Image Processing: Homework 2

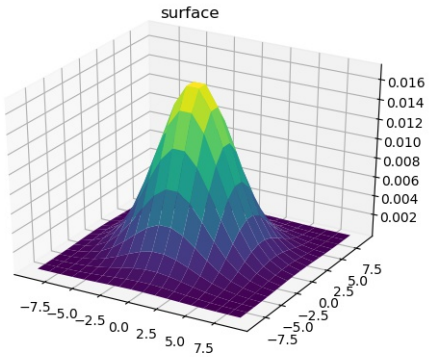
Due on November 30, 2019 at 10:00pm

Yamin Sarcheshmehpour
94109827

Problem 1

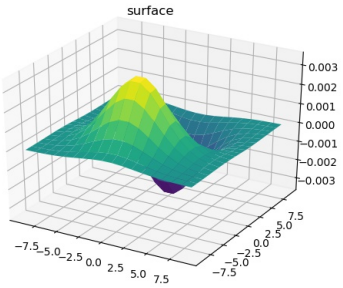
Solution

In this problem, I have implemented edge detection using gaussian filters. For doing this, I have considered 2 Gaussian derivative filters in the horizontal and vertical direction with a specific standard deviation (For example σ) with the window size of 6σ.
The selected σ is 3 thus the window size is $3 * 6 = 18$, the Gaussian filter with $\sigma = 3$ is as follows (the saved plot with the name of $Q1 - 3 - gaussian.jpg$ in the directory).

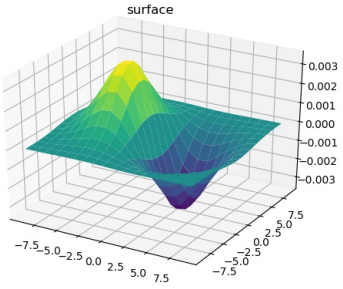


The gaussian vertical and horizontal filters are dx and dy.

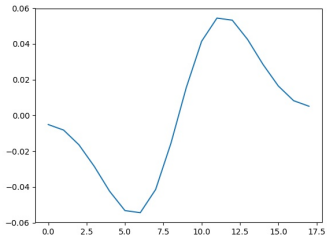
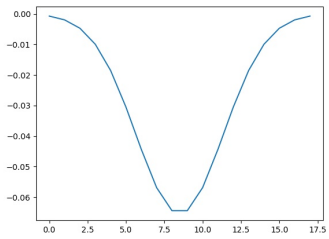
Gaussian vertical direction plot has saved with the name of $Q1 - dx.jpg$ in the directory.



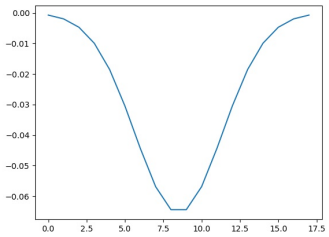
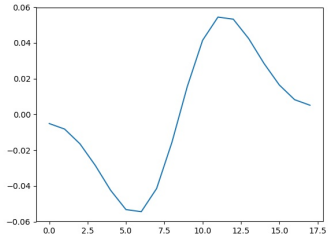
And its horizontal direction plot has saved with the name of $Q1 - dy.jpg$ in the directory.



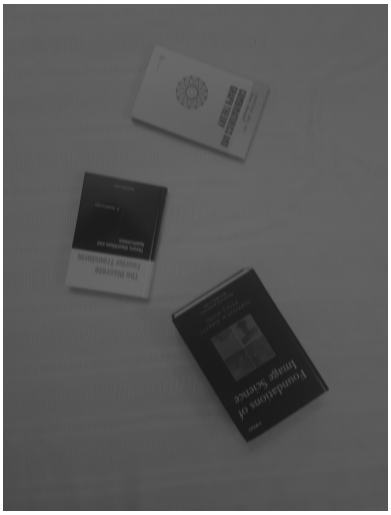
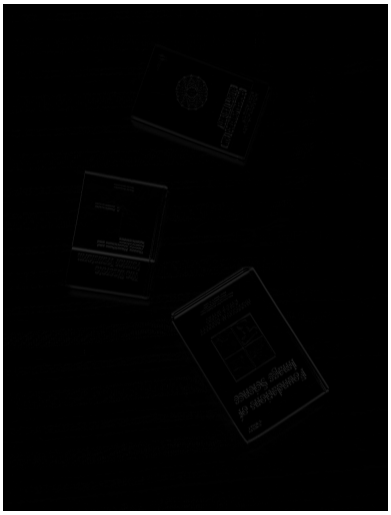
We know that the Gaussian derivative filter is separable.
The row and the column filters of vertical filter are ux and vx (which have concluded from SVD of dx) and have saved with the names of $Q1 - ux.jpg$ and $Q1 - vx.jpg$ respectively, in the directory.

Also, the row and the column filters of horizontal filter are uy and vy (which have concluded from SVD of dy) and have saved with the names of $Q1 - uy.jpg$ and $Q1 - vy.jpg$ respectively, in the directory.

By convolving uy and ux filters on the image, we would have horizontal row and vertical row values respectively, the absolute of results are saved as $Q1 - 01 - hor - row.jpg$ and $Q1 - 02 - ver - row.jpg$ in the directory.

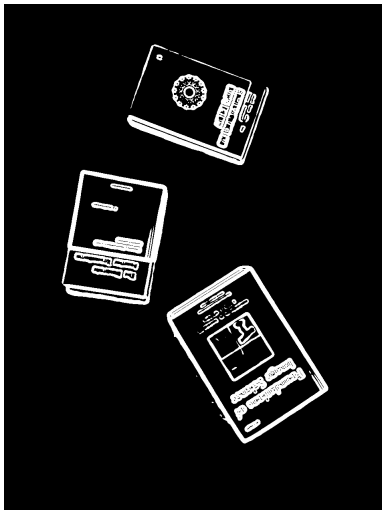
Also by convolving vy and vx filters on the image, we would have horizontal column and vertical column values respectively, the absolute of results are saved as $Q1 - 03 - hor - col.jpg$ and $Q1 - 04 - ver - col.jpg$ in the directory.

Convoling $hor - row$ and $hor - col$ results (assume that its name is $separated - value$) is the same as convolving dy (which is horizontal filer) and the image (assume that its name is $direct - value$).
To show this I have convoled $hor - row$ and $hor - col$ (As you know convolving $hor - row$ and $hor - col$ is the same as convolving uy and vy and the image) and calculated the norm of the diferences of $separated - value$ and $direct - value$, which is too small (about 10^{-12}). The original results are $Q1 - separated - hor.jpg$ and $Q1 - 05 - hor.jpg$ in the directory, but for making the results more obvious I have made them brightened. You can see the brightened results in the following.

It is the same for vertical direction too.

By convolving ver and hor images we could get into gradient values.
Pixels gradient values and their directions have saved with the names of $Q1 - 07 - grad - mag.jpg$ and $Q1 - 08 - grad - dir.jpg$, respectively.

By considering a threshold for gradient values we could find edges. To be general, I have taken the threshold as a coefficient of the mean of gradient values, the selected coefficient is 3.5, and the final result has saved with the name of $Q1 - 09 - edge.jpg$.



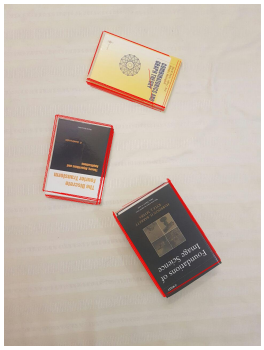
Describe functions

1. *separate_matrix*: This function has calculated the SVD of its input matrix.
2. *calculate_grad_mag*: This function has calculated the pixels gradient values by the image vertical and horizontal convolved values.
3. *calculate_grad_dir*: This function has calculated the pixels gradient directions.

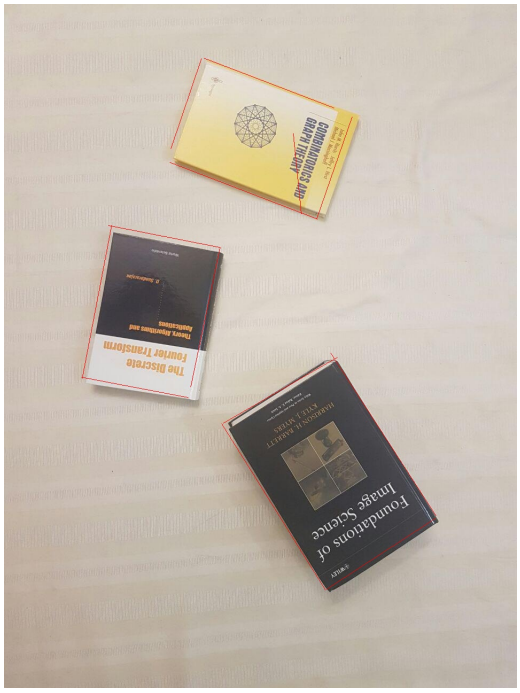
The code is in $Q1.py$, for runing the code please run *python* $Q1.py$.

Problem 2

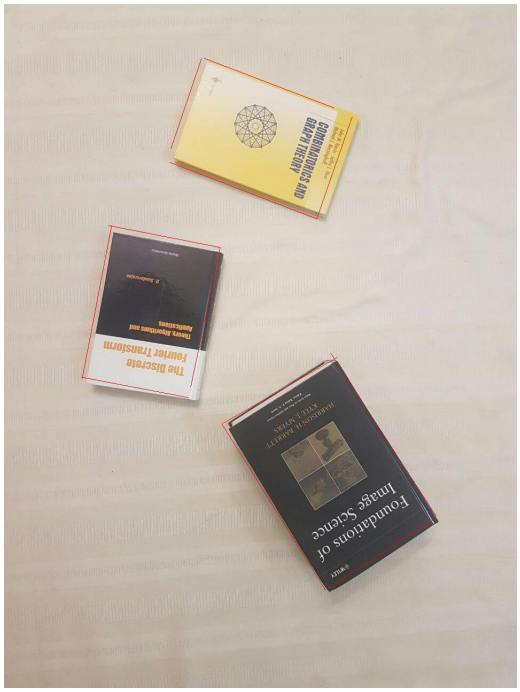
Solution
At this problem, I use the edges from the previous problem with the threshold of 3. Then I get Hough lines. Detected lines are like bellow.



Then I merge lines which are close to each other and calculate unique lines. As you see there is some outliers



By removing outliers we conclude to the following result.



Then for each pair of inlier lines, I will find the intersection of them, assume that lines are $l1$ and $l2$ and the intersection is p, then for each line, I find the nearest end of it to the intersection p and calculate the distance between p and *neatest* end, assume that the distance between p and *nearest* end for $l1$ is $d1$ and for $l2$ is $d2$, if both $d1$ and $d2$ are smaller than a threshhold, I match $l1$ and $l2$ as adjacent lines and create the graph of adjacent lines and their intersections.
Then by using DFS, I find line clusters (books) and print their intersections as corner points.
book 1 corners are : (809, 960), (618, 663), (407, 785), (601, 1096)

book 2 corners are : (146, 698), (200, 415), (403, 455), (344, 736)
book 3 corners are : (369, 99), (652, 207), (584, 401), (303, 293)

Describe functions

1. *_unique_lines*: It uniques lines with a given threshold for slopes and endpoint distances, if 2 lines slopes differences are smaller that slope threshold and their endpoints are near each other, it removes one of them.
2. *_remove_outliers*: Using the assumption that the edges of the book are parallel, I remove lines that don't have any parallel.
3. *_get_graph*: It finds the intersection of adjacent lines and creates their graph.
4. *_dfs_point*: By applying DFS on the graph, it finds books and their corners.

The code is in *Q2.py*, for running the code please first run *ipython* then run copy the *Q2.py* to *ipython* and then click enter.

Problem 3

Solution At this problem, we want to crop and rotate each book. Consider that for each book we have the coordinates for its corners and assume that the trasnformation matrix is Euclidean.
In Euclidean trasformation we have:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Thus if we have 3 point we could solve this. Assume that our points are $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ and we want to trasnform these point to $(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3)$.
So our equations would be as following:

$$x_1\cos(\theta) - y_1\sin(\theta) + t_x = x'_1, \quad x_1\sin(\theta) + y_1\cos(\theta) + t_y = y'_1$$
$$x_2\cos(\theta) - y_2\sin(\theta) + t_x = x'_2, \quad x_2\sin(\theta) + y_2\cos(\theta) + t_y = y'_2$$
$$x_3\cos(\theta) - y_3\sin(\theta) + t_x = x'_3, \quad x_3\sin(\theta) + y_3\cos(\theta) + t_y = y'_3$$

For each book, the rotated points would be (0,0), (0, book's height), (book's width, book's height), (book's width, 0).
The results have been saved with the names of *Q3_book1.jpg*, *Q3_book2.jpg* and *Q3_book3.jpg*.

Problem 3 continued on next page...

Describe functions

1. *_get_matrix_transfrom*: It solves 6 equaions to find trasformation matrix.

The code is in *Q*3.*py*.

Problem 4

Solution At this problem, I have implemented hybrid images.
Selected images are:

For doing this, I select 2 face images and choose 3 points in each one, coordinates of left eye center, coordinates of right eye center, and coordinates of nose center, and find the Affine matrix between them, and then wrap images to fit each other.
Fitted images are:

Then for both near and far images, I should get a gaussian filter with a proper σ, and apply a cutoff on them. I have choosen $\sigma = 10$ for the near image and $\sigma = 30$ for the other one, and for each filter the cutoff is $\frac{1}{2\pi\sigma}$.
The gaussian filter for the far image has saved as *Q*4_08_*lowpass*_30.*jpg* and its cutoff is *Q*4_10_*lowpass*_*cutoff*.*jpg* in the directory. For the near image, the used filter should be $1 - gaussian$, which is *Q*4_07_*highpass*_10.*jpg* and the cutoff is *Q*4_09_*highpass*_*cutoff*.*jpg*.
Then I take photos to the frequency domain and then shift the results (the shifted result for the near image is *Q*4_05_*dft*_*near*.*jpg* and for the far image is *Q*4_06_*dft*_*far*.*jpg*) Then I convolve shifted results with the

corresponding Gaussian filters (for the near image convolve result has saved as $Q4\_11\_highpassed.jpg$ and for the far image the result is $Q4\_12\_lowpassed.jpg$).
Here you can see the near and the far images after applying Gaussian filters to them.

For creating the hybrid result I combine the near and the far images in the frequency domain (the result has saved with the name of $Q4\_13\_hybrid\_frequency.jpg$ in the directory). Then I inverse the domain to reach the hybrid image (the image is $Q4\_14\_hybrid\_near.jpg$ in the directory).
The scaled hybrid image is as follows.

Describe functions

1. $\_get\_matrix\_transfrom$: It solves 6 equaions to find trasformation matrix.
2. $\_matche\_images$: It matches the images using their respective points.
3. $get\_gaussian\_filter$: It returns the Gaussian filter with a given sigma and sizes.
4. $\_apply\_cutoff$: It zeroes values of the filter which are smaller than the threshold.

5. *_apply_fourier*: It takes the photo to the frequency domain by applying FFT on it, and shifts the result, and then convolve it with the given Gaussian filter.

6. *_get_hybrid*: It calculates the hybrid image.

The code is in *Q4.py*.