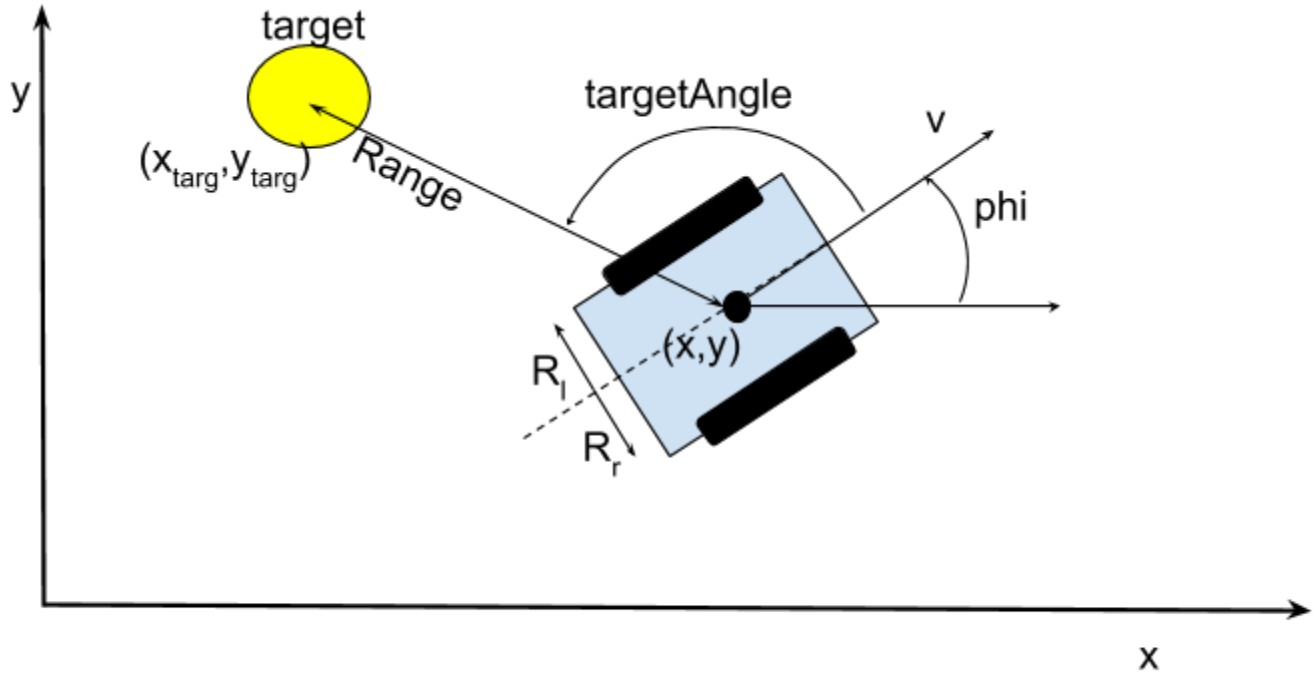
Robot localization based on Extended Kalman Filter (EKF)

The goal of this work is to estimate the location of FRC robot on the playing field using encoders, IMU, range and angular measurements to a target. The geometry is described in the following drawing, where a robot has two rows of wheels located at Rl and Rr distances from robot center.



The filter is based on EKF library that implements a generic extended Kalman filter using 4 basic classes:
KalmanFilter.java - the implementation of the filter itself
ObservationModel.jave - abstract class that defines the observation model
ProcessModel.java - abstract class that defines the process model
Matrix.java - utility that implements all the required linear algebra functionality, including basic matrix inversion.

The filter estimates the following 5 variables organized as a state vector

(1)
$$X = \begin{bmatrix} x \\ y \\ v \\ \phi \\ \omega \end{bmatrix}$$

Where (x,y) are the position on the field, v is the robot speed, ϕ is the angle of robot, and ω is the angular velocity.

Kalman filter requires to define the process equations, thus, how the state evolves in time. The process is described by a time derivative of the state vector. For each variable we define its time derivative as a function of other states or external inputs.

(2)
$$\frac{\partial}{\partial t}\begin{bmatrix} x \\ y \\ v \\ \phi \\ \omega \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\phi) \\ v \cdot \sin(\phi) \\ Acc \\ \omega \\ 0 \end{bmatrix}$$

Pay attention that acceleration measured by the IMU enters the process model as a derivative of speed (per recommendation of best practices for navigation filters)

The other part of Kalman filter is the observation, or measurement model. Thus, how state variables are reflected in measurements. The model assumes 5 measurements

1. Angle (from gyro)
2. Left track encoder speed
3. Right track encoder speed
4. Range to a target squared
5. Angle to a target relative to robot body

(3)
$$\begin{bmatrix} \text{yaw} \\ \Delta\text{LeftTrackSpeed} \\ \Delta\text{RightTrackSpeed} \\ \text{Range}^2 \\ \text{TargetAngle} \end{bmatrix} = \begin{bmatrix} \phi \\ v - R_l\omega \\ v + R_r\omega \\ (x - x_{targ})^2 + (y - y_{targ})^2 \\ \text{atan2}(y - y_{targ}, x - x_{targ}) - \phi \end{bmatrix}$$

The range and angle to the target are not implemented in the first version of the filter.

The filter requires not only the process and observation models, but also their derivatives by state. Meaning, how each state or observation is changing with respect to other states.

The derivative by state is called Jacobian and is a matrix, where each (i,j) cell means “derivative of equation i by state j”.

The Jacobian of the process model is given by the following matrix, which is the derivative of the process equation . Each cell is $\frac{\partial f_i}{\partial x_j}$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 0 & \cos(\phi) & -v\sin(\phi) & 0 \\ 0 & 0 & \sin(\phi) & v\cos(\phi) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The Jacobian of observation model is given by the following matrix, that is the derivative of the observation model

$$\frac{\partial H}{\partial x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -R_l \\ 0 & 0 & 1 & 0 & R_r \\ 2x & 2y & 0 & 0 & 0 \\ -\frac{y-y_{targ}}{(x-x_{targ})^2+(y-y_{targ})^2} & \frac{x-x_{targ}}{(x-x_{targ})^2+(y-y_{targ})^2} & 0 & -1 & 0 \end{bmatrix}$$

As mentioned before, the last two rows are not implemented, since range and bearing measurements are not yet supported.

Appendix

The filter as implemented in the library is of continuous time type.
For com[pleteness, a discrete time formulation is written in the sequel:

Discrete time:

$$\begin{bmatrix} x \\ y \\ v \\ \phi \\ \omega \end{bmatrix}_{k+1} = \begin{bmatrix} x + v \cdot \cos(\phi) \cdot \Delta t \\ y + v \cdot \sin(\phi) \cdot \Delta t \\ v + \Delta V_{acc} \\ \phi + \omega \cdot \Delta t \\ \omega \end{bmatrix}_k$$

$$F_k = \begin{bmatrix} 1 & 0 & \cos(\phi)\Delta t & -v\sin(\phi)\Delta t & 0 \\ 0 & 1 & \sin(\phi)\Delta t & v\cos(\phi)\Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_k = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta t & 0 & -\Delta t R_l \\ 0 & 0 & \Delta t & 0 & \Delta t R_r \\ 2x & 2y & 0 & 0 & 0 \\ TBD & TBD & 0 & 1 & 0 \end{bmatrix}$$