DIPCV Assignment 1: Reading and Point Operation

|  |
| --- |
| Eric Cheng |
| *Department of Earth Science* |
| *National Cheng Kung University* |
| *Anping, Tainan 70847, Taiwan* |
| l46104020@gs.ncku.edu.tw |

***Abstract* -This report shows (1) principals when reading images if the third-party package cannot be used, (2) what point-operation methods are and how to implement them and (3) what image scaling is and how to implement them.**

***Index Terms -Reading, Point Operation, Image Scaling***

I. Introduction

We are offered 6 image files in this assignment, as shown in **Fig 1.** 3 of them can be rendered automatically by Microsoft Word, and 3 of them cannot. Our goals are (1) to successfully read the images (into the computer memory) and plot them out, (2) to implement point-operation methods on them, and (3) to apply image scaling on them. The source code can be found at [Ratherman/Computer-Vision (github.com)](https://github.com/Ratherman/Computer-Vision)[1]. This assignment has already been put in the following path, HW1\_Python/ and HW1\_C++/. These two different folders contain codes based on different programming languages. My goal are to use Python to write everything from scratch. As for C++, I might use 3-party package.

This report is organized as follows: in section II, we will have a look of reading image; in section III, the principals of point-operation methods would be discussed; in section IV, we will go through interpolation methods; and we will state the conclusion in section V. Moreover, in the end of this report, all the generated images are arranged in **Table 1.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

Fig 1. The 6 files we have.

II. Read Image

*A. For pictures which “can” be rendered*

These pictures belong to “\*bmp” file. By using “imread()” method from OpenCV, we can directly read this kind of image.

*B. For pictures which “cannot” be rendered*

These pictures end with \*.raw\*. Based on the document provided, we know that the pixel values were arrange in the sense of “row-major order”, as shown in **Fig 2.**

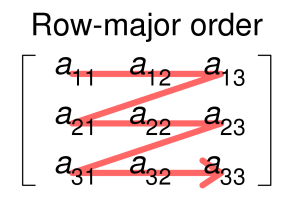
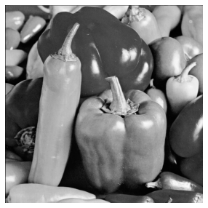


Fig 2. Row-major order

For coding principal, in this scenario, we will need a nested loop for traverse this image. The outer one represented the “height position (Y)” of the image, and the inner one represented the “width position (X)” of the image. When holding Y fixed, and iterate X, we will form “Row-major order”.

In **Fig 3.** shows the pictures (and their corresponding central area) that cannot be rendered by Microsoft Word, they are goldhill, peppers, and lena.



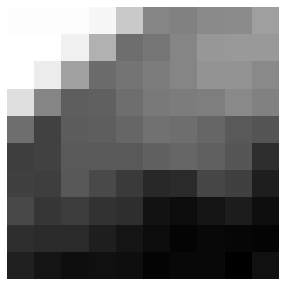
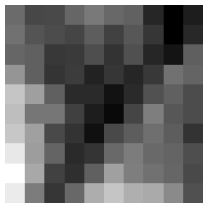
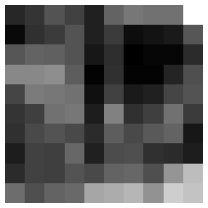


Fig 3. Reading results.

From the left are goldhill, peppers, and lena.

III. Point Operation Methods

In the world of digital image processing, there are at least two kinds of operation that we usually implement. One is using “kernels” to extract information from the given image, usually, the input for a kernel is always a “region” of an image (not just single point). The other one “point operation method” is which we will discussed in more detail in this report.

The methods we are going to discussed is used for “enhance” the original image, including image negative, log transform, and gamma transform. When finding the right function to enhance the image, human can more easily get information from the images.

We know that the range of pixel value is ranged from 0 to 255. Based on this prior knowledge, we can go forward and see the algorithm of each method.

1. *Image negative*

Its formula can be written as follows:

,

where **r** is input value and **y** is output value.

In **Fig 4.** shows the result of the image-negative version of baboon, F16, and boat (and their corresponding central area as well). For the comparing purpose, I will also list out the original image.

一張含有 文字, 靈長類動物, 哺乳類, 猴子 的圖片

自動產生的描述一張含有 文字, 飛機, 室外, 運輸 的圖片

自動產生的描述一張含有 文字, 船, 室外, 天空 的圖片

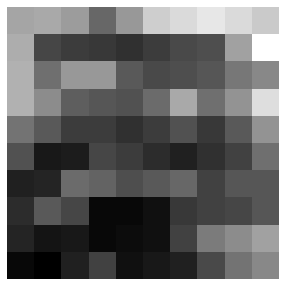
自動產生的描述

一張含有 文字, 凝視 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述一張含有 文字, 船, 室外, 水 的圖片

自動產生的描述

一張含有 廣場 的圖片

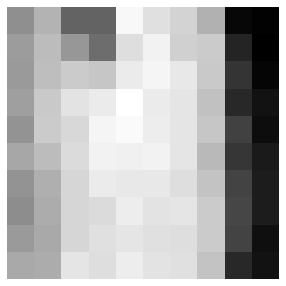
自動產生的描述

Fig 4. Image Negative

The first row shows the original version of image; the second row shows the image-negative one; and the third row shows central area of generated image. From the left are baboon, F16, and boat.

*B. log transform*

In this part, I followed the instructions from [this website](https://theailearner.com/2019/01/01/log-transformation/) [2] and implemented the idea of log transform. Its formula can be written as follows:

,

where **r** is input value, **y** is output value, and **c** is the constant defined as follows:

c = 255/(log(1+max)), where max is 255.

In **Fig 5.** shows the result of the log-transform version of generated image. Alike above, I will list out its corresponding original and central ones.

一張含有 文字, 室外 的圖片

自動產生的描述一張含有 室內, 蔬菜, 排列 的圖片

自動產生的描述一張含有 文字, 個人, 服飾, 髮 的圖片

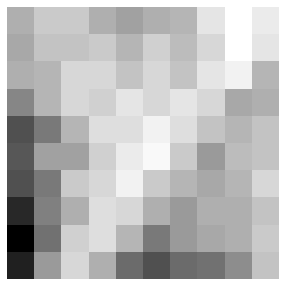
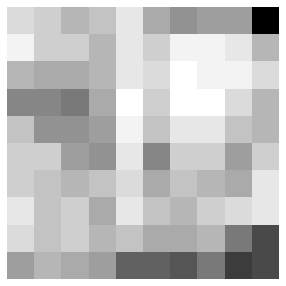
自動產生的描述

一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述一張含有 文字, 室內 的圖片

自動產生的描述

一張含有 廣場 的圖片

自動產生的描述

Fig 5. Log Transform

The first row shows the original version of image; the second row shows the log-transform one; and the third row shows central area of generated image. From the left are goldhill, peppers, and lena.

1. *gamma transform*

In this part, I followed the instruction from [this website](https://theailearner.com/2019/01/26/power-law-gamma-transformations/)

[3] and implement the idea of gamma transform. It’s also known as power law transformation. Its formula can be written as follows:

where **r** is the input value, **y** is the output value, **c** is set to 1 in this case, and **gamma** is the hyper parameter that we can adjust.

In **Fig 6.** show the results of gamma-transform version where gamma is set to 0.4. As expected, when gamma is smaller than 1.0, the image becomes whiter.

一張含有 文字, 靈長類動物, 哺乳類, 猴子 的圖片

自動產生的描述一張含有 文字, 飛機, 室外, 運輸 的圖片

自動產生的描述一張含有 文字, 船, 室外, 天空 的圖片

自動產生的描述

一張含有 文字, 凝視 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述一張含有 文字, 船, 室外, 天空 的圖片

自動產生的描述

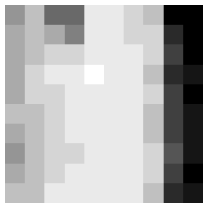
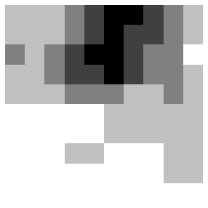
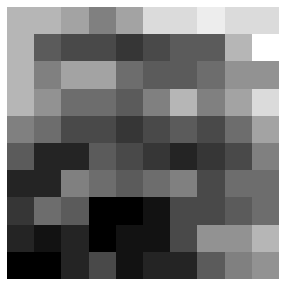


Fig 6. Gamma Transform (Gamma=0.4)

The first row shows the original version of image; the second row shows the gamma-transform one; and the third row shows central area of generated image. From the left are baboon, F16, and boat.

In **Fig 7.** show the results of gamma-transform version where gamma is set to 20. As expected, when gamma is larger than 1.0, the image becomes darker.

一張含有 室內, 蔬菜, 排列 的圖片

自動產生的描述一張含有 文字, 室外 的圖片

自動產生的描述一張含有 文字, 個人, 服飾, 髮 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述一張含有 文字, 個人 的圖片

自動產生的描述

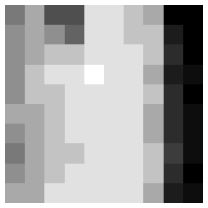
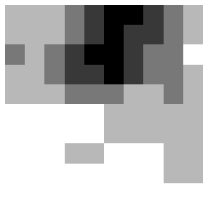
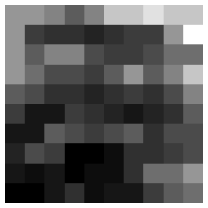


Fig 7. Gamma Transform (Gamma=20)

The first row shows the original version of image; the second row shows the gamma-transform one; and the third row shows central area of generated image. From the left are goldhill, peppers, and lena.

IV. Interpolation Methods

In this section, we will implement two interpolation methods: (1) Nearest Neighbor and (2) Bilinear. Both are used when we need to upscale or downscale the image for other usages. Note that, to the best of my knowledge, when we want to upscale an image, there are some deep learning methods that we could apply to generate more high-quality images, which are called to super resolution methods. But, here, we only discuss the two methods mentioned above.

Besides using two interpolation methods, in this report, 5 different scaling process will be implemented: (1) From 512x512 to 128x128, (2) From 512x512 to 32x32, (3) From 512x512 to 1024x512, (4) From 128x128 to 256x512, and (5) From 32x32 to 512x512.

*A. Nearest Neighbor*

Basically, when using this method, the interpolated pixel will reference the value of the nearest pixel in original image. The hardest idea for me is to come up with the idea about “factor” between the original size and the target size. So long as we have this “factor”, it’s easy to map the position of original image to the target position, and the algorithm could be done.

*B. Bilinear*

Different from nearest neighbor, bilinear interpolation uses values from 4 surrounding points to decide the value of the target point. When implementin, firstly, the horizontal interpolation will be implemented twice, then the vertical interpolation will be implemented once. Finally, we will get the target value.

I will apply these 5 different scaling processes on 5 different images (baboon, F16, boat, peppers, goldhill), and each process would be implemented by using two interpolation methods. For comparison purpose, I will also show the original one as well. Please check **Fig 8.** to **Fig 17.**

一張含有 文字, 靈長類動物, 哺乳類, 猴子 的圖片

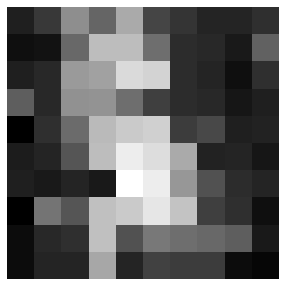
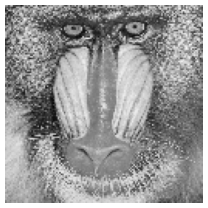
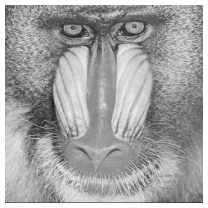
自動產生的描述

Fig 8. From 512x512 to 128x128, Nearest Neighbor, baboon

一張含有 文字, 靈長類動物, 黑色, 舊 的圖片

自動產生的描述一張含有 廣場 的圖片

自動產生的描述

Fig 9. From 512x512 to 128x128, Bilinear, baboon

一張含有 文字, 飛機, 室外, 運輸 的圖片

自動產生的描述一張含有 文字, 填字遊戲, 黑色, 白色 的圖片

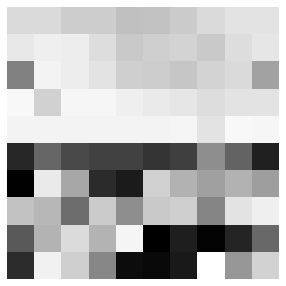
自動產生的描述

Fig 10. From 512x512 to 32x32, Nearest Neighbor, F16

一張含有 文字, 飛機, 室外, 運輸 的圖片

自動產生的描述一張含有 文字, 填字遊戲, 黑色, 並排的 的圖片

自動產生的描述一張含有 廣場 的圖片

自動產生的描述

Fig 11. From 512x512 to 32x32, Bilinear, F16

一張含有 文字, 船, 室外, 天空 的圖片

自動產生的描述一張含有 文字, 船, 室外, 多雲 的圖片

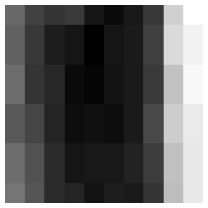
自動產生的描述

Fig 12. From 512x512 to 1024x512, Nearest Neighbor, boat

一張含有 文字, 船, 室外, 天空 的圖片

自動產生的描述一張含有 文字, 船, 室外, 多雲 的圖片

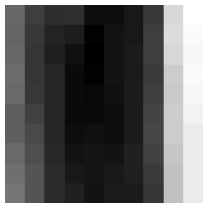
自動產生的描述

Fig 13. From 512x512 to 1024x512, Bilinear, boat

一張含有 室內, 蔬菜, 排列 的圖片

自動產生的描述一張含有 差異, 排列 的圖片

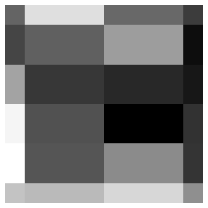
自動產生的描述

Fig 14. From 128x128 to 256x512, Nearest Neighbor, peppers

一張含有 室內, 蔬菜, 排列 的圖片

自動產生的描述一張含有 室內 的圖片

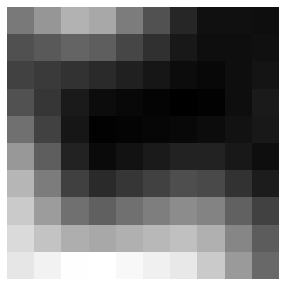
自動產生的描述

Fig 15. From 128x128 to 256x512, Bilinear, peppers

一張含有 文字, 室外 的圖片

自動產生的描述一張含有 文字, 白色, 黑色, 並排的 的圖片

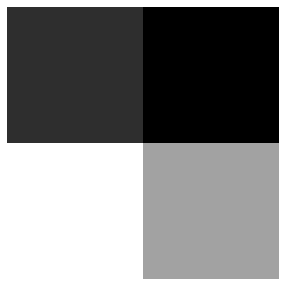
自動產生的描述

Fig 16. From 128x128 to 256x512, Nearest Neighbor, goldhill

一張含有 文字 的圖片

自動產生的描述一張含有 軟體動物, 無脊椎動物 的圖片

自動產生的描述

Fig 17. From 128x128 to 256x512, Bilinear, goldhill

Here, I found an interesting fact that when implementing bilinear interpolation. Because the target value is calculated based on 4 different original pixel values, the generated image will be smoother than the one generated by using nearest neighbor method. Especially, in **Fig 16.** and **Fig 17.**, the top-right corner of the cropped area is darker, in the scenario of bilinear interpolation, the overall value changes **gradually**, and in the scenario of nearest neighbor, the value changes **sharply**.

The other idea I’d like to discuss is the “factor”. When horizontal factor and vertical factor are different, the generated image will become a rectangle, like **Fig 12.** and **Fig 14.** When observing the central area of generated image, it’s clearly that the original images were stretched (in which direction, … etc.) compared to the ones generated by using bilinear interpolation.

IV. Conclusion

In this report, we go through a complete process of digital image process where we implement image reading, point-operation methods, and even image up-/down-scaling.

With the central area of each generated image, I’ve gained more intuitions for the algorithms we’ve implemented. Take interpolation method for an example, bilinear interpolation is obviously having stronger effect on “smoothing” the images than nearest neighbor. Also, after implementing nearest neighbor on images, if the scaling factors of width and height are different, the central area will present this fact clearly, i.e., we could easily understand how were generated images stretched.

Acknowledgment

This is the first assignment of DIPCV, good luck to myself in the future😊.

References

1. Source code: [Ratherman/Computer-Vision (github.com)](https://github.com/Ratherman/Computer-Vision)
2. Log transform: [Log Transformation | TheAILearner](https://theailearner.com/2019/01/01/log-transformation/)
3. Gamma transform: [Power Law (Gamma) Transformations | TheAILearner](https://theailearner.com/2019/01/26/power-law-gamma-transformations/)

Table. 1

Use this table to organize all generated images. used methods rae (1) original, (2) image negative, (3) log transform, (4) gamma transform, where gamma is 0.4, (5) gamma transform, where gamma is 20, (6) ~ (15) interpolation method whose algorithms including nearest neighbor, NN, and bilinear interpolation, and scales of generated images including128x128, 32x32, 1024x512, 256x512, and 512x512.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | baboon | F16 | boat | peppers | goldhill | lena |
| 1. Original |  |  |  |  |  |  |
| 1. Image negative |  |  |  |  |  |  |
| 1. Log transform |  |  |  |  |  |  |
| 1. Gamma transform (gamma=0.4) |  |  |  |  |  |  |
| 1. Gamma transform (gamma=20) |  |  |  |  |  |  |
| 1. NN 512x512 🡺 128x128 |  |  |  |  |  |  |
| 1. Bilinear 512x512 🡺 128x128 |  |  |  |  |  |  |
| 1. NN 512x512 🡺 32x32 |  |  |  |  |  |  |
| 1. Bilinear 512x512 🡺 32x32 |  |  |  |  |  |  |
| (10) NN 512x512 🡺 1024x512 |  |  |  |  |  |  |
| (11) Bilinear 512x512 🡺 1024x512 |  |  |  |  |  |  |
| (12) NN 128x128 🡺 256x512 |  |  |  |  |  |  |
| (13) Bilinear 128x128 🡺 256x512 |  |  |  |  |  |  |
| (14) NN 32x32 🡺 512x512 |  |  |  |  |  |  |
| (15) Bilinear 32x32 🡺 512x512 |  |  |  |  |  |  |