|  |
| --- |
| môn THỊ GIÁC MÁY – thầy nguyễn trọng tài |
| BÁO CÁO BÀI TẬP LỚN 1 |
| NHẬN DẠNG CHƠI CỜ CARO |
|  |
| **Bùi Anh Huy – 1411408** |
| **10/23/2017** |

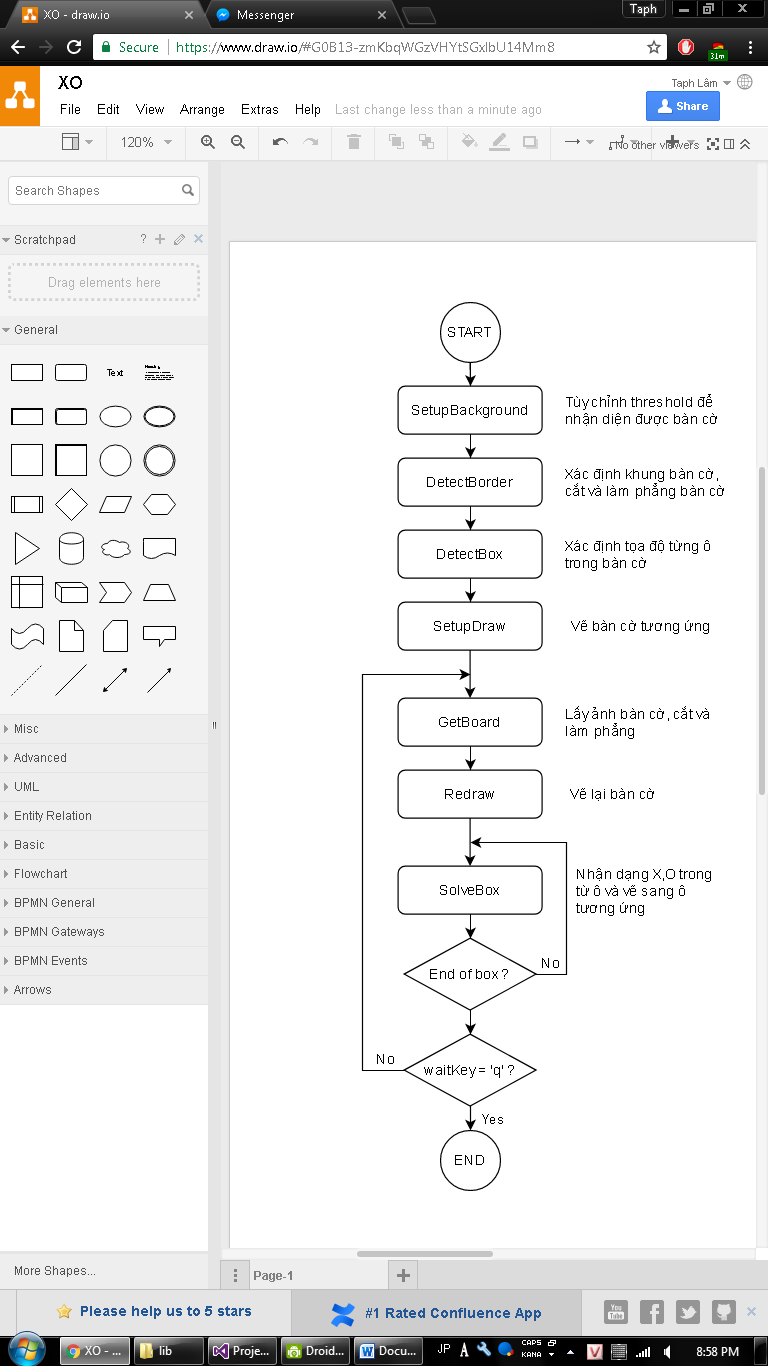
|  |
| --- |
|  |



1. **Giới thiệu đề tài**

Hệ thống sử dụng camera của điện thoại kết hợp với OpenCV để theo dõi và ghi nhận giá trị hiện tại của bàn cờ caro.

1. **Lưu đồ giải thuật**



1. **Đọc, nhận dạng bàn cờ.**
2. **SetupBackground:** Điều chỉnh threshold để xử lý bàn cờ

* Code:

*void SetupBackground() /\* Adjust threshold \*/*

*{*

*namedWindow("SetupBackground", WINDOW\_NORMAL);*

*createTrackbar("Threshold", "SetupBackground", &Threshold, 255);*

*while (waitKey(20) != 13) // If OK -> Enter*

*{*

*cvtColor(src, src\_g, CV\_BGR2GRAY);*

*medianBlur(src\_g, src\_g, 1);*

*threshold(src\_g, src\_b, Threshold, 255, THRESH\_BINARY);*

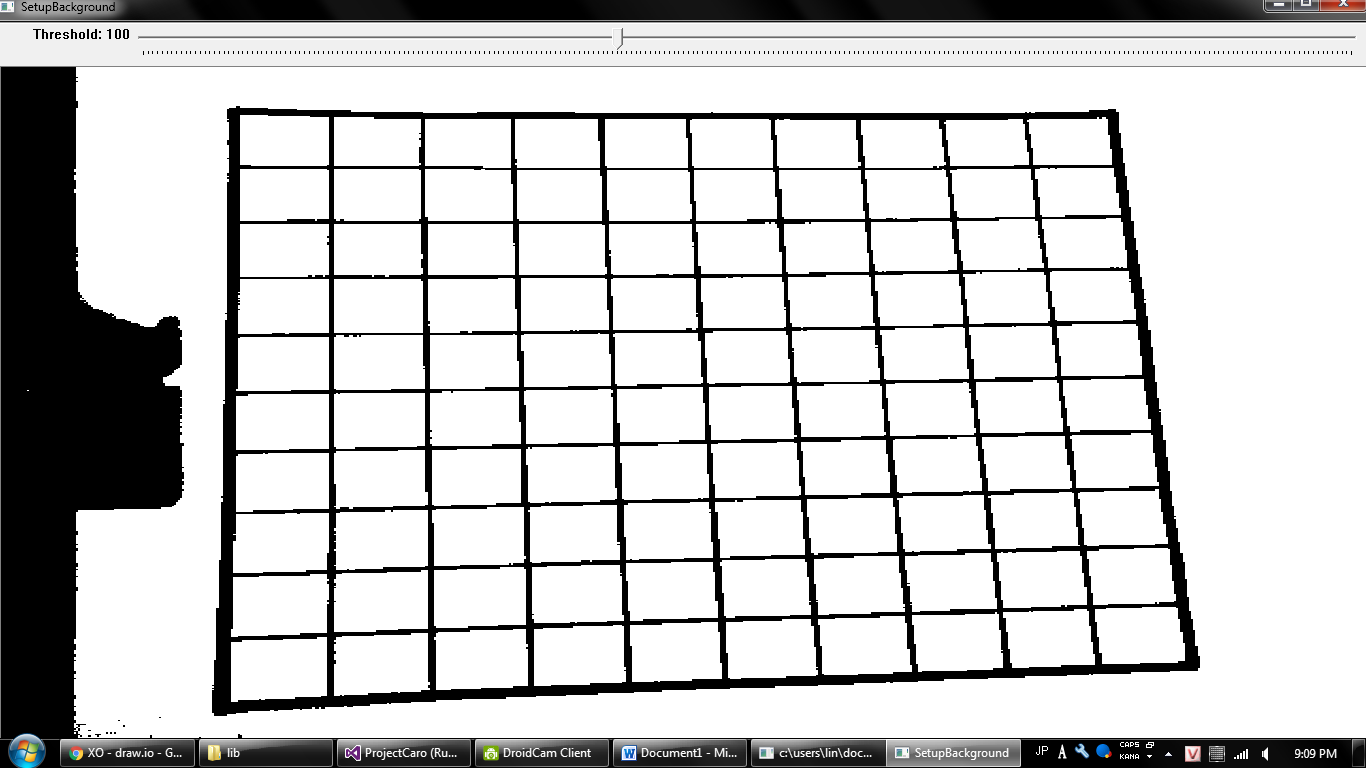
*imshow("SetupBackground", src\_b);*

*}*

*destroyWindow("SetupBackground");*

*}*

* Mục tiêu: chọn được threshold để tách biệt bàn cờ ra khỏi nền



1. **DetectBorder:** Xác định khung bàn cờ, cắt và làm phẳng

Mục tiêu: xác định đúng được khung viền và làm phẳng chính xác

1. Thuật toán xác định khung bàn cờ: dùng findContours để tìm các viền trong ảnh bàn cờ, sau đó dùng approxPolyDP để xác định hình tứ giác, sau đó dựa vào đường chéo lớn nhất của mỗi hình để tìm ra tứ giác lớn nhất, xác định tứ giác đó là khung bàn cờ

* Code

*void DetectBorder()*

*{*

*/\* Detect boder \*/*

*int diag, diag\_max = 0;*

*int diag\_src = sqrt(src.cols\*src.cols+ src.rows\*src.rows);*

*findContours(src\_b, contours, RETR\_LIST, CHAIN\_APPROX\_SIMPLE);*

*for (size\_t i = 0; i < contours.size(); i++)*

*{*

*approxPolyDP(contours[i], vertices, arcLength(contours[i], 1)\*0.02, 1);*

*if (vertices.size() == 4) //Kiểm tra tứ giác*

*{*

*diag = Diagonal(vertices);*

*if ((diag>diag\_max) && (diag<diag\_src - 20)) //Chọn tứ giác có kích thước lớn nhất*

*{*

*diag\_max = diag;*

*border = vertices;*

*}*

*}*

*}*

*line(src, border[0], border[1], cvScalar(255, 0, 0), 4);*

*line(src, border[1], border[2], cvScalar(0, 255, 0), 4);*

*line(src, border[2], border[3], cvScalar(0, 255, 0), 4);*

*line(src, border[3], border[0], cvScalar(0, 255, 255), 4);*

*namedWindow("DetectBorder", WINDOW\_NORMAL);*

*while (waitKey(50) != 13)*

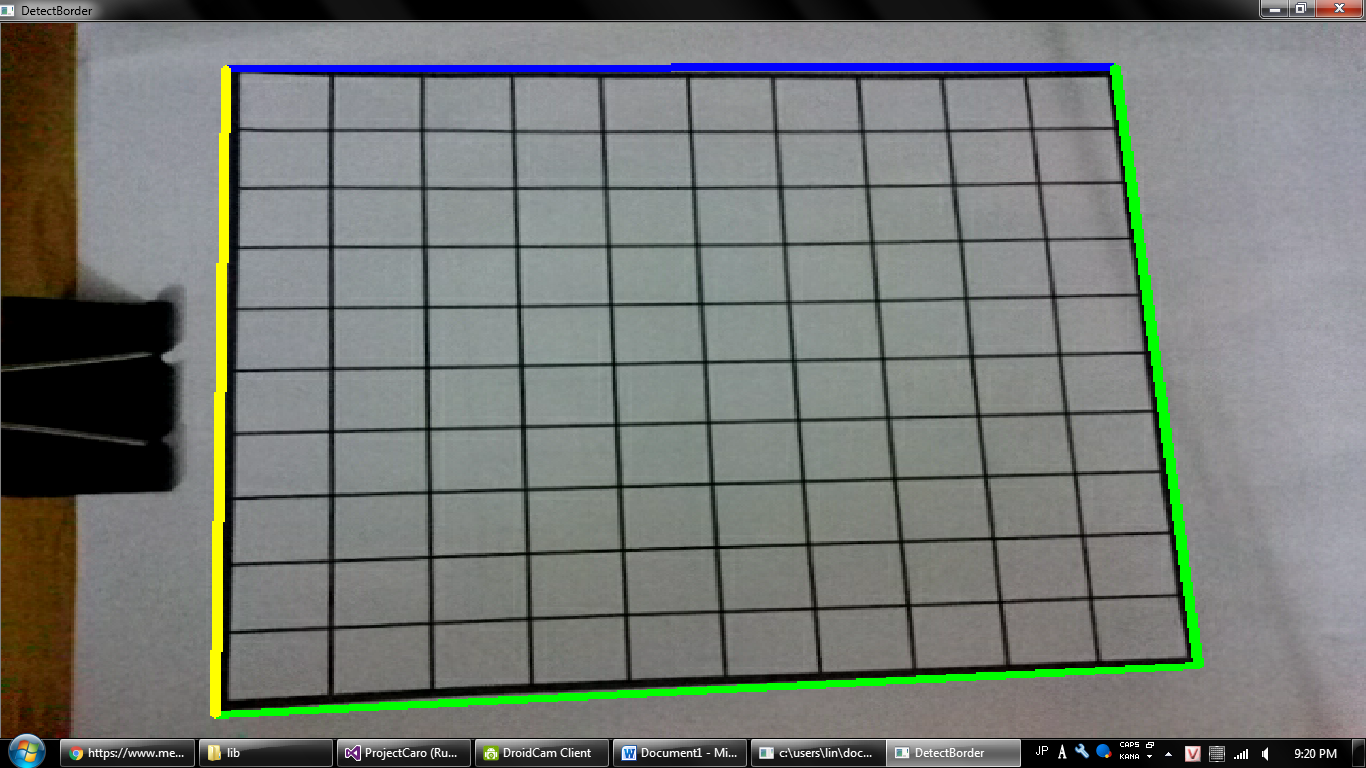
*{*

*imshow("DetectBorder", src);*

*}*

*destroyWindow("DetectBorder");*

*}*



1. Thuật toán cắt và làm phẳng bàn cờ: từ tọa độ của khung bàn cờ, ta xác định điểm cần kéo đến và dùng warpPerspective để cắt và làm phẳng bàn cờ.

*/\* Transform border \*/*

*Point2f inputQuad[4], outputQuad[4];*

*for (size\_t i = 0; i < 4; i++)*

*{*

*inputQuad[i] = Point2f(border[i]); //Xác định vị trí của 4 góc bàn cờ*

*if (inputQuad[i].x > src.cols / 2) outputQuad[i].x = src.cols;*

*else outputQuad[i].x = 0;*

*if (inputQuad[i].y > src.rows / 2) outputQuad[i].y = src.rows;*

*else outputQuad[i].y = 0;*

*}*

*for (size\_t i = 0; i < 4; i++)*

*{*

*for (size\_t j = 0; j < 4; i++)*

*{*

*if (i != j)*

*{ //Loại bỏ nếu có 2 góc trùng nhau*

*if ((outputQuad[i].x == outputQuad[j].x) && (outputQuad[i].y == outputQuad[j].y))*

*break;*

*}*

*}*

*}*

*M = getPerspectiveTransform(inputQuad, outputQuad);*

*warpPerspective(src, board, M, board.size());*



1. **Nhận dạng XO và vẽ lại.**
2. **DetectBox:** Xác định tọa độ từng ô trong bàn cờ

* Thuật toán xác định ô bàn cờ: sử dụng findContours và approxPolyDP tương tự như khi xác định khung bàn cờ
* Mục tiêu: xác định được tất cả các ô trong bàn cờ, đồng thời tính được số cột, số hàng để sử dụng về sau
* Code

*void DetectBox() // Detect number of boxs in board, in each col, in each row, box's width and box's height*

*{*

*int x,y;*

*cvtColor(board, board\_g, CV\_BGR2GRAY);*

*threshold(board\_g, board\_b, Threshold, 255, CV\_THRESH\_BINARY);*

*namedWindow("DetectBo", WINDOW\_NORMAL);*

*imshow("DetectBo", board\_g);*

*findContours(board\_b, contours, CV\_RETR\_LIST, CV\_CHAIN\_APPROX\_SIMPLE);*

*for (size\_t i = 0; i < contours.size(); i++)*

*{*

*approxPolyDP(contours[i], vertices, arcLength(contours[i], 1)\*0.02, 1);*

*if (vertices.size() == 4)*

*{*

*x = abs(vertices[0].x - vertices[2].x);*

*y = abs(vertices[0].y - vertices[2].y);*

*if ((x>5) && (y > 5) && (x<board.cols - 10) && (y<board.rows - 10))*

*{*

*box[box\_count] = vertices;*

*box\_count++;*

*sum\_x += x;*

*sum\_y += y;*

*line(board, vertices[0], vertices[1], cvScalar(255, 255, 0), 4);*

*line(board, vertices[1], vertices[2], cvScalar(0, 255, 255), 4);*

*line(board, vertices[2], vertices[3], cvScalar(0, 255, 255), 4);*

*line(board, vertices[3], vertices[0], cvScalar(100, 255, 255), 4);*

*}*

*}*

*}*

*box\_w = (float)(sum\_y + calibLine) / box\_count;*

*box\_h = (float)(sum\_x + calibLine) / box\_count;*

*board\_rows = board.rows / box\_w;*

*board\_cols = board.cols / box\_h;*

*namedWindow("DetectBox", WINDOW\_NORMAL);*

*while (waitKey(50) != 13)*

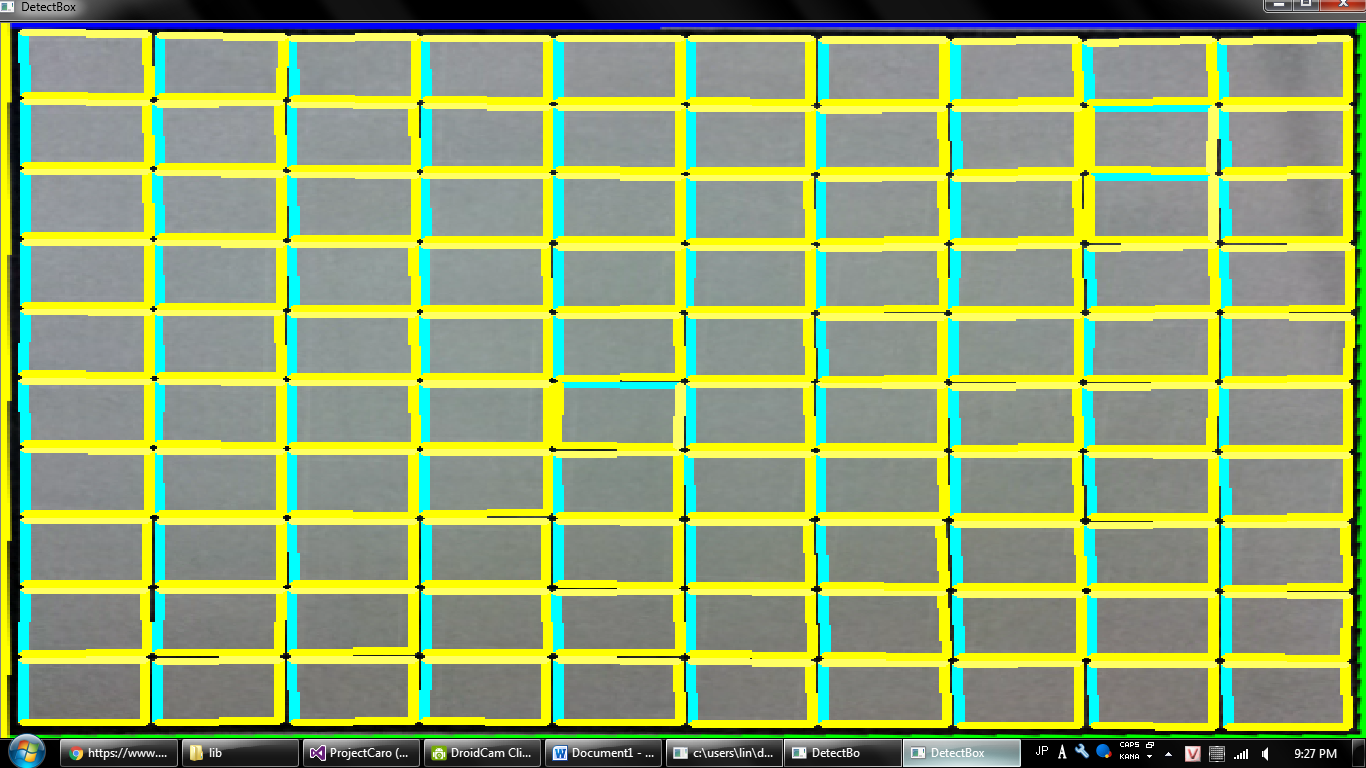
*{*

*imshow("DetectBox", board);*

*}*

*destroyWindow("DetectBox");*

*}*



1. **SetupDraw:** Vẽ bàn cờ tương ứng

Từ số hàng, số cột, ta vẽ lại bàn cờ mới với mỗi ô vuông tương ứng 50 pixel. Do có sai số do đường kẻ có độ dày mỏng không giống nhau nên tại đây có trackbar để điều chỉnh bàn cờ cho đúng.

* Code

*void SetupDraw()*

*{*

*drawX = imread("X.jpg",CV\_LOAD\_IMAGE\_GRAYSCALE);*

*drawO = imread("O.jpg",CV\_LOAD\_IMAGE\_GRAYSCALE);*

*namedWindow("Draw", WINDOW\_AUTOSIZE);*

*createTrackbar("Calib line", "Draw", &calibLine, 100);*

*createTrackbar("Level X", "Draw", &level1, 300);*

*createTrackbar("Level O", "Draw", &level2, 300);*

*while (waitKey(20) != 13) // If OK -> Enter*

*{*

*board\_rows = board.rows / (box\_w + calibLine);*

*board\_cols = board.cols / (box\_h + calibLine);*

*draw = Mat::ones(board\_rows \* 50, board\_cols \* 50, CV\_8U)\*255.;*

*for (int i = 1; i < board\_cols; i++)*

*line(draw, Point2f(i\*(draw.cols / board\_cols), 0), Point2f(i\*(draw.cols / board\_cols), draw.rows), (0, 0, 0), 5, 8, 0);*

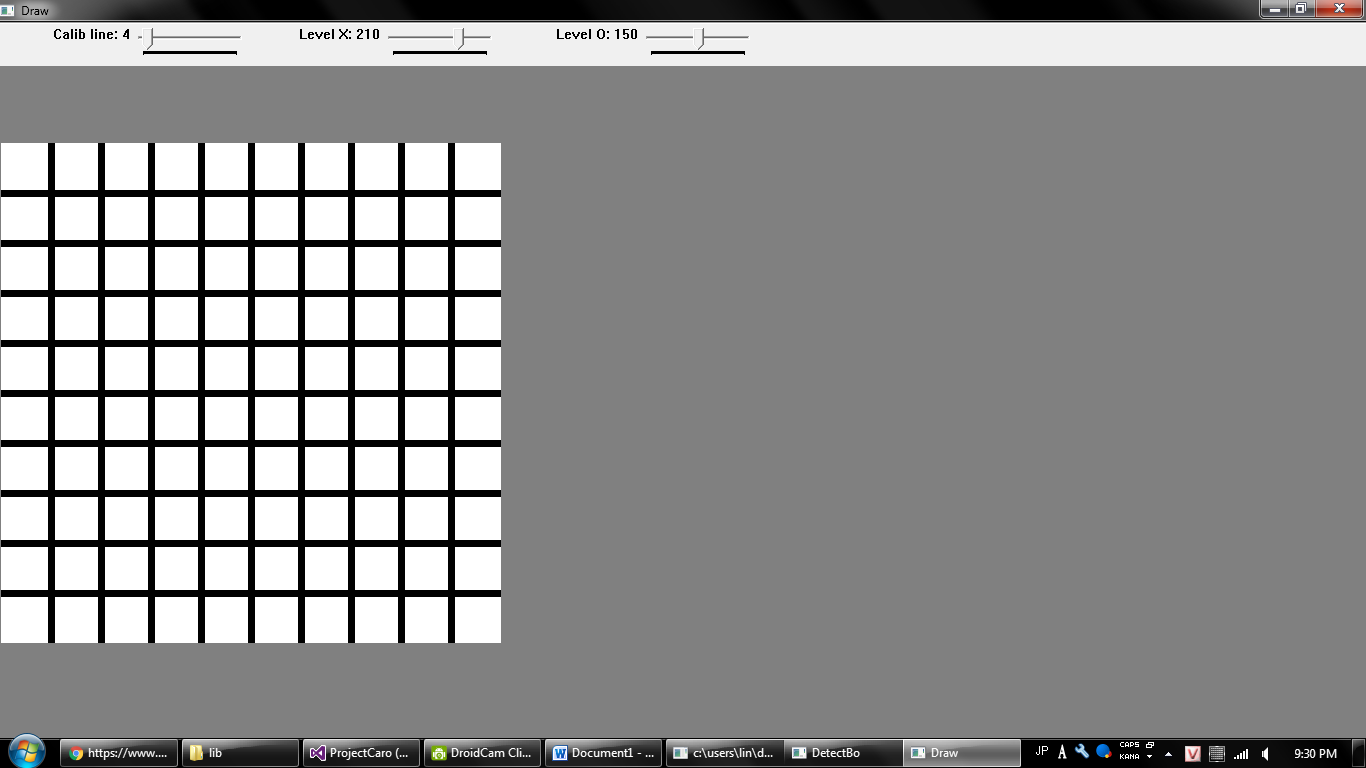
*for (int i = 1; i < board\_rows; i++)*

*line(draw, Point2f(0, i\*(draw.rows / board\_rows)), Point2f(draw.cols, i\*(draw.rows / board\_rows)), (0, 0, 0), 5, 8, 0);*

*imshow("Draw", draw);*

*}*

*}*



1. **SolveBox():** Nhận dạng X, O trong ô trống

* Sau khi xác định được toa độ 4 góc của mỗi ô trong bàn cờ, tạo ma trận nhớ tạm chứa dữ liệu hình *ảnh trong 1 ô của bàn cờ với toạ độ 4 góc.*

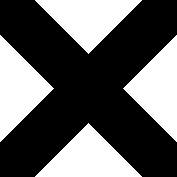
*Mat temp = board(Range(yMin, yMax), Range(xMin, xMax));*

*cvtColor(temp, temp, CV\_BGR2GRAY);*

* Xác định vị trí ô được lấy bằng cách tính toạ độ x và y trung điểm của 2 cạnh

*xMed = (float)(xMax + xMin) / 2;*

*yMed = (float)(yMax + yMin) / 2;*

* Xác định X, O
* Thuật toán: Sau khi tách được từng ô trong bàn cờ, ta đem xác định X, O hoặc không có gì
* Đếm số pixel màu đen của hình. Chọn và thay đổi một ngưỡng pixel xác định, nếu dưới ngưỡng này, kết luận ô trống.
* Sử dụng mặt nạ X, O đè lên hình (cộng ma trận).   
    
   
* Đếm số pixel còn lại.
* Tính tỉ lệ pixel sau/ đầu. So sánh tỉ lệ giữa mask X và mask O. Nếu tỉ lệ của (hình + mask X)> (hình + mask O) thì nhận dạng là X và ngược lại.
* Code

*int findXO(Mat img)*

*{*

*int rec = -1; //biến kiểm tra X (rec = 1), O (rec = 0), blank (rec = -1)*

*float t = (float)findblack(img) / (img.cols\*img.rows) - (float)level1/1000;*

*if (t>0)*

*{*

*int check = 0;*

*Mat modelx = imread("xmask1.jpg", CV\_8UC1); //load maskX*

*Mat modelo = imread("omask.jpg", CV\_8UC1); //load maskO*

*resize(modelx, modelx, Size(img.cols, img.rows), 1, 1, INTER\_LINEAR);*

*resize(modelo, modelo, Size(img.cols, img.rows), 1, 1, INTER\_LINEAR);*

*//float initx = findblack(modelx);*

*float init = findblack(img); //Tim so pixel mau den trong hinh ban dau*

*modelx = modelx + img; //đè mask lên hình*

*modelo = modelo + img;*

*float kx = findblack(modelx) / init; //tỉ lệ số pixel đen sau khi đưa qua mask*

*float ko = findblack(modelo) / init;*

*if (kx > ko) rec = 1;*

*else if (kx < ko) rec = 0;*

*}*

*else rec = -1;*

*return rec;*

*}*

* Vẽ X hoặc O vào vị trí đã nhận diện lên bàn cờ đã khởi tạo.
* Code

*void SolveBox(vector<Point> boxA)*

*{*

*box\_w = board.rows / board\_rows;*

*box\_h = board.cols / board\_cols;*

*float xMed, yMed;*

*//Đánh dấu ô đã nhận diện trên bàn cờ*

*line(board, boxA[0], boxA[1], cvScalar(255, 0, 0), 4);*

*line(board, boxA[1], boxA[2], cvScalar(0, 255, 0), 4);*

*line(board, boxA[2], boxA[3], cvScalar(0, 255, 0), 4);*

*line(board, boxA[3], boxA[0], cvScalar(0, 255, 255), 4);*

*int xMax = boxA[2].x, xMin = boxA[0].x;*

*int yMax = boxA[2].y, yMin = boxA[0].y;*

*if (boxA[0].x > boxA[2].x)*

*{*

*xMax = boxA[0].x;*

*xMin = boxA[2].x;*

*}*

*if (boxA[0].y > boxA[2].y)*

*{*

*yMax = boxA[0].y;*

*yMin = boxA[2].y;*

*}*

*Mat temp = board(Range(yMin, yMax), Range(xMin, xMax));*

*cvtColor(temp, temp, CV\_BGR2GRAY);*

*//Xác định vị trí ô đang xét*

*xMed = (float)(xMax + xMin) / 2;*

*yMed = (float)(yMax + yMin) / 2;*

*if ((yMed / box\_w < board\_rows) && (xMed / box\_h < board\_cols))*

*{*

*Mat temp\_draw = draw(Range(floor(yMed / box\_w) \* 50, (floor(yMed / box\_w) + 1) \* 50), Range(floor(xMed / box\_h) \* 50, (floor(xMed / box\_h) + 1) \* 50));*

*flag = findXO(temp); //Nhận dạng X, O*

*// Vẽ lên bàn cờ khởi tạo nếu là X hoặc O*

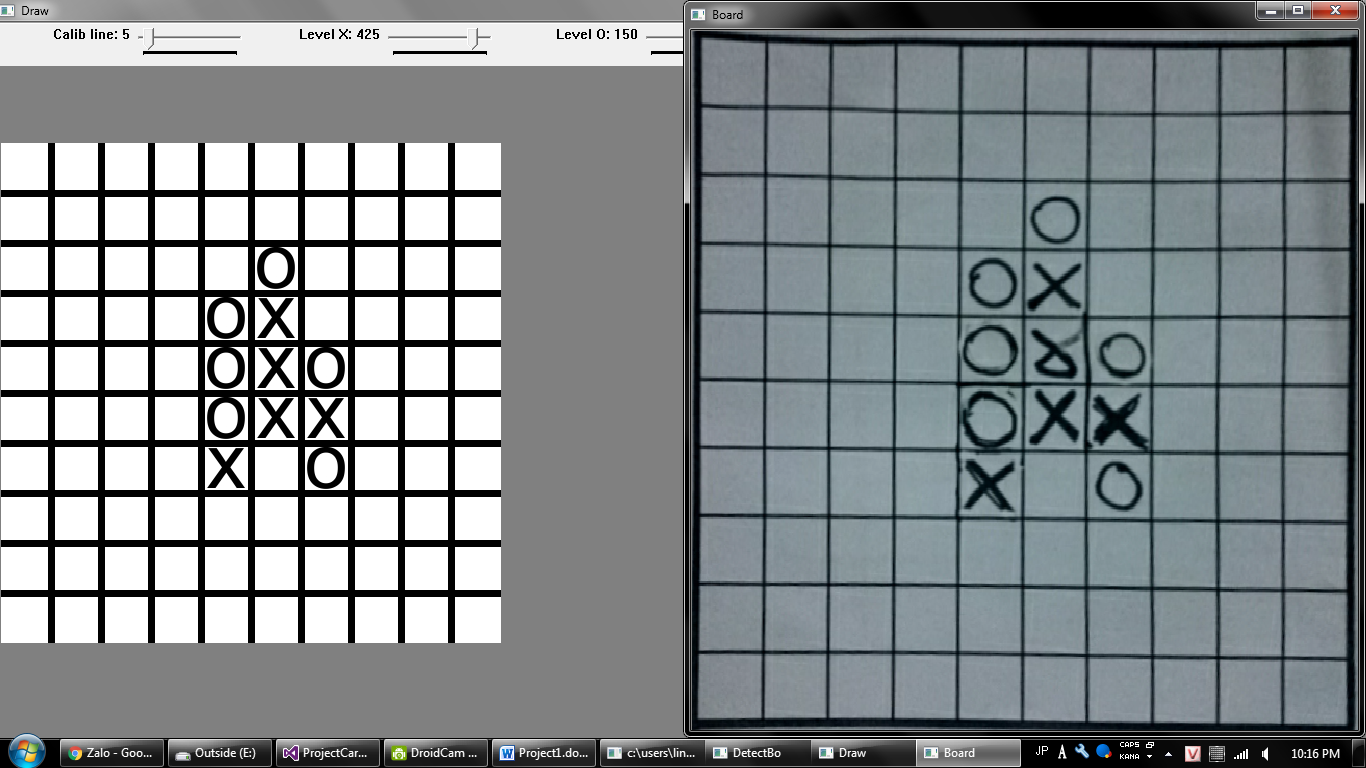
*if (flag == 1) bitwise\_and(temp\_draw, drawX, temp\_draw);*

*else if (flag == 0) bitwise\_and(temp\_draw, drawO, temp\_draw);*

*}*

*}*

1. **Kết quả thực tế**



1. **Nhận xét**

Hệ thống phụ thuộc khá nhiều vào điều kiện ánh sáng, độ tương phản, đậm nhạt của đường kẻ.

Hệ thống hiện tại có thể theo dõi kịp thời các nước đi của người chơi nhưng chưa thể nhận diện được người chơi nào chiến thắng.