Assignment 2
Digital Image Processing
Monsoon 2024

Deadline: 26th September 2024 11:59pm

General Instructions

1. Follow the specified repository structure.
2. Make sure you run your Jupyter notebook before submitting to save all outputs.
3. Appropriately label every figure and subplot.
4. Answer any questions asked in a markdown cell in your .ipynb file, as it carries marks.
5. Allowed libraries - numpy, matplotlib, opencv(only basic functions like imread, imwrite, resize, cvtColor)
6. Make an effort to vectorize your code as it contains 20% weight.

---

1 Let's Get Blurry[15]

(a) IMG1

1.1 Write a function `conv2D(img, kernel)` to convolve a 2D image with a 2D kernel of size `k x k` (Handle padding so that the resulting image is of the same size as input).

1.2 Apply a Mean Filter with default k=5 on `IMG1` by writing a new function `meanFilter(img, k)` that takes in, a 2D image, kernel size, generates the appropriate kernel and filters the image using `conv2D()` .

1.3 Similar to the above question, apply a Median Filter with default k=5 on `IMG1` by writing a new function `medianFilter(img, k)` that takes in, a 2D image, kernel size, generates the appropriate kernel and filters the image using `conv2D()` .

1.4 Make 2 plots for the time taken for the operations vs kernel size k(= any 3 values) for different image dimensions(=256x256, 512x512, 1024x1024) for `IMG1` for both filters `meanFilter()` and `medianFilter()`. Comment on your observations.

1.5 As the filter is moved from one location to the next, the filter window shares many common pixels in adjacent movements. On this note write a more efficient version, `fastMeanFilter(img, k)`. Make 2 plots comparing `meanFilter()` and `fastMeanFilter()` similar to plots in question 1.4. Comment on your observations.

2 Distribution is the Solution[15]

(a) IMG2

2.1 Apply Gaussian Filter with default k=7 on `IMG2` by writing a new function `gaussFilter(img, k, σ )` that takes in, a 2D image, kernel size, σ and generates the appropriate kernel and filters the image using `conv2D()`.

2.2 Display the results by trying few different values of σ (standard deviation) and find the optimal σ value for k=7, and show the plot which proves so. Hint: You can verify that most of the gaussian distribution's area (around 99%) is included within the kernel distribution.

2.3 Now vary kernel(k=any 3 values) sizes (keep σ as the above found optimal value) and display results. Comment on your observations of how the function is affected by changing values of k and σ.

2.4 Apply Bilateral Filter with default k=7 on `IMG2` by writing a new function `bilaterFilter(img, k, σs, σr )` that takes in, a 2D image, kernel size, σ_s (standard deviation for spatial gaussian) , σ_r (standard deviation for intensity range gaussian) and generates the appropriate kernel and filters the image using `conv2D()`.

2.5 Apply `bilaterFilter()` on `IMG2` with different values of k, σ_s and σ_r show how they affect the outcome by changing one parameter at a time. Comment on your observations.

2.6 Comment on the difference between the results of gaussian and bilateral filters.

3 Edge of Glory[10]

3.1 Apply a Sobel filter on `IMG3` by writing a function `sobelFilter(img)` that takes in a 2D image, applies the Sobel filter (for both horizontal and vertical edges), and combines the results to get the magnitude of the gradient.

3.2 Write a function `prewittFilter(img)` that applies the Prewitt filter to `IMG3`. The function should calculate the gradient in the horizontal and vertical directions using Prewitt kernels and combine them to get the edge magnitude.

(a) IMG3

3.3 Write a function `robertsFilter(img)` that applies the Roberts cross operator to `IMG3`. The function should compute the gradient magnitude using the Roberts kernels and return the edge map.

3.4 Make 3 subplots corresponding to the X gradient, Y gradient and the Edge Map for each of the filters.

3.5 Apply a Laplacian filter on `IMG3` by writing a function `laplacianFilter(img, k)` that takes in a 2D image, kernel size and applies the Laplacian filter. Vary the kernel(k=any 3 values) size and comment on the effects.

3.6 Analyze and comment on which types of edges are best identified by each of the filters.

4 Sharpen Up[10]

(a) IMG4

4.1 Write a function `unsharpMask(img, k, sigma)` that takes in a 2D image, a kernel size k, and a standard deviation σ, and applies the unsharp masking technique. Apply this function to `IMG4`.

4.2 Vary the kernel size k(for any 3 values) and the standard deviation σ in `unsharpMask(img, k, sigma)` and comment on how these parameters affect the sharpness of the image.

4.3 Write a function `highboostFilter(img, k, sigma, A)` that applies highboost filtering to `IMG4`. The function should take in a 2D image, a kernel size k, a standard deviation σ, and a boost factor A. Apply the function with different values of A(for any 3 values), comment on the role of A and compare the results.

4.4 Make 3 subplots showing the original image, the unsharp masked image, and the highboost filtered image. Which method provides better results for enhancing fine details?

5 Rain Rain Go Away![15]

(a) IMG5: Rainy Image

(b) Required Image

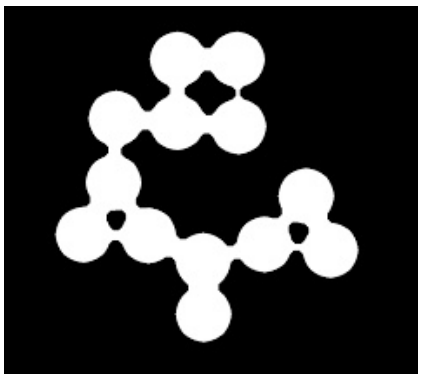
5.1 Here we are yet again where the rainy forecast is spoiling a IIIT event :(. Use your newfound powers of filtering and convolution to remove the rain in `IMG5` so that the event can happen :). Use your functions for mean, median, gaussian and bilateral filtering and report and display the best parameters for each of the 4 methods. Also, add your observations on the performance of each of the methods, which seems to be the best and why?

(a) IMG6: Mystery Noise Image

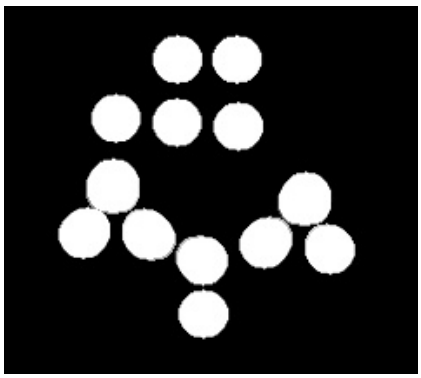
5.2 You are given `IMG6`, but it is not clear. What type of noise is there in `IMG6`? What is the best filtering method to remove it? Display your results when using this method to remove the noise in `IMG6`.

6 Transform and Conquer[15]

Counting Coins



(a) IMG7



(b) Target IMG7

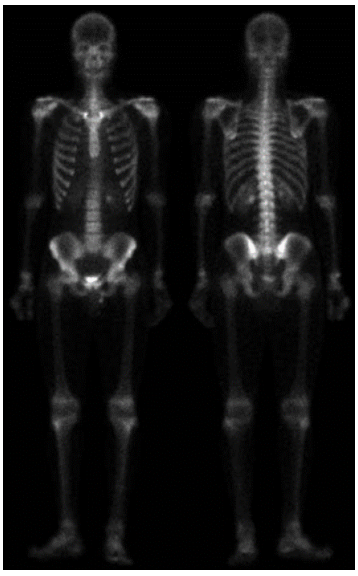
6a.1 Given an image, `IMG7`, of overlapping coins in a cash sorting system, develop an image processing pipeline to separate each coin for accurate classification. The resulting image should be as shown in `Target IMG7`. Use morphological operations you deem suitable (functions should be written on your own). Explain the steps taken. Hint: You can use different kernel shape to achieve a closer match.

Signature Verification

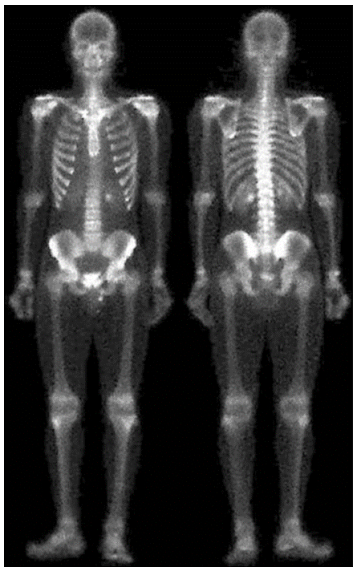
6b.1 Download any 2 authentic and any 2 fraud signatures corresponding to a certain person from CEDAR-Dataset(`https://www.kaggle.com/datasets/shreelakshmigp/cedardataset`).

6b.2 Find skeleton of each of the signatures, using your own function, and comment on the stability of this feature across various signatures of a particular person. Hint: Mirror the process of skeletonize (`https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html`) in your function.

7 Body Scan Enhancement[10]



(a) IMG8



(b) Enhanced IMG8

7.1 `IMG8` is a nuclear whole body bone scan, used to detect diseases such as bone infection and tumors. Your objective is to enhance this image by sharpening it and by bringing out more of the skeletal detail as can be seen in `Enhanced IMG8`. Use functions written so far (and more) to achieve the above, elaborate on the steps involved.

8 You Can't See Me[10]

(a) IMG9

(b) Clean IMG9

8.1 You have been given an image `IMG9` which is corrupted. Apply various methods learned so far to obtain the clean version (as close as you can get!). Explain your process and reasoning.