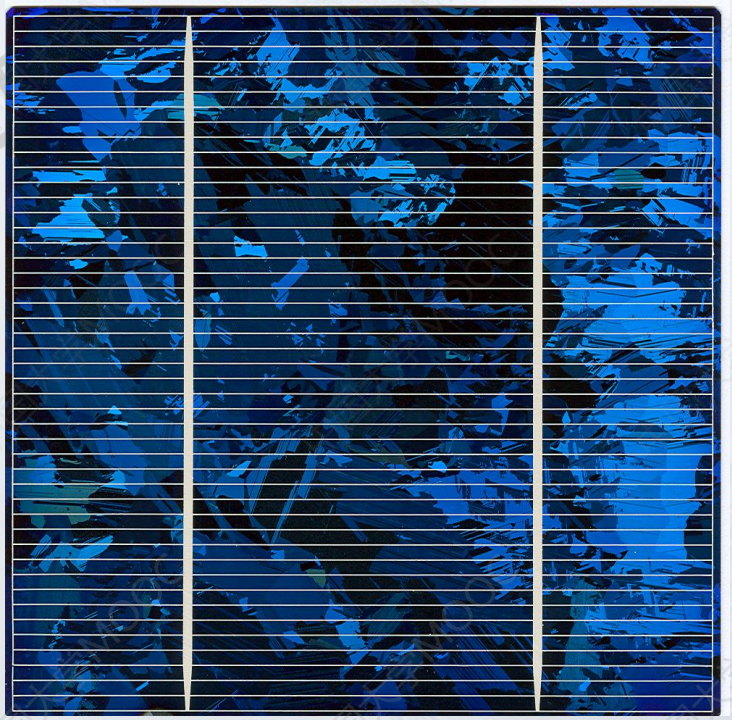
练习1

利用图像在色彩上的一些特点，可以在图像处理中进行一些巧妙的处理。通过opencv将下图的rgb三个通道分离，并观察每个通道的图像。



练习1

读取一张图片，存入名为src_color的Mat容器中

声明vector，作为分离后3个通道图像的保存容器

调用split函数，分离src_color的rgb三通道

定义3个Mat容器获得3个通道的分离结果，注意opencv中三个通道的排列顺序是B,G,R。

显示3个通道的图像，以及原图

```
int main()
{
    cv::Mat src_color = imread("solar_cell_01.png");
    std::vector<cv::Mat> channels;
    cv::split(src_color, channels);
    cv::Mat B = channels.at(0);
    cv::Mat G = channels.at(1);
    cv::Mat R = channels.at(2);
    cv::imshow("red", R);
    cv::imshow("blue", B);
    cv::imshow("green", G);
    cv::imshow("original Mat", src_color);
    waitKey(0);
}
```

练习2 调用本机的摄像头。

实例化一个VideoCapture类，名称为cap

cap(0)表示打开本机的第一个摄像头，()中如果是本地视频地址如cap(“D:\\1.avi”)则可以打开本地视频。

isOpened()检查视频是否开启，正常开启返回1

通过get()及不同参数可以获得视频的不同参数，如本例获得视频的fps

声明Mat类型图片，名称为frame，并通过read()获得视频的当前帧

通过imshow显示当前帧

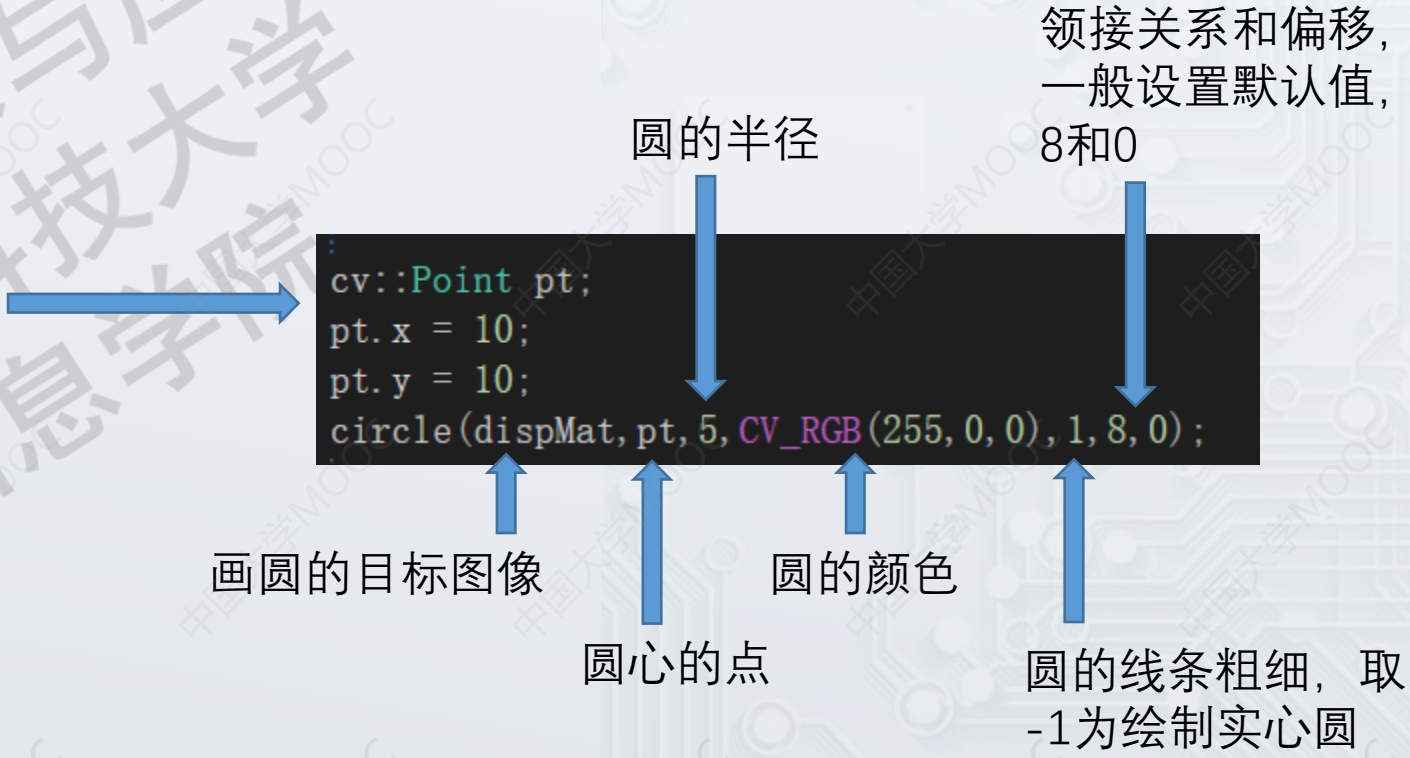
waitKey延时，如缺少，无法显示

```
int main()
{
    VideoCapture cap;

    cap.open(0);

    if(!cap.isOpened())
    {
        std::cout << "不能打开视频文件"<< std::endl;
        return-1;
    }

    double fps = cap.get(CAP_PROP_FPS);
    std::cout << "fps" << fps << std::endl;
    while (1)
    {
        cv::Mat frame;
        bool rSucess = cap.read(frame);
        if (!rSucess)
        {
            std::cout << "不能从视频文件中读取帧" << std::endl;
            break;
        }
        else
        {
            cv::imshow("frame", frame);
        }
        waitKey(30);
    }
```

练习3 opencv的基本绘图功能

画圆：

Opencv中定义一个点的语句，然后对x,y的坐标值赋值



练习3 opencv的基本绘图功能

画线段：

```
line(dispMat,pt1,pt2,CV_RGB(255,0,0),1,8,0);
```

领接关系和偏移，一般设置默认值，8和0

画线的目标图像

线段的起点和终点

线段的颜色

线段的粗细

画矩形框：

Opencv中定义一个矩形语句，定义完成后首先对x,y的坐标值赋值，然后对宽度和高度赋值

被绘制的矩形

```
cv::Rect rect;
rect.x = 10;
rect.y = 10;
rect.width;
rect.height;
rectangle(dispMat,rect,CV_RGB(255,0,0),1,8,0);
```

领接关系和偏移，一般设置默认值，8和0

画矩形的目标图像

矩形的颜色

矩形的粗细，-1则为实心矩形

练习4 直方图计算

定义一个容量为256的float型数组，遍历图像的每个像素，并计算直方图，将结果存入数组中。

数组声明方式

Float histgram[256];

练习5 直方图绘制

利用练习4的结果，以及画线或者画圆的函数，绘制一副直方图。