Particle Filter

Project Basics

The goal of this project was to use a 2D particle filter to help localize a car placed in an unknown location. At first less accurate map data (like GPS) was used to initialize the car's location, then the new location was predicted based on velocity and yaw rate, we transformed the sensor observation points into map coordinates, associated the observations with the landmarks on the map and then computed the likelihood that a given particle made those observations based off the landmark positions in the map. These particles were afterwards resampled based on how likely a given particle was to making similar observations of the landmarks, which leads to a more accurate localization of the vehicle. The project also required a small amount of time for reaching good localization results.

Localization Algorithm

The general steps of the localization algorithm are given bellow. The alogirthm makes use of GPS information, range sensors like radar, lidar etc, landmark information and a global map.

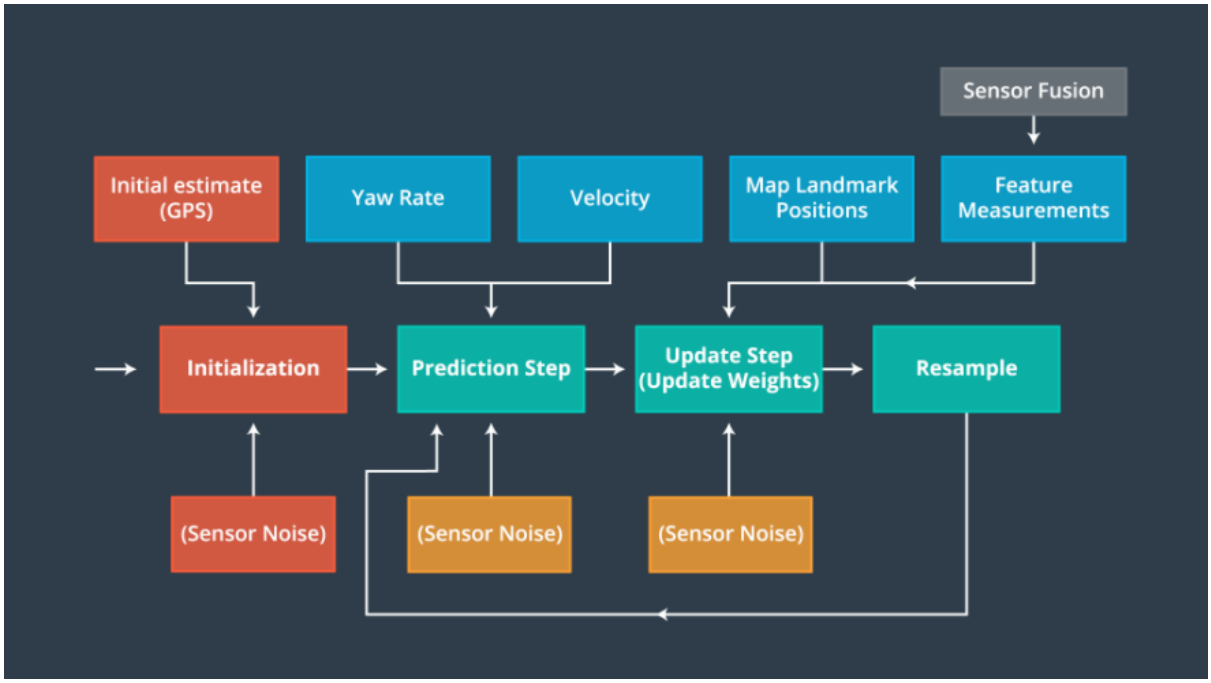
1. The self-driving vehicle is deployed in a global map, having different areas. In this map we have knowledge of different features present in the map which are not subject for change for a longer period of time (these are called landmarks). Such landmarks can be edges of buildings, signal posts, intersections and others. The map is updated often in order to add new features and refresh the locations of existing features.

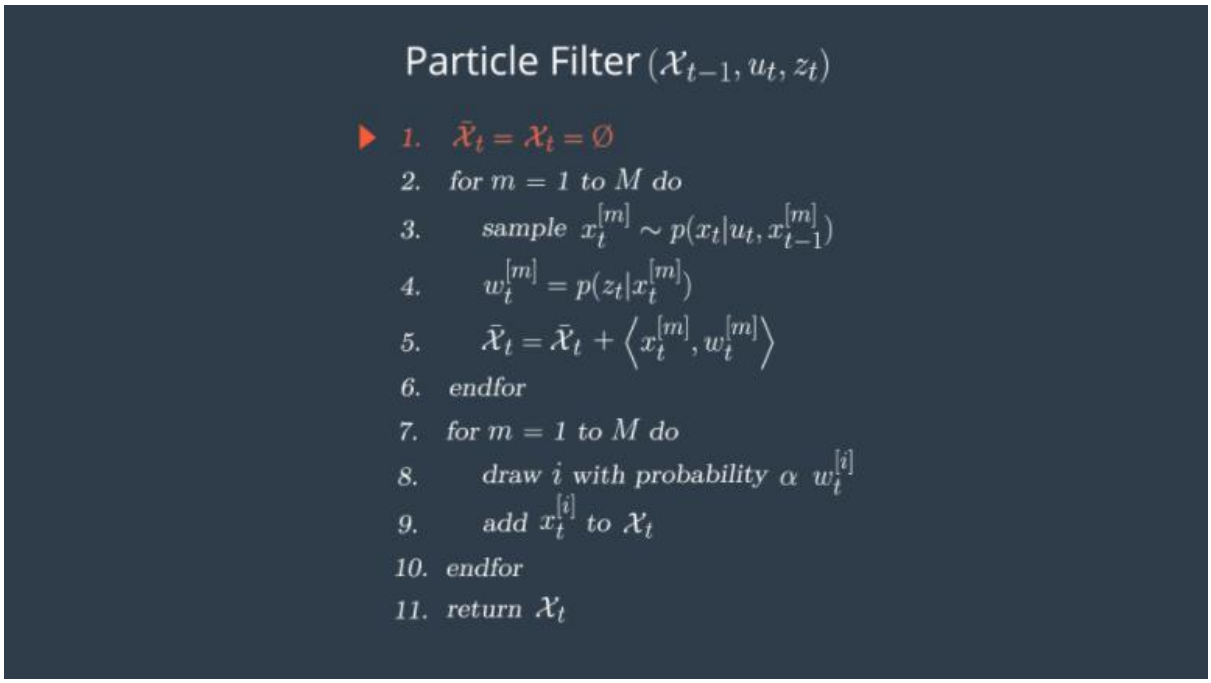
2. The GPS sensor is installed inside a vehicle in order to predict the locality in which the car is situated. Since the GPS is noisy it cannot be used alone to localize the car. Also just a portion of the map is taken from the vicinity of the car.

3. For measuring the distance to certain landmarks, lidar and radar sensors are installed. This helps further pinning down the location of the vehicle, since the relative distance to the landmarks can help us better position the car in the map. Since LIDAR and Radar sensors are noisy, we use a technique called particle filter to aid us with the localization process.

4. Particle filter uses the information stated above to localize the car in the map with high precision.

Implementation





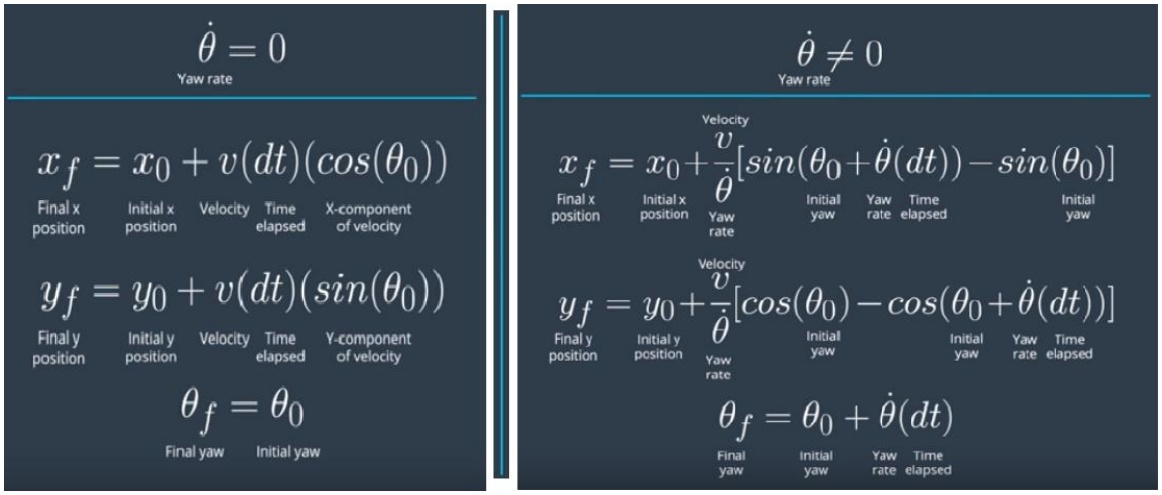
The major steps implemented in the C++ program are described below:

1. After receiving a noisy GPS measurement we initialize the position of the vehicle. The noisy measurement is modelled by a Gaussian distribution with standard deviation on the 3 measured values(x coordinate, y coordinate and orientation - theta). We initialize a set of particles to represent the location of the vehicle. We initialized a set of particles to represent the locations taken from the normal distribution with mean equal to GPS uncertainty and

standard deviation equal to GPS measurement uncertainty. The number of particles was a tunable parameter.

2. A global map having landmarks (represented by x and y positions in the map) is initialized.

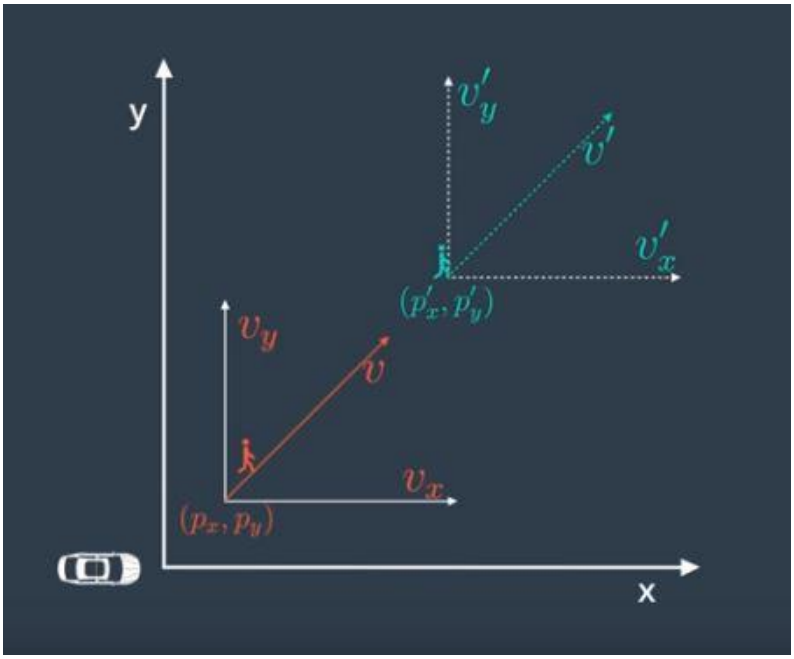
3. The next step represents the Prediction step. In this step the location of each particle at the next step is predicted. This is done using information of the control inputs (magnitude of velocity (v) and yaw rate (θ)) and time difference. The location update is done using the equations bellow:

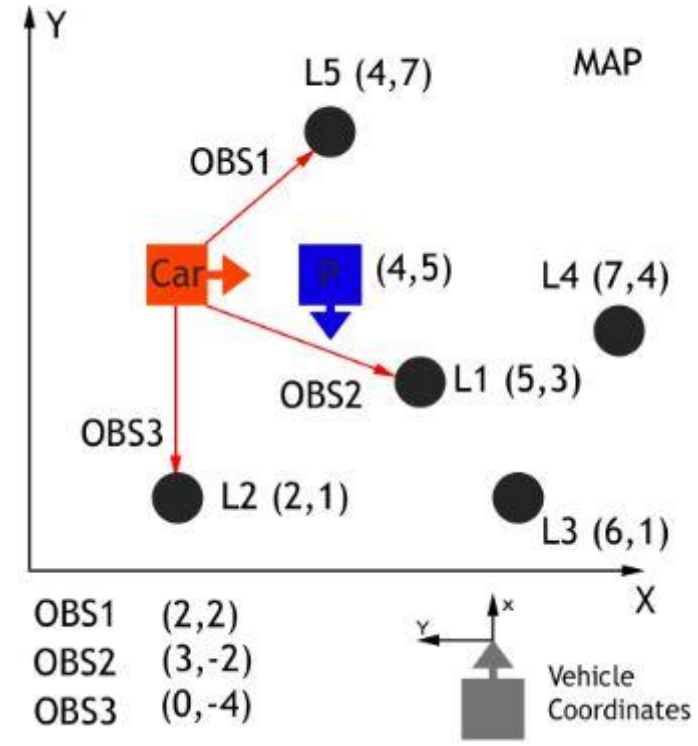


4. In the update step the particles are assigned with weights based on the results of their prediction. The update step is run for every particle and the weight of each particle is calculated. The main steps of the update stage of the particle filtering processing pipeline are underlined bellow.

- The vehicle uses range sensors to measure the distance from the landmarks and predicts the location of landmarks. The x, y coordinates are received along with their standard deviation. The observations have to be changed from the robot reference frame to the map reference frame. The

relation of the two reference frames is illustrated bellow.





The landmarks are depicted with annotations L1-L5, observations are annotated with obs1-obs3 and the ground truth is shown with red while the prediction is with blue.

- To map the observations to a global reference frame, homogenous coordinate transformations are done using the equations below

Homogenous Transformation

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_p \\ \sin\theta & \cos\theta & y_p \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}$$

$$x_m = x_p + (\cos\theta \times x_c) - (\sin\theta \times y_c)$$

$$y_m = y_p + (\sin\theta \times x_c) + (\cos\theta \times y_c)$$

In the equations above xm,ym represent the result of the transformation, xc,yc are observations in the vehicle coordinate system and xp and yp the location of the particle in the global map coordinate system.

- The landmarks are filtered to retain only those that are in the range of the particle. This is done since all landmakrs cannot be in the range of a vehicle at a given point of time.
- After the filtering each observation is mapped to a landmark using a nearest neighbour approach. We perform associations between an observation and all landmarks. The landmark with the smallest Euclidean distance is associated to the corresponding observation.
- After the association step, the weight of the particle is calculated using a Multivariate Gaussian distribution. The total weight of the particle is the product of probabilities calculated by multivariate Gaussian formula for the observations associated to landmarks.
- The weight of the particle is a measure of proximity to the ground truth of the vehicle. The higher the weight, the more accurate is the particle prediction.

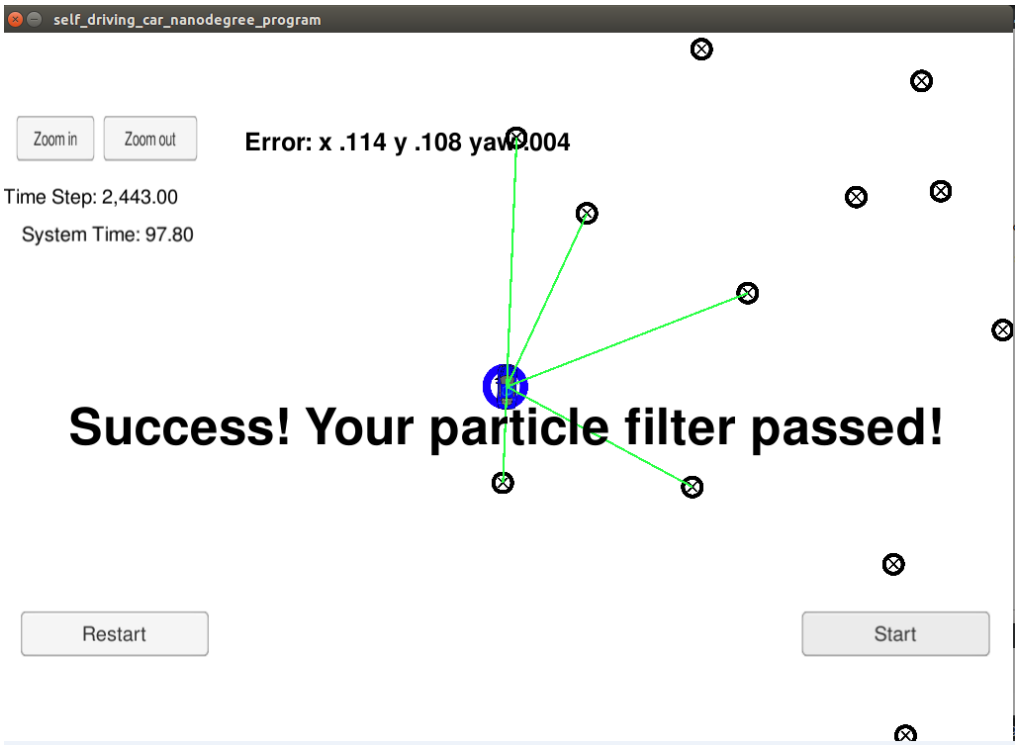
6. After the update step, the resampling of the particle is done. In this stage the particles with higher weight are kept while the particles having a smaller weight are eliminated. The particles with highest weight are chosen.

7. The location provided by the particle having the highest weight is compared with the ground truth is calculated and so the error of the system is calculated.

8. After all the stages are implemented the program is tested and the system error and running time are noted.

Output

The result of the particle filter when run with a number of 120 particles.



Steps for building the project

Dependencies

- cmake >= 3.5
- make >= 4.1(mac, linux), 3.81(Windows)
- gcc/g++ >= 5.4
- uWebSockets
- Simulator.

Running the project in Ubuntu

1. Execute every step from ./install-ubuntu.sh. This will install gcc, g++, cmake, make and uWebsocketIO API.

2. Build and run project by running ./build.sh.

3. In case of error, manually build the project and run using: a. mkdir build && cd build b. cmake .. c. make d. ./particle_filter.

4. Run the Udacity simulator and check the results