Spine

Acad Year 2017/2018

Project No. (B211)

VISION AND ODOMETRY FUSION FOR INDOOR WAREHOUSE ROBOT LOCALIZATION

cover

VISION AND ODOMETRY LOCALIZATION FOR INDOOR ROBOT NAVIGATION SYSTEM

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

YEAR 2017/2018

Title Page

VISION AND ODOMETRY LOCALIZATION FOR INDOOR ROBOT NAVIGATION SYSTEM

SUBMITTED

BY

TAN YOU LIANG

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

A Final Year Project Report
presented to
Nanyang Technological University
in partial fulfilment of requirements for the
Degree of Bachelor of Engineering (Mechanical Engineering)
Nanyang Technological University

YEAR 2017/2018

TABLE OF CONTENTS

ABSTRACT

Realtime indoor localization is integral to an automated robotic system. Here, we will discuss using low-cost monocular vision as the main pose estimation input to achieve a required precision in indoor localization. Encoder and IMU sensor inputs will be further fused with the vision input in order to obtain a pose estimation reading with greater accuracy and stability.

For vision-based pose estimation, redesigned VITag markers [1] similar to Arcuco markers detection are used for marker detection. Frame and contour filtering, then perspective transformation are performed to identify the position of the marker respective to the camera. Subsequently, basic 2D transformation are used to obtain the position of the camera respective to the world frame.

For IMU and Encoder sensor pose estimation input, sensor odometry will be used. Rotation matrix from the IMU output is used extensively to provide accurate absolute pose estimation heading. Consequent sensor fusion via kalman fusion is integrated to fused 3 sensor readings, then output a final localization output to the localization system.

To get the error deviation of the pose determined by the robot, simple experiments were conducted. Different localization methodology via varied combination among the 3 sensors used (camera, IMU, encoder) were tested to show their respective performances. Testing results will be further discussed and evaluated in the report.

ACKNOWLEDGEMENT

Firstly, I would like to express gratitude towards my parents for their support along my bachelor studies in NTU. I would also like to thank to my supervisor Professor Chen I-Ming and School of Mechanical and Aerospace Engineering, for providing me with the opportunity to work on this robotics related project as my FYP. Also, I would also like to thank Mr Albert Causo for providing guidance and advices throughout this project. Not to mention the staffs in Robot Research Centre which has provided me support on resources and guidance.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background

In the age of automation, from self-driving car to commercial UAV, localization plays an integral part in an automated system. Localization is not a new concept. Ancient sailors gazing the stars for direction seeking, whale echoing in the ocean, etc, these are some natural approaches to localize oneself in the environment. Modern localization in robotics is about understood the relative position and the orientation of a mobile robot/device. In recent years, many research on localization technology, which clearly aligned to the development of the autonomous system is being conducted. In fact, localization system is essential to the navigation and motion planning process of a mobile robot [2].

A variety of algorithmic approaches combined with different sensors configuration were proposed in the past for localization, navigation, and mapping. Meanwhile, throughout the years, development of localization technology has gotten increasingly robust and effective (e.g. SLAM), especially, via sensor fusion of multiple perception pipelines [3] . With the help of sensor fusion, the perception suites are getting more data from the surrounding, consequently eliminating false positive and negative detection.

With regards of the this, localization strategies are largely based on the various criteria and environmental conditions. In fact, most localization problems are within a controlled enclosed environment, which also known as indoor positioning system. This application is widely used in automated warehouse and logistics centers. Big players such as Amazon and Alibaba has devoted significant amount of investments into robotics in order to automate their distribution line [4]. The drive for a cost-saving automated system will further boost global sales in robotics

with a factor of 15 in 5 years time [5]. Thereby, the importance of mastering indoor localization system is vital to scale up the automation process in the industry.

Lidars, ultrasonic sensor, radar, camera, IMU (Inertial Measuring Unit), and GPS are common sensors configuration that is being used to solve localization problems [6]. Conversely, for indoor environment, high precision lidars and beacons (wireless-based positioning system via trilateration) are recurrent devices which enable accurate positioning for an autonomous robot. However, for lidar, extensive feature mapping is required to map the enclosed environment. On the other hand, installation of extra devices (lidars and beacons) is a liability to the already costly mobile robot. The existing low-cost solution which currently used a large number of automated distribution center is by having unique makers tagged on the floor to assist the moving robot in identifying its exact location.

Figure 1: Proposed Mobile Picker robot used in E-commerce Warehouse

Currently, in the lab, an environment with shelves filled with distinct items was constructed. A newly conceptualized robot "picker" will navigate through aisles to find a required item to pick. According to this scenario, high moving flexibility is desired. Moreover, cameras are already installed for detecting the items' visual markers and the items itself via image processing. By leveraging on the real-time imagery input, localization solution via position based on visual

markers is highly effective [7]. Furthermore, encoders were already installed and linked to the robot's wheels. IMU is also relatively cheap and easy to install. Hence, here we proposed a vision and odometry localization to perform the task of localization.

1.2 Objective

The aim of this project is to develop a low cost localization system by using existing hardware of an indoor warehouse environment. A real-time localization system is needed to provide accurate pose estimation result (position and orientation) in a dynamic and generic warehouse environment. An error range within 15 cm is needed in order to have a precise localization performance. Here, standard camera, VITag visual markers, IMU and encoders are used to perform the task.

1.3 Scope

The scope of the project covers:

- To perform VITag visual marker detection
- To perform pose estimation with detected VITag contours as inputs
- To obtain IMU odometry from raw IMU sensor inputs
- To obtain Encoder odometry from a moving device
- To develop a communication framework with integrated localization system
- To perform sensor fusion for camera, IMU and encoder
- To construct a setup for testing and experimentation
- To visualize pose estimation datas
- To carry on testing and experiments on localization performance, between pure vision-based localization and different combinations of sensor fusion

1.4 Proposed Solution

The proposed solution is to create a vision and odometry indoor localization system for the warehouse mobile robot. Here, the odometry is referred to the usage of IMU or dual-axis encoders. The ultimate objective is to obtain a precise and accurate pose information of the robot in real time. While vision-based localization will be the primary localization system, IMU and encoders will be integrated to enhance the pose estimation result.

The equipments used are a standard camera, IMU and encoder sensors. A mobile platform with mounting sensors will be constructed to conduct the experiment in a indoor environment. Info markers will be placed on the shelves, each marker will be placed on the front side of a container on top of the shelf, which it's index represents the corresponded product ID.

Both vision based localization, IMU and encoder odometry will be performed independently. Additional implementation of sensor fusion will be performed, providing further accuracy to the localization system. Scripting language, Python will mainly be used in this process, while complement with C++, XML and YAML. Additionally, open source Robot Operating System (ROS) [8] and OpenCV Library [9] are the main tools being used to achieve the objective. Several visualization tools namely, Rviz [10] and Matplotlib [11] will be used to visualize datas.

Meanwhile, localization performances such as error range and detection range will be evaluated to ensure the computed position is within the desired tolerance (15cm). Further elaboration of this localization method will be explained in the main content.

2. REVIEW OF THEORY AND PREVIOUS WORK

2.1 General Literature Review

About the context of this final year project, a robot localization mechanism is needed to implement on the warehouse mobile robot. As mentioned before, a mobile robot with UR arm has already been constructed to perform detection of marker and motion planning of its robotic arm [1]. A redesigned marker, known as VITag is used and being placed under each item's container. VITag marker provides user the information of the target item's id and its attributes. As said, with the existing developed and installed hardware the robot can further use this visual marker for localization.

Figure 2: Item and VITag markers Shelf Placement

Before diving straight into vision-based localization, we explored some existing indoor localization strategies [12]. In this report, it provides comprehensive insights in bringing up the advancements and features of each localization techniques. Wireless localization methodology using fingerprinting and ToA (time of arrival) has an overall higher accuracy, about 1 meter. This is insufficient according to the $\pm 15cm$ precision that we desired. Furthemore, heavy calibration, time synchronization, and installations of antennas (wireless beacons) are the cost for these wireless solution.

Hence, vision-based localization is the preferred solution. To enhance the pose estimation results, IMU and encoder odometry input will further implemented. In general, we will need to

seperate the localization into four major parts: vision-based localization, IMU odometry, Encoder odometry and sensor fusion.

With the help of VITag markers, vision-based localization was selected as our primary solution. Similar to Arcuco Marker Detection [13] which can be used in pose estimation, the same method can be used for the detection of VITag marker. In general, Arcuco is a square shaped visual marker which commonly uses in robotics testing as landmark marker. The basic idea of frame filtering, marker detection, and bit extraction can be referred to the official OpenCV site. The challenge here for VITag is to quickly identify potential ViTag markers in the frame without creating computationally intensive process. The documentation for Arcuco marker detection has documented clearly the steps to achieve pose estimation.

A research done on vision-based pose estimation for micro drone in an indoor environment was also referred [14].This gave an idea to use onboard hardware to host the pose estimation algorithm. This can be convenient to emulate a mobile robot for performance testing regards the camera localization. On top of that, the micro drone in the research used initial sensor fusion with Kalman filter was implemented to enhance the result, reducing the noise of pure vision based pose estimation. Drifting of an initial measurement sensor can be eliminated with vision absolute measurement updates, removing the accumulation of odometric deviation along the moving process. A similar published paper regards pose estimation has effectively fused both IMU and Vision Data with Extended Kalman Filter (EKF) [15].

wheeled mobile robot is often equipped with embedded encoder. We can leverage on the encoder odometry to further enhance the estimation of pose estimation. Henceforth, for this project, both IMU and encoder odometry were subsequently being implemented on top of the vision pose estimation to enhance the results.

2.2 Concepts on Homography

Camera Calibration [16]

Calibration is needed to remove the distortion and minimize the reprojection error of the mapped points in the 2 perspectives planar frame. To have an accurate reprojection, a precise camera calibration need to make. After all, there are 2 matrices that we are trying to obtain: camera intrinsic parameters, and distortion coefficients. It's also known as calibration matrix, K.

$$\mathbf{K} = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s_\alpha & h_x \\ 0 & f_y & h_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \quad (1)$$

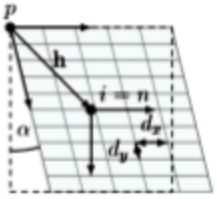


Figure 3: Camera Intrinsic Parameters

The intrinsic parameters above consists of, the focal length, f_x & f_y ; the translational offset of the frame, x_d & y_d . The figure below represents the distortion coefficients that we are trying to obtain in order to remove the reprojection error.

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6)}_{\text{radial distortion}} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

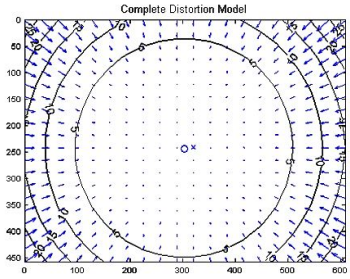


Figure 4: Distortion Coefficients

Homography Transformation [17]

To perform perspective transformation in the for the vision localization, it's important to understand the idea of homography transformation. The Camera Matrix equation shown below indicated the transformation from a point in a 3D space to a 2D space plane.

$$\mathbf{b} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Intrinsic camera parameters — Extrinsic camera parameters

(2)

The Camera Matrix consists of a 3x3 matrix of Intrinsic Parameters, representing the intrinsic attributes of a camera. Nextly, the Extrinsic Parameters is made up of the camera rotation and translation. Note that in this equation, distortion is not incorporated in the camera matrix equation.

As known, the marker (VITag) has a planar profile. Hence, we can assume the target rectangular corner points merely consist of x and y coordinates, while z remains 0 throughout the plane. As such, further simplify the equation to the one below:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \overbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}}^{\text{Homography } \mathbf{H}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

(3)

By combining both the intrinsic and extrinsic parameters matrices, we will be able to obtain a 3x3 homography matrix. Homography matrix represents the transformation of 2D planer points from 2 different viewing perspectives in a 3D space.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{\text{Homography } \mathbf{H}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

(4)

With the equation above, we will be able to find the homography matrix by entering the reprojection of VITag contour corners in the equation (X,Y and X',Y'). Having a homography matrix enable use to perform warp perspective to all image frame.

Nextly, the challenge is to derive the pose estimation of the camera. The pose estimation here is represented by the translation and rotation between the two observed planes, x', y' and x, y. With K as the calibration matrix, if there's no rotation and translation in **equation 2*, we will have a home position - projection equation: [18]

$$x = K \begin{bmatrix} I & | & 0 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = KX \qquad (5)$$

Resemble **equation 5* with an additional Rotation by a matrix R, the projection equation will be:

$$x' = K \begin{bmatrix} R & | & 0 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = KRX \qquad (6)$$

$$By\ compare\ equations\ 5\&6,\ x' = KRK^{-1}x = H\,x$$

$$Homography,\ H = KRK^{-1} \qquad (7)$$

Here in **equation 7*, we could observed the relationship between the homography matrix, calibration matrix and rotation matrix. Euler angles can be further obtained from by decompose the matrix R. By substitute elements of R into equation **equation 2*, we can get the 3 translation values. Ultimately, the 6 DOF pose estimation is calculated.

2.3 Odometry Localization

IMU (Inertial Measuring Unit)

IMU is an electronic device that measures and reports a body's specific acceleration, angular rate, and the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. It is commonly used in pose estimation for dynamically moving object.

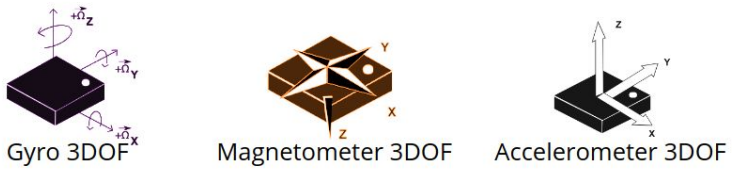


Figure 5: Built-in Sensors within IMU

The Gyro sensor is able to measure angular velocity of each axis from the Coriolis Force that applied to a vibrating element in the sensor. Similarly, accelerometer contains small micro masses that enable the detection to acceleration of 3 axis. For Magnetometer that uses the principle of Hall Effect, voltage differences can be detected when earth magnetic field influences the current within the sensor. [19]

Incremental Encoder

An encoder is an electrical mechanical device that converts linear or rotary displacement into digital or pulse signals. An incremental encoder generates a pulse for each incremental step in it's rotation. Multiple photo detectors within the encoder enables generation of multiple “offsetted” pulses. As such, multi-channel output pulses aid in determine the rotating direction of a encoder.

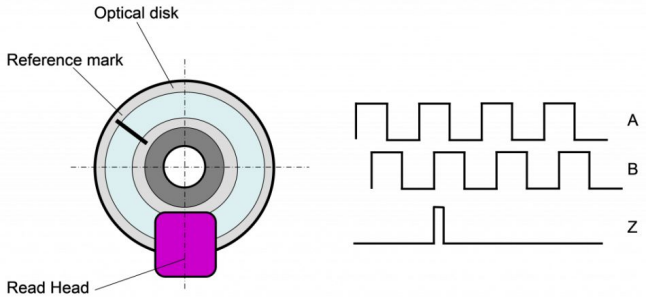


Figure 6: Output Pulse for Channel A, B, Z in an Encoder

IMU and Encoder Odometry

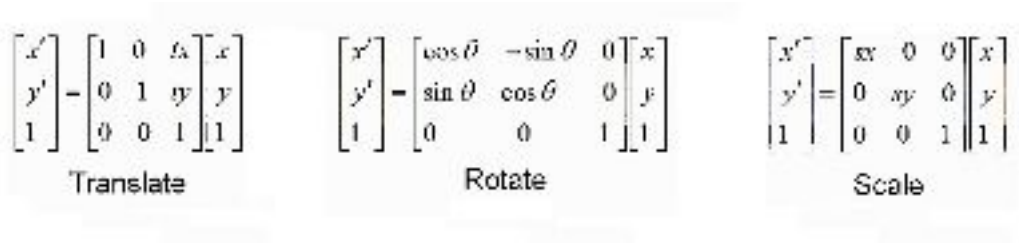
With the mentioned characteristics of IMU and Encoders, we are able to utilize the characteristic into our project. For IMU, displacement can be gained by simple double integration. Also, the absolute output values of the IMU rotation matrix is extremely useful and reliable. On the other

hand, encoder displacement can be derived by measuring the number of pulses. Both IMU and encoder will enable users to track the odometry of a moving device, which respective to its origin. This odometrical values are accumulation of displacement unit, thus drifting error occurs.

2.4 Frame Transformation

2D Transformation

A robot that moves in a 3D space required 6 variables to represent its pose. However, instead looking into 3D transformation, we simplified the problem to a 2D space, a top-down planar view of the environment. Essentially, with only 3 variables (x, y, and yaw) in a 2D-environment, it will be sufficient to depict the camera current pose in the world frame.



Fundamentally, there are multiple reference frames in the overall localization system. Different reference frames will need to be transformed to a final common frame, in this case the world frame. To transform different coordinate respective to various frame, basic 2D transformation is used. As a whole. the user will be able to obtain the localization info which refers to the world frame with 3 variables (x, y, theta).

3D Transformation

Similar to a 2D transformation, 3D transformation has an additional dimension on top of it. 3D Rotational matrix is needed to represent the IMU rotational angle relative to the ground. To implement sensor fusion, understanding of 3D rotation matrix is needed.[20]

Roll α, Pitch β, Yaw γ represent in a 3x3 matrix rotational matrix form. Each rotation matrix is a simple extension of the 2D rotation matrix

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \; R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}. \; R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix}.$$

The yaw, pitch, and roll rotations can be used to place a 3D body in any orientation. A single 3D rotation matrix can be formed by multiplying the yaw, pitch, and roll rotation matrices to obtain:

$$R(\alpha,\beta,\gamma) = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}.$$

2.5 Kalman Filter

Kalman filter [21], [22] is a fusion algorithm for situation when there are uncertain information about some dynamic system, and an educated guess will be computed to guess the state of the system at the next iteration/frame. The application is widely used in tracking, guidance, navigation, and control. For the case here, it's extremely useful to apply Kalman Filter for sensor fusion. In fact, It's common that drones (UAV) fuse noisy GPS reading, which is absolute, and IMU reading together to obtain a much precise localization output.

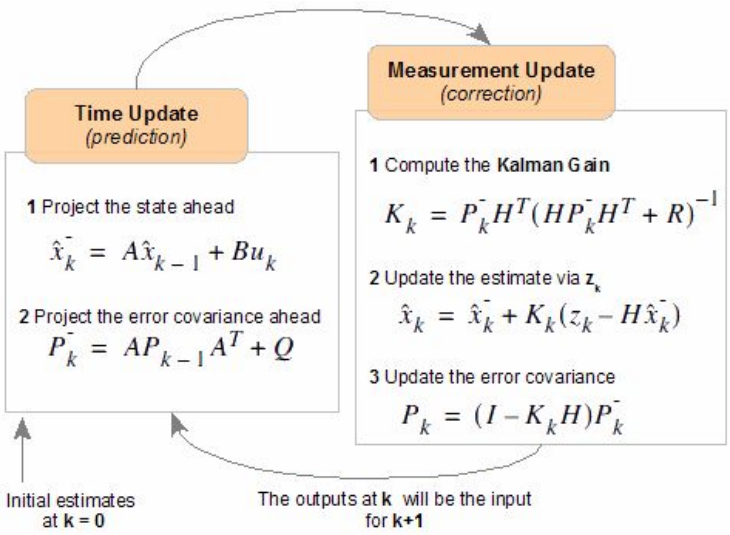


Figure 7: Kalman Filter Process and Equations

According to the figure above, kalman filter can be broken down into two major steps: prediction step (Time Update) and update step (measurement update). Another important concept to understand Kalman filter is the idea of combining two estimated values, which involved 2 Gaussian distribution curves.

For our case here, we can think as prediction state estimate is the previous reading of the IMU input, measurement is essentially the vision pose measurement update. The reason normal distribution curves are used is due to the noisy nature of both sensing input. By combining both estimation, this will enable us to obtain a optimal state estimate, $\hat{x}$ as shown at the figure below. The variances of both predicted state estimate and measurements will be combined to a new variance. Covariance matrices, P will be used to represent these variances. Subsequently, these optimal state estimate, $\hat{x}$ and P will be used for the next "measurement update" step.

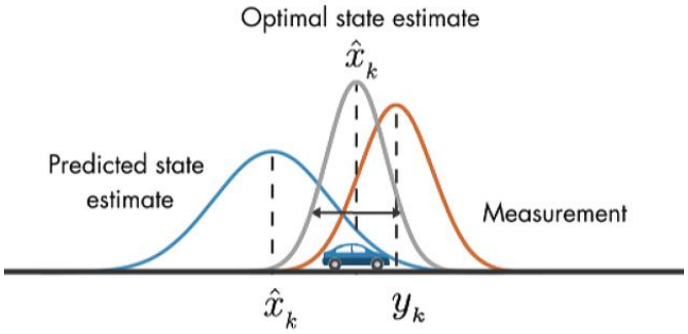


Figure 8: Gaussian Distribution of Measurement, Predict and Optimal State Estimate

Covariance Matrix

To represent a 1D gaussian estimation, a single value variance coefficient is sufficient. However in a 2D gaussian distribution representation, we would need more information the show the probability distribution. Thus a 2x2 matrix, known as covariance matrix, Σ is used to bring a better representation of the state estimation distribution.

$$Covariance\ Matrix,\ \Sigma = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix} = \begin{bmatrix} Var[X_1] & Cov[X_1, X_2] \\ Cov[X_2, X_1] & Var[X_2] \end{bmatrix}$$

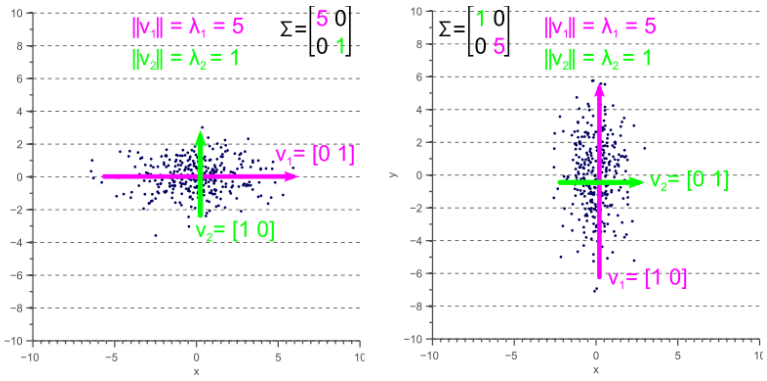


Figure 9: 2D gaussian distribution of covariance matrices

3. IMPLEMENTATION: VISION POSE ESTIMATION

It is known that 3 sensors will eventually be used to create a robust indoor localization system. To accomplish this goal, the project is broken down into several parts to better elaborate on the tasks that were being done. Firstly, vision pose estimation is used as a primary pose estimation toolbase. The objective here is to create an effective vision based localization system via pre-known VITag markers. Steps to be taken are as follows:

3.1 Camera Calibration

A 640x480 LogiTech C170 webcam is used to perform the camera detection. To begin with, in order to achieve a better reprojection of the points in a 2D captured planer frame, it's important to have a good calibration. By using openCV camera calibration chessboard script[23], it eases the process to obtain the camera matrix (3x3 matrix) and distortion coefficients (a vector with 4, 5, or 8 parameters). Multiple images of the chessboard need to be captured from different position and orientation. Eventually, the script will output the intrinsic parameters and distortion coefficients to the user.

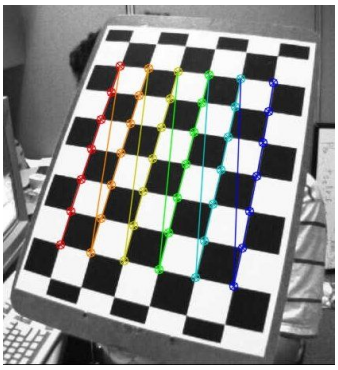


Figure 10: Chessboard used in Camera Calibration

To determine a successful calibration, a low "reprojection error" needs to be obtained, this will be shown at the end of the calibration. The reprojection error shows the discrepancy of the reprojection points to the original image frame, via the newly obtain parameters matrix. Thus, if

the error is high, calibration needs to be redone. Note that different camera even the one with the same model will have different intrinsic and distortion coefficients.

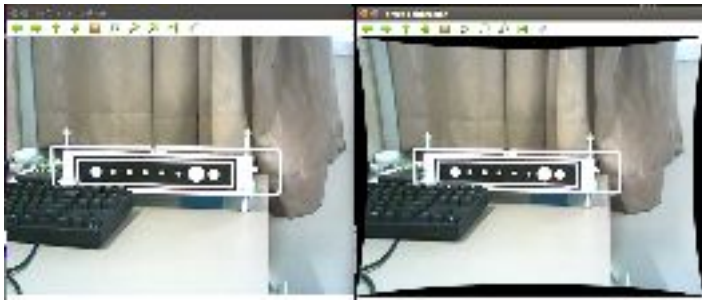


Figure 11: Before and After Calibration of the Imagery

3.2 Detection of ViTag from raw frame

The next operation here is to identify the VITag marker from the environment. Several steps are taken to complete the detection.

1) **Undistorted frame to eliminate distortion**
   Intrinsic Camera matrix and distortion coefficients which were determined from the previous calibration were being used to convert the raw frame to an undistorted frame.

2) **Canny Edge Detection**[24] **for Contour Detection**
   In order to remove unnecessary elements in the raw frame, canny edge detection is used so that only prominent edges remain. This further enhances the performance of contour detection of the frame without detecting undesired contours.

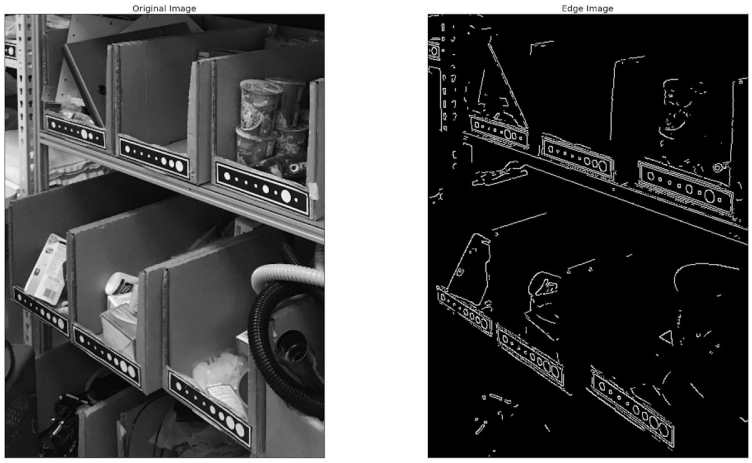


Figure 12: Canny Edge Detection (Right)

3) Contour filtering via hierarchy and sizing

After contour detection, filtering is needed to eliminate unwanted contours which don't match the feature of the VITag marker. Two features are being evaluated here, contour size and hierarchy. For size, a threshold for the minimum contour area is used. Whereas for contour hierarchy (RETR_TREE), we determined contours with no "first child" (h[2] is -1). These are the potential contours of the VITag data circles as they are the lowest level contours which don't contain any contour in within. The parent contour of the VITag data circles is the rectangular border of the marker. With this, parents of these potential data circle contours are being determined as potential VITag outer layer rectangular contour. This process will return only a few of contours, usually less than 10 contours. This significantly improve the detection performance.

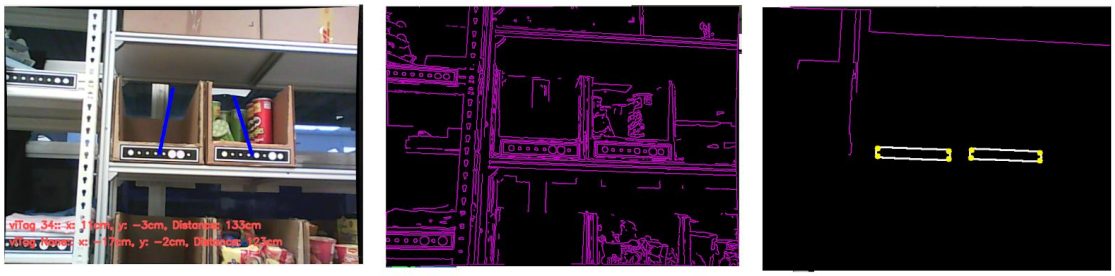


Figure 13: Result of Contour Filtering:

Original Frame (Left); All Detected Contours (Centre); Filtered contours and detected VITag (Right)

4) **Blob Detection**[25] **and Identification of VITag**

To detect VITag circles, openCV *SimpleBlobDetector()* was used. Some useful function parameters are filter by size and shape (circularity, convexity, inertia ratio). By tuning these parameters, this aids in further identify potential desired features, which here is the potential VITag data circle.

After blob detection, we are able to locate the x, y coordinates of all blobs in the frame. Each VITag consists of 8 data circles in its contour. The function *pointPolygonTest()* is used to check the position of a point, whether is it within a stated contour. The algorithm loop through each potential contours and blobs, identify target VITag that consists of exact 8 blobs presented in a contour. When this condition is fulfilled, the target VITag is successfully being identified.

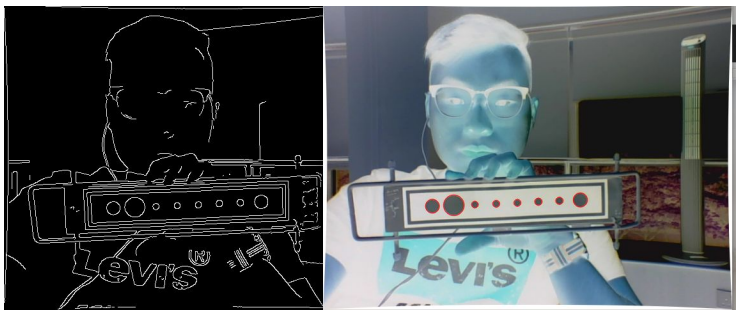


Figure 14: Edge and Blob Detection

3.3 Perspective Transformation

Perspective transformation is a process of changing the frame of reference from different viewpoints viewing the target object. This process is required to get the homography matrix

which relates the mapped points of the VITag contour to a target marker planer view. After the matrix being obtained, wrap perspective imagery of the undistorted VITag can be retrieved. With this process in mind, we break down the task into several subtasks.

1) **Identify 4 extreme corners of the Target ViTag Contours**

   Corner points are needed to do perspective transformation from one quadrangle shape to the target rectangular VITag planer imagery. Accurately identify the coordinates of such corners is very important to reflect the current viewing state of the target.

   For this application, Shi Thomasi method was being used [26]. The function *cv2.goodFeaturesToTrack()*, which uses Shi Thomasi Method requires image frame as input. In this case, the detected contour will be drawn in a plain background, and further input into the Shi Thomasi corner detection function as an image frame. Coordinates of the 4 corners will be output from the function subsequently. Nonetheless, the benefit of using Shi Thomasi method enables more control of the output via its parameter (output number, block size, and k value) and faster performance in computing.

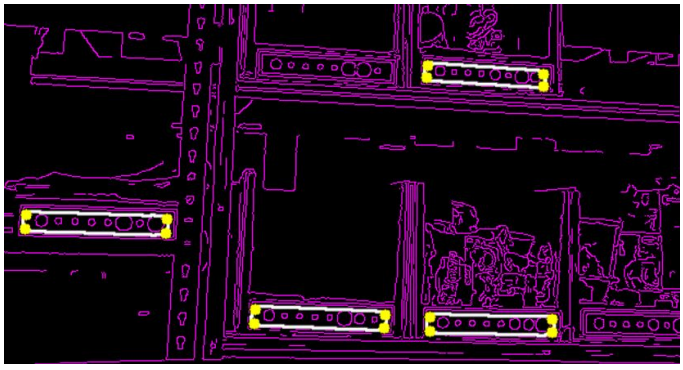


Figure 15: Corner Detection on VITag Contours

2) **Measurement of target VITag dimensional matrix**

   Width and height of the target planar VITag (inner border) is needed for the transformation process. Before that, measurement of the VITag was done physically and

algorithmically, comparing the value measured by the ruler and the derived pixel length. Nextly, a ratio is calculated (width / height) . In this case, a ratio of 8.0 is measured and used to as the target VITag dimensions.

3) Perspective Transformation[27]

By using *getPerspectiveTransform()* function, we will able to obtain the homography matrix of the transformation. The equation is shown here

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}$$

From here, the H (3x3) matrix is obtained and input to a subsequent function, *warpPerspective()*. This function reproject each pixel points from the captured VITag (p, q) to it's planar undistorted view (x, y) after a product with the homography matrix. Consequently, an image frame of an undistorted VITag planar image is being output for further application.

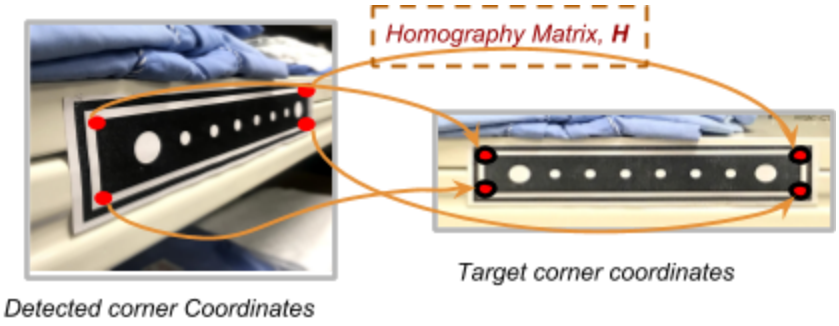


Figure 16: Illustration of Perspective Transformation

4) Obtain pose estimation of the camera frame

To find the pose of the marker in the raw image frame, we used *solvePnP() [28]* function to determine the translation and rotation (T and R) of the two mapped points. The pose estimation process of *solvePnP* was discussed previously on chapter 2.2.

Firstly, 4 pairs of perspective corners coordinates, K and H matrices are provided to the function. Then *solvePnP()* function will be able to output a translation and rotation matrix. The method being selected in this function is SOLVEPNP_ITERATIVE. Iterative method is based on Levenberg-Marquardt optimization. In this case the function finds such a pose that minimizes reprojection error.

5) **Perform multiple ViTags detection**

   The perspective transformation process above is repeatedly performed for all detected VITag contours in the previous contour detection. Output of both rotation and translation matrices of each contour is stored in a global list variables.

3.4 Identification and Matching of markers information

After obtaining the wrap perspective imagery of the VITag, we can further process the obscure data information of the detected VITag. A brief feature of a VITag is shown at the figure below. Data circles comes with 3 sizes, depicts a ternary representation of each VITag index/ ID. The retrievement process flow for getting the pose information is written in 4 steps.

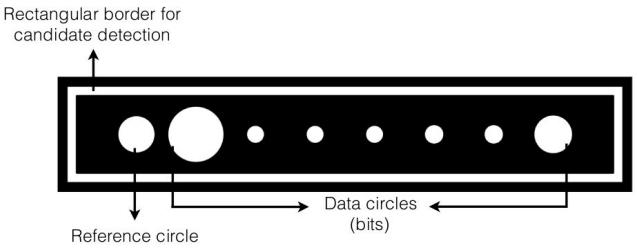


Figure 17: VITag Marker Design [1]

1) **Yaml file for all markers pose information**

   In order to identify the pose of the camera in the world frame, we need to have identifiable landmarks for localization purpose. The landmarks need to have fixed known

coordinates to the camera. In this case, the fixed known coordinates are VITag markers. Having a list of the pose information of each marker in the world frame is essential.

Yaml format is selected as a storage data structure for each VITag marker. As stated previously, 2D instead of 3D environment is considered. Hence, information of the height profile (z-axis), pitch and roll are not important. The resulted data structure is as shown below:

```
# VITAG POSITION YAML
description:
- yaml storing x, y coor and yaw of each marker
- all respective to origin
- Date: 14 March 2018
- Version: 1.0

markers:
  1 : {x: 10, y: 30, yaw: 1.571}
  2 : {x: 40, y: 30, yaw: 1.571}
  3 : {x: 25, y: 60, yaw: 0.785}
  7 : {x: 80, y: 70, yaw: 0.785}
  14 : {x: 135, y: 70, yaw: 0.785}
  22 : {x: 150, y: 80,, yaw: 0.785}
  33 : {x: 180, y: 90,, yaw: 0.785}
```

Figure 18: Example of markers_config.yaml File:

***Key**: marker's index;*

***Elements**: x, y, yaw as pose info*

Take note that each VITag's pose information is respect to the reference point (left top corner of the marker) to world frame origin. Values are in cm or radian. Measurements are being measured then input manually to the yaml file.

2) Detection of contour sizes in markers

Occasionally due to inconsistency in corner detection, black borderline of the VITag imagery will be included in VITag planar target image frame. Thus, a crop factor is used to crop the edges of the frame, which this helps to eliminate unnecessary intruded border lines. This ensured the following contour detection to only detect 8 contours. The detection detects the presence of each VITag data circles, with each circle size is being calculated.

3) **Identify marker's index via data circles**

The leftmost data circle is the reference circle, with the medium size. According to the size of the reference circle, two factored threshold, one smaller and one larger are computed to obtain 2 threshold sizes. 2 threshold sizes help in seperate the ternary data circle sizes into 3 classes.

In relation to the ternary bit format, small data circle (size smaller than the smaller threshold) represents bit '0'; for the larger circle (the size larger than the larger threshold) represents bit '2'; the size which in between the two thresholds represents bit '1'. This ternary form will then convert to a decimal index

.

Figure 19: Showing VITag Data Circles Contours and Recognized VITag with ID 7

4) **Extract pose information Yaml File**

According to the attained decimal marker's index/ ID number, the algorithm is able to match, then get the pose information from the "markers_config.yaml" Yaml File.

3.5 Display Marker and Camera pose

1) **Display marker pose respective to camera**

From the rotation matrix, yaw, pitch and roll is provided. To better visualize the VITag marker pose orientation, a center axis is drawn with the yaw and pitch results.

Figure 20: Output Imagery of VITag Translation (cm) and Normal axis

Nextly, the translation matrix consists of 3 axis values, which these values represents the relative position from the observer camera to the reference point (top left corner of the detected VITag). As the output translation matrix's unit is in pixel, a scale factor is used to factored the matrix, convert the unit to centimeters.

2) **Calculate Absolute Camera Pose in world frame**

Simpler 2D frame transformation was used with x, y, and yaw information. The process transforms points from the camera frame to VITag frame, then to the world frame. Regards the VITag frame to the world frame, the pose info of the VITag is obtained from the YAML file. After the 2D transformation, we are able to calculate the absolute camera pose in the world frame merely via VITag detection.

3) **Average Absolute Camera pose output**

As mentioned, the transformation process involved when one VITag being detected. If multiple VITag being detected by the camera, its pose estimation and transformation will be separately computed. With a list of results from each VITag, the end results will be averaged, providing a better reading of the pose estimation. In this case, more detected VITag will provide a much accurate value of the camera's pose in a 2D space.

4. IMPLEMENTATION: IMU AND ENCODER

Here, we will discuss on integrating the IMU and encoder readings into the localization system. Raspberry pi [29] will be used as the medium to receive the sensor reading.

4.1 Raspberry and IMU Setup

Raspberry Pi 3[29] is a powerful low-cost microcomputer which commonly used for prototyping. Sensors can be connected to the embedded Raspberry Pi GPIO pin. IMU model L3GD20[30] from Pololu is being selected for our setup

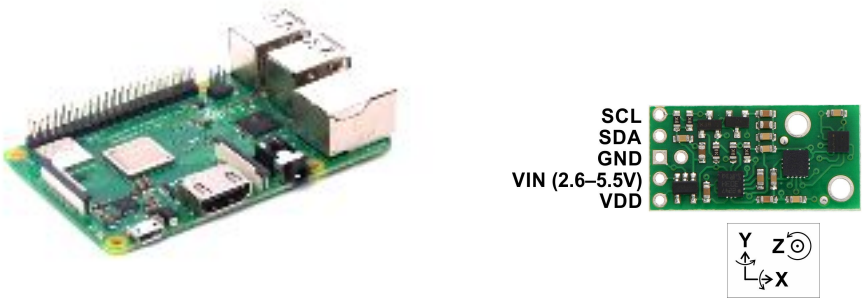


Figure 21: Selected Raspberry Pi 3 and IMU

1) **Setup of Raspberry Pi 3**

   Ubuntu Mate is used as the OS of the Raspberry pi module. Relevant libraries and dependencies are being installed.

2) **Connect IMU to GPIO**

   IMU is a accelerometer, compass and gyroscope on its own. By having 9 DOF, we are able to capture the acceleration of the IMU with a sample rate of 1kHz, and also the yaw information with its compass, sample rate of 8Hz. The pins of the IMU are being connected to the respective GPIO pin. The reading of the sensor input is streamed and read with a Python script.

3) **IMU Setup and Calibration via RTIMU**[31]

RTIMU Library is being used for managing the raw IMU data input from the sensor to the device. After installation of all relevant dependencies, a RTIMULib/Calibration.pdf in github provided further resources on the calibration process. Generally, the calibration process involved normalization and offset computation for all the IMU sensors: accelerometer, gyroscope and magnetometer.

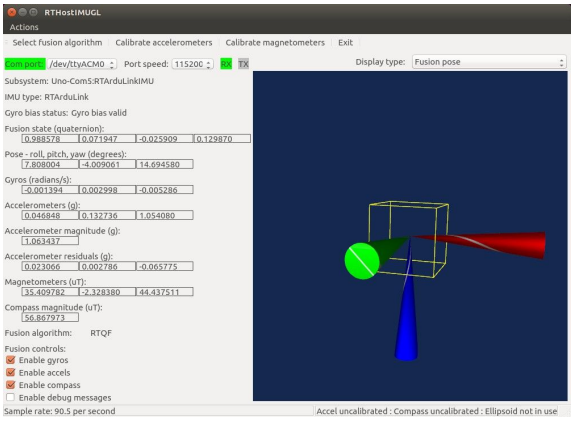


Figure 22: RTIMU IMU Calibration GUI Tool

4.2 IMU Integration

1) **Reading of Raw IMU Input with RTIMU**[31]

By using RTIMU module in python code, the RTIMU class simplify the process of obtaining the IMU sensor reading from the I2C bus. Upon the execution of the code, the module conduct an initialization process with the *RTIMU_xxx.ini* calibration file. This module able to write and read from the IMU addresses. Internally, the module will operate a stripped down kalman filter process, known as RTQF sensor fusion. RTQF will fuse the 9 DOF sensor reading to a 6 DOF output. The output comprises of acceleration values on xyz-axis and yaw-pitch-roll sensor pose, hence 6 DOF in total.

2) Initial Managing of Sensor Input

For pure odometry localization, double integration is needed so that the displacement profile is obtained. However due to the death reckoning nature of the IMU sensor, constant odometry drifting will occur along time. Because of this error accumulation, a careful management of sensor input is important to minimize the drift.

a) Recalibration and noise elimination.

It is observed that the IMU sensor is noisy in nature, especially with the presence of gravity. Thus, it's important to filter the raw input in order to reflect the true odometry of the robot. To begin with, observation has shown that the first tens of data samples during the starting stage is hugely inaccurate. This is when the I2C bus deliver the first few number of sample readings. As a result, first few samples are ignored by the system.

Additionally during the initial stage of the program, a recalibration is being conducted. User is required to place the IMU stationary upright, later then few hundred of samples are being collected and averaged out. This will help us to get the offset of each accelerometer. This offset will be used for the subsequent accelerometer readings. The removal of the offset is shown at the figure below.

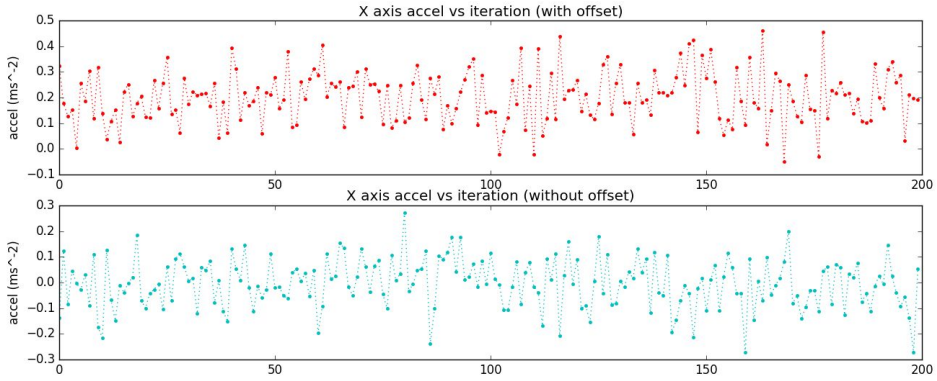


Figure 23: Plotting of IMU Acceleration Reading:
with offset of 0.2 ms^{-2} (above), removed offset (below)

b) Transformation of acceleration axis

The captured acceleration values are in terms of the x-y-z axis of the IMU sensor. In contrast, the roll, pitch and raw readings are respect to the earth frame. This implies that the yaw direction of the magnetic North is 0 degrees. To simplify the coordinate system of multiple frames, x-y-z axis acceleration readings will be transformed to the earth frame. Which now, the 3 output values will be

- Coordinate system: North-South axis, East-West axis
- Yaw angle of the IMU respective to the magnetic north.

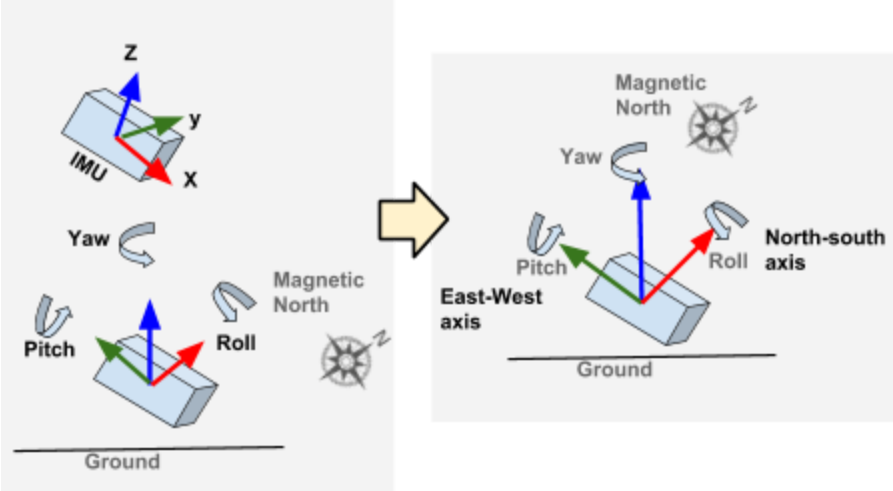


Figure 24: Diagram showing the transformation of xyz acceleration axises

To conduct the 3D transformation, a 3D rotation matrix with representation of RPY is used to transform the acceleration values from the original frame to the earth reference frame.

c) High sampling rate

High sampling rate ensure that the full acceleration profile is being captured accurately, then send to the user.

4.3 Mobile Platform with Encoder Integration

1) Fixing of Camera and IMU position

A mobile platform was used to create a stationary platform for the IMU and Camera. Both IMU and Camera frame orientation to the platform was recorded, with 0 and -90 degrees respectively. Fixed parameters are stored in the markers_config.yaml file.

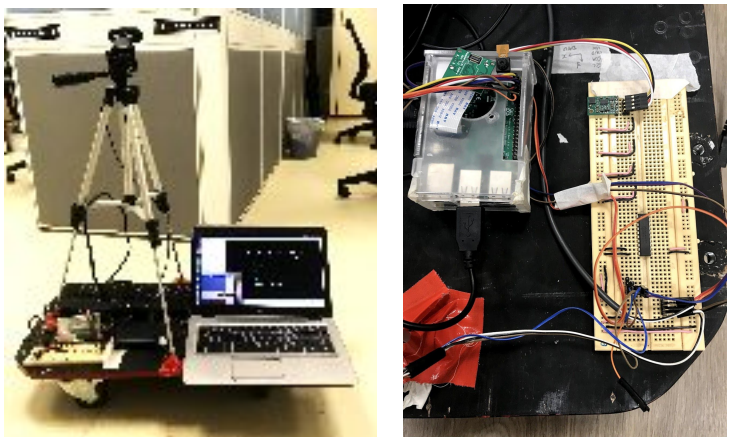


Figure 25: Fixing of Camera and IMU on Mobile Platform

2) Attach 2 Encoders on 2 axis

Encoder Model: OMRON E6A2-CWZ3-E[32] was selected for this project. 2 OMRON incremental encoders were used to record the change of displacement for the platform's x and y axis. To begin the hardware installation, 2 normal wheels with proper diameter were attached to each encoder. Subsequently, 2 encoders were installed under the moving platform with a right angle to each other, ensure the x and y axis motions are captured. The encoders installation is shown below.

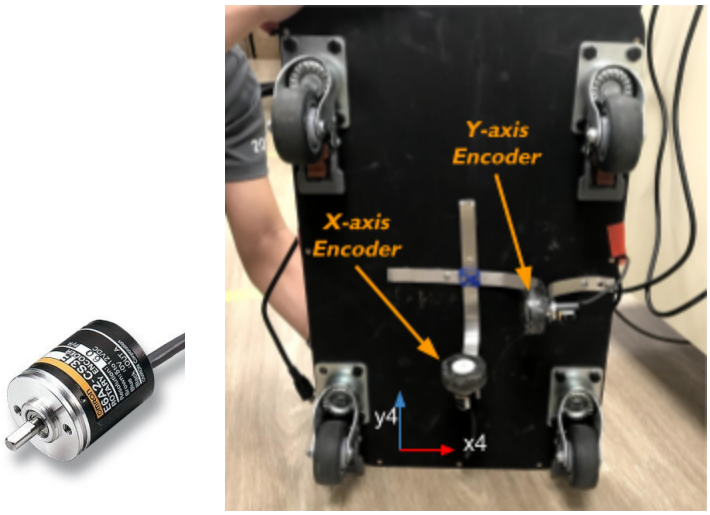


Figure 26: Model of Encoder and Installation Underneath Mobile Platform

3) Capture of Input Pulses Data

The model of encoder used consisted of 3 channels (A, Band Z). The pulse phase diagram for each channel is shown below. It is known that a single rotation will create 100 pulses (channel A&B). Thus with the pre-known diameter of the program, we will be able to estimate the displacement. Both channel A and B is known to have a quarter wavelength of phase difference. This further helped in getting the rotating direction of the encoder.

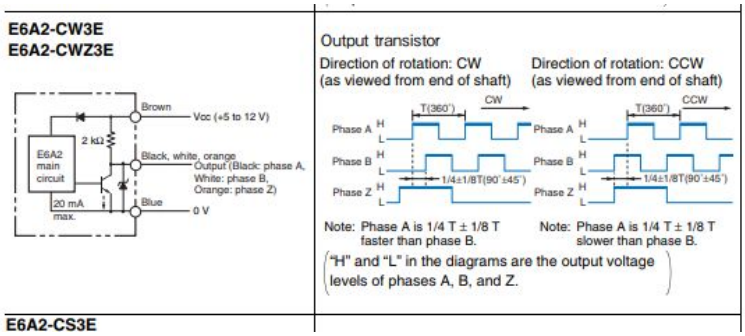


Figure 27: Encoder Internal Circuit and Output Diagram

4.4 Server-Client Communication

IMU and encoders raw datas are both received by the Raspberry Pi. Next, the signal will be further sent via wifi to a device performing the pose estimation computation, in this case an operating laptop. To achieve this, a python server-client creator module, “socket” is used. Generally, the laptop will host a server, then the raspberry pi (client) will send the sensor reading to the server via local network connection. 2 threads were generated in the hosting server process to receive signal from each input sensor independently.

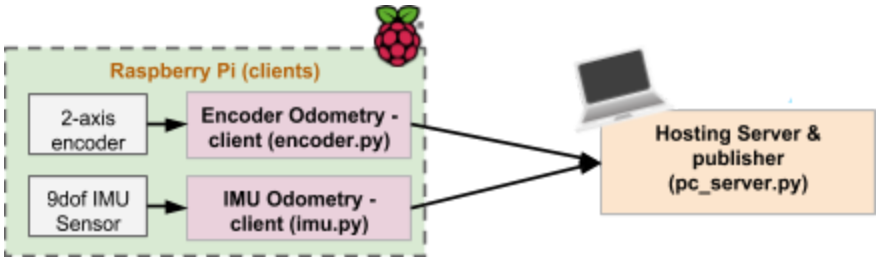


Figure 28: Laptop and Raspberry pi Communication

To reduce the overwhelmed usage of server-client data transmission, sample sizes from the Raspberry Pi were reduced. For IMU, 10 samples with computed acceleration will be averaged, then subsequently being send to the server. Encoder send displacement data every 0.05s after process the number of pulses. Both resulted in a lower sampling rate of about 20 Hz.

5. IMPLEMENTATION: SYSTEM INTEGRATION

5.1 Frame Transformation

Due to having various hardwares and sensors used, each part/component will have their own respective reference frames. A simplified diagram showing all reference frames is illustrated below. Proper 2D frame transformation was done to transform all frame to a common frame, in this case, the world frame.

In general, points in IMU, camera and Encoder(platform) frames were transformed to the earth magnetic frame, then the world origin frame. The camera position in VITag marker's frame was also transformed to the world origin frame. 2D frame transformation is extensively used here.

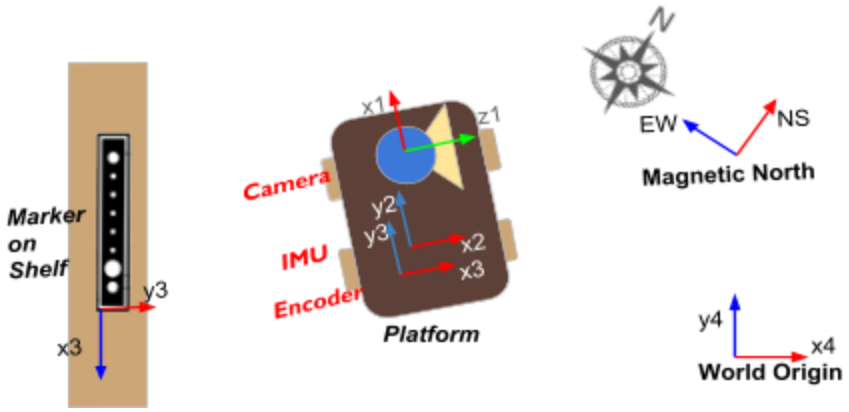


Figure 29: Simple Diagram of Reference Frames Used

5.2 Sensor Fusion

To implement sensor fusion, Kalman filter (linear quadratic estimation) is being used for pose estimation enhancement. This conventional sensor fusion method could effectively reduce the effect on sudden deviation in the vision localization. Also, multiple sensor readings can be fused ellegantly with the help of gaussian-based Kalman filter. Here we will discuss on how the fusion script: fusion.py perform its sensor fusion task.

1) Multi-Callback

By this point of time, it is obvious to realize that the fusion algorithm is required to run 4 tasks independently, hence multi-callback is used to achieve efficient splitting of tasks.

	Task	**Communication**	**I/O**
Main Thread	Timestamp matching, frame transformation then Kalman Fusion	Tf.transformations [33] to Rviz visualizer [10]	Output

1st Callback	Receive reading of Vision Pose Estimation	ROS std msg [34], with Float32MultiArray	Input
2nd Callback	Receive IMU accel and yaw reading from pc_server.py host	ROS std msg, with Float32MultiArray	Input
3rd Callback	Receive encoder displacement reading from pc_server.py host	ROS std msg, with Float32MultiArray	Input

Table 1: Multi-callbacks in Sensor Fusion Process

2) Sensor Fusion with Kalman filter

Each axis, namely x and y-axis will be fused independently via Kalman filter. The final output yaw angle will only dependent to the IMU yaw reading as it's much accurate than the vision's reading.

As mentioned before, covariance matrix is one of the main component of Kalman filter. Covariance matrix coefficients are variances of the probability in gaussian distributions for the desired value. Selection of coefficients are according to the observed noise level and various adjustment during trial and error. The coefficients represent the degree of uncertainty or reliability of the sensor reading. Here we used 1x1 and 2x2 covariance matrices (depends on application) to represent the uncertainty level.

3) Kalman Filter: Odometry with time update (IMU case)

IMU odometry is commonly known as death reckoning. The prediction of the next state is based of the result of the previous state. Henceforth, a error will slowly accumulated, which eventually will create a large output deviation. In this case, the state estimate (x^k) here consisted of [x, v], which represent displacement and velocity respectively. Thus we will need to obtain then compute the value estimation of both displacement and velocity from IMU and vision pose estimation input. [35]

$$Position:\ x^t = x^{t-1} + v^{t-1}\Delta t + \tfrac{1}{2}a^t\Delta t^2$$

$$Velocity:\ v^t = v^{t-1} + a^t\Delta t$$

Two desired variables is needed for the time update process, which is the state estimate ($\hat{x}$) and covariance matrix (P). These two is updated along time, which depends on the each previous value. Given these, the equation of the time update is shown below:

$$State\ Estimate:\ \hat{x}_k = A_k\hat{x}_{k-1} + B_k a_k$$

$$Estimated\ Covariance\ Matrix:\ \hat{P}_k = A_k\hat{P}_{k-1}A_k^T + Q_k$$

$$Where,\ A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix},\quad B = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix},\quad \hat{x}_k = \begin{bmatrix} Position \\ velocity \end{bmatrix},\quad P_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

4) Kalman Filter: Measurement Update (IMU Case)

Measurement itself refers to the vision pose estimation input. The pose reading from vision is a absolute pose information, along with certain variance of uncertainty. Here, the measurement will only update when a new vision reading is being send over. Thus, measurement update happen less frequent than time update.

To compute a fusion result, 3 more equations are being here to update the measurement. First, Kalman gain is calculated along the update of the new covariance and state estimate. Then, step measurement update is conducted for both $\hat{x}'_k$ and $\hat{P}'_k$. In this case, z_k is a 1 x 2 matrix [x, v] from the camera input. The [x, v] matrix consists of an absolute position of and estimated velocity of the camera.

$$Kalman\ Gain:\ K' = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

$$State\ Estimate:\ \hat{x}'_k = \hat{x}_k + K'(z_k - H_k\hat{x}'_k)$$

Estimated Covariance Matrix: $\widehat{P}'_k = \widehat{P}_k (1 + K' H_k)$

The H_k matrix, which is used to relate the sensor reading to the acceleration. Take note that H_k is omitted because the scaling has been done in the previous function. Also, the R_k here is sensor (vision input) covariance of uncertainty, which can also be the noise of the vision input.

$$\begin{bmatrix} Position \\ velocity \end{bmatrix}, \quad P_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

5) Kalman Filter on Encoder Odometry

The above kalman filter is based on a much complicated 1 x 2 matrix [x, v] input while using IMU as the time update. For the case of encoder, if encoder odometry and vision measurement update is used, the equation is simplified to merely a single [x] as input. Covariance matrix will be also be a single value coefficient instead of a 2 x 2 matrix.

Simplified Encoder Odometry State Estimate: $\widehat{x}_k = \widehat{x}_{k-1} + \widehat{x}_{encoder}$

6) Result Testing and Optimization

With the input of IMU or encoder and vision result, we used timestamp to further synchronize the two independent sensing process. As both sensing sources have different sampling rate, we will need ensure the right match of both signal to achieve a better representation of values during sensor fusion.

Prior to the integration of sensor fusion, simulation of 2 noisy sensor inputs with simulated noisiness was being conducted. Matplotlib[11] was used to visualize the datas Kalman Filter process. Right below, the diagram will show a simulated fusion result, with camera(low sampling rate, red) and IMU (high sampling rate).

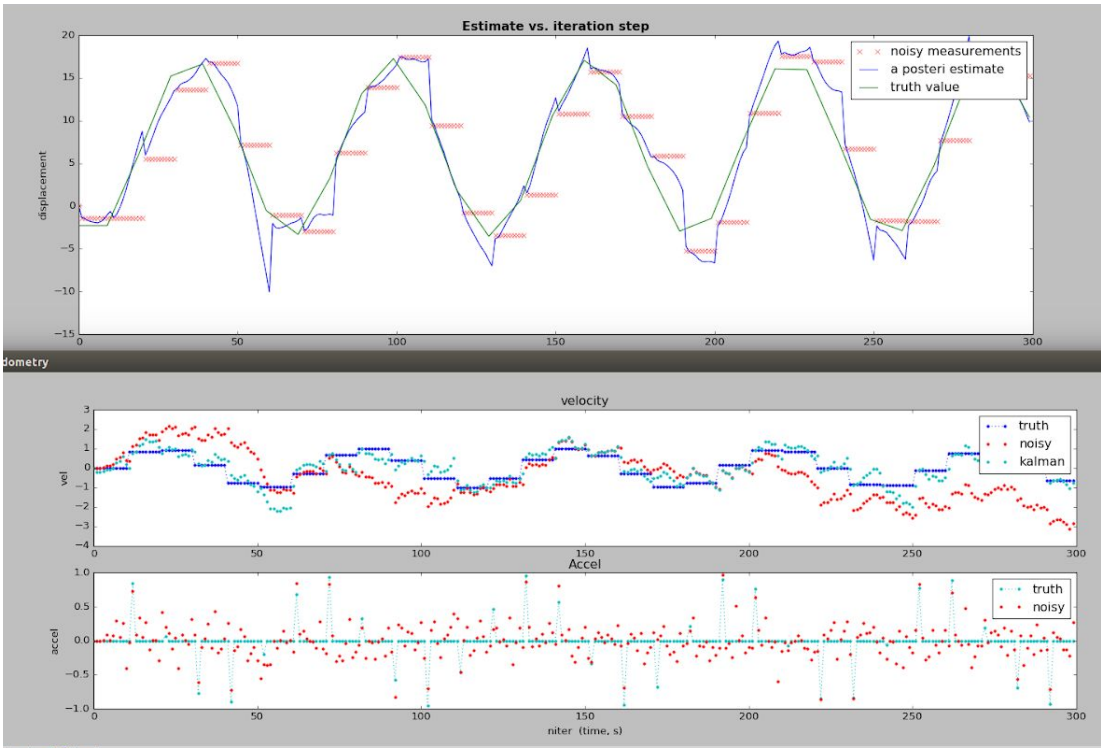


Figure 30: Kalman Filter result on Single Value Displacement

The First diagram shows the effect of kalman filter on displacement. Red crosses represent the camera measurement input, and blue represent the time update which dependent of IMU. Green line is the ground truth. The Graph below shows the breakdown of velocity and acceleration reading/profile.

With the comparison above, due to the noisy acceleration reading from IMU, we are able to observe that the fusion output is not perfect comparing with the ground truth. Still, the reading is still good enough to represent the position of the robot. Also noting that kalman filter for encoder odometry will have a better result due to its displacement inputs. In general, this fusion algorithm is then being integrated in the main fusion system.

5.3 Overall System Integration

1) Setup of ROS

ROS works as a communication channel between different nodes. By using Rviz [10] in ROS, we are able to visualization the position of the VITags and camera in a common world frame. Prior to the execution, ROS was set up with a private package file. Python Language is mainly used for the execution of each nodes.

2) Communication Framework

Multiple devices and processes are involved in this localization project. In order to accomplish this, a clear framework helps in understanding of the whole communication processes. The simplified framework is shown below:

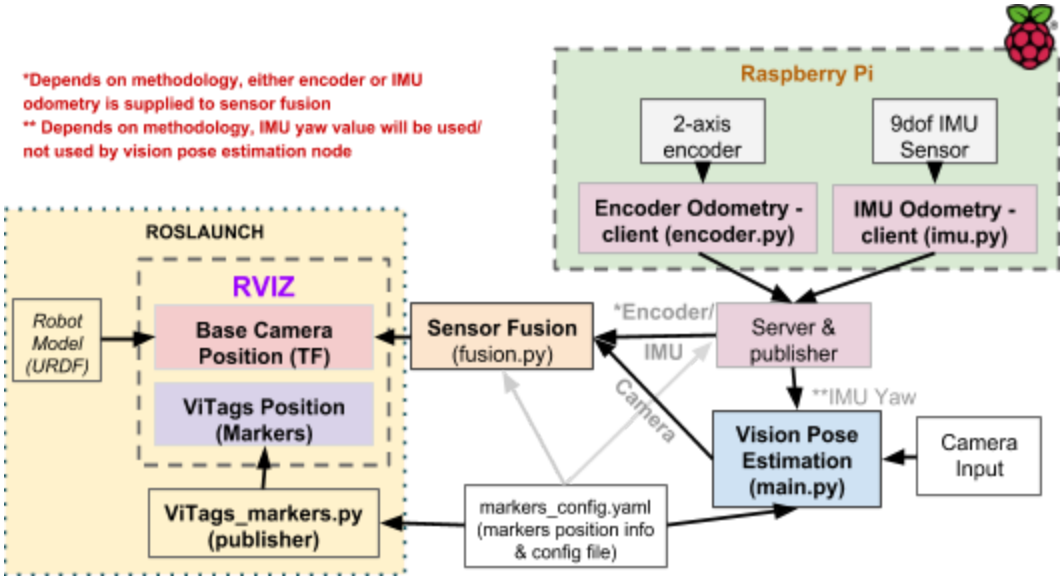


Figure 31: System Framework of localization process

With the framework shown above, a clearer picture is given to help in the integration of various tasks. In general, various node were done separately before the final integration. Note that the encoder & IMU odometry and IMU yaw value communication vary on different methodology. The variation will be discussed in the experiment below.

5.4 Visualization Tools

With the help of kalman filter, we are able to output the fusion results to the TF topic in ROS. Despite that, visualization tool is essential to the showing of data to the user. This assists in debugging and analysing. Here, the table below shows the tools that are being used for visualization.

Tools	**Task**	**Role**
Rviz[10]	3D Robot Environment visualization tool	TF.transform and visualization_msgs subscriber. Provide 2D visualization of all elements in a 2D planar environment
URDF[36]	Representation of CAD Model of the robot.	XML format (or import .stl, .obj...) showing link and joint, access by Rviz for visualization. This helps to show the pose estimation of the robot in Rviz.
Markers	VITag markers and navigation path updates	Marker Publisher to Rviz. Enable Rviz to show the position of each ViTag markers, and navigated path for the robot.
Matplotlib [11]	Visualization of datas in chart/graphical form	Python Graph visualizer. to show datas of each process.

Table 2: Visualization Tools used in Localization Process

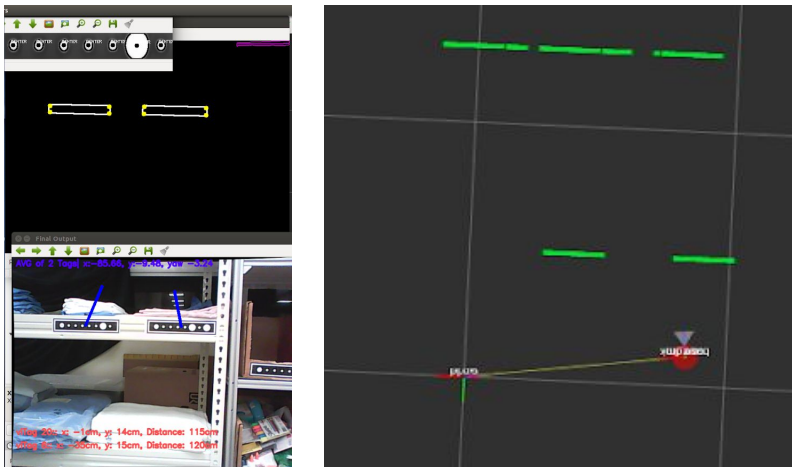


Figure 32: Rviz used in visualization of pose estimation

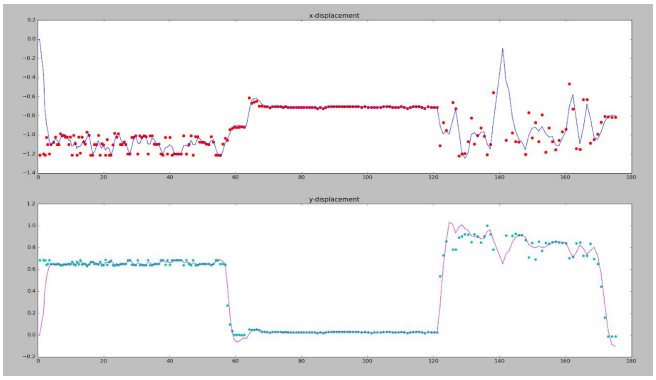


Figure 33: Matplotlib plotting of Fusion Output

With all these tools equipped for the testing and experimental process, we are prepared to conduct some testing and experiment. Results of the completed localization system will be evaluated with the help of these visualization tool.

6. EXPERIMENT

After the implementation of the localization system, experiments were being conducted to obtain the results of each performance. Here we separated these experiments into multiple configurations based on different sensor combinations:

1) Vision-Based Localization
2) Vision and IMU Odometry Localization
3) Encoder Odometry and IMU heading Localization
4) Vision + Encoder Odometry + IMU heading localization

6.1 Setup of experiment

The experiment is required to conduct a manual drive of a “mobile” testing device to test out its localization effectiveness. As that, we used the previous constructed mobile platform to emulate an autonomous indoor mobile robot. Relevant nodes (according to the communication framework) need to be activated in order to run the localization system. In effect, Rviz and Matplotlib will be used to observed the testing results.

The Robotics Research Center (RRC) staff office cubicle area was used to conduct the experiment. This office environment represented an indoor environment, which equipped with VITag visual markers fixed along the corridors.

Figure 34: Testing Environment and Setup

A fixed path around the indoor testing space was also selected, as shown below.

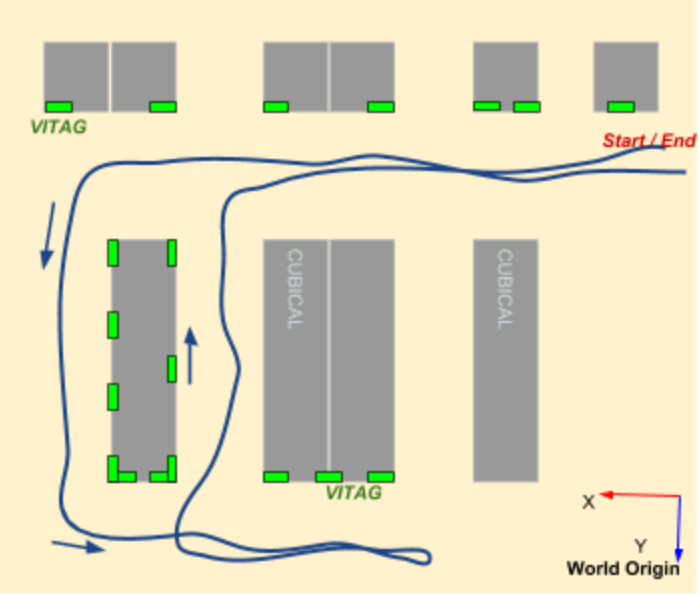


Figure 35: Fixed close-loop path in testing space with labelled VITag locations

6.2 Vision-based Localization

Vision-based localization indicates that vision pose estimation output is the main source of the overall localization system. According to the diagram below, we will test out two cases, which are case A and B. The significant of B is that the yaw angle of the camera is not based on the pose estimation output, but the IMU absolute yaw angle value.

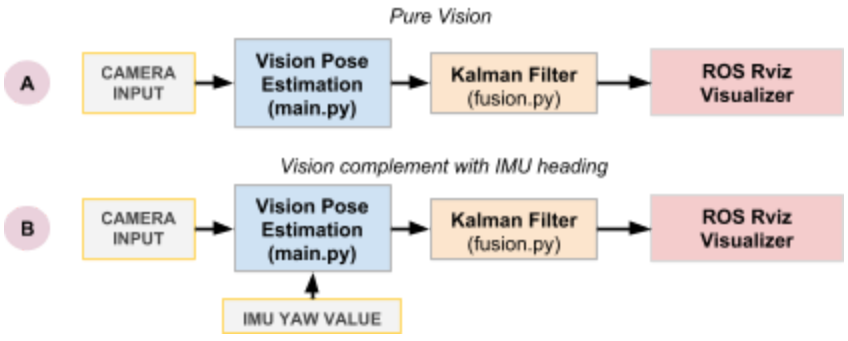


Figure 36: Simplified Process Diagram of Vision-based Localization

A) Pure Vision

This experiment is conducted based on a single camera as the only sensor input for localization. IMU reading is ignored from this process. Within the Rviz interface, VITags position are labelled as green boxes. A red circle is used to represent the sensing device, while a grey triangle is the camera viewing angle. This shows the rotation of the camera in the 2D environment.

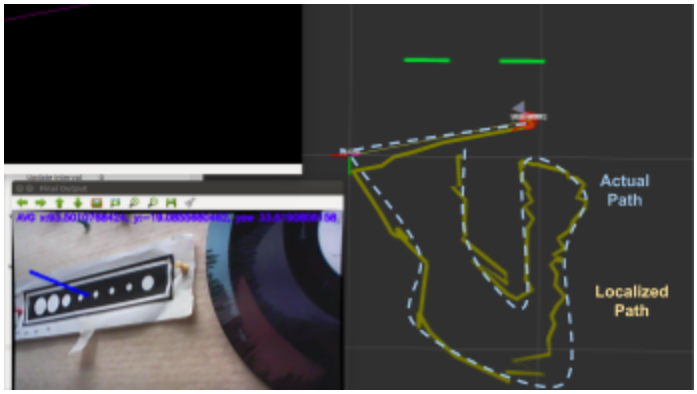


Figure 37: Showing Deviation of Path due to Inconsistency on Localization

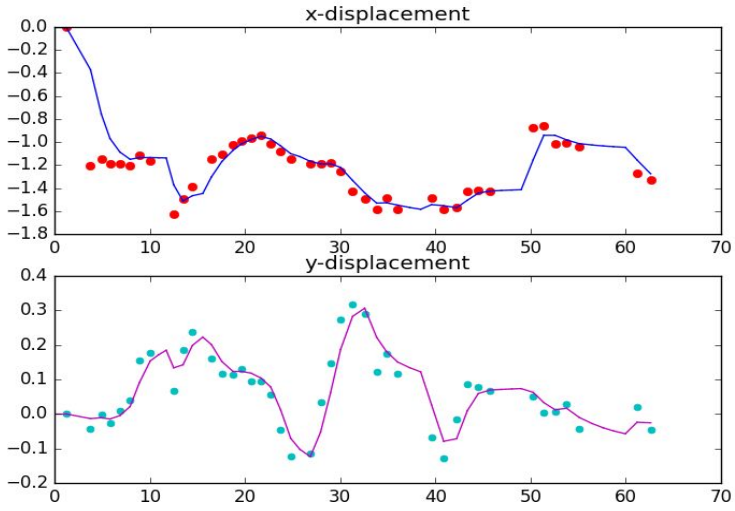


Figure 38: Kalman Filter of vision based localization (without IMU input)

According to our observation, the path was not consistent while there were some leaping of localized points during the real time process. This is mainly due to the varied and deviated yaw angle output of the homography pose estimation process. Notably, deviation will be larger when the camera is located further to the VITag marker.

B) Complement with IMU heading

As stated above, it was observed that the rotation matrix from marker's homography transformation contains certain error. Along the 2D transformation process, the deviation of the yaw angle will magnify. Thus, pure vision pose estimation yaw value is less reliable. To overcome this issue, we used the IMU absolute rotation matrix (yaw) value in the vision pose estimation node. Here, the IMU yaw value is used during the 2D frame transformation process.

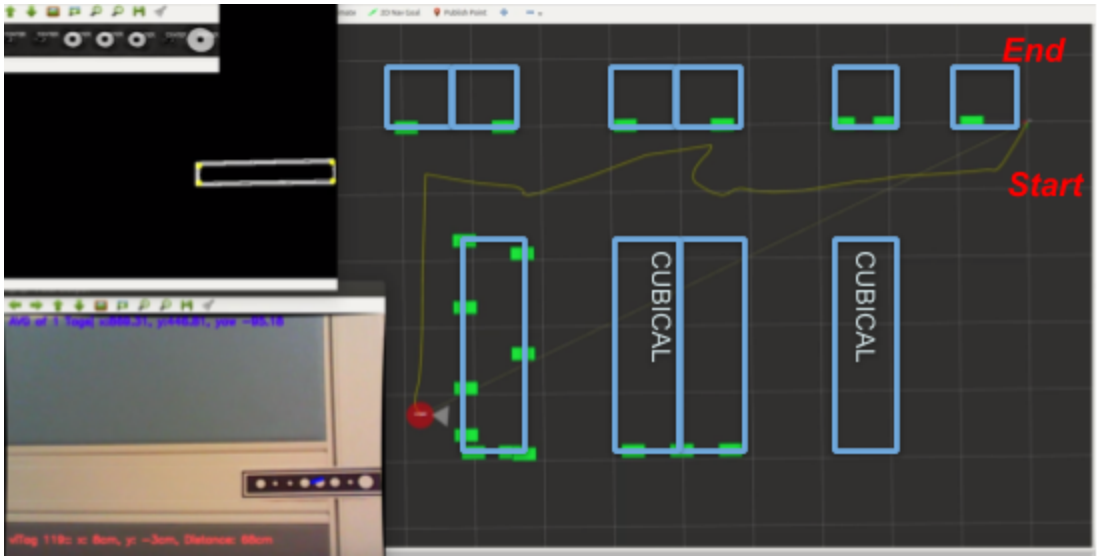


Figure 39: Path (yellow line) of Vision and IMU heading localization

(Video refer to Appendix)

According to the shown results, it can be seen that the vision-based localization is accurate, with a error range of $\pm 12.5cm$. The vision localization system that complemented with IMU heading provided a better performance.

However, one main limitation is the need to constantly pointing the camera to the markers, ensure the device is constantly being localized. The effect of non-continuous localization can be seen in video. This discrete samples along with low sampling rate (6.7 hz) create a less reliable localization system.

6.3 Vision and IMU Odometry Localization

Kalman Filter was used again to fuse both localization results from camera vision and IMU odometry. In theory, it will provide support in obtaining a reading with greater accuracy.

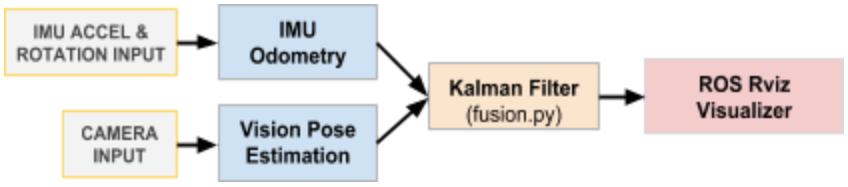


Figure 40: Simplified Process Diagram of Vision and IMU Odometry Localization

Incorporation of IMU odometry enables a less restrictive localization zone. Frequent measurement update from vision is needed with the purpose of eliminate the severe odometry drift. However, results are contrasted to the one expected. Tests were being conducted and results were being shown here.

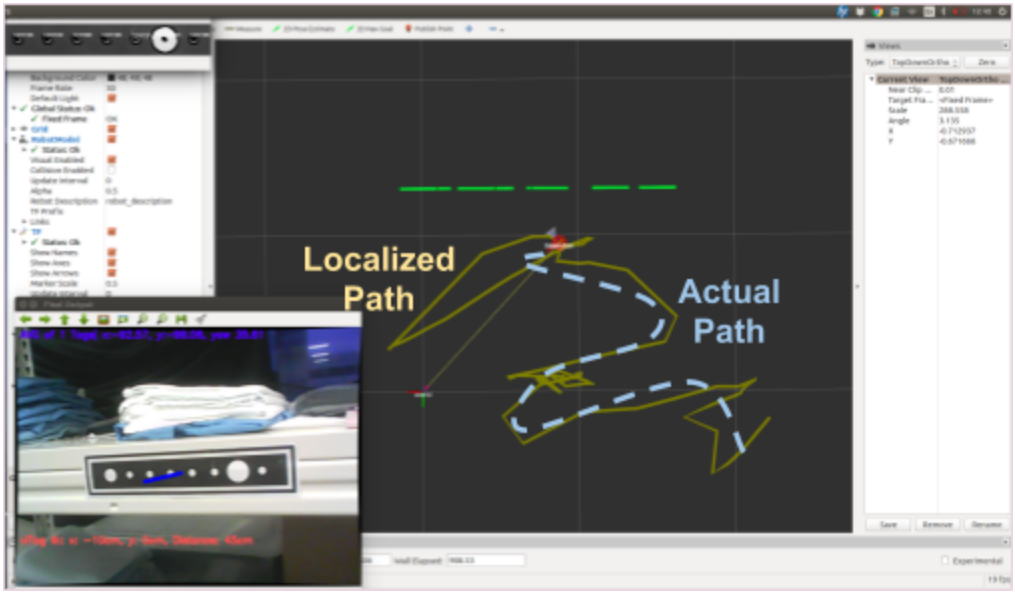


Figure 41: IMU odometry Sensor Fusion Testing Result 1

Figure 42: IMU odometry Sensor Fusion Testing Result 2

As shown in the figures above, results have shown that incorporation of IMU has worsen the localization results. The localized path was significantly deviated from the actual path. The above experiments were conducted with constant camera occlusion. Although when absolute

measurement (vision) update is still accurate, the interval between each measurement update will be affected by drifting.

Due to the noisy and sensitive nature of a IMU acceleration value, IMU odometry as input is highly not desirable. Therefore, testing of this system on the experimental path was not conducted. Despite that, it was observed that the IMU rotation matrix (absolute yaw heading) is still reliable to used in a localization system.

6.4 Encoder Odometry and IMU Heading Localization

Incorporation of encoder was tested to determine its odometry output compared with IMU odometry.

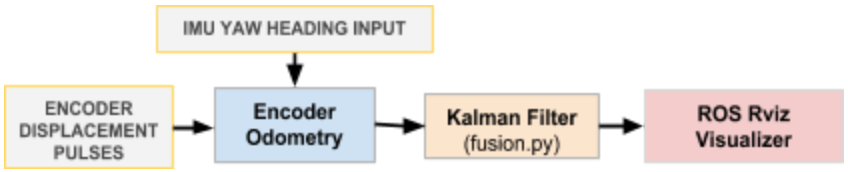


Figure 43: Simplified Process Diagram of Encoder Odometry and IMU Heading Localization

As named, this localization method consisted of encoder x-y axis odometry for its translation, and IMU heading (yaw angle from rotation matrix). This enables death reckoning localization based on origin. Drifting will occur due to its non absolute value. This accumulation of error deviates the groundtruth path, as shown as below. Still, the result was relatively consistent and stable than IMU odometry.

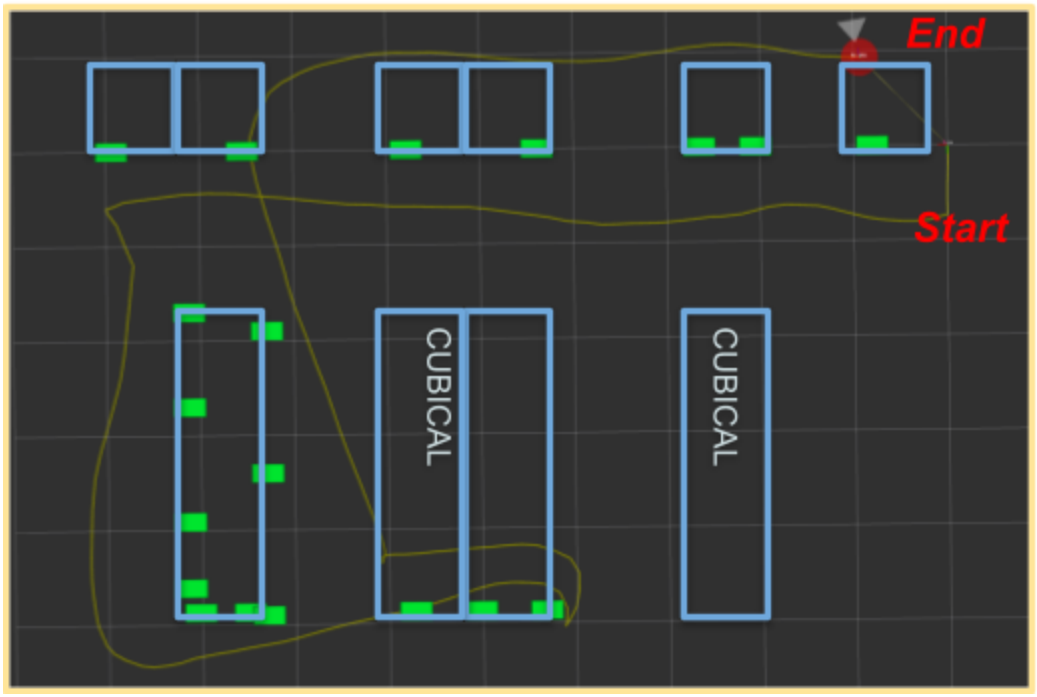


Figure 44: Drifted Path (yellow line) of Encoder Odometry and IMU heading localization

6.5 Vision + Encoder Odometry + IMU Heading Localization

Pros and cons of each sensors were evaluated based on the previous experiments. Thus, by leveraging on each sensors advantages, the last attempt: vision, encoder odometry and IMU heading localization is being implemented and conducted.

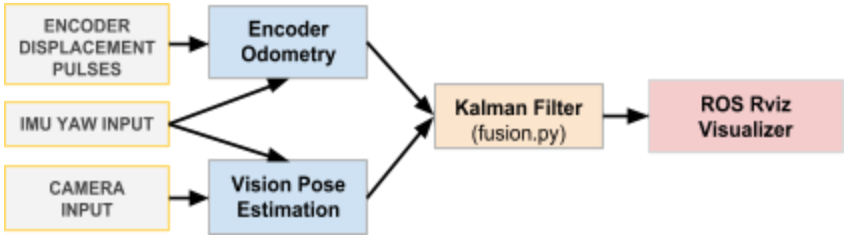


Figure 45: Simplified Process Diagram of Vision + Encoder Odometry + IMU Localization

This localization fused all sensors readings, while incorporated kalman filter to achieve a better and reliable real time localization performance. The results is shown here:

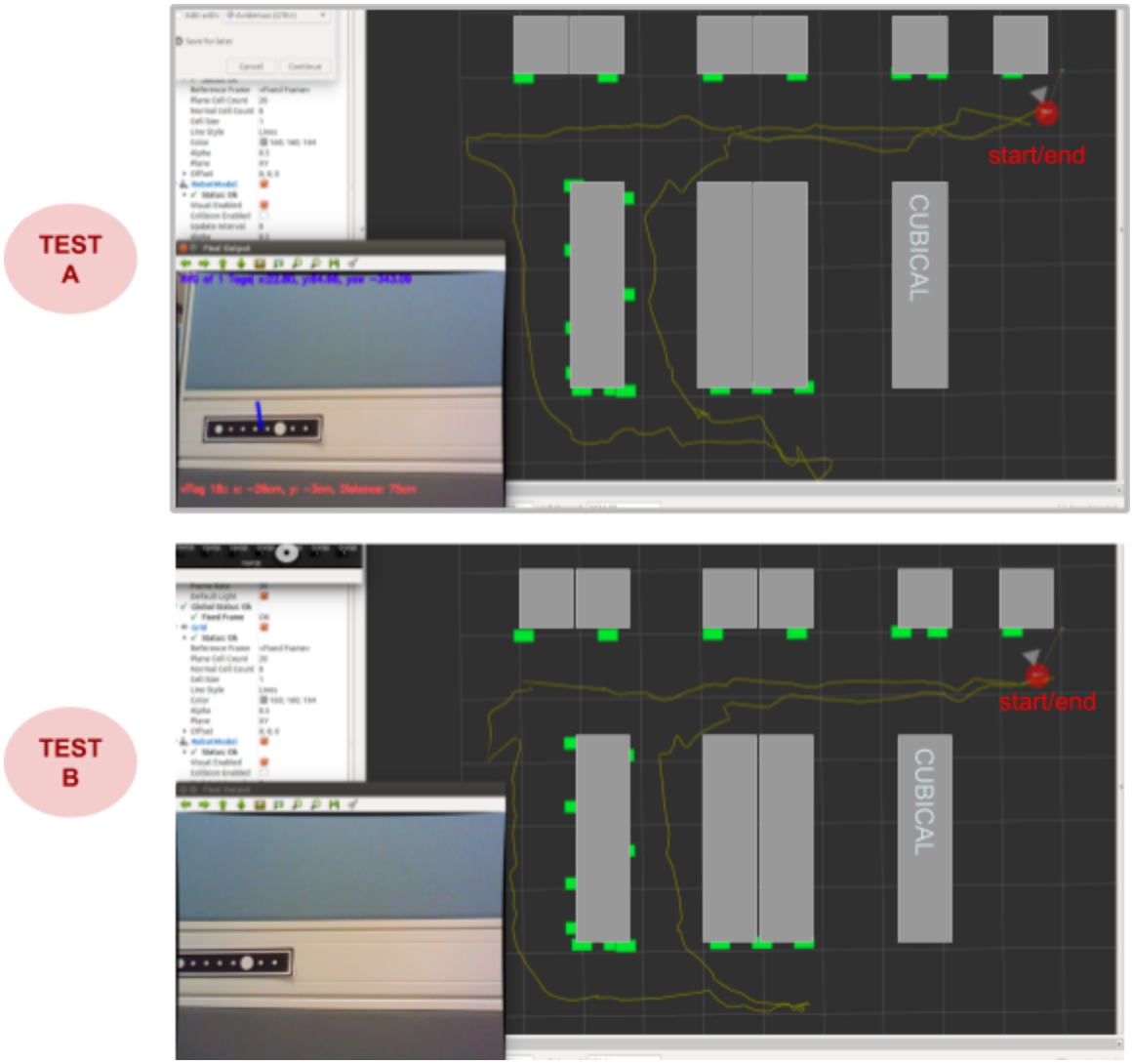


Figure 46: Path (yellow line) of Vision, Encoder and IMU fusion based localization (Refer full video at Appendix)

6.5 Overall Comparison on Localization System

After conducted experiments on 4 different methods of sensor configurations, some conclusion on each method is shown in the table below.

Localization Method/ configuration	**Comments**
Vision-based	Discrete and non-continuous localization, Camera input: 3- 6.7 hz
Vision + IMU Odometry	Being affected by IMU, severe drifting during vision interval
Encoder Odometry + IMU Heading	Stable, less computationally intensive, odometry drift
Vision + Encoder Odometry + IMU Heading	- Best performance among all; - Stable and accurate pose estimation readings; - Within precision of +- 12.5cm if have markers in every 2m interval - Precision increases when nearer to detecting marker - Most complex, involved running of multiple nodes concurrently

Table 3: Comparison of different localization method/configuration

In summary, the 4th configuration which used vision pose estimation, encoder odometry and IMU heading for localization resulted with the best performance.

7. CHALLENGES AND DISCUSSIONS

Different challenges were faced along the way. To tackle on the surfaced problems, multiple solutions or attempts were used, all these will be further discussed here.

7.1 Attempts on Vision Localization

During the implementation of the vision localization, several problems were faced. To have better performance, different approaches were implemented to obtain the best result among all. Some of the ineffective attempts will be discussed here.

1) **Detection of Data Circles in Raw frame**

   During the initial stage of identifying the potential VITag in a raw grayscale frame, distinct feature of a VITag (data circles) need to be recognized. Being said so, potential data circles need to be spotted within the raw frame.

   a) **Failed Attempt 1: Contour Detection**

      Contour detection is relatively inconsistent when the VITag is placed further away from the camera. Small data circles can be too small to be identified as a contour. On the other hand, further analyzing the shape feature of each contour will be computationally intensive.

   b) **Failed Attempt 2: Hough circle** [37]

      The *houghCircle()* function is a feature extraction function which uses hough transform to identify potential circles in an image. However, similar to houghLine() function, this process is computationally intensive. Constant flickering issue will create inconsistent result to the VITag detection output.

Figure 47: Hough Circle Method (output circles show in green)

c) Successful Attempt: Blob Detection

SimpleBlobDetector() function is the chosen way to identify potential data circles. With higher performance and more controls of parameters (e.g. circularity threshold) feature, it is able to effectively recognize blobs pixel coordinate from the input frame. Subsequent filtering with contour detection was used to filter false positives.

Figure 48: Blob Detection on Data Circles with False Positives (Red)

2) VITag Corners Detection

3 methods were implemented for corner detection in order to obtain a better result, which is the coordinates of the 4 corners.

a) Failed Attempt 1: Contour Approximation[38]

This was the first attempt of the corner detection, with a OpenCV function: *cv2.approxPolyDP()*. List of contour points of the VITag boundary was provided to the function. Within the process, it will identify the shape with the closest fit to the respective contour shape. Input value epsilon, ε represents the maximum allowable distance of the original contour to the new best-fit contour. Maximizing ε will enable output as few points as possible, in this case, 4 points are needed.

However, according to this method, the algorithm tries to fit the shape with the new contour, minimizing the difference of distance of both contours. With no perfect quadrangle shape, this will cause unstable movement and shifting of the corners coordinates. This resulted in a distorted quadrangle contour output.

b) Failed Attempt 2: Corner Harris[39]

Due to the shifting and inconsistency of corner points, Corner Harris method was being implemented to deal with issue. With the *cv2.cornerHarris()* function, the advantage here is the algorithm evaluate one as corner in a certain pixel block area. Block size can be adjusted as parameter in the function. An image frame will be inputted into the function, which respective potential corners will be outputted. Appropriate block size and K value are determined by adjusting, which eventually provide accurate contour corners coordinates. However, intensive scanning resulted in slower performance is the flipside.

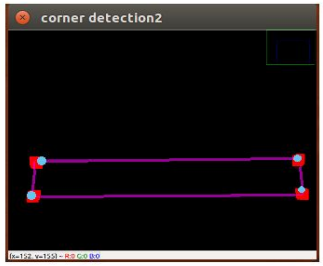


Figure 49: Result of Corner Corner Detection; Left Image: Contour Approximation (Blue), Harris Corner Method (Red)

As shown at the figure above, Corner Harris Method, in Red is much better than Contour Approximation method, in blue. The drifting of blue corner coordinates with Contour Approximation method will result in a distorted output VITag target frame, as shown at the right figure above.

c) **Successful Method: Shi Thomasi**

Similar to Corner Harris method, block size instead of individual pixel point is being evaluated to locate the position of the corner of the VITag contour. Contrary to Corner Harris method, the opencv function for Shi Thomasi method provides more parameters control to the output. The output itself can be set to 4 coordinate points with the strongest corner feature. Performance is relatively faster than Corner Harris approach.

As a result, Shi Thomasi method shows a better overall performance, which is being used in current corner detection operation. This provides stable corner points which accurately reflect the actual position of the corners in the frame, resulted in a reliable perspective transformation.

7.2 Attempts on IMU and Fusion Integration

Similar to vision pose estimation, different approaches are attempted to integrate IMU and sensor fusion into the localization system. Some IMU challenges and testing attempts will be discussed here.

1) **Testing with IMU Reading**

Before the IMU integration into Kalman filter, IMU was separately being tested to observed the result. Several testing were done before in order to examine its sensing performance. Among all, one test was being done with a motion as below. The response graph of the NS-EW (north south east west) axis was also being collected.

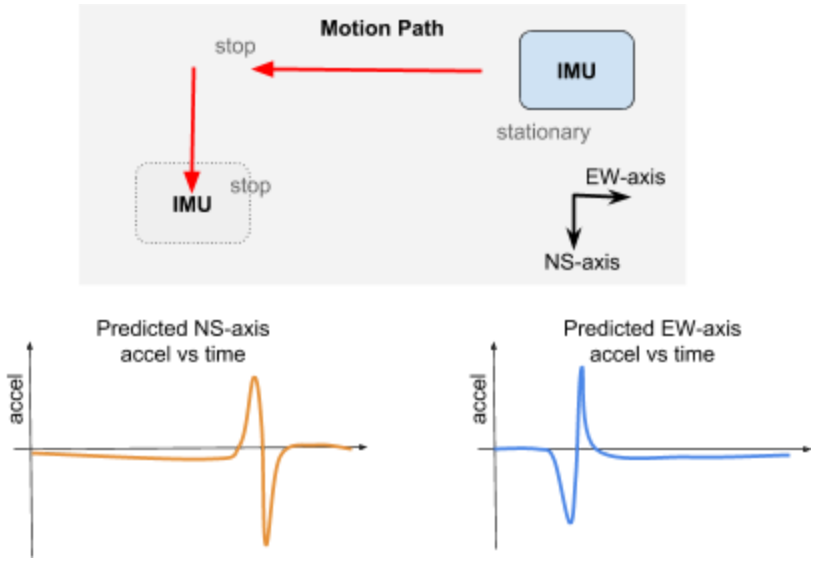


Figure 50: Experiment with Predicted IMU Response Graph

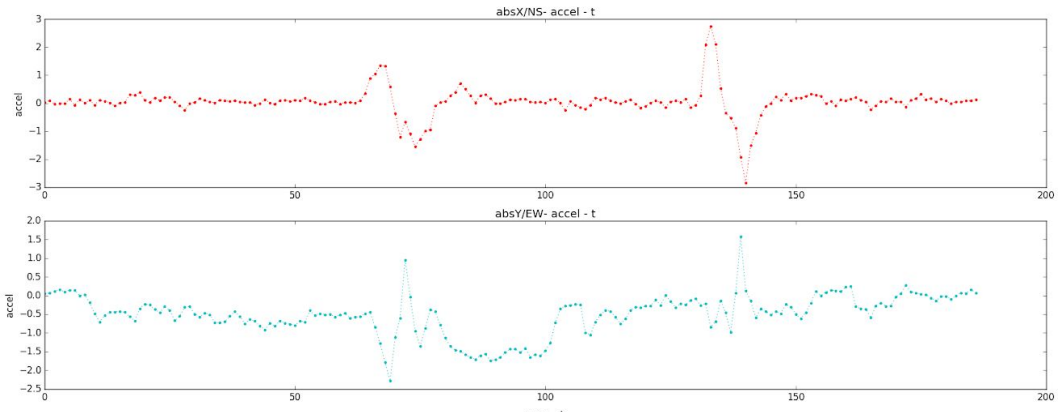


Figure 51: Actual Reading of IMU Acceleration with Motion Above

As shown above, the testing was conducted with a hand-held stabilizer. the actual acceleration graph was noisy in comparison to the predicted response graph. Although 3D matrix transformation was used to transform acceleration xyz axis to the NSEW earth axis, still the output reading is very noisy. This is because dynamic motion will create changes toward the IMU rotation matrix, resulted in interfered absolute acceleration values of the NS&EW axises. Thus the modified mobile platform that was being built is important to eliminate these unwanted noise.

Another IMU experiment was conducted on the modified mobile platform. A Diagram of a further complex motion test is also shown below. Continuous accelerations were exerted on the EW axis during the first half, the on the NS for the next half.

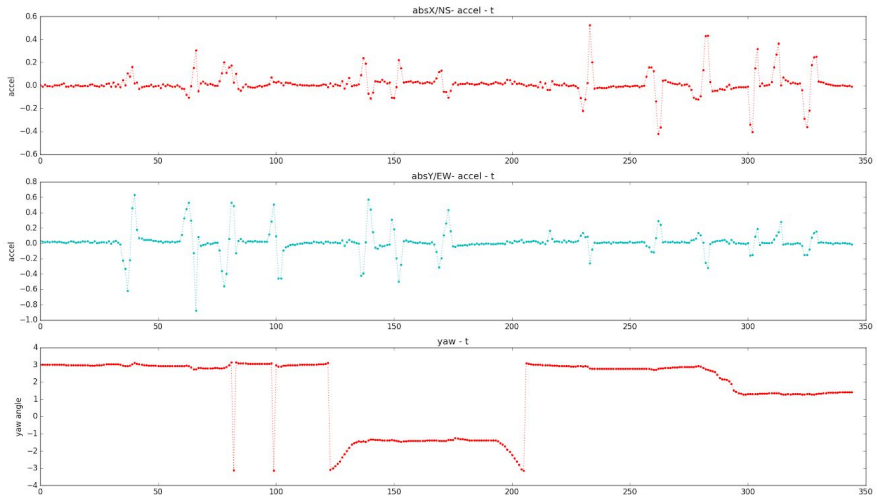


Figure 52: Showing of IMU Acceleration and Yaw Reading in a Complex Motion: Moving in NSEW axial direction. Acceleration value was normalized from -1 to 1

It can be observed that true acceleration on EW will still interfere with the other axis, and vice versa. As such, it can be concluded that acceleration reading is noisy for relatively small distance testing.

2) Severe drifting of IMU Odometry reading

After using the IMU, observation of severe drifting was being noticed. The displacement error drift can be severe, with a drift of 1 meter around 8 seconds. The presence of gravity with a acceleration of $9.8ms^{-2}$ is hugely greater than the acceleration of our robot. Thus, this drifting is severe in relative while also having the contribution of gravity. The simulated drift is shown here:

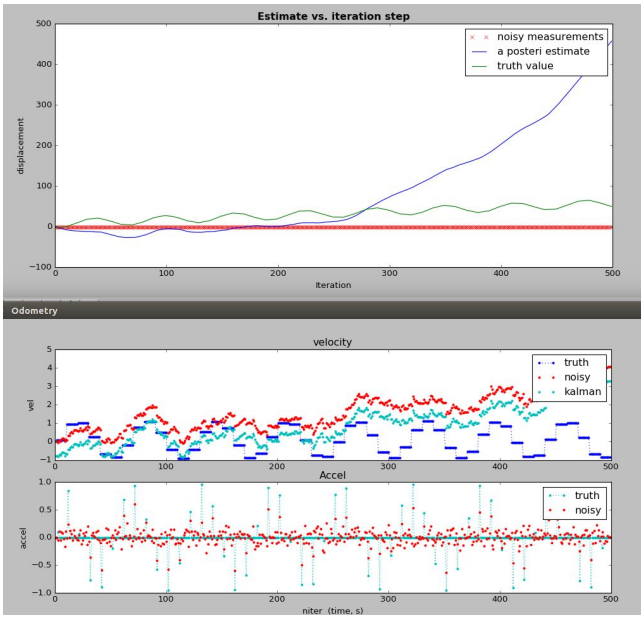


Figure 53: Drifting effect of IMU without frequent measurement update from vision input

7.3 Improvements on Overall Localization System

1) Change on Testing Hardware Setup

The handheld stabilizer, equipped with a camera and IMU was first used to emulate a mobile robot. Upon completion of the localization system, both IMU and camera will be statically mounted on the robot.

As stated, only 3 output datas (x, y, yaw) will be deployed, while roll and pitch should be omitted. Henceforth, this hand-held 2-axis stabilizer/gimbal was fabricated to perform the task.

Figure 54: Fabricated 2-axis Gimbal with Mounted Camera and IMU

However, mechanical based stabilizer still unable to fully remove the roll and pitch angle during testing. Although the IMU acceleration reading was transformed from the IMU frame to earth magnetic north frame (multiple a 3D rotation matrix), still the IMU is highly sensitive and noisy to the environment. Therefore, a much stable setup was used to conduct the experiment. In this case, a mobile platform.

2) Restriction on Operating Area of Vision-based Localization (BlindSpot)

The operational area of the camera to a marker in a environment was tested. The effective operating range of the current vision pose estimation system has a maximum of 0.18cm to 120cm. The restriction is largely due to the logitech webcam 1.3MP resolution and field of view (58°). Also, the tiled viewing angle from the sides are $\pm 58°$. One noticeable observation is, the further away the camera is located from the marker, the less accurate the output value. This resulted in a noisy pose estimation output during longer distance sensing.

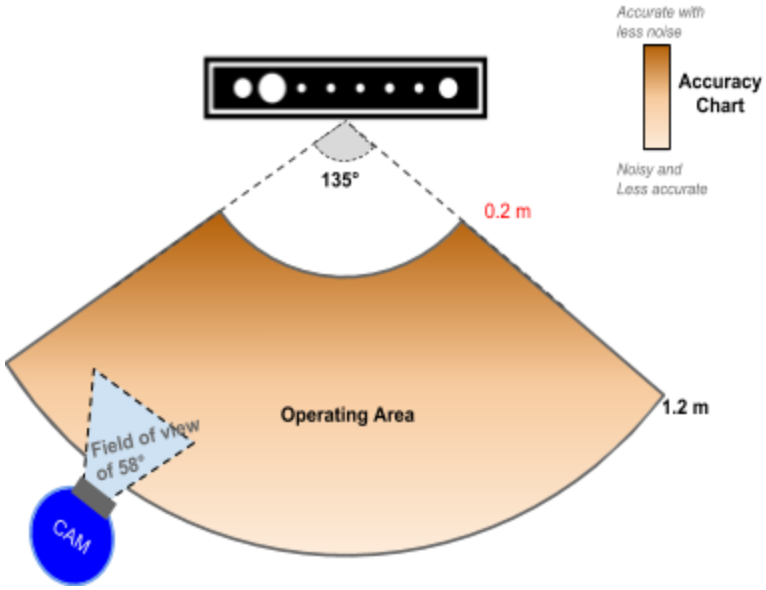


Figure 55: Operating Area of Vision-based VITag Detection

To effectively overcome this "blindspot" problem, incorporation of different sensor odometry was considered to improve this.

3) From Vision based to vision, encoder, and IMU based localization

The implementation of this indoor localization process underwent 3 phases of execution. Continuous improvements were constantly being made to create a much reliable localization system.

PHASE 1: *Camera*	- Purely based on vision pose estimation. - Localization is being restricted by camera viewing angle and marker's position relative to camera - non-continuous localization, discrete sample with low sampling rate
PHASE 2: *+ IMU*	- Odometry was being considered to solve the vision's limitations - Incorporated IMU odometry and kalman filter for sensor fusion - IMU acceleration odometry value was too noisy, error in accuracy is larger than desired accuracy
PHASE 3:	- Set up a proper mobile platform with 2 encoders installed

+ Encoder	- Localization is based on fusion of much stable encoder odometry and vision pose estimation - Camera pose estimation rotation matrix based on IMU rotation matrix (remove vision pose estimation deviation) - IMU heading used in encoder and vision nodes. - Most robust, stable and accurate localization performance - within accuracy of $\pm 15cm$ while markers were 2 m - 2.5 m apart

Table 4: Three phases of Implementation Process

Overall along the implementation phases, the complication of the localization system increases along with the improvement of the localization performance.

Generally speaking, sensor fusion of the 3 sensor inputs (camera, IMU and encoders) enables a smoother odometry output. Main localization problems namely vision blindspot, deviation of vision pose estimation output and nosiness of IMU accelerometer were solved. With a combination of discrete vision input and high frequency IMU & encoder inputs, the final localization system is able to perform well in reflecting the actual pose of the device.

Of course, more experimental works with different scenarios can be done to test out the performance of the final localization system. Due to limited timeframe, this report will only show the feasibility and performance of the final localization system, with the support of testing figures and videos.

8. CONCLUSION AND FUTURE WORK

In a nutshell, vision and odometry localization provides great pose estimation accuracy with low cost. According to the conducted experiment, the developed tri-sensors localization system is able to obtain an accuracy of $\pm 15cm$ in an indoor environment. With sufficient number of VITag markers, kalman filter is able to assisted in maintaining an accurate localization result. This clearly shows the feasibility of the indoor localization solution with camera, imu and encoder.

As we move forward into the future, dependence towards automation and autonomous mobile robot will be trend in various manufacturing and service sectors. Localization plays an integral part in a robotics system, ensuring a highly capable robot in navigating an environment. In fact, vision and odometry localization that we discussed here is a less cost and highly flexible system, suitable in a wide varieties of localization use cases.

For further improvement, proper precise construction of the encoder needs to be achieve in order to obtain a much accurate result. Algorithm for the encoder odometry can be further improve to incorporate non-linear motion. Furthermore, the vision pose estimation process consumes lots of CPU processing power. Thus, there's a need to be further optimized the process to increase its sampling rate. Consequently, installation of multiple wide angle HD cameras could remove blindspot, which in turn improves the localization result.

In addition, codes can be further improved to one with greater robustness and redundancy. Lastly, real testing on the actual robot needs to be conducted. With all these, high precision yet low-cost localization can be achieved in a minimally set up indoor environment.

In conclusion, by implementing the discussed vision and odometry localization pipeline, this will significantly improve the current robot capability with less effort. This is highly effective for the current indoor localization precision. Its able to achieve the goal of complete it's picking task along corridors of shelves in given error range.

9. REFERENCES

[1] A. Causo, Z.-H. Chong, R. Luxman, and I.-M. Chen, "Visual marker-guided mobile robot solution for automated item picking in a warehouse," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017.

[2] C. Siagian, C. K. Chang, and L. Itti, "Autonomous Mobile Robot Localization and Navigation Using a Hierarchical Map Representation Primarily Guided by Vision," *Journal of Field Robotics*, vol. 31, no. 3, pp. 408–440, 2014.

[3] D. K. T. A. S. F. F. Le Gland INRIA France Liyun He INRIA France Yann Oster Thales Alenia Space France, "Multi-Sensor Fusion for Localization Concept and Simulation Results." [Online]. Available: https://www.irisa.fr/aspi/legland/pub/ion-gnss09.pdf.

[4] T. Bonkenburg, "Robotics in Logistics; A DPDHL perspective on implications and use case for the logistics industry," DHL, Mar. 2016.

[5] "Robots in Logistics and Transportation," *Robotics Online*. [Online]. Available: https://www.robotics.org/blog-article.cfm/Robots-in-Logistics-and-Transportation/43. [Accessed: 28-Apr-2018].

[6] S. Isaac, "Sensing and Perception: Localization and positioning," *WASP Sweden Graduate School - Autonomous System*, 2016. [Online]. Available: http://wasp-sweden.org/custom/uploads/2016/10/michael-felsberg-overview-20161005.pdf.

[7] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer, "Mapping and localization from planar markers," *Pattern Recognit.*, vol. 73, pp. 158–171, 2018.

[8] "Documentation - ROS Wiki." [Online]. Available: http://wiki.ros.org. [Accessed: 28-Apr-2018].

[9] "OpenCV library." [Online]. Available: https://opencv.org/. [Accessed: 28-Apr-2018].

[10] "rviz - ROS Wiki." [Online]. Available: http://wiki.ros.org/rviz. [Accessed: 28-Apr-2018].

[11] "Matplotlib: Python plotting — Matplotlib 2.2.2 documentation." [Online]. Available: https://matplotlib.org/. [Accessed: 28-Apr-2018].

[12] Z. Farid, R. Nordin, and M. Ismail, "Recent Advances in Wireless Indoor Localization Techniques and System," *Journal of Computer Networks and Communications*, vol. 2013, Sep. 2013.

[13] "OpenCV: Detection of ArUco Markers." [Online]. Available: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html. [Accessed: 28-Apr-2018].

[14] P. Rudol, M. Wzorek, and P. Doherty, "Vision-based pose estimation for autonomous indoor navigation of micro-scale Unmanned Aircraft Systems," in *2010 IEEE International Conference on Robotics and Automation*, 2010.

[15] M. B. Alatise and G. P. Hancke, "Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter," *Sensors* , vol. 17, no. 10, Sep. 2017.

[16] "ags.cs.uni-kl.de; Camera Model and Calibration Lecture Note Chp 3." [Online]. Available: https://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws11-12/3DCV_WS11-12_lec03.pdf. [Accessed: 28-Apr-2018].

[17] "DFK institution: 2D projective transformations(homographies) lecture note." [Online]. Available: https://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws11-12/3DCV_WS11-12_lec04.pdf. [Accessed: 28-Apr-2018].

[18] "Carleton University School of Computer Science - Homography Lecture Slides." [Online]. Available: http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf. [Accessed: 28-Apr-2018].

[19] "Inertial measurement unit - Wikipedia." [Online]. https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Accessed: 28-Apr-2018].

[20] R. Bieda and K. Jaskot, “Determinig of an object orientation in 3D space using direction cosine matrix and non-stationary Kalman filter,” *Arch. Contol. Sci.*, vol. 26, no. 2, 2016.
[21] “The Kalman Filter,” *MIT Tut Notes*. [Online] .http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf.
[22] “Kalman filter - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter. [Accessed: 28-Apr-2018].
[23] “Camera calibration With OpenCV — OpenCV 2.4.13.6 documentation.” [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html. [Accessed: 28-Apr-2018].
[24] “Canny Edge Detection — OpenCV 3.0.0-dev documentation.” [Online]. Available: https://docs.opencv.org/3.4.1/da/d22/tutorial_py_canny.html. [Accessed: 28-Apr-2018].
[25] “OpenCV: cv::SimpleBlobDetector Class Reference.” [Online]. Available: https://docs.opencv.org/3.3.1/d0/d7a/classcv_1_1SimpleBlobDetector.html. [Accessed: 28-Apr-2018].
[26] “Shi-Tomasi Corner Detector & Good Features to Track — OpenCV 3.0.0-dev documentation.” [Online].https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html. [Accessed: 28-Apr-2018].
[27] “Geometric Image Transformations — OpenCV 2.4.13.6 documentation.” [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html. [Accessed: 28-Apr-2018].
[28] “Camera Calibration and 3D Reconstruction — OpenCV 2.4.13.6 documentation.” [Online]. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Accessed: 28-Apr-2018].
[29] “Raspberry Pi - Teach, Learn, and Make with Raspberry Pi,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 28-Apr-2018].
[30] “Pololu - L3GD20 3-Axis Gyro Carrier with Voltage Regulator.” [Online]. Available: https://www.pololu.com/product/2125. [Accessed: 28-Apr-2018].
[31] Open Source IMU library, “RPi-Distro/RTIMULib,” *GitHub*. [Online]. Available: https://github.com/RPi-Distro/RTIMULib. [Accessed: 28-Apr-2018].
[32] “Encoder Model.” [Online]. Available: http://www.mouser.com/ds/2/307/e6a2-c_ds_e_7_1_csm490-597999.pdf. [Accessed: 23-May-2018].
[33] “tf - ROS Wiki.” [Online]. Available: http://wiki.ros.org/tf. [Accessed: 28-Apr-2018].
[34] “msg - ROS Wiki.” [Online]. Available: http://wiki.ros.org/msg. [Accessed: 28-Apr-2018].
[35] T. Babb, “How a Kalman filter works, in pictures | Bzarg.” [Online]. Available: http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures. [Accessed: 28-Apr-2018].
[36] “Geometric Image Transformations — OpenCV 2.4.13.6 documentation.” [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html. [Accessed: 28-Apr-2018].
[37] “OpenCV: Feature Detection.” [Online]. Available: https://docs.opencv.org/3.1.0/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d. [Accessed: 28-Apr-2018].
[38] “OpenCV: Contour Features.” [Online]. Available: https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html. [Accessed: 28-Apr-2018].
[39] “Harris Corner Detection — OpenCV 3.0.0-dev documentation.” [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html. [Accessed: 28-Apr-2018].

APPENDIX

Source Codes are on in GitHub: https://github.com/tanyouliang95/FYP_IndoorLocalization

Videos:

1) Pure Vision Localization
   https://drive.google.com/file/d/122Mb8sE_tPIuZuXWPEA8nGmZ_66KtOIZ/view?usp=sharing

2) Full Vision + encoder odometry and IMU heading localization
   https://drive.google.com/file/d/1eMS-X7TGw8viWN-jl2K27Q0o8giECmtx/view?usp=sharing

Google Drive:
https://drive.google.com/file/d/1eMS-X7TGw8viWN-jl2K27Q0o8giECmtx/view?usp=sharing