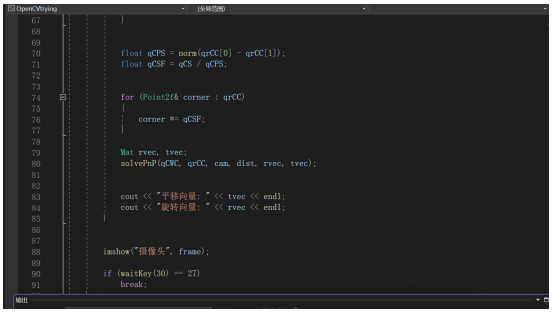
Task 1 必做题

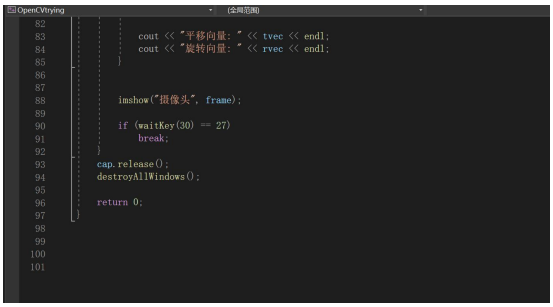
```
#include <opencv2/opencv.hpp>
#include <zbar.h>

using namespace cv;
using namespace std;
using namespace zbar;

int main()
{
    Mat cam = cv::Mat::eye(3, 3, CV_64F);
    cam.at<double>(0, 0) = 1268.909515807967;
    cam.at<double>(1, 1) = 1267.655763569812;
    cam.at<double>(0, 2) = 994.4963608576663;
    cam.at<double>(1, 2) = 506.8575983680511;
    Mat dist = cv::Mat::zeros(1, 5, CV_64F);
    dist.at<double>(0, 0) = 0.2111796702015687;
    dist.at<double>(0, 1) = -0.854848584673445;
    dist.at<double>(0, 2) = 0.004455538510197431;
    dist.at<double>(0, 3) = 0.003979156532473041;
    dist.at<double>(0, 4) = 1.154977352044173;

    vector<Point3f> qCWC
    {
        Point3f(-1, 1, 0),
        Point3f(1, 1, 0),
        Point3f(1, -1, 0),
```

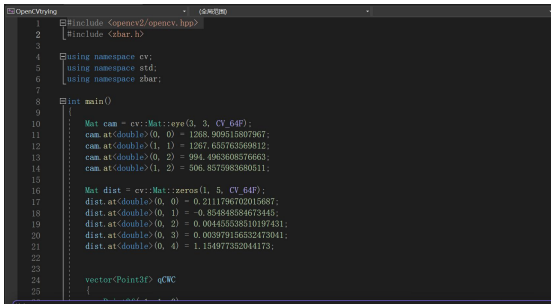
```
        Point3f(-1, -1, 0)
    };
    float qCS = 0.02;

    VideoCapture cap(0);
    if (!cap.isOpened())
    {
        cout << "无法打开摄像头！" << endl;
        return -1;
    }
    ImageScanner scanner;

scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);
    while (true)
    {

        Mat frame;
        Mat gray;
        Mat undF;
        cap >> frame;
        cvtColor(frame, gray, COLOR_BGR2GRAY);
        undistort(frame, undF, cam, dist);
        Image image(undF.cols, undF.rows, "Y800", gray.data, undF.cols * undF.rows);

        scanner.scan(image);

        for (Image::SymbolIterator symbol = image.symbol_begin(); symbol != image.symbol_end(); ++symbol)
        {
```

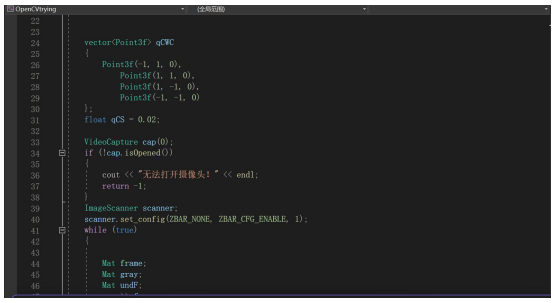
```
            vector<Point2f>
qrCC;
            for (int i = 0; i <
symbol->get_location_size(); i++)
            {

qrCC.emplace_back(symbol->get_
location_x(i),
symbol->get_location_y(i));
            }
            for (const Point2f&
corner : qrCC)
            {
                circle(frame,
corner, 3, Scalar(0, 255, 0), 2);
            }

            float qCPS =
norm(qrCC[0] - qrCC[1]);
            float qCSF = qCS /
qCPS;
            for (Point2f& corner :
qrCC)
            {
                corner *= qCSF;
            }

            Mat rvec, tvec;
            solvePnP(qCWC, qrCC,
cam, dist, rvec, tvec);

            cout << "平移向量: "
<< tvec << endl;
            cout << "旋转向量: "
<< rvec << endl;
        }

        imshow("摄像头", frame);
```

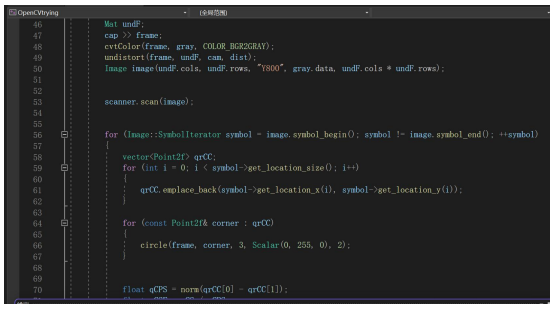
```
        if (waitKey(30) == 27)
            break;
    }
    cap.release();
    destroyAllWindows();

    return 0;
}
```

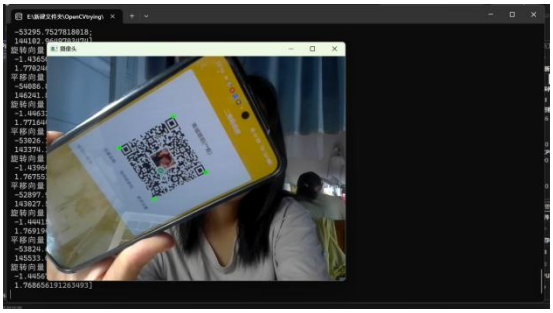












大概思路就是，首先要标定电脑上的摄像头得到内参和畸变矩阵，这个我就按 b 站教程【2023 年度最佳 Open Cv 学习教程，C++ 向 ！ - 哔哩哔哩】 https://b23.tv/MGkzb8n 跟着敲(后来知道 MATLAB 可以直接标定，哭死)。

然后位姿估计的时候也先参照了一下这个博主的教程视频，了解了 PnP 解算函数（宝藏博主）

我按 solvepnp 函数所需的参数来思考，还需要二维码角点在世界坐标系中的坐标和在图像坐标系下的坐标就可以算出平移和旋转矩阵。

世界坐标系的坐标好搞，可以自己设四个角点的坐标，我把二维码的中心设为原点，整个二维码设在 z 平面上，确定角点的世界坐标系的三维坐标后，再考虑怎么得到图像坐标系下的坐标。

图像坐标系需要调用能够识别二维码的库，开始想用 opencv/aruco 库，但是找不到清晰的下载介绍（哭），于是换了 zbar 库，参照 csdn 上一些检测和识别二维码内容的例子，先调用摄像头，再对图像进行灰度化和正畸，捕取帧，遍历图像找到二维码的四个角点(遍历找点这一步看了很多资料，最后用的是 zbar 里面的符号迭代器类），读取坐标。然后是根据前面量的二维码尺寸按比例缩放。这样就得到图像坐标系下的坐标。

再调用 pnp 函数，输入参数即可。

边学边玩，断断续续用了两个星期才搞完第一个😭