86805: Software Architectures for Robotics
Localization system for a wheeled humanoid robot

Date: Wed 23:55, 01.02.16

Prof. F. Mastrogiovanni, PhD C. Recchiuto

Rabbia Asghar, BEng, Ernest Skrzypczyk, BSc

Contents

Rollo - Humanoid robot

Odometry

Odometry is one of the most widely used technique for determining the position of a mobile robot [1] [2] . It mainly involves use of various encoders, for example on wheels, as sensors to estimate the robot's position relative to a starting or previous location. Usually, it is used for real-time positioning in the between the periodic absolute position measurements, for example GPS (Global Positioning System) provides absolute position feedback, however it updates at $0.1 \div 1s$ interval, during which odometry could be used for localization. One of the major downsides of odometry is its sensitivity to errors. There are various error sources discussed in section Odometry errors on page 5 in detail. One significant source of error influencing the accuracy of odometry that is worth mentioning however, is the integration of velocity measurements over time to give position estimates.

First the odometry based physical motion model for the robot will be derived. The model is derived based on the following important assumptions:

1. The robot is a rigid body
2. The model represents a differential drive robot
3. The wheels of the robot are perfect discs and have the same diameters
4. The wheels have no thickness
5. There is no slip in the wheels
6. Both wheels are turning in the forward direction

A differential drive robot runs straight when the linear speed of both the left and right wheel is same. If the speed of one wheel is greater than the other, the robot runs in an arc. This derivation can be divided in three distinct cases of robot motion:

1. Clockwise direction
2. Counterclockwise direction
3. Straight line

Clockwise direction

First the case is considered, when the speed of left wheel is greater than the right and the robot will run in clockwise direction. Both right and left wheel will rotate around the same center of a circle.

P_i represents the initial position of the robot defined by the center of the line joining two wheels, while l represents wheel base, ergo distance between or axle length of two wheels. S_L and S_R represent the distance travelled by left and right wheel respectively. $\Delta\Theta$ represents the angle of travel for both the wheels and r is the radius of travel from the center of robot. With encoder feedback from left and right wheel, S_L and S_R can simply be computed using equations (1) and (2)

$$S_L = \frac{n_L}{60} 2\pi r_L t \tag{1}$$

$$S_R = \frac{n_R}{60} 2\pi r_R t \tag{2}$$

[1] UMBmark – A Method for Measuring, Comparing, and Correcting Odometry Errors in Mobile Robots
[2] Comparative Study of Localization Techniques for Mobile Robots based on Indirect Kalman Filter

where n_L and n_R are the revolutions per minute ($[rpm]$) of left and right wheel, r_L and r_R are the radii of the two wheels and t is time step in seconds.

S_L and S_R can be related to r and $\Delta\Theta$ using formulas (3) and (4)

$$S_L = (r + \frac{l}{2})\Delta\Theta \tag{3}$$

$$S_R = (r - \frac{l}{2})\Delta\Theta \tag{4}$$

The equations (3) and (4) above can be solved simultaneously to compute $\Delta\Theta$ defined in (5).

$$\Delta\Theta = \frac{S_L - S_R}{l} \tag{5}$$

The length between initial position P_i and final position P_f can be approximated by taking average of S_L and S_R .

$$\Delta S = \frac{S_L + S_R}{2} \tag{6}$$

Final position of the robot and its orientation can then be derived using basic trigonometry and geometry relations as displayed in equation (7). In general this length is approximated to ΔS for a very small step of time.

$$P_f = \left[P_{ix} + \Delta S(\cos(\Theta_i - \tfrac{\Delta\Theta}{2})), \quad P_{iy} + \Delta S(\sin(\Theta_i - \tfrac{\Delta\Theta}{2}))\right] \tag{7}$$

$$\Theta_\mathrm{f} = \Theta_\mathrm{i} - \Delta\Theta$$

where Θ_f and Θ_i are the final and initial orientation respectively.

Counterclockwise direction

Similar derivation of the robot can be derived for counterclockwise rotation, when the revolutions per minute of right wheel are higher than that of the left wheel. $\Delta\Theta$ can be computed as shown in (8)

$$\Delta\Theta = \frac{S_R - S_L}{l} \tag{8}$$

The relations for final position and rotation are as follows (9).

$$P_f = \left[P_{ix} + \Delta S(\cos(\Theta_i + \tfrac{\Delta\Theta}{2})), \quad P_{iy} + \Delta S(\sin(\Theta_i + \tfrac{\Delta\Theta}{2}))\right] \tag{9}$$

$$\Theta_\mathrm{f} = \Theta_\mathrm{i} + \Delta\Theta$$

where Θ_f and Θ_i are the final and initial orientation respectively.

Straight line motion

In case of straight line motion, the orientation of the robot stays the same, therefore $\Delta\Theta$ in this case is 0 and S_L is the same as S_R. The relations for final position and rotation are as follows:

$$P_f = \left[P_{ix} + S_{L=R}\cos(\Theta_i), \quad P_{iy} + S_{L=R}\sin(\Theta_i)\right] \tag{10}$$

$$\Theta_\mathrm{f} = \Theta_\mathrm{i}$$

Final set of equations

With the derivation of formulas understood for all three cases, for the ease of computation it is desirable to have the same set of equations in all cases.

The equations selected for this model are as follows.

$$\Delta\Theta = \frac{S_L - S_R}{l} \tag{11}$$

$$\Delta S = \frac{S_L + S_R}{2} \tag{12}$$

$$P_f = \left[P_{ix} + \Delta S(\cos(\Theta_i - \tfrac{\Delta\Theta}{2})), \quad P_{iy} + \Delta S(\sin(\Theta_i - \tfrac{\Delta\Theta}{2}))\right] \tag{13}$$

$$\Theta_\mathrm{f} = \Theta_\mathrm{i} - \Delta\Theta$$

Adaptation of odometry model for Rollo

During the project realization the encoders feedback was unavailable in the robot and a physical odometry model could not have been implemented. However, an attempt has been made to implement its modified version. In absence of encoders feedback, the distance covered by left and right wheels is instead estimated from the control command.

In order to implement this simplification initial tests were conducted with Rollo. The robot was run in a straight line, constant curvature arcs and performed rotations at different commands. Rollo's location feedback from the motion capture system was logged. For a given command, average speed of left and right wheel was determined from the recorded data. Matlab environment was used to perform analysis of logged data according to following algorithm:

1. The position in x and y coordinates of Rollo was plotted along with *line of best fit* for the data set.
2. For the displacement of the robot total distance travelled by the robot and change in orientation was determined using the *line of best fit*.
3. Using the timestamps from the recorded data, distance travelled and change in orientation were determined for a unit of time.
4. The rotation speeds of the wheels were determined using the relation in equations (1) ÷ (6).

The above procedure is based on the assumption that there was no linear acceleration at a given velocity command and the rate of change of robot's orientation was constant.

For simplicity only the forward motion was modelled.

Table 2 shows the predetermined angular velocities of the wheel for given command velocities determined according to the above algorithm.

Velocity command left v_L [%]	Velocity command right v_R [%]	Angular velocity left n_L [rad/s]	Angular velocity right n_R [rad/s]
6	6	1.381	1.382
12	12	9.530	9.531
19	19	16.729	16.736
56	56	32.040	31.250
12	19	11.778	11.998
19	12	11.978	11.720
31	38	27.974	28.113
31	31	23.798	23.684

Table 1: Predetermined wheels angular velocity based on test logs for given forward command.

Velocity command		Angular velocity	
Left v_L [%]	Right v_R [%]	Left n_L [rad/s]	Right n_R [rad/s]
6	6	1.381	1.382
12	12	9.530	9.531
19	19	16.729	16.736
56	56	32.040	31.250
12	19	11.778	11.998
19	12	11.978	11.720
31	38	27.974	28.113
31	31	23.798	23.684

Table 2: Predetermined wheels angular velocity based on test logs for given forward command.

Challenges in the modelling of Rollo

1. Due to the mechanical differences in the two legs of Rollo, it is almost impossible for the two wheels to run at the same speed. Moreover, the two motors are powered using two different battery packs and are run in open loop. The difference in voltage level of battery packs results in a different current at the same velocity command using pulse-width modulation increasing the overall velocity error.

2. The robot randomly skids and changes its orientation even for a straight line run. This is partly due to mechanical properties of the robot, partly due to the surface of the laboratory, which serves as testing area and is uneven. The rubber of the tyres lacks sufficient grip on the floor to prevent skidding. One of the Rollo's wheels is visibly skewed. This is a major source for deviation in behaviour of robot's adjusted odometry model.

3. The odometry model is based on the assumption that the wheels have no thickness and have only one point of contact with ground surface. Rollo has four powered wheels by two motors, where each wheel is a $4cm$ thick. Additionally there are four support wheels in front and back of the two major wheel sets, so that Rollo does not fall over.

Odometry errors

As briefly discussed above, odometry is very sensitive to errors. However, results of odometry based models can be improved by rapid and accurate data collection, equipment calibration and processing of acquired data.

The sources of odometry errors can be broadly divided into two categories:

1. Systematic errors

2. Non-systematic errors

Systematic errors

Systematic errors are caused by inherent properties of the system. They are usually the result of imperfections in the design, production and mechanical construction of a mobile robot and it's components. In ideal environment conditions, like most smooth indoor surfaces, systematic errors play a much bigger role in than non-systematic errors. Systematic errors also accumulate constantly over time.

In general case there are three main sources of systematic errors in odometry of wheeled robots during a robot's movement:

- Distance
- Rotation
- Skew

Distance error accumulates with the robot travelling. Rotation and skew errors are of less significance with small distance and speed movements, however they are usually dominant with time and influence the overall error more than the distance error.

The major sources of the systematic error in an odometry model of a wheeled robot from a mechanical point of view are unequal diameters of wheels and the uncertainty about the effective wheelbase. Rollo consists of four plastic tyres covered in rubber stripes, two wheels on each side with additional four supporting wheels preventing Rollo from falling over.

Rubber on tyres helps improve traction, however with time rubber compresses differently under asymmetric load distribution. The diameters of wheels are used in computing S_L and S_R as shown in equations (1) and (2).

Furthermore, the contact area between the wheels of Rollo and the floor is comparatively wide for a robot of it's size. This introduces difficulties in determining the wheelbase of the robot. Wheelbase is used in computation of angle of travel $\Delta\Theta$ as shown in equation (11).

Other examples of sources of mechanical errors include:

- Average of both wheel diameters differ from nominal diameter
- Misalignment of wheels
- Unbalanced wheels
- Digitalization of acquired data (encoder sampling rate and digital to analog converter resolution)

Non-systematic errors

Non-systematic errors are caused by unknown or unpredictable changes in the system and in the environmental conditions. In wheeled mobile robots, they are usually caused by uneven floors, like in the case of testing area of Rollo. On rough surfaces with significant irregularities, non-systematic errors may be more dominant over systematic errors. On contrary to systematic errors, non-systematic errors do not accumulate with time, but occur randomly.

Possible examples of non-systematic errors are:

- Travel over uneven floors or unexpected objects on the floor
- Wheelslip due to:
  - Slippery floors
  - Over-acceleration
  - Fast turning (skidding)
  - Temporal loss of wheel contact with the floor

Measuring odometry errors

University of Michigan Benchmark test (UMBmark) procedure was supposed to be used to determine odometry errors and calibrate the provided system. This procedure was developed by Johann Borenstein and Liqiang Feng around 1995 at the University of Michigans Advanced Technologies Laboratories. Further details of the test can be found in the research paper UMBmark: A Benchmark Test for Measuring Odometry Errors in Mobile Robots.

As discussed in Systematic errors on the previous page, there are two dominant sources of systematic errors in odometry: unequal wheel diameters and the uncertainty about the effective wheelbase.

Unfortunately, due to unavailability of encoder feedback estimations of S_L and S_R are done using the command given to the robot, so the control input. This means, that equations (1) and (2) are not used in the modified model.

Even if it should be possible to determine the error in the relative size of the wheels and the error due to the uncertainty in effective wheelbase, those can not be corrected or calibrated in the current model. However, the details of the procedure to determine effective wheelbase are described in this section.

As can be seen in the equation (11), incorrect wheelbase value will introduce a significant error in computation of angle of travel $\Delta\Theta$. This error E_b, is directly proportional to the effective wheelbase in relation to the nominal wheelbase l:

$$E_b = \frac{effectivewheelbase}{l} \tag{14}$$

After computing error E_b, the nominal wheelbase in equation (11) can be replaced by effective wheelbase. Thus, this error can be corrected by multiplying the nominal wheelbase b_n with the error factor E_b resulting in the effective wheelbase b_e:

$$\Delta\Theta = \frac{S_L - S_R}{E_b \cdot l} \tag{15}$$

The error E_b can be determined using UMBmark square test procedure, whose summary for Rollo is as follows:

1. At the beginning of or before the actual run, measurements of the absolute position and orientation of Rollo using motion capture system are performed along with initialization of vehicle's odometry program to the starting point.
2. Running the vehicle through a 2×2 m square path in clockwise direction while making sure to:
   (a) stop after each 2 m straight segment,
   (b) make a total of four 90°-turns on the spot,
   (c) run the vehicle slowly to avoid slippage.
3. After returning to the starting area, again measurements of the absolute position and orientation of Rollo are performed using motion capture system.
4. Comparing between the absolute position and robot's calculated position using equations (16), (17) and (18).

$$\epsilon x = x_{abs} - x_{calc} \tag{16}$$

$$\epsilon y = y_{abs} - y_{calc} \tag{17}$$

$$\epsilon \theta = \theta_{abs} - \theta_{calc} \tag{18}$$

ϵx, ϵy, $\epsilon\theta$ – Position and orientation errors due to odometry model

x_{abs}, y_{abs}, θ_{abs} – Absolute position and orientation of Rollo

x_{calc}, y_{calc}, θ_{calc} – Calculated position and orientation of Rollo using odometry model

5. Repetition of steps $1 \div 4$ for four more times, giving a total of five runs.
6. Repetition of steps $1 \div 5$ in counterclockwise direction.
7. Computation of center of gravity coordinates for each cluster in clockwise and counterclockwise directions using equations (19) and (20), where n is the number of runs, so $n = 5$ in the proposed scenario.

$$x_{cg,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} \epsilon x_{i,cw/ccw} \tag{19}$$

$$y_{cg,cw/ccw} = \frac{1}{n}\sum_{i=1}^{n} \epsilon y_{i,cw/ccw} \tag{20}$$

8. Computation of E_b using equations (21),(22) and (23)

$$E_b = \frac{90^\circ}{90^\circ - \alpha} \tag{21}$$

where α can be defined as

$$\alpha = \frac{x_{cg,cw} + x_{cg,ccw}}{-4L}\frac{180^\circ}{\pi} \tag{22}$$

or

$$\alpha = \frac{y_{cg,cw} - y_{cg,ccw}}{-4L}\frac{180^\circ}{\pi} \tag{23}$$

and L represents the side of square segment, for the given test area conditions in case of Rollo $L = 2\ m$.

System and measurements model

System model

In order to implement Kalman filter, the previously described model must be first represented in state space. Location of the robot can be defined at instant k using state variables (24), which include position in x and y coordinates and orientation θ.

$$x_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} \tag{24}$$

The control input provided to robot consists of left and right wheel speeds. This defines the distance covered by both the wheels in a given unit of time. The relative displacement of the robot at instant k can be notated by d_k. Using equations (7), (9) and (10), derived on page on page 3, relative displacement can be expressed in terms of ΔS and $\Delta\Theta$. Thus, control input u_k can be expressed as a function of relative displacement d_k.

$$u_k = j(d_k) \tag{25}$$

$$u_k = \begin{bmatrix} u_{[\Delta S],k} \\ u_{[\Delta\theta],k} \end{bmatrix}$$

Given x_{k-1} and u_{k-1}, the next location of the robot x_k can be computed.

$$x_k = f(x_{k-1}, u_{k-1}) = \begin{bmatrix} f_x(x_{k-1}, u_{k-1}) \\ f_y(x_{k-1}, u_{k-1}) \\ f_\theta(x_{k-1}, u_{k-1}) \end{bmatrix} \tag{26}$$

In the above derivation of the system model it was assumed that there are no noise sources. In the next section, noise has been modelled into the system.

System model with noise

First, noise was added to the relative displacement with the assumption, that it can be modelled by a random noise vector q_k such that, the noise is a Gaussian distribution with zero mean $\hat{q_k}$ and covariance matrix U_k.

$$q_k \sim N(\hat{q_k}, U_k) \tag{27}$$

where

$$\hat{q_k} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and

$$U_k = \mathrm{E}(q_k - \hat{q_k})(q_k - \hat{q_k})^T$$

$$U_k = \begin{bmatrix} \sigma^2_{q[\Delta S],k} & \sigma_{q[\Delta\theta],k}\sigma_{q[\Delta S],k} \\ \sigma_{q[\Delta\theta],k}\sigma_{q[\Delta S],k} & \sigma^2_{q[\Delta\theta],k} \end{bmatrix}$$

With the assumption that the noise sources are independent, the off-diagonal elements of the covariance matrix U_k are equal to zero.

The control input, or relative displacement, can be expressed as shown in (28).

$$u_k = j(d_k) + q_k \tag{28}$$

$$u_k = \begin{bmatrix} u_{[\Delta S],k} \\ u_{[\Delta\theta],k} \end{bmatrix} + \begin{bmatrix} q_{[\Delta S],k} \\ q_{[\Delta\theta],k} \end{bmatrix}$$

This makes u_k a random vector. Assuming, that $u_{[\Delta S],k}$ and $u_{[\theta],k}$ are deterministic, uncertainty in u_k equals the uncertainty in the noise term q_k.

The system noise can similarly be modelled by a random noise vector w_k such that, the noise is a Gaussian distribution with zero mean $\hat{w_k}$ and covariance matrix Q_k. This noise source is not directly related to the

relative displacement, however it is inherent to the system. Modelling errors, odometry errors, discretization and approximations involved in the derivation of the model all play a part in this noise source. Noise vector w_k is represented in equation (29).

$$w_k \sim N(\hat{w_k}, Q_k) \tag{29}$$

where

$$\hat{w_k} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and

$$Q_k = \mathrm{E}(w_k - \hat{w_k})(w_k - \hat{w_k})^T$$

$$Q_k = \begin{bmatrix} \sigma^2_{w[x],k} & \sigma_{w[y],k}\sigma_{w[x],k} & \sigma_{w[\theta],k}\sigma_{w[x],k} \\ \sigma_{w[x],k}\sigma_{w[y],k} & \sigma^2_{w[y],k} & \sigma_{w[\theta],k}\sigma_{w[y],k} \\ \sigma_{w[x],k}\sigma_{w[\theta],k} & \sigma_{w[y],k}\sigma_{w[\theta],k} & \sigma^2_{w[\theta],k} \end{bmatrix}$$

With the assumption that the noise sources are independent, the off-diagonal elements of the covariance matrix Q_k are presumably zero. Determination of matrix Q_k is discussed in section Practical implementation of extended Kalman filter on page 15.

The system can be expressed as shown below.

$$x_k = f(x_{k-1}, u_{k-1}) + w_k \tag{30}$$

$$x_k = \begin{bmatrix} f_x(x_{k-1}, u_{k-1}) \\ f_y(x_{k-1}, u_{k-1}) \\ f_\theta(x_{k-1}, u_{k-1}) \end{bmatrix} + \begin{bmatrix} w_{[x],k-1} \\ w_{[y],k-1} \\ w_{[\theta],k-1} \end{bmatrix} \tag{31}$$

x_k is a random vector and with every time step the variance of system noise increases. Therefore the variance of the location grows with every time step also.

The proposed system model was prepared for the implementation of Kalman filter, which requires as next step the preparation of the measurement model.

Measurement model with noise

Starting with the assumption that there is no noise in the measurement, measurement vector z_k simply consists of the corresponding state variables and is defined by equation (32).

$$z_k = \begin{bmatrix} z_{[x],k} \\ z_{[y],k} \\ z_{[\theta],k} \end{bmatrix} \tag{32}$$

In the proposed system, motion capture system available in the laboratory is used to localize Rollo. It acts as absolute sensor by providing directly the position in x and y coordinates and orientation θ of the robot. Thus, z_k in this case is simply expressed as (33).

$$z_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} \tag{33}$$

Provided that the motion capture system is not noise free, this has been incorporated into the measurement model. With the assumption, that measurement noise can be modelled by a random noise vector v_k such that, the noise is a Gaussian distribution with zero mean $\hat{v_k}$ and covariance matrix R_k.

$$v_k \sim N(\hat{v_k}, R_k) \tag{34}$$

where

$$\hat{v_k} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and

$$R_k = \mathrm{E}(v_k - \hat{v_k})(v_k - \hat{v_k})^T$$

$$R_k = \begin{bmatrix} \sigma^2_{v[x],k} & \sigma_{v[y],k}\sigma_{v[x],k} & \sigma_{v[\theta],k}\sigma_{v[x],k} \\ \sigma_{v[x],k}\sigma_{v[y],k} & \sigma^2_{v[y],k} & \sigma_{v[\theta],k}\sigma_{v[y],k} \\ \sigma_{v[x],k}\sigma_{v[\theta],k} & \sigma_{v[y],k}\sigma_{v[\theta],k} & \sigma^2_{v[\theta],k} \end{bmatrix}$$

Again, assuming that the noise sources are independent, the off-diagonal elements of the covariance matrix R_k are presumably zero.

The final measurement model can be expressed as shown in (36).

$$z_k = x_k + v_k \tag{35}$$

$$z_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} + \begin{bmatrix} v_{[x],k-1} \\ v_{[y],k-1} \\ v_{[\theta],k-1} \end{bmatrix} \tag{36}$$

Since the measurement noise v_k is a Gaussian random vector, this makes z_k a random vector.

Kalman filter

Kalman filter is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone[3].

A physical system is driven by a set of external inputs or controls. Its outputs are evaluated by sensors such that, the knowledge on the system's behaviour is solely given by the inputs and the observed outputs. For example, in case of Rollo, the physical system is a mobile robot and given is the problem of localizing

[3]e-Study Guide for: Introduction to Probability Theory and Stochastic Processes

it. The control, or the input, is defined by the velocity command given to the left and right wheels, which determines the relative displacement of Rollo. The observed output is the position and orientation of Rollo measured from the motion capture system. The observations convey the errors and uncertainties in the process, namely the sensor noise and the system errors. Based on the available information (control inputs and observations), Kalman filter computes an estimate of the system's state by minimizing the variance of the estimation error. From a theoretical standpoint, the main assumptions of the Kalman filter are that the underlying system is a linear dynamical system and that all error terms and measurements have a Gaussian distribution.

Kalman filtering is a recursive analytical technique involving two main steps: prediction and innovation.

Prediction is also known as *time update* and refers to prediction of state variables using information from previous state and control input. This step does not involve using measurement.

Innovation which is also known as *correction* or *measurement update* refers to estimation of state variable using measurement.

Kalman filter is broadly used in robotics for solving the localization problem, because of its efficiency and accuracy. Some of its characteristics are:

1. Very easy implementation
2. Computationally efficient, updates filter when providing new measurements to the existing data set
3. Uncertainty estimates are provided as part of the filter
4. Recursive algorithm, which makes it suitable for real time applications using only the present input measurements and the previously calculated state.

All members of Kalman family of filters like extended Kalman filter and unscented Kalman filter, comply with a structured sequence of six steps per iteration:

1. **State estimate time update:** An updated state prediction $\hat{x}_{k|k-1}$ is made, based on a priori information and the system model.
2. **Error covariance time update:** The second step is to determine the predicted state-estimate error covariance matrix $\Sigma_{k|k-1}$ based on a priori information and the system model.
3. **Estimate system output:** The third step is to estimate the system's output $\hat{z}_k$ corresponding to the timestamp of the most recently received measurement using present a priori information.
4. **Estimator gain matrix:** The fourth step is to compute the estimator gain matrix H_k.
5. **State estimate measurement update:** The fifth step is to compute the a posteriori state estimate $\hat{x}_{k|k}$ by updating the a priori estimate using the estimator gain and the output prediction error.
6. **Error covariance measurement update:** The final step computes the a posteriori error covariance matrix $\Sigma_{k|k}$.

Kalman filter equations for linear system

In order to implement Kalman filter first represent the linear system as follows.

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + w_k \\ z_k &= C_k x_k + v_k \end{aligned} \tag{37}$$

where $w_k \sim N(0, Q_k)$ and $v_k \sim N(0, R_k)$ as described in System model with noise on page 9 and Measurement model with noise on page 10 respectively.

For Kalman filter A, B, C, Q and R can be time variant. However, for simplicity it is assumed that they are time invariant for the provided system. Given initial state estimate $\hat{x}_{0|0}$ and initial state covariance matrix $\Sigma_{0|0}$ the Kalman filter can be computed using following set of equations.

Prediction (time update)

$$\begin{aligned} \hat{x}_{k|k-1} &= A\hat{x}_{k-1|k-1} + Bu_{k-1} \\ \Sigma_{k|k-1} &= A\Sigma_{k-1|k-1}A^T + Q \end{aligned} \tag{38}$$

Innovation (measurement update)

$$\begin{aligned} H_k &= \Sigma_{k|k-1}C^T(C\Sigma_{k|k-1}C^T + R)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + H_k[z_k - C\hat{x}_{k|k-1}] \\ \Sigma_{k|k} &= [I - H_kC]\Sigma_{k|k-1} \end{aligned} \tag{39}$$

Kalman filter equations for non-linear system

Now we consider a non-linear extension to the Kalman filter, *Extended Kalman Filter (EKF)*. The EKF implements Kalman filter for non-linear system dynamics that result from the linearization of the original non-linear filter dynamics around the previous state estimates.

A non linear system can be represented as follows.

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}) + w_{k-1} \\ z_k &= h(x_k) + v_{k-1} \end{aligned} \tag{40}$$

where $w_k \sim N(0, Q_k)$ and $v_k \sim N(0, R_k)$.

Another assumption is that $f(\cdot)$, $h(\cdot)$, Q and R are time invariant for the provided non-linear system. $f(\cdot)$ and $h(\cdot)$ are linearized about the prior best estimate of the states at each instant of time by finite difference method in order to compute covariances. Given initial state estimate $\hat{x}_{0|0}$ and initial state covariance matrix $\Sigma_{0|0}$ the extended Kalman filter can be computed using following set of equations.

Prediction (time update)

$$\begin{aligned} \hat{x}_{k|k-1} &= f\left(\hat{x}_{k-1|k-1}, u_{k-1}\right) \\ \Sigma_{k|k-1} &= J_f\Sigma_{k-1|k-1}J_f^T + Q \end{aligned} \tag{41}$$

Innovation (measurement update)

$$\begin{aligned} H_k &= \Sigma_{k|k-1}J_h^T(J_h\Sigma_{k|k-1}J_h^T + R)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + H_k[z_k - h\left(\hat{x}_{k|k-1}\right)] \\ \Sigma_{k|k} &= [I - H_kJ_h]\Sigma_{k|k-1} \end{aligned} \tag{42}$$

where J_f and J_h are the jacobian matrices.

J_f is the jacobian matrix containing the partial derivative of the system function $f(\cdot)$ with respect to the state x, evaluated at the last state estimate $\hat{x}_{k-1|k-1}$ and control input u_{k-1}.

J_h is the jacobian matrix containing partial derivatives of the measurement function $h(\cdot)$ with respect to the state x, evaluated at the prior state estimate $\hat{x}_{k|k-1}$.

Extended Kalman filter equations for odometry

After deriving physical model of the provided system for Odometry and understanding the extended Kalman filter in Kalman filter equations for non-linear system, preparation for the EKF equations for odometry can be done.

$$\begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} = \begin{bmatrix} f_x(x_{k-1}, u_{k-1}) \\ f_y(x_{k-1}, u_{k-1}) \\ f_\theta(x_{k-1}, u_{k-1}) \end{bmatrix} + \begin{bmatrix} w_{[x],k-1}) \\ w_{[y],k-1}) \\ w_{[\theta],k-1}) \end{bmatrix} \tag{43}$$

$$\begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} = \begin{bmatrix} x_{[x],k-1} + u_{[\Delta]S,k-1} \cdot \cos(x_{[\Theta],k-1} - \frac{u_{[\Delta\Theta],k-1}}{2}) \\ x_{[y],k-1} + u_{[\Delta]S,k-1} \cdot \sin(x_{[\Theta],k-1} - \frac{u_{[\Delta\Theta],k-1}}{2}) \\ x_{[\theta],k-1} - u_{[\Delta\Theta],k-1}) \end{bmatrix} + \begin{bmatrix} w_{[x],k-1}) \\ w_{[y],k-1}) \\ w_{[\theta],k-1}) \end{bmatrix} \tag{44}$$

Prediction (time update)

$$\begin{bmatrix} \hat{x}_{[x],k|k-1} \\ \hat{x}_{[y],k|k-1} \\ \hat{x}_{[\theta],k|k-1} \end{bmatrix} = \begin{bmatrix} \hat{x}_{[x],k-1|k-1} + u_{[\Delta]S,k-1} \cdot \cos(\hat{x}_{[\Theta],k-1|k-1} - \frac{u_{[\Delta\Theta],k-1}}{2}) \\ \hat{x}_{[y],k-1} + u_{[\Delta]S,k-1} \cdot \sin(\hat{x}_{[\Theta],k-1|k-1} - \frac{u_{[\Delta\Theta],k-1}}{2}) \\ \hat{x}_{[\theta],k-1|k-1} - u_{[\Delta\Theta],k-1}) \end{bmatrix} \tag{45}$$

$$\begin{aligned} J_{f,k} &= \frac{\partial f(x)}{\partial x}\Big|_{x=\hat{x}_{k-1|k-1}, u=u_{k-1}} \\ &= \begin{bmatrix} \frac{\partial f_x}{\partial x_{[x]}} & \frac{\partial f_x}{\partial x_{[y]}} & \frac{\partial f_x}{\partial x_{[\theta]}} \\ \frac{\partial f_y}{\partial x_{[x]}} & \frac{\partial f_y}{\partial x_{[y]}} & \frac{\partial f_y}{\partial x_{[\theta]}} \\ \frac{\partial f_\theta}{\partial x_{[x]}} & \frac{\partial f_\theta}{\partial x_{[y]}} & \frac{\partial f_\theta}{\partial x_{[\theta]}} \end{bmatrix}_{x=\hat{x}_{k-1|k-1}, u=u_{k-1}} \\ &= \begin{bmatrix} 1 & 0 & -u_{[\Delta]S} \cdot \sin(x_{[\Theta]} - \frac{u_{[\Delta\Theta]}}{2}) \\ 0 & 1 & +u_{[\Delta]S} \cdot \cos(x_{[\Theta]} - \frac{u_{[\Delta\Theta]}}{2}) \\ 0 & 0 & 1 \end{bmatrix}_{x=\hat{x}_{k-1|k-1}, u=u_{k-1}} \end{aligned} \tag{46}$$

$$\Sigma_{[x],k|k-1} = J_f \Sigma_{[x],k-1|k-1} J_f^T + Q \tag{47}$$

Innovation (measurement update)

$$\begin{aligned} J_{h,k} &= \frac{\partial h(x)}{\partial x}\Big|_{x=\hat{x}_{k|k-1}} \\ &= \begin{bmatrix} \frac{\partial h_x}{\partial x_{[x]}} & \frac{\partial h_x}{\partial x_{[y]}} & \frac{\partial h_x}{\partial x_{[\theta]}} \\ \frac{\partial h_y}{\partial x_{[x]}} & \frac{\partial h_y}{\partial x_{[y]}} & \frac{\partial h_y}{\partial x_{[\theta]}} \\ \frac{\partial h_\theta}{\partial x_{[x]}} & \frac{\partial h_\theta}{\partial x_{[y]}} & \frac{\partial h_\theta}{\partial x_{[\theta]}} \end{bmatrix}_{x=\hat{x}_{k|k-1}, u=u_{k-1}} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{x=\hat{x}_{k|k-1}, u=u_{k-1}} \end{aligned} \tag{48}$$

Using the above derived jacobian matrix (48) for computation of Kalman gain matrix H_k, a posteriori state estimate and a posteriori error covariance matrix following concludes.

$$\begin{aligned} H_k &= \Sigma_{k|k-1}(\Sigma_{k|k-1} + R)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + H_k[z_k - \hat{x}_{k|k-1}] \\ \Sigma_{k|k} &= [I - H_k J_h]\Sigma_{k|k-1} \end{aligned} \tag{49}$$

Practical implementation of extended Kalman filter

Proceeding with the EKF model for odometry, initialization of the model is necessary before practical implementation, which requires following components to be initialized:

1. Initial state estimate $\hat{x}_{0|0}$,
2. Initial state covariance matrix $\Sigma_{0|0}$,
3. Process covariance matrix Q,
4. Measurement covariance matrix R.

Initial state estimate $\hat{x}_{0|0}$

This is the initial estimate of the state vector required for the first iteration of the EKF. In the provided system, Optitrack Motion Capture system is used for measurement. Optitrack provides ground truth and positional accuracy of up to sub-millimeter range. For performed tests the measurement sensor readings were used to determine $\hat{x}_{0|0}$.

Initial state covariance matrix $\Sigma_{0|0}$

Initial state covariance matrix is based on the initialization error of the state. If the initial state estimate is very close to the actual state, this matrix will contain very small values. With the assumption that noise sources for different state elements are independent, the off diagonal elements of the covariance matrix can be assumed to equal zero.

Usually if this matrix is unknown identity matrix is used, since state covariance matrix is updated in every iteration of EKF. For a stable EKF this matrix should be converging over time. An initial estimate closer to the actual state will offer faster convergence of the Kalman filter.

Since the initial state is determined from the motion capture system, initial state covariance matrix could be assumed to be zero. However, since the chosen procedure simulates a sensor not as accurate as motion capture, the initial state covariance matrix is taken as identity.

Process covariance matrix Q

This is the error covariance matrix of the process and gives an estimate of uncertainty in the state equations. The uncertainty in the process could be a result of various sources of error, including modeling errors, odometry errors, discretization and approximations involved in the derivation of the model.

In order to get a better understanding of error in the model, computation of error based on odometry model from log files and analysis were performed.

For testing a different range of values for Q was chosen in order to understand the behaviour of the EKF better.

Measurement covariance matrix R

This is the error covariance matrix of measurement sensor and provides a measure of how uncertain the measurement is. As discussed earlier, Optitrack Motion Capture system is used for measurement and it provides ground truth. For the case when the measurement is very accurate, matrix R will carry very small variances and can be approximated to zero. However, in order to simulate EKF for a system, whose measurement sensor is not as accurate, different range of values for the measurement covariance matrix R was used.

Divergence of state covariance matrix $\Sigma_{k|k}$

Proceeding with initialization of EKF as discussed above, it was observed that the filter behaved well for a specific period of time after initialization. However, after a certain amount of time the state covariance matrix started diverging very fast, resulting in failure of the filter.

It was understood that a possible reason behind this could be lack of input excitation that results in growing values of state covariance matrix and large spread of eigenvalues. In order to solve this issue, different techniques can be used to stabilize estimation and prevent windup of state estimation and covariance matrix.

The issue has been solved by performing Cholesky decompostion in the computation of Kalman gain matrix. Cholesky factorization is a decomposition of a Hermitian, a positive-definite matrix, into the product of a lower triangular matrix, or alternatively an upper triangular matrix, and its conjugate transpose. Cholesky decomposition offers numerical stability to the system, which is the main reason it was chosen.

Implementation was based on the assumption that the state covariance matrix $\Sigma_{k|k}$ is symmetrical. This resulted in acceptable behaviour of the filter, however, it slowed down the performance of EKF. With Cholesky decomposition, the innovation update of EKF can be written as shown in (50)

$$\begin{aligned} U &= CholsekyDecomposition(J_h \Sigma_{k|k-1} J_h^T + R) \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + [\Sigma_{k|k-1} U^{-1}] U^{T^{-1}} [z_k - h\left(\hat{x}_{k|k-1}\right)] \\ \Sigma_{k|k} &= \Sigma_{k|k-1} - [\Sigma_{k|k-1} U^{-1}][\Sigma_{k|k-1} U^{-1}]^T \end{aligned} \tag{50}$$

where $CholsekyDecomposition(X)$ produces an upper triangular matrix U from the diagonal and upper triangle of matrix X, thus satisfying the equation $U^T U = X$.

Robot Operating System (ROS)

Robot Operating System (ROS) is a collection of software frameworks for robot software development, it works as middleware for work with complex robotic problems. ROS uses the publisher-subscriber software architecture model for communication between nodes. The presented problem of localization of a humanoid robot using extended Kalman filter, which has been described in detail in Kalman filter, has been solved using ROS. Five nodes have been developed respecting the aspects of modularity and interchangeability of software:

- Preprocessor node
- Control node
- Communication node
- Extended Kalman filter node
- Visualization node

Each node is supposed to fulfil one specific function, so that the software can be used for different tasks and in different environments. However there are several aspects that have been merged into same nodes, that might be contra-intuitive. Functionalities of the nodes are distributed the following way:

- Preprocessor node:
  1. Filter the raw data from motion capture system
  2. Publish filtered data along with timestamp for modelling of odometry and the measurement in Kalman filter
- Control node:
  1. Translate input from keyboard into linear and angular velocities within unitary limits ($-100\% \div 100\%$)
  2. Publish data as Twist message for further processing
- Communication node:
  A. Normal operation mode
     (a) Translate data from control node into a three byte message for Rollo
     (b) Publish decoded velocities
     (c) Send commands continuously at a given rate to Rollo
     (d) Perform emergency procedure if connection to control node is lost
  B. Square test of n-th order based on the UMBmark test procedure describer in detail in Measuring odometry errors
     (a) Perform a cycle of forward movement and turns to move the robot along a square
     (b) For higher than first order square test turn around between runs
- Extented Kalman filter node:
  1. Initialize EKF
  2. Preprocessing of acquired data to make comparison of relevant information
  3. Estimate of state using EKF

4. Odometry data calculation
5. Publish results

- Visualization node:

1. Visualize data holding position and orientation of robot, from preprocessor and extended Kalman filter, which provides estimates of state and odometry data

Nodes have been written in C++ programming language with the exception of visualization node, which has been written in Python, mainly because of easy access to the powerful Matplotlib library. There has been also a significant amount of testing and prototyping of software done in Matlab and Python, mainly for the odometry model and extended Kalman filter implementation as well as analysis. Additional scripts in Bash shell were also written, mostly for the purpose of easier repetition of tasks. All of the significant files can be found in the GitHub repository established for this project.

The basic software architecture model implemented is publish-subscribe given the architecture ROS uses represented by (**??**) shows the correlation between individual nodes and environment.

More in depth documentation of the actual code and algorithms has been done using Doxygen in two formats: HTML version and local LaTeX document form. Only the most important points shall be described here.

A few aspects have been addressed while developing the software. One of those aspects is safety of operation. Should the control node loose connection to the master and at the same time be sending commands other than full stop, where linear and angular velocities are equal to zero, then in normal case the communication node would continue sending data with last values received from control node and the robot would be uncontrollable. This is the reason a safety, or emergency, procedure has been implemented. After a period of time, which can be specified from the command line using the `em` parameter, a full stop command is being send for ten times. Should the control node reestablish connection, the communication node will continue to work in normal procedure.

Since the robot has been tested in laboratory conditions, the algorithm implemented is sufficient for given condition. However it is not very robust and could be further improved. For example, the stop command could be send continuously, until control node is reconnected with ROS master or the communication node forcefully exited. Another improvement would be to send a special message that this event occurred, so that the user is informed about it without delay.

Next feature implemented into the communication node is the square test. This particular test has been implemented in accordance to UMBmark test procedure described in Measuring odometry errors. The implemented algorithm lets the user specify the forward and turning times for the square test, which is necessary in case of not having access to direct odometry data in form of encoder readings. The square test can be performed in multiple runs, so that each cycle includes:

1. Moving forward
2. Turning 90°
3. Moving forward
4. Turning 90°
5. Moving forward
6. Turning 90°
7. Moving forward
8. Turning 90°

Should there be multiple runs specified, the robot would turn around after each cycle, which is to rotate 180° and continue with step 1. In other case that would be the last or only run. Ideally the robot would stop at exactly the same position it started with the initial orientation. Since there are systematic and random errors, this is only achievable for a given tolerance. However in case of Rollo, the errors and mechanical structure differences are so great, that the test is basically not practical. During tests runs it has been only confirmed, what the analysis of the logs performed for the extended Kalman filter node concluded, namely that the turning times differ significantly, which is most probably due to mechanical asymmetry.

This makes the source code for this particular robot somewhat irrelevant. However it has been decided to leave the current implementation at this state, since it actually is sound from the theoretical point of view. It can be used with other robots should they have similar turning speeds for both directions.

Simulation and testing

Simulation of extended Kalman filter

EKF algorithm was first simulated in MATLAB using logs from Rollo's test runs. Measurement logs were collected for the robot's straight line test runs and initial value of measurement was set as the initial state estimate. EKF update was performed using this initial state estimate, control input for the given log and measurement for the log.

Figure 1 shows the results of EKF based on one of a log file. `x state[1]` and `x state[2]` represent position of robot in x and y coordinates in meters respectively. `x state[3]` represents orientation Θ of the robot in radians. The blue lines shows the state estimates updates with EKF, while the red lines represent the measurement which is the actual state.

It can be observed from the plots that the EKF based state estimates are influenced by measurement and are able to converge to the varying values of measurement.

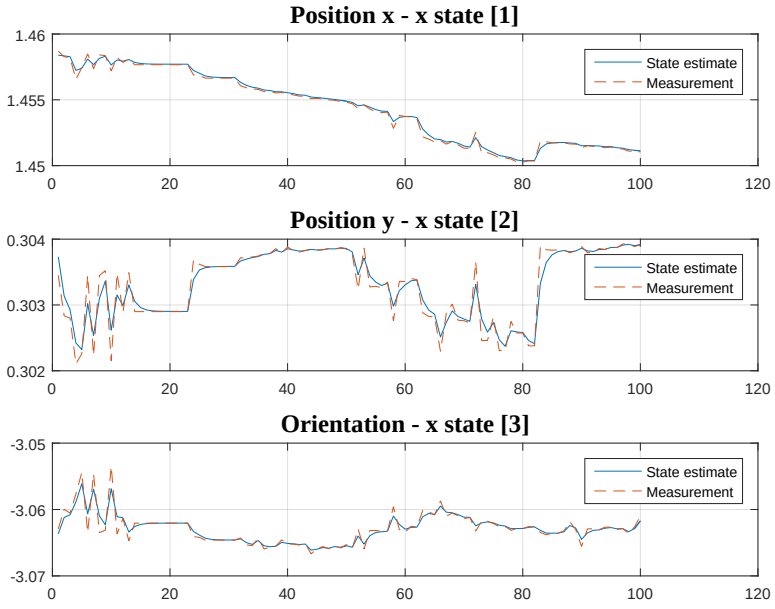

Figure 1: Results of EKF simulation based on a log file.

Figure 2 shows the results of EKF update for longer iterations to verify the stability of EKF. If original EKF equations are directly used with the system, the state estimate and state covariance matrix starts to

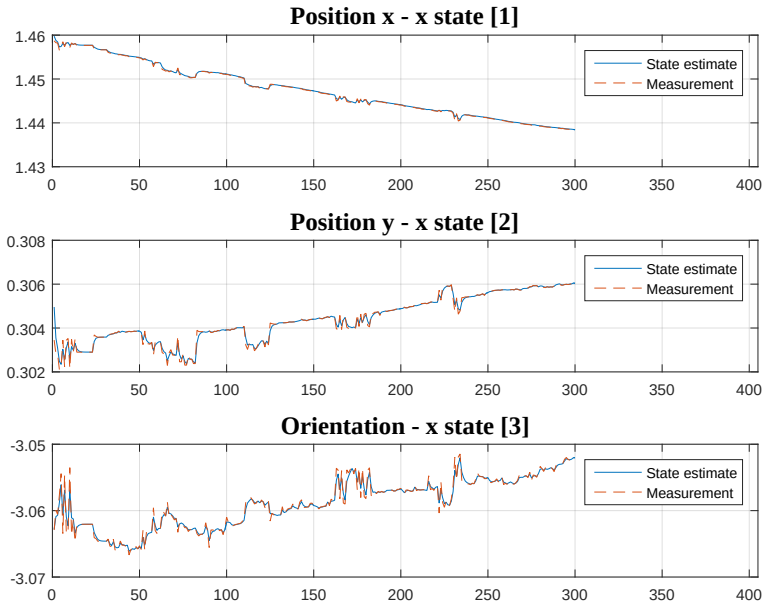

Figure 2: Results of EKF simulation based on a log file for longer iterations to verify stability.

diverge after sometime, as discussed in previous section **??**. This figure shows result of performing Cholesky decomposition for measurement update.

Testing of extended Kalman filter

Test setup

In order to conduct the tests both battery packs were fully charged and five markers were placed on Rollo at different points so it could be reliably tracked by motion capture system.

Motion capture system was calibrated so that the x-axis and y-axis for localization were defined according to Cartesian coordinate system. The Robot was placed in the arena with its orientation aligned to x-axis and then defined as a rigid body in the motion capture system. This ensured that the orientation of the robot was 0 radians when it was aligned to x-axis similar to the odometry model developed.

The complete software that received localization from the motion capture system, controlled Rollo, processed data to implement EKF and visualization was developed in ROS framework. Rollo was manually controlled using the keyboard and communication with Rollo was done at 10Hz. The measurement node was updated 25 Hz. Hence, the EKF update was also performed approximately at a rate of 25Hz.

Unfortunately, the encoder feedback was unavailable for the test but the communication node was pre-programmed to send certain angular velocities for left at a given command. The complete range was not covered, only the command values shown in the table 2. were tested.

In the start of each test run, the robot position and orientation read from the motion capture system was taken as the initial state estimate. Initial state covariance matrix was taken as identity matrix.

State prediction using the odometry based model alone was also studied for comparison with EKF based model. However, in the absence of encoder, this model was very crude. Moreover, the testing floor was uneven and had visible irregularities, which introduced non systematic errors that are not modelled in the odometry.

Initial state for the odometry model was taken the same as for the EKF based model. The logs were collected sometime after the test was conducted and analyzed. This was due to lag in the system. However, the videos of online visualization for the tests were recorded from the very start of the run.

q	r
0.1	0.1
0.1	1.0
0.1	10.0
1.0	10.0
10.0	10.0
100.0	10.0
8.0	32.0

Table 3: Predetermined wheels angular velocity based on test logs for given forward command.

Testing variables

We tested Rollo with different process covariance matrix, Q and measurement covariance matrix, R to analyze EKF. We define the two matrices here again. (For details, sections **??**, **??** and **??**can be referred again.)

For process noise covariance matrix, it was assumed that the noise sources were independent for the three states (position in x and y coordinates and the orientation) and the noise variances for all three states were taken equal and constant. Assigning the standard deviation for all states as q, the Q matrix can be written as shown in (51).

$$Q_k = \begin{bmatrix} q^2 & 0 & 0 \\ 0 & q^2 & 0 \\ 0 & 0 & q^2 \end{bmatrix} \tag{51}$$

For measurement noise covariance matrix, it was assumed that the noise sources were independent for the three states (position in x and y coordinates and the orientation) and the noise variances for all three states were taken equal and constant. Assigning the standard deviation for the all states as r, the R matrix can be written as shown in (52).

$$R_k = \begin{bmatrix} r^2 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & r^2 \end{bmatrix} \tag{52}$$

Table 3 enlists the combinations of q and r that were tested.

The EKF update rate was 10Hz for all test runs. For every test the initial state estimate was reassigned and the EKF node was restarted.

Test results and analysis

During on line testing the localization was studied using the visualization node. A snapshot of the real time visualization is shown in figure.

Logs were generated for off line and in-depth analysis of the system. For every test run, two different plots were generated. One showed position of the robot computed according to odometry, EKF and motion capture system. Second plot displayed position and orientation of the robot with respect to time. Some plots of the test runs are shown in this section.

Initial tests were conducted for $q = 0.1$ and $r = 0.1$. Figures 4 and 5 show a straight line run for 19% velocity command. It can be observed from the plot that the state estimate is almost superimposed on measurement line. This shows EKF is able to estimate the state very well. Odometry results have however

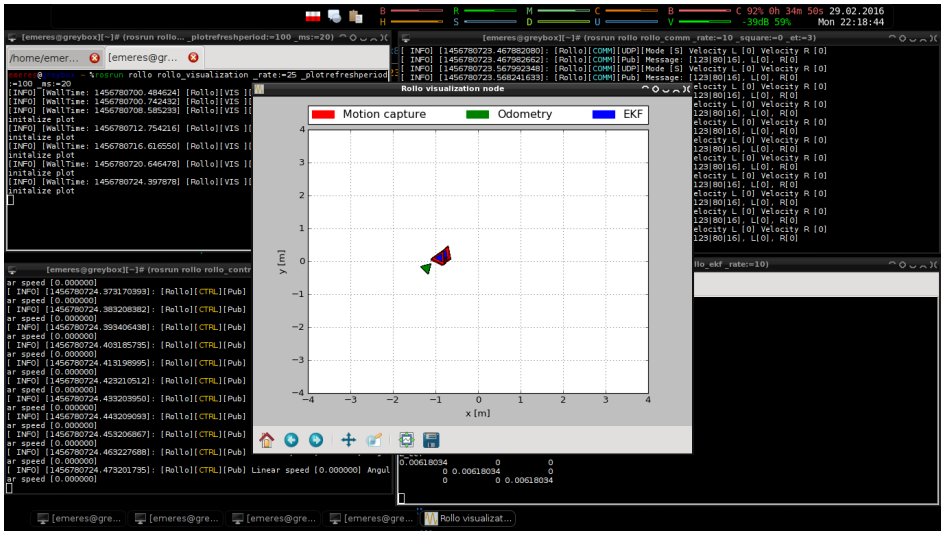

Figure 3: Real time visualization: localization of the robot based on EKF, odometry alone and the measurement.

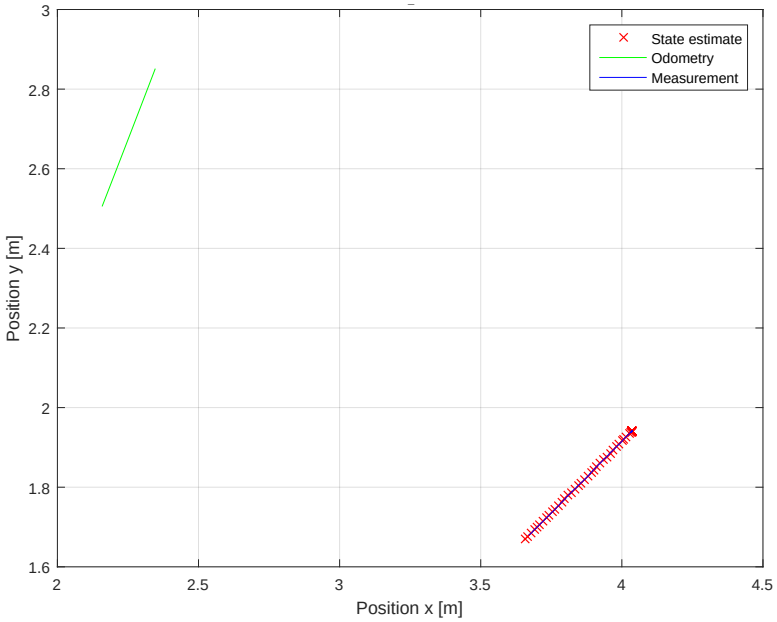

Figure 4: Test 1, velocity command 19% for both left and right wheels. q = 0.1, r = 0.1.

drifted far from actual position of the robot with time. Since this was only crudely modelled without encoders feedback, no efforts were made to improve it.

The initial values of q and r involved in EKF yielded respectable performance. Further tests were performed by adding more noise to the measurement model by taking a larger r.

By taking $q = 0.1$ and $r = 1.0$, the tracking ability of EKF was reduced. Figures 6 and 7 show a straight line run for 12% velocity command. It can be observed from the plots that the state estimate is close to the measurement line but is unable to reach it. It remains at a varying offset to the measurement.

We added further noise to the measurement model by taking $q = 0.1$ and keeping $r = 10.0$. Figures

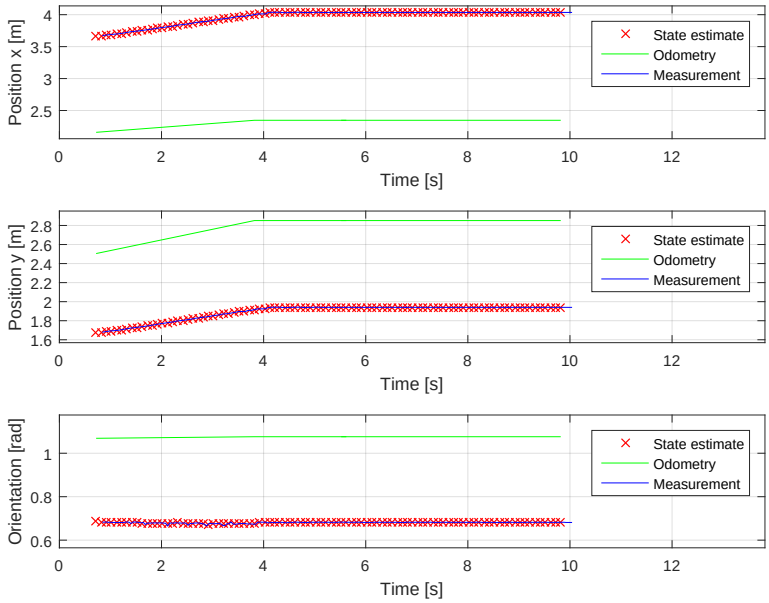

Figure 5: Test 1, velocity command 19% for both left and right wheels. q = 0.1, r = 0.1. Figure shows all three states with respect to time.

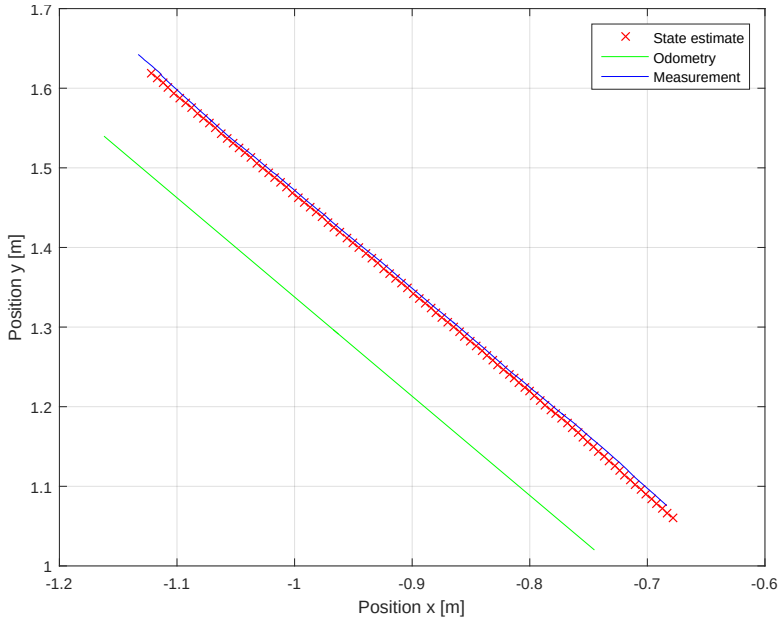

Figure 6: Test 2, velocity command 12% for both left and right wheels. q = 0.1, r = 1.0.

8 and 9 show plots of a counter clockwise rotation for 12% left velocity command and 19% right velocity command. State estimate is visibly drifted from the true position. With the increase in measurement noise, measurement loses its wightage in EKF. Further tests were conducted with keeping r constant at 10.

Figures 10 and 11 show another counter clockwise rotation at 12% left velocity command and 19% right velocity command for $q = 1.0$ and $r = 10.0$. Now, the EKF response is widely improved. By increasing the

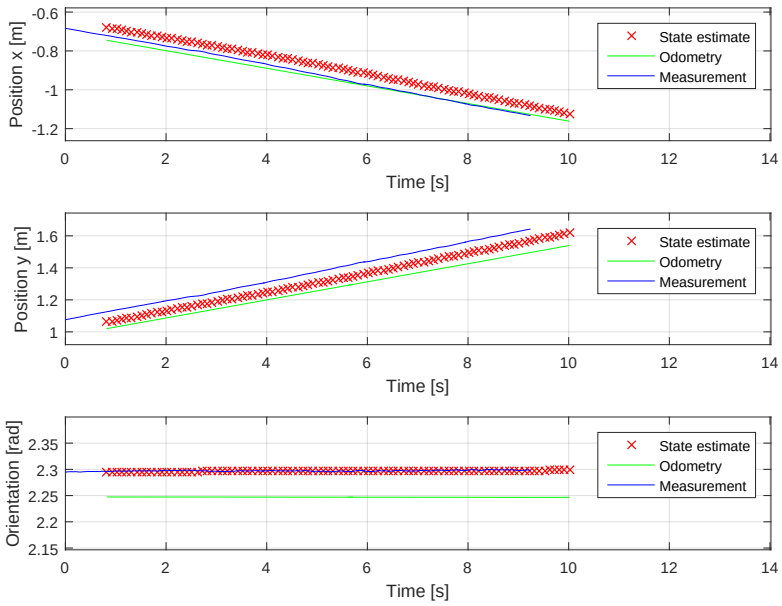

Figure 7: Test 2, velocity command 12% for both left and right wheels. q = 0.1, r = 1.0. Figure shows all three states with respect to time.

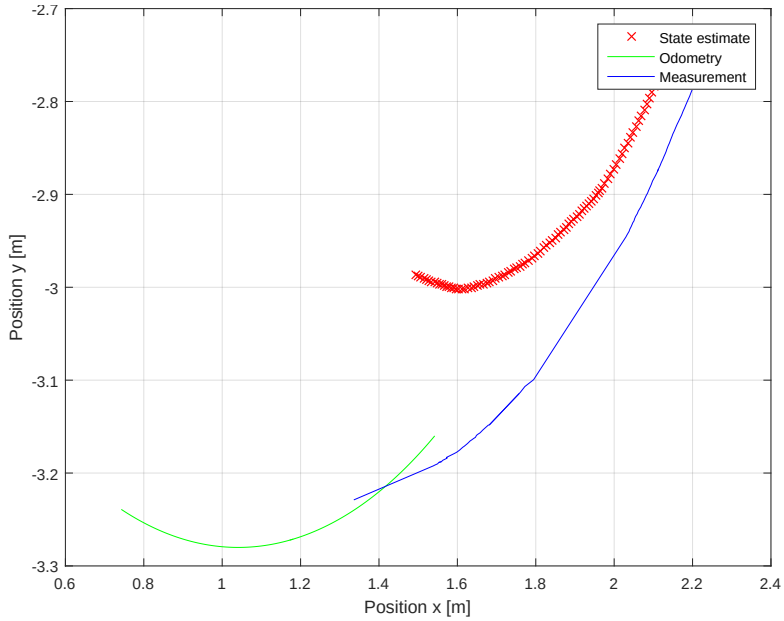

Figure 8: Test 3, velocity command 12% for left wheel and 19% for right wheel. q = 0.1, r = 10.

value of q, we are increasing the uncertainty in our process model. In comparison to the test anaylzed above, the measurement has gained its weightage here. However, it can be seen that there is still an error between the EKF based state estimate and the measurement.

We further added more noise to the process model by increasing q. Figures 12 and 13 show plots of a counter clockwise rotation at 12% left velocity command and 19% right velocity command for $q = 10.0$

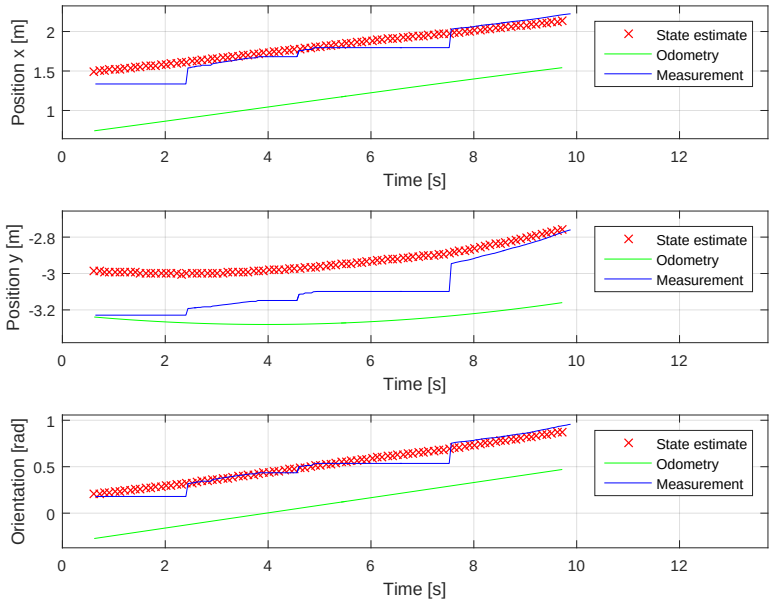

Figure 9: Test 3, velocity command 12% for left wheel and 19% for right wheel. q = 0.1, r = 10. Figure shows all three states with respect to time.

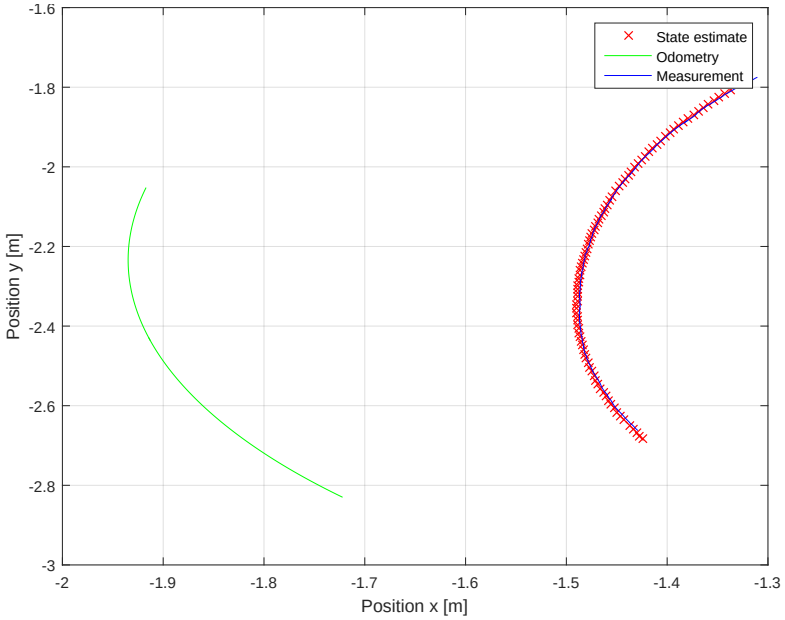

Figure 10: Test 4, velocity command 19% for left wheel and 12% for right wheel. q = 1.0, r = 10.

and $r = 10.0$. It can be observed that the EKF is able to track the localisation of the robot very well.The performance is comparable to results seen in figure 1 when the values for q and r are the same as 0.1 and 0.1. It can be understood that it is basically the relative ratio between Q and R matrices that determine the performance of EKF not the values of Q and R alone.

In figure 14 and 15, we plot a clockwise rotation at 19% left velocity command and 12% right velocity

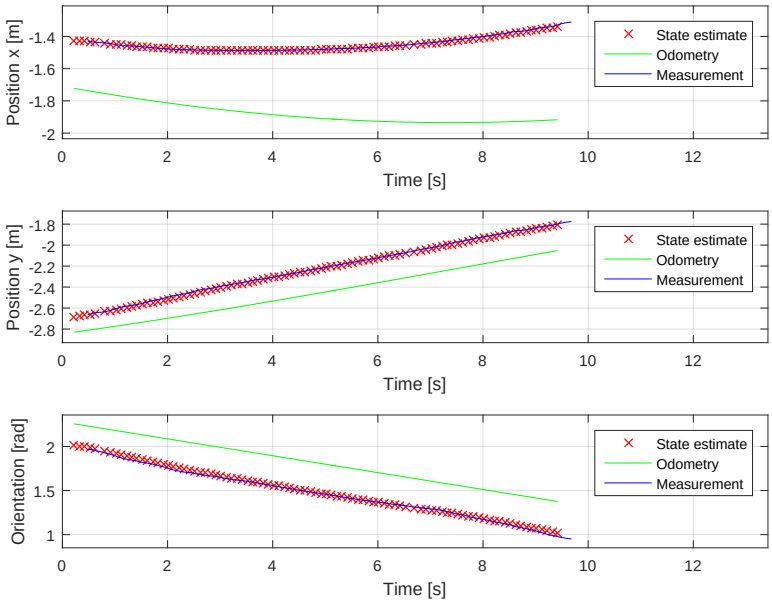

Figure 11: Test 4, velocity command 19% for left wheel and 12% for right wheel. q = 1.0, r = 10. Figure shows all three states with respect to time.

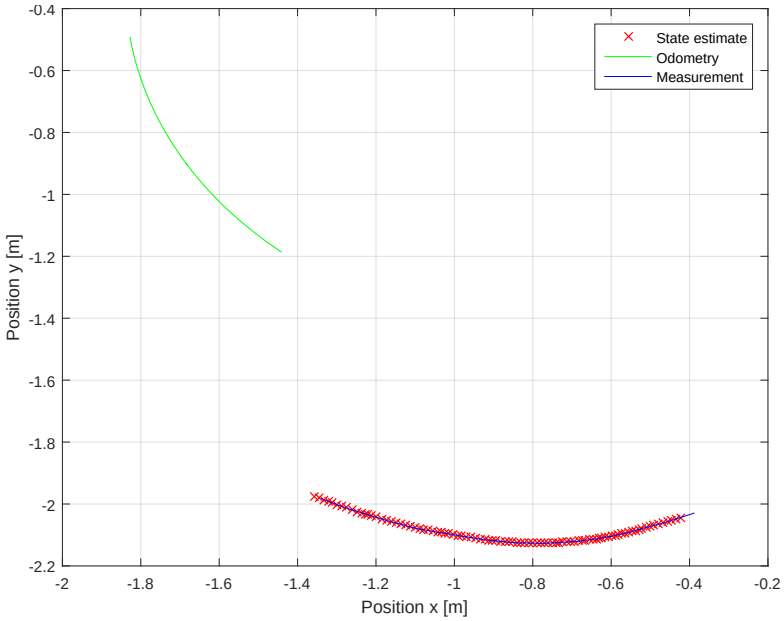

Figure 12: Test 5, velocity command 12% for left wheel and 19% for right wheel. q = 10, r = 10.

command for $q = 100.0$ and $r = 10.0$. By increasing the value of q, we are adding more noise to the process model, the measurement gains weightage in the EKF model. As can be observed in the figure, the EKF is able to localize the robot very well. It must be understood that this is because our measurement sensor provides very accurate localization of the robot.

Final test was performed with values $q = 8.0$ and $r = 32.0$. Figures 16 and 17 show plots of a counter

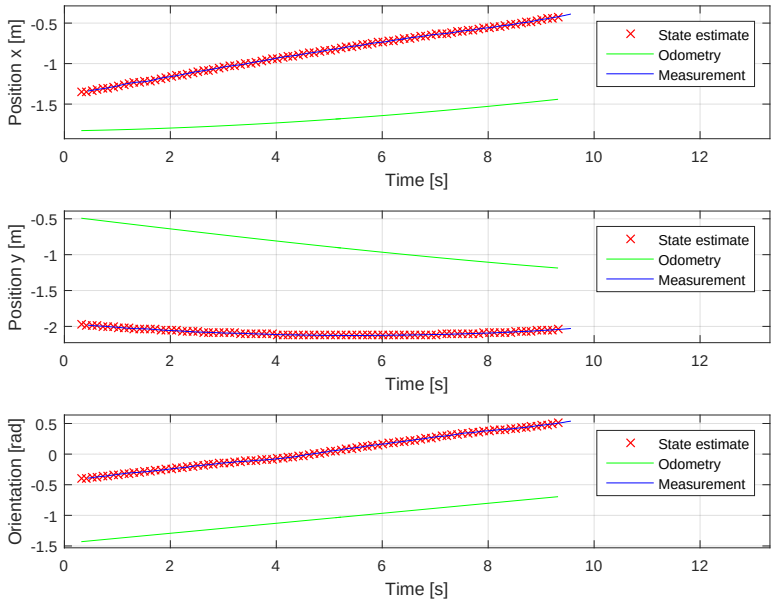

Figure 13: Test 5, velocity command 12% for left wheel and 19% for right wheel. q = 10, r = 10. Figure shows all three states with respect to time.

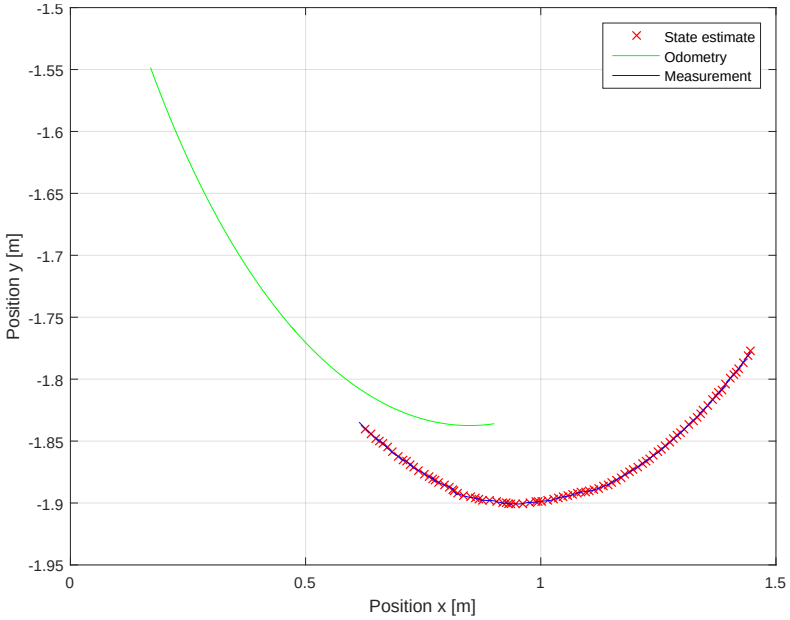

Figure 14: Test 6, velocity command 19% for left wheel and 12% for right wheel. q = 100, r = 10.

clockwise rotation at 12% left velocity command and 19% right velocity command. Since r is greater than q, the measurement model has a lower weightage as compared to the process model. It can be observed from the plots that the EKF based state estimate is unable to localize the robot well and state estimate is visibly drifted from the true position especially when running a curve.

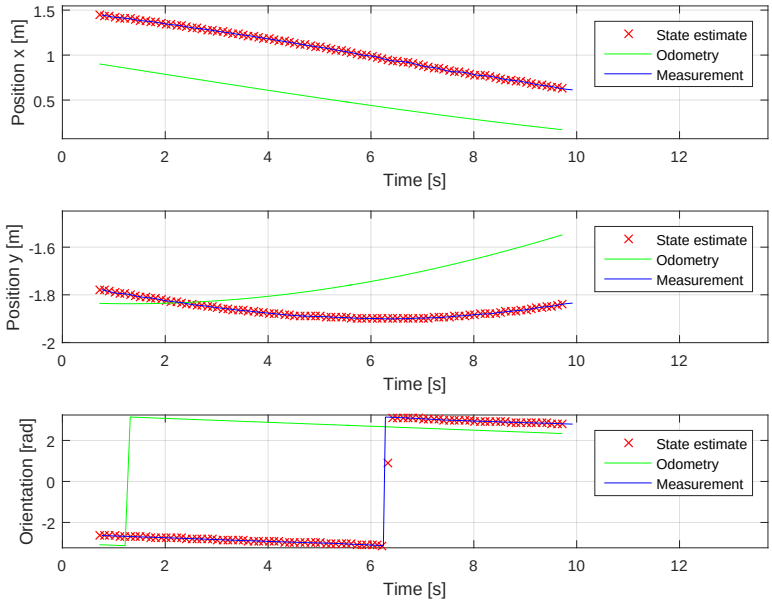

Figure 15: Test 6, velocity command 19% for left wheel and 12% for right wheel. q = 100, r = 10. Figure shows all three states with respect to time.

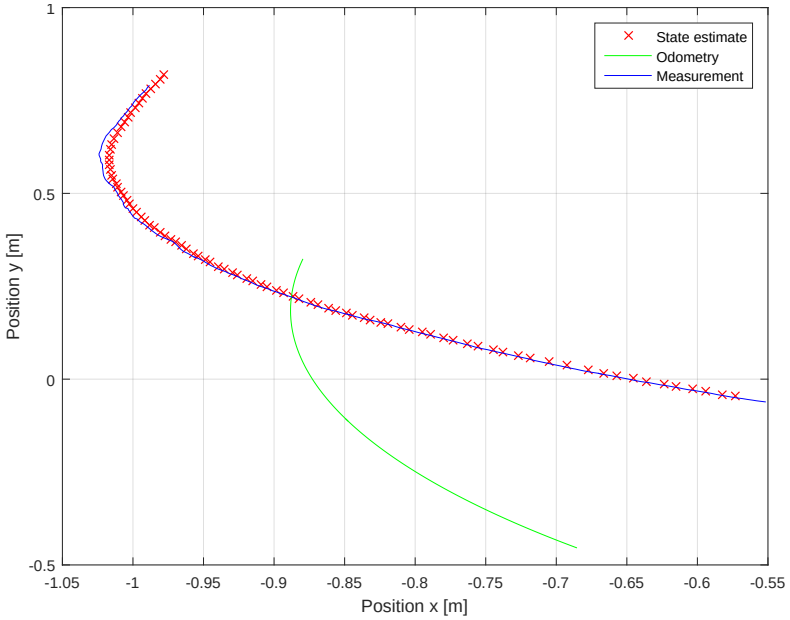

Figure 16: Test 7, velocity command 12% for left wheel and 19% for right wheel. q = 8, r = 32.

Conclusion

Localization is the estimation of position and state of a system in the world frame of reference making use of limited information. Localization has been one of the primary challenges of working with mobile robots. It is very common in mobile robots to use odometry and inertial measurement unit (IMU) as relative position

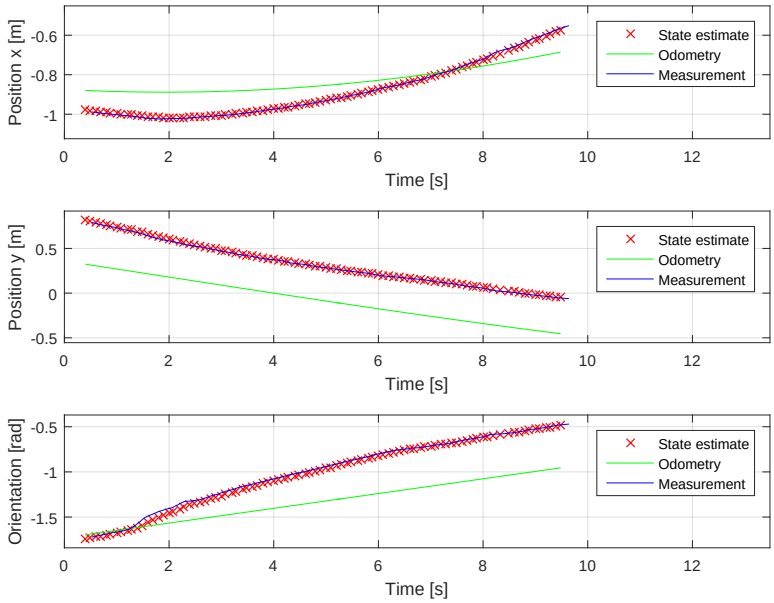

Figure 17: Test 8, velocity command 12% for left wheel and 19% for right wheel. q = 8, r = 32. Figure shows all three states with respect to time.

sensors and GPS as absolute position sensor to estimate state of the robot. Extended Kalman filter is a widely used filtering technique in estimating the localization of a robot that processes feedback from two or more sensors.

The presented project starts with preparing the odometry based physical model of Rollo, showing it's advantages and disadvantages with respect to the localization problem. Furthermore errors associated with odometry, divided into systematic and non systematic, were presented along with the UMBmark based square test technique to determine dominant odometry errors.

Next the preparation of the system and measurement models was conducted and the equations for the implementation of EKF were defined. EKF was first simulated and verified in Matlab. The final tests were conducted on Rollo and the performance of EKF was analyzed by varying process covariance matrix, Q and measurement covariance matrix, R.

For future work, it is proposed to first implement encoders feedback in the system and validate the EKF with working odometry. Even though the encoders feedback is currently unavailable in Rollo, the modeling of the system and software was designed for a swift and easy transition to odometry feedback. Next, another sensor should be introduced in the system since the motion capture system will be unavailable in field environment of the robot.

In regards to the mechanical system of the robot, it is recommended that a different material for the wheels should be used to improve traction and reduce skidding of the robot. The wheels of the robot also need to be properly aligned so that the odometry errors are reduced.