AA 274A: Principles of Robot Autonomy I
Problem Set HW3

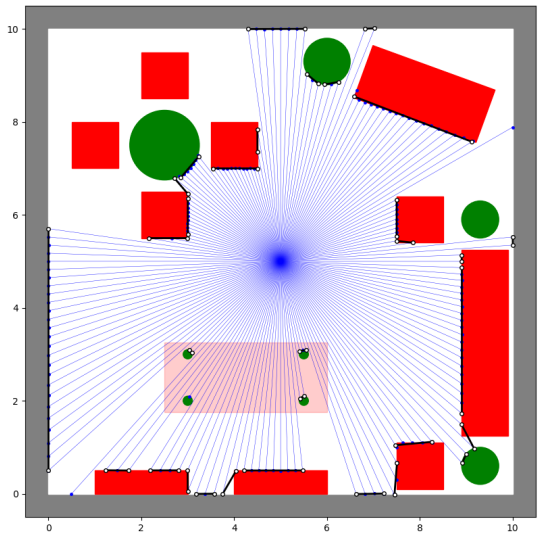
SUID:06734162
Name: Karthik Pythireddi
HW Group: Alberto Guiggiani, Avrum Noor, Josie Oetjen

Problem 2: Line Extraction

As mentioned in the Problem 2, We used different parameter values specific to each data set to be able to tune and better fit the lines of each set of range data.

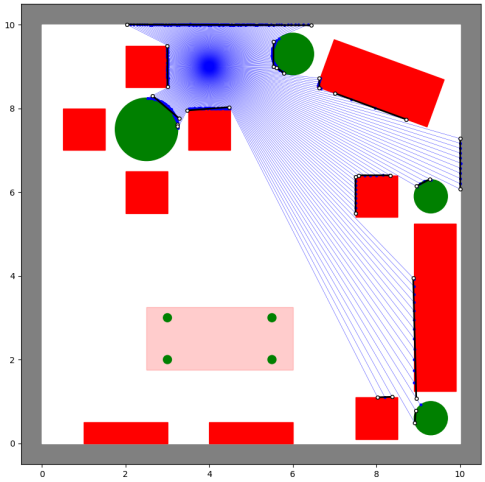
Parameters used for the line extraction as part of the data set `rangeData_5_5_180.csv` :
`MIN_SEG_LENGTH` = 0.05
`LINE_POINT_DIST_THRESHOLD` = 0.04
`MIN_POINTs_PER_SEGMENT` = 2
`MAX_P2P_DIST` = 0.7

Figure 1: Plot for the extracted lines using the range data `rangeData_5_5_180.csv`



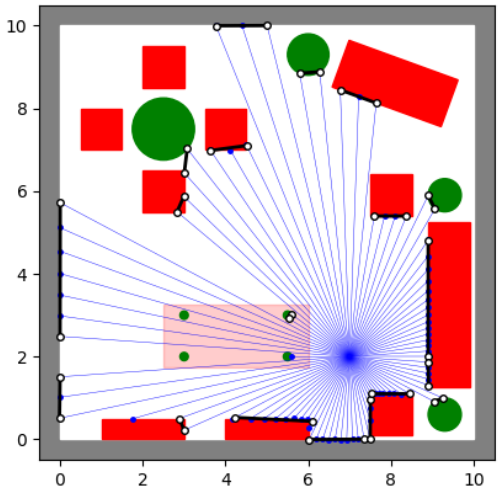
Parameters used for the line extraction as part of the data set rangeData_4_9_360.csv :
MIN_SEG_LENGTH = 0.04, LINE_POINT_DIST_THRESHOLD = 0.08
MIN_POINTs_PER_SEGMENT = 2, MAX_P2P_DIST = 0.8

Figure 2: Plot for the extracted lines using the range data rangeData_4_9_360.csv



Parameters used for the line extraction as part of the data set rangeData_7_2_90.csv :
MIN_SEG_LENGTH = 0.08, LINE_POINT_DIST_THRESHOLD = 0.06
MIN_POINTs_PER_SEGMENT = 2, MAX_P2P_DIST = 0.6

Figure 3: Plot for the extracted lines using the range data rangeData_7_2_90.csv



Problem 3: Linear Filtering

(i) From the Problem, we have an Image $I, F \in \mathbb{R}$ with a order of m x n and k x l respectively. We can calculate the output of the image $G(i,j)$ by expanding on the below given set of equations.

$$Given \quad G(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{l-1} F(u,v).\bar{I}(i+u, j+v) \tag{1}$$

If we consider the *section*(a) in the Problem 3, we have a filter F in the order of 3 x 3. We can plug the order of F in the k and l of the (1). Then we can expand over the summation from $v = 0$ to 2 to have the (1) as below.

$$G(i,j) = \sum_{u=0}^{3-1} \sum_{v=0}^{3-1} [F(u,v).\bar{I}(0+u)(0+v) \tag{2}$$

We will consider the calculation of the output image G at the pixels values of $i = 0$ and $j = 0$

$$G(0,0) = \sum_{u=0}^{2} [(F(u,0)\bar{I}(u,0) + (F(u,1)\bar{I}(u,1) + (F(u,2)\bar{I}(u,2)] \tag{3}$$

Expand the summation of the output G from $u = 0$ to 2 to get the respective summation for the above mentioned pixel location in the output image G.

$$G(0,0) = [(F(0,0)\bar{I}(0,0) + (F(0,1)\bar{I}(0,1) + (F(0,2)\bar{I}(u,2)] + [(F(1,0)\bar{I}(1,0) + (F(1,1)\bar{I}(1,1) + (F(1,2)\bar{I}(1,2)] + [(F(2,0)\bar{I}(2,0) + (F(2,1)\bar{I}(2,1) + (F(2,2)\bar{I}(2,2)] \tag{4}$$

Using the calculation we used in (4) for the pixels values of $i = 0$ and $j = 0$, we can expand in the same fashion with respect to the other pixels locations. By calculating the other pixel values, we arrive at the below output for the respective filters.

(a) For the filter F $= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$

(b) For the filter F $= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 7 & 4 \\ 0 & 8 & 5 \end{bmatrix}$

(c) For the filter F $= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} -13 & -15 & -7 \\ -4 & -6 & -4 \\ 13 & 15 & 7 \end{bmatrix}$

Explanation: For the Filter F in the part c, we have an resultant output matrix G. From the matrix coefficients, it looks like the filter F will help in identifying the change in the pixels of a image Horizontally.

(c) For the filter $F^1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} 9 & -12 & -9 \\ 15 & -18 & -15 \\ 11 & -12 & -11 \end{bmatrix}$

Explanation: For the Filter F^1 in the part c, we have an resultant output matrix G. From the matrix coefficients, it looks like the filter F^1 will help in identifying the change in the pixels of a image

vertically.

(d) For the filter $F = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} 3.5625 & 3.25 & 1.3125 \\ 5.25 & 5 & 2.25 \\ 4.3125 & 4.25 & 2.0625 \end{bmatrix}$

Explanation: The above given filter looks like is a Gaussian smoothing filter which is also similar to a moving average filter. Here the filtering mask is symmetric so we should see similar results for correlation and convolution operations. The main difference here is that this filter provides more weight to the neighboring pixels that are closer.This kind of filter can also be called as a low pass filter

(d) For the filter $F^1 = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ over the Image I $= \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$ output image G $= \begin{bmatrix} 2.666 & 3 & 1.333 \\ 4.333 & 5 & 2.333 \\ 3.111 & 3.666 & 1.777 \end{bmatrix}$

Explanation: From the filter's matrix coefficients, this is a moving average filter. It returns the average of the pixels in the mask. This filter will return a smoothing effect by removing the sharp features of an image. Similar to the filter F above, this will also return similar results for correlation and convolution given the symmetry of the filter mask.

(ii) Below is the derivation part which proves that we can write a correlation as a dot product of two specially formed vectors.

$$Given \quad G(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{l-1} \sum_{w=0}^{c-1} F(u,v,w).\bar{I}(i+u, j+v, w) \tag{5}$$

Expanding on the $eq.(5)$, we arrive at the below step.

$$G(i,j) = \sum_{w=0}^{c-1} [\sum_{u=0}^{k-1} [\sum_{v=0}^{l-1} [F(u,v,w).\bar{I}(i+u, j+v, w)]] \tag{6}$$

$$\begin{aligned} G(i,j) = \sum_{w=0}^{c-1} [\sum_{u=0}^{k-1} [F(u,0,w).\bar{I}(i+u, j, w) + \sum_{u=1}^{k-1} [F(u,1,w).\bar{I}(i+u, j+1, w) \\ + \sum_{u=2}^{k-1} [F(u,2,w).\bar{I}(i+u, j+2, w) ..... \sum_{u=k-1}^{k-1} [F(u,k-1,w).\bar{I}(i+u, j+k-1, w)] \end{aligned} \tag{7}$$

If we expand it further from $w = 0$ to $w = c - 1$, we have the $eq.(7)$ as below.

$$\begin{aligned} G(i,j) = \sum_{w=0}^{c-1} [F(0,0,w).\bar{I}(i,j,w) + F(0,1,w).\bar{I}(i,j+1,w) + ...] \\ \sum_{w=1}^{c-1} [F(0,0,w).\bar{I}(i,j,w) + F(0,1,w).\bar{I}(i,j+1,w) + ...]. \\ . \\ . \end{aligned} \tag{8}$$

If we can separate the output G as individual vectors of f and t_{ij} by flattening the respective filter F and the associated pixel values such that their product delivers a scalar as an output , $eq.(8)$ can be written as below:

$$f^T = \begin{bmatrix} \mathrm{f}_w^T = 0 \\ \mathrm{f}_w^T = 1 \\ . \\ . \\ . \end{bmatrix}, \; t_{ij} = \begin{bmatrix} \mathrm{t}_w = 0 & t_w = 1 & t_w = 2..... \end{bmatrix}$$

(iv) The correlation function run-time on my personal machine for different filters are as below:

Correlation function runtime for filt0: $1.0704290866851807s$
Correlation function runtime for filt1: $1.0742990970611572s$
Correlation function runtime for filt2: $1.1041593551635742s$
Correlation function runtime for filt3: $1.1621050834655762s$

Explanation: The above mentioned correlation function run time is higher than the given $20ms$.

But i think we can use the below two ways to speed up and reduce the run time:

1. We can use the outer product implementation to reduce the computation time, we can notice the advantage of this when we have higher order matrices. For our case, we used a filter of the size 3 x 3 matrix for which we needed to a total of 9 computations i.e (3*3). But if we use the $F = f.f^T$ implementation, we only need to do 6 computations (2*3). As we scale up in the order of the filters, we can notice the computational advantage of having the outer product.

2. If we can have these correlation computations done in a concurrent way i.e parallel computing of the correlation calculations will help in optimizing the correlation function run time.

(v) From the $eq(1)$, we can split the filter F into f and f^T.

$$From \quad G(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{l-1} F(u,v).\bar{I}(i+u, j+v) \tag{9}$$

$$G(i,j) = \sum_{u=0}^{k-1} \sum_{v=0}^{l-1} f_u.f_v\bar{I}(i+u, j+v) \tag{10}$$

$$G(i,j) = \sum_{u=0}^{k-1} f_u \sum_{v=0}^{l-1} f_v\bar{I}(i+u, j+v) \tag{11}$$

Here f_u and f_v are the respective f and f^T. Let's suppose the order of the matrix in our case is 2 x 2. Then $f = \begin{bmatrix} \mathrm{f}_0 \\ \mathrm{f}_1 \end{bmatrix}$ and $f^T = \begin{bmatrix} \mathrm{f}_0 & \mathrm{f}_1 \end{bmatrix}$. If we expand the $F = f.f^T$, we arrive at the output matrix as $F = \begin{bmatrix} \mathrm{f}_0^2 & \mathrm{f}_0.f_1 \\ \mathrm{f}_1.f_0 & \mathrm{f}_1^2 \end{bmatrix}$. This can be expanded to any other order of filter matrices as well. Also the filter F needs to be symmetric and the values along the main diagonal are squared. We can calculate the vector f by taking the square root of the squared values along the diagonal of the the matrix.

(*vii*) In convolution, we flip the filter along both the axes first, and then we perform the correlation operation. Also convolution is associative, which helps to combine the smoothing and and edge detection into one filter i.e $(F) \circledast (G \circledast I) = ((F) \circledast G) \circledast I$

Figure 4: Correlation output plots using the filters in the `linear_filter.py` against the given image `test_card`

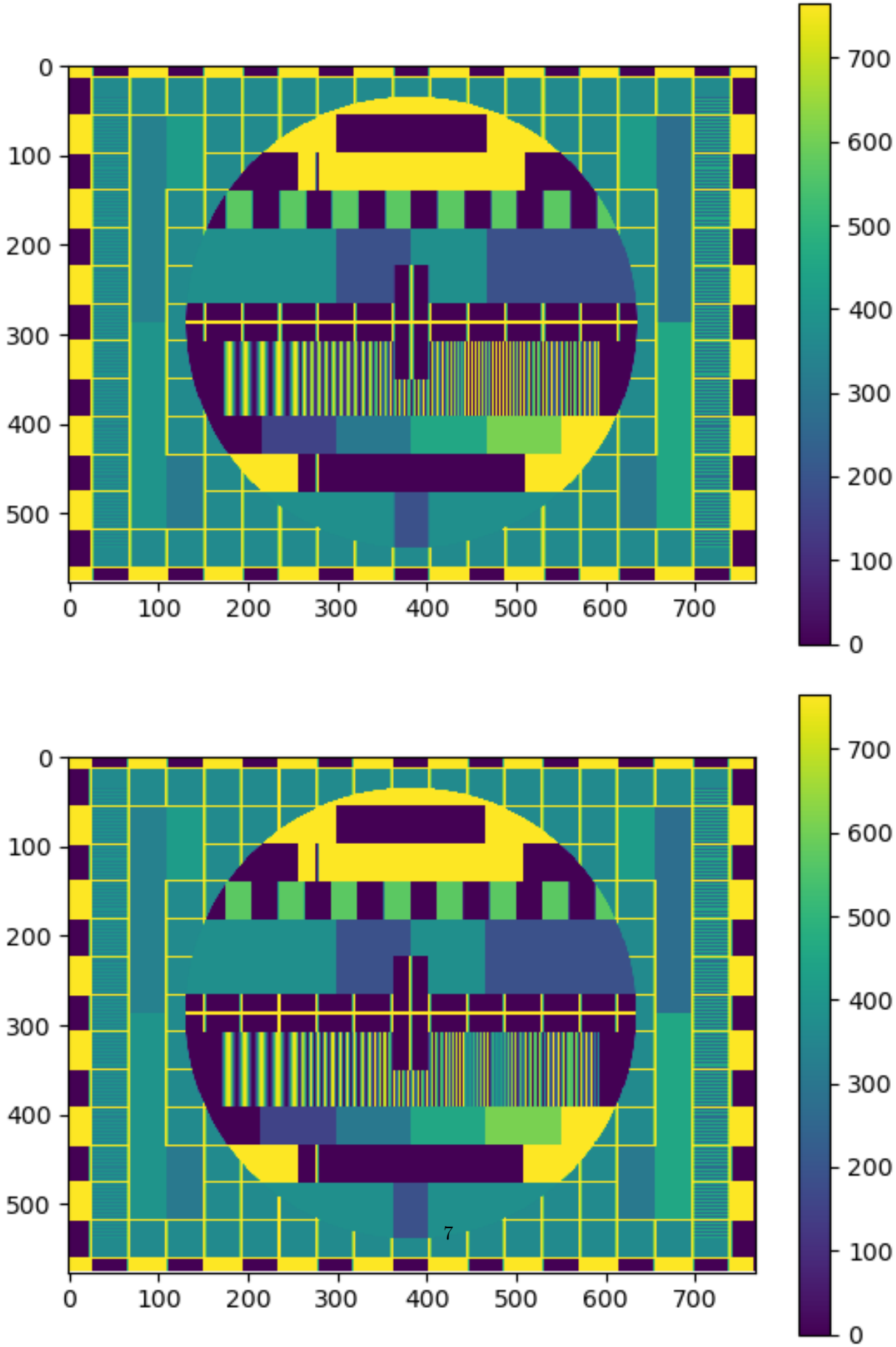


Figure 5: Correlation output plots using the filters in the `linear_filter.py` against the given image `test_card`