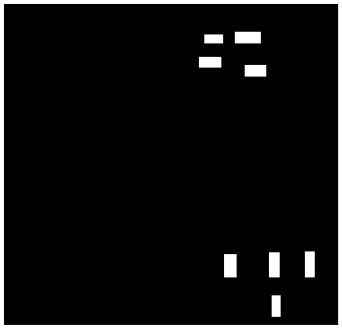


Figure 3.10: Incorrectness of pixel accuracy

Pixel accuracy, or simply accuracy is percentage of how much pixels are correctly classified. It is easiest metric which does not properly reflect the correctness of image segmentation. To demonstrate, consider two classes 'car' and 'background' colored white and black as seen in the figure 3.10. Let us suppose that out of 100% area, car occupies 15% of area and background occupies remaining 85% area. In such case, the pixel accuracy will be atleast 85% because system could classify the 85% of the background image. So, pixel accuracy does not specifically target to the desired class and rather gives the overall measure.

3.8.2 Mean Intersection Over Union (mIOU)

mIOU or Jaccard Index is the mean of IOU of each segmented classes. IOU gives the ratio of intersected area to the union of area between the segmented image and ground truth image. mIOU takes the mean of each IOU.

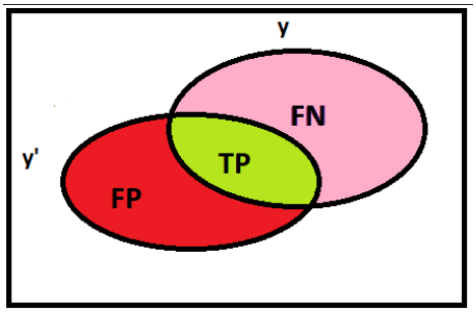


Figure 3.11: IOU concept

The concept of mIOU is explained in figure 3.11. Let us consider y be the ground truth object to be segmented and y' be the predicted mask for that object outputted by the segmentation system. In figure 3.11, there are regions for False Positive(FP), True Positive(TP) and False Negative(FN). FN is the region where system failed to classify. This means that, actual object do exist but the system has classified to be false. FP is the region where system has falsely classified to be true. TP is the region where system actually predicted true. Hence, IOU is given by,

$$\begin{aligned} \text{IOU} &= \frac{\text{Area of Intersection}}{\text{Area of Union}} \\ &= \frac{TP}{FP + TP + FN} \end{aligned} \tag{3.28}$$

When the mean of IOU of M objects are taken, it is considered as mIOU given as:

$$\text{For M objects (classes), mIOU} = \frac{\sum_{i=0}^{M}(IOU_i)}{M} \tag{3.29}$$

The equation 3.29 represents the mean IOU calculation where mean is taken across every classes present.

3.8.3 Dice Coefficient

Even though mIOU is better prediction metric than pixel accuracy, it does not tries to penalize the effects. Dice Coefficient also known as F1 score is harmonic mean of precision and recall. Being harmonic mean, it tries to penalize the effect of minimum valued term. Like the IoU, they both range from 0 to 1, with 1 signifying the greatest similarity between predicted and truth.

Dice Coefficient is given as:

$$\begin{aligned} \text{Dice Coefficient} &= \frac{2 \text{ Area of intersection}}{\text{Area of Union + Area of Intersection}} \\ &= \frac{2TP}{FP + TP + FN + TP} \\ &= \frac{2TP}{FP + 2TP + FN} \end{aligned} \tag{3.30}$$

The equation 3.30 is same as F1 score which is harmonic mean of precision and recall.

3.9 Theoretical Knowledge before Segmentation Model

3.9.1 ResNet

Emergence of ResNet has created huge impact on the deep learning and computer vision field. The development of ResNet has allowed to train the convolution layers

to much deeper level and opened the path for more advanced object detection and segmentation task. The problem of vanishing/exploding gradients has been addressed by establishing identity mapping with the help of skip connections. The figure 3.12 from original ResNet paper shows that for regular plain network, the train error (shown in thin line) is more for deeper network in comparison to ResNet where train error is smaller for deeper layer. This demonstrates the significance of the identity mapping in ResNet. The figure 3.12 represents train and validation error variation between the plain network and ResNet where thin line represents the training error and thick line represents the validation error.

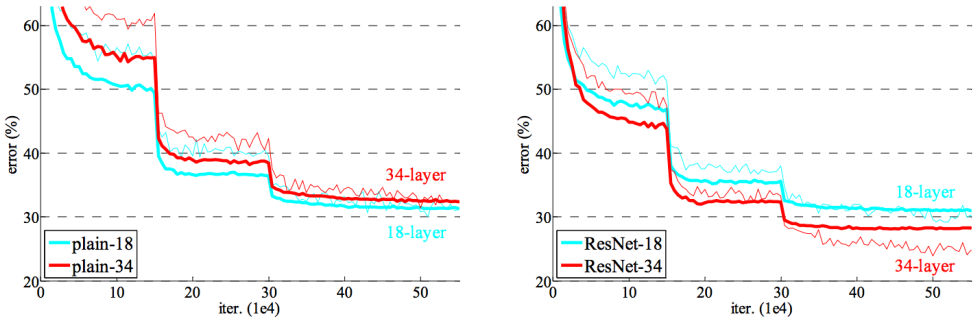


Figure 3.12: Comparison of ResNet vs Plain network. [7]

3.9.2 Skip Connection

The idea of skip connection was not the original, and had been used in "highway networks". However, ResNet was able to establish the skip connection without the addition of new parameters compared to the gated skip connection used in "highway networks" that added the extra parameters.

Skip connection tries to maintain the identity mapping between the skipped blocks. This does not degrade any performance since the stacking of layers still maintain the identity mapping and results to same performance.

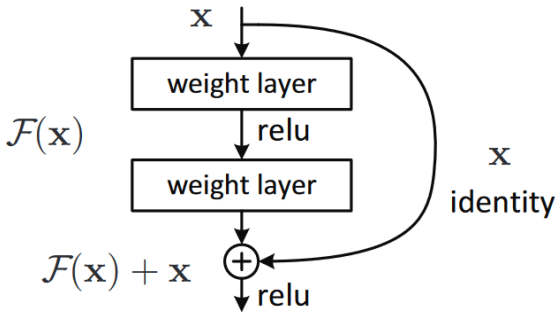


Figure 3.13: Skip Connection in ResNet [7]

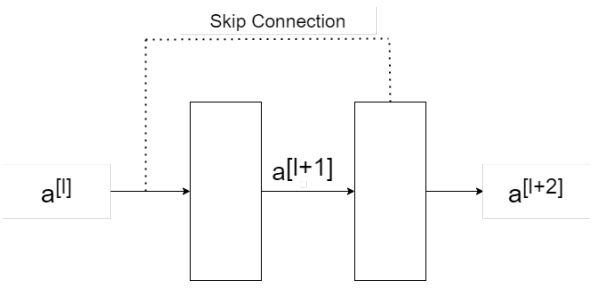


Figure 3.14: Identity Mapping in skip connection

The figure 3.13 shows identity mapping by skip connection that formulates the F(x) + x relation. The figure 3.14 shows more detailed elaboration of skip connection relationship. Let us assume that, we have $a^{[l]}$ as intermediate output in some hidden layers. Then from figure 3.14, if weights are negligibly small, w=0, and bias is considered as zero, we can easily establish the relation as:

$$\begin{aligned} a^{[l+2]} &= ReLU(z^{[l+2]} + a^{[l]}) \\ &= ReLU(W^{[l+2]}a^{[l+2]} + a^{[l]}) \\ &= ReLU(a^{[l]}) \end{aligned} \tag{3.31}$$

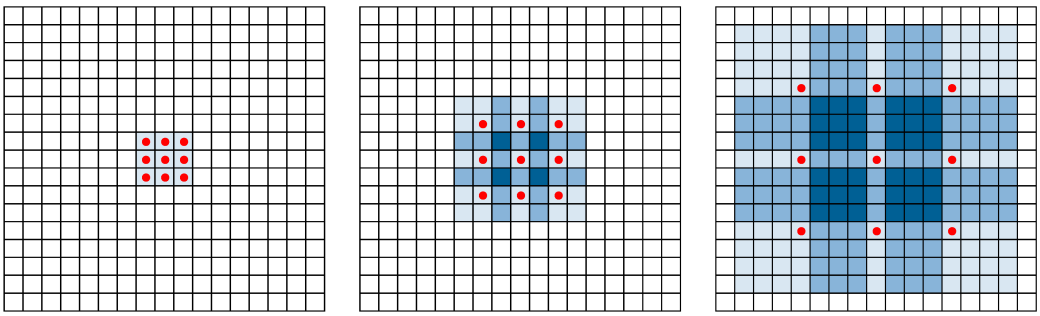
The equation 3.31 represents the identity mapping.

3.9.3 ResNet as Backbone

Because of the ability of the ResNet to allow to train to very deep layers, ResNet has been widely used as the backbone in many architecture. ResNet as backbone has been used as feature extractor.

3.10 Dilated Convolution

The idea of dilated convolution is to increase the receptive field at same parameter counts by introducing the dilated kernel. Dilated kernels can be seen in figure 3.15, where dilation factor of 1 is equal to regular convolution.



(a) 1-dilated convolution **(b)** 2-dilated convolution **(c)** 4-dilated convolution

Figure 3.15: Dilated Convolution with increasing dilation factor [7]

Mathematically, let $F : Z^2 \to R$ be a discrete function. Let $\Omega_r = [-r, r]^2 \cap Z^2$ and let $k : \Omega_r \to R$ be a discrete filter of size $(2r+1)^2$ The discrete convolution operator * can be defined as:

$$(F * k)(p) = \sum_{s+t=p} F(s)k(t)$$

If l is the dilation factor, dilation convolution given as $*_l$ is given as:

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t)$$

In figure 3.15, the original 3×3 kernel expands up to 15×15. The number of parameters is same as $3 \times 3 = 9$, the receptive field has increased tremendously. Hence, with dilated convolution, the receptive fields can be increased exponentially.

3.11 Instance Segmentation Model

3.11.1 MaskRCNN

MaskRCNN (MRCNN) MaskRCNN is instance segmentation model that produces state of art instance segmentation output built on top of faster RCNN method proposed. There are three main regions in MRCNN:

1. Backbone + FPN: Backbone is the featre extractor that takes the input image and extracts the feature. Feature extractors are in addition combined with the FPN architecture that adds the scaling invariance to the backbone features. The typical backbone architectures used are Resnet, VGG.

2. Region Proposal Network: RPN is lightweight network that proposes top anchors (region) for existence of objects in image. In fact, it makes the use of anchor boxes to propose the region where the object might lie. RPN operates on the feature map rather the input map to optimize the computation performance.

   RPN proposes the two anchor classes: foreground class and background class which tells if there is object or not respectively. Several foreground anchors may overlap with each other which are refined using bounding box refinement technique

3. ROI classifier and Bounding Box Regressor: ROI classifier helps classify the specific classes to the foreground objects detected and also refines the location and size of bounding box to encapsulate the object. ROI classifier also contains the ROI pooling that handles the variable input size

Instance Segmentation Mask is finally created for each ROI proposed region with the help of convolution network (mask branch). It generates the 28×28 sized mask which is compared with the resized 28×28 ground truth to calculate the loss. However during inferencing, 28×28 mask generated is scaled upto the original size.

3.12 Semantic Segmentation Model

3.12.1 PSPNet

PSPNet stands for Pyramid Parsing Network. It is semantic network that tries to achieve accurate prediction on even open and diverse environment. The network has scored high records of mIOU accuracy 85.4% on PASCAL VOC 2012 and accuracy 80.2% on Cityscapes. It achieves good prediction result even for the most challenging dataset ADE20k. Although being highly accurate model, PSPNet is far away from the real time semantic segmentation.

The ideas of the PSPNet is built around the concept of global context understanding. Traditional FCN based model failed to recognize the context and was far behind the successful semantic prediction on open and diverse dataset. Even with the implementation of CNN based approach, the result was not accurate enough when the objects look similar to each other. The boat was recognized as car due to similar appearance. This would not have happened if system could understand the global context of 'car can not be on water surface'. Hence PSPNet combines the traditional FCN based approach with sub-region based pyramid pooling module to understand the global context.

3.12.2 Internal Architecture of PSPNet

In practice, momentous global scene prior is usually missed (not incorporated) due to the fact that empirical receptive field of the CNN is small especially for high level layers even though it is theoretically proven to be enough according to [22]. To address this issue, sub-regions can be created to extract the sub-region contexts along with the global context information.Thus, PSPNet combines a hierarchical global prior with information from different scales and sub-regions.

There are four sub-regions under that forms the average pooled or max-pooled representation for different locations. The pooling module for PSPNet is a four-level one with bin sizes of 1×1, 2×2, 3×3 and 6×6 respectively. In comparison to average and max pooling, average pooling seems to perform the better.

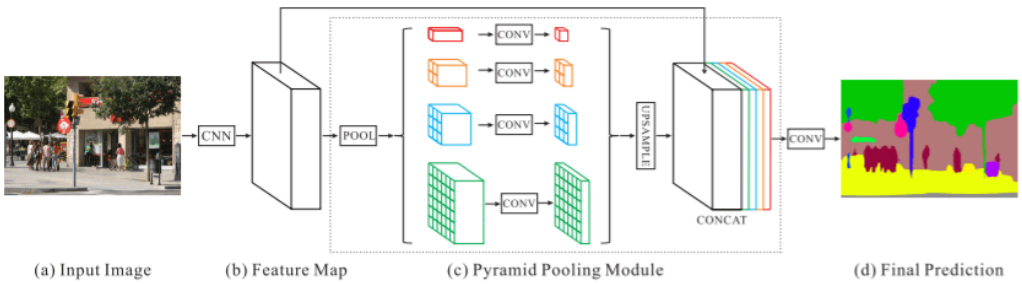


Figure 3.16: PSPNet [21]

As seen in figure 3.16, the PSPNet consists of pyramid pooling module implemented to achieve the concatenated feature map to pass to the convolution layer to generate the final prediction. PSPNet uses the pretrained ResNet model with dilated convolution for feature map.

In the architecture shown in figure 3.16, 1×1 size kernel generate the global pooled context. Followed by 1×1 is 2×2, 3×3 and 6×6 that generate the sub-region contexts. The bilinear interpolation is then used to up-sample the contexts to original size. After up-sampling step, original feature map combines with the sub-regions feature maps. Use of ResNet improves the performance loss due to the deep layer by establishing identity mapping through skip connections. This maintains the computational complexity of PSPNet same as that of FCN based architecture.

3.13 ICNet

ICNet stands for Image Cascade Network which is semantic segmentation model targeted to achieve real time inference of high resolution 1024×2048 size image at real time inference speed of 30fps with considerably high value of mIOU at low computation cost. The system has shown to be very effective across all standard datasets such as COCO-Stuff, CityScapes and CamVid datasets.

The base idea of ICNet is to pass original image to three branches of various scales (resolutions). With the help of low resolution branch, the coarse feature map is extracted which is further refined with the help of medium and high resolution branches.The low resolution branch takes the input image of $H/4 \times W/4$ which gets passed through the series of the convolution layers. This branch down-samples at the rate of 1/8. Finally, low resolution feature map of 1/32th (=1/4 image $Size \times 1/8$ down-sample rate) is obtained. The middle branch or half branch takes $H/2 \times W/2$ sized image which gets passed through three convolution layers each of which down-samples at the rate of 2 such that total down-sample rate for whole convolution layers is 1/8. Final size of feature map is 1/16th of original due to 1/2 image size times the 1/8 down-sample size.The highest branch or unit scaled branch takes original $H \times W$ image that gets passed through very small number of convolution layers compared to lowest branch. The total down-sample rate of 8 is achieved that generates the final feature map of size 1/1 image $Size \times 1/8$ down-sample rate $= H/8 \times W/8$ size.

3.13.1 Internal Architecture of ICNET

The time budget analysis carried out by [20], tells that there is trade off between the speed and accuracy. It even mentions that it is more computationally more expensive to feed the high dimensional image input directly to the system. To mitigate these issues, ICNet takes three separate scaled images and combines the result from branches using cascading unit known as Cascade Feature Fusion (CFF).

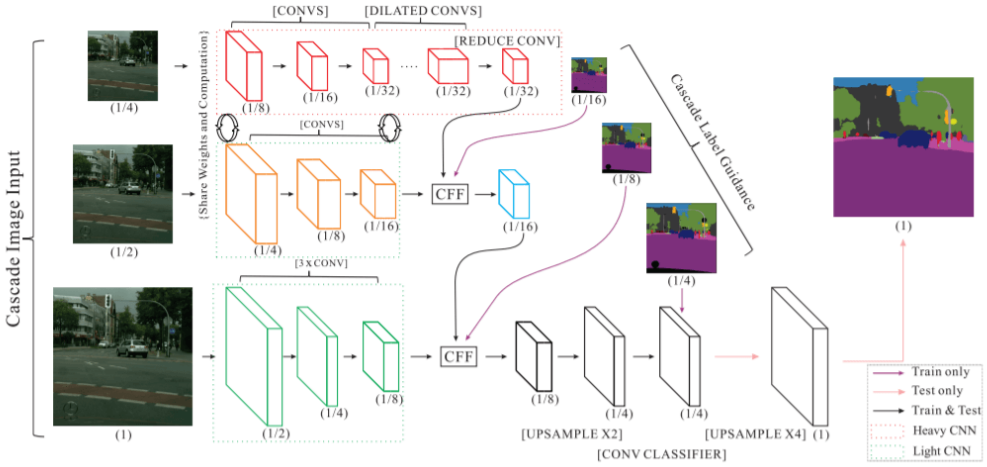


Figure 3.17: ICNET Architecture [20]

The details of the branching scheme are shown in figure 3.17. The topmost, middle and lowermost branches in figure 3.17 are low-resolution,medium-resolution and high-resolution branches respectively. There is trade off between the scale of images and size of convolution layers. More convolution layers are present in the low resolution branch to extract the most of the semantic details. Minor details are extracted from the high resolution branch with only few number of convolution layers. Such minor details are extracted within fast time because of less number of convolution layers.

3.13.2 Lowest Resolution Branch

Lowest Resolution branch is FCN based architecture branch modelled using PSPNet for dilated convolution in ResNet backbone. The dilation factor is set as 2 for all other layers except at stage four and stage five of ResNet where dilation factor is set as 4. As seen in figure 3.17, each convolution layer is consecutively down-sampling at rate of 8. Final output of down-sample and dilated convolution of size $C \times H/32 \times W/32$ is passed to $C' \times 1 \times 1$ convolution filter to reduce the size of output channel appropriate for the CFF.

3.13.3 Medium Resolution Branch

Medium resolution branch takes half size image input. This input is passed along the series of the convolution layers each having the down-sample rate of 2. After being passed through three convolution layers, the output feature map of $H/16 \times W/16$ is obtained. In order to provide the missing details, this branch shares the computation with the lowest resolution branch. Finally with the help of CFF, the 1/16 scaled

output from this branch is combined with the 1/32 scaled output feature map from low resolution branch yielding the resultant feature map of $H/16 \times W/16$.

3.14 High Resolution Branch

In this branch, the number of convolution is limited to less number. Only three convolution layers are present that yields a $H/8 \times W/8$ output. Cascading of this output map with $H/16 \times W/16$ output feature map from medium resolution branch happens at CFF. The output from CFF is then passed to bilinear interpolation based up sampler to achieve the full scaled output feature map.

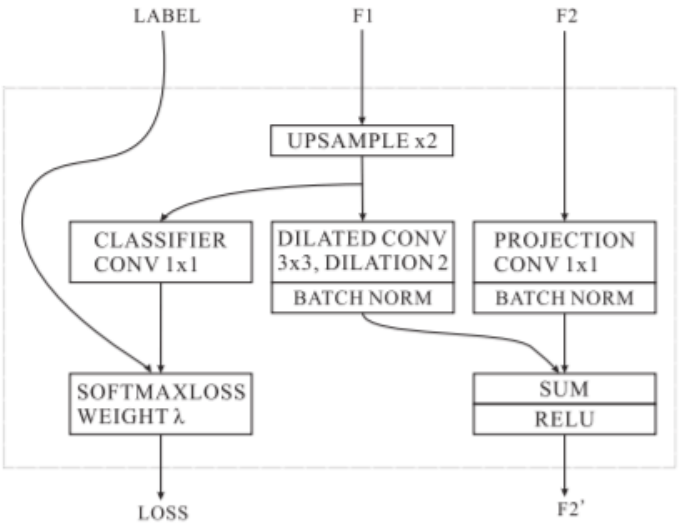


Figure 3.18: Cascade Feature Fusion [20]

As seen in figure 3.18, this unit takes the two features map inputs F1 and F2 of size $C1 \times H1 \times W1$ and $C2 \times H2 \times W2$ respectively where F2 is double the spatial size of the F1. The fusion unit even takes another input of ground truth label of size $1 \times H2 \times W2$ from Cascade Label Guidance. Up-sampling is performed with F1 to make its spatial size equal to F2. Dilated 3×3 convolution is performed on the up-sampled F1. F2, on the other hand is passed to the projection convolution having kernel of size 1×1. The dilated F1 and projected F2 are summed to obtain F2' whose size is $C2 \times H2 \times W2$. The label input is directly fed to the softmax cross entropy unit where the loss is found. This loss represents the semantic accuracy.

In case of ICNet network, for CFF present in medium branch, takes F1 and F2 as $C \times H/32 \times W/32$ output feature map (of low resolution branch) and $H/16 \times W/16$ feature map of medium branch along with the 1/4 scaled ground truth as label.

3.14.1 Cascade Label Guidance

Cascade Label Guidance enhances the learning process. Ground truth label of 1/16, 1/8 and 1/4 are given to respective branch in order to train the ICNET with the help of softmax cross entropy loss given as:

$$L = -\sum \lambda_t \frac{1}{Y_t X_t} \sum_{y=1}^{Y_t} \sum_{x=1}^{X_t} log \frac{e^{F^t \hat{n},y,x}}{\sum_{n=1}^{N} e^{F^t \hat{n},y,x}} \tag{3.32}$$

Motion segmentation in case of static camera can be easy done with methods like background subtraction, but the latter one might need to compensate for the ego motion.

3.15 Differential Drive Model

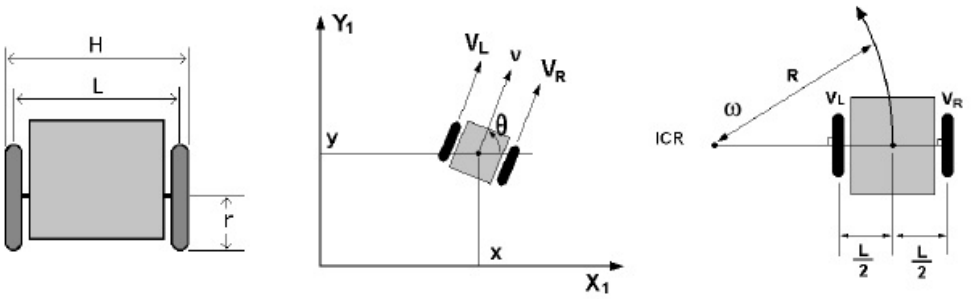


Figure 3.19: Differential Drive Model

Source: `https://image.slidesharecdn.com/mobot-1227974950676397-8/95/introduction-to-robotics-25-728.jpg?cb=1227946763`

Let L be the length of the wheel base i.e. distance between two wheels of radius r. Suppose the required linear velocity and angular velocity of robot be v and w respectively. Then,

$$\begin{aligned} \dot{x} &= vcos\theta, \\ \dot{y} &= vsin\theta, \\ \dot{\theta} &= w \end{aligned} \tag{3.33}$$

where, $\dot{x}$ is x-component and $\dot{y}$ is y-component of velocity and θ is angular displacement. Let $R = v/w$ radius of the path that the robot is moving. and e_l and e_r be the respective arc length covered by left and right wheel at time $\triangle t$ and v_l and v_r be linear velocities

and w_l and w_r be angular velocities of wheel. Then,

$$\theta = \frac{e_l}{(R - \frac{L}{2})} \quad , \quad \theta = \frac{e_r}{(R + \frac{L}{2})}$$

$$e_r - e_l = \theta L$$

$$\theta = \frac{e_r - e_l}{L}$$

$$\dot{\theta} = \frac{v_r - v_l}{L}$$

$$\dot{\theta} = \frac{r(w_r - w_l)}{L} \tag{3.34}$$

Similarly considering circular sector approximation valid for small movements,

$$x = \frac{e_l + e_r}{2} cos\theta \quad , \quad y = \frac{e_l + e_r}{2} sin\theta$$

$$\dot{x} = \frac{r(w_l + w_r)}{2} cos\theta \tag{3.35}$$

$$\dot{y} = \frac{r(w_l + w_r)}{r} sin\theta \tag{3.36}$$

Comparing equation (3.33) ,equation (3.34), equation (3.35) and equation (3.36) we have,

$$v = \frac{r(w_l + w_r)}{2}$$

$$w = \frac{r(w_r - w_l)}{L}$$

Solving above equations we get,

$$w_l = \frac{2v - wL}{2r} \quad , \quad w_r = \frac{2v + wL}{2r} \tag{3.37}$$

These are the angular velocities of the left and right wheel in terms of robot dimensions and the required angular and linear velocities of the robot.

4 METHODOLOGY

4.1 General Setup

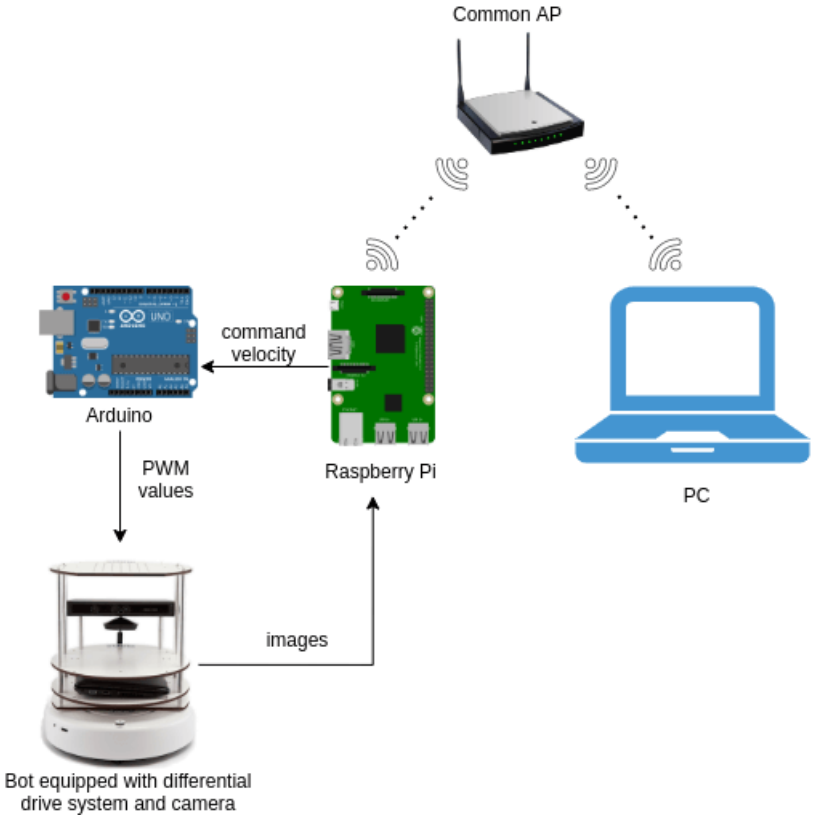


Figure 4.1: General communication outline

Figure 4.1 shows the general hardware setup with the data flows. Here, a differential drive robot equipped with cameras, RaspberryPi and arduino is connected to PC via common access point. The need of connection to a PC is to get good processing power. Using raspberry pi for all the tasks wasn't possible due to its computational limitation. Here, arduino performs all the controlling tasks, raspberry pi basically accumulates all the sensor information and sends it to PC via common access point. It also receives motion commands from PC and sends it to arduino for controlling the bot.

4.2 ROS environment setup

The entire system is built on top of Robot Operation System (ROS). ROS helps to easily communication between different independent processes running in same or different devices. Currently 8 processes are running in parallel communicating to each other via ROS topics. The message flow between various ROS nodes can be seen in figure 4.2

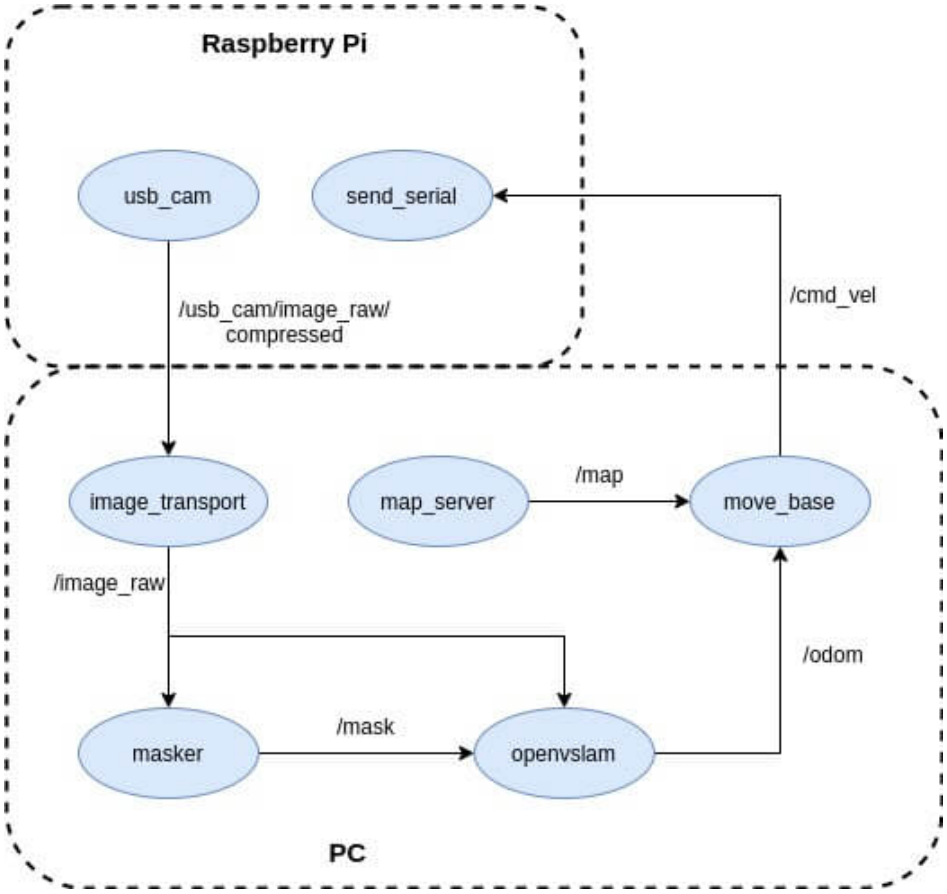


Figure 4.2: ROS setup

Brief description of various ROS nodes:

1. usb_cam
   It reads the image from USB camera connected to raspberry pi and publishes compress image it in ROS environment with topic name */usb_cam/image_raw/compressed*.

2. send_serial
   This node controls the physical robot. It subscribes to the */cmd_vel* topic and sends serial data to arduino, which then controls the motors.

3. image_transport
   It subscribes the compressed image topic from raspberry pi, uncompress it, and then publishes the */image_raw*.

4. masker
   It subscribes the image topic */image_raw*, then computes the mask of the image and publish it in another topic */mask*. The process of obtaining mask from the given rgb image is explained in details below.

5. openvslam
   This is the most important node of the whole system. OpenVSLAM forms

the backbone of SLAM process. OpenVSLAM subscribes to topics */image_raw, /mask* from the two different nodes as shown in block diagram 4.2. Then it uses graph based optimization techniques to obtain odometry of the robot and publishes it in topic */odom*.

6. map_server
   This node serves/publishes the 2D map pf the environment if the ROS topic */map*. The main purpose of 2D map is for navigation in the environment. As, *move_base* package can understand only 2D maps for generating navigation commands, this map is required and enough to achieve 2D navigation goals.

7. move_base
   This is the main node which handles all the navigation and movement of turtlebot. Once it has a good map and accurate localization given by openvslam, it uses A-star algorithm to navigate in the environment and reach the goal. It publishes */cmd_vel* to move the differential drive bot.

4.3 OpenVSLAM

We have used OpenVSLAM Framework in this project which is based upon the Structure from Motion Paradigm. The block diagram showing basic overlay of SFM paradigm is shown in figure figure 4.3. In simple words the Visual features are extracted from the input image which is used to generate the 2D-2D correspondence by matching it with the keypoints of previous frame or reference frame. The 3D points are generated by the process of Triangulation from those correspondence given the pose of camera estimated from the Linear PnP. The generated 3D point cloud is stitched together generating the map hence called Mapping. From the generated 3D map and the keypoints of current frame the 2D-3D correspondence is generated which used to estimate the pose of the camera by Linear PnP thus carrying out the process of Localization. These two process goes hand in hand simultaneoulsy in iterative way or in loop therefore the whole process is termed as Simultaneous Localisation and Mapping(SLAM). And Since we have used camera or Visual features for this purpose it is referred to as Visual SLAM. Since the input of Linear PnP is output of Triangulation and vice versa there is presence of noise hence the optimization is required. OpenVSLAM uses graph optimization for that purpose. The detail explanation of how OpenVSLAM carries out these tasks are explained in following sections.

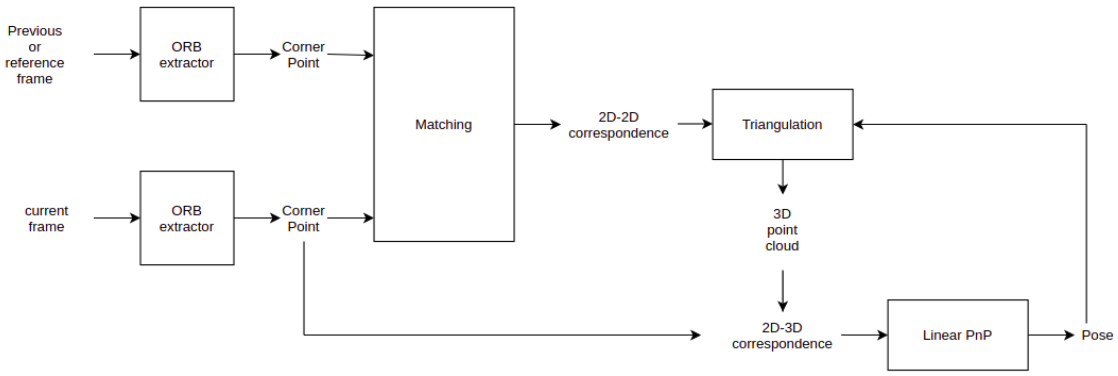


Figure 4.3: BLock Diagram of Structure from Motion Paradigm

OpenVSLAM is a graph based SLAM technique that relies on graph optimization techniques which was described with much detail in theory portion(See section 3.5. It consists of 3 sub modules running parallely:

1. Mapping module
2. Tracking module
3. Global Optimization module

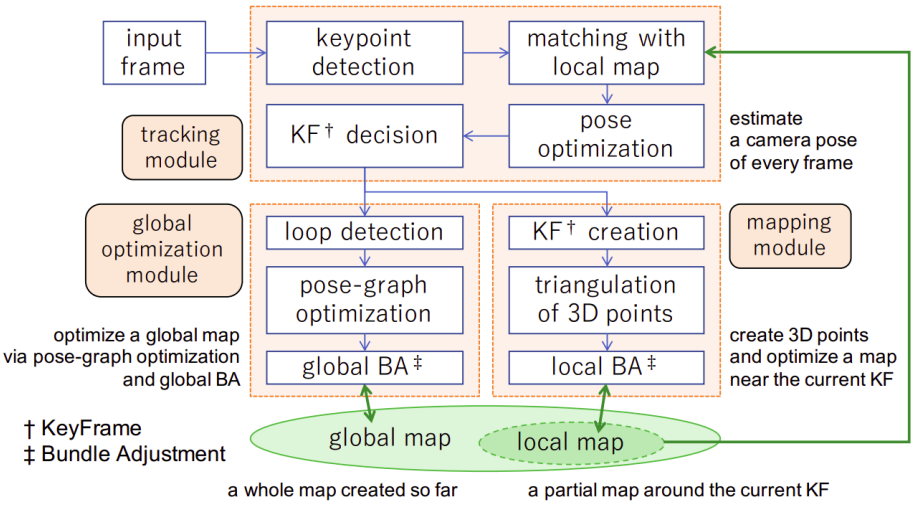


Figure 4.4: Main Modules of OpenVSLAM [17]

4.3.1 Mapping module

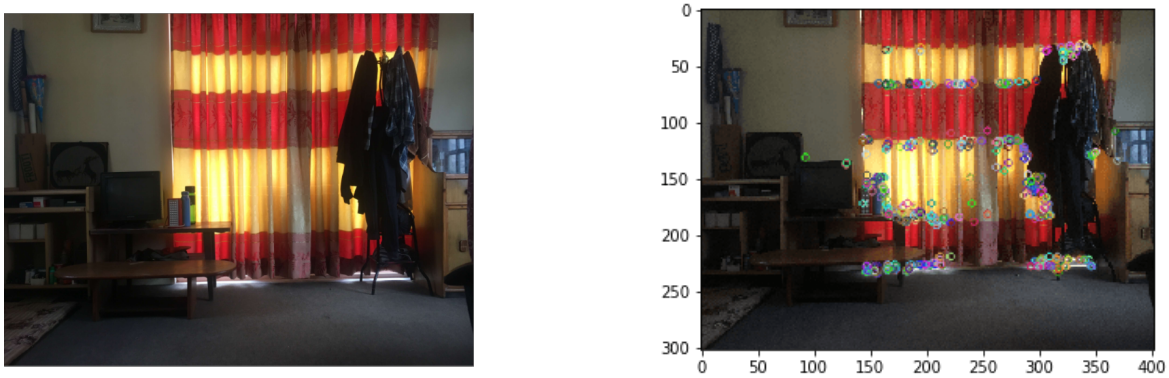
Mapping Module is one of the part of the system. Along with Localisation Mapping must go hand in hand in SLAM. This module is responsible for generating landmarks(3D points) and then selecting the keyframes also known as reference frames on

the basis of the feature points detected. As discussed in section 3.3 given the correspondence and the camera projection matrix the position of the point in 3D space can be estimated by the method known as triangulation(See 3.3.3). This is known as Scene Geometry. In the tracking module the pose of the camera will be estimated using various algorithms. Also the 2D-2D correspondence between two frames are deduced on the basis of the BRIEF descriptor of the ORB features(See section 3.2). Hamming distance is used to estimate the similarity of the descriptor of the keypoints. Thus using the estimated pose and the 2D-2D correspondence the 3D land mark is triangulated thus creating the 3D point cloud which results into occupancy gird map when projected onto 2D plane. The obtained landmarks is simultaneously used by tracking module to estimate the pose of the camera. Thus the process of localizing and mapping occurs simultaneously. The obtained 3D points are in the camera coordinate system. These points are transformed into world coordinates system, in other words the local map is stitched to previous global map. This is done by solving the Procrustes problem(See section 3.4.3). The transformation between the local map and global map is estimated using correspondence between overlapped landmarks and thus global map is formed.

During Initialisation in other words when we have no 3D map at beginning the tracking module is not able to generate the pose of the camera in the absence of the 2D-3D correspondence. Hence all we have is the 2D-2D correspondence at the beginning. Hence the Pose of camera is estimated from it as explained in section 3.4.1. Then this pose is used to generate the 3D points by triangulation. Due to this reason in case of monocular camera the global scale is unknown.

4.3.2 Tracking module

This module of openvslam is the most important part of the algorithm as the estimation of pose and tracking the trajectory of camera is carried out in this portion. The foremost task carried out by this module before starting the process of pose estimation is to read the input frames and extract the orb features points also known as keypoints and its descriptors as explained in section 3.2.



(a) Original image

(b) ORB keypoints detected

Figure 4.5: ORB feature points detected in input image

Then using these descriptors further localisation algorithms are implemented.Three localisation algorithm has been implemented in this section. They are

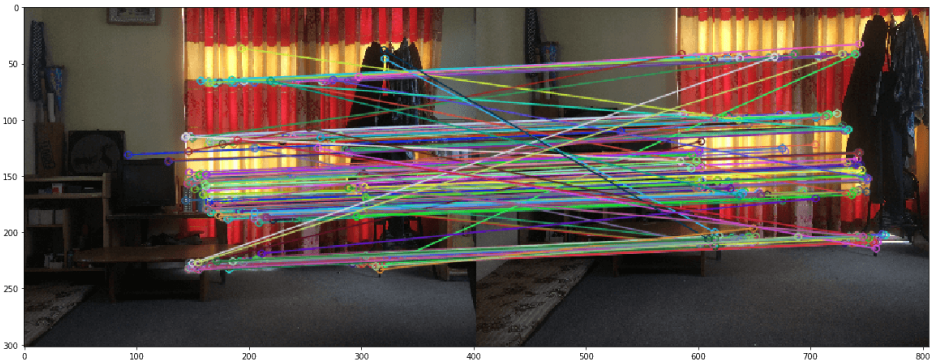
- Motion based Tracking
- BoW based Tracking
- Robust Based Tracking

The above order is the precedence of the algorithm that is used to track the pose of camera. In case the previous method doesn't succeed the algorithm switches to next method. In case all three method fails then the **LOST STATE** is said to be encountered. Then the system need to implement the process of relocalization explained in section 4.3.4.

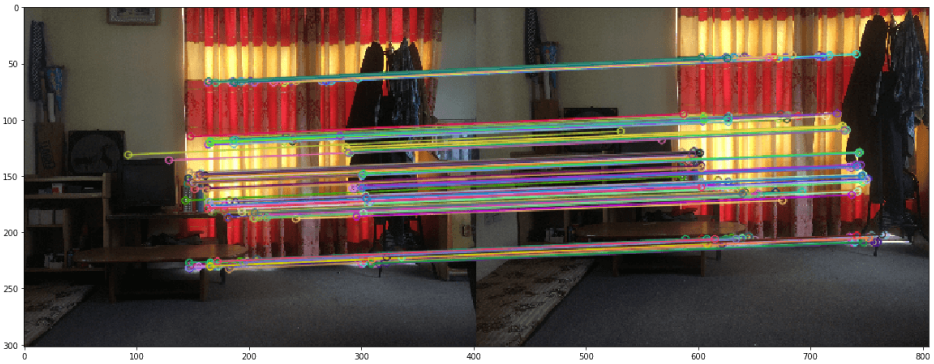
Motion based Tracking
This is the first localization algorithm that is implemented. This method uses the information of the motion model hence termed as motion based tracking. The rough estimate of the camera motion is estimated from the motion model and using that rough pose of camera with respect to previous pose the landmarks(3D points) of keypoints detected in the first frame are back projected on the consecutive frame using equation (3.6). The back projected 2D points are matched with the points in the second frame on the basis of the nearest descriptor which is determined using the Hamming Distance between the BRIEF descriptors. Thus we obtained the 3D-2D correspondence between the a point in 3D world Coordinate system and a 2D image point in the camera coordinate system. Hence the pose of camera can be estimated using the method of Linear PnP as explained in section 3.4.2. In case the number of matches between backprojected points and image points is less than the threshold defined then Motion based Tracking is said to be failed in estimating the pose hence the system shifts to next method explained below.

BoW based Tracking
In case the Motion Based tracking fails the BoW based tracking comes into action. BoW stands for Bag of Words, in this case bag of visual words. It indexes and converts images into bag of word representations. Bag of visual words is the representation of image that describes the occurrence of keypoint descriptor within the image. It looks at the histogram of the kepoints within the image considering each descriptor count as a feature This method implements the hierarchical tree for approximating nearest neighbours in the image feature space using pretrained orb visual vocabulary. Then the input frame is matched with the reference keyframes using this pretrained hierarchical tree. The keypoints from the current frame is matched with that of reference frame, corresponding landmarks are thus associated with the keypoints providing 3D-2D correspondence. Once we have the 3D-2D correspondence pose can be estimated similar to motion based tracking using Linear PnP. In case the number of bow matchs is below the threshold this method is also discarded then the system goes for the last method known as Robust Based Tracking.

Robust based Tracking
In the case that both motion based and BoW based tracking doesn't estimate the pose successfully, Robust based tracking is implemented. In method it uses **Brute-Force** matcher to match the keypoints in the current frame and reference frame on the basis of Hamming Distance between the BRIEF descriptors. It uses ratio test to select good matches. Yet due to noise and various factors lot of outliers are present. Hence it uses epipolar constraint[equation (3.10)] to select the inliers efficiently. The points are normalised using camera matrix K. Then the essential matrix is computed using RANSAC algorithm. For correspondence the points need to satisfy the epipolar constraint containing Essential matrix. The points which doesn't satisfy are considered as outliers. The figure 4.6a shows the result of the use of BF matcher and the ratio test to select good matches. As seen there are lot of outliers which is main reason estimating of false pose. In figure 4.6b, the rejection of outliers with the help of epipolar constraint is shown.



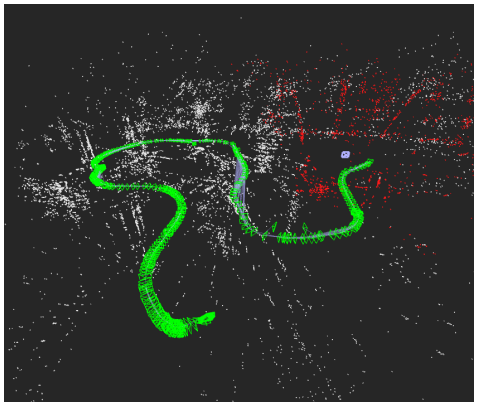
(a) Matches obtained after using BF matcher and ratio test



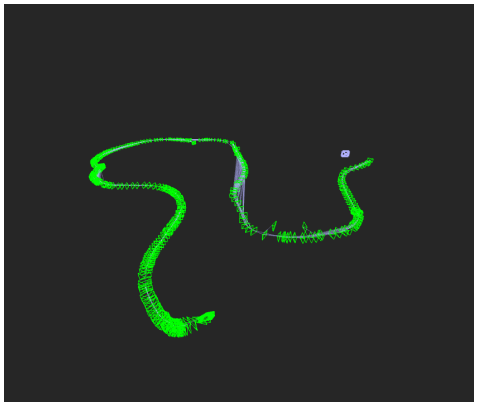
(b) Rejection of outliers using epipolar constraint

Figure 4.6: Effective 2D-2D correspondence estimation

From the 2D-2D correspondence between the current frame and the selected reference frame the 3D landmarks of corresponding points in the reference frame can be associated with the 2D points in current frame. Then the pose is estimated using Linear PnP. In case the number of correspondence after outlier rejection is less than threshold then all three method failed to carry out the localisation and system enters into the Lost State. Then the flow is transferred to the relocalisation section to relocalise the position of camera.



(a) Trajectory along with the point cloud map generated



(b) The trajectory of the camera only

Figure 4.7: Tracking of the camera

Figure 4.7 shows the trajectory of the camera tracked by the tracking algorithm with and without 3D landmarks. This trajectory is obtained by implementing above mentioned methods in the fr2_pioneer_slam3 dataset which is standard TUM dataset.

4.3.3 Global Optimization module

This module does two main tasks: detecting the loop closure and applying global graph optimization once loop is detected.

Loop detection
Loop here means, visiting the same part of map again which was already visited and mapped before. Loop closure helps to reduce the error due to drifting of odometry data. Loop candidates are searched by inquiring to the BoW database. Before inquiring, the minimum score of BoW similarity between the current and each of the covisibilities is computed. And the candidates having score less than minimum scores are selected as loop candidates. If loop candidates are not found, loop correction cannot be done.

Global graph optimization
Global graph optimization is a form of graph optimization technique explained in 3.5. During loop closure, a new edge is introduced into the graph which connect the two nodes of entirely different time. So, graph optimization is applied globally in each and every nodes to minimize the overall error.

4.3.4 Re-localization algorithm

Relocalization algorithm is a part of tracking module of OpenVSLAM. It is one of the most important part of whole algorithm as it helps to find the absolute localization at the beginning of localization process and also if the robot get lost in between.

Relocalization algorithm basically incorporates most the functionality used in three localization algorithms explained above. Figure 4.8 shows the complete relocalization algorithm. The flowchart is self-explanatory. It first finds the list of most probable candidates i.e it chooses some keyframes from its database of keyframes which is most close to the current frame, then it tries to find which of them is the closest one using various metrics and threshold values as shown clearly in flowchart in figure 4.8

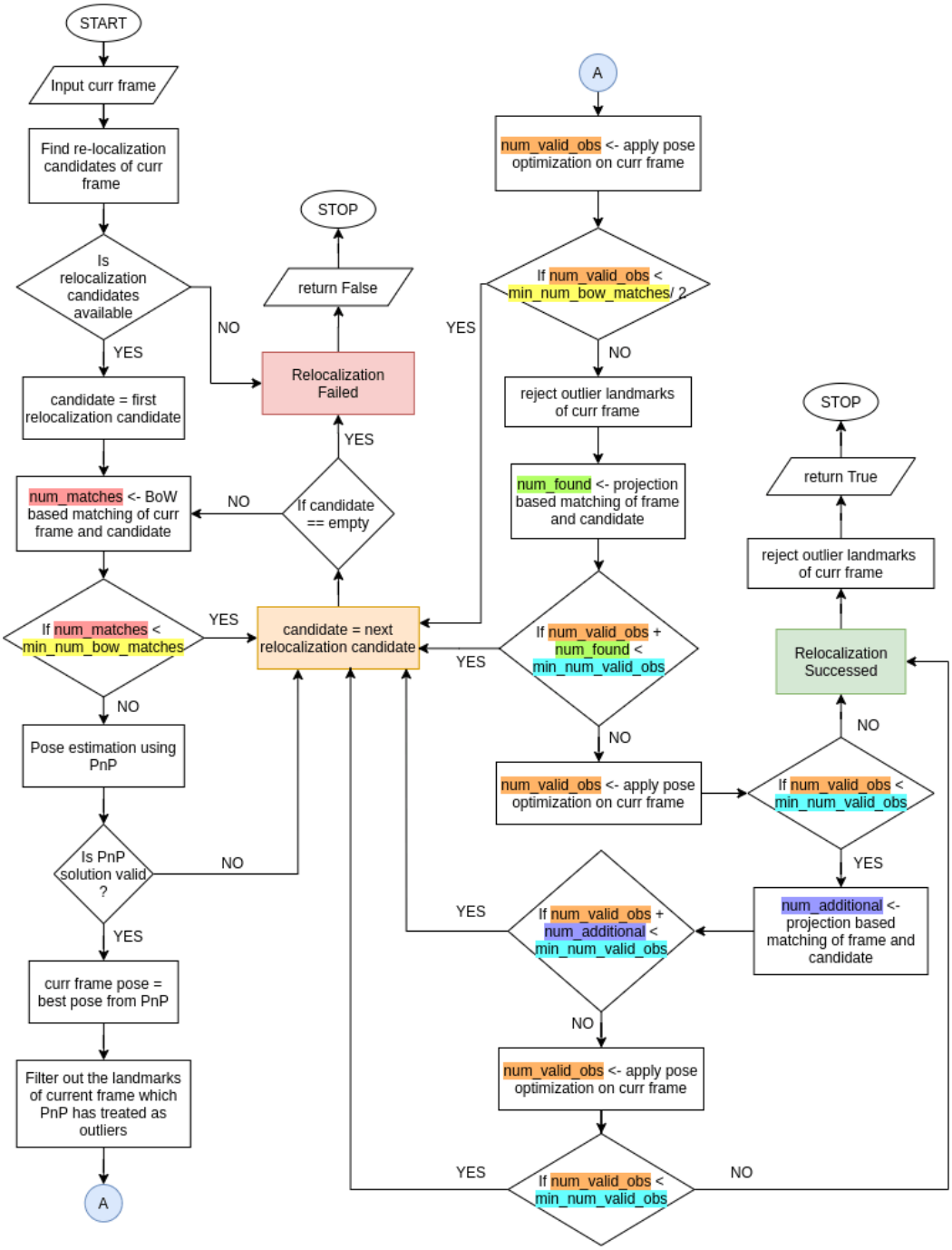


Figure 4.8: Relocalization Algorithm

4.4 Navigation

4.4.1 ROS Navigation Stack

ROS navigation stack is one of the major package that handles all the navigation tasks from mapping the environment to path planning. Here, we have used a part of navigation stack i.e *move_base* package. This package was primarily developed to deal with global as well as local path planning tasks. But, currently we are more concerned with the global path planning than local one. So, for now, local path planning have been ignored.

move_base needs data in well defined form as shown in block diagram in figure 4.9. It expects the ROS topics */odom, /map* and ROS transform messages from *map* to *base_footprint*. The */odom* topic which was primarily supplied by traditional odometry sources such as IMU and wheel encoders are replaced by visual odometry supplied by OpenVSLAM. The transform message from *map* to *base_footprint* can together be generated with */odom* topic.

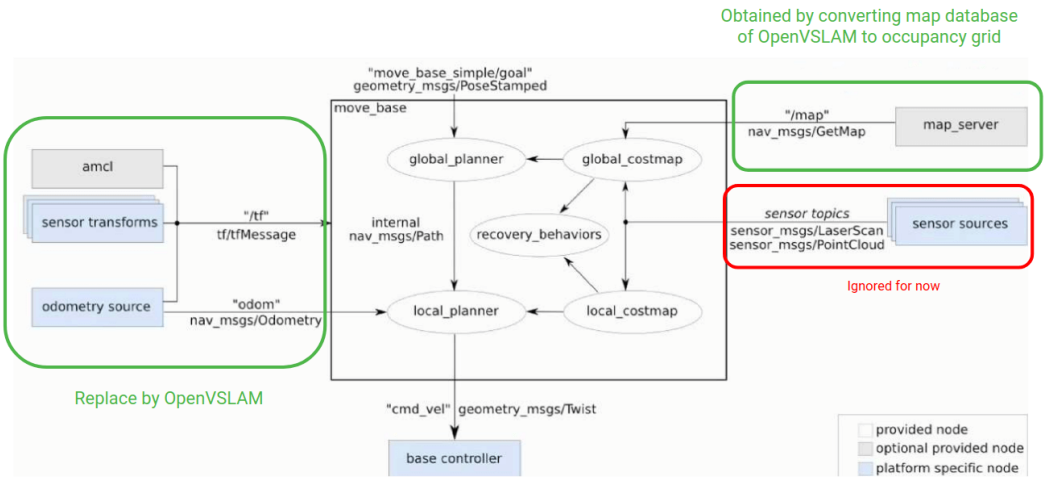


Figure 4.9: Modification in navigation stack

4.4.2 Occupancy grid map

The sparse map created by openvslam (using monocular camera) was projected to 2D plane. Using photo editing software, the map was scaled to actual dimension using dimension of the room.

As the map made using monocular camera lack the actual scale of the room, so these maps need to be scaled to the size of actual room. This can be done by measuring the actual dimension of the room and scaling up the map accordingly.
Here, Pixel distance = $\sqrt{280^2 + 380^2} = 472.016949$
Actual measurement of room = 3.35 meter (11 feets)

Map resolution = 3.35/472.016949 = 0.0070972 meters/pixel
Map origin = (−500, −500) pixels = 0.0070972∗(−500, −500) m = (−3.548601, −3.548601) m

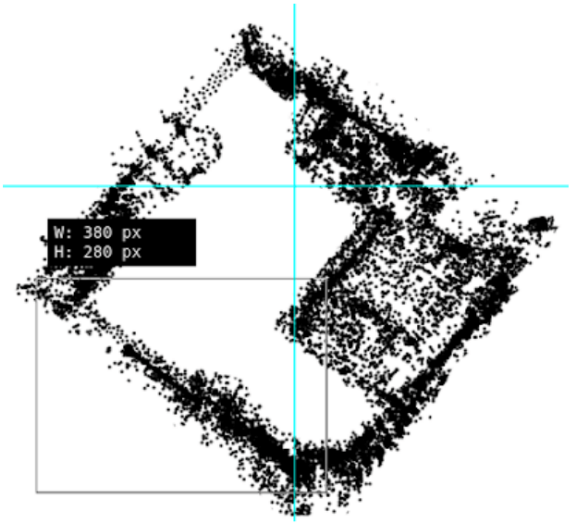


Figure 4.10: Map scaling

Now, these maps are just suitable to work with ROS navigation stack and later used for path planning purpose.

4.4.3 Scaling Odometry

The odometry information published by ROS node *openvslam* is completely based on its pre-built 3D map of the room. But, as we know map built using monocular camera lack scale information, hence odometry information also need to be scaled up to match the scale version of occupancy grid map.

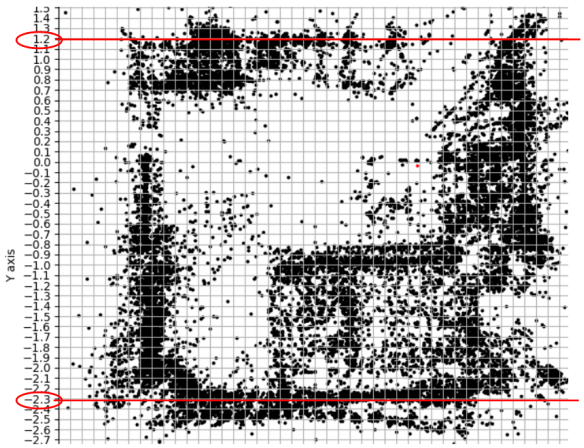


Figure 4.11: Scaling odometry information

Figure 4.11 shows the scale of a portion of 3D map built by openvslam. Here, a known distance is measured in map scale and find its ratio to the actual dimension of the room.
Readings in map = 0.10-(-0.06) = 0.16
Actual measurement of room = 3.35 meter (11 feets)
Scale factor = 3.35/0.16 = 21 (approx)

Now, every odometery topic generated by openvslam is scaled up by the factor of 21 to match it perfectly with the occupancy grid map.

4.5 Dynamic Obstacle Avoidance

Dynamic objects such as human beings, vehicles etc create problem while mapping and tracking the pose of camera. Such objects are temporary on the scene and therefore should not be included while mapping. In case of tracking the pose of camera is estimated using the method of Linear PnP i.e. the correspondence between 3D-2D points as explained in section 3.4.2. Since this method finds the correspondence between two frames on the basis of BRIEF descriptor of the orb features(See section 3.2), the matched feature points detected in consecutive frames will no longer be associated with same 3D point in the space due to the movement of the dynamic object in the 3D space This results in false 2D-3D correspondence thus leads to error in the pose estimation of the camera. In more technical way, the keypoints from the dynamic objects should not be considered. For this purpose, the mask of the dynamic objects needs to be created before extracting the keypoint features.

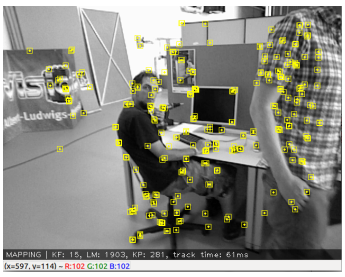


Figure 4.12: Before masking

Figure 4.13: Mask

Figure 4.14: After masking

The figures 4.12, 4.13, 4.14 demonstrate the use of masking for keypoints extraction. The figure 4.12 contains the keypoints features from whole image even from dynamic objects, such as human beings. Such points need to be excluded. For removing keypoints from the scene, at first the mask of dynamic object is created as shown in figure 4.13. The mask in figure 4.13 is blackened for dynamic objects and whitened for other scenes. Finally, with the help of mask, the keypoints of only static objects are obtained as shown in figure 4.14 and only those points used in further computations.

4.6 Mask Generation Using ICNet

For generation of human segmentation mask, ICNet was taken as final choice for the masking. For finalizing the mask, speed vs mIOU trade off between different models were taken into consideration. The repository "https://github.com/thuyngch/Human-Segmentation-PyTorch", gives the pretrained human segmentation models.

4.6.1 Model Comparison

Various pretrained models given in the repository were inferenced with the Automatic Portrait Segmentation for Image Stylization (APSIS) dataset (total of 1531 images) to see the speed and mIOU trade off. As seen from the figures 5.3, it can be seen that even the presence of mask improves the tracking error. However TUM dataset does not contain the ground truth label of semantic segmentation, hence actual mIOU can not be computed. In order to account this, APSIS dataset containing the segmentation label are inferenced using all the available models provided in the repository he so as to have detail mIOU and speed comparison of models. (whose result can be seen in second column of table 5.3). From there it is clear that ICNet provides the best inference speed for our work on CPU based system.

4.6.2 Custom Dataset Generation

After ICNet was fixed as segmentation model, custom dataset was generated in order to improve the segmentation capacity. Our custom dataset is based on walking of person (focused on leg of human being since it mainly falls on the field of vision of robot). The video was taken for person(or persons) walking on multi environment at various lighting condition. For the validation set, inner building of Locus Office has been taken. The necessary video was converted to the sequences of images. In total, multi environment walking dataset consists of 1435 images divided and validation set consists of 1350 images.

For custom dataset, both labels and raw data are required. As the labels are not available, the required labels were created by instance segmentation of using the pretrained model of R50-FPN given in detectron2 model zoo.
(reference: https://github.com/facebookresearch/detectron2/blob/master/configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml)

The code has been modified to generate the binary mask output from instance segmentation.

The models trained for human segmentation were compared based on their speed and accuracy. ICnet was chosen due to the fast speed on the CPU inference speed. To further improve the accuracy of segmentation, it was fine tuned on our custom multi environment walking dataset.

4.6.3 ICNet Training and Freezing of layers

Multi environment dataset is small due and can overfit the ICnet model, hence the feature extracting backbone was frozen during fine tuning on the dataset. This results in only about 400K trainableparameters from total of 11.4M parameters.

4.7 Mobile Robot

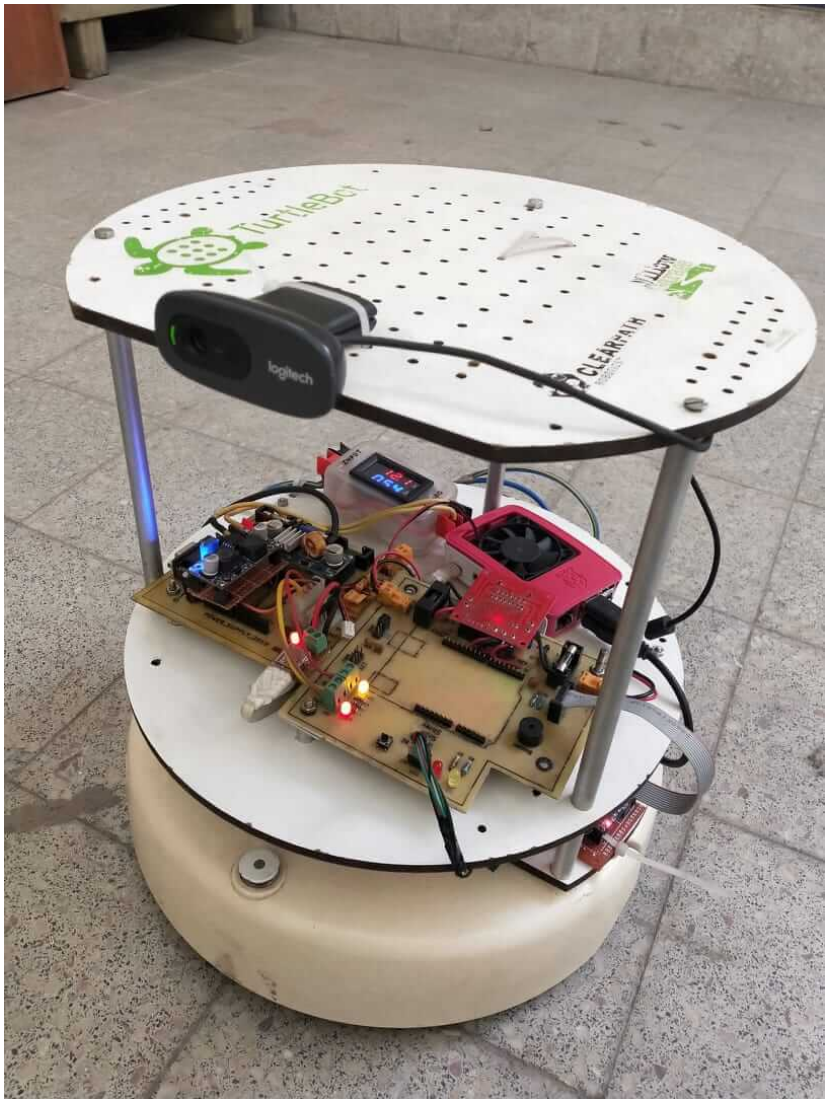


Figure 4.15: Differential Drive Mobile Robot(Model:Turtlebot1)

For the purpose of testing in real environment, we have used the differential drive mobile robot. The microcontroller receives the command velocity from the Raspberry Pi and converts it into the respective wheel velocity in order to achieve the required motion.

We have reconditioned the turtlebot with our own circuits as shown in figure 4.15.

The radius and length of wheel base of the turtlebot was measured and found to be,
L = 26.15cm
r = 3.15cm
Hence from equation (3.37) using the above parameters and the received linear and angular velocity for robot the respective wheel velocities are computed and required motion of robot is obtained.

5 RESULT

5.1 In standard datasets

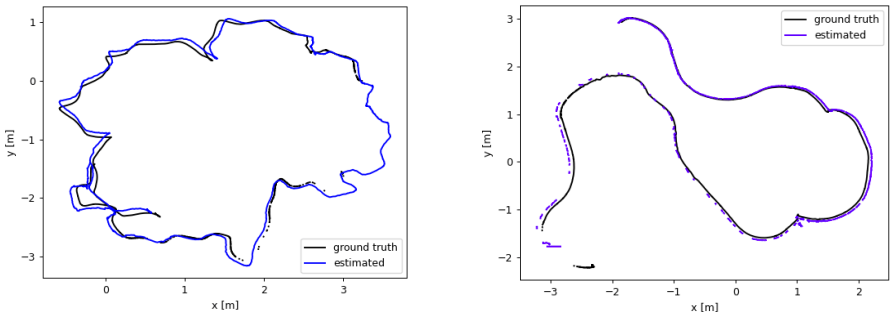
We have tested our method in standard TUM datasets [**TUMdataset**] and analysed its efficiency with certain error metrices and visualising the trajectories and plots. The Standard datasets that we have used can be categorised into Static and Dynamic Environments.

5.1.1 Static Environment datasets

For static environment we have tested on following datasets [16]

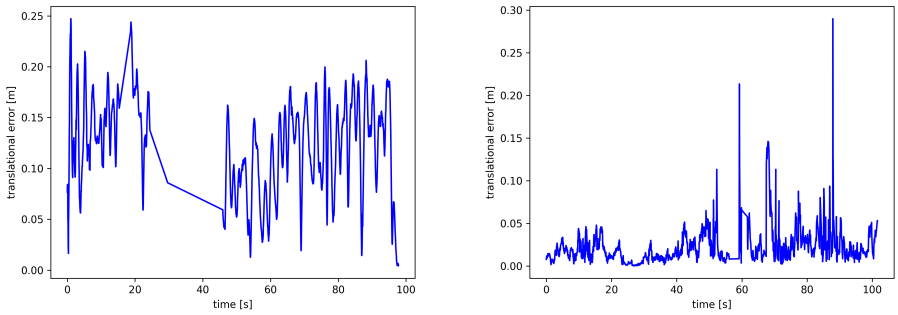
- fr2_desk
- fr2_pioneer_slam3

In fr2_desk dataset camera is moved around the two tables such that the loop is closed. Whereas the fr2_pioneer_slam3 is the dataset recorded from a Kinect mounted on top of a Pioneer robot.



(a) fr2_desk Dataset (b) fr2_pioneer_slam3 Dataset

Figure 5.1: Estimated Trajectory Plot along with ground truth for Static Environment dataset



(a) fr2_desk Dataset (b) fr2_pioneer_slam3 Dataset

Figure 5.2: Relative translational error for Static Environment dataset

Metric	fr2_desk	fr2_slam3
Average RMS error(m)	0.097710	0.101926
Relative Translational error(m)	0.129474	0.028162
Relative Rotational error(deg)	14.376855	1.059033

Table 5.1: Error obtained on the static environment datasets

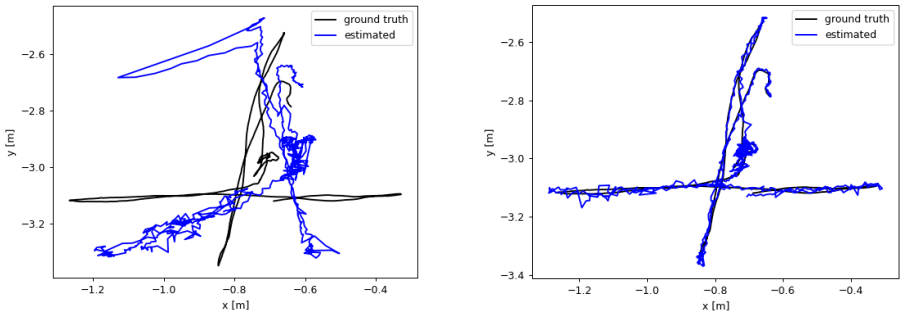
The estimated trajectories of both sattic dataset has been shown in the figure 5.2. The rms error obtained in the case of fr2_desk dataset is **9.7710 cm** whereas that in case of fr2_pioneer_slam3 is found to be **10.1926cm**. The relative translational and rotational error has been tabulated in table 5.1.

5.1.2 Dynamic Environment datasets

In case of Dynamic environments we have tested on two datasets [16]

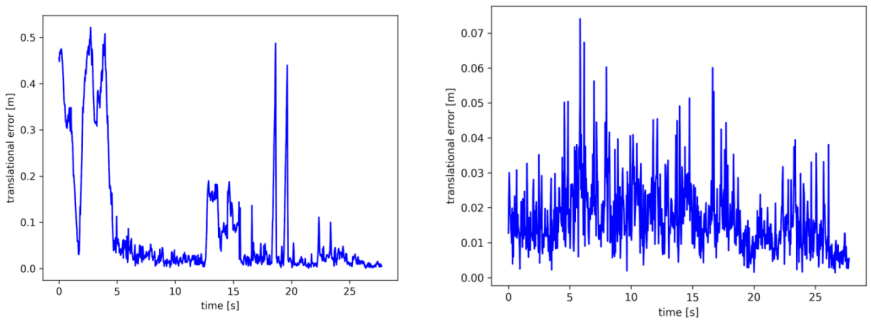
- walking_xyz
- walking_rpy

The *walking_xyz* is the dataset in which camera has been moved along three directions with same orientation whereas *Walking_xyz* is the dataset in which camera has been rotated along principal axis (roll, pitch and yaw) at same position. Hence the effect of translation and rotation both can be analysed using above datasets. These datasets are for the dynamic environment where people are moving. As explained in section 4.5 the visual features keypoints from the dynamic objects are to be ruled out. The results of cases when dynamic objects are not considered and the case when masking is carried out in order to take dynamic objects into account are summarized below in figures 5.3, 5.4, 5.5 and 5.6.



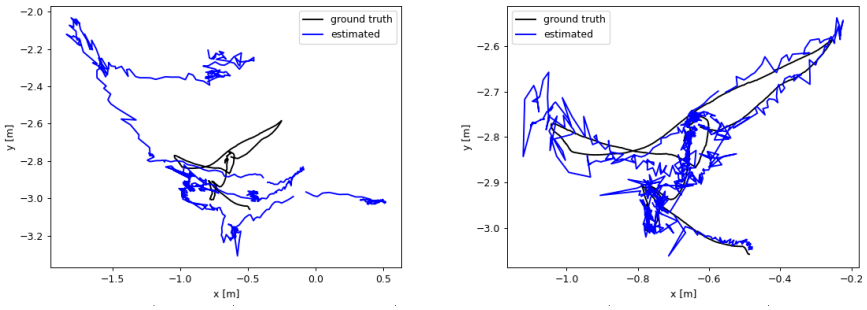
(a) Without Mask (b) With Mask

Figure 5.3: Estimated Trajectory Plot along with ground truth for Walking_xyz dataset



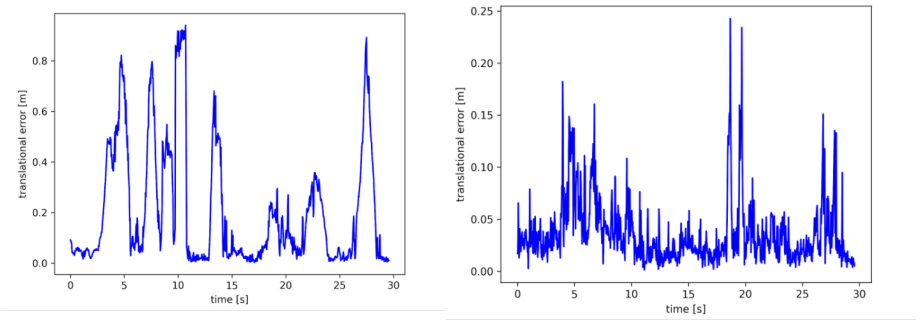
(a) Without Mask (b) With Mask

Figure 5.4: Relative Translational Error Plot for Walking_xyz dataset



(a) Without Mask (b) With Mask

Figure 5.5: Estimated Trajectory Plot along with ground truth for Walking_rpy dataset



(a) Without Mask (b) With Mask

Figure 5.6: Relative Translational Error Plot for Walking_rpy dataset

The effect of using mask on the dynamic object can be clearly seen on the plots of trajectory and relative pose error. For further numerical analysis various metrices has been tabulated below,

Metric	Walking_xyz		Walking_rpy	
	Without Mask	With Mask	Without Mask	With Mask
Average RMS error(m)	0.237222	0.0179716	0.514982	0.039883
Best Case RMS error(m)	0.188568	0.015409	0.470009	0.037272
Relative Translational error(m)	0.1569966	0.022598	0.3059184	0.0511032
Relative Rotational error(deg)	3.0934894	0.6158846	6.0403042	1.1446668

Table 5.2: Error obtained on the dynamic environment datasets

In case of tracking without the use of mask the absolute rms error in the estimated trajectory and the ground truth trajectory were found to be **23.722cm and 51.4982cm** in xyz and rpy datasets respectively. Whereas in case of implementation of mask the error drastically reduced to **1.79716cm and 3.9883cm** respectively as shown in table 5.2. Since the tracking is relative to initial pose and new pose is estimated with respect to that, and previously computed 3D points in the map being generated the performance of algorithm is variable. The above data is the average error obtained from few iteration on the same dataset. The error in the best case scenario has also been listed in the table. Beside that the relative error in the translation and rotation(orientation) of the camera also has been included in the table 5.2.

The above metric is based upon the use of ICNet Mask. The use of other masking techniques such as DeepLab, Unet, CenterMask leads to better accuracy but on the cost of high computational time. Hence the trade off between speed and accuracy has to be made which is further illustrated in section 5.4.

5.2 In real world

5.2.1 Mapping

A complete 3D map of room was created using methodology described in 4.3.1. The mapping was done using monocular camera setup. Results of the mapping is shown in figure 5.7. The map basically stores the 3D landmarks, which are actually the orb features of the keypoints in sequence of frames.

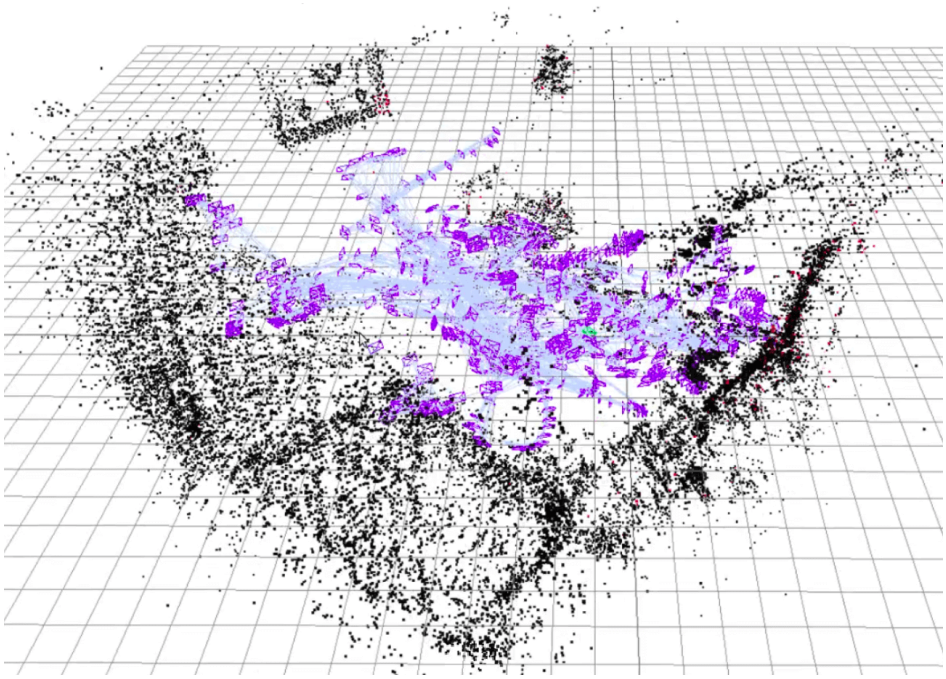
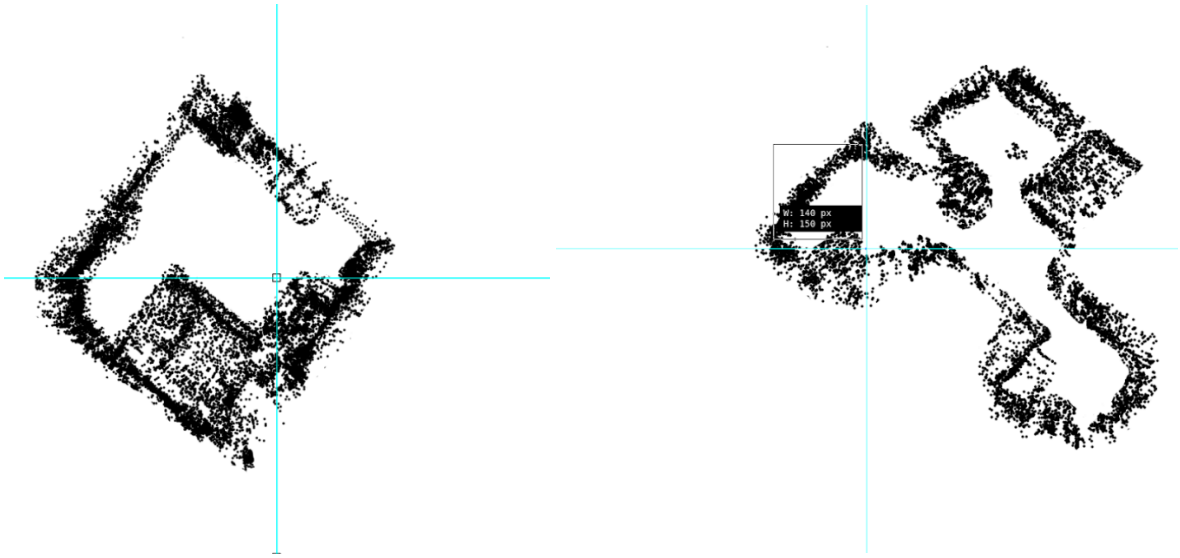


Figure 5.7: 3D map of the room

In figure 5.7, black dots represents the 3D landmarks and purple lines represents the cameras poses during the mapping process. As you can see in the figure 5.7, there are sufficient landmarks(3D points) on two sides of the room and other two sides has very less points. This is due to the fact that, orb feature works well only in textured environment. As, two sides of room had plane walls, less number of 3D points was extracted there.

Figure 5.8 shows the occupancy grid maps of single room and of whole flat. Occupancy grid map was generated as described in 4.4. These occupancy grid map can be directly used with ROS navigation stack to carry out path planning.



(a) Single room (b) Whole flat

Figure 5.8: Occupancy grid maps

5.2.2 Localization

The real life localization was tested in the map formed above. We found localization accuracy has been greatly affected by lightning condition. When localization was done right after mapping the environment, the accuracy was good as there was no significant change in lightening condition. But, when it was tried in different lightening conditions, accuracy was greatly reduced and there was lots of localization failures.

There was no ground truth to quantify the localization accuracy. But we infer it by comparing the motion of camera and localization results given by openvslam. Figure 5.9 shows the odometry information about the camera movement by red arrows. We have moved the camera in similar fashion, so it can be concluded that it is pretty good localization accuracy.

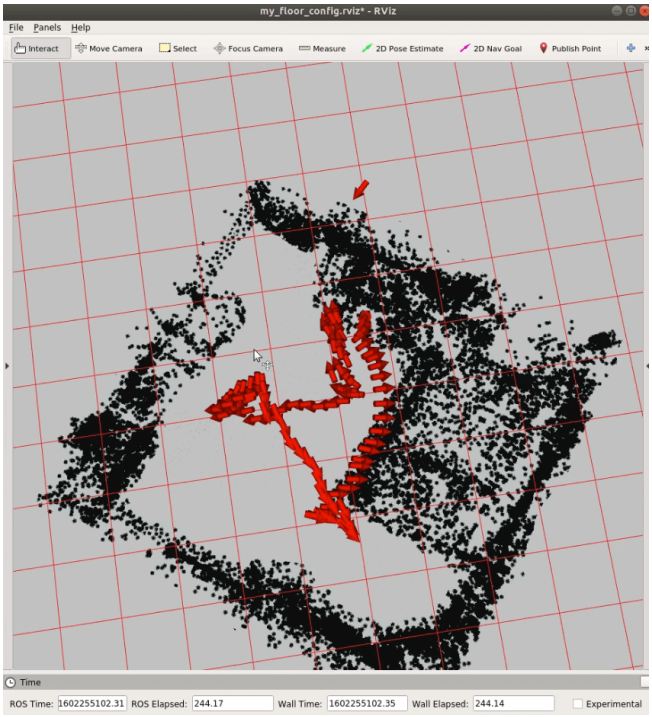


Figure 5.9: Localization result

Figure 5.9 shows the localization results right after mapping the environment. Still, we can see some imperfections in the odometry. The localization accuracy was tested in different lightening conditions. Following are some results of localization testing:

- Localization right after mapping
- Localization in similar lighting condition
- Localization in different lighting condition

5.2.3 Navigation

Path planning was done using ROS navigation stack as explained in 4.4. Figure 5.10 shows the path planning algorithm in action. Here, long red arrow represents the final goal, black line represents planned path, red line represents planned path at the beginning of the algorithm and short red arrows represents odometry information.

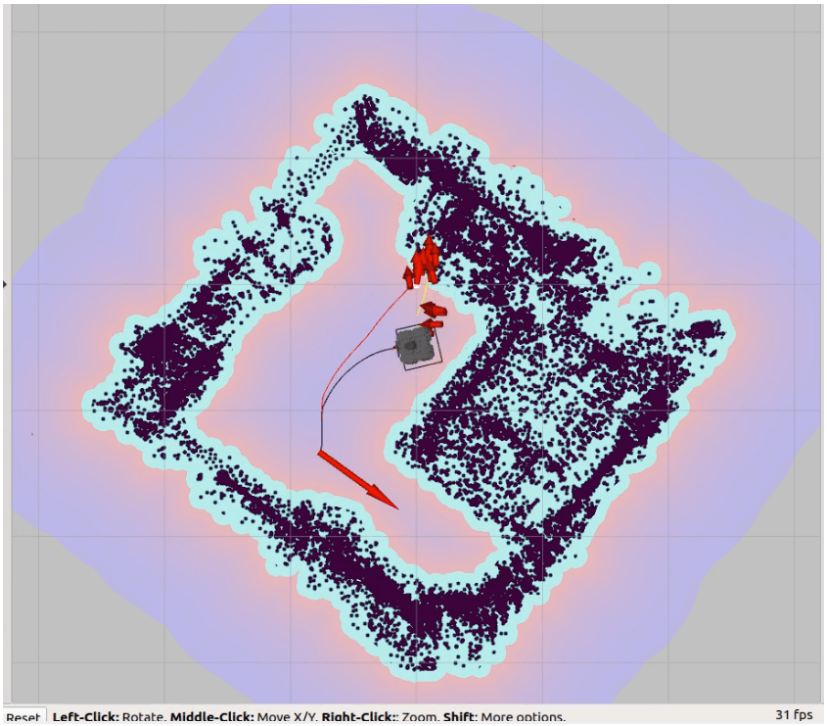
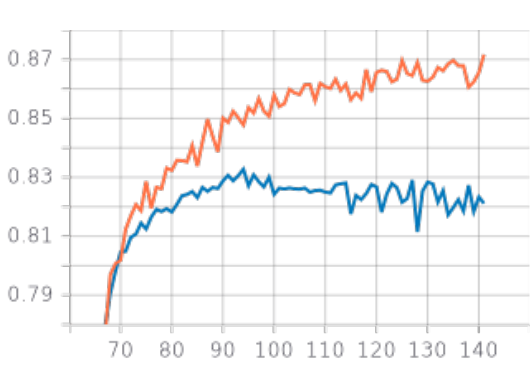


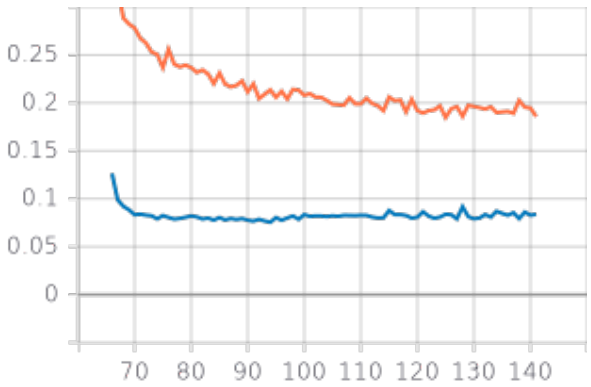
Figure 5.10: Path planning in pre-built map

5.3 Fine-tuning ICNet

The training log of the ICNet can be seen using the tensorboard log in figure 5.11. 94th training epoch model was selected for masking purpose, as it has the lowest loss value and high miou on the validation (locus office dataset). On further training, the miou of training keeps increasing but validation starts to decrease due to over-fitting.



(a) miou vs training epoch



(b) Loss vs training epoch

Figure 5.11: Loss and mIOU progression during fine-tuning

5.4 Comparison Between Masking Techniques

There can be various techniques to generate the masks. According to subsection 3.7.1, panoptic segmentation is slowest one and semantic segmentation is the fastest one. Instance segmented mask are generated via popular MRCNN methods [8], Faster-RCNN methods [14], detectron2 [18]. However these methods are slow and far away from the real time inference.

Semantic segmentation is faster than instance segmentation since, each class is not further instantiated. Semantic Segmentation models such as [20], [3] [15] were tested. The selected choice of the semantic models were based on the fact that these models focused on real time inference rather than the accuracy.

The detail comparison results of masking schemes using given pre-trained models and our fine tuned ICNet model can be seen in table 5.3 and in figures 5.12, 5.14, 5.13 and 5.15. These shows that the masking result for UNet and DeeplabV3Plus are detailed and finer compared to those of ICNet and BiSeNet. The second image shows that gap between two legs are distinct in DeepLabV3Plus and UNet, but are considered as lump mask in ICNet and BiSeNet.

However, the 5.14 shows that, even though it is trained for human, DeepLabV3Plus and UNet try to mask the stationary vase which we do not want to be masked.

Even though the masking result seem obviously better for the other models compared to the ICNet, the speed comparison table 5.3 clearly shows its advantage regarding the speed which is basis for our CPU based system. That is, all inference results were carried out on computer of **Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz CPU. (CPU only, no GPU)**.

The trade off result of speed vs mIOU are summerized in figure 5.16.

(a) Ground Truth

(b) ICNet Masking

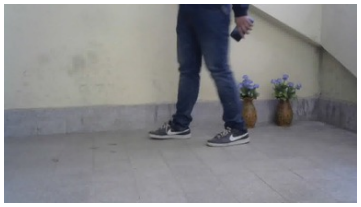
(c) BiSeNet

(d) DeepLabV3Plus Mask

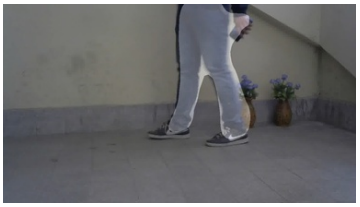
(e) UNet Mask

(f) Epoch 94, Custom Model Mask

Figure 5.12: Masking Results on Multi environment walking dataset



(a) Original Image



(b) ICNet Overlay

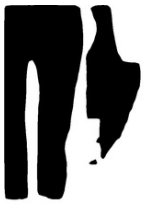
(c) BiSeNet Overlay

(d) DeepLabV3Plus Overlay

(e) UNet Overlay

(f) Epoch 94, Custom Model Overlay

Figure 5.13: Overlay Results on Multi environment walking dataset



(a) Ground Truth

(b) ICNet Masking

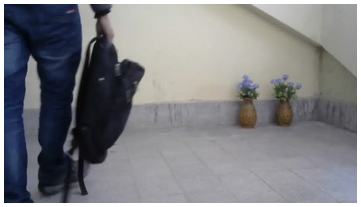
(c) BiSeNet Masking

(d) DeepLabV3Plus Mask

(e) UNet Mask

(f) Epoch 94, Custom Model Mask

Figure 5.14: Masking Results on Multi environment walking dataset



(a) Original Image

(b) ICNet Overlay

(c) BiSeNet Overlay

(d) DeepLabV3Plus Overlay

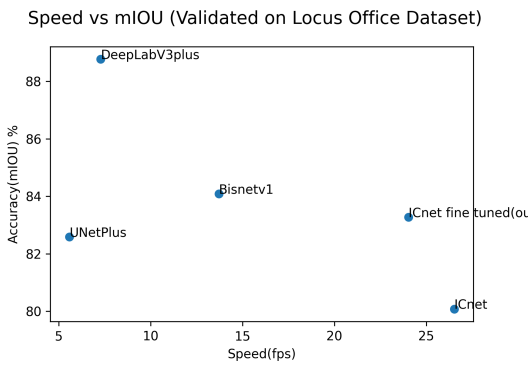
(e) UNet Overlay

(f) Epoch 94, Custom Model Overlay

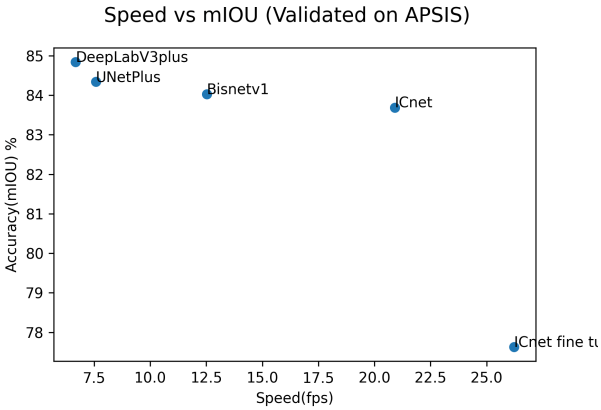
Figure 5.15: Overlay Results on Multi environment walking dataset

Dataset & Methods	Validated on Locus Office Dataset		APSIS	
	mIOU(%)	FPS	mIOU (%)	FPS
ICNet	80.08	26.51525	83.69	20.90771
BiSeNetv1	84.09	13.71467	84.03	12.52348
DeepLabV3plus	88.77	7.28928	84.84	6.67264
UNetPlus	82.59	5.58920	84.34	7.57311
ICnet fine-tuned(ours)	83.27	24.03161	77.63	26.21884

Table 5.3: Inference Speed mIOU Comparison of Segmentation Models



(a) Speed vs mIOU validated on Locus Office Dataset



(b) Speed vs mIOU validated on APSIS Dataset

Figure 5.16: Speed vs Accuracy Comparison of Models

6 FUTURE ENHANCEMENT

There are lots of possible areas which can be enhanced in future. The project is mainly focused on the indoor environment and we have considered mainly human beings as the dynamic objects in case of dynamic environment. The possible future enhancement in the project is to develop system that will be robust for both indoor as well as outdoor environment.

The technique used is affected by the lighting variations. Accuracy is reduced while performing localization in different lighting condition than it was while mapping. The system that is robust and can operate well in any lighting condition can be possible future enhancement of this project. The system works poorly in case it encounters the environment where the feature points are minimal such as texture less plain wall. In such environment the robot gets lost frequently and mapping of such environment is difficult. Hence the system can be improved to work in such texture less environments.

Also, instead of just considering just humans as dynamic objects, motion segmentation technique can be used to segment all the moving objects in case of moving camera.

7 CONCLUSION

There is a trade-off between processing power and price of sensor. Cheaper the price of sensors, larger the processing power needed to achieve the same accuracy. In this project an effort have been made to increase the accuracy of mapping using cheap sensor (i.e camera) and lowest processing power as possible. The accuracy achieved in localization was not revolutionary, and cannot be directly implemented in products. The accuracy might have increased drastically if expensive sensors such as 3D lidars and RADARS was used or high end GPUs were used to increase the processing power. Most of today's state of art system like self driving cars, autonomous drones, food delivery robots by amazon etc. uses lidar technology and fusion of multiple sensors to achieve highest possible accuracy.

But the goal of this project was not to achieve best localization accuracy, rather to develop an best algorithm which can perform well with cheap sensors and low processing power. So, it can be concluded that the project has been able to achieve its objectives.

REFERENCES

[1] Berta Bescos et al. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes". In: 3 (July 2018), pp. 1–1. DOI: 10.1109/LRA.2018.2860039.

[2] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: *arXiv preprint arXiv:2007.11898* (2020).

[3] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *ECCV*. 2018.

[4] Dimo Chotrov et al. "Mixed-Reality Spatial Configuration with a ZED Mini Stereoscopic Camera". In: Nov. 2018.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10605-2.

[6] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521540518.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[8] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV].

[9] Mathieu Labbé and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446. DOI: https://doi.org/10.1002/rob.21831. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21831. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831.

[10] Youngwan Lee and Jongyoul Park. *CenterMask : Real-Time Anchor-Free Instance Segmentation*. 2020. arXiv: 1911.06667 [cs.CV].

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[12] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[13] Adam Paszke et al. *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation*. 2016. arXiv: 1606.02147 [cs.CV].

[14] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[16] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.

[17] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. “OpenVSLAM: A Versatile Visual SLAM Framework”. In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM ’19. Nice, France: ACM, 2019, pp. 2292–2295. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3350539. URL: http://doi.acm.org/10.1145/3343031.3350539.

[18] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[19] Changqian Yu et al. *BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation*. 2018. arXiv: 1808.00897 [cs.CV].

[20] Hengshuang Zhao et al. *ICNet for Real-Time Semantic Segmentation on High-Resolution Images*. 2018. arXiv: 1704.08545 [cs.CV].

[21] Hengshuang Zhao et al. *Pyramid Scene Parsing Network*. 2017. arXiv: 1612.01105 [cs.CV].

[22] Bolei Zhou et al. *Object Detectors Emerge in Deep Scene CNNs*. 2015. arXiv: 1412.6856 [cs.CV].

A APPENDIX: LINEAR ALGEBRA

A.1 Singular Value Decomposition

Singular Value decomposition also called SVD is one of most useful tools in computation of computer vision problems. SVD decomposes a matrix A into three separate components U, D and V^T.

$$A = UDV^T$$

Here,
U and V are column orthogonal matrix
D is the diagonal matrix

The U and V matrix have the property that all the columns are perpendicular to each other i.e. the dot product between any two columns is eqal to 0. And the dot product of a column with itself is equal to 1.

$$\|U_i\| = 1 \quad and \quad U_j^T U_i = 0, fori \neq j$$

The D being the diagonal matrix is a singular value matrix haing diagonal elements as the singular values.

$$D = diag\{\sigma_1, \sigma_2, ..., \sigma_n\} \quad where\ \sigma_1 \geq \sigma_2 \geq .... \geq \sigma n$$

The rank of the A is equal to the number of non zero singular values. The rank of a matrix $m \times n$ must be less than or equal to min(m, n). It can be understood as the dimension of vector space spanned by its rows or columns.
Suppose the rank of A is equal to k, then the first k columns of U, provide an orthonormal basis for the column space of A. Similarly, the first k rows of V^T, provide an orthonormal basis for the row space of A. The most important property that will be used to solve the least square problems is that the last columns of V , or the last rows of V^T, provide an orthonormal basis for the null space of A.

$$null(A) = V_{k:n} \quad \text{where A is matrix of size } m \times n \text{ and k is rank of matrix A}$$

The null space of a matrix is the set of vectors that are mapped to origin by that matrix. The above property can be proved as below,
Suppose vector x is represented as the linear combination of last n-k columns of the V,

$$\vec{x} = \sum_{i=k+1}^{n} w_i \vec{v_i}$$

We need to show that $\vec{x}$ is the null space of A i.e, $A\vec{x} = 0$

$$A\vec{x} = A \sum_{i=k+1}^{n} w_i \vec{v_i}$$

$$or, A\vec{x} = \sum_{i=k+1}^{n} w_i (A\vec{v_i})$$

Here, the term $A\vec{v_i} = (UDV^T)\vec{v_i} = UD(V^T\vec{v_i})$
Since $\vec{v_i}$ is orthognal with all row of V^T except the i^{th} row, the resultant of $(V^T\vec{v_i})$ is the vector consisting all 0 elements except the i^{th} element which is 1. Since the rank of A is equal to k, D has non zero elements uptok^{th} row or column and i ranges from k+1 to n. Hence,

$$D(V^T\vec{v_i}) = 0$$

Therefore,

$$A\vec{v_i} = UD(V^T\vec{v_i}) = 0$$

Thus,

$$A\vec{x} = \sum_{i=k+1}^{n} w_i(A\vec{v_i}) = 0$$

So we conclude that $\vec{x}$ i.e, the last(n-k) columns of V is the null space of matrix A.

A.2 Least Square Problem

The method of least squares is a standard approach in regression analysis in order to approximate the solution of overdetermined systems. The system where no. of equation is greater than no. of unknowns. There are two types of least square problem

$$\min_x ||Ax - b||^2 \quad with\ ||b|| \neq 0$$

and,

$$\min_x ||Ax||^2 \quad s.t.\ ||x|| = 1$$

The first type of least square problem can be understood with the example of line fitting in 2D space. Suppose points(x_i, y_i) in the 2D space. We require to fit the line that best represent those points given by $y = cx + d$. The error for a single point is given by $E_i = |y_i - cx_i - d|$. Hence the total line fitting error is given by the sum of square of those error

$$E = \sum_{i}^{N} (y_i - cx_i - d)^2$$

The goal is to estimate parameter c and d such that the error is minimum. This problem can be rewritten in form of least square problem as below

$$E = \left\| \begin{bmatrix} y_1 \\ . \\ . \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ . & . \\ . & . \\ x_N & 1 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} \right\|^2 = \|b - Ax\|^2$$

Hence given b and A we require to compute x = $[c\ d]^T$.If we have only one point then we have infinite solution. Infinite number of line can be drawn passing through a single point. In case we have exactly two points then the solution is unique as only single line exist that pass through two points and error would be zero. Now if we have multiple

points then we cant have a single line that satisfy Ax =b constraint. Hence we try to find the line that minimizes the error.

$$E = (b - Ax)^T(b - Ax) = b^Tb - 2b^TAx + x^TA^TAx$$

taking derivative of the error w.r.t x and equating to 0

$$\frac{\partial E}{\partial x} = -2b^TA + 2A^TAx = 0$$

$$or, A^TAx = b^TA$$
$$or, x = (A^TA)^{-1}b^TA$$
$$or, x = (A^TA)^{-1}A^Tb$$

This term $(A^TA)^{-1}A^T$ is known as pseudoinverse of matrix A.

If A be matrix of size $m \times n$, general solutions to the least square problem can be summarised as below:

- $rank(A) = r < n$: Then we have infinite number of solutions. The solution is computed by taking inverse of A and all possible linear combination in null space of A is added to it.

  $$x = VD^{-1}U^Tb + \lambda_{r+1}V_{r+1} + ... + \lambda_nV_n$$

  where $A = UDV^T$ and V = $[V_1, ..., V_n]$

- $rank(A) = n$: Then we have exact solution given by

  $$x = A^{-1}b$$

- $n < m$: In case if no of unknown is less than the constraints we don't have exact solution in general so,

  $$\min_x \|Ax - b\|^2 \rightarrow x = (A^TA)^{-1}b$$

Now, for second type of least square problem of type $Ax = 0$ i.e. we minimize the quantity $\|Ax\|$. Let us again consider the line fitting problem but in this case we consider the perpendicular distance between points and line as error instead of vertical distance. This can be done by representing line in homogeneous coordinate as $ex + fy + g = 0$ and points are reprensented by (x,y,1). The dot product between point and the line gives the scaled perpendicular distance. Hence the line fitting error can be represented as,

$$E = (ex_1 - fy_1 - g)^2 + ... + (ex_N - fy_N - g)^2 = \sum_i^N (ex_i - fy_i - g)^2$$

$$or, E = \left\| \begin{bmatrix} x_1 & y_1 & 1 \\ . & . & . \\ . & . & . \\ x_N & y_N & 1 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \end{bmatrix} \right\|^2 = \|Ax\|^2$$

The problem is

$$\min_x \|Ax\|^2 \quad \text{subject to } \|x\| = 1$$

The solution to this problem is given by

$$x = V_3, \quad \text{where SVD of A is given by} A = UDV^T, V = \begin{bmatrix} V1 & V2 & V3 \end{bmatrix}$$

In summary,

- $rank(A) = r < n - 1$: Then we have infinite number of solutions. The solution is any linear combination of vector in the null space of A.

$$x = \lambda_{r+1} V_{r+1} + ... + \lambda_n V_n \quad where \sum_{i=r+1}^{n} \lambda_i^2 = 1$$

- $rank(A) = n - 1$: Then we have one exact solution given by the column in the null space.

$$x = V_n$$

- $n < m$: In this case when no of unknown is less than constraints, we don't have exact solution in general. The vector that minimizes the norm is also the last column of V so,

$$\min_x \|Ax\|^2 \rightarrow x = V_n$$

A.3 RANSAC

Ransac also known as Random Sample Consensus is the method used to reject the outliers. Suppose the third case least square problem of the type $\|Ax = 0\|$ where the no of unknowns is less than the constraints. In such case we find the solution that incorporates all the data points by minimizing the error. But in case of presence of few outliers the solution may result in drastic deviation from the ground truth. The least square problem is very sensitive to the outliers. Hence before fitting the solution over all the data we first remove certain data points(outliers). This is achieved using Ransac algorithm. The strategy is to find the model that accords with maximum number of samples. As the name suggest we first randomly sample certain group from the whole data. The number of data in the group is the minimum number of data required in case of least square problem it is equal to the number of unknown. Using those data we obtain the solution. Then we measure how good the solution is in accordance with other data. On the basis of certain threshold the inliers are selected. We again choose

another group and carry out same process. We iterate for fixed number of times and the solution which incorporates maximum number of inliers is chose to be optimal solution and the data that doesn't satisfy the obtained solution is considered as outliers. Like this RANSAC helps in optimizing the least square problems.

The number of times we need to sample depends upon the no of data, technically the probability of inliers. Suppose Probability of choosing inlier = $\mathbf{w} = \frac{\textbf{no of inliers}}{\textbf{no of samples}}$ then, Probability of building correct model is $\mathbf{w^n}$, where n is no of samples to build a model. Probabilty of not building a correct model during k iterations is $(\mathbf{1} - \mathbf{w^n})^\mathbf{k} = \mathbf{1} - \mathbf{p}$, where p is desired RANSAC success rate. Hence no of iteration $\mathbf{k} = \frac{\textbf{log(1-p)}}{\textbf{log(1-w}^\textbf{n}\textbf{)}}$ This is estimated empirically looking at probability of inlier.

B APPENDIX: PROBABILITY THEORY

B.1 Bayes Theorem

Bayes theorem is based on the fact that new information doesn't create new belief, it just updates the prior belief.

$$PriorBelief + NewInformation = PosteriorBelief$$

Consider an example, In a city, 90% of people are farmers and 10% are teachers. If you choose a person in random, is the person more likely to be farmer or teacher?

Answer to this question is straight forward, there are 9 times more farmers than teacher, so randomly picked person is more likely to be a farmer. Here, our prediction is only based on our prior belief.

Now, we add additional information to our belief. Suppose, 50% of teacher loves to read book and only 10% of farmers love to read book. Now, if a randomly selected person loves to read book, is he more likely to be farmer or teacher?

Answer to this question is a bit tricky. If a person loves to read book then it seems that he/she is more likely to be teacher than farmer. But, we shouldn't ignore the fact that there are 9 times more farmers in the city than teachers.

Lets solve this numerically, Suppose there are 100 people in the city.
So,
No. of farmers $= 0.9 * 100 = 90$
No. of teachers $= 0.1 * 100 = 10$
No. of farmers who read book $= 0.1 * 90 = 9$
No. of teachers who read book $= 0.5 * 10 = 5$
Total no. of people who read book $= 9 + 5 = 14$
So,
Prob. of person being farmer $= 9/14 = 0.64$
Prob. of person being teacher $= 5/14 = 0.36$

Hence, if randomly selected person reads book, he is still more likely to be a farmer.

Here,

$$Prob(X \text{ is farmer given he reads book}) = \frac{Prob(\text{Farmer reads a book}) * Prob(X \text{ is a farmer})}{Prob(X \text{ reads a book})}$$

In more general term,

$$\boxed{P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)}} \tag{B.1}$$

This is called Bayes Theorem.
Where,
$P(X|Y)$ is posterior probability
$P(X)$ is prior probability
$P(Y|X)$ is information