M21CS7.404: Digital Image Processing
Monsoon-2021
Assignment-2
Posted on: 22/09/2021
Due on: 23:55hrs, DUE DATE: 22/10/2021

Introduction

1. All images in this assignment should be considered to be used as grayscale unless specified otherwise. Use the appropriate function to convert colour images to grayscale, or read the images directly as grayscale.
2. All functions in this assignment need to be implemented from scratch unless specified otherwise.
3. Follow the specified repository structure.
4. `src` will contain the Jupyter notebooks used for the assignment.
5. `images` will contain images used for the questions.
6. Make sure you run your Jupyter notebook before committing, to save all outputs.
7. Make sure you commit and push your work regularly.

Questions

1. (50 points) The implementation of linear spatial filters requires moving a mask centered at each pixel of the image, computing sum of products of mask coefficients and corresponding pixels. Padding can be used to perform filtering at the corners. Apply all the techniques to `ceramic.jpg`.

    1. Implement an algorithm for low-pass filtering a grayscale image by moving a $k \times k$ averaging filter of the form **ones((k,k))/(k^2).**
    Try with k = 3, 5, 7, 11 and 13.
    (Note: Ones((k,k))/(k^2) implies a filter with all the entries of a k*k matrix as $1/(k^2)$.
    2. As the filter is moved from one spatial location to the next one, the filter window shares many common pixels in adjacent neighborhoods. Exploit this observation and implement a more efficient version of averaging filter.
    3. To appreciate the benefits of the efficient version, generate a plot of k vs run-time for multiple images of different sizes (8x8, 16x16, 32x32, .. 512x512) by resizing `ceramic.jpg`. The plot diagram should contain a line plot for each image size. Use different marker types to distinguish the default implementation and improved implementation.
    Just to give you a rough idea, look at `https://imgur.com/a/0HtYlTE`.
    4. Utilize the observation similar to above to implement an efficient version of a $k \times k$ median filter. Generate a plot figure similar to previous question.

5. After observing the image `brainNoisy.jpg`, answer the following questions:
   - Which filter will be the best suited to remove the noise and why?
   - Experiment with different filter sizes and report your observations. Which filter size is optimal?
   - Get `brainNoisy.jpg` to resemble the original `brain.jpg` as much as possible by removing the noise.

2. (60 points) Edge Detection

   1. Read about Canny Edge detector. Apply the Canny edge detector (use inbuilt *cv2.Canny* function for this task) to `cup.jpg`.
   (You can read about the Canny Edge Detector here : Canny Edge Detector)
   2. Tweak the values of the arguments (*minVal*and *maxVal*), and report the values.
   that give the best results for each image. (Hint: Try to get a good outline of the cup without too many fine lines.)
   3. Consider **Roberts, Prewitt, Sobel** and **Laplacian** filters discussed in the class. Implement and apply these filters on `girl.jpg`. and make observations upon comparing their outputs. Compare these with the output of Canny edge detector on the same image.
   (You can read more about the Roberts filter here : Roberts Filter)

$$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Figure 1: Prewitt filter

$$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Figure 2: Sobel filter

$$M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$$

Figure 3: Robert filter

$$\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figure 4: Two forms of the laplacian filter

   4. What will the 5×5 variants of **Sobel** and **Prewitt** filters look like? Apply these larger filters on `girl.jpg` and make observations upon comparing their outputs with the corresponding smaller filters.

5. Zero-mean gaussian noise with $\sigma = 10$ and $\sigma = 30$ has been added to `girl.jpg` to obtain `girlNoisy1.jpg` and `girlNoisy2.jpg`, respectively.

   Study the effect of applying the above filters (from 2nd Question) on the noise-affected inputs.

3. (30 points) Gaussian Filtering and Bilateral Filtering

   1. Implement Gaussian filter and apply it to `mountain.png`. Vary σ value for filter size, 5x5. Find the optimal σ value for k=5 and compare the obtained outputs.

      Now vary the filter sizes (take optimal σ value for each filter size) and report your observations along with the images.

      Note: Optimal filter size for a given σ is the one that snugly fits 99% of the gaussian curve.

   2. Implement bilateral filtering and apply it to `mountain.png`.
   3. Vary the effect of domain and range components of the bilateral filter and report your observations along with the images.

4. (20 points) Rain removal

   In rainy days, the performance of outdoor vision systems will significantly degrade due to visibility obstruction, deformation, and blurring caused by raindrops. Therefore, it is highly desirable to remove raindrops from rainy images to ensure the reliability of outdoor vision systems.

Figure 1: Rain Removal

Note: This question has to be done on RGB images only. But you can work on each channel separately, and then combine them for final image.

1. Apply gaussian filter on `rain.png`. Report the σ value and filter size for the best output.
2. Apply median filter on `rain.png`. Report the filter size for the best output.

3. Apply bilateral filter on `rain.png`. Report the σ values(domain and range components) and filter size for the best output.
4. Compare the outputs from the above three filters and report your observations.

   Feel free to try out any other methods to achieve the intended result.

   Note: The best output is one where the rain drops are minimally visible without affecting the rest of the scene or introducing new artifacts into the scene.

5. (30 points) Cartoon Creation

   In order to achieve a basic cartoon effect, all you need is essentially - a bilateral filter and some edge detection. The bilateral filter will reduce the color palette,necessary for the cartoon look, and edge detection will allow production of bold silhouettes. To get cartoonized effects following steps are needed:

   1. Apply the bilateral filtering to the color image. Keep this filtered image aside for the last step.
   2. Now, take a copy of original color image, **convert it to gray scale**.
   3. Apply **blurring** on this gray scale image to reduce noise. [You can try different variants of blurring(box filter, gaussian filter, repeated gaussian filter,..) with different filter sizes and see the performance]
   4. Now, create an **edge mask** from this blurred, gray scale image. An edge mask can be obtained using adaptive thresholding.

      (Links for reading about adaptive thresholding : link1 and link2)

      [You can also try obtaining the edge mask by pixel wise division of the grayscale image by the *blurred image (of the edges)* to generate an outline similar to pencil sketches.]
   5. Combine the bilateral filtered image from step 1 with edge mask(by taking bitwise AND of both at each pixel location).

Figure 2: Cartoon effect

Now try to do these for images of your choice. Feel free to experiment with different filters and even methods to generate even better cartoons.

Note: This question has to be done on RGB images only.

6. (15 points) Unsharp Masking and Highboost filtering

   1. Implement Unsharp Masking and apply it to `orion.jpg`.
   2. Implement Highboost Filtering and apply it to `orion.jpg`. Increase the coefficient values for the edge mask and report your observations.

      (Take the same filter size as in unsharp Masking.)
   3. Choose any three different filter sizes. Compare the two methods for each filter size and report your observations along with the images.

7. (40 points) Fourier Transform

   1. Implement 2D DFT.
   2. Implement 1D FFT. Use it to implement 2D FFT and display the result on suitable images of your choice.

      Compare the runtimes of your version of DFT and FFT on different sized images (as in Q1) and plot them.
   3. Similarly, implement 2D inverse FFT.
   4. Calculate the Fourier transform of the Fourier transform of any image. How is it different from the original image? How can you fix this in the frequency domain so that we would get the original image back?

8. (25 points) Low Pass Filtering using Fourier Transfrom

   1. Apply ideal lowpass filter of cutoff radius, D = 30 in the frequency domain for image `mountain.png`. Also try for D = 15 and 50.

      Include the original image and all three ideal lowpass filtered images and briefly discuss your results.
   2. Apply gaussion lowpass filter in the frequency domain for image `mountain.png`. Try for D_0 values = 30, 15 and 50 (D_0 refers to the measure of spread in a gaussian curve).

      Include the original image and all three gaussian lowpass filtered images and briefly discuss your results.
   3. Discuss the difference between your results for gaussian lowpass filtering and ideal lowpass filtering.

9. (25 points) Morphological Operations

   Morphological gradient is one of the morphological operators which is used for generating the outline of the image. The gradient can be obtained using a combination of morphological operators like dilation, erosion, closing and opening.

Explain the combination of operations and choice of structuring element used while performing morphological gradient on the input image `eye.png`.

Refer: Morphological Gradient

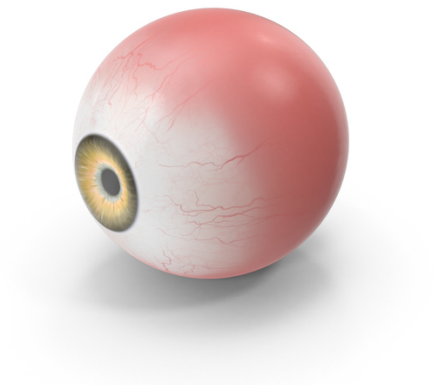

Figure 3: Eye

10. (30 points) Geometrical Transformations

    1. Write a function `rotImage` which rotates an input image about its center and takes the following parameters:
       - Input Image
       - Angle of rotation in degrees

       Note: The output dimensions should accommodate the entire image (without cropping) after rotation.
    2. Show outputs of a rectangular image of your choice with angle 30°, 45°, 90° and 2 other angles of your choice.

11. (15 points) Image Denoising

    Use a combination of any of the techniques in this assignment to bring `corruptedLena.jpg` as close to `lena.jpg` as possible. Explain your process.

(a) corruptedLena.jpg

(b) lena.jpg

Figure 4: Lena