《图像处理导论》

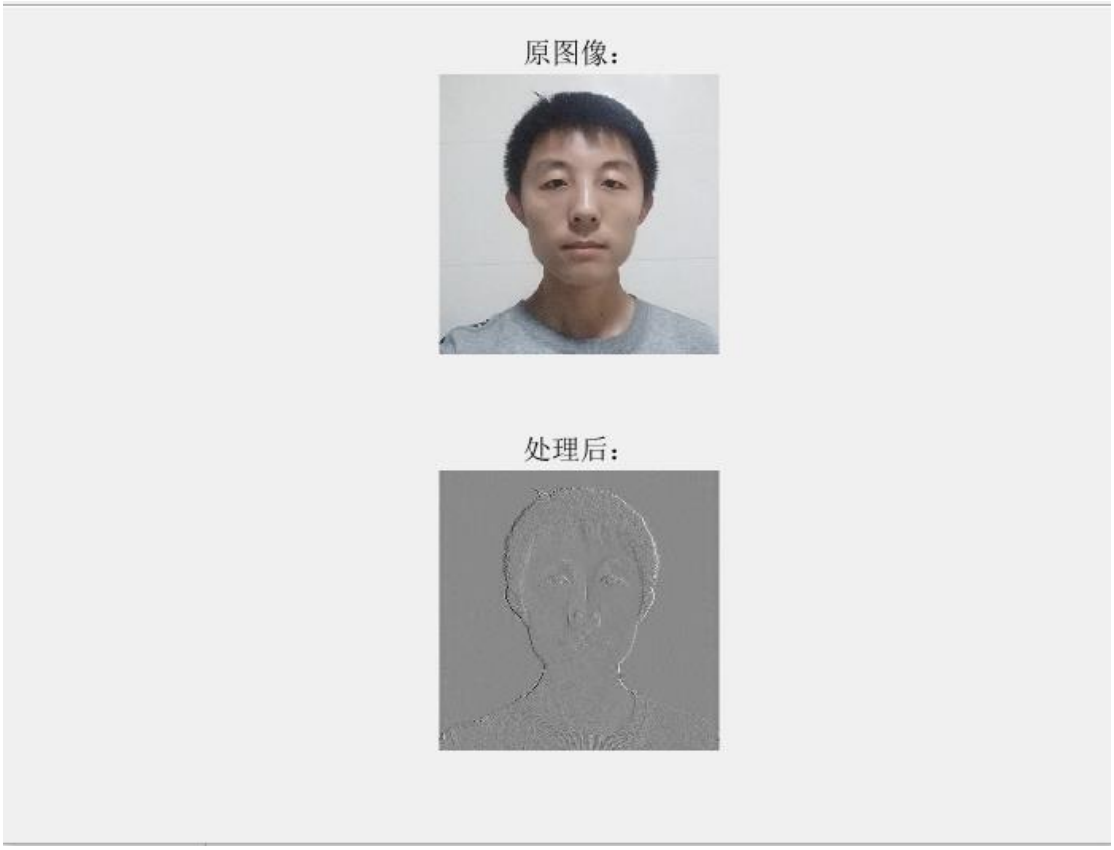
第二次作业

学院：计算机科学与工程学院
班级：计算机 1606 班
姓名：戚子强
学号：20164625

题目：

用傅里叶变换实现个人头像的边缘检测。

结果展示：



代码：

```
//Matlab 实现
imo = imread('tx.jpg');
im = rgb2gray(imo);
% 计算频域中的差分算子
nx = size(im, 2);
hx = ceil(nx/2)-1;
ftdiff = (2i*pi/nx)*(0:hx);
ftdiff(nx:-1:nx-hx+1) = -ftdiff(2:hx+1);   % 共轭对称
g = ifft2( bsxfun(@times, fft2(im), ftdiff) ); % FFT
% Result
figure;
subplot(2,1,1);imshow(imo);title('原图像：');
```

subplot(2,1,2);imshow(g,[]);title('处理后：');

//OpenCV & C++实现

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace cv;
using namespace std;
int main()
{
    //读取图像
    Mat src_image = imread("t2.jpg");
    //图像读取出错处理
    if (!src_image.data)
    {
        cout << "src image load failed!" << endl;
        return -1;
    }
    //显示源图像
    namedWindow("原图像", WINDOW_NORMAL);
    imshow("原图像", src_image);
    //此处高斯去燥有助于后面二值化处理的效果
    //Mat blur_image;
    //GaussianBlur(src_image, blur_image, Size(15, 15), 0, 0);
    //imshow("GaussianBlur", blur_image);
    /*灰度变换与二值化*/
    Mat gray_image, binary_image;
    cvtColor(src_image, gray_image, COLOR_BGR2GRAY);
    threshold(gray_image, binary_image, 30, 255, THRESH_BINARY |
THRESH_TRIANGLE);
    //imshow("binary", binary_image);
    /*形态学闭操作*/
    Mat morph_image;
    Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3), Point(-1, -1));
    morphologyEx(binary_image, morph_image, MORPH_CLOSE, kernel, Point(-1, -1),
2);
    //imshow("morphology", morph_image);

    /*查找外轮廓*/
    vector< vector<Point> > contours;
```

```
    vector<Vec4i> hireachy;
    findContours(binary_image, contours, hireachy, CV_RETR_EXTERNAL,
CHAIN_APPROX_NONE, Point());
    int l;//目标轮廓索引
    //寻找最大轮廓，即目标轮廓
    for (size_t t = 0; t < contours.size(); t++)
    {
        /*过滤掉小的干扰轮廓*/
        Rect rect = boundingRect(contours[t]);
        if (rect.width < src_image.cols / 2)
            continue;
        //if (rect.width >(src_image.cols - 20))
        l = t;//找到了目标轮廓，获取轮廓的索引
    }
    //画出目标轮廓
    Mat result_image = Mat::zeros(src_image.size(), CV_8UC3);
    vector< vector<Point> > draw_contours;
    draw_contours.push_back(contours[l]);
    drawContours(result_image, draw_contours, -1, Scalar(255, 255, 255), 1, 8,
hireachy);
    namedWindow("处理后", WINDOW_NORMAL);
    imshow("处理后", result_image);
    //计算轮廓的傅里叶描述子
    Point p;
    int x, y, s;
    int i = 0, j = 0, u = 0;
    s = (int)contours[l].size();
    Mat src1(Size(s, 1), CV_8SC2);
    float f[9000];//轮廓的实际描述子
    float fd[16];//归一化后的描述子，并取前15个
    for (u = 0; u < s; u++)
    {
        float sumx = 0, sumy = 0;
        for (j = 0; j < s; j++)
        {
            p = contours[l].at(j);
            x = p.x;
            y = p.y;
            sumx += (float)(x * cos(2 * CV_PI * u * j / s) + y * sin(2 * CV_PI
* u * j / s));
            sumy += (float)(y * cos(2 * CV_PI * u * j / s) - x * sin(2 * CV_PI
* u * j / s));
        }
        src1.at<Vec2b>(0, u)[0] = sumx;
```

```
            src1.at<Vec2b>(0, u)[1] = sumy;
            f[u] = sqrt((sumx * sumx) + (sumy * sumy));
        }
        //傅立叶描述字的归一化
        f[0] = 0;
        fd[0] = 0;
        for (int k = 2; k < 17; k++)
        {
            f[k] = f[k] / f[1];
            fd[k - 1] = f[k];
            cout << fd[k - 1] << endl;
        }
        //保存数据
        for (int k = 0; k < 16; k++)
        {
            FILE* fp = fopen("1.txt", "a");
            fprintf(fp, "%8f\t", fd[k]);
            fclose(fp);
        }
        FILE* fp = fopen("1.txt", "a");
        fprintf(fp, "\n");
        fclose(fp);
        waitKey();
        return 0;
}
```