INTRODUCTION

The goal of this assessment is to create a map and localize the robot in that by using the measurements from odometer and radar.

METHOD

The technique I used is called FastSLAM which is a particle filter algorithm combined with Extended Kalman Filter. What this means is every particle has its own map with a 2x2 Kalman filter for each landmark. The basic steps for this are given as follows, Step 1: Prediction

In this step the robot's next state is predicted using its previous state and control data (odometry).

Step 2: Measurement Update

For each observed landmark, the correspondence (closeness) to existing landmarks in each particle's landmarks list is found, if none are found a new landmark is added to the list. But in our case the landmark names are given which can be taken advantage of. Next, the mean (Kalman filter state) and its covariance are updated according to standard Kalman filter equations.

Step 3: Importance weight

The importance weight for each particle is computed according to the closeness of the sensor measurements to expected measurements of a particular particle.

Step 4: Resampling

Particles with higher weight are more likely to survive the resampling process. Since each particle is representing the robot's location, the mean position of the surviving particles is the approximated robot's location at each timestep. At the end, the map is obtained from the particle closest to the approximated location.

ALGORITHM

- Predict next state(odometry) For each timestep:
  - For each landmark name:
    - For each particle:
      - New weights = Update particle and compute weight
        - Update particle: Old Landmark - Update state and variance, return weight.
          New Landmark – new state and variance, return weight=1.
- Resample (weights)
- Compute mean position of all particles, i.e., robot approximation. Find particle closest to this mean.
- Extract landmarks from this particle

KALMAN FILTER EQUATIONS

- Updating old landmark :

Kalman gain : $\mathbf{K = \sum_{old} + H^T (H \sum_{old} H^T + Q_t)^{-1}}$

Landmark state : $\mathbf{X_{new} = X_{old} + K (Z - h)}$

Landmark variance : $\mathbf{\sum_{new} = (I - KH) \sum_{old}}$

Where $\sum_{old}$ is the old landmark variance, Q_t is measurement covariance, H is the jacobian of expected measurement h. Expected measurement is landmark position from current sensor position. Z is current sensor measurement.

Note : Z and h can be either polar or cartesian depending on the measurement covariance matrix. If the matrix is covariance between range and bearing, then they must be in polar or the other way. I had converted measurements into range and bearing to obtain measurement covariance (I will attach the code).

Weight : $\mathbf{w = \frac{1}{2\pi\sqrt{\det(Q)}} \exp(-0.5 * (Z-h) * Q^{-1} * (Z-h)^T)}$

$Q = H_T (H \sum_{old} H_T + Q_t)^{-1}$

❾ New landmark:

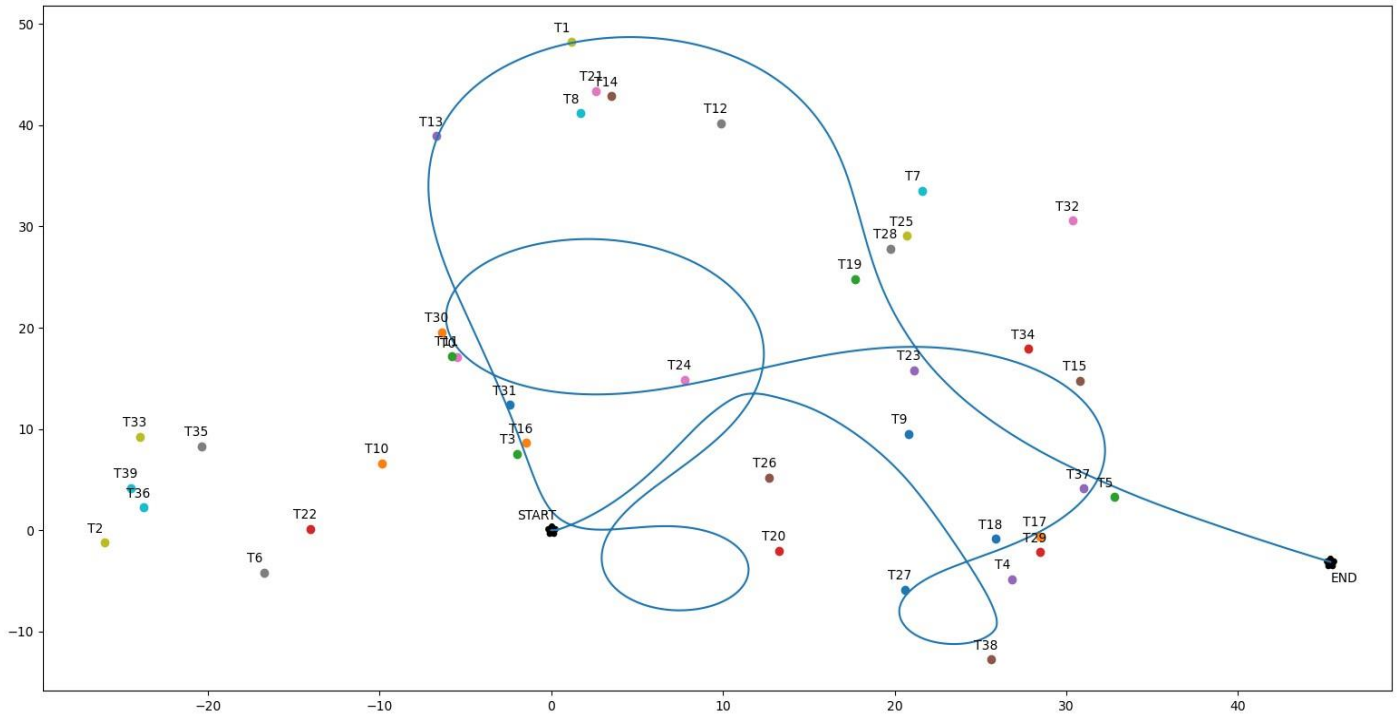
Landmark state is initialized as the global coordinates of current sensor measurement Z.

Landmark variance : $\mathbf{\sum = H^{-1} Q_t (H^{-1})^T}$

Weight : $\mathbf{w = 1}$

RESULT

I have run the algorithm for 25 particles and obtained the following plot of trajectory and landmarks,



FINAL THOUGHTS

- The method can be optimized furthermore by using a different language which offers features like pointers. This can be used for both algorithmic optimization and micro-optimization.
- Current time complexity of the algorithm is O(MN), where M is number of particle and N is number of landmarks.
- By using advanced features, it can be reduced to O(M logN).