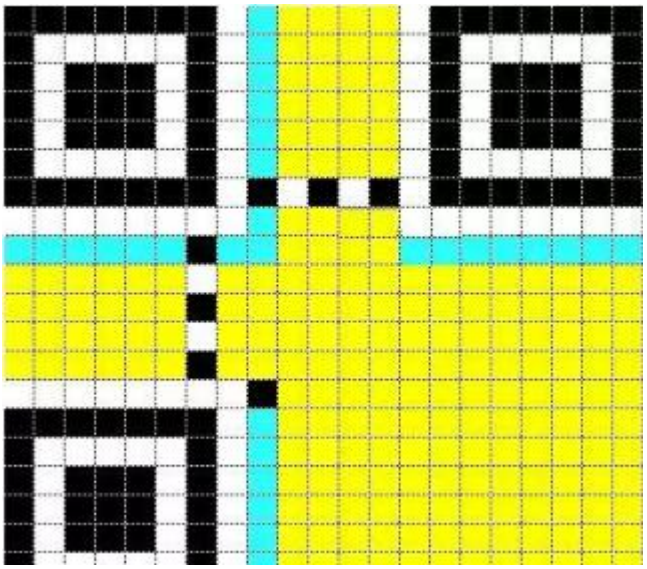
# OpenCV 二维码检测与定位

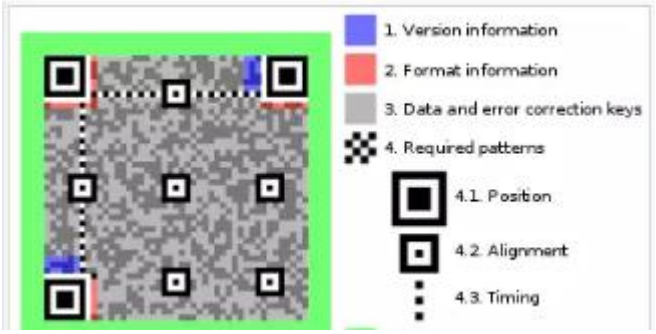
在如今流行扫描的年代，应用程序实现二维码扫描检测与识别已经是应用程序的标配、特别是在移动端、如果你的应用程序不能自动发现检测二维码，自动定位二维码你都不好意思跟别人打招呼，二维码识别与解析基于 ZXing 包即可。联为智能教育机器视觉金老师说：难点就在于如何从画面中快速而准确的找到二维码区域，寻找到二维码三个匹配模式点。

一：二维码的结构与基本原理

标准的二维码结构如下：



特别要关注的是图中三个黑色正方形区域，它们就是用来定位一个二维码的最重要的三个区域，我们二维码扫描与检测首先要做的就是要发现这三个区域，如果找到这个三个区域，我们就成功的发现一个二维码了，就可以对它定位与识别了。二维码其它各个部分的说明如下：




三个角上的正方形区域从左到右，从上到下黑白比例为 1:1:3:1:1。

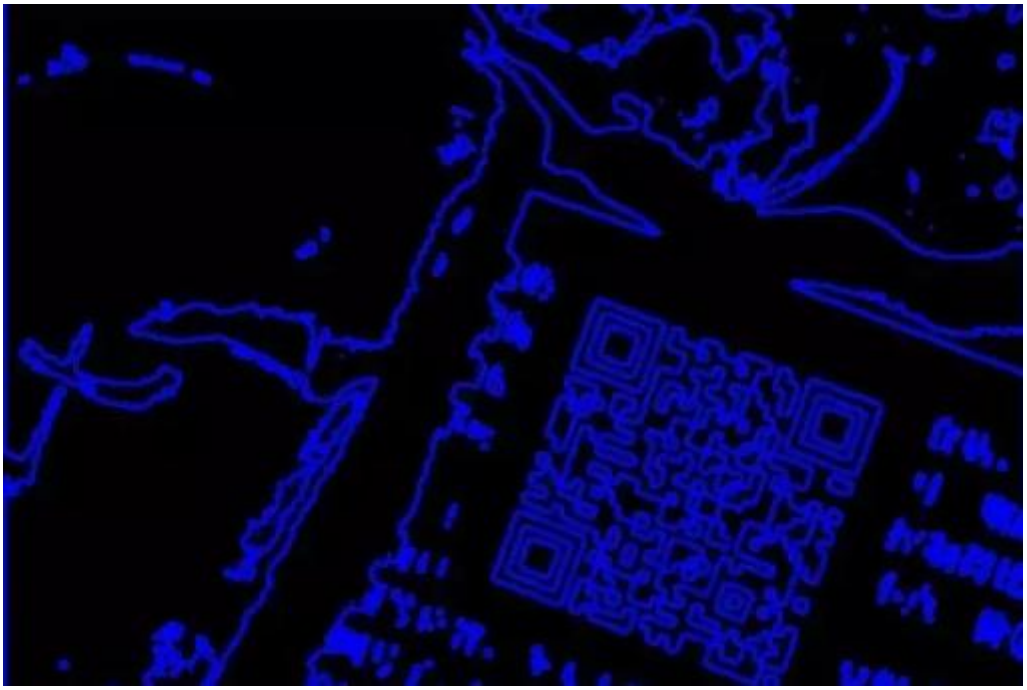

不管角度如何变化，这个是最显著的特征，通过这个特征我们就可以实现二维码扫描检测与定位。

## 二：算法各部与输出

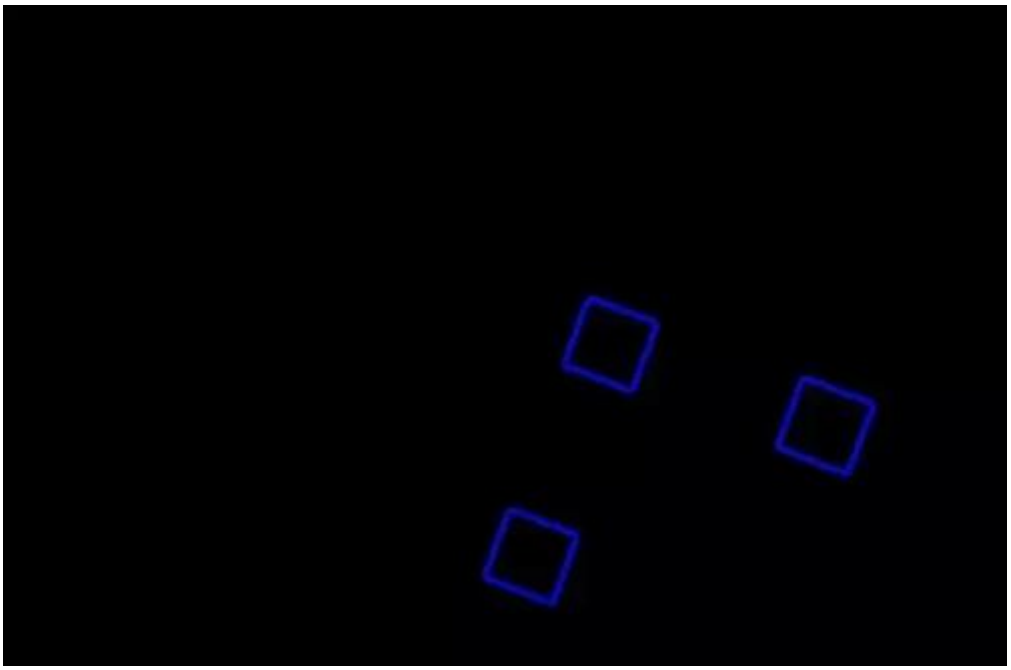
1. 首先把输入图像转换为灰度图像(cvtColor)

2. 通过 OTSU 转换为二值图像(threshold)

3. 对二值图像使用轮廓发现得到轮廓(findContours)

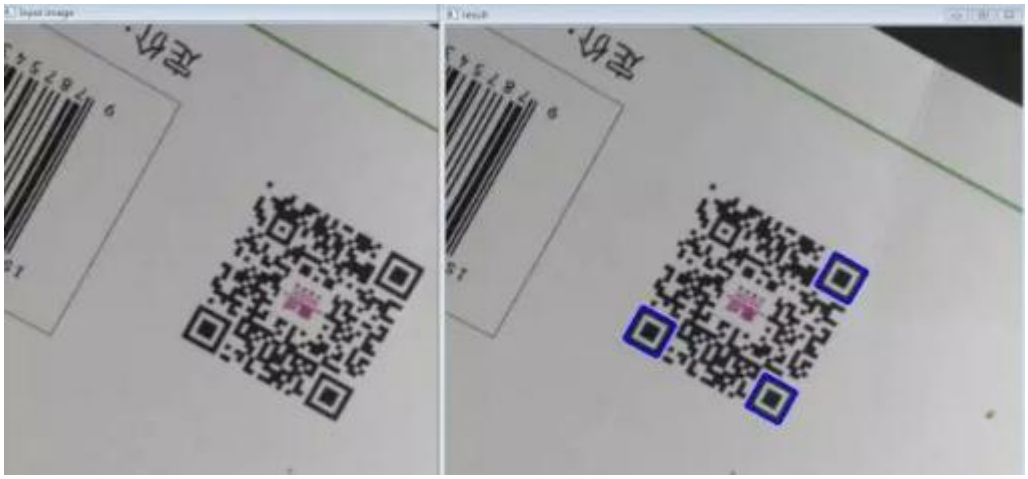


4. 根据二维码三个区域的特征 ,对轮廓进行面积与比例过滤得到最终结果显示如下：



## 三：程序运行结果演示

上述程序运行的最终结果，左侧为原图，右侧为检测结果



四：各个步骤代码实现

```
#include <opencv2/opencv.hpp>
#include <math.h>
#include <iostream>

using namespace cv;
using namespace std;

bool isCorner(Mat &image);
Mat transformCorner(Mat &image, RotatedRect &rect);
int main(int argc, char** argv) {
    Mat src = imread("D:/gloomyfish/qrcode_05.jpg");
    if (src.empty()) {
        printf("could not load image...\n");
        return -1;
    }
    namedWindow("input image", CV_WINDOW_AUTOSIZE);
    imshow("input image", src);

    Mat gray, binary;
    cvtColor(src, gray, COLOR_BGR2GRAY);
    imwrite("D:/gloomyfish/outimage/qrcode_gray.jpg", gray);

    threshold(gray, binary, 0, 255, THRESH_BINARY | THRESH_OTSU);
    imwrite("D:/gloomyfish/outimage/qrcode_binary.jpg", binary);


    vector<vector<Point>> contours;
    vector<Vec4i> hireachy;
    Moments monents;
    findContours(binary.clone(), contours, hireachy, RETR_LIST, CHAIN_APPROX_SIMPL
    Mat result = Mat::zeros(src.size(), CV_8UC3);
    for (size_t t = 0; t < contours.size(); t++) {
        double area = contourArea(contours[t]);

        if (area < 100) continue;
        RotatedRect rect = minAreaRect(contours[t]);

        float w = rect.size.width;
        float h = rect.size.height;
        float rate = min(w, h) / max(w, h);
        if (rate > 0.85 && w < src.cols/4 && h<src.rows/4) {
            printf("angle : %.2f\n", rect.angle);
            Mat qr_roi = transformCorner(src, rect);
            if (isCorner(qr_roi)) {
                drawContours(src, contours, static_cast<int>(t), Scalar(255, 0, 0)
```