In this exercise you will extend last week's Kalman filter to the nonlinear case, which is then called an Extended Kalman Filter (EKF). We make the following changes:

- the state includes heading (yaw) of the quadrotor,
- replace the constant velocity model with the odometry you implemented in Week 2,
- replace the GPS sensor with a camera, which detects markers in the environment. This allows us to measure the quadrotors heading as well.

In the following we describe the quadrotors state, the state and measurement prediction function.

State

The quadrotors state $\mathbf{x}$ includes its x and y position in world coordinates and the current heading ψ:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \psi \end{pmatrix}$$

State Prediction Function

The prediction function tells us what the next state $\mathbf{x}_{t+1}$ will be given our current state $\mathbf{x}_t$ and control inputs $\mathbf{u}_t$, i.e.,

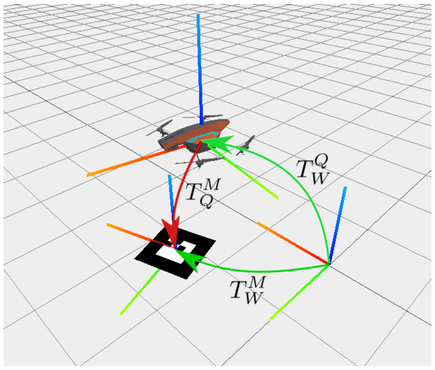
$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

Here we will use the odometry from Week 2 to predict the next state. This means we will integrate up our current, noisy velocity measurements. Of cause, we will accumulate drift over time, which we will correct with the marker observations. Using the odometry f can be defined as:

$$f(\mathbf{x}_t, \mathbf{u}_t) = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \psi_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + \Delta\, t \cdot\ (\cos(\psi_t) \cdot\ \dot{x}_t -\ \sin(\psi_t) \cdot \dot{y}_t) \\ y_t + \Delta\, t \cdot (\sin(\psi_t) \cdot \dot{x}_t + \cos(\psi_t) \cdot \dot{y}_t) \\ \psi_t + \Delta t \cdot \dot{\psi}_t \end{pmatrix}$$

Measurement Prediction Function

To correct for the drift we accumulate by integrating the odometry, we placed markers in the world, which can be detected by the down facing quadrotor camera. For each marker we know its position and orientation in the world. Therefore, we can predict the pose of a marker relative to the quadrotor given our current state $\mathbf{x}_t$ and the world pose of the marker. By comparing the predicted relative pose and the actual measured relative pose we can correct our state estimate. The following image illustrates the involved transformations:



The quadrotor pose T_W^Q is given by our current state estimate, the marker pose T_W^M is known from our marker map, and the relative marker pose T_Q^M can be measured by the camera.

A marker measurement comprises the relative x_m and y_m position and the relative orientation ψ_m:

$$\mathbf{z} = \begin{pmatrix} x_m \\ y_m \\ \psi_m \end{pmatrix}$$

The measurement prediction function $\mathbf{z} = h(\mathbf{x}_t)$ is then defined as

$$h(\mathbf{x}_t) = \begin{pmatrix} \cos(\psi_t) \cdot (x_W^M - x_t) + \sin(\psi_t) \cdot (y_W^M - y_t) \\ -\sin(\psi_t) \cdot (x_W^M - x_t) + \cos(\psi_t) \cdot (y_W^M - y_t) \\ \psi_W^M - \psi_t \end{pmatrix}$$

where x_W^M,y_W^M, and ψ_W^M are the world marker position and orientation.

For the implementation of the EKF we will need the Jacobian matrices of the state prediction function f and the measurement prediction function h. The 3×3 Jacobian matrix of f is defined as:

$$\mathbf{F} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial\, \mathbf{x}}\bigg|_{\mathbf{x}_t} = \begin{pmatrix} 1 & 0 & \Delta t \cdot (-\sin(\psi_t) \cdot \dot{x}_t - \cos(\psi_t) \cdot \dot{y}_t) \\ 0 & 1 & \Delta t \cdot (\cos(\psi_t) \cdot \dot{x}_t - \sin(\psi_t) \cdot \dot{y}_t) \\ 0 & 0 & 1 \end{pmatrix}$$

The Jacobian matrix of h is as well a 3×3 matrix. It is defined as:

$$\mathbf{H} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_t} = \begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ 0 & 0 & -1 \end{pmatrix}$$

Please calculate the missing 6 entries, i.e., $H_{1,1} = \frac{\partial x_m}{\partial x_t}$, $H_{1,2} = \frac{\partial x_m}{\partial y_t}$, $H_{1,3} = \frac{\partial x_m}{\partial \psi_t}$, $H_{2,1} = \frac{\partial y_m}{\partial x_t}$, $H_{2,2} = \frac{\partial y_m}{\partial y_t}$, and $H_{2,3} = \frac{\partial y_m}{\partial \psi_t}$.

You can use `sin`, `cos` functions, `*` for multiplication, `x` and `y` for the drone position, `psi` for the drone yaw angle, `x_m` and `y_m` for the marker world position.

H11 (1 point possible)

We will use this value to familiarize you with formula input. The solution is $-\cos(\psi)$. Enter `-cos(psi)`.

$H_{1,1}$ =

Check Save *You have used 0 of 3 submissions*

H12 (1 point possible)

$H_{1,2}$ =

Check Save *You have used 0 of 3 submissions*

H13 (1 point possible)

$H_{1,3}$ =

Check Save *You have used 0 of 3 submissions*

H21 (1 point possible)

$H_{2,1}$ =

Check Save *You have used 0 of 3 submissions*

H22 (1 point possible)

$H_{2,2}$ =

Check Save *You have used 0 of 3 submissions*

H23 (1 point possible)

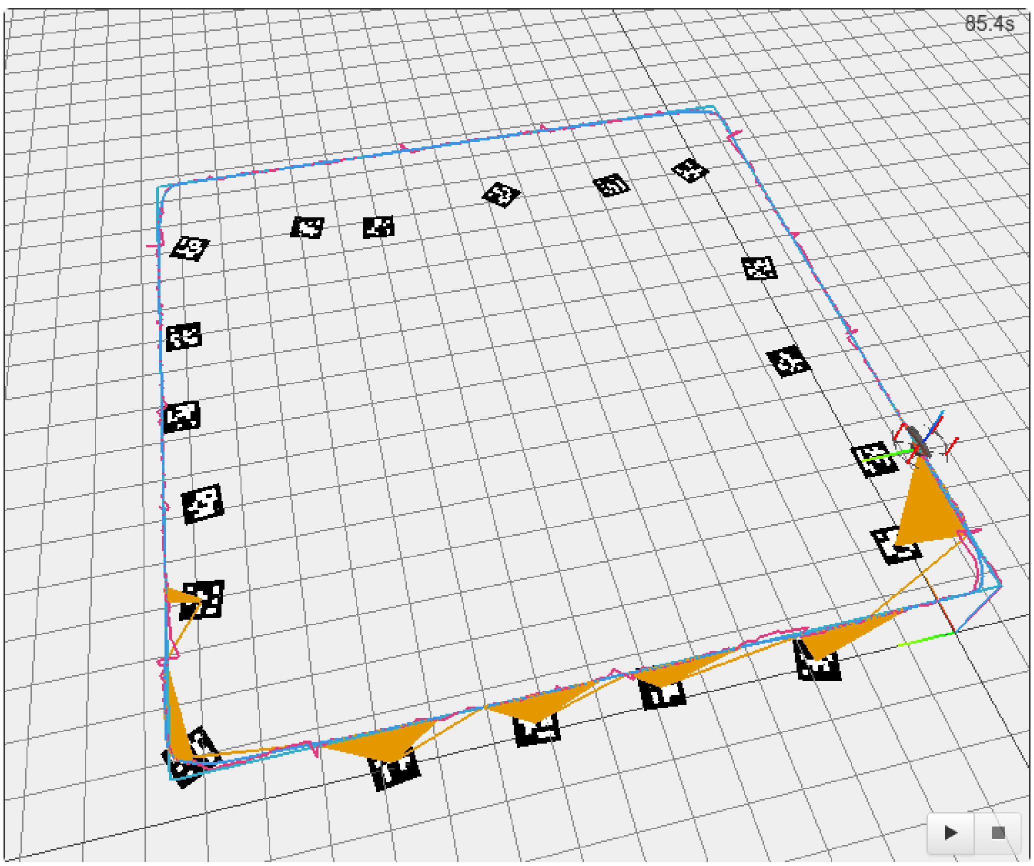
$H_{2,3}$ =

Check Save *You have used 0 of 3 submissions*

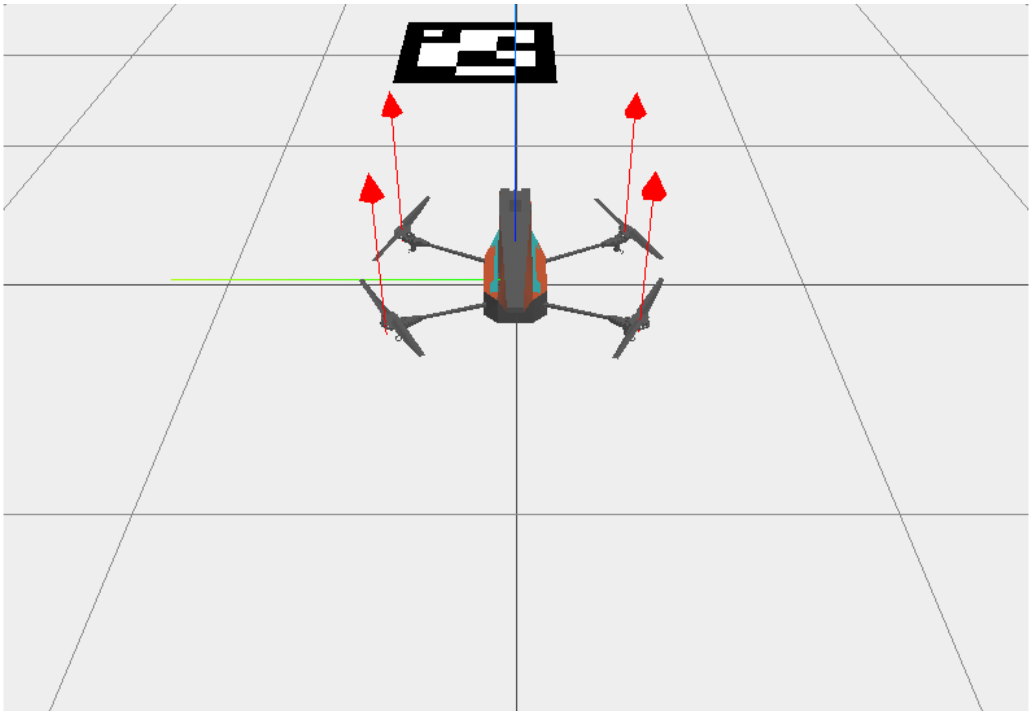
The code below provides a framework for the EKF described above. It is an adapted version of last week. The goal is again to fly the quadrotor along a square trajectory. This time the heading is also controlled.

The prediction steps have already been implemented. For the correction step the method `calculatePredictMeasurementJacobian` is missing, which computes H given the current state and a global marker pose. Use your solutions from above. Furthermore, you have to implement the `correctState` method, which performs the EKF state correction step.

If your solution is correct you will get a similar output to the image shown below. In the simulator, the pink trajectory visualizes the Kalman filter state. The orange lines indicate that the quadrotor observes a marker. Note that the quadrotor can only observe a marker while it is located above the marker. The blue line is the groundtruth trajectory. The cyan line shows the setpoint position.



CODE (15 points possible)



Check Save *You have used 0 of 6 submissions*

Show Discussion New Post