AE3M33MKR
Kalman Filter

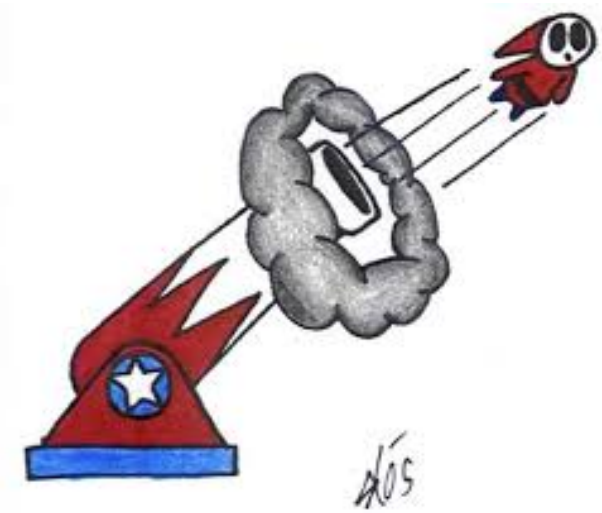
Ing. Karel Košnar PhD., RNDr. Miroslav Kulich, Ph.D.,
Dr. Gaël Écorchard

Intelligent and Mobile Robotics Group
Czech Technical University in Prague

November 7, 2018

Assignment

We will use a common physics problem.
This assignment will involve firing a ball from a cannon at an unknown angle and at an unknown velocity.
However, we will have a camera that will record the ball's position from the side. The positions measured from the camera have significant a measurement error.
We want to know the position of the ball as precisely as possible.



Mathematical Description

The position $\mathbf{x} = (x, y)$ of the ball is computed from its initial position, $\mathbf{x} = (x_0, y_0)$, its initial velocity $\mathbf{v_0} = (v_{0x}, v_{0y})$ and the gravitational acceleration g:

$$\begin{aligned} x(t) &= x_0 + v_{0x} t \\ y(t) &= y_0 - v_{0y} t - \frac{1}{2} g t^2 \\ v_x(t) &= v_{0x} \\ v_y(t) &= v_{0y} - g t \end{aligned} \tag{1}$$

Mathematical Description

In the discrete case the system can be described by:

$$
\begin{aligned}
x_n &= x_{n-1} + v_{n-1_x}\Delta t \\
y_n &= y_{n-1} - v_{n-1_y}\Delta t - \frac{1}{2}g\Delta t^2 \\
v_{nx} &= v_{n-1_x} \\
v_{ny} &= v_{n-1_y} - g\Delta t
\end{aligned}
\tag{2}
$$

State description

$$x_n = \mathbf{A}x_{n-1} + \mathbf{B}u \tag{3}$$

u is the control input, in this case the control input is the influence of the gravitational acceleration.

Kalman filter

used constants:
A: State transition matrix. Basically, multiply state by this and add control factors, and you get a prediction of the state for the next time step.
B: Control matrix. This is used to define the linear relation between control and state.
H: Observation matrix. Multiply a state vector by H to translate it into a measurement vector.
Q: Covariance of the estimated state error.
R: Covariance of the estimated measurement error.

Kalman filter

Inputs:
u: Control vector. Constant in this case.
z_n: Measurement vector. Position of the cannon ball measured in this time step. Measured values contain noise.
Outputs:
x_n: Newest estimate of the current "true" state.
P_n: Newest covariance of the estimate of the average error for each parts of the state.

Kalman filter - How to compute it

Predict where the cannon ball is going to be:

$$x_{predicted} = \mathbf{A}x_{n-1} + \mathbf{B}u \tag{4}$$

Predict the covariance of the error:

$$P_{predicted} = \mathbf{A}\mathbf{P}_{n-1}\mathbf{A}^T + \mathbf{Q} \tag{5}$$

Compare the reality against the prediction, we call it innovation:

$$\tilde{y} = z_n - \mathbf{H}x_{predicted} \tag{6}$$

Compute the covariance of the innovation:

$$S = \mathbf{H}\mathbf{P}_{predicted}\mathbf{H}^T + \mathbf{R} \tag{7}$$

Kalman filter - How to compute it, cont.

Now the key part, the Kalman Gain:

$$K = \mathbf{P}_{predicted}\mathbf{H}^T\mathbf{S}^{-1} \tag{8}$$

and finally we update the state:

$$x_n = x_{predicted} + \mathbf{K}\tilde{y} \tag{9}$$

and the covariance of the state:

$$P_n = (\mathbf{I} - \mathbf{KH})\mathbf{P}_{predicted}, \text{ with } \mathbf{I} \text{ the identity matrix} \tag{10}$$

code explanation

Initialization of the matrices, the size of the state vector is 4 and the size of measurement vector is 2.

```
Matrix4f A,B,P,Q;
Matrix2f R;
Matrix<float,n,m> H;
Vector4f x;
float dt = 0.1;
```

The matrices can be filled like this:

```
P << 1, 0, 0, 0,
     0, 1, 0, 0,
   0, 0, 1, 0,
   0, 0, 0, 1;
```

Code explanation, cont.

Initialization of the GUI and the system simulator:

```
Gui gui;
System system;
```

A point is defined by two co-ordinates x and y.
Used variables:

measurement holds the measured position of the ball

truth holds the true position (used for painting)

kfPosition holds the position estimated by the Kalman filter

Code explanation, cont.

The main loop simulates one step of the system, stores the true and measured positions in the apropriate variables

```
for(int i=1; i<nSteps; i++) {
   system.makeStep();
   truth = system.getTruthPosition();
   measurement = system.getMeasurement();
```

estimates the position by Kalman filter

```
   kfPosition = KalmanFilter(measurement);
```

and plots all the positions by calling

```
   gui.setPoints(truth, measurement, kfPosition);
}
gui.startInteractor();
```

Assignment

Your task is to implement a Kalman filter in the KalmanFilter function:

```
Point  KalmanFilter(const Point measuredPosition )
{


}
```

which returns the estimated position as a variable of type Point.