Programming Exercise #1 Topic: Kalman Filter

Issued: Apr 28, 2010 Due: May 12, 2010

Sebastian Trimpe (strimpe@ethz.ch) , April 30, 2010

Extended Kalman Filter for Tracking a Two-wheeled Robot

An Extended Kalman Filter is to be designed for tracking the position and orientation of a two-wheeled robot that is moving on a plane. A schematic drawing of the robot is shown in Fig. 1.

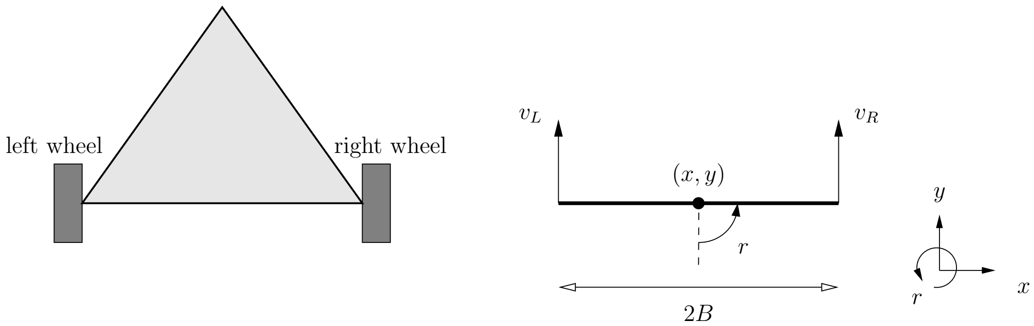

Figure 1: Top view of the two-wheeled robot (left) and relevant physical quantities (right).

The robot can command its left and right wheel angular velocities, $u_L(t)$ and $u_R(t)$ (in rad/s), respectively, which are assumed to be followed instantaneously. The left and right wheel radii, W_L and W_R (in m), are not known perfectly; they are modeled as random variables according to

$$W_L = W_0(1 + \xi_L)$$
$$W_R = W_0(1 + \xi_R),$$

with the known nominal wheel radius W_0 (in m) and the uniformly distributed random variables $\xi_L, \xi_R \in [-\bar{\xi}, \bar{\xi}]$. The wheel radii are assumed constant with time.
The translational speed $v_t(t)$ (in m/s) of the vehicle is

$$v_t(t) = \frac{v_R(t) + v_L(t)}{2} \qquad \text{with} \quad v_R(t) = W_R u_R(t) \quad \text{and} \quad v_L(t) = W_L u_L(t).$$

The rotational speed $v_r(t)$ (in rad/s) of the vehicle is

$$v_r(t) = \frac{v_R(t) - v_L(t)}{2B},$$

where B is the known wheel base (distance of the wheels from the robot center), see Fig. 1. With these quantities, the kinematic equations read as follows:

$$\dot{x}(t) = v_t(t) \cos(r(t)) \tag{1}$$
$$\dot{y}(t) = v_t(t) \sin(r(t)) \tag{2}$$
$$\dot{r}(t) = v_r(t), \tag{3}$$

where $(x(t), y(t))$ is the position of the robot (in m) and $r(t)$ its orientation (in rad). The robot is assumed to start at $(x(0), y(0)) = (x_0, y_0)$ with orientation $r(0) = r_0$, where $x_0, y_0 \in [-\overline{p}, \overline{p}]$ and $r_0 \in [-\overline{r}, \overline{r}]$ are uniformly distributed random variables.
At varying instances of time, the robot may receive measurements of its position and orientation that are corrupted by sensor noise, i.e.

$$\begin{aligned} z_x &= x + w_x \\ z_y &= y + w_y \\ z_r &= r + w_r, \end{aligned}$$

with $w_x, w_y \in [-\overline{w}_p, \overline{w}_p]$, $w_r \in [-\overline{w}_r, \overline{w}_r]$ uniformly distributed. At any instance of time t_k, measurements may be available from one, two, three, or none of the sensors.
All random variables ξ_L, ξ_R, r_0, x_0, y_0, w_x, w_y, and w_r are assumed to be mutually independent and independent over time.

Objective

The objective is to design an Extended Kalman Filter to estimate the position and orientation of the two-wheeled robot. The estimator will be implemented in discrete time. At time t_k, the estimator has access to the time t_k, the control inputs $u_L(t_k)$ and $u_R(t_k)$, and possibly the measurements $z_x(t_k)$, $z_y(t_k)$, or $z_r(t_k)$. Furthermore, the values of all physical constants W_0, $\overline{\xi}$, B, $\overline{w}_p$, $\overline{w}_r$, $\overline{p}$, and $\overline{r}$ are known to the estimator. The orientation, the position, and the wheel radii are estimator states.

Provided Matlab Files

A set of Matlab files is provided on the class website. Please use them for solving the above problem.

`script.m`	Matlab script that is used to simulate the truth system, run the estimator, and display the results.[1]
`Estimator.m`	Matlab function template to be used for your implementation of the Extended Kalman Filter.
`PhysicalConstants.m`	Physical constants, known to the estimator.
`SimulationConstants.m`	Sample problem data, **not** known to the estimator.
`CalculateInputs.m`	Matlab function used to calculate the input wheel speeds.
`Uniform.m`, `UniformMinMax.m`	Uniform random number generators.

[1] In `script.m` the following one-step method for numerical integration is implemented: Assuming $v_t(t)$ and $v_r(t)$ are constant over the sampling interval $[t_k, t_{k+1}]$ (which they are since piecewise constant inputs are considered in the simulation), we have from (3) for $t \in [t_k, t_{k+1}]$

$$\dot{r}(t) = v_r \quad \Rightarrow \quad r(t) = r(t_k) + (t - t_k)v_r \quad \Rightarrow \quad r(t_{k+1}) = r(t_k) + (t_{k+1} - t_k)v_r,$$

where $v_r = v_r(t)$ for all $t \in [t_k, t_{k+1}]$. With this and (1), we write for $x(t)$, $t \in [t_k, t_{k+1}]$

$$\begin{aligned} \dot{x}(t) &= v_t \cos(r(t_k) + (t - t_k)v_r) \\ &= v_t \big(\cos(r(t_k)) \cos((t - t_k)v_r) - \sin(r(t_k)) \sin((t - t_k)v_r) \big) \\ &\approx v_t \cos(r(t_k)) - v_t v_r \sin(r(t_k))(t - t_k), \qquad \text{for small } (t - t_k) \\ \Rightarrow x(t_{k+1}) &= x(t_k) + v_t \cos(r(t_k))(t_{k+1} - t_k) - \tfrac{1}{2} v_t v_r \sin(r(t_k))(t_{k+1} - t_k)^2, \end{aligned}$$

where $v_t = v_t(t)$ for all $t \in [t_k, t_{k+1}]$, and similarly for $y(t)$, $t \in [t_k, t_{k+1}]$

$$y(t_{k+1}) = y(t_k) + v_t \sin(r(t_k))(t_{k+1} - t_k) - \tfrac{1}{2} v_t v_r \cos(r(t_k))(t_{k+1} - t_k)^2.$$

Clearly, one may also employ other integration schemes such as forward Euler (less accurate) or higher-order methods (e.g. in the `Matlab` ODE suite; usually slower).

Task

Implement your solution for the Extended Kalman Filter in the file `Estimator.m`. Your code has to run with the Matlab script `script.m` and problem data as for example given in `PhysicalConstants.m` and `SimulationConstants.m`. For your estimator, use the function definition as given in the template `Estimator.m`.
For evaluating your solution, we will test it on the given problem data. Moreover, we will do suitable modifications of the parameters in `PhysicalConstants.m` and `SimulationConstants.m` and also test your estimator on those.
For judging your own solution, a typical performance of an Extended Kalman Filter implementation for the given problem is shown in Fig. 2 and 3.

Deliverables

Please hand in by e-mail your implementation of the Extended Kalman Filter in `Estimator.m`. Include the file into a `zip`-file, which you name `RFE10Ex1_Names.zip`, where *Names* is a list of the **pre- and surnames** of all students[2] who have worked on the solution (for example `RFE10Ex1_AngelaSchoellig_SebastianTrimpe.zip`).

Send your file to Sebastian (`strimpe@ethz.ch`) until the due date indicated above. We will send a confirmation e-mail upon receiving your solution. You are ultimately responsible that we receive your solution in time.

[2] Up to three students are allowed to work together on the programming exercise. They will all receive the same grade.

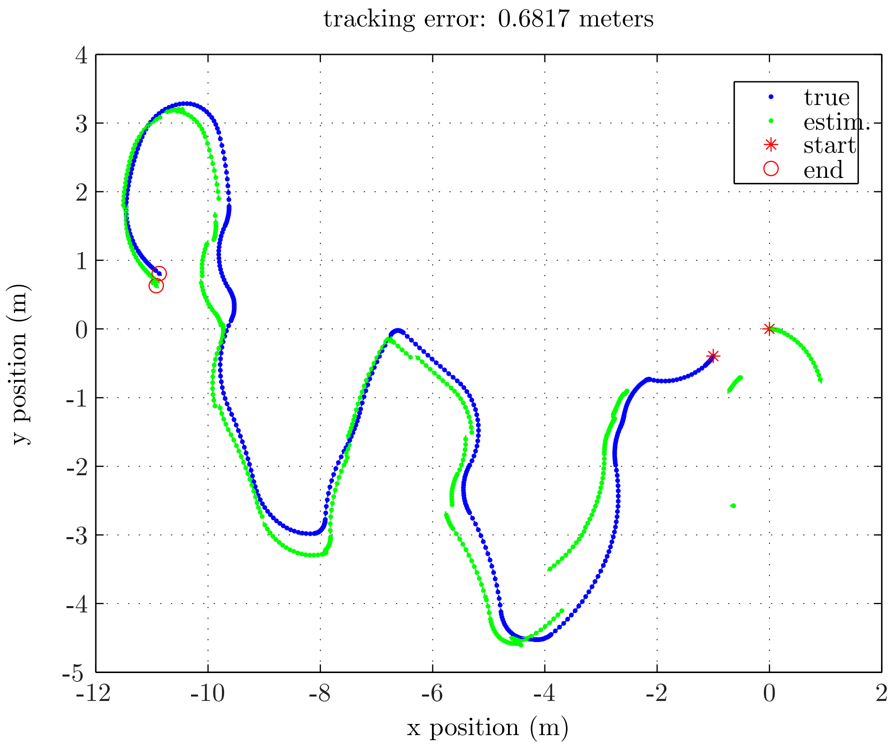

Figure 2: Typical tracking performance of an estimator for the given problem data.

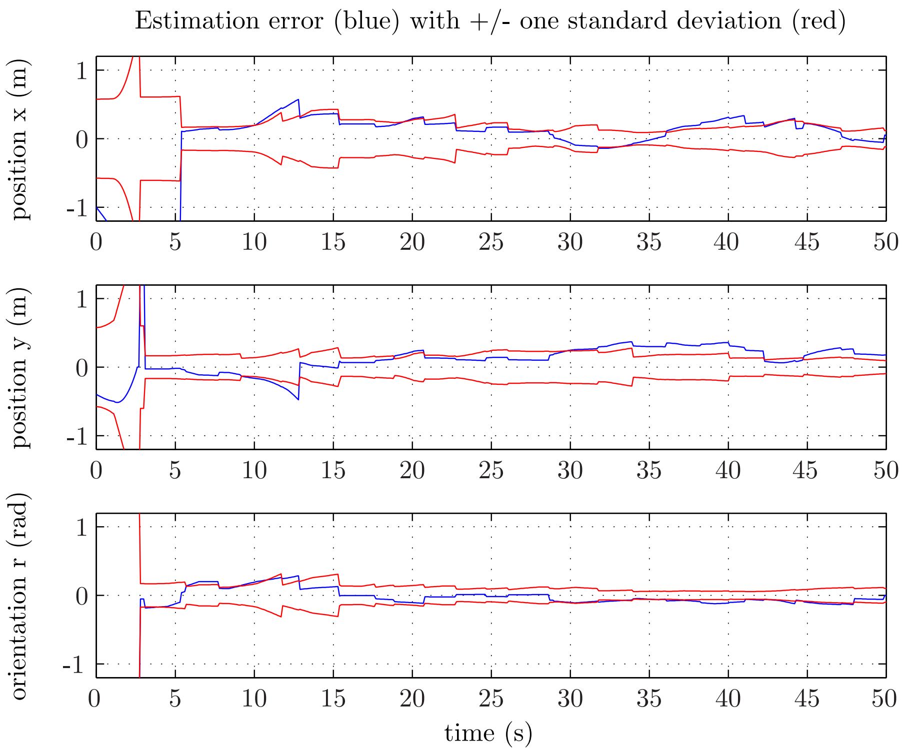

Figure 3: Typical estimation errors with +/- one standard deviation for the given problem data.