Exercise 1:

a) 
\[
p(X = \text{green}|Z = \text{green}) = \frac{p(Z = \text{green}|X = \text{green})p(X = \text{green})}{p(Z = \text{green})} \\
= \sum_{x \in \{\text{red, green, blue}\}} p(Z = \text{green}|X = x)p(X = x) \\
= 0.6^2 \times 0.3 + 0.6^2 \times 0.2 + 0.24 \times 0.32 \\
= 0.24 + 0.24 + 0.04 \\
= 0.32 = 4 = 0.75
\]

b) 
\[
p(z \mid x) = \mathcal{N}(z \mid x, \sigma_1^2) = \frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{(z-x)^2}{2\sigma_1^2}} = \frac{1}{0.3\sqrt{2\pi}}e^{-5.55(z-x)^2}
\]
where $\sigma_1 = 0.3$ is the sensor noise.

c) The motion can also be modeled with a Gaussian. We just need to think about what is our mean and what is our variance. The variance is given as the actuator noise $\sigma_2 = 0.1$. Our mean is the position we expect our robot to be at, after the motion $u_t$. Since our robot moves with constant speed $v$, the expected position is simply $\mu = x_{t-1} + v\Delta t$. Therefore we have
\[
p(x_t|x_{t-1}, u_t) = \mathcal{N}(x_t|x_{t-1} + v\Delta t, \sigma_2^2) \\
= \frac{1}{\sqrt{2\pi}\sigma_2}e^{-\frac{1}{2}\frac{(x_t-(x_{t-1}+v\Delta t))^2}{\sigma_2^2}} \\
= \frac{10}{\sqrt{2\pi}}e^{-50(x_t-(x_{t-1}+v\Delta t))^2}
\]
d) We model the state variable $x$ as a discrete random variable with values between 0 and 5, where 0 means that the robot is at the door. We want to compute the robot’s belief. Initially, the robot knows it is located at the door ($x=0$), therefore we have $Bel(x_0 = 0) = 1$. We then use the Bayes filter algorithm to compute the belief after 3 seconds, namely $Bel(x_3)$. Since it is a recursive algorithm we have to compute the belief at every time step. The general equation of the Bayes filter is:

$$Bel(x_t) = \eta \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

(2)

Our $u_i$, namely our action is always the same: move with constant speed $v = 1m/s$. Here is the first step:

$$Bel(x_1) = \eta_1 p(z_1|x_1) \int p(x_1|u_1, x_0) Bel(x_0) dx_0$$

(3)

$$= \eta_1 p(z_1|x_1) \int p(x_1|u_1, x_0) dx_0$$

(4)

$$= \eta_1 N(z_1|x_1, \sigma_1^2) \sum_{x_0=0}^{5} N(x_1|x_0 + 1, \sigma_2^2)$$

(5)

$$= \eta_1 N(z_1|x_1, \sigma_1^2) N(x_1|x_0 + 1, \sigma_2^2)$$

(6)

Let us separate the computation to the motion and the sensing part. Since we have $Bel(x_0 = 0) = 1$ we begin from the motion $u_1$.

$$p(x_1|u_1, x_0) = N(x_1|x_0 + 1, \sigma_2^2)$$

(7)

The belief of the robot after the first motion can be estimated as

$$Bel'(x_1) = \sum_{x_0=0}^{5} N(x_1|x_0 + 1, \sigma_2^2)$$

(8)

Now we take into account the sensor measurement $z_1$:

$$p(z_1|x_1) = N(z_1|x_1, \sigma_1^2)$$

(9)

Therefore

$$Bel(x_1) = \eta_1 p(z_1|x_1) Bel'(x_1)$$

(10)

Since our positions are restricted to a space $x_t \in \{0, 1, 2, 3, 4, 5\}$, we can compute our normalizers $\eta_i$ using (the inverse of) the sum of the probabilities for all possible states.

$$\eta_i^{-1} = \sum_{x'_i=0}^{5} Bel(x_1 = x'_i)$$

(11)
If we recursively substitute the beliefs we get:

\[ Bel(x_3) = \eta_3 p(z_3|x_3) \int p(x_3|u_3, x_2) Bel(x_2) \, dx_2 \]

\[ = \eta_3 p(z_3|x_3) \int p(x_3|u_3, x_2) \eta_2 p(z_2|x_2) \int p(x_2|u_2, x_1) Bel(x_1) \, dx_1 \, dx_2 \]

Plugging the numbers in we get the following table:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( Bel(x_0) )</th>
<th>( Bel(x_1) )</th>
<th>( Bel(x_2) )</th>
<th>( Bel(x_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.0001</td>
<td>3.9381e-11</td>
<td>2.6757e-26</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.9998</td>
<td>6.8798e-24</td>
<td>2.3196e-07</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0762</td>
<td>8.3773e-27</td>
<td>0.2499</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.9237</td>
<td>3.7763e-59</td>
<td>0.7500</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2.6318e-95</td>
<td>2.1622e-27</td>
<td>4.3797e-63</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5.5977e-215</td>
<td>3.7476e-138</td>
<td>0</td>
</tr>
</tbody>
</table>

We can see that the robot believes that it is 3m away from the door and is about 75% certain.

**Exercise 2:**

Here are several examples of learning algorithms:

- Mean-shift clustering: Unsupervised learning
- Perceptron algorithm: Discriminant function
- Bayes classifier: Generative model
- Conditional Random Field: Discriminative model
- AdaBoost: Discriminant function

For a detailed explanation, please see the textbook *Pattern Recognition and Machine Learning* by C.M. Bishop or the slides.
Exercise 3:

Abbildung 1: Tracker data from quadro copter. The lines are just an interpolation between the tracked positions (data points).

a) See figure 1.

b) The task is to estimate the speed of the quadrocopter. We do this using polynomial regression. The functions that we learn are dependent on time. We have to find three functions, one for each coordinate \((x, y, z)\). The regression is done with the matrix \(\Phi\) and vectors \(t_i\):

\[
\Phi = \begin{pmatrix}
1 & 0 \\
1 & 1 \\
1 & 2 \\
1 & 3 \\
1 & 4 \\
1 & 5
\end{pmatrix}
\]

\[
t_x = \begin{pmatrix}
2 \\
1.08 \\
-0.83 \\
-1.97 \\
-1.31 \\
0.57
\end{pmatrix}
\]

\[
t_y = \begin{pmatrix}
0 \\
1.68 \\
1.82 \\
0.28 \\
-1.51 \\
-1.91
\end{pmatrix}
\]

\[
t_z = \begin{pmatrix}
1 \\
2.38 \\
2.49 \\
2.15 \\
2.59 \\
4.32
\end{pmatrix}
\]

The second column of \(\Phi\) are the timestamps at which the measurements have been taken. In this first case, we assume constant velocity, i.e. we don’t have acceleration and the motion equation has only two unknowns \(w_0\) and \(w_1\), i.e. for the case of the \(x\)-coordinates we have

\[
x(\tau) = w_0 + w_1 \tau, \quad \mathbf{w}_x = (w_0, w_1)^T
\]

where \(\tau = 0, 1, \ldots\) is the time stamp. Thus, \(\Phi\) has two columns.

The pseudoinverse of \(\Phi\) is

\[
\Phi^\dagger = \begin{pmatrix}
0.524 & 0.381 & 0.238 & 0.095 & -0.048 & -0.190 \\
-0.143 & -0.086 & -0.029 & 0.029 & 0.086 & 0.143
\end{pmatrix}
\]
With this we compute \( w_i = \Phi^\dagger t_i \):
\[
\begin{align*}
w_{x1} &= \Phi^\dagger t_x = \begin{pmatrix} 1.0267 \\ -0.4421 \end{pmatrix} \\
w_y &= \begin{pmatrix} 1.5383 \\ -0.5918 \end{pmatrix} \\
w_z &= \begin{pmatrix} 1.2825 \\ 0.4830 \end{pmatrix}
\end{align*}
\]

To compute the speed we need \( v = (-0.4421, -0.5918, 0.4830)^T \). The speed is \( \|v\| = 0.8827 \).

The residual errors are defined as
\[
\begin{align*}
r_x &= \|\Phi w_x - t_x\| = 2.8902 \\
r_y &= \|\Phi w_y - t_y\| = 2.4571 \\
r_z &= \|\Phi w_z - t_z\| = 1.2807
\end{align*}
\]

c) Now we have a quadratic motion equation:
\[
x(\tau) = w_0 + w_1 \tau + w_2 \tau^2, \quad w_x = (w_0, w_1, w_2)^T,
\]
where \( w_1 \) is velocity and \( w_2 \) is (half the) acceleration. This means we have to estimate 3 function parameters. Thus, the matrix \( \Phi \) has one more column, i.e.
\[
\Phi = \begin{pmatrix}
1 & 0 & 0 \\
1 & 1 & 1 \\
1 & 2 & 4 \\
1 & 3 & 9 \\
1 & 4 & 16 \\
1 & 5 & 25
\end{pmatrix}
\]

Again, we compute the pseudoinverse and multiply it with the vectors \( t_i \). We obtain:
\[
\begin{align*}
w_x &= \begin{pmatrix} 2.4739 \\ -2.6128 \\ 0.4341 \end{pmatrix} \\
w_y &= \begin{pmatrix} 0.4573 \\ 1.0297 \\ -0.3243 \end{pmatrix} \\
w_z &= \begin{pmatrix} 1.4656 \\ 0.2084 \\ 0.0549 \end{pmatrix}
\end{align*}
\]

The residual errors are now
\[
\begin{align*}
r_x &= 1.1474 \\
r_y &= 1.4527 \\
r_z &= 1.2359
\end{align*}
\]

d) If we want to estimate the position in the next second we can imagine a new row in our \( \Phi \) matrix \( \phi_6 = (1 \ 6 \ 36) \). Multiplying this row with our model parameters \( w \) for the last model gives us the estimate:
\[
t'_6 = \phi_6 w = (2.4259 \ -5.0397 \ 4.6930)^T
\]

Exercise 4: Programming
See Code.