The First assignment of Imaging Procesing

Yuyl-2016210426

March 13, 2017

1 Input the test images to the MATLAB

For solving this task, all we need to do is just simply using the imread() function.

For example im = imread('strawberry');

2 Counting and representing the pixels for different intensities or different colors

I will directly get to solve the color images, the core of the implementation is to use find() and length() functions. By taking advantage of loop instruction, we can realize the algorithm:

```
for ii = 1:3
   for jj = 1:256
        ...
   end
end
```

3 Implementing the interpolation functions for a gray-scale images

The principle is just to use linear interpolation twice, and the formula is like:

$$f(x,0) = \frac{x-1}{0-1} * f(0,0) + \frac{x-0}{1-0} * f(1,0) \quad (1)$$

$$f(x,1) = \frac{x-1}{0-1} * f(0,1) + \frac{x-0}{1-0} * f(1,1) \quad (2)$$

Conbine two formula above:

$$f(x,y) = \frac{y-1}{0-1} * f(x,0) + \frac{y-0}{1-0} * f(x,1)$$

4 Rotating an gray-scale image

The main idea of rotation is to match each pixel to origin pixel. There are some points that need to be paid attention to:

1: The transformed images may need a larger matrix, since rotation may make diagonal line to width or height.

2: There are some points that we can't find the original matching points directly, so it will need to use binary interpolation that i mentioned above.

3: The method for find the matching pixels is to invert rotation matrix.

4: Another interpolation method is called weighted interpolation, which take advantage of the four corner points to get the value of required points. But each corner will be added a 'weight'. The details is illustrated in the code file.

```
pix_up_left = [floor(p(1)) floor(p(2))];
pix_up_right = [floor(p(1)) ceil(p(2))];
pix_down_left = [ceil(p(1)) floor(p(2))];
pix_down_right = [ceil(p(1)) ceil(p(2))];

value_up_left =( 1 - float_X) * (1 - float_Y);
value_up_right =float_X * (1 - float_Y);
value_down_left = (1 - float_X) * float_Y;
value_down_right = float_X * float_Y;

im3_r(i + delta_y, j + delta_x) = ...
       value_up_left * im3(pix_up_left(1), pix_up_left(2))+ ...
       value_up_right * im3(pix_up_right(1), pix_up_right(2))+ ...
       value_down_left * im3(pix_down_left(1), pix_down_left(2))+ ...
       value_down_right * im3(pix_down_right(1), pix_down_right(2));
```

5 For color image

It is very similar to do implementation for color image.