王　兆東

マルチメディア信号解析

2020年4月16日

Report of First Assignment

In this report I will discuss how I design my program to complete the assigned task, it include how to read an image by program and several image transferring operation like enlargement, reduction and sampling.

1. Environment Introduction

· Programming language: Python3.6

· library would be used: OpenCV [1]

· text book：jupyter [2]

2. Implementation

First, use the OpenCV library to decode a JPEG image file, to transfer the pixels into a BGR matrix, as we can see the shape of the matrix: 850x651 shows the width and length of the image.

```
In [2]: import cv2

In [9]: img = cv2.imread('IMG_0122.jpeg',1)# read a image  1: name 2 : 0 gray 1 color

In [11]: (b,g,r) = img[100,100]

In [12]: print(b, g, r)
         245 242 244

In [13]: img
Out[13]: array([[[240, 244, 245],
                 [240, 244, 245],
                 [240, 244, 245],
                 ...,
                 [246, 246, 246],
                 [246, 246, 246],
                 [246, 246, 246]],

                [[240, 244, 245],
                 [240, 244, 245],
                 [240, 244, 245],
                 ...,
                 [246, 246, 246],
                 [246, 246, 246],
                 [246, 246, 246]],

                [[240, 244, 245],
                 [240, 244, 245],
```

Picture1: the matrix after decoding

```
In [14]: img.shape
Out[14]: (850, 651, 3)
```

Picture2: the shape of matrix

And if we want to reduce the image size, one algorithm to finish that is to project the image size into smaller size, we choose the scattered pixels to form a smaller one, the code shows the method.

```
dst_height = int(img_height*0.5)
dst_width = int(img_width*0.5)
```

```
# info
# create a blank pattern
# calculate the x and y
import numpy as np
```

```
dst_img = np.zeros((dst_height,dst_width,3), np.uint8) #0-255
```

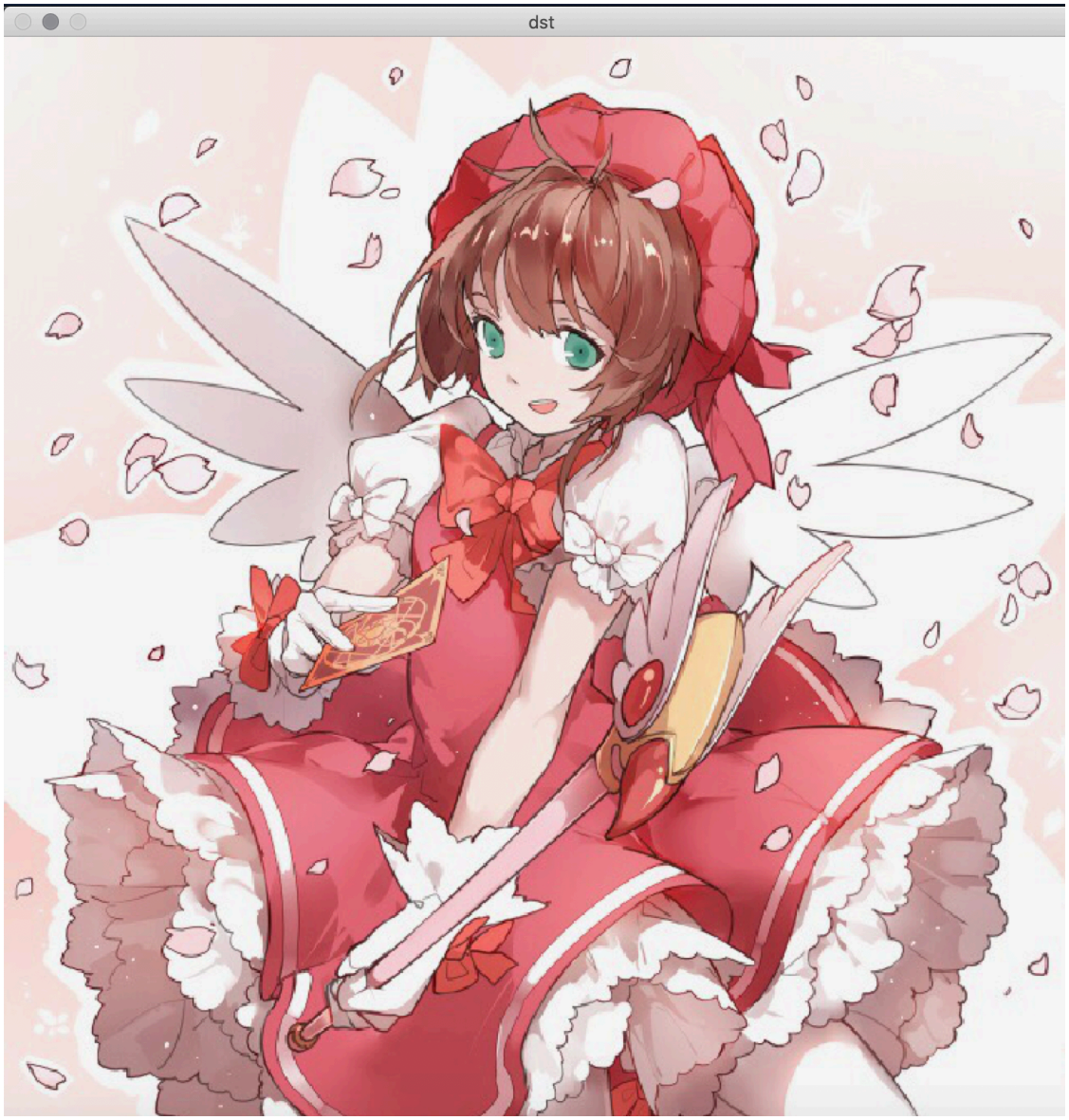
```
for i in range(0, dst_height):
    for j in range(0, dst_width):
        i_new = int(i*(img_height*1.0/dst_height))
        j_new = int(j*(img_width*1.0/dst_width))
        dst_img[i, j] = img[i_new, j_new]
```

Picture3: algorithm of reduction

We use dst_height and dst_width to define the size we want to reduce. The i_new and j_new represents the point which to be projected. The result is showed as below.

Picture4: picture after reduction

For the expansion we doing the same way, just need to adjust the parameter of multiplication. The result was showed below.



Picture5: picture after expansion

The next part is sampling (量子化レベルを変更し)， this part is the challenging part I think, because the clustering algorithm is difficult to describe in program. I figure out a way that is first finding the levels which could occurred, like the 8bit levels should be as below:

```
color_array
```

```
[0, 64.0, 128.0, 192.0, 256.0]
```

Picture6: 8Bit color levels

The later progress is same, we create a zero matrix with the same size, and every pixel point finding the closest color level and replace it. Because of the picture size could influence the program's terminating time, so I have to use the picture with 0.5 reduction size to complete. The code and result are as below.

```
half = (256/sampling_bit)/2
```

```
half
```

```
32.0
```

```
color = img[0][0]
print(color)
```

```
[240 244 245]
```

```
dst_height,dst_width
```

```
(425, 325)
```

```
for i in range(0, dst_height):
    for j in range(0, dst_width):
        color = dst_img[i, j]
        new_color = []
        for number in range(3):
            for level in color_array:
                if (color[number] < level+half*2) and (level < color[number]):
                    if ((color[number]-level) < half):
                        dst2_img[i, j][number] = level
                    else:
                        dst2_img[i, j][number] = level+half
                    print(dst2_img[i, j][number],number,i,j)
```

Picture 7: sampling algorithm

Picture 8: sampling result

Last but not least, the picture saving still use the library:

```
cv2.imwrite('IMG_0129.jpeg', dst2_img, [cv2.IMWRITE_JPEG_QUALITY,100])
```

```
True
```

Picture 9: saving with OpenCV

That’s all about for this task’s programming.

3. Running method

The program can run with at least the OpenCV and numpy library, make sure the picture and code are in the same folder. The output for this program containing two pictures with expansion and sampling (8bit in default).

[1] OpenCV, https://opencv.org/

[2] Jupiter, https://jupyter.org/