Tracking of a high precision robot

Abdullah Shahin[1] **Vinay Venkanagoud Patil**[2]

[1]sve15ash@student.lu.se [2]vi0507pa-s@student.lu.se

Abstract: The project is centered around the evaluation of sensor fusion with the help of a filtering algorithm to filter out noise from the process and the measurement of a high precision robot that is being built for MAX IV, the filtered position estimate is used to track the robot in the environment. The filter in focus is the extended Kalman filter (EKF). This is a continuation of the work done by the author, Abdullah Shahin and Vinay Venkanagoud Patil, the collaboration will use the dual robot (omnidirectional/delta) that has been built by Vinay and implement the filters that Abdullah worked on in the localization node, the main focus is to find the measurement model of the IMU and Encoders. Further, the project will work on the inverse kinematics of the robot, so that the control signal units correspond to the measurement units, the final result has been tested at MAX IV, the robot is equipped with 3 omni wheels and 3 actuators that has a delta configuration. The results of this project was a maximum radius error of 3.5mm, this is more than a 10 fold improvement in the precision of the robot as our most accurate sensor has a 4cm resolution.

1. Introduction

In this project, a fine-tracking mobile robot for high-precision positioning and localization is implemented and used at the beam-line laboratory MAX IV in Lund, Sweden. The requirement comprises of the positioning and marking of relatively exact points on concrete floor where the beam line equipment and machinery will be positioned. Currently, there are 16 funded beam-line experiments and 6 are being installed. All of the beam-line equipment has been placed manually by construction workers. This machinery needs to be placed very accurately since the radiated beam itself is highly sensitive to deviations along its path and will have a direct impact on the resolution this beam has. Therefore, a poor placement will interfere with the experiments and measurements. A computer based blue lining system is being used to map the construction area in order to help the engineers to place the aforementioned equipment. Thus, the task is repetitive and physically demanding for the workers due to the goal precision of ±60 microns they need to reach for each mark using the current equipment. Consider that there are a couple of hundred points to be marked. Another drawback is the time the worker spends to reach each point because of the natural inaccuracies of the human hand. Thus, it takes several tries to reach the position. It is so a highly accurate robotic system with advanced control techniques is required to accomplish this task more efficiently and in less time.

Previously, the authors (Patil V., Carrera L.) [2] have implemented a dual robotic system for accurate positioning consisting of a Delta-configuration robot over an omnidirectional mobile robot. This system features the coarse navigation (omnidirectional) and the fine positioning (Delta) in two consecutive stages, once the omnidirectional robot has reached a reasonable ±2 centimeters in radius from its target the Delta robot goes into action by fine positioning the end effector to the target with an estimate of ±300 microns of error from the actual target. Although the robot is capable of a considerable high accuracy on its positioning it does not perform at its full potential due to the control system employed. Alternatively to this work, the author (Shahin A.) [4] has developed the control software intended to solve the blue lining task using advance control techniques with outstanding localization and navigation features. The objective was to investigate, via simulation, an EKF (Extended Kalman Filter), an UKF (Unscented Kalman Filter) and a PF (Particle Filter) over a car-like steering vehicle and its corresponding state-space model. This work concluded that the filters have potential to navigate the MAX IV robot with accuracy, the simulation had both process and measurement noise applied to it. Therefore, and in order to take advantage of the full potential of the authors work, the authors have studied the adaptation of these filters on the omniwheel/delta dual robot and investigated with potential implementation in this dual robot for later testing on-site at MAX IV, as this robot will run the algorithm on a raspberry pi, the PF is computationally demanding, the UKF a more advanced version of the EKF and thus is more demanding on the group to implement. The objective is therefore to reach a fine positioning suitable enough to perform the blue lining process with great accuracy and cheap but reliable components with an EKF algorithm. The book [5] is used extensively during the previous project, that was investigated by (Shahin.A) and has been continued, with a real world robot with more sensors and actuators than the simulated robot, the chapters studied will mainly be chapter 1,2,3,7 and 8.

2. Modeling

The present project has a list of key components in which the functioning is based on. It is described from the hardware components to the subsystems they form. Also, the essential math background is described for each subsystem so the whole model can achieve the localization and navigation process.

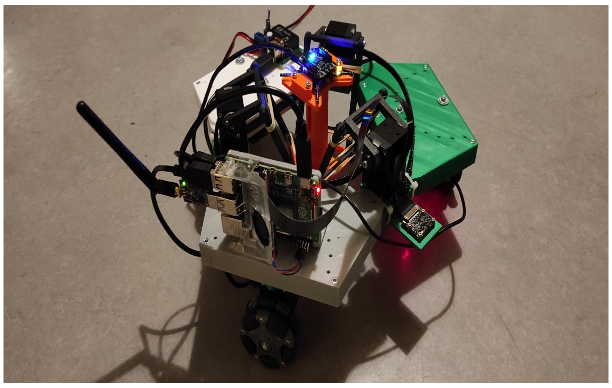

Figure 1. Dual robot (omnidirectional/delta) used in this project

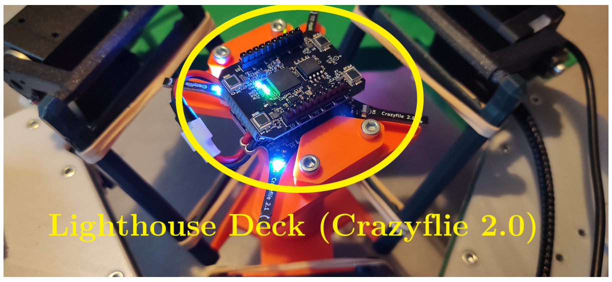

Figure 2. Crazyflie sensor (Lighthouse Deck)

IMU sensor

Since the dual robot uses relative position and rotation measurements, the considered sensor for this setup is the 6-DOF (Degrees Of Freedom) IMU (Inertial Measurement Unit) which is mounted on the center of the omnidirectional robot frame.
The crazyflie drone has been retained to compute the inertial measurements and use the sole measurements from the on-board accelerometer and gyroscope. This sensory data has been logged using the crazy-radio(RF interface to control and log data from the crazyflie drone). The sensory data logged from the crazyflie drone will consist of X and Y position using the lighthouse deck, acceleration along X and Y using the accelerometer. However, more accurate and reliable IMU's could be mounted on the robot in the future

Wheel Encoders Along with the IMU sensor, wheel encoders are used to extract wheel velocities which are in turn transformed to the robot body velocities using the forward kinematics of the robot. The encoder data is polled from the dynamixel motor [1] every instance a control signal is sent in. Additionally, the encoders on the motor provide us a resolution of 4096 ticks per revolution which corresponds to 0.00153radians.

Leica Absolute Tracker The Leica Absolute Tracker is a laser based device which uses a laser beam directly pointed to a reflector to estimate with metrology-grade accuracy the 3D-position, see Figure 3. In the dual robot, the Leica reflector has been installed in a position near the IMU sensor in order to obtain the position of the robot with respect to the Leica tracker. This enables the robot to know its initial position in the space and compute the trajectory to the target. The important role of the Leica Absolute tracker in this project is to provide us with absolute position in space during the initial calibration and also help measure the performance of the localization on arrival to the target.

Figure 3. Leica Absolute Tracker

Crazyflie Lighthouse deck

The lighthouse deck is one of many decks developed by Bitcraze. This deck is custom designed to acquire position data using the htc vive lighthouse base stations. The base stations used produce a light signal that sweeps in the horizontal and the vertical plane with a unique frequency which is quite similar to a conventional lighthouse we see at the sea shores. The flydeck is equipped with 4 mirrors that are photo sensitive that sense the lighthouse signals to compute its position from the lighthouse.

Omnidirectional Robot

The considered model is the omnidirectional setup shown in Figure 4.

An omnidirectional platform is used since it can perform translations in any direction without the need to reorient. Moreover, due to its symmetric construction, the omnidirectional platform can also rotate about its Z axis seamlessly. These features are achieved by using the resulting velocities of the omnidirectional wheels that are controlled dictated by its kinematics. The command to the robot is in form of $u_t = (V_x, V_y, \omega_z)$ which is the body velocities in X and Y, and the angular velocity along Z. Thus, the robot kinematics equation will correlate the command vector variables with the actual controllable variables in the omnidirectional platform i.e. the wheel velocity of each wheel $(\omega_1, \omega_2, \omega_3)$. This relation is derived from the inverse kinematic transformation of the robot.

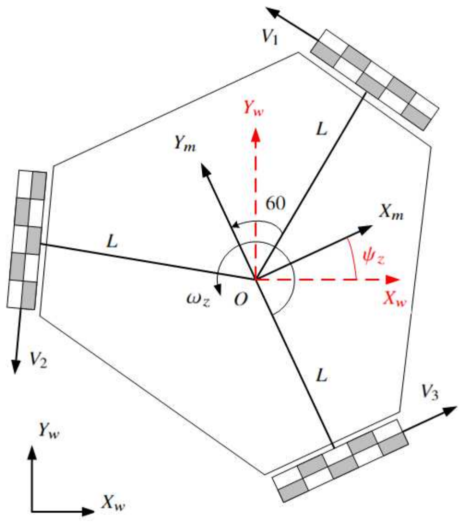

Figure 4. Model of the omnidirectional robot

2.1 State Space Model

First, the state transition model for this setup is described in Eq. 1

$$x_t = g(u_t, x_{t-1}) + v_t \tag{1}$$

Where, x_t is the state vector, the control vector u_t, v_t is a Gaussian process noise in the form of a Gaussian vector.

The state vector x_t is given by (Eq. 2),

$$x_t = (x_{world}, y_{world}, \psi_z)^\top \tag{2}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi}_z \end{bmatrix} = \begin{bmatrix} (u_1) * cos(\psi_z) - (u_2) * sin(\psi_z) \\ (u_1) * sin(\psi_z) + (u_2) * cos(\psi_z) \\ u_3 \end{bmatrix} \tag{3}$$

Where x and y are the positions in the world coordinates, u_1(linear velocity along X), u_2(linear velocity along Y) and u_3(angluar velocity along Z) are the control velocities, ψ_z and ω_z are the angular position and angular velocity respectively.

The control vector is given by (Eq. 4),

$$u_t = (V_{xref}, V_{yref}, \omega_{zref})^\top \tag{4}$$

Where u_t is the control signal containing the body velocities in the x,y and ψ direction respectively. This is used to predict the estimated state in EKF algorithm.

Measurement Model

From Eq. 5 and the state vector in Eq. 2 the measurement model is then given by the vector,

And the mobile robot measurements,

$$y_t = h(x_t) + e_t \tag{5}$$

Where e_t is a Gaussian measurement noise in Eq. 1 and Eq. 5 accordingly.

The y_t is the measurement vector containing the measurements form the IMU sensor, Light house position data and the encoder data, the IMU sensor outputs data from the accelerometor which is the measure of a_x, a_y this data has been fused with the Light house data which outputs the position for X_{world}, Y_{world} and ψ_{body} and the encoder data V_x,V_y and w_z to correct the estimated state from the control signal. This has been done in the final step of the EKF algorithm called the correction step.

$$\begin{bmatrix} y_t \end{bmatrix} = \begin{bmatrix} x \\ y \\ \psi_z \\ V_x \\ V_y \\ \omega_z \\ a_x \\ a_y \end{bmatrix} \tag{6}$$

3. Extended Kalman filter

As it was mentioned before, an EKF (Extended Kalman Filter) has been employed to preform the localization of this robot. Such algorithm relies on two steps sequence: **Prediction step** and the **Correction step**. The updates for the state and the covariances are performed in Eq. 1 and 7 respectively,

$$P_{t+1|t} = FP_{t|t}F^\top + Q \tag{7}$$

Where $\hat{x}_{t|k}$ is intended to be the estimate of x at time t given the control signal up to time k. Now, when the state space model is linearized to F the covariance update is possible. Further, the Kalman filter equations for the measurement update is seen in 8,

$$K_t = P_{t|t-1}H_t^\top (H_t P_{t|t-1} H_t^\top + Q_e)^{-1} \tag{8}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - h(\hat{x}_{t|t-1})) \tag{9}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{10}$$

3.1 Predict for EKF

The prediction for the EKF is preformed by equation 1 and 7.However, the prediction in this project differs from the previous in the following way, we have a more states in our state space model, where our control signal is updating the position states and the orientation states. The acceleration states are updated using the accelerometer readings. Thus, there are in total 5 sates as can be seen in equation 2. To predict the covariance matrix it is then required to linearizer the state space model with a Taylor series expansion, this creates a 5x5 matrix which is the Jacobian matrix, the Jacobian matrix has been used to predict the covariance of the system.

3.2 Correct for EKF

After the prediction is complete, the correction of the predicted state is facilitated from the measurements on the robot, the measurements are two IMU readings i.e. the accelerations along X axis and the Y axis, the position data from the HTC vive lighthouse deck and three encoder readings, by taking the difference in the innovation seen in equation 9, the Kalman gain calculated will help to weigh the different measurements form the sensors and the final estimated state is reached, this is

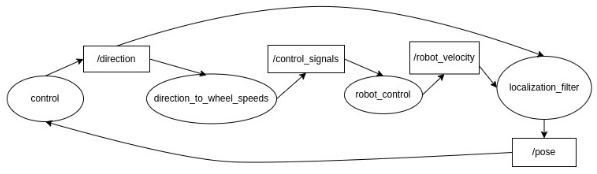

Figure 5. ROS graph

where the 'sensor fusion' is completed. The corrected estimate is then sent to the controller as input. for the next iteration the correction of the covariance matrix is preformed and is fed back to the predicted step along with the new control signal and the corrected estimated state, the cycle continues until the final desired position is achieved.

4. Implementation

4.1 ROS

The software for the robot to control and estimate its position is done using ROS. ROS that stands for Robot Operating system is an open-source robotics middle-ware. Although it is not an actual operating system but a collection of software frameworks for robot software development. It provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. The operating processes in ROS can be represented using a ROS graph where the operations happen in the nodes that may receive, send or multiplex sensor data and other messages such as control signals and set points. The other features that ROS provides is suite of debugging tools that enable us to plot data from the ROS topics in real time and check for inconsistencies. In this project ROS is used as framework for multiple python nodes that perform different tasks within the robot application. The ROS graph in the following figure 5 is the ROS graph that shows all the nodes and how they communicate with each other and the hardware using the respective ROS topics.[3]

4.2 Message passing

Since the body velocity obtained by the wheel velocities is one of the sensory data used in the EKF, we need to extract wheel speeds from the motors at every iteration. The ROS-node we initially used didn't accommodate for timing and this led to the robot loosing its control over the motors every time we read the sensor data. This was a result of the bus used to read and write to the motor being a shared variable between the reading and the writing functions. To fix this conflict, we introduced a ROS service which whenever invoked would obtain a lock over the bus to read the sensor data and release it whenever it is not using it thus removing the conflict between the 2 processes trying to access the bus.

4.3 Control signal

From the above section we can see that the control node in the ROS system, subscribes to the $/pose$ and the $/setpoint$ topics. It computes the errors along X, Y and ψ. It then runs the error through a PI controller and computes the corresponding correction body velocities that are fed to the robot as control signals in the form (V_x,V_y,ω_z). This body velocity is then converted to wheel velocities and then to motor PWMs in the direction_to_wheel_speeds, and the control_signals nodes respectively.

4.4 Kinematics

As we are using a 3-wheel omnidirectional setup, we need an inverse kinematic model to transform the linear and angular velocities of the body to wheel speeds. In our case, d is the distance of each wheel from the centre of the robot, r is the radius of the wheel and in a general case, angle α_i is the angle between the axes of the wheels. The angle θ is the angle of the first wheel from the X axis of the robot body frame. In our robot, the values of θ, α_1, α_2 and α_3 are 30 deg,0 deg, 120 deg, and 240 deg respectively. The corresponding transformation can be represented using the following matrix.

$$R_{IK} = \begin{pmatrix} -sin(\theta) & cos(\theta) & d \\ -sin(\theta+\alpha_2) & cos(\theta+\alpha_2) & d \\ -sin(\theta+\alpha_3) & cos(\theta+\alpha_3) & d \end{pmatrix} \tag{11}$$

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} = R_{IK} \cdot \begin{pmatrix} V_x \\ V_y \\ \omega_z \end{pmatrix} \tag{12}$$

This matrix was used in the direction_to_wheel_speeds node to convert the robot velocity from the control node to wheel velocities that is further used to move the robot in the desired way.

Additionally, a rotation matrix (Eq. 13) is applied to the motion body frame to fully describe the velocities V_x, V_y and angular velocity ω_z from the wheel velocities V_1, V_2 and V_3 (Eq. 14). Due to the symmetric nature of the robot, the forward kinematics can be computed by directly inverting the Inverse Kinematic transformation from (Eq. 11).

$$R_{FK} = R_{IK}^{-1} \tag{13}$$

$$\begin{pmatrix} V_x \\ V_y \\ \omega_z \end{pmatrix} = R_{FK} \cdot \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \tag{14}$$

This matrix was used to compute the body velocities from the wheel velocities in the robot_control node which is further used in the localization_filter.

4.5 Calibration of encoder measurements

The measurements from the encoders had to be calibrated for the controller, as the EKF algorithm is running at 10000 Hz the encoder measurements had to be scaled to compensate for the frequent update of the EKF. Thus, at first the encoder values where too small in the algorithm, the controller would run the control commands more frequently than necessary, the encoder measurement where multiplied by a thousand and a destination of 10cm was given to the robot, the robot would reach around 3.1cm which was then compensated for by dividing the 1000 by 3.19 to reach the desired scaling factor which would reach the 10cm mark with a ±1mm error.

4.6 Tilt Compensation

Due to the unevenness of the ground, the robot would have a tilt error form the IMU reading, that would result in the robot control to incorrectly compensate for this tilt. This is implemented to remove any effect of the acceleration due to gravity appearing on the other axes namely x and y. In an ideal scenario, when the robot is stationary on a completely flat surface, the acceleration across z should be -g and 0 across x and y axes. In order to achieve tilt compensation, we acquire the accelerometer data along x,y and z. We then compute the orientations of the IMU in the X-Z plane and the Y-Z plane. Upon obtaining these angles, we know that the accelerometer orientations in the space. We then project the raw values onto the corrected axes. This is well described in the following equation 15-18.

$$\phi_x = tan^{-1}(acc_x^{raw}/acc_z^{raw}) \tag{15}$$

$$\phi_y = tan^{-1}(acc_y^{raw}/acc_z^{raw}) \tag{16}$$

$$acc_x^{compensated} = acc_x^{raw} \cdot cos(\phi_x) \tag{17}$$

$$acc_y^{compensated} = acc_y^{raw} \cdot cos(\phi_y) \tag{18}$$

4.7 Path planning

An algorithm was implemented to automate the motion of the robot in a continuous loop of points chosen. This algorithm keeps track of the error i.e. the euclidean distance between the current position and the destination. When the error is smaller than the resolution of the robot, the next destination is automatically selected and the robot starts moving to the next position. The path planned is a square with the sides of length 10 cm.

4.8 Extended Kalman filter

The pseudo algorithm seen in algorithm 1 represents the EKF algorithm.

Algorithm 1: Extended Kalman Filter Algorithm

```
while currentSimulationTime < simulationTime do
    Calculate the state estimate with the control commands from the previous time step;
    Calculate the error covariance with the Jacobian of the model;
    Calculate the measurement residual;
    Calculate the Kalman gain;
    Correct the state estimate with the Kalman gain;
    Correct the error covariance;
end
```

As in the previous project the group relied on the help of [5, pp. 203–220]. for the completion of the EKF algorithm, as well as the experience the group members gained from the previous projects.

Predict algorithm Equation 21 shows the Jacobian matrix of the state space, the Jacobian is required to predict the covariance matrix.

$$a = -dt * ((u_1) * sin(\psi_z) + (u_2) * cos(\psi_z)) \tag{19}$$

$$b = dt * ((u_1) * cos(\psi_z) - (u_2) * sin(\psi_z)) \tag{20}$$

$$F = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \tag{21}$$

Correct algorithm As our observation model needs to be weighed for the correction step its Jacobian is shown in equation 22, as previously mentioned the output of the update step is again fed into the Predict State and the cycle goes on until the final position is estimated with in a satisfactory frame.

$$H_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \\ 0.5 * (dt * dt) & 0 & 0 \\ 0 & 0.5 * (dt * dt) & 0 \end{pmatrix} \tag{22}$$

5. Results

The EKF algorithm with sensor fusion from the three encoders is finalized and have the same units as the control signal values, the IMU units are adjusted to the control signals units as well as the data from the HTC VIVE light house. As well as the re-tuning of the PI controller to accommodate the new control signal units. The ROS network and the inverse kinematics have been adjusted to the project, debugged and are running as expected. The MSE error for the localization was computed and the values stayed between $6.53 * 10^{-8}$ to $7.4 * 10^{-8}$ along X and $5.23 * 10^{-8}$ to $5.59 * 10^{-8}$ along Y when run for 15 minutes in a square trajectory.

5.1 Repeatability and Reliability

From the path planning node the robot had an automated loop that moved in a square with sides of length 10cm, at every corner of the square a point is made, this test is done to visually see the error distribution as the robot moves along the square in an infinite loop, the position estimate of the EKF as well as the measured light house measurements are plot and can be seen in figure 6 where the units for the x and y axis are in meters, the EKF position estimate is in red and the light house position measurement is in blue. Further, the light house measurement was blocked to test the effect of the light house measurement, the results of that test can be seen in figure 7, the light house measurement is the dominant measurement in the sensor fusion done by the EKF, the light house was blocked in the right top corner of the square the robot then moved around 17cm in the x direction 7.4cm in the y direction, the EKF flowed the Light house. In figure 8 the markings on the floor have a error radius distribution of maximum of 3.5mm

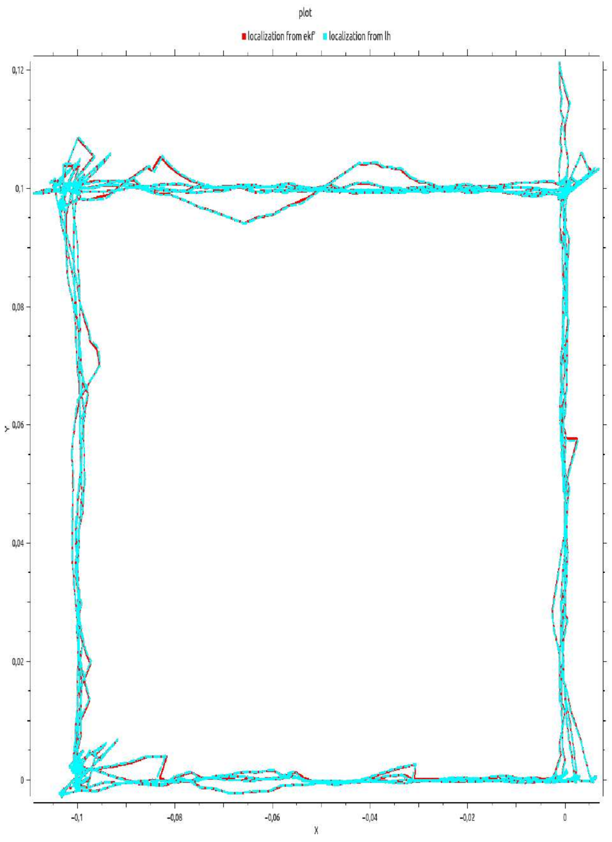

Figure 6. EKF estimate VS Light house measurement

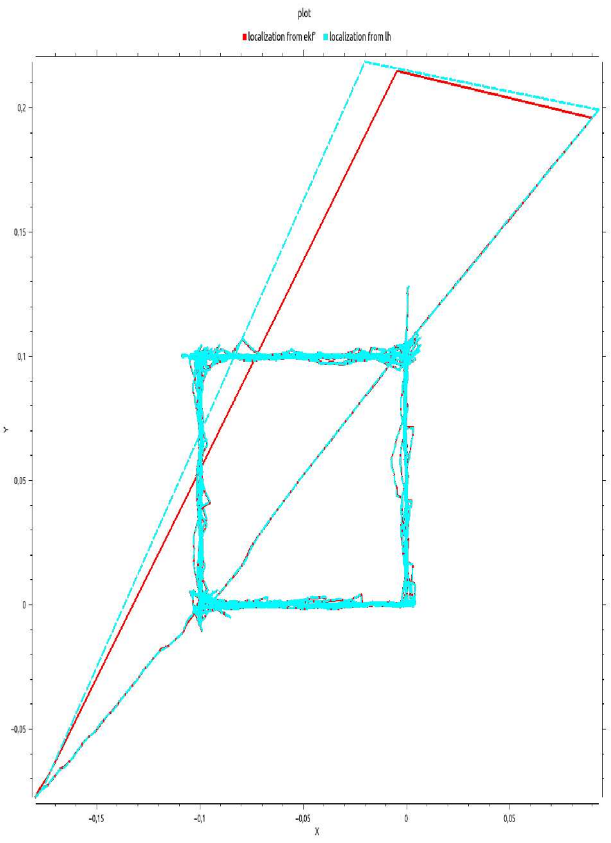

Figure 7. EKF estimate VS Light house measurement blocked

this is measured after the robot has moved clockwise and the over shoots are mostly seen on the y axis, the robot is then moved anti clock wise and a different corner have the largest error radius to be 3mm as seen in figure 9. In figure 10 and 11 where the robot reaches the 10cm destination with an error radius of around 2.5mm for both the x axis and the y axis respectively.

6. Discussion

6.1 Results

The testing on the 8th of December at MAX IV, resulted in acknowledging that more work needed to be done on the network for the message passing to the motors. It was also observed that the environment was dusty where the robot would be working, thus error would arise from the encoder measurements. Further the leica laser could not be relied on for measurement values as jitters where present in the leica laser measurements, the jitters are due to the fact that the leica laser is not capable to update the measurements of a moving target as well as it does for a stationary one, thus different covariances was tuned in the EKF for when the robot is moving and when it is stationary. As the robot should trust the IMU measurements more when the robot is moving and trust the encoders as well as the leica laser more when stationary, the covariance matrix must be tuned accordingly. As every sensor has disadvantages and advantages when in operation the sensor fusion is not only beneficial for the operation of the system but necessary. The testing of this EKF design could not be done in due time,

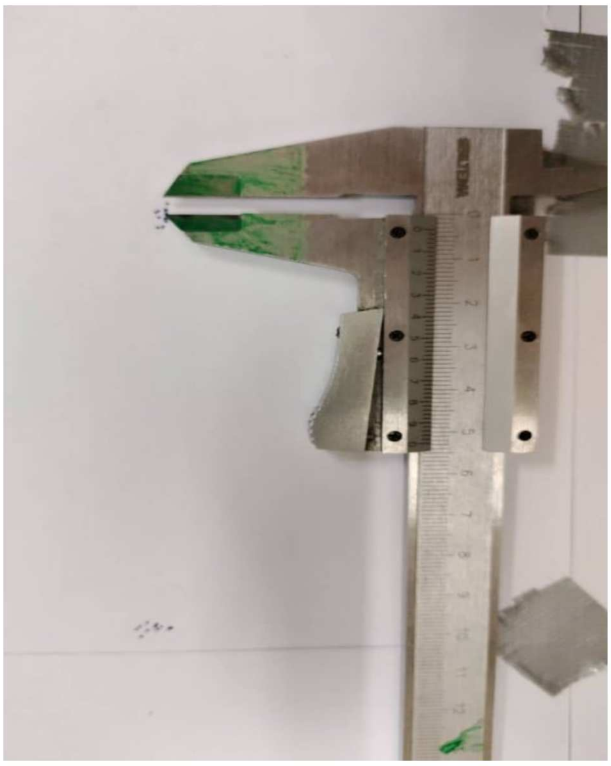

Figure 8. 3.5mm radius error distribution when running the robot clockwise

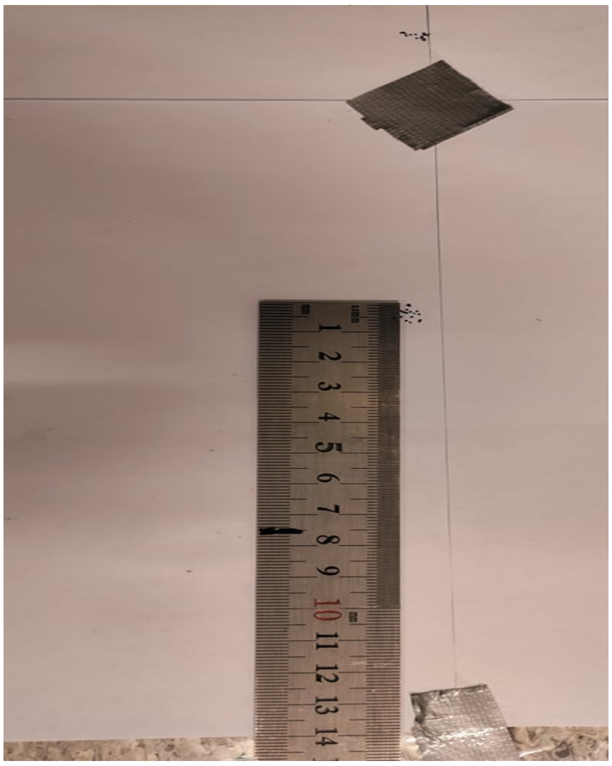

Figure 9. 3mm radius error distribution when running the robot clockwise and anti clock wise

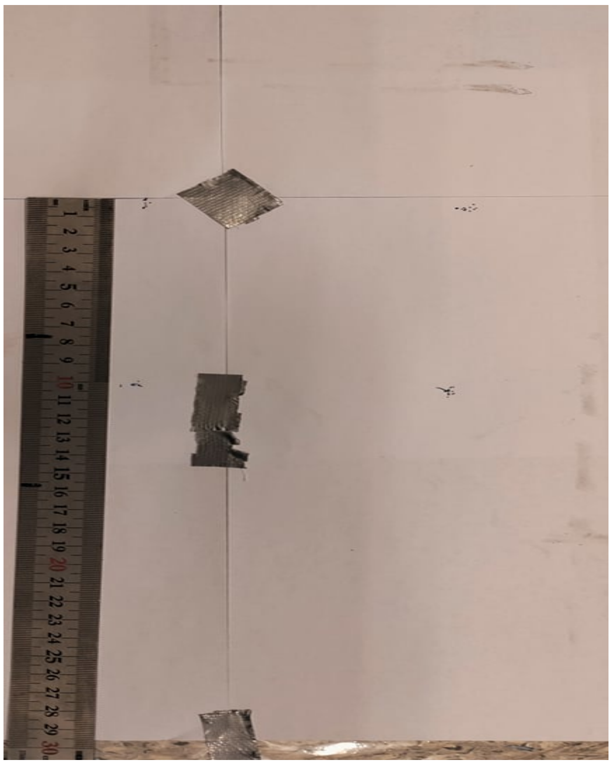

Figure 11. Vertical total distance and error when running clock wise and anti clockwise

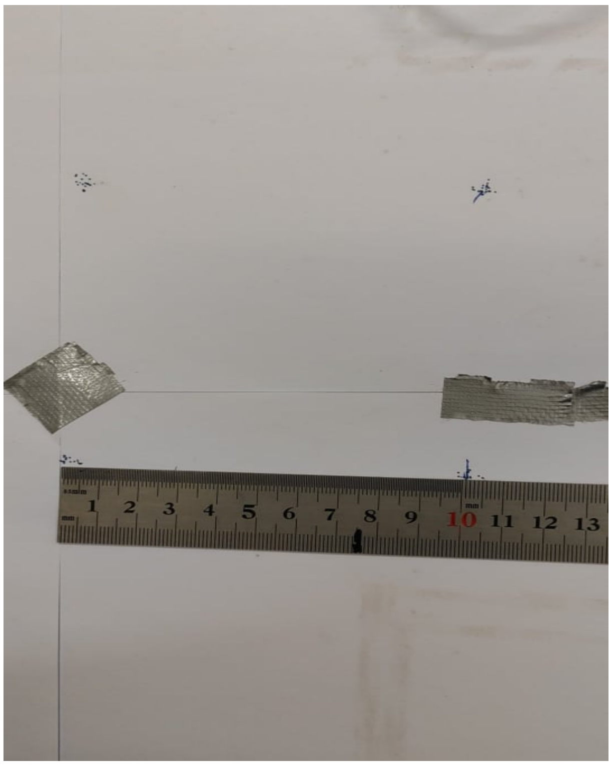

Figure 10. horizontal total distance and error when running clock wise and anti clockwise

as the measurement engineer Alina did not have a possibility to meet with us during the holidays. Thus, the team replaced the Leica measurements with the Light House measurement, the light house measurements are less accurate with 4cm but is more reliable when it comes to moving target, as seen in 5.1 the resolution of the position estimate relative to the resolution of the Light house measurement resulted in a 10 folds improvement form 4cm to around 3mm. The final set up of the algorithm and results were presented on the 14th of January to the faculty members and students at LTH.

6.2 Project dynamics

The dynamics of the group has been under pressure, as one of our group members decided to drop the course. Further, the testing of the robot was scheduled to be on the 8th of December, thus the group was required to be 4 weeks ahead of schedule, this resulted in long night (the longest night lasted until 5 in the morning) in the lab to get everything debugged and working. Results of the work were investigated on the 8th and reported soon after. Finally, the group missed one week of supervisor help due to the robot lab week, which resulted in the investigation of quaternion which was not necessary to the project, the python package built by Anders Blomdell that is responsible for controlling Dynamixel motors did not account for negative values in the register. Thus, some hours were spent to understand and process the negative values from the Dynamixel motors. Due to the circumstances and the nature of a project was proposed by the authors. Resulted in the authors having to be more self reliant, compared to other projects.

6.3 Project outcome

The project was a challenging endeavour for the time assigned to it, many over time hours were invested into the project for the final results, the majority of the time was investigating and debugging the code implementation for better results. As the project has not been done before many unknown unknowns where realised as the project developed, this project contains information form the majority of the courses taken previously by the authors and some courses that haven't been taken at all. The investigation in this project will further develop in the authors thesis work, that will be worked on in the spring.

References

[1] *DynamixelSDK*. https://github.com/ROBOTIS-GIT/DynamixelSDK. Accessed: 2021-12-1.

[2] V. V. Patil and L. D. Carrera. *OmniWheel Robot*. The Faculty of Engineering at Lund University, 2021-06-01.

[3] *ROS*. https://ros.org. Accessed: 2021-12-1.

[4] A. Shahin. *Tracking of a high precision robot*. The Faculty of Engineering at Lund University, 2021-06-01.

[5] S. Thrun. *Probabilistic robotics*. Vol. 45. 3. ACM New York, NY, USA, 2002.