计算机视觉实验报告

——王培钰 16340220 软工 6 班

1. CImg 的配置

(1) 下载 CImg 的包到指定的文件夹下，解压

(2) 然后只需要在编写的 cpp 或 hpp 文件中加入

```
#include "XX/CImg.h" // XX/是指 CImg 所在的路径
using namespace cimg_library;
```

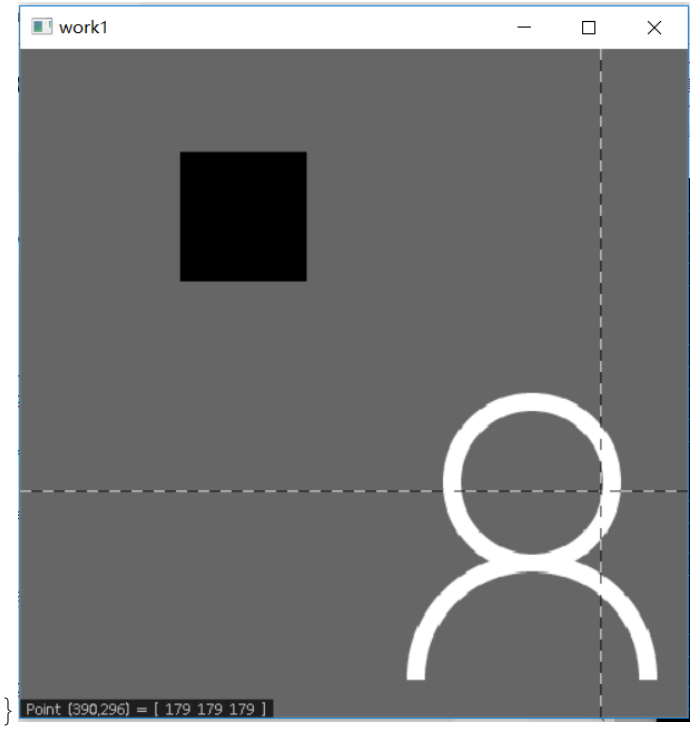
(3) 编译命令（**环境为 windows 下的 MinGW**）

```
g++ -o HelloWorld.exe HelloWorld.cpp -O2 -lgdi32
```

2. 简单使用

(1) 读入 1.bmp 文件并显示

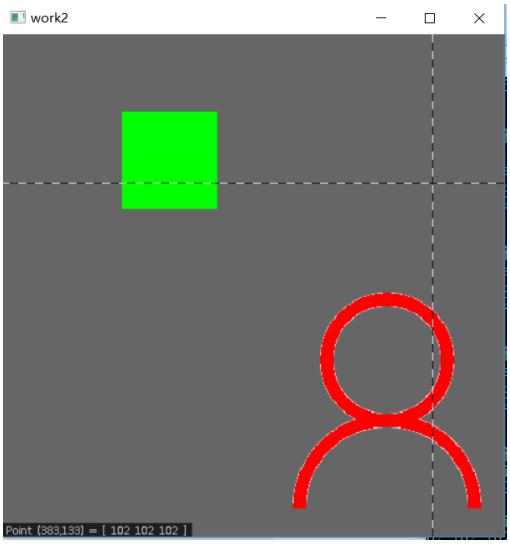
```
CImg<unsigned char> SrcImg;
SrcImg.load_bmp("1.bmp");
void test_display(CImg<unsigned char> SrcImg) {
    SrcImg.display("work1");
```



(2) 把 1.bmp 文件的白色区域变成红色，黑色区域变成绿色

```
void test_change(CImg<unsigned char> SrcImg) {
    CImg<unsigned char> Img = SrcImg;
    cimg_forXY(Img, x, y) {
        if (Img(x,y,0) == 255 && Img(x,y,1) == 255 && Img(x,y,2) == 255)
{
            Img(x,y,0) = 255;
            Img(x,y,1) = 0;
            Img(x,y,2) = 0;
        }
    }
    cimg_forXY(Img, x, y) {
        if (Img(x,y,0) == 0 && Img(x,y,1) == 0 && Img(x,y,2) == 0) {
            Img(x,y,0) = 0;
            Img(x,y,1) = 255;
            Img(x,y,2) = 0;
        }
```

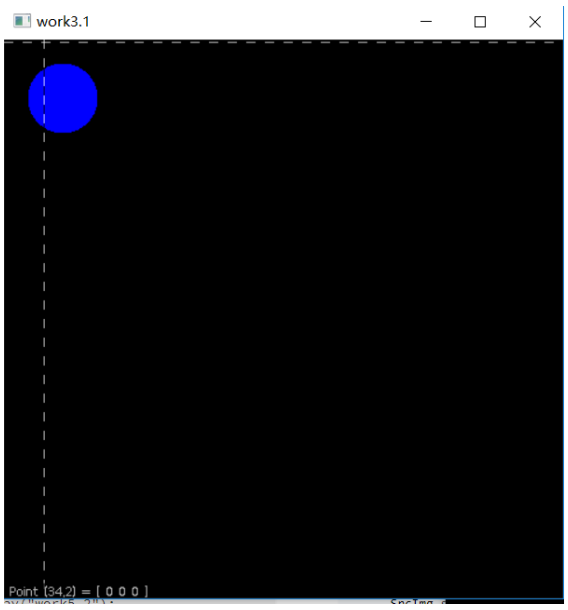
```
    }
    Img.display("work2");

}
```



(3) 在图上绘制一个圆形区域，圆心坐标(50, 50)，半径为 30，填充颜色为蓝色

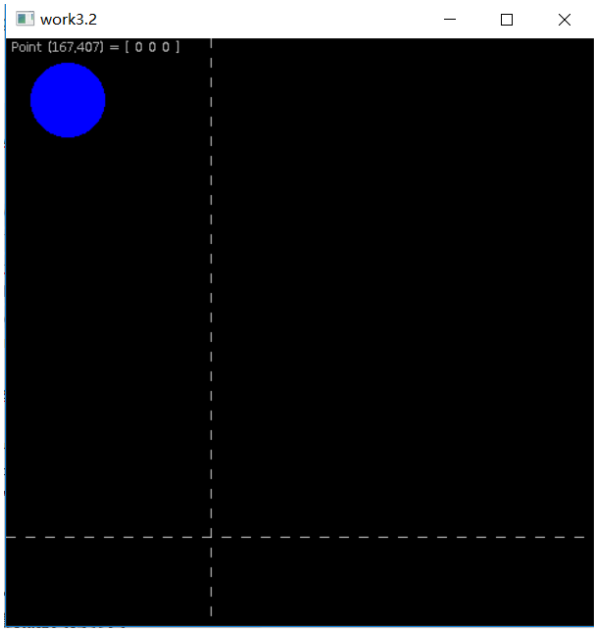
- 未使用 **CImg** 的接口函数

```
void DrawCircle_blue1(CImg <unsigned char> TempImg) {
    CImg<unsigned char> SrcImg = TempImg;
    cimg_forXY(SrcImg, x, y) {
        if (pow(pow(x-50,2)+pow(y-50,2),0.5) < 30) {
            SrcImg(x,y,0) = 0;
            SrcImg(x,y,1) = 0;
            SrcImg(x,y,2) = 255;
        }
    }
    SrcImg.display("work3.1");

}
```



- 使用接口函数 **draw_circle()**

```
void DrawCircle_blue2(CImg <unsigned char> TempImg) {
    unsigned char blue[] = {0,0,255};
    TempImg.draw_circle(50, 50, 30, blue);
    TempImg.display("work3.2");

}
```

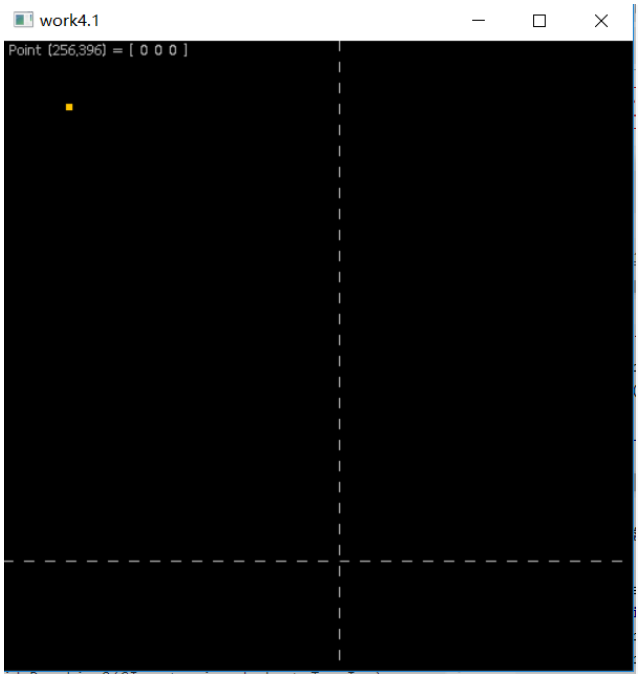


PS: 这个感觉自己实现的算法和接口函数实现的效果差不多

(4) 在图上绘制一个圆形区域，圆心坐标(50,50)，半径为 3，填充颜色为黄色

- 未使用 **CImg** 的接口函数

```
void DrawCircle_yellow1(CImg <unsigned char> TempImg) {
    CImg<unsigned char> SrcImg = TempImg;
    cimg_forXY(SrcImg, x, y) {
        if (pow(pow(x-50,2)+pow(y-50,2),0.5) < 3) {
            SrcImg(x,y,0) = 200;
            SrcImg(x,y,1) = 155;
            SrcImg(x,y,2) = 0;
        }
    }
    SrcImg.display("work4.1");

}
```



- 使用接口函数 **draw_circle()**

```
void DrawCircle_yellow2(CImg <unsigned char> TempImg) {
    unsigned char yellow[] = {200, 155, 0};
    TempImg.draw_circle(50, 50, 3, yellow);
    TempImg.display("work4.2");

}
```

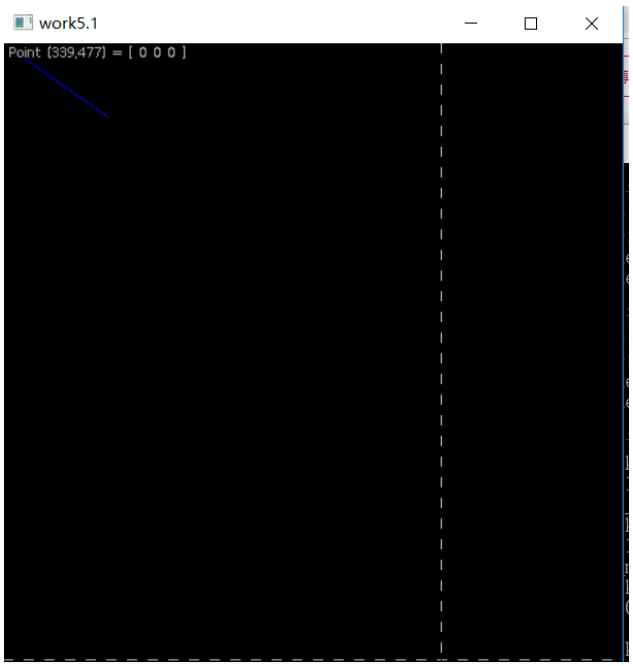
PS: 这个虽然接口函数的效果也并不是特别好，但是明显比我的要好很多，但是因为遍历图片的像素点是对图片进行采样所以点坐标为整数值，圆的半径越小意味着我们取的像素点范围越少，点越少，所以图片的效果会较差，甚至接近正方形，可能接口的函数有更好的实现方法，尝试去翻了翻源码没太看懂。

（5）在图上绘制一条长为 100 的直线段，起点坐标为(0，0)，方向角为 35 度，直线的颜色为蓝色

- 未使用 **CImg** 的接口函数

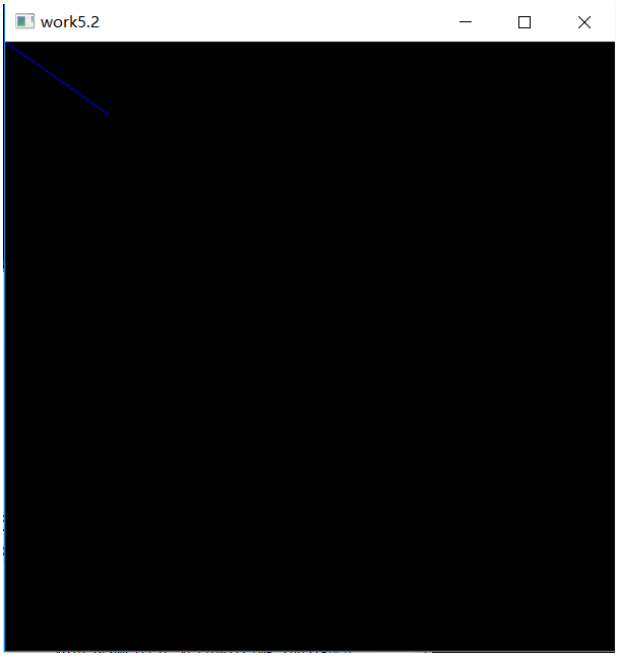
```
void DrawLine1(CImg <unsigned char> TempImg) {
    CImg<unsigned char> SrcImg = TempImg;
    double x0 = 100*cos(35*pi/180);
```

```
    double y0 = 100*sin(35*pi/180);
    cimg_forXY(SrcImg, x, y) {
        if (x == 0) {
            if (y == 0) {
                SrcImg(x,y,0) = 0;
                SrcImg(x,y,1) = 0;
                SrcImg(x,y,2) = 255;
            }
        }
        else {
            if (cmp((double)y, (double)x*tan(35*pi/180)) && (double)x <=
x0 && (double)y <= y0) {
                SrcImg(x,y,0) = 0;
                SrcImg(x,y,1) = 0;
                SrcImg(x,y,2) = 255;
            }
        }
    }
    SrcImg.display("work5.1");
}

//用于上面函数的一个辅助函数，来判断相等条件
bool cmp(double x , double y) { //compare x and y，如果差值小于一定范围则
近似相等
    if (abs(x - y) <= 0.5)
        return 1;
    return 0;
}
```



- 使用接口函数 **draw_line()**

```
void DrawLine2(CImg <unsigned char> TempImg) {
    unsigned char blue[] = {0,0,255};
    TempImg.draw_line(0,0,100*cos(35*pi/180),100*sin(35*pi/180),blue);
    TempImg.display("work5.2");
}
```



PS: 这个比较效果感觉肉眼很难看出差距，但是因为遍历图片的像素点是对图片进行采样所以点坐标为整数值，所以不可能存在彻底等于所得的浮点数 tan 角要求的坐标，只能通过设置误差尽可能小，也就是直线的粗细，通过设置 cmp 函数调节几个值（0.1, 0.3, 0.5, 1.0）发现 0.5 的效果是最好的。

（6）把上面的操作结果保存为 2.bmp

```
SrcImg.save("2.bmp");
```

3. 将上述实现的函数封装为 c++的一个类

ex.hpp:

```
#ifndef _EX_HPP_
#define _EX_HPP_
#include "../CImg.h"
#include <cmath>
#include <string>
using namespace std;
using namespace cimg_library;
const double pi(3.14159265);
class Test
{
   public:
        Test();
        ~Test();
        void Todisplay();
        void change(); //把 1.bmp 文件的白色区域变成红色，黑色区域变成绿色
        void DrawCircle_blue1(); //不使用 CImg 函数在图上绘制一个圆形区域，圆心坐标(50,50)，半径为 30，填充颜色为蓝色
```

```
        void DrawCircle_yellow1();//不使用 CImg 函数在图上绘制一个圆形区域，圆心坐标(50,50)，半径为 3，填充颜色为黄色
        void DrawLine1();//不使用 CImg 函数 在图上绘制一条长为 100 的直线段，起点坐标为(0, 0)，方向角为 35 度，直线的颜色为蓝色。
        //下面三个函数分别对应使用 CImg 函数的上述三个操作
        void DrawCircle_blue2();
        void DrawCircle_yellow2();
        void DrawLine2();
        CImg<unsigned char> getSrcImg();
    private:
        //string name; //图片的名称
        CImg<unsigned char> SrcImg; //定义一副图片
};
#endif
```

ex.cpp:

```
#include "ex.hpp"
using namespace std;
bool cmp(double x , double y);

Test::Test() {
    //name = "work1";
    SrcImg.load_bmp("1.bmp");
}
Test::~Test() {}
void Test::Todisplay() {
    //string t = name;
    SrcImg.display("homework");
```

```
}
CImg<unsigned char> Test::getSrcImg() {
    return SrcImg;
}
void Test::change() {
    //name = "work2";
    //CImg<unsigned char> Img = SrcImg;
    cimg_forXY(SrcImg, x, y) {
        if (SrcImg(x,y,0) == 255 && SrcImg(x,y,1) == 255 &&
SrcImg(x,y,2) == 255) {
            SrcImg(x,y,0) = 255;
            SrcImg(x,y,1) = 0;
            SrcImg(x,y,2) = 0;
        }
    }
    cimg_forXY(SrcImg, x, y) {
        if (SrcImg(x,y,0) == 0 && SrcImg(x,y,1) == 0 && SrcImg(x,y,2) ==
0) {
            SrcImg(x,y,0) = 0;
            SrcImg(x,y,1) = 255;
            SrcImg(x,y,2) = 0;
        }
    }
}
void Test::DrawCircle_blue1() {
    cimg_forXY(SrcImg, x, y) {
        if (pow(pow(x-50,2)+pow(y-50,2),0.5) < 30) {
            SrcImg(x,y,0) = 0;
            SrcImg(x,y,1) = 0;
            SrcImg(x,y,2) = 255;
```

```
        }
    }
}
void Test::DrawCircle_yellow1() {
    cimg_forXY(SrcImg, x, y) {
        if (pow(pow(x-50,2)+pow(y-50,2),0.5) < 3) {
            SrcImg(x,y,0) = 200;
            SrcImg(x,y,1) = 155;
            SrcImg(x,y,2) = 0;
        }
    }
}
void Test::DrawLine1() {
    double x0 = 100*cos(35*pi/180);
    double y0 = 100*sin(35*pi/180);
    cimg_forXY(SrcImg, x, y) {
        if (x == 0) {
            if (y == 0) {
                SrcImg(x,y,0) = 0;
                SrcImg(x,y,1) = 0;
                SrcImg(x,y,2) = 255;
            }
        }
        else {
            if (cmp((double)y, (double)x*tan(35*pi/180)) && (double)x <=
x0 && (double)y <= y0) {
                SrcImg(x,y,0) = 0;
                SrcImg(x,y,1) = 0;
                SrcImg(x,y,2) = 255;
            }
```

```
        }
    }
}
void Test::DrawCircle_blue2() {
    unsigned char blue[] = {0,0,255};
    SrcImg.draw_circle(50, 50, 30, blue);
}
void Test::DrawCircle_yellow2() {
    unsigned char yellow[] = {200, 155, 0};
    SrcImg.draw_circle(50, 50, 3, yellow);
}
void Test::DrawLine2() {
    unsigned char blue[] = {0,0,255};
    SrcImg.draw_line(0,0,100*cos(35*pi/180),100*sin(35*pi/180),blue);
}
bool cmp(double x , double y) { //compare x and y，如果差值小于一定范围则近似相等
    if (abs(x - y) <= 0.5)
        return 1;
    return 0;
}
```

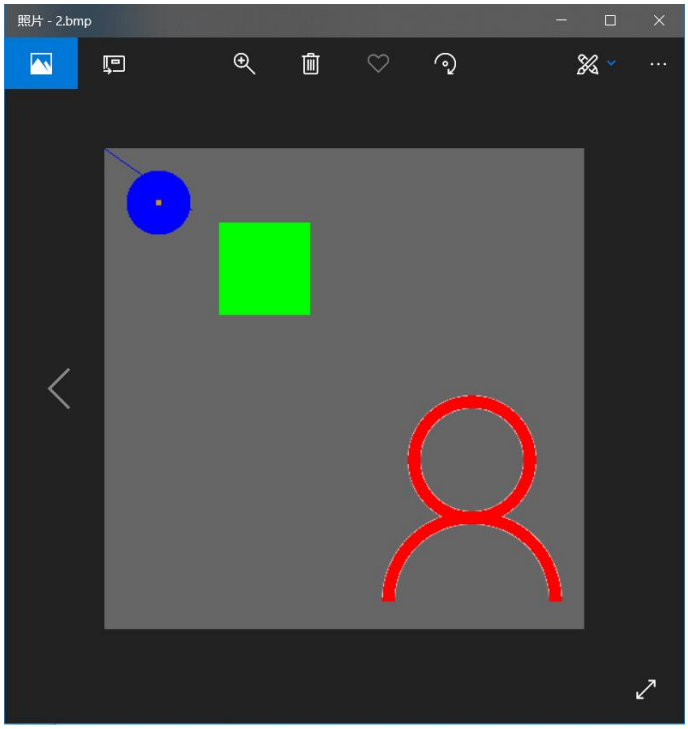
main.cpp

```
#include "ex.hpp"
using namespace std;
int main(int argc, char const *argv[])
{
    Test pic;
    Test pic1;
```

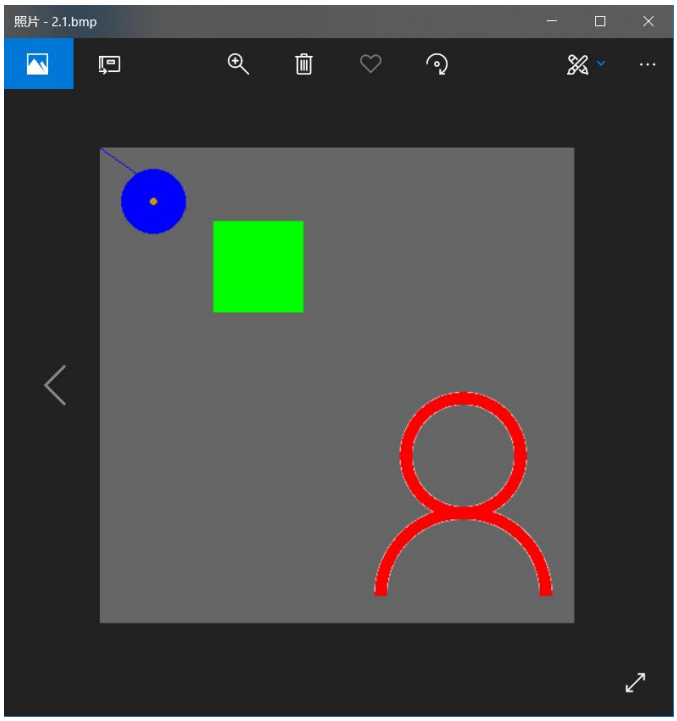
```
    pic.change();
    pic.DrawCircle_blue1();
    pic.DrawCircle_yellow1();
    pic.DrawLine1();
    pic.Todisplay();
    CImg<unsigned char> temp = pic.getSrcImg();
    temp.save("2.bmp");
    pic1.change();
    pic1.DrawCircle_blue2();
    pic1.DrawCircle_yellow2();
    pic1.DrawLine2();
    pic1.Todisplay();
    CImg<unsigned char> temp1 = pic1.getSrcImg();
    temp1.save("2.1.bmp");
    return 0;
}
```

存储后图片总体效果如下：

未使用接口函数：



使用接口函数：



4. 思考：

为什么第四步绘制的圆形区域形状效果不好？

正如前文图像比较时所说，图像处理的方式是采取采样处理的方式，所以对图像

的像素点的遍历是只会去整数值，而第四步对应的图像范围小，也就采样能取到的像素点的个数少，误差也就越大。