Image Processing Homework1

黄千慧

2019 年 11 月 17 日

使用 matlab 写一个函数, img = generateFigure(imgW, imgH), 其作用为产生一幅的彩色图像, 图像中用红色显示 [0, 2π] 的正弦波, 用绿色显示 [0, 2π] 的余弦波，蓝色显示 [0, 2π] 的 $y = x^2$ 图像。

MATLAB code:

```
function generateFigure(imgW, imgH)

figure;
x = [0: 0.1 : 2*pi];
y = x.*x;
plot(x, cos(x), "-g;cos(x);", x, sin(x), "-r;sin(x);", x, y ,
    "-b;x^2;");
xlabel('x');
ylabel('y');
axis([0 2*pi -1 1]);

% ================================================================
end
```

邻近插值和双线性插值算法

```
import numpy as np
import matplotlib.pyplot as plt

image_in = plt.imread('./1-512.jpg')
plt.imshow(image_in)
plt.axis('off')
plt.show()


def bilinear_interpolation(image_in, frac):
    m, n, f = image_in.shape
    w, h = int(m * frac), int(n * frac)

    image_out = np.zeros((w, h, f), dtype='uint8')
```

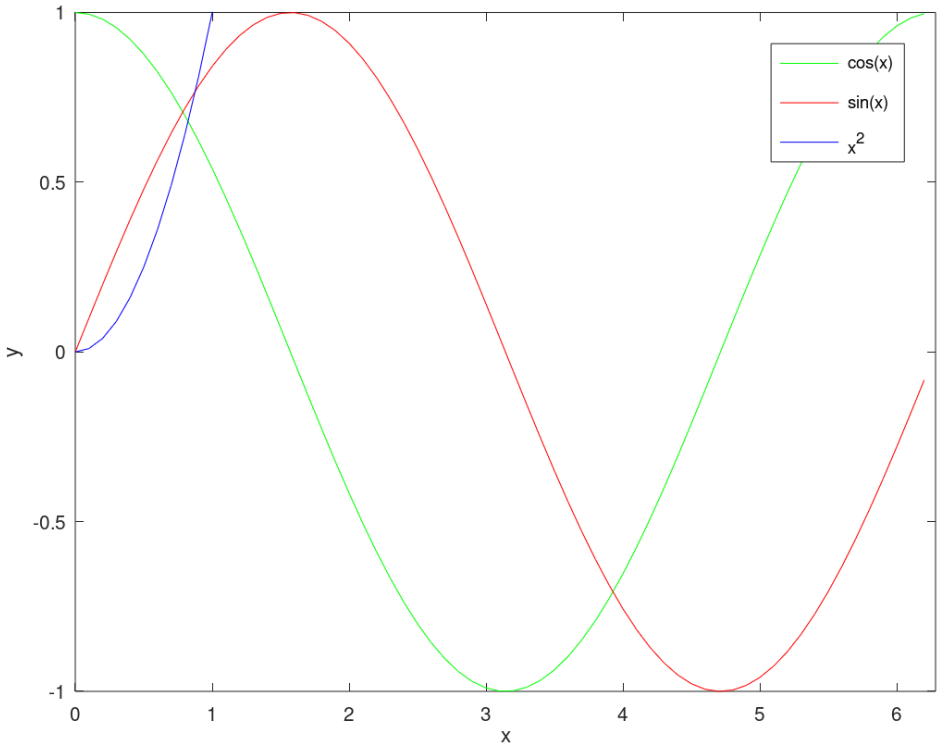

图 1: generateFigure() 运行结果

```
    for i in range(w - 1):
        for j in range(h - 1):
            r_out = i / frac
            c_out = j / frac

            int_r = int(np.floor(r_out))
            int_c = int(np.floor(c_out))

            delta_r = r_out - int_r
            delta_c = c_out - int_c

            int_r = max(min(m - 2, int_r), 0)
            int_c = max(min(n - 2, int_c), 0)

            for k in range(f):
                image_out[i, j, k] = image_in[int_r, int_c, k]
                    * (1 - delta_r) * (1 - delta_c) + \
                                     image_in[int_r + 1, int_c,
                                         k] * delta_r * (1 -
                                        delta_c) + \
                                     image_in[int_r, int_c + 1,
                                         k] * (1 - delta_r) *
                                        delta_c + \
                                     image_in[int_r + 1, int_c
                                        + 1, k] * delta_r *
                                        delta_c

    return image_out


def nearest_neighbor(image_in, frac):
    m, n, f = image_in.shape
    w, h = int(m * frac), int(n * frac)

    image_out = np.zeros((w, h, f), dtype='uint8')

    for i in range(w):
        for j in range(h):
            r_out = i / frac
            c_out = j / frac

            int_r = int(np.floor(r_out))
```

```
50                int_c = int(np.floor(c_out))
51
52                int_r = max(min(m - 1, int_r), 0)
53                int_c = max(min(n - 1, int_c), 0)
54
55                for k in range(f):
56                    image_out[i, j, k] = image_in[int_r, int_c, k]
57
58        return image_out
59
60
61    near_out = nearest_neighbor(image_in, 2)
62
63    plt.imshow(near_out)
64    plt.axis('off')
65    plt.show()
66    plt.imsave("./1-1024.png", near_out)
67
68    bilinear_out = bilinear_interpolation(image_in, 2)
69
70    plt.imshow(bilinear_out)
71    plt.axis('off')
72    plt.show()
73    plt.imsave("./2-1024.png", bilinear_out)
```

运行结果在文件目录的 1-1024.png 和 2-1024.png