CarND-Kidnapped-Vehicle-Project

GOAL :

```
* Design particle filter that help the car to determine its location using :
* a map,
* a (noisy) GPS estimate of its initial location, and
* lots of (noisy) sensor and control data.
```

BACK GROUND :

1. If there is something we want to know or
2. If We can measure something, related to what we want to know or
3. if We know something about the relationship between the measurements and what we want to know

Particle Filter is a technique to systematically handle such situation

- The underlying stesps:
    1. Update the weights using the measuremetns
    2. Resample with respect to the weights
    3. Propagate the particles in time using a model

Inputs to the Particle Filter:

The particle filter will be given a map and some initial localization information (analogous to what a GPS would provide). At each time step your filter will also get observation and control data. the position of landmarks (in meters) on an arbitrary Cartesian coordinate system. Each row has three columns : x position y position landmark id

Simulator will send control data : car speed and yaw_rate sense-x, sense_y, theta

sense-x, sense_y, theta is used to initialize filter

Project Steps :

1. A global map of different areas is constructed, in which the self driving vehicle is to be deployed. This map contains information about different location of 'landmarks'.

1. Initialization function: estimate position from GPS data using particle filters, add random noise)
2. Prediction function : predict position based on adding velocity and yaw rate to particle filters, add random noise)
3. Update Weights function :
    - Transformation of observation points to map coordinates (given in vehicle coordinates)
    - Association of landmarks to the transformed observation points
    - Calculation of multi-variate Gaussian distribution
4. Resample function - Resamples particles, with replacement occurring based on weighting distributions Optimizing algorithm - Performance within required accuracy and speed

Criteria #1 : Does your particle filter localize the vehicle to within the desired accuracy?

With a lot pain in debugging, the Particle filter meet the error rate expected. error : x .130 y.122 yaw .004

Criteria #2 : Does your particle run within the specified time of 100 seconds?

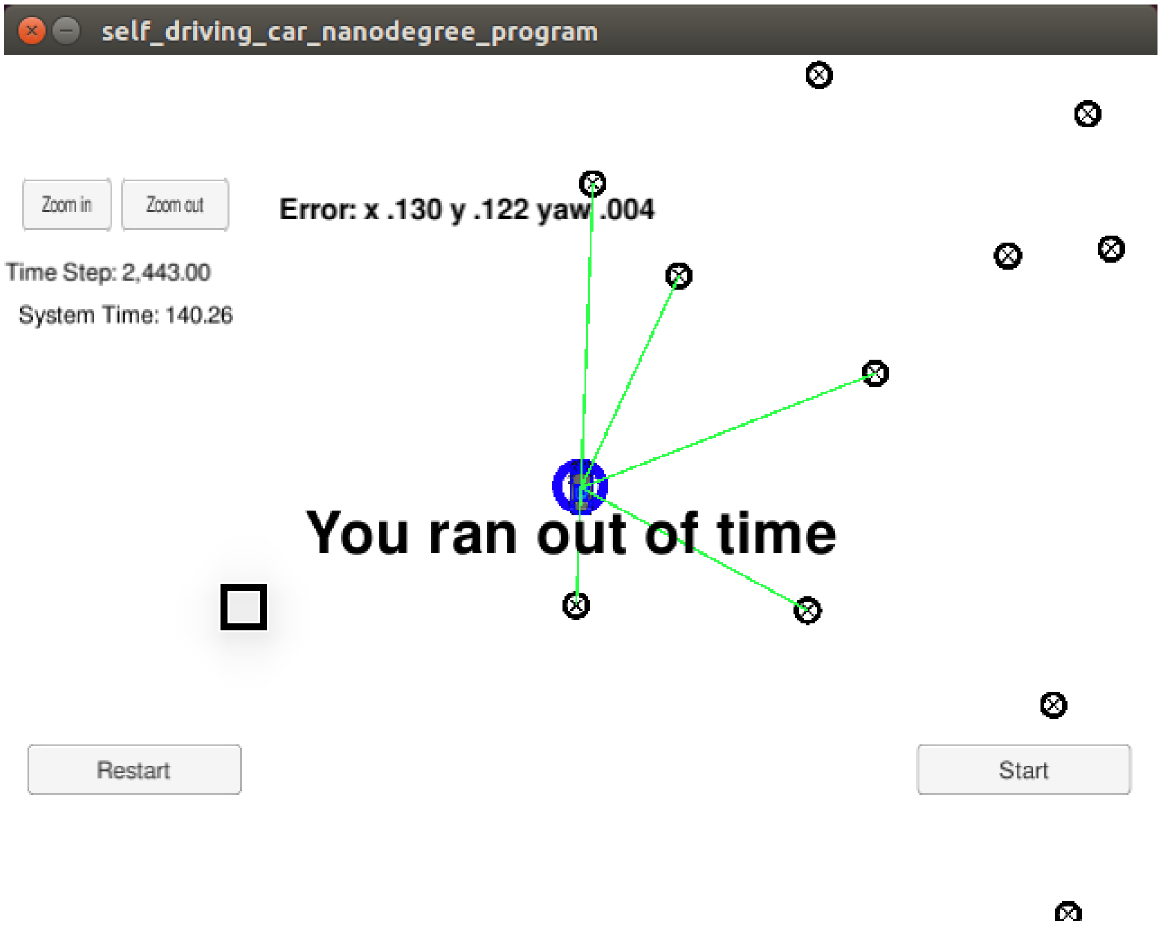
However, this particle filter barely missed the timing 122sec vs 100 sec.

Criteria #3 : Does your code use a particle filter to localize the robot?

Step by step go through the lectures, sample code and google search, I was able to piece together my first particle filter

In [6]:
```
from IPython.display import Image
Image(filename="./ParticleFilterResult.png")
```

Out[6]:



Error: x .130 y .122 yaw .004

Time Step: 2,443.00

System Time: 140.26

You ran out of time

In [ ]:
```

```