KTH - Royal Institute of Technology

Applied Estimation Project

Basic Aircraft Autopilot System

Nizar Gandy Assaf Layouss
KTH Id: 851122-T510

23/01/2012

Abstract

Navigation Systems nowadays are crucial in any vehicle regardless of being a car, aircraft, robot or a satellite and other types of vehicles. This report exposes an implementation of a basic aircraft autopilot using a particle filter for tracking and estimating the aircraft position and altitude and for applying the correspondent actuator to fly successfully the waypoints given in a map.

1 Brief Introduction to Air Navigation

To make the air navigation safer, airspaces contain airways where aircrafts can fly and track to move from the origin airport to destination airport. An airway is an invisible route that lies between two navigational aids (VOR,NDB,FIX..), that for the scope of this project, is simplified to a waypoint and to be able to reach the destination an aircraft has to track and fly through various airways to reach the destination.

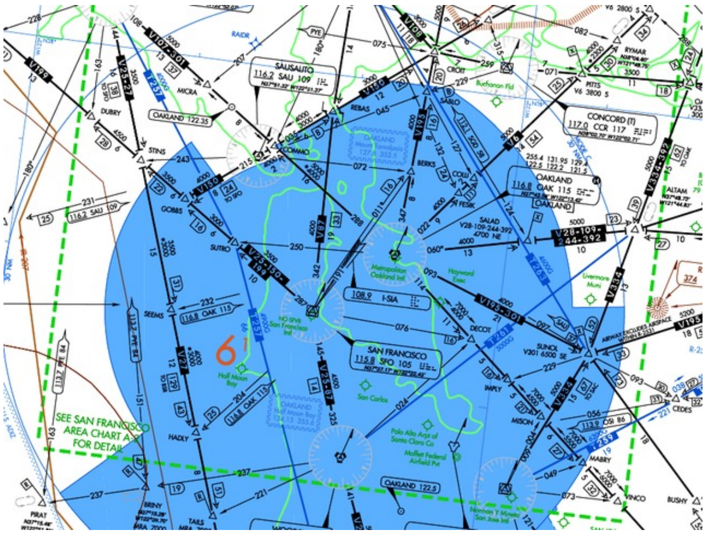


Figure 1: Chart of Airways

The autopilot system in any aircraft is composed by two subsystems:

- LNAV: Lateral Navigation and is responsible for tracking and maintaining the aircraft on the right course.
- VNAV: Vertical Navigation and is responsible for tracking and maintaining the altitude of the aircraft correctly according to the airway.

Navaids can be over-flown at different altitudes giving different types of airways (High Altitude Airways, Low Altitude Airways..) resulting that the same airway can be flown at different ranges of altitude and that each navaid(waypoint) could be used for different airways.

In this project a different perspective was taken regarding the waypoints. The waypoints not only have a position(coordinates) but also a specific altitude thus restricting the aircraft to overfly the waypoint unless the aircraft has the corresponding altitude. In other words, the aircraft flies through a waypoint and does not overfly the waypoint resulting in a more restrictive and interesting situations like the case of a steep climb.

2 Aircraft Model

The aircraft rotates in three dimension and the three principle aircraft flight controls (actuators) that affects them:

- Ailerons: used to control the banking/roll of the aircraft around the longitudinal axis.
- Elevators: used to control the pitch angle of the aircraft around the lateral axis.
- Rudder: used to control the yaw rotation around the vertical axis of the aircraft.

In this project it's assumed directly the result of applying the first two of these flight controls that is the angular velocity in the heading of the aircraft and the angular velocity of the pitch in the aircraft for complexity reasons of implementing the mapping of the actions of the flight controls to angular velocities before mentioned. Thus in this project there are only changes in the pitch angle and in the heading angles.

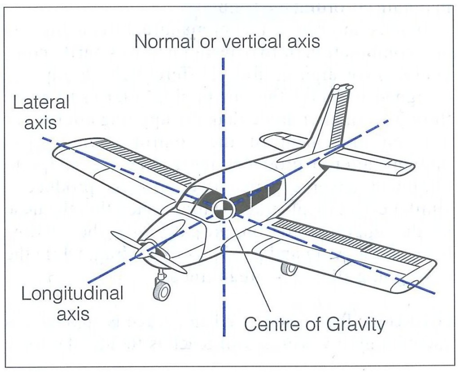


Figure 2: Aircraft Principal Axis

3 Particle Filter Model

The particle filter used in this project is an extended version of the particle filter of the Lab2. The resample step was implemented using the systematic resampling method to achive more robustness in the resampling step. At any moment an aircraft is represented by its coordinates, altitude, heading and pitch angle thus fulfilling the condition of a markov process and completeness of the state space.

$$X = \begin{bmatrix} x \\ y \\ z \\ \theta \\ \alpha \end{bmatrix}$$

Where θ corresponds to the heading angle, α the pitch angle and z the altitude of the aircraft.

Given that the airways are lines, the best model would be the linear motion model and model the aircraft as a three dimensional vector that its norm is the speed resulting in the following motion model:

$$u_t = \begin{bmatrix} dx_t \\ dy_t \\ dz_t \\ d\theta \\ d\alpha \end{bmatrix} = dt \begin{bmatrix} v_{t-1} cos X_{t-1,\theta} \\ v_{t-1} sin X_{t-1,\theta} \\ v_{t-1} sin X_{t-1,\alpha} \\ \omega_\theta \\ \omega_\alpha \end{bmatrix} \tag{1}$$

In the motion model, the aircraft adapts it's velocity dynamically depending on the distance left until the waypoint. The same applies with both angular velocities of pitch and heading.

As for the measurement model, basically the aircraft measures at each timestep t the range, heading angle and the pitch angle respect all the waypoints thus the waypoints work also as landmarks to keep track of the aircraft position:

$$\hat{z}_t = \begin{bmatrix} distance \\ relative\ heading \\ relative\ pitch \end{bmatrix} = \begin{bmatrix} \sqrt{(W_{i,x} - x)^2 + (W_{i,y} - y)^2 + (W_{i,z} - z)^2} \\ atan2(W_{i,y} - y, W_{i,x} - x) - \theta_t \\ atan2(W_{i,z} - z, distance) - \alpha_t \end{bmatrix} + Q \tag{2}$$

Where $distance = \sqrt{(W_{i,x} - x)^2 + (W_{i,y} - y)^2 + (W_{i,z} - z)^2}$.

4 Simulation

Some of the parameters in this application are set automatically given a map. The process noise is set automatically depending on the resolution/total runtime but the number of particles and the measurements noise are set by the user depending on his needs and tha map size to tune correctly the particle filter. It's crucial to introduce high noise to the measurements model given the known problem of the particle filters to operate correctly with accurate measurements and this was one of the reasons that this project took much more time than expected.

The code is run by calling the main function *runAutopilot* and it receives the map file, initial pose, total time to run and a verbose flag where 0 means show only the simulation, 1 shows the particles and 2 will show mean error results.

```
runAutopilot('map3.txt',[0 0 0 0 0]',100,1);
```

The time parameter is used as a resolution reference for the measurements along the path so the bigger the time is generally the better is the navigation for the same map. The process noise initialization depends on the time as here the time acts like a resolution of the path and indeed it's initialized a matrix of $5x5$ where each element correspond to $1/(total_time^2)$.

The map file format is just specifying in order the three coordinated of the first waypoint, next line the second and so on.

```
1 1 0
2 2 0
3 2 1
4 2 2
5 2 1
```

Given the map and the initial pose, the program creates a path from the initial pose passing through the differents waypoints automatically by calling the function *create_path*. This function returns the simulated measurements and pose at each time step of the aircraft navigationin a way that the last time step is over the final waypoint according to the order of the waypoints in the map file. Specifically what it does is to convert each airway to a three dimensional vector and samples points along the vector according to a given resolution that depends on the time and the distance of the airway.

5 Results

In the data folder there are the two basic maps with numerous waypoints to simulate the autopilot system although any map file could be created manually and moved to the data folder within the project.

5.1 map1.txt

Results of simulating the autopilot with map 1

```
t=100;
n_particles=200;
Q = diag([0.5;0.5;0.5]);

map1=[1 1 0
      2 2 0
      3 2 1
      4 2 2
```

```
       5  2  1];
```

The total error of the simulation:

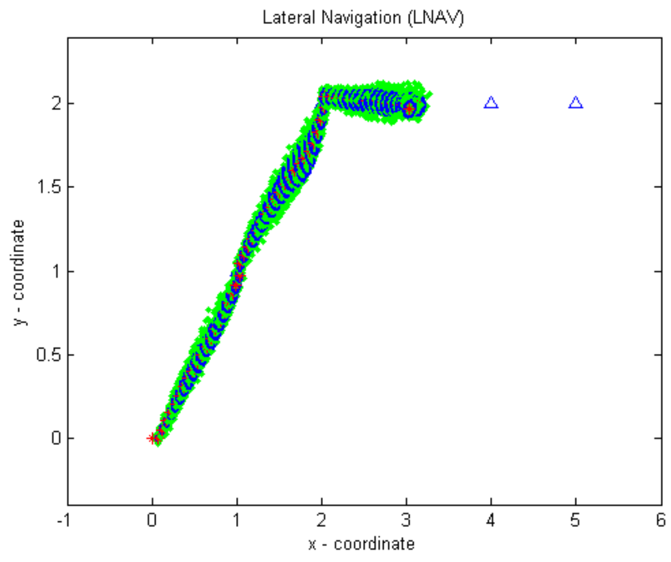
```
Mean LNAV Error x 0.069948
Mean LNAV Error y 0.008053
Mean VNAV Error z 0.015976
```

The reason that the x direction had the biggest mean of error is because when reaching the third waypoint and given the steep climb, the aircraft doesn't reach the same altitude of the waypoint thus starts doing a hold pattern over the waypoint until it reaches the waypoint altitude and then continues to the next waypoint. It's to see the situation described before in the subfigure 5b. In the final figures, the black line corresponds to the exact airway.

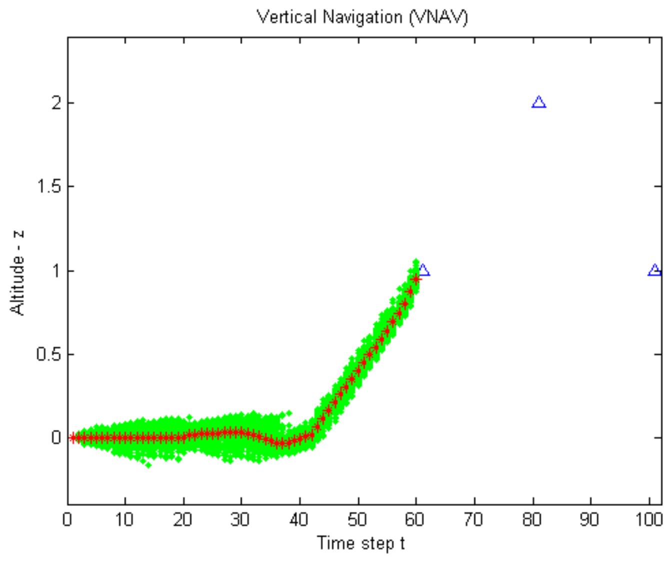
5.2 map2.txt

Results of simulating the autopilot with map 2:

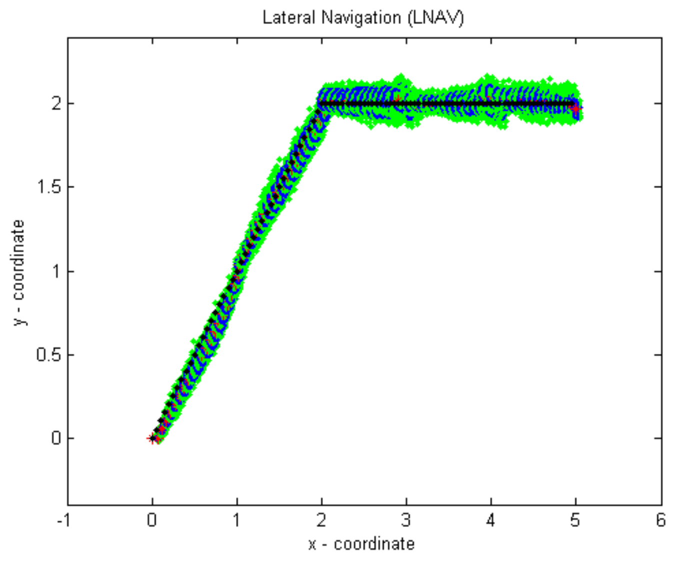
```
t=100;
n_particles=200;
Q = diag([0.5;0.5;0.5]);

map2=[5  0  0
     10  5  3
     10  10  5
     5  10  4
     2  10  2];
```



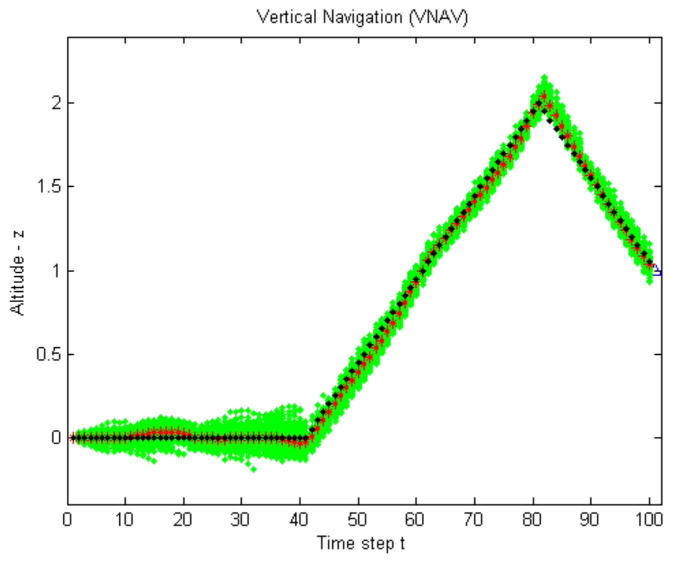
(a) Autopilot Lateral Navigation at t=60



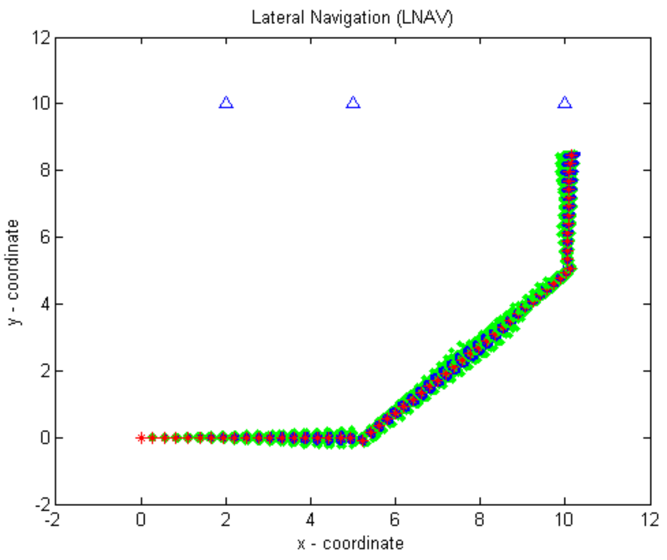
(b) Autopilot Vertival Navigation at t=60



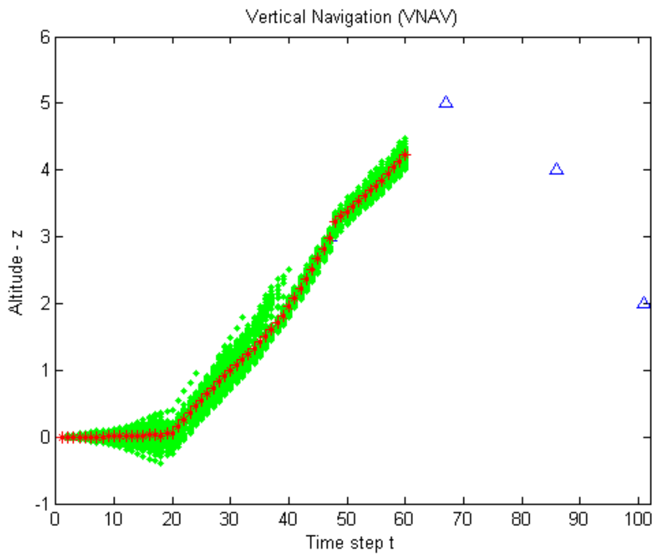
(a) Autopilot Vertival Navigation at the end



(b) Autopilot Vertival Navigation at the end



(a) Autopilot Lateral Navigation at t=60

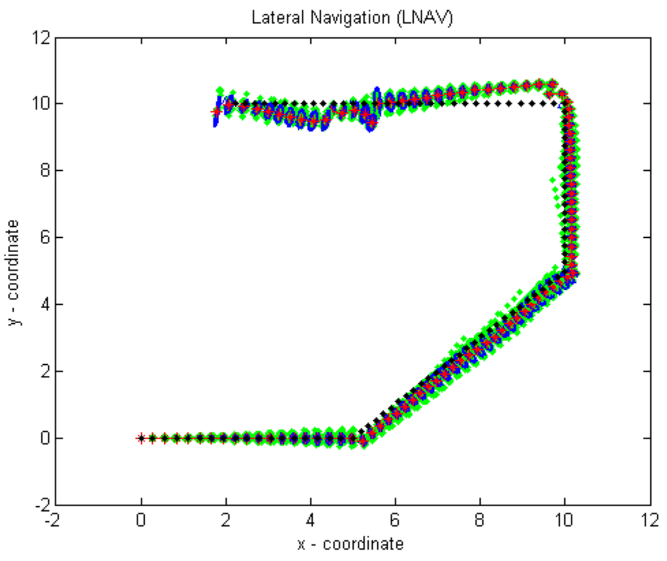


(b) Autopilot Vertival Navigation at t=60

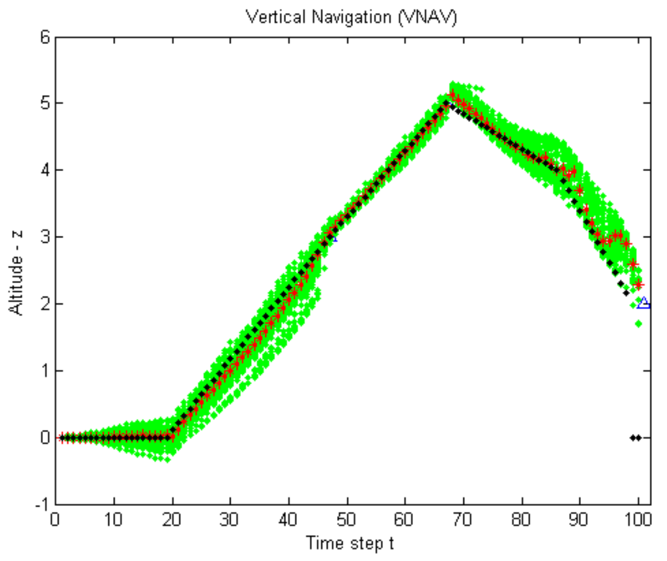
The total error of the simulation:

```
Mean LNAV Error x 0.213985
Mean LNAV Error y 0.208650
Mean VNAV Error z 0.058959
```

It's observable in this map that the farther the aircraft gets from the waypoint, the bigger the uncertainity is until it starts to approach the next waypoint then it starts to decrease. From it can be understood that the



(a) Autopilot Vertival Navigation at the end



(b) Autopilot Vertival Navigation at the end closer the waypoints are the better the aircraft will navigate through them.

6 References

- Probabilistic Robotics by Sebastian Thrun, Wolfram Burgard and Dieter Fox.
- Monte Carlo Localization With Mixture Proposal Distribution by Sebastian Thrun, Wolfram Burgard and Dieter Fox.
- Simplified Aircraft Motion by NASA. http://www.grc.nasa.gov/WWW/k-12/airplane/smotion.html
- Applied Estimation Course EL2320 Lab 2.