SYSC 5104

Methodologies for discrete event modelling and simulation

Winter 2016

Assignment 01

DEVS model for one dimensional particle filter localization

Name: A. K. Isuru U. W. Gunasekara

Student #: 8491795

University of Ottawa

**Part 1**

A one dimensional particle filter based localization for a helicopter will be modelled. The model consists of two atomic models and one coupled model consisting of three atomic sub models.

Flight computer

Position

Error

Estimated position

Position

Particle filter

Comparator

Actual height

Desired speed

Desired height

Height and speed adjustment

Laser sensor

Actual speed

Position calculation

Laser reading

Position Difference

Figure 1: Block diagram of the particle filter

1. Height and speed adjustment unit: When the pilot issues a speed or height adjustment command, this unit gradually brings the height and speed of the helicopter to the desired value. In addition to adjusting height and speed, this model is also responsible for reporting the current speed and current height periodically to the other models.
2. Flight computer composite model with three sub models
   1. Position calculation: This model calculates the position of the helicopter depending on the current speed and reports it.
   2. Laser sensor: The laser sensor is responsible for generating the measurements that would be obtained if a real sensor was fixed on a real helicopter. A preloaded one dimensional map of the ground is saved on this block and, measurements are based on the values obtained from the map depending on the actual height and position of the helicopter. Due to this reason, the height and position of the helicopter needs to be known by the sensor block to mimic the measurements of an actual laser sensor.
   3. Particle filter: The particle filter compares the data obtained from the sensor to a known map and estimates the position of the helicopter on an axis in 1D.
3. Comparator: The comparator compares the positon value calculated using the position calculation block and the particle filter block and outputs the error in the particle filter based estimation.

In a real helicopter, the position calculated using the speed of the helicopter accumulates errors due to linearization error, speed sensor errors, etc. This error can be corrected using the ground distance sensor if the helicopter is flying above a known environment. Even though particle filter based localization is mostly used in ground robots for localization, a helicopter is modelled here because a one dimensional particle filter makes more sense on a helicopter rather than a ground robot.

**Part 2**

As shown in the figure 1, the helicopter control system consists of two inputs and one output. When a desired speed and height is given into the helicopter, it slowly adjusts the speed and height using a proportional controller and then position is estimated using odometry (simulated) data and cross checked with the position estimated using a particle filter.

**Formal Specifications**

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

**Height and speed adjustment unit:**

S = {{phase, speed, height, referenceSpeed, referenceHeight}}

X = {Desired Speed, Desired Height}

Y = {Actual Height, Actual Speed}

δint ( phase, speed, height, referenceSpeed, referenceHeight )

{

case phase:

active:

if ( speed == referenceSpeed==0 && height == referenceHeight == 0)

{ phase = passive; //the helicopter has landed

sigma = infinity; }

else

{ adjust speed by a small step, if different from reference

adjust height by a small step, if different from reference

}

passive: //Never happens

}

δext (**phase**, **referenceSpeed**, **referenceHeight,** e, x)

{

if ( x is from desiredSpeed port)

{ referenceSpeed = desiredSpeed;}

if (x is from desiredHeight port)

{ referenceHeight = desiredHeight;}

}

λ(active)

{ Send speed and height values on the output ports “actualSpeedOut” and “actualHeightOut”

}

ta(passive) = INFINITY

ta(active) = publishPeriod

**Position calculation unit:**

S = {{phase, position, positionDifferenc}}

X = {speed}

Y = {position, position difference}

δint (active) = passive

δext (**phase, position, position difference**, e, speed)

{

calculate new position and position difference values

phase = active;

sigma = calculationTime; //simulate the delay in calculating positions

}

λ(active)

{ send position and position difference on output ports “positionOut” and “positionDiffOut”}

ta(passive) = INFINITY

ta(active) = calculationTime

**Laser sensor unit:**

S = {{phase, currentPosition, laserReading, currentHeight}}

X = {position, height}

Y = {laser reading}

δint (active) = passive

δext (**phase,currentPosition,laserReading,currentHeight,** e, x)

{

if (x is from positionIn port)

{ currentPosition = msg.value();

Calculate laser reading;

phase = active;

sigma = laser read time;

}

If(x is from actualHeightIn port)

{ currentHeight = msg.value();

}

}

λ(active)

{ send laser reading

}

ta(passive) = INFINITY

ta(active) = laser read time

**Particle filter**

S = {{phase, laserReading, estimatedPose, particles [NUM\_PARTICLES], height}}

X = {height, position difference, laser reading}

Y = {estimated position}

δint (active) = passive

δext (**phase, laserReading, estimatedPose, particles [NUM\_PARTICLES], height**, e, x)

{ if (x is from laserReadingIn port)

{ laserReading = msg.value();

Estimate new weights of the particles[NUM\_PARTICLES]

Resample particles depending on the new weights

Choose particle with maximum weight from new particles

Phase = active;

Sigma = calculation time

}

If(x is from positionDiffferenceIn port)

{ update all particles in the particles[NUM\_PARTICLES] array }

If(x is from actualHeightIn port)

{ height = msg.value();}

}

λ(active)

{ send estimated position

}

ta(passive) = INFINITY

ta(active) = calculation time

**Comparator**

S = {{phase, pfPosition, odomPosition, error}}

X = {Odometry\_Position, Estimated\_pose\_PF}

Y = {error}

δint (active) = passive

δext (**phase, pfPosition, odomPosition, error**, e, x)

{ if (x is from Estimated\_pose\_PF)

{ pfPosition = msg.value();

Calculate new error

Phase = active;

Sigma = 0; }

if (x is from Odometry\_Position)

{ odomPosition = msg.value(); }

}

λ(active)

{ send the error value on output port “errorOut”

}

ta(passive) = INFINITY

ta(active) = 0

The formal specification of the coupled models are as follows:

<X, Y, D, {Mi}, {Ii}, {Zij}, SELECT >

**Flight computer:**

X = {speed, height};

Y = {position, estimated position};

D = {position calculation, laser sensor, particle filter};

I (position calculation) = {self, laser sensor, particle filter};

I (laser sensor) = particle filter;

I (particle filter) = self;

Z(position calculation) = self; Z(position calculation) = particle filter;

Z(position calculation) = laser sensor;

Z(laser sensor) = particle filter;

Z(particle filter) = self;

SELECT: ({position calculation, laser sensor, particle filter}) = position calculation;

({laser sensor, particle filter}) = laser sensor

**PF\_Localization:**

X = {desired speed, desired height};

Y = {error};

D = {height and speed adjustment, flight computer, comparator};

I (height and speed adjustment) = {flight computer};

I (flight computer) = comparator;

I (comparator) = self;

Z (height and speed adjustment) = {flight computer};

Z (flight computer) = comparator;

Z (comparator) = self;

SELECT:

({height and speed adjustment, flight computer, comparator}) = height and speed adjustment

({flight computer, comparator }) = flight computer

**Testing strategy:**

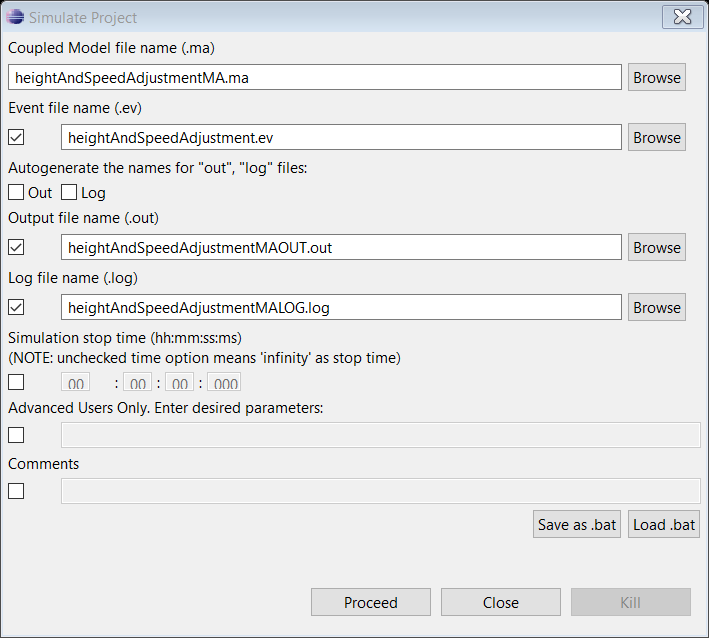
The Height and speed adjustment unit, position calculation unit, laser sensor unit and the comparator unit can be tested individually. Individual event files and models has been created in order to facilitate the unit testing.

**Part 3**

**Height and speed adjustment unit**

The height and speed adjustment unit can be tested by inputting desired heights and speeds and observing the output of the actual speed and actual height outputs. The actual speed and actual height should move towards the desired speed and heights.

(Run **1d\_ParticleFilter\_LocalizationheightAndSpeedAdjustment.bat** )



**”heightAndSpeedAdjustment.ev”**

00:00:00:00 desiredSpeedIn 10

00:00:00:00 desiredHeightIn 15

00:00:16:00 desiredSpeedIn 8

00:00:17:00 desiredHeightIn 15

00:00:20:00 desiredSpeedIn 8.5

00:00:21:00 desiredHeightIn 17

00:00:22:00 desiredSpeedIn 0

00:00:23:00 desiredHeightIn 0

“heightAndSpeedAdjustmentMAOUT.out” contains the results of this simulation. Some of the extracted data are as follows:

**00:00:00:000 actualspeedout 0**

**00:00:00:000 actualheightout 10**

………

00:00:05:000 actualspeedout 9.23055

00:00:05:000 actualheightout 14.6153

00:00:05:100 actualspeedout 9.26902

00:00:05:100 actualheightout 15

00:00:05:200 actualspeedout 9.30557

00:00:05:200 actualheightout 15

00:00:05:300 actualspeedout 9.34029

00:00:05:300 actualheightout 15

00:00:05:400 actualspeedout 9.37328

00:00:05:400 actualheightout 15

00:00:05:500 actualspeedout 9.40461

00:00:05:500 actualheightout 15

00:00:05:600 actualspeedout 9.43438

00:00:05:600 actualheightout 15

00:00:05:700 actualspeedout 9.46266

00:00:05:700 actualheightout 15

00:00:05:800 actualspeedout 9.48953

00:00:05:800 actualheightout 15

00:00:05:900 actualspeedout 9.51505

00:00:05:900 actualheightout 15

00:00:06:000 actualspeedout 9.5393

00:00:06:000 actualheightout 15

00:00:06:100 actualspeedout 9.56234

00:00:06:100 actualheightout 15

00:00:06:200 actualspeedout 9.58422

00:00:06:200 actualheightout 15

00:00:06:300 actualspeedout 9.60501

00:00:06:300 actualheightout 15

**00:00:06:400 actualspeedout 10**

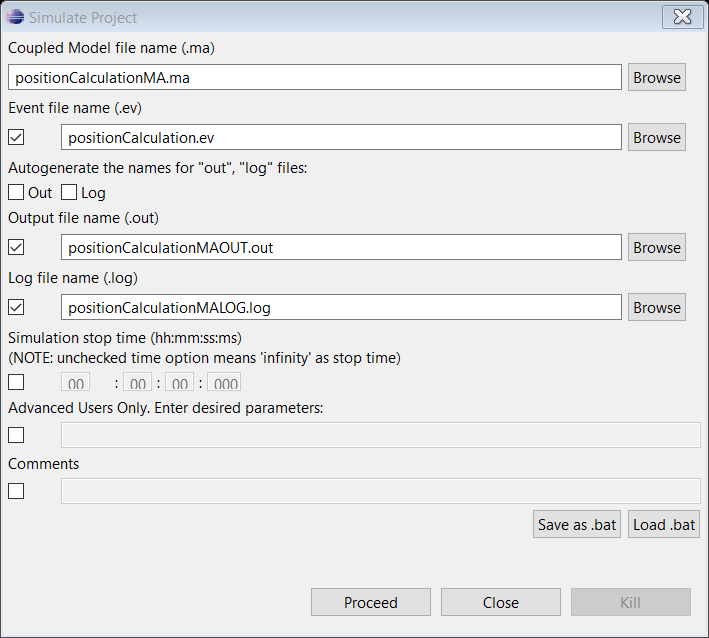
**00:00:06:400 actualheightout 15**

It can be seen that the outputs have settled at the desired values after 6.4 seconds. Similarly, the outputs have settled for the other inputs too after similar periods.

**Position calculation unit**

The position calculation unit can be tested by inputting speeds and examining the calculated position and the difference in the positions as follows,

(Run **1d\_ParticleFilter\_LocalizationpositionCalculation.bat**)

****

“positionCalculation.ev”

00:00:00:00 speedIn 10

00:00:20:00 speedIn 20

00:00:30:00 speedIn 40

00:00:40:00 speedIn 20

00:00:50:00 speedIn 14

Output with the above event file is as follows:

00:00:00:000 positionout 0

**00:00:20:000 positionout 300**

00:00:30:000 positionout 600

00:00:40:000 positionout 900

00:00:50:000 positionout 1070

As seen above, the position calculation unit assumes the helicopter is moving with constant acceleration. Therefore, between 0 and 20 seconds, the velocity is assumed to be linearly increasing. This would give a position of:

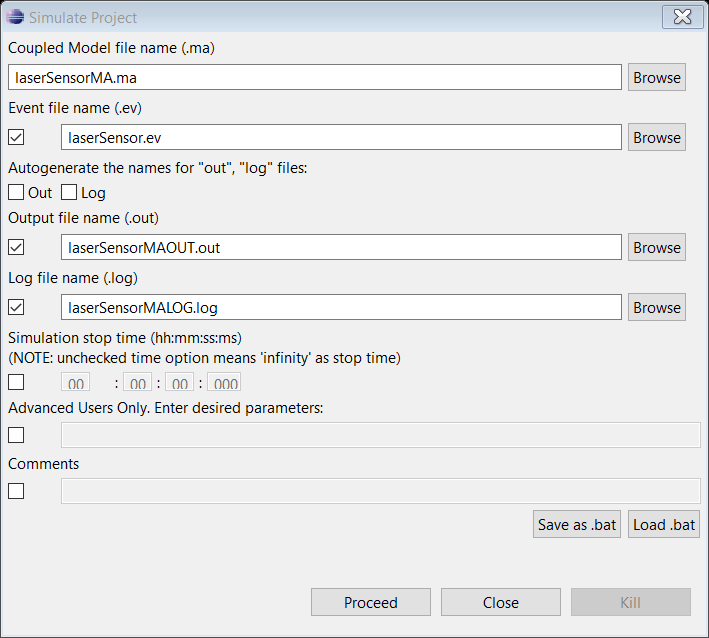
Position at 20 seconds = (20-0)\*(10+20)/2 = 300

Similarly, the other positions can also be verified

**Laser sensor:**

The laser sensor can be verified by inputting pseudo positions and heights and observing its output:

Run: **1d\_ParticleFilter\_LocalizationlaserSensor.bat**



**“laserSensor.ev”**

00:00:00:00 position 1

00:00:00:00 height 1

00:00:01:00 position 2

**00:00:01:00 height 5**

00:00:02:00 position 2.5

00:00:02:00 height 8

00:00:03:00 position 3

00:00:03:00 height 8

00:00:04:00 position 5

00:00:04:00 height 7

00:00:05:00 position 7

**00:00:05:00 height 6**

00:00:06:00 position 10

00:00:06:00 height 5

The output with the above inputs are as follows:

00:00:00:010 laserreading 0.846672

00:00:01:010 laserreading 1.85809

**00:00:02:010 laserreading 5.50097**

00:00:03:010 laserreading 8.02925

00:00:04:010 laserreading 8.64277

**00:00:05:010 laserreading 7.29843**

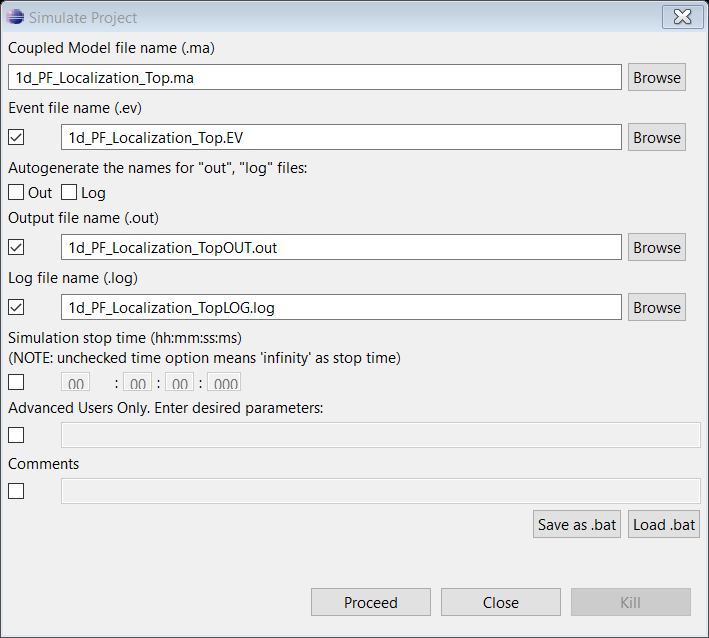
00:00:06:010 laserreading 6.14592

As seen above, each output is 10 milliseconds delayed from each input. This simulates the delay in obtaining measurements from the laser sensor. As seen above, it can be seen that the laser reading changes according to the height and position of the plane. Since the ground level is modelled as below 0 level in this simulation, it can also be seen that the laser reading is always higher than the “height” value of the helicopter.

**1D\_PF\_Localization\_Top:**

The complete model can be simulated using the following file:

**1d\_ParticleFilter\_Localization1D\_PF\_Localization\_top.bat**



**“1D\_PF\_Localization\_Top.ev”**

00:00:00:00 speed 1

00:00:00:00 height 10

00:00:01:00 speed 0.5

00:00:02:00 height 10

00:00:03:00 speed 1

00:00:03:00 height 10

00:00:04:00 speed 0

00:00:04:00 height 0

The output after running with the above input is as follows:

00:00:00:013 error 0.687438

00:00:00:113 error 0.702549

00:00:00:213 error 0.710487

00:00:00:313 error 0.723658

00:00:00:413 error 0.734193

00:00:00:513 error 0.747181

00:00:00:613 error 0.751051

00:00:00:713 error 0.764728

00:00:00:813 error 0.778228

00:00:00:913 error 0.770646

00:00:01:013 error 0.775652

00:00:01:113 error 0.779322

00:00:01:213 error 0.777582

00:00:01:313 error 0.798334

00:00:01:413 error 0.802469

00:00:01:513 error 0.79408

00:00:01:613 error 0.800082

00:00:01:713 error 0.824246

00:00:01:813 error 0.80945

00:00:01:913 error 0.833059

00:00:02:013 error 0.799281

00:00:02:113 error 0.759375

00:00:02:213 error 0.737666

00:00:02:313 error 0.71524

00:00:02:413 error 0.690805

00:00:02:513 error 0.667998

00:00:02:613 error 0.647539

00:00:02:713 error 0.623756

00:00:02:813 error 0.600779

00:00:02:913 error 0.581642

00:00:03:013 error 0.636505

00:00:03:113 error 0.64624

00:00:03:213 error 0.670378

00:00:03:313 error 0.693789

00:00:03:413 error 0.718438

00:00:03:513 error 0.743433

00:00:03:613 error 0.767262

00:00:03:713 error 0.781049

00:00:03:813 error 0.790033

00:00:03:913 error 0.772231

00:00:04:013 error 0.767756

00:00:04:113 error 0.79162

00:00:04:213 error 0.766856

00:00:04:313 error 0.753287

00:00:04:413 error 0.744724

00:00:04:513 error 0.731641

00:00:04:613 error 0.707387

00:00:04:713 error 0.682625

00:00:04:813 error 0.657774

00:00:04:913 error 0.635553

00:00:05:013 error 0.665397

00:00:05:113 error 0.689609

00:00:05:213 error 0.714499

00:00:05:313 error 0.712695

00:00:05:413 error 0.697192

00:00:05:513 error 0.699523

00:00:05:613 error 0.723914

00:00:05:713 error 0.737401

00:00:05:813 error 0.756902

00:00:05:913 error 0.759782

00:00:06:013 error 0.757429

00:00:06:113 error 0.76097

00:00:06:213 error 0.76746

00:00:06:313 error 0.74492

00:00:06:413 error 0.722024

00:00:06:513 error 0.742731

00:00:06:613 error 0.749181

00:00:06:713 error 0.760978

00:00:06:813 error 0.780822

00:00:06:913 error 0.764705

00:00:07:013 error 0.765068

00:00:07:113 error 0.782959

00:00:07:213 error 0.790019

00:00:07:313 error 0.812369

00:00:07:413 error 0.82133

00:00:07:513 error 0.818713

00:00:07:613 error 0.81655

00:00:07:713 error 0.802558

00:00:07:813 error 0.823447

00:00:07:913 error 0.830001

00:00:08:013 error 0.835341

00:00:08:113 error 0.831864

00:00:08:213 error 0.823686

00:00:08:313 error 0.815193

00:00:08:413 error 0.828173

00:00:08:513 error 0.852081

00:00:08:613 error 0.861087

00:00:08:713 error 0.877007

00:00:08:813 error 0.900832

00:00:08:913 error 0.921463

00:00:09:013 error 0.906604

00:00:09:113 error 0.901672

00:00:09:213 error 0.901347

00:00:09:313 error 0.881768

00:00:09:413 error 0.884214

00:00:09:513 error 0.862811

00:00:09:613 error 0.859742

00:00:09:713 error 0.871221

00:00:09:813 error 0.893394

00:00:09:913 error 0.884264

00:00:10:013 error 0.878837

00:00:10:113 error 0.902705

00:00:10:213 error 0.909499

00:00:10:313 error 0.900987

00:00:10:413 error 0.886214

It can be seen that the difference between the position estimated using the particle filter and the position calculated using the odometry data is always less than 1. The implemented particle filter assumes a Gaussian distribution for the sensor noise and, a uniform distribution for the odometry noise. The resampling step is done by a simple method of eliminating particles with weights less than a random number between 0 and the cumulative weight of all particle weights. Adjustments to these methods such as using a more accurate model for the odometry noise or using a more advanced resampling method can improve the accuracy of the particle filter even further.