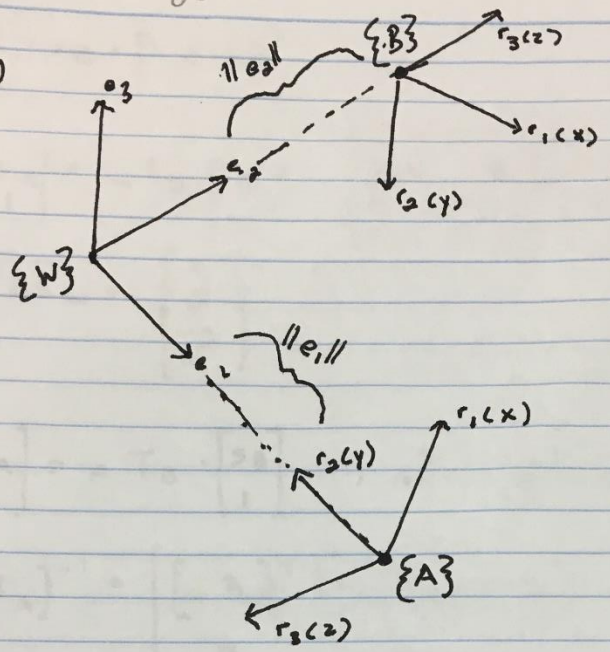
Joseph Bell
ECE 276A
HW 2

1)

a)



b) $\{A\}$

$r_1 = e_3$, $r_2 = -e_1$, $r_3 = -e_2$ $\Rightarrow [r_1, r_2, r_3] = [e_3, -e_1, -e_2]$

$p = 3 \cdot e_1$

$$\therefore \quad {}^wT_A = \begin{bmatrix} 0 & -1 & 0 & 3 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\{B\}$

$r_1 = e_1$, $r_2 = -e_3$, $r_3 = e_2$ $\Rightarrow (r_1, r_2, r_3) = [e_1, -e_3, e_2]$

$p = 2 \cdot e_2$

$$\therefore \quad {}^wT_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) $s_B = [1\ 2\ 3]^T$

$$\begin{bmatrix} s_w \\ 1 \end{bmatrix} = {}_w T_B \cdot \begin{bmatrix} s_B \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ -2 \\ 1 \end{bmatrix}$$

$$\therefore s_w = \begin{bmatrix} 1 \\ 5 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} s_A \\ 1 \end{bmatrix} = {}_A T_B \cdot \begin{bmatrix} s_B \\ 1 \end{bmatrix} \quad , {}_A T_B = {}_A T_w \cdot {}_w T_B = \left[{}_w T_A \right]^{-1} \cdot {}_w T_B$$

$$\left[{}_w T_A \right]^{-1} = \begin{bmatrix} \left[{}_w R_A \right]^T & \left[-{}_w R_A \right]^T \cdot {}_w P_A \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} s_A \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} s_A \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ 5 \\ 1 \end{bmatrix} \Rightarrow \therefore s_A = \begin{bmatrix} -2 \\ 2 \\ 5 \end{bmatrix}$$

d) $f = 0.2m$, $(C_u, C_v) = (160.5, 120.5)$, $(s_u, s_v) = (10, 10)$, $s_t = 0$

$p = [-1\ 1\ 0]^T$, $q = [\sqrt{3}/2\ 0\ 0\ 1/2]^T$

$$R = \begin{bmatrix} 0.5 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ ← from quaternion conversion done in python

$R(q) = E(q) \cdot G(q)^T$

optical coordinates

$$^oR_r = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} {}^oR_r \cdot R^T & -{}^oR_r \cdot R^T \cdot p \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}/2, & -1/2, & 0, & \sqrt{3}/2 + 0.5 \\ 0, & 0, & -1, & 0 \\ 1/2, & \sqrt{3}/2, & 0, & -\sqrt{3}/2 + 0.5 \\ 0, & 0, & 0, & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ -2 \\ 1 \end{bmatrix}$$ ← world coordinates

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} = \begin{bmatrix} -0.26795 \\ 2 \\ 4.46410 \\ 1 \end{bmatrix}$$

★ Full calculations on attached python script

pixel coordinates

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot s_u & f \cdot s_t & C_u \\ 0 & f \cdot s_v & C_v \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{Z_o} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 160.38 \\ 121.40 \\ 1 \end{bmatrix}, \quad \therefore \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 160.38 \\ 121.40 \end{bmatrix}$$

2)

[**All code in attached python script**]

For this problem I used two different approaches that give different treatments to the boundary conditions of the world upon shifting the data. The first approach assumes that applying a control unit shifts edge bins off the world – i.e. a control unit of +1 shifts bin 24 into bin 25 and bin 25 is removed. The second approach accumulates the shift at the edges – i.e. a control unit of +1 shifts bin 23 into bin 24 and bin 24 is appended to bin 25 (i.e. no data is shifted off the world). The specification of the problem that a control unit forces a state to be greater than 25 resets the state to 25 and vice versa for state 0 is not clear as to if this applies to the motion model only or if it also applies to the data shift. Hence, I did both and compared the results. The motion model accounts for moving off the world by compounding the probability of moving to the end bin and the probability of moving past the end bin. In other words, if the intent is to move 1 unit into the last bin, then the probability of moving into that bin is the sum of the probability of moving into the bin ($\frac{1}{2}$) and the probability of moving past the end bin ($\frac{1}{3}$) which results in $\frac{5}{6}$. The double accumulation of probabilities in the second approach results in some odd results.

The method to find the most likely position at t=0 after solving for the most likely position at t=4 was to simply back track the control units under the assumption that the robot moved as commanded. This assumption, however, is not unwarranted as the most likely movement based on the motion model is that the robot moves as commanded (the correct movement has a probability of $\frac{1}{2}$ as opposed to the $\frac{1}{3}$ and $\frac{1}{6}$ probabilities of over and under stepping). With the most likely position of approach 1 being at bin 13 at t=4, **the most likely position at t=0 by back tracking the control units is bin 9**. With the most likely position of approach 2 being at bin 25 at t=4, **the most likely position at t=0 by back tracking the control units is bin 21**. The reason I prefer the result of approach 1 is because the observations at t=0 and t=4 are both doors, and in approach 1 the position at t=0 is bin 9 with a 70% probability of observing a door (since we know a door is at bin 8) and the position at t=4 is bin 13 with a 90% probability of observing a door (since we know a door is at bin 13). However, in approach 2 the positions at t=0 and t=4 only have a 20% chance of observing a door. The second most likely position of approach 2, however, is the same as the most likely position of approach 1.

Figure 1 illustrates the probability distribution at t=4 for the first approach and figure 2 illustrates the probability distribution at t=4 of the second approach. In my opinion, the first approach makes more sense and it also results in a more desirable output. However, approach 2 follows more closely to how the problem is set up and how the professor has explained to treat the boundaries.

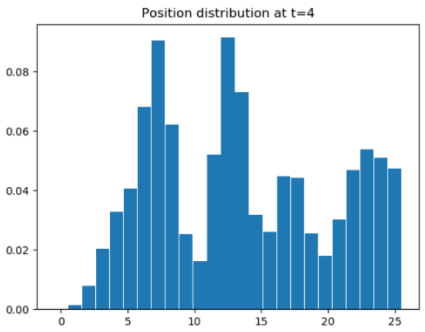


Figure 1: position distribution of approach 1

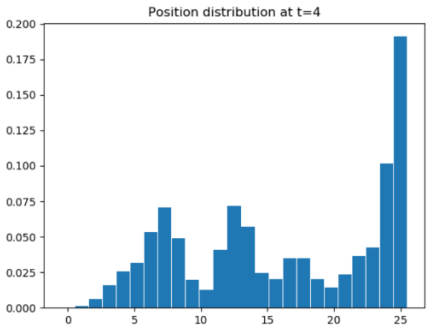


Figure2: position distribution of approach 2

The 10x25 table of the update and prediction steps for approach 1 is below:

0.0198	0.0198	0.0198	0.0198	0.0198	0.0198	0.0693	0.0891	0.0693	0.0198	0.0198	0.0693	0.0891	0.0693	0.0198	0.0693	0.0891	0.0693	0.0198	0.0198	0.0198	0.0198	0.0198	0.0198	0.0198
0.0066	0.0165	0.0198	0.0198	0.0198	0.0198	0.0363	0.0677	0.0792	0.0561	0.0281	0.0363	0.0677	0.0792	0.0561	0.0446	0.0677	0.0792	0.0561	0.0281	0.0198	0.0198	0.0198	0.0198	0.0198
0.0113	0.0282	0.0339	0.0339	0.0339	0.0339	0.0233	0.0145	0.0508	0.096	0.048	0.0233	0.0145	0.0508	0.096	0.0286	0.0145	0.0508	0.096	0.048	0.0339	0.0339	0.0339	0.0339	0.0339
0.0292	0.032	0.0339	0.0339	0.0321	0.0254	0.0235	0.0463	0.073	0.0599	0.0301	0.0235	0.0463	0.0697	0.0487	0.0252	0.0463	0.073	0.0617	0.0386	0.0339	0.0339	0.0339	0.0282	0.0113
0.0422	0.0463	0.049	0.049	0.0464	0.0367	0.0127	0.0084	0.0395	0.0866	0.0435	0.0127	0.0084	0.0378	0.0704	0.0137	0.0084	0.0395	0.0891	0.0558	0.049	0.049	0.049	0.0408	0.0163
0.0	0.0141	0.0365	0.0465	0.0485	0.0481	0.0436	0.0303	0.0153	0.0195	0.05	0.0644	0.0404	0.0164	0.0189	0.0438	0.0461	0.0214	0.0196	0.0509	0.0697	0.0591	0.0501	0.049	0.049,
0.0	0.0195	0.0506	0.0645	0.0673	0.0668	0.0227	0.0053	0.0079	0.027	0.0694	0.0335	0.007	0.0085	0.0262	0.0228	0.008	0.0111	0.0272	0.0706	0.0968	0.0819	0.0695	0.0679	0.0679
0.0	0.0065	0.0266	0.0501	0.0631	0.0667	0.0522	0.0242	0.0091	0.0139	0.038	0.0504	0.0306	0.0119	0.0142	0.0221	0.0184	0.0115	0.016	0.039	0.0721	0.0875	0.0803	0.0711	0.0682
0.0	0.0041	0.0167	0.0314	0.0396	0.0419	0.1146	0.0684	0.0199	0.0087	0.0238	0.1107	0.0866	0.0262	0.0089	0.0486	0.052	0.0253	0.01	0.0245	0.0453	0.0549	0.0504	0.0446	0.0428
0.0	0.0014	0.0076	0.0195	0.0317	0.039	0.0657	0.0871	0.06	0.024	0.0156	0.0503	0.0882	0.0705	0.0305	0.025	0.0431	0.0425	0.0246	0.0174	0.029	0.045	0.0518	0.0492	0.0456

The 10x25 table of the update and prediction steps for approach 2 is below:

0.0198	0.0198	0.0198	0.0198	0.0198	0.0198	0.0693	0.0891	0.0693	0.0198	0.0198	0.0693	0.0891	0.0693	0.0198	0.0693	0.0891	0.0693	0.0198	0.0198	0.0198	0.0198	0.0198	0.0198	0.0198
0.0066	0.0165	0.0198	0.0198	0.0198	0.0198	0.0363	0.0677	0.0792	0.0561	0.0281	0.0363	0.0677	0.0792	0.0561	0.0446	0.0677	0.0792	0.0561	0.0281	0.0198	0.0198	0.0198	0.0264	0.0363
0.0109	0.0272	0.0326	0.0326	0.0326	0.0326	0.0224	0.0139	0.0489	0.0924	0.0462	0.0224	0.0139	0.0489	0.0924	0.0275	0.0139	0.0489	0.0924	0.0462	0.0326	0.0326	0.0326	0.0435	0.0598
0.0281	0.0308	0.0326	0.0326	0.0309	0.0244	0.0226	0.0445	0.0702	0.0576	0.0289	0.0226	0.0445	0.0671	0.0469	0.0243	0.0445	0.0702	0.0593	0.0371	0.0326	0.0344	0.0426	0.0444	0.0199
0.0399	0.0438	0.0463	0.0463	0.0439	0.0347	0.012	0.0079	0.0374	0.0819	0.0411	0.012	0.0079	0.0357	0.0666	0.0129	0.0079	0.0374	0.0843	0.0528	0.0463	0.0489	0.0605	0.0631	0.0283
0.0	0.0133	0.0345	0.044	0.0459	0.0455	0.0412	0.0287	0.0144	0.0184	0.0473	0.0609	0.0382	0.0155	0.0179	0.0414	0.0436	0.0202	0.0186	0.0481	0.066	0.0559	0.0483	0.0828	0.1347
0.0	0.0165	0.0428	0.0544	0.0568	0.0564	0.0191	0.0044	0.0067	0.0228	0.0586	0.0283	0.0059	0.0072	0.0221	0.0192	0.0067	0.0094	0.023	0.0596	0.0817	0.0692	0.0598	0.1025	0.1668
0.0	0.0055	0.0225	0.0423	0.0533	0.0563	0.044	0.0204	0.0076	0.0117	0.0321	0.0425	0.0259	0.0101	0.012	0.0187	0.0155	0.0097	0.0135	0.0329	0.0609	0.0738	0.0681	0.1312	0.2344
0.0	0.0034	0.0141	0.0265	0.0335	0.0353	0.0968	0.0578	0.0168	0.0073	0.0201	0.0934	0.0731	0.0221	0.0075	0.041	0.0439	0.0213	0.0085	0.0207	0.0382	0.0464	0.0428	0.0824	0.1471
0.0	0.0011	0.0064	0.0165	0.0268	0.0329	0.0555	0.0735	0.0506	0.0205	0.0132	0.0424	0.0744	0.0595	0.0257	0.0211	0.0364	0.0359	0.0208	0.0147	0.0245	0.038	0.0438	0.1056	0.1984

Joseph Bell
ECE 276A
HW 2

(3)

a) $z_t = h(x_t, v_t) = v_t / x_t$ ⟸ observation model function

$p_{v_t}(v) = \beta e^{-\beta v}$, $z = v/x_t = h(v, x_t)$

$v = z \cdot x_t = h^{-1}(z, x_t)$

$\frac{dh^{-1}}{dz} = x_t$

$z = z_t$

$p_h(z_t | x_t) = p_v(z_t \cdot x_t) \cdot \frac{dh^{-1}}{dz}$

$p_h(z_t | x_t) = x_t \cdot \beta e^{-\beta(z_t \cdot x_t)}$ - $\int_{-\infty}^{\infty} p_h = 1$

let $x_t \cdot \beta = \gamma$ $\Rightarrow \int_{-\infty}^{\infty} \gamma e^{-\gamma z_t} dz_t = 0 + \int_0^{\infty} \gamma e^{-\gamma z_t} dz_t$

$= -e^{-\gamma z_t} \Big|_{z_t = 0}^{z_t = \infty}$ $\Rightarrow$ $0 + 1 = 1$ ✓

b) $p(x_0 | z_0) = \frac{p(z_0 | x_0) \cdot p(x_0)}{p(z_0)} = \frac{p(z_0 | x_0) \cdot p(x_0)}{\int p(z_0 | x_0) \cdot p(x_0) \cdot dx_0}$

$= \frac{x_0 \beta e^{-\beta(z_0 \cdot x_0)} \cdot \lambda e^{-\lambda \cdot x_0}}{\int_0^{\infty} x_0 \beta e^{-\beta(z_0 \cdot x_0)} \cdot \lambda e^{-\lambda \cdot x_0} \cdot dx_0}$

3 b) continued

$$= \frac{x_0 \cdot \beta \cdot \lambda \cdot e^{-\beta(z_0 \cdot x_0) - \lambda x_0}}{\left\{ \frac{\beta\lambda}{(\beta \cdot z_0 \cdot \lambda)^2} \right\}}$$

obtained from online integral calculator

$$\int_{-\infty}^{\infty} p(x_0|z_0)\,dx_0 = 1$$

$$= x_0 \cdot (\beta \cdot z_0 \cdot \lambda)^2 e^{-\beta(z_0 \cdot x_0) - \lambda x_0}$$

$$p(x_0|z_0) = x_0 (\beta z_0 + \lambda)^2 e^{x_0(-\beta z_0 - \lambda)}$$

c) $x_{t+1} = \alpha \cdot x_t, \quad \alpha > 0$ Motion model function

$$P_{t+1|t} = \begin{cases} 1, & x_{t+1} = \alpha x_t \\ 0, & \text{otherwise} \end{cases}$$

PMF, same as $\int$ dirac delta

d) $P(x_1|z_0) = P(x_1|x_0) \cdot P(x_0|z_0)$

$$= \sum P(x_1|x_0) \cdot x_0 \cdot (\beta z_0 + \lambda)^2 e^{x_0(-\beta z_0 - \lambda)}$$

equals 1 when $x_1 = \alpha x_0$ and 0 otherwise

$$P(x_1|z_0) = 1 \cdot \frac{x_1}{\alpha} \cdot (\beta z_0 + \lambda)^2 e^{x_1/\alpha(-\beta z_0 - \lambda)}$$

$$\int_{-\infty}^{\infty} p(x_1|z_0)\,dx_1 = 1 \quad \text{with online calculator}$$

CODE:

```
import numpy as np
import matplotlib.pyplot as plt

##############################
# CALCULATIONS FOR NUMBER 1d #
##############################

p = np.array([[-1],[1],[0]])
world_coordinates = np.array([[1],[5],[-2],[1]])

q0 = np.sqrt(3)/2
q1 = 0
q2 = 0
q3 = 1/2

R11 = 1 - 2*(q2**2 + q3**2)
R12 = 2*(q1*q2-q0*q3)
R13 = 2*(q0*q2 + q1*q3)
R21 = 2*(q1*q2 + q0*q3)
R22 = 1 - 2*(q1**2+q3**2)
R23 = 2*(q2*q3 - q0*q1)
R31 = 2*(q1*q3 - q0*q2)
R32 = 2*(q0*q1 + q2*q3)
R33 = 1 - 2*(q1**2 + q2**2)

Rotation_Matrix = np.array([[R11, R12, R13],[R21, R22, R23],[R31, R32, R33]])

R_or = np.array([[0, -1, 0],[0, 0, -1],[1,0,0]])
M11 = R_or@Rotation_Matrix.T
M12 = -1*R_or@Rotation_Matrix.T@p
M = np.concatenate((M11,M12), axis=1)
M = np.concatenate((M, np.array([[0,0,0,1]])), axis=0)
optical_coordinates = M@world_coordinates
print(optical_coordinates)
K = np.array([[0.2*10,0.2*0,160.5],[0,0.2*10,120.5],[0,0,1]])
pixel_coordinates =
K@(1/optical_coordinates[2,0]*np.array([[1,0,0,0],[0,1,0,0],[0,0,1,0]]))@optical_coordinates
print(pixel_coordinates)


##############################
# CALCULATIONS FOR NUMBER 2 #
##############################

def prob_z_given_x(z, x_index):
        front_of_door = [7, 12, 16]
```

```
    near_door = [6, 8, 11, 13, 15, 17]
    prob_z = 0
    if x_index in front_of_door:
        prob_z = 0.9
    elif x_index in near_door:
        prob_z = 0.7
    else:
        prob_z = 0.20
    if z == -1:
        prob_z = 1 - prob_z

    return prob_z


def prediction_step(u ,states):
    # shift data
    states_moved = [0]*25
    states_predicted = [0]*25
    for x in range(25):
        new_index = x + u

        if new_index > 24:
            states_moved[24] = states_moved[24] + states[x]
            # continue

        elif new_index < 0:
            states_moved[0] = states_moved[0] + states[x]
            # continue
        else:
            states_moved[new_index] = states[x]
        # states_moved[new_index] = states_moved[new_index] + states[x]
        # states_moved[new_index] = states[x]




    if u < 0:
        conv_kernel = [1/6, 1/2, 1/3]
    elif u > 0:
        conv_kernel = [1/3, 1/2, 1/6]
    elif u == 0:
        conv_kernel = [0, 1, 0]


    prediction = np.convolve(conv_kernel, states_moved, 'same')
    if u < 0:
        prediction[0] = prediction[0] + states_moved[0]*1/3
        prediction[24] = prediction[24] + states_moved[24]*1/6
    elif u > 0:
        prediction[0] = prediction[0] + states_moved[0]*1/6
```

```
                prediction[24] = prediction[24] + states_moved[24]*1/3


        return prediction

def update_step(z, states):
        states_update = states.copy()
        for i in range(25):
                p_z = prob_z_given_x(z, x_index=i)*states[i]
                states_update[i] = p_z
        return states_update

states_init= [1/25]*25
observations = [1, -1, -1, -1, 1]
controls = [1, -1, 2, 1, 1]
solution_grid = np.zeros((10,25))
state = states_init
for j in range(5):
        updated_state = update_step(z=observations[j], states=state)
        state = updated_state / np.sum(updated_state)

        for i in range(25):
                solution_grid[j*2,i] = round(state[i], 4)
        state = prediction_step(u=controls[j], states=state)

        for i in range(25):
                solution_grid[j*2+1,i] = round(state[i], 4)
normalize_state = state/np.sum(state)

plt.bar(np.linspace(0,25,25), normalize_state, width=1.0)
plt.title("Position distribution at t=4")
plt.show()
```