SCC5830— Prof. Moacir Ponti

PAE: Leo Sampaio Ferraz Ribeiro

Assignment 2: enhancement and superresolution

Code the assignment by yourself. Ask if you need help. Plagiarism is not tolerated.

1 Introduction

1.1 Goal

To have the students implementing their first pixel and histogram-based enhancement methods as well as present the concept of superresolution for the first time.

1.2 Task

In this assignment you have to implement 3 distinct image enhancement techniques, as well as a superresolution method based on multiple views of the same image. Students are required to use `python 3` and the libraries `numpy` and `imageio` to complete the task.

Follow the instructions carefully:

1. Find and load all low resolution images $l_i \in L$ that match the basename `imglow` (i.e. filenames that start with `imglow`)
2. Apply the selected enhancement method F to all low resolution images, using parameter γ when appropriate
3. Combine the low resolution images into a high resolution version $\hat{H}$
4. Compare $\hat{H}$ against reference image H using Root Mean Squared Error (RMSE)

1.3 Input Parameters

The following parameters will be input to your program in the following order through `stdin`, as usual for run.codes:

1. basename `imglow` for low resolution images $l_i \in L$. The basename references the start of the filenames for 4 low resolution images l_1, l_2, l_3, l_4.[1]
2. filename `imghigh` for the high resolution image H
3. enhancement method identifier F (0, 1, 2 or 3)
4. enhancement method parameter γ for $F = 3$

[1] They are all `.png` files and follow the pattern `[imglow]1.png`, `[imglow]2.png`, `[imglow]3.png` e `[imglow]4.png`

2 Image Enhancement

There are three options for Image Enhancement, with Option 0 indicating that no enhancement is to be done:

Option 0: No Enhancement : Do not apply any enhancement technique to the image and instead skip to the superresolution step.

Options 1 and 2 are histogram-based methods while Option 3 uses pixel-based Gamma correction.

2.1 Histogram-based Enhancement

You should implement two methods of Histogram Equalization and apply them to all low resolution images L. For Option 1 you should use the cumulative histogram of each image as the transform function for your image, as presented in class. For Option 2 however you should compute the cumulative histogram based on all images in the L set *together* (as if they were a single image), and then use it as the transform function.

Option 1: Single-image Cumulative Histogram : Compute the Cumulative Histogram $hc(l_i)$ for each image $l_i \in L$ and use it as a transform function to equalize the histogram of each image

Option 2: Joint Cumulative Histogram : Compute a single Cumulative Histogram $hc(L)$ over all images in L and use it as a transform function to equalize each image

2.2 Gamma Correction

Option 3: Gamma Correction Function : Implement the pixel-wise enhancement function called Gamma Correction, using the following:

$$\hat{L}_i(x, y) = \left\lfloor 255 \cdot \left((L_i(x, y)/255.0)^{1/\gamma} \right) \right\rfloor ,$$

where $\hat{L}_i$ is the resulting image and γ is a parameter input by the user as described in Sec 1.3.

3 Superresolution

Let's assume that each of the low resolution images L is a different "view" of the exact same scene. We can use those images (post enhancement) to compose a higher resolution version $\hat{H}$ (to simplify our task, this higher resolution will always be double the original). We propose a very simple composition method, as in the example below:

$$l_1 = \begin{bmatrix} 100 & 101 \\ 110 & 111 \end{bmatrix}, l_2 = \begin{bmatrix} 200 & 201 \\ 210 & 211 \end{bmatrix}, l_3 = \begin{bmatrix} 300 & 301 \\ 310 & 311 \end{bmatrix}, l_4 = \begin{bmatrix} 400 & 401 \\ 410 & 411 \end{bmatrix}$$

$$\hat{H} = \begin{bmatrix} 100 & 200 & 101 & 201 \\ 300 & 400 & 301 & 401 \\ 110 & 210 & 111 & 211 \\ 310 & 410 & 311 & 411 \end{bmatrix}$$

Even though it is simple, this method can yield impressive results on some images. You can assume the resolution of reference image H (and your version $\hat{H}$) will be double the resolution of the images L and that images L will always share the same resolution.

4 Comparing against reference

Your program must compare your enhanced image $\hat{H}$ against reference image H. This comparison must use the root mean squared error (RMSE). Print this error in the screen, rounding to 4 decimal places.

$$\mathcal{L}_{RMSE}(H, \hat{H}) = \sqrt{\frac{\sum_i \sum_j (H(i,j) - \hat{H}(i,j))^2}{N \cdot N}}$$

where $N \times N$ is the resolution of images H and $\hat{H}$.

Better Superresolution Methods

Your evaluation will take into consideration the results expected by the superresolution method suggested for this assignment (in Sec. 3); this method is however very simple, so you can, to learn more and for fun, look for and implement better post-processing or pixel composition methods that yield better RSME results. If it decreases the error more than the suggested method, you are good to go!

5 Input and Output

Input Example 01: Low resolution images L `boat1.png`, `boat2.png`, `boat3.png`, `boat4.png`; High resolution reference image H `boathigh.png`; Enhancement method $F = 2$ (joint cumulative histogram); Parameter γ is ignored and can be anything:

```
boat
boathigh
2
1
```

Output Example 01: Just the RSME result with 4 decimal points:

```
10.1864
```

6 Submission

Submit your source code to run.codes (only the `.py` file).

1. **Comment your code**. Use a header with name, USP number, course code, year/semestre and the title of the assignment. A penalty on the evaluation will be applied if your code is missing the header and comments.

2. **Organize your code in programming functions**. Use one function for each enhancement method and a separate function for your superresolution method.