EDL Project Proposal - Group B7

Radar tracking Algorithm

Team members : **Preetam Pinnada** **170070042**

Ameya Anjarlekar **170070013**

Meruva Anjaneya Prasad **170070052**

Faculty mentor : **Prof. Jayanta Mukherjee**

1 Abstract

Radar tracking algorithms are important for detecting a wide range of air, water and land-borne objects using a radar. However, the signal detected by the receiver antenna gets corrupted by noise and disturbances due to terrain, weather, etc. Hence, we plan to develop an algorithm for the efficient estimation of the position of the target. We propose to introduce an automatic radar tracking system. It will receive multiple observations from Multistatic radar. Accurate coordinates will be obtained using a Particle filter as a nonlinear predictor for data fusion and prediction. The data will be loaded on a microprocessor using an SDK chip. It would further be processed using the developed algorithm. Finally, the accurate position of the coordinates will be displayed using a GUI on a monitor.

2 Motivation and Project Application

Radar tracking algorithms are important for accurate detection of a wide range of objects using a radar. A radar consists of a transmitter and a receiver antenna. Some of the waves transmitted by the transmitter get incident on the object and reflected back. These waves will contain information about the target object. However, the detection of the properties of the target object is a non-trivial task. Moreover, the signal detected by the receiver antenna gets corrupted by noise and interference due to terrain, weather and other disturbances. Hence, there arises a need for processing the data obtained by the receiver so as to extract the target features from the received wave which is corrupted by noise and interference. A particle filter is used as a nonlinear predictor for data fusion and prediction. The Filter receives the observation from the radar receiver or combination between them. The filter processes this data in the presence of noisy observations to estimate the correct path of the target.

Radars have got a wide application in military operations. They are used in air, naval and ground for defense purposes. They are also used for tracking, surveillance, and detection of the target. Radars are used for safety controlling the air traffic. It is used in the vicinity of airports for guiding airplanes for proper landing in adverse weather conditions. A multistatic radar system contains multiple monostatic radar components with a shared area of coverage. Its major difference from individual radar geometries is the added requirement for some level of data fusion to take place between component parts. As it contains multiple radar components, it can be used to provide a wide area of coverage for detection, tracking, and surveillance. The algorithm developed in the project will improve the fusion and prediction estimate of the non-linear moving object in the presence of measurement errors.

3 Project Objectives

i) Understand and solve the problem of state estimation using particle filtering

ii) Get familiar with Raspberry Pi and implement particle filter on it

iii) Design and aesthetics involved in product development viz. GUI, casing

4 Project Description

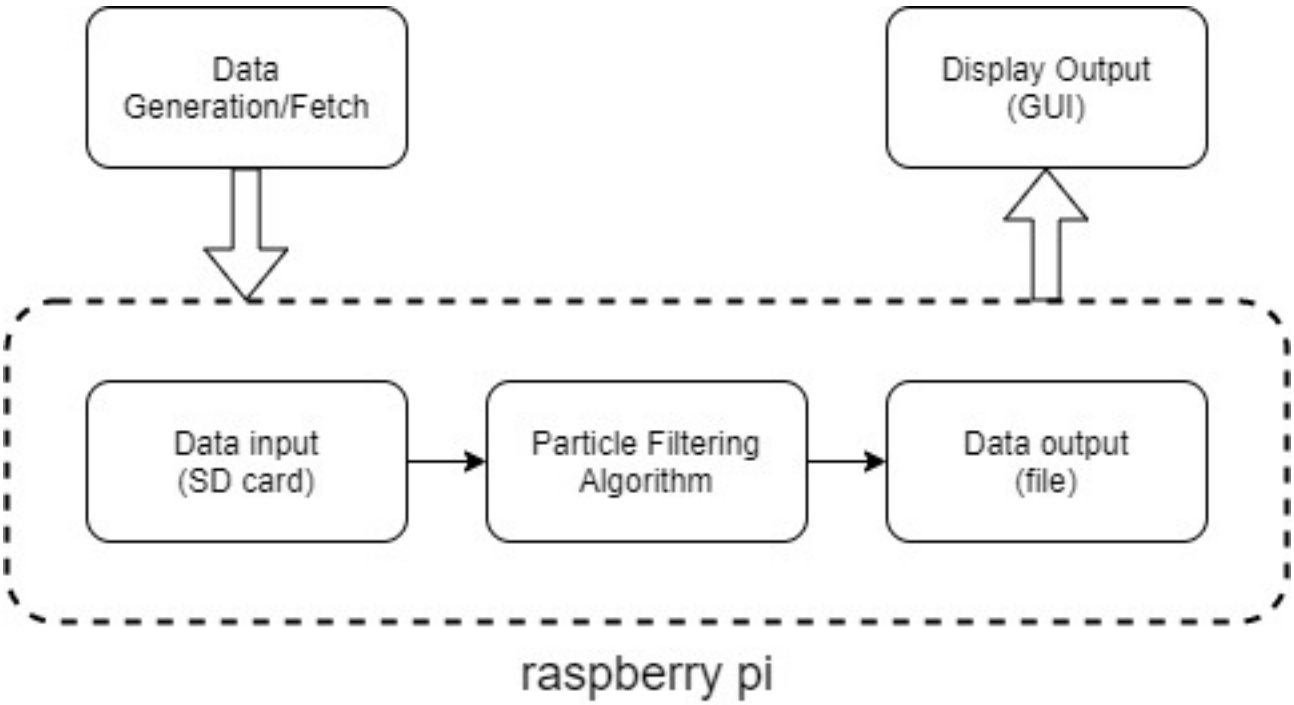

Figure 1: Block Diagram

- Data Generation is done manually through some known algorithms.
- The data generated is stored in an SD card which is given as an input to raspberry pi.
- The rpi takes data from sd card and performs particle filtering algorithm using the program built into it
- The output of the algorithm is stored in a file which is then displayed on to customized GUI.

5 Project Implementation Details

5.1 Dataset Generation

Authentic datasets for multistatic radar are classified as they concern mostly with sensitive areas like radar, air-traffic control, etc. Hence, we will be modeling the radar surroundings and thus generate a dataset using the model.

We describe our monostatic radar system using a nonlinear model. A stochastic state-space model is used to generate the samples. Let [x_1,x_2,.....x_n] represent the states of the system and [y_1,y_2,.....y_n] represent the observed variables. The state-space system is thus represented by:

$$x_t = f(x_{t-1}) + n_t \tag{1}$$
$$y_t = g(x_t) + v_t \tag{2}$$

Where n_t and v_t are zero-mean Gaussian random variables with std deviations b and d.
We consider the position to vary linearly while the observations vary in a nonlinear manner. For dataset creation, we consider

$$f(x_t) = a \cdot x_t \tag{3}$$
$$g(x_t) = c \cdot e^{x_t} \tag{4}$$

These equations are used to simulate about T = 100 observations.

5.2 Problem to be dealt with and expected output

For the parameter estimation, we implement a Particle filter as a nonlinear predictor for data fusion and prediction. Particle filter improves the estimate of a non-linear moving object in the presence of high measurement errors.
Given the observed variables and the parameters a,b,c,d, we need to accurately determine the state of the system i.e. the position of the particle at every time step. A particle filter uses the previous observations along with the current observation to correctly estimate the state of the system or in our case, the position of the object. We consider p = 100 particles for estimating the position of the object.
Thus, we expect the accuracy of the filter to improve with time.

5.3 Particle filter implementation

5.3.1 Initialization

We assume initially that the object is uniformly distributed. Thus, the initial position before any observation is taken by sampling p=100 particles from the uniform distribution. The initial position is the mean of all these samples.

5.3.2 Calculation of likelihood

For finding the location of the object, we update the previous location of the particle by adding a random Gaussian perturbation to it. This is just a random intial guess. We then calculate the likelihood of the object to be in given position provided by the location of the particle. The likelihood for particle i is calculated by:

$$L^i = \frac{exp(-(y^i_{x-pred} - y_{observed})^2)}{\sum_{i=1}^{100} exp(-(y^i_{x-pred} - y_{observed})^2)} \tag{5}$$

5.3.3 Resampling

Using this new likelihood, the particles are further resampled to estimate the position of the object after the first observation. The sampling is done from a distribution which is Gaussian wherein the probability of the particle at a coordinate is given by a histogram which is constructed using the calculated likelihood.

5.3.4 Displaying the predicted location

After resampling, the predicted location is the mean of the locations predicted by each particle. This is further displayed on a display screen using the raspberry pi interfaced to the display.

5.4 GUI

The above mentioned particle filtering algorithm will be implemented on Raspberry Pi Zero.The output of the algorithm is displayed on the GUI (viz. TFT displays available in lab).The algorithm will be coded in python language and place it in GUI APP.
When it comes to GUI application,it contains a desktop icon to launch the application.The application contains interface to choose the "Raw data file" and options to apply the coded

algorithm on the data.The output of the algorithm is stored in a text file in known location(internal data of app) which is then accessed to plot the same using plot option. In diagram Below, each buttons self explains about their functionality

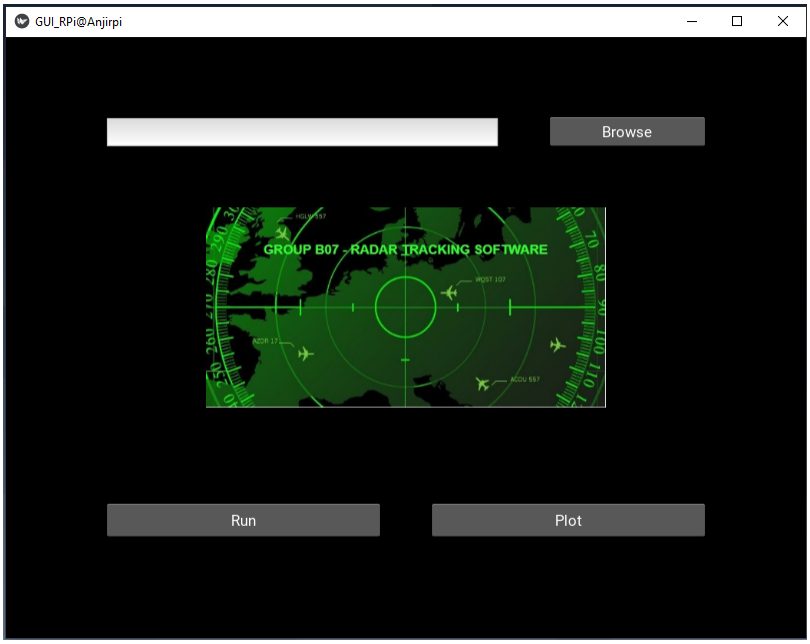

Figure 2: This is the present version of our GUI

We would like to improve and add some sophisticated features depending upon time available.

5.5 Thermal management

From our observation working with Raspberry Pi, normal computations generate hot surfaces with temperatures ranging above 50 degrees centigrade. Thus, while planning to develop a closed compact module, thermal management using heat-sinks and fans is essential for endurance of product. We propose to install a cooling fan and related circuitry to control it using raspberry pi GPIO pins, which shall power on/off the fan based on the operation temperatures of the microcontroller. Further an extra outer casing is made to make it a complete enclosed product.

Towards the end we propose to develop a standalone compact particle filtering module which takes in data and tracks an particle on GUI. Hardware design also includes heating considerations of various components in the circuit.

6 Milestones

Evaluation 1:

- Literature survey regarding particle filtering algorithm and its variants ✓
- Understanding and configuring the Raspberry Pi development environment ✓
- Creation of GUI to display the object location ✓

Evaluation 2:

- Generation or procurement of radar data ✓
- Implementing the particle filter algorithm ✓
- Interfacing the Pi with TFT display ✓

Final evaluation:

- Implementing the algorithm on Pi
- Design of a thermal management system ✓
- Design and production of a compact module encompassing the display and the Pi

7 Work division

1) **Anjaneya Prasad Meruva**: Design of an app along with GUI interfaced with the raspberry pi module. Design and creation of a compact standalone module encompassing the display and the raspberry pi.

2) **Ameya Anjarlekar**: Generation of radar dataset and the implementation of a modified particle filter algorithm on a python environment to accurately estimate the position of the object using the generated dataset.

3) **Preetam Pinnada**: Configuring the Raspberry Pi environment such that the Pi takes in the dataset and using the code displays the position of the object using a GUI. Design of the thermal management system.

8 Bill of Materials (with approx. cost)

- Raspberry Pi Zero kit : Rs 1699 (available in lab)
- TFT LCD display : Rs 1,050 (available in lab but it is not touchscreen)
- Pi Fan : Rs 129 (available in lab)

9 Work done till March 13

9.1 Temperature control

Designed an automated temperature control circuit, printed the corresponding PCB and implemented the circuit. The Eagle schematic is shown in the figure below. Raspberry Pi

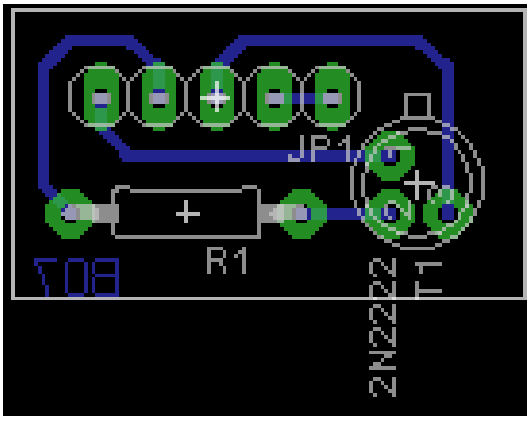

Figure 3: PCB layout

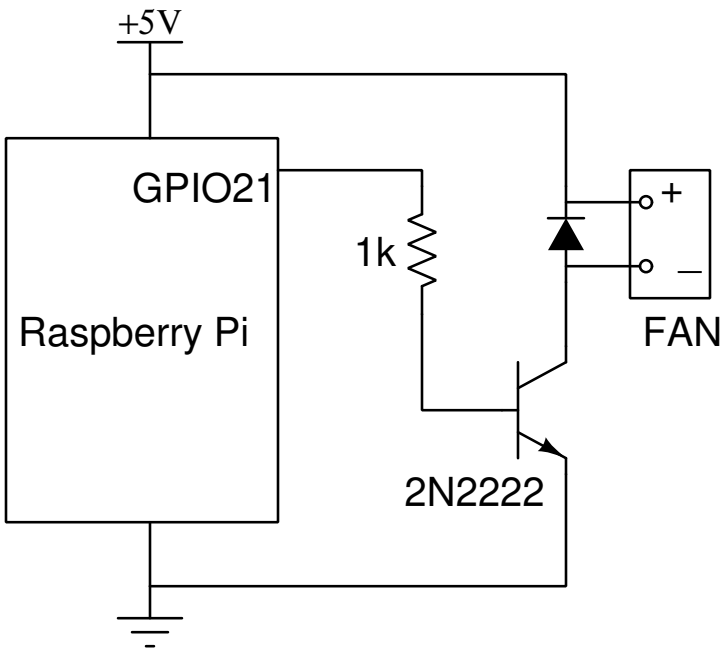

Figure 4: Temperature control circuit schematic

supplies 3V but the Pi Fan requires 5V to operate. So we've introduced a transistor into the simple circuit of powering the fan.
Later the Pi was coded such that whenever the CPU temperature rises above 50 degrees

centigrade, it'd turn on the pins connected to the fan. And as soon as the temperature dropped below 45 degrees the pins were turned off.

9.2 Particle Filter code

We first generated the data for the particle filter. We ensure that the path of the particle is highly random and try to estimate this path using the particle filter algorithm.
Results for the 1D case:

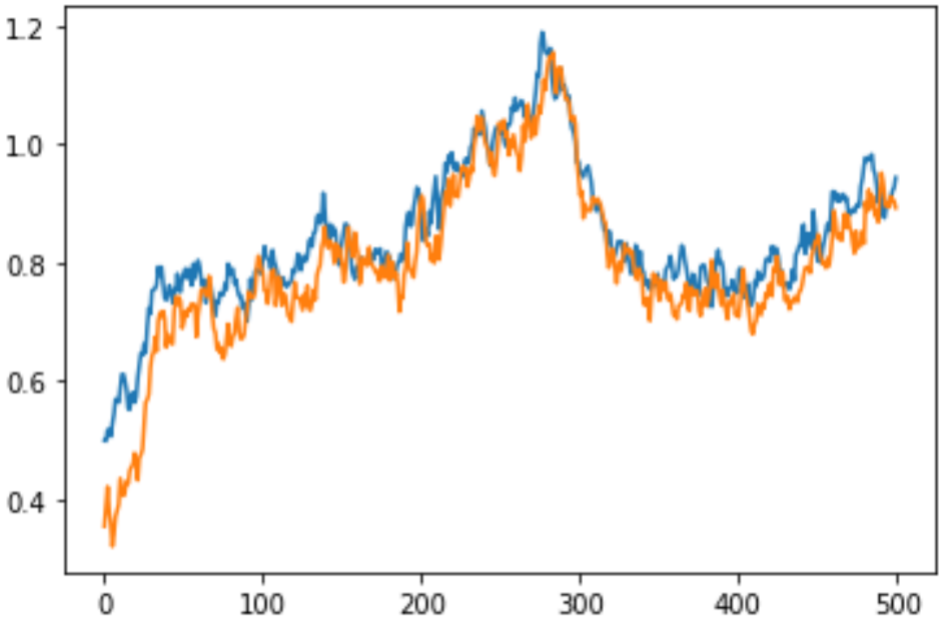

Figure 5: 1D path estimation

The blue line shows the actual coordinates while the orange line shows the predicted coordinates. The y-axis represent the coordinate of the point in 1D space while the x-axis represents the observation no. We can see that the estimate improves as the observation no. increases.

Results for the 2d case:

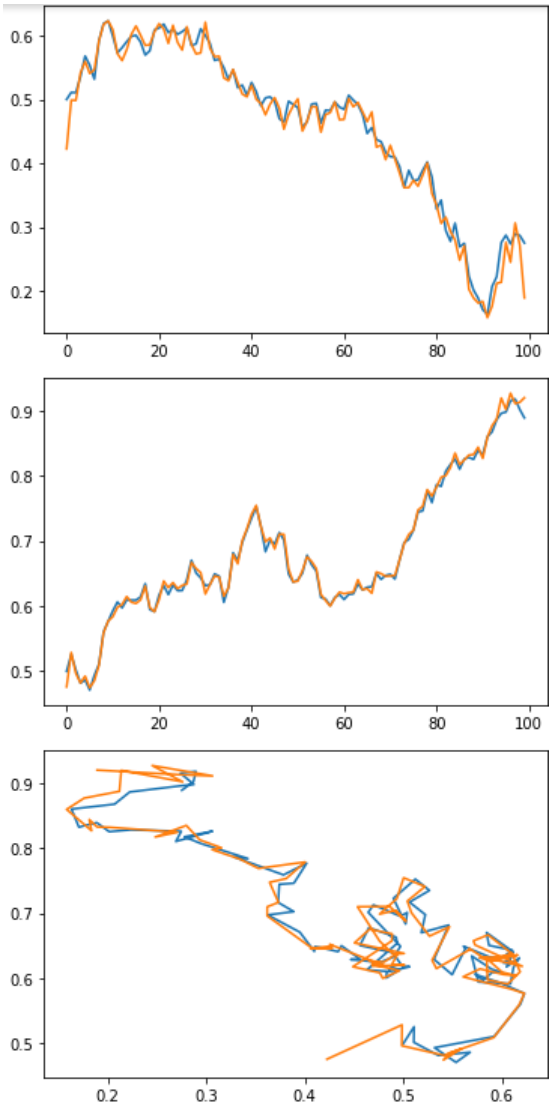

Figure 6: 2D path estimation

The first graph shows the estimate of the x-coordinate, the second graph shows that of the y-coordinate while the third graph shows the path of the object in the 2d plane. We choose a highly random path and predict the path so as to highlight the robustness of the algorithm. We also consider the observations to be highly noise with the std deviation of the noise to be around 10% of the average value of the observations.
The codes for data generation and path prediction are available **here**. The link is also included in the references.

9.3 GUI

When it comes to GUI improvements, we added code for popping up windows when buttons shown in main window (Figure 2) is clicked. When Browse button is clicked, a window will pop up as shown in Figure 7, in which we choose our Raw data file which needs to processed. When "RUN" button is clicked, it performs the algorithm coded in the application on the raw data file that we choose in figure 3, and output data is stored in a temporary file. When "PLOT" button is clicked, it reads the data from that temporary file and plots in a separate window. A sample plot of tracing a circle is shown in Figure 8.

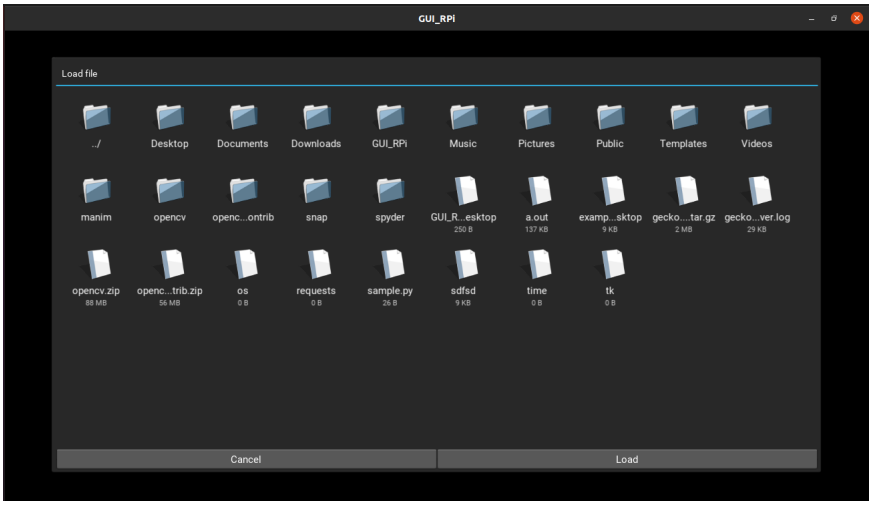

Figure 7: Window when Browse button is clicked.

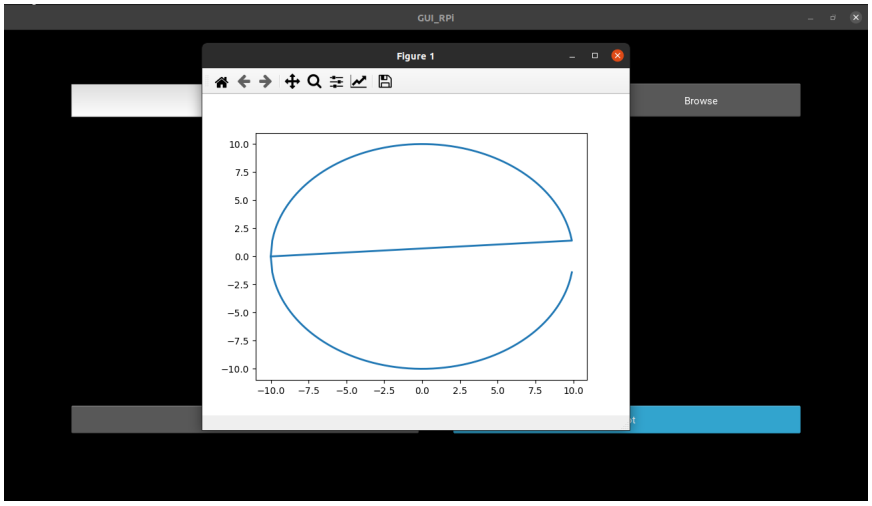

Figure 8: window when Plot button is clicked.

10 References

- Particle filter explained
- A Particle filter for multistatic radar tracking

11 Appendix

Github repository link