BONN-RHEIN-SIEG UNIVERSITY OF APPLIED SCIENCE
SCIENTIFIC EXPERIMENTATION AND EVALUATION

Motion Model Parameter Estimation: Sample Solution

In this exercise, we are trying to estimate the parameters of the velocity motion model described in Thrun et al. [1]. According to this model, the probability of a robot's pose at time t given a motion command and the pose at time $t-1$ is given as:

$$\mathrm{P}(x_t|x_{t-1},u_t) = \mathcal{N}(0,\alpha_1 v^2+\alpha_2\omega^2)\cdot\mathcal{N}(0,\alpha_3 v^2+\alpha_4\omega^2)\cdot\mathcal{N}(0,\alpha_5 v^2+\alpha_6\omega^2) \tag{1}$$

where v and ω are the commanded linear and angular velocities and the three normal distributions are evaluated at the points $v-\hat{v}$, $\omega-\hat{\omega}$, and γ respectively; $\hat{v}$, $\hat{\omega}$, and γ are calculated by algorithm 5.3 in [1].

The problem of estimating the motion model parameters can be approached as maximum likelihood estimation; in other words, we should maximise the posterior $\mathrm{P}(\alpha|X)$, where α is a vector representation of the six parameters and X is the observed motion data, such that each element of X is a quadruple $(x_{t-1}, x_t, u_t, \Delta t)$. Using Bayes' rule, we can rewrite the posterior as

$$\mathrm{P}(\alpha|X) = \frac{\mathrm{P}(X|\alpha)\mathrm{P}(\alpha)}{\mathrm{P}(X)} \tag{2}$$

Here, we can notice that $P(X)$ does not depend on the motion model parameters, so the optimisation is independent of this term; maximising the posterior is therefore equivalent to maximising the numerator. If we further assume that $P(\alpha)$ is a uniform distribution, maximising the posterior becomes equivalent to maximising the likelihood $\mathrm{P}(X|\alpha)$. For computational simplicity, we are going to maximise the log-likelihood instead of the likelihood.

Taking $\mathrm{P}(x_t|x_{t-1},u_t)$ to be $\mathrm{P}(x|\alpha)$ we have

$$\begin{aligned}\mathrm{P}(x|\alpha) &= \mathcal{N}(0,\alpha_1 v^2+\alpha_2\omega^2)\cdot\mathcal{N}(0,\alpha_3 v^2+\alpha_4\omega^2)\cdot\mathcal{N}(0,\alpha_5 v^2+\alpha_6\omega^2) \\ &= \frac{1}{\sqrt{2\pi}\sqrt{\alpha_1 v^2+\alpha_2\omega^2}}\,exp\left(-\frac{1}{2}\frac{(v-\hat{v})^2}{\alpha_1 v^2+\alpha_2\omega^2}\right)\cdot\frac{1}{\sqrt{2\pi}\sqrt{\alpha_3 v^2+\alpha_4\omega^2}}\,exp\left(-\frac{1}{2}\frac{(\omega-\hat{\omega})^2}{\alpha_3 v^2+\alpha_4\omega^2}\right) \\ &\cdot\frac{1}{\sqrt{2\pi}\sqrt{\alpha_5 v^2+\alpha_6\omega^2}}\,exp\left(-\frac{1}{2}\frac{\hat{\gamma}^2}{\alpha_5 v^2+\alpha_6\omega^2}\right)\end{aligned} \tag{3}$$

for a single data point x. The log-likelihood is then given as

$$\begin{aligned}\ln \mathrm{P}(x|\alpha) &= \ln\left(\frac{1}{\sqrt{2\pi}\sqrt{\alpha_1 v^2 + \alpha_2 \omega^2}}\right) - \frac{1}{2}\frac{(v-\hat{v})^2}{\alpha_1 v^2 + \alpha_2 \omega^2} + \ln\left(\frac{1}{\sqrt{2\pi}\sqrt{\alpha_3 v^2 + \alpha_4 \omega^2}}\right) - \frac{1}{2}\frac{(\omega-\hat{\omega})^2}{\alpha_3 v^2 + \alpha_4 \omega^2} \\ &+ \ln\left(\frac{1}{\sqrt{2\pi}\sqrt{\alpha_5 v^2 + \alpha_6 \omega^2}}\right) - \frac{1}{2}\frac{\hat{\gamma}^2}{\alpha_5 v^2 + \alpha_6 \omega^2} \\ &= -\frac{1}{2}\left[3\ln(2\pi) + \ln\left(\alpha_1 v^2 + \alpha_2 \omega^2\right) + \frac{(v-\hat{v})^2}{\alpha_1 v^2 + \alpha_2 \omega^2} + \ln\left(\alpha_3 v^2 + \alpha_4 \omega^2\right) + \frac{(\omega-\hat{\omega})^2}{\alpha_3 v^2 + \alpha_4 \omega^2}\right. \\ &\left. + \ln\left(\alpha_5 v^2 + \alpha_6 \omega^2\right) + \frac{\hat{\gamma}^2}{\alpha_5 v^2 + \alpha_6 \omega^2}\right]\end{aligned} \tag{4}$$

The derivatives of $\ln \mathrm{P}(x|\alpha)$ with respect to the motion model parameters can be found to be equal to:

$$\begin{aligned}\frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_1} &= -\frac{1}{2}\left(\frac{v^2}{\alpha_1 v^2 + \alpha_2 \omega^2} - \frac{(v-\hat{v})^2 v^2}{\left(\alpha_1 v^2 + \alpha_2 \omega^2\right)^2}\right) \\ \frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_2} &= -\frac{1}{2}\left(\frac{\omega^2}{\alpha_1 v^2 + \alpha_2 \omega^2} + \frac{(v-\hat{v})^2 \omega^2}{\left(\alpha_1 v^2 + \alpha_2 \omega^2\right)^2}\right) \\ \frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_3} &= -\frac{1}{2}\left(\frac{v^2}{\alpha_3 v^2 + \alpha_4 \omega^2} + \frac{(\omega-\hat{\omega})^2 v^2}{\left(\alpha_3 v^2 + \alpha_4 \omega^2\right)^2}\right) \\ \frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_4} &= -\frac{1}{2}\left(\frac{\omega^2}{\alpha_3 v^2 + \alpha_4 \omega^2} + \frac{(\omega-\hat{\omega})^2 \omega^2}{\left(\alpha_3 v^2 + \alpha_4 \omega^2\right)^2}\right) \\ \frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_5} &= -\frac{1}{2}\left(\frac{v^2}{\alpha_5 v^2 + \alpha_6 \omega^2} + \frac{\hat{\gamma}^2 v^2}{\left(\alpha_5 v^2 + \alpha_6 \omega^2\right)^2}\right) \\ \frac{\partial \ln \mathrm{P}(x|\alpha)}{\partial \alpha_6} &= -\frac{1}{2}\left(\frac{\omega^2}{\alpha_5 v^2 + \alpha_6 \omega^2} + \frac{\hat{\gamma}^2 \omega^2}{\left(\alpha_5 v^2 + \alpha_6 \omega^2\right)^2}\right)\end{aligned} \tag{5}$$

If we assume that the samples in X are independent, we have that

$$\mathrm{P}(X|\alpha) = \prod_{i=1}^{n} \mathrm{P}(x_i|\alpha) \tag{6}$$

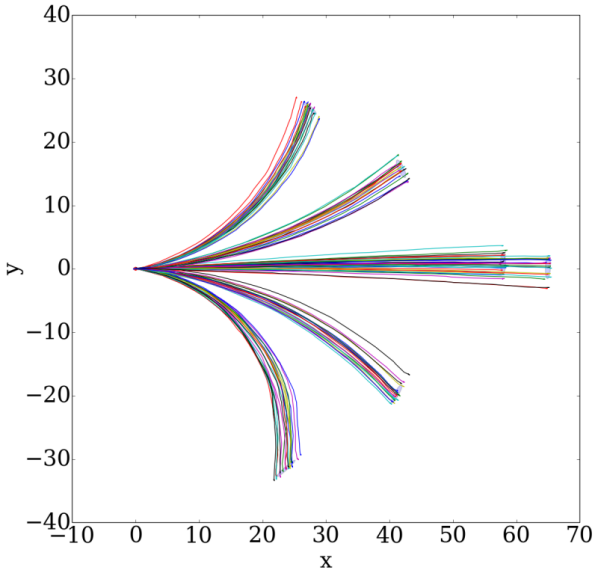
$$\implies \ln \mathrm{P}(X|\alpha) = \sum_{i=1}^{n} \ln \mathrm{P}(x_i|\alpha) \tag{7}$$

which means that

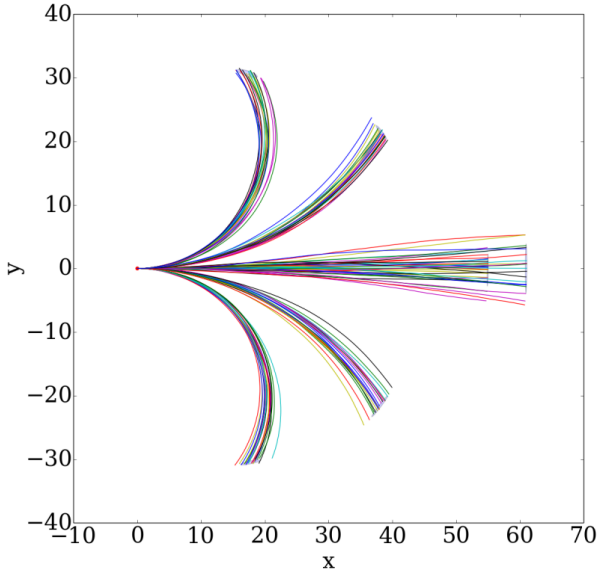
$$\frac{\partial \ln \mathrm{P}(X|\alpha)}{\partial \alpha_j} = \sum_{i=1}^{n} \frac{\partial \ln \mathrm{P}(x_i|\alpha)}{\partial \alpha_j} \tag{8}$$

This is an optimisation problem that can be solved by applying a numerical optimisation procedure, but we should note that we need to use constrained optimisation because the αs cannot be negative.

If we try to optimise the parameters using maximum likelihood estimation, we may obtain αs that



(a) Observed data



(b) Sampled trajectories

produce sampled trajectories resembling the ones shown in Figure 1b. As can be seen, the sampled data are fairly close to the observed data; however, the current model cannot deal with the fact that the observed left and right trajectories are slightly different, which is why we can also see that the sampled right trajectories are much closer to the observed ones than the sampled left trajectories. One way to avoid this problem would be to repeat the above procedure for each type of motion, thereby computing different sets of motion model parameters for different types of motion.

Sample Python code that preprocesses the data and performs the described optimisation - computing one global set of αs in the process - is given in the appendix.

References

[1] S. Thrun *et al.*, "Velocity Motion Model," in *Probabilistic Robotics*, ch. 5, pp. 121–132, Cambridge, MA: The MIT Press, 2006.

Appendix - Sample Code

Listing 1: parameter_optimisation.py

```
import numpy as np
from scipy.optimize import minimize

def dF1(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[0] * v**2 + alphas[1] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (v**2 / k - ((v - v_hat)**2 * v**2) / (k**2))
    return d

def dF2(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[0] * v**2 + alphas[1] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (omega**2 / k - ((v - v_hat)**2 * omega**2) / (k**2))
    return d

def dF3(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[2] * v**2 + alphas[3] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (v**2 / k - ((omega - omega_hat)**2 * v**2) / (k**2))
    return d

def dF4(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[2] * v**2 + alphas[3] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (omega**2 / k - ((omega - omega_hat)**2 * omega**2) / (k**2))
    return d

def dF5(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[4] * v**2 + alphas[5] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (v**2 / k - (gamma_hat**2 * v**2) / (k**2))
    return d

def dF6(alphas, v, omega, v_hat, omega_hat, gamma_hat):
    k = (alphas[4] * v**2 + alphas[5] * omega**2)
    if k < 1e-100:
        k = 1e-100
    d = -0.5 * (omega**2 / k - (gamma_hat**2 * omega**2) / (k**2))
    return d

def F(alphas, X):
    s = 0.
    for x in X:
        k = (alphas[0] * x[0]**2 + alphas[1] * x[1]**2)
        if k < 1e-100:
            k = 1e-100
        temp = -0.5 * (np.log(2 * np.pi) + np.log(k) + (x[0] - x[2])**2 / k)

        k = (alphas[2] * x[0]**2 + alphas[3] * x[1]**2)
        if k < 1e-100:
            k = 1e-100
        temp += -0.5 * (np.log(2 * np.pi) + np.log(k) + (x[1] - x[3])**2 / k)

        k = (alphas[4] * x[0]**2 + alphas[5] * x[1]**2)
        if k < 1e-100:
            k = 1e-100
        temp += -0.5 * (np.log(2 * np.pi) + np.log(k) + x[4]**2 / k)

        s += temp
    return -1. * s

def F_prime(alphas, X):
```

```
    d = np.zeros(len(alphas))
    for i in xrange(len(alphas)):
        s = 0.
        if i == 0:
            for x in X:
                s += dF1(alphas, *x)
        elif i == 1:
            for x in X:
                s += dF2(alphas, *x)
        elif i == 2:
            for x in X:
                s += dF3(alphas, *x)
        elif i == 3:
            for x in X:
                s += dF4(alphas, *x)
        elif i == 4:
            for x in X:
                s += dF5(alphas, *x)
        elif i == 5:
            for x in X:
                s += dF6(alphas, *x)
        d[i] = s
    return -1. * d

def optimise_parameters(data, init_alphas):
    bounds = [(0., None), (0., None), (0., None), (0., None), (0., None), (0., None)]
    r = minimize(F, init_alphas, args=(data,), method='L-BFGS-B', jac=F_prime, bounds=bounds, options={'disp': True})
    alphas = r.x
    return alphas
```

Listing 2: motion_model_mle.py

```
import glob
import os
import numpy as np
from parameter_optimisation import optimise_parameters

class Data(object):
    def __init__(self):
        self.trajectories = list()
        self.trajectories_end = list()
        self.time_deltas = list()

def read_data(directory_names):
    data_store = Data()
    for directory_name in directory_names:
        os.chdir(directory_name)
        trajectories = list()
        trajectories_end = list()

        files = list()
        for f in glob.glob("*.log"):
            files.append(f)

        for f in files:
            data = np.genfromtxt(f)
            poses = np.zeros((data.shape[0], 3))
            time_deltas = np.zeros(data.shape[0]-1)

            poses[0,0] = 0.
            poses[0,1] = 0.
            poses[0,2] = 0.
            for i in xrange(1, data.shape[0]):
                poses[i,0] = data[i,1] - data[0,1]
                poses[i,1] = data[i,2] - data[0,2]
                poses[i,2] = data[i,3] - data[0,3]
                time_deltas[i-1] = data[i,0] - data[i-1,0]

            trajectories.append(poses)
            trajectories_end.append(poses[-1])
            times.append(time_deltas)
```

```
40
41          data_store.trajectories.append(trajectories)
42          data_store.trajectories_end.append(trajectories_end)
43          data_store.time_deltas.append(times)
44
45          os.chdir('..')
46      return data_store
47
48  def motion_model_velocity(xt, ut, x_prevt, delta_t):
49      mu = 0.5 * ((x_prevt[0] - xt[0]) * np.cos(x_prevt[2]) + (x_prevt[1] - xt[1]) * np.sin(x_prevt[2])) / ((x_prevt[1] - xt[1]) *
            np.cos(x_prevt[2]) - (x_prevt[0] - xt[0]) * np.sin(x_prevt[2]))
50      x_star = 0.5 * (x_prevt[0] + xt[0]) + mu * (x_prevt[1] - xt[1])
51      y_star = 0.5 * (x_prevt[1] + xt[1]) + mu * (xt[0] - x_prevt[0])
52      r_star = np.sqrt((x_prevt[0] - x_star)**2 + (x_prevt[1] - y_star)**2)
53      delta_theta = np.arctan2(xt[1] - y_star, xt[0] - x_star) - np.arctan2(x_prevt[1] - y_star, x_prevt[0] - x_star)
54      v_hat = delta_theta / delta_t * r_star
55      omega_hat = delta_theta / delta_t
56      gamma_hat = (xt[2] - x_prevt[2]) / delta_t - omega_hat
57      return np.array([ut[0], ut[1], v_hat, omega_hat, gamma_hat])
58
59  directory_names = ['straight_short', 'straight_long', 'left_deep', 'left_shallow', 'right_deep', 'right_shallow']
60  v = [-22., -24., -17., -18., -17., -18.]
61  omega = [0., 0., np.radians(-48.), np.radians(-24.), np.radians(48.), np.radians(24.)]
62  data = read_data(directory_names)
63
64  motion_model_data = list()
65  for i, trajectories in enumerate(data.trajectories):
66      for j, trajectory in enumerate(trajectories):
67          for k in xrange(len(trajectory)-1):
68              delta_t = data.time_deltas[i][j][k]
69              mmv = motion_model_velocity(trajectory[k+1], (v[i], omega[i]), trajectory[k], delta_t)
70              motion_model_data.append(mmv)
71  motion_model_data = np.array(motion_model_data)
72
73  init_alphas = np.random.uniform(1e-7, 1e-5, 6)
74  alphas = optimise_parameters(motion_model_data, init_alphas)
```