EE604: Assignment 4

{sandhan, chiranjv, dbharti}@iitk.ac.in

Semester-I, 2021

Due date: 15^{th} November, 2021 Weight: 33%
Due time: 23:59PM Submission: MookIT

Introduction

In this work you are handling a core part of the image processing i.e. image filtering and denoising. You will also create graphical or comic images as another application of filtering, and make use of the image compression and image quality assessment concepts. Below references are useful for this work.

References

- [1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images", IEEE International conference on computer vision (ICCV), 1998.
- [2] Q. Yang, K. Tan and N. Ahuja, "Real-time $O(1)$ bilateral filtering", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.
- [3] H. Winnemoller, SC. Olsen and B. Gooch, "Real-Time Video Abstraction", ACM Transactions on Graphics, 2006.

Questions

Questions are indicated as $[Q_i](w)$, where $Q \in \{P$:program, I:image, R:report$\}$, i is the question number having w% weight for evaluation. You have to submit separate python program files $[P_i]$, images $[I_i]$ for each i, whereas submit the combined single pdf report $[R_i]$ $\forall$ i. You have been offered 2 images: `{rome, iitk}.jpg`, which are adulterated by environmental noise of burning flakes and camera system noise due to heated electronic circuitry. Though the noise is same, underlying image content and resolutions are distinct; so your algorithm might need to use different parameters for filtering each of those images.

Table 1: Algorithm performance analysis

'rome.jpg' (and do similar separate table for iitk.jpg)			
Algorithm	**Speed (sec.)**	**PSNR**	**SSIM**
anyf			
mybf (original)		N/A	
mybf (your speed improved)			
rtbf (para-1: good accuracy)			
rtbf (para-2: good speed)			
rtbf (para-3: best para)			
rtbf (best para) (JPEG 99)			
rtbf (best para) (JPEG 95)			
rtbf (best para) (JPEG 90)			

1 Any Filter: `anyf`

Use any image filter or image enhancement technique (other than bilateral filter) and denoise the `rome` and `iitk`. You may reuse your previous codes or directly use library functions for it.

- $[R_1](2) \rightarrow$ Write all details about the filter you have used e.g. its parameters. Paste the input and output images side-by-side in the report (small resized images are also alright). Write your observations about its denoising performance.

2 Bilateral filter: `mybf`

Implement the bilateral filter that we have learned in the class or you can refer [1]. You can not use direct library functions for bilateral filter.

- $[P_1](6) \rightarrow$ Let's call original implementation of bilateral filter as `mybf_original.py`. Optimize it for the highest accuracy i.e. the best denoising performance for each image and obtain the cleanest possible images and save them in raw format i.e. without compression `{rome_clean, iitk_clean}.raw`. These are your reference images for comparing PSNR and SSIM for all algorithms. Now improve the speed of `mybf_original.py` by whatever ideas you can think of and write a function `'def my_BF(img):'` in the file `mybf.py`.
- $[R_2](1) \rightarrow$ Report all the improvements or ideas you have used for speeding up original bilateral filter to obtain `mybf.py`.

3 Real-time bilateral filter: `rtbf`

- $[P_2](10) \rightarrow$ Analyze the work by [2] and implement real-time bilateral filter function `'def rt_BF(img):'` in the file `rtbf.py`.

4 Analysis

You can use direct library functions for JPEG, PSNR and SSIM.

- $[R_3](5) \rightarrow$ Adjust the parameters of `rtbf.py` for good accuracy (para-1), for good speed (para-2) and optimal parameters for high speed as well as maintaining acceptable denoising performance (para-3). Set the `rtbf.py` with para-3 and save the output denoised images using JPEG compression with 99, 95 and 90% quality levels respectively. Read back these jpg images and compare them with raw clean reference images for PSNR and SSIM values, also note down the algorithm speed for this whole process (filter + compress + readback). Note the compressor indirectly acts as an extra denoiser. Compare the denoised images obtained via each of the algorithm above to the reference images and fill the Table-1 for `rome` and `iitk` separately, and write down your analysis, observations.

5 Application

You may refer class lecture or [3] and use `rtbf.py` for cartoonizing the image.

- $[I_1](0.5) \rightarrow$ Select any image from your image gallery (e.g. monuments, travel, nature, animals etc.) which is worth of cartoonization and name it `your.jpg`.
- $[I_2](1.5) \rightarrow$ Cartoonize `your.jpg` to generate `cartoon.jpg`.
- $[R_4](1.0) \rightarrow$ Report the parameters and algorithm you have used for $[I_2]$.

6 Segmentation: `seg`

IITK is planning to use drones for spraying insecticide over vegetation. Drones should spray insecticide accurately over desired area. Use the algorithm of your choice to help these drones on `iitk.jpg` i.e. algorithm should output binary intensity image `fordrone.jpg` (same height x width size as input, stored as gray scale) containing pixel values 255 corresponding to spray-able area, also called as region of interest (ROI) and rest of the pixels should set to 0. You may use any library functions. You may have to first cleanup `iitk` before finding ROI for better accuracy.

- $[P_3](3.5) \rightarrow$ Write your function `'def my_SEG(img):'` in the file `seg.py`.
- $[R_5](2.0) \rightarrow$ Report all things about the algorithm, intermediate images and your analysis.
- $[I_3](0.5) \rightarrow$ Submit the final image which drone should see: `fordrone.jpg`.

The $[P_1]$, $[P_2]$ and $[P_3]$ should work as below: (Note- there is no extra input parameter to function apart from `img`, so algorithm should automatically adjust the best para depending upon whether it is at `rome` or `iitk`)

```
:  img = cv2.imread("iitk.jpg")
:  img_mybf = my_BF(img)
:  img_rtbf = rt_BF(img)
:  img_seg  = my_SEG(img)
```

Submission

Zip only the below files to a single file `rollno_A4.zip` (eg. `181234_A4.zip`) and upload it to mooKIT before the submission deadline. Evaluation criterion: non-similarity with other's work, speed, SSIM, PSNR (recalculated on our end), and quality of the report and images.

- `report.pdf`
- `mybf.py`
- `rtbf.py`
- `seg.py`
- `your.jpg`
- `cartoon.jpg`
- `fordrone.jpg`

■