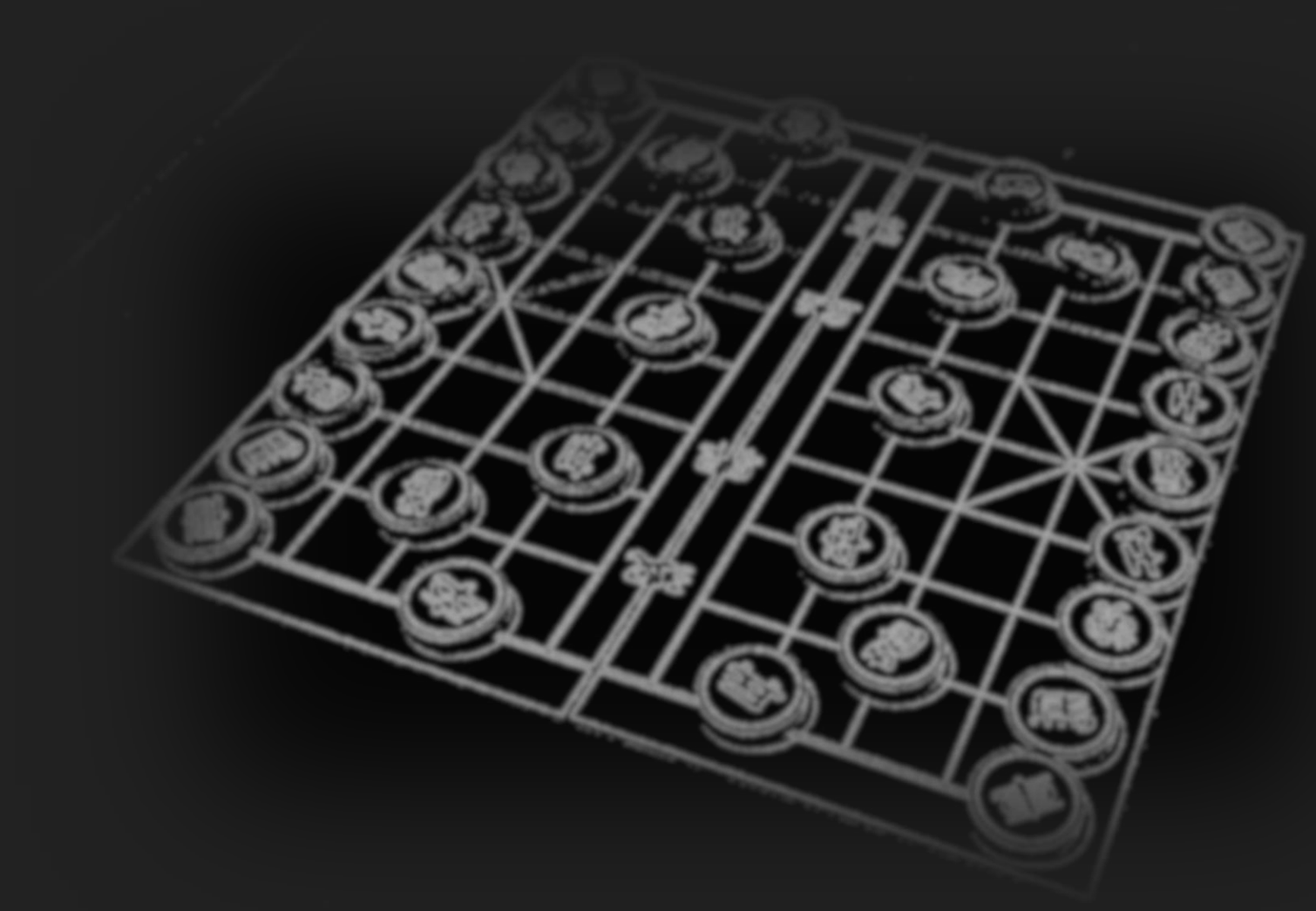


基于MATLAB的

象棋棋盘及棋子识别

张凌霄 邱子濛 郑睦炜

CHESS BOARD AND CHESS PIECES IDENTIFICATION

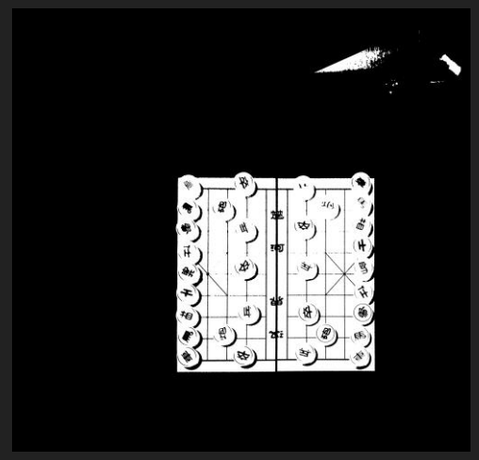
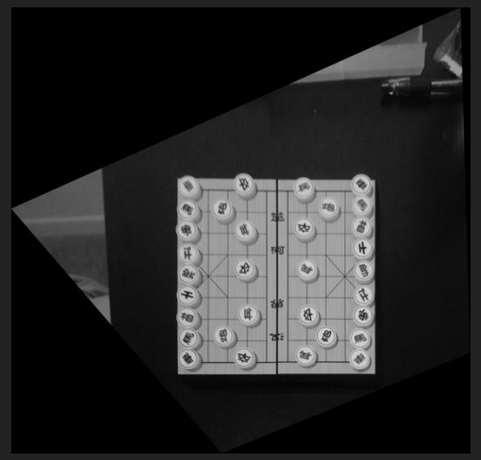
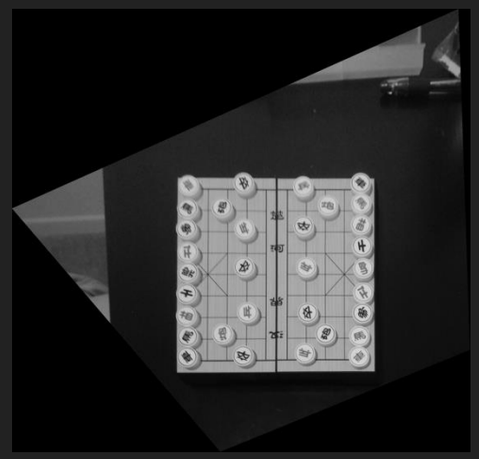
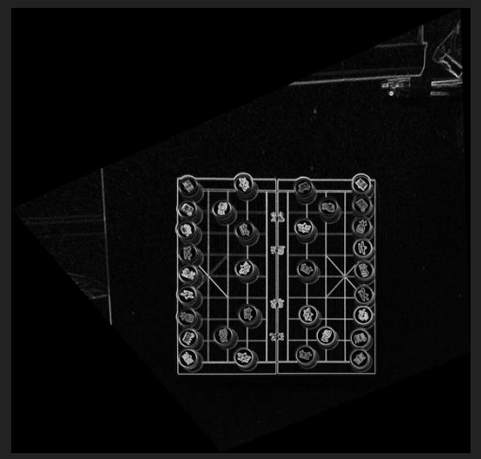
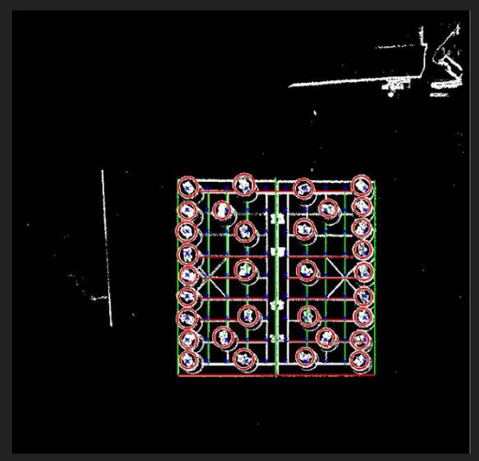
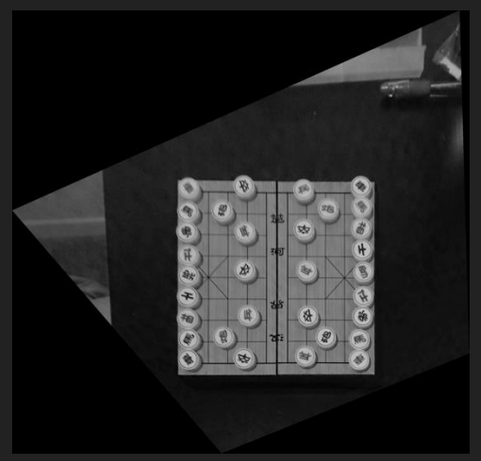
According to the real-time board photo taken (e.g. Test.png), and the previously known standard chess board(e.g. Ref.png), automatic positioning and output coordinates of all the pieces of both the red side and the black side.

Input: 2 pics
Output: 2 sets of coordinates

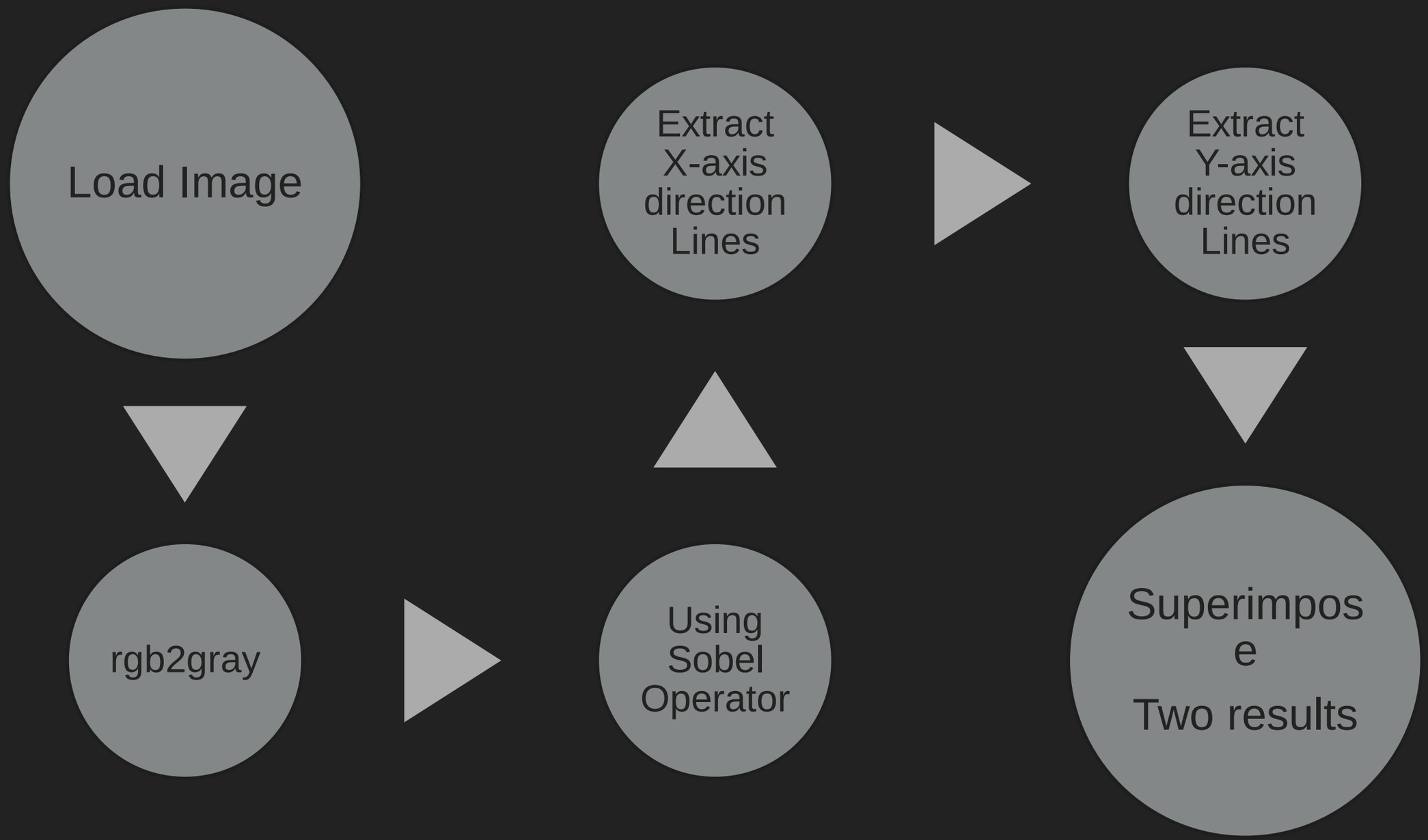
Demand of project

PROCESSING STEPS

- Edge Detection
- Perspective Transformation
- Image Quality Optimize
- Line detection
- Position Chess Pieces
- Identify Pieces’ color
- Identify Pieces’ Character

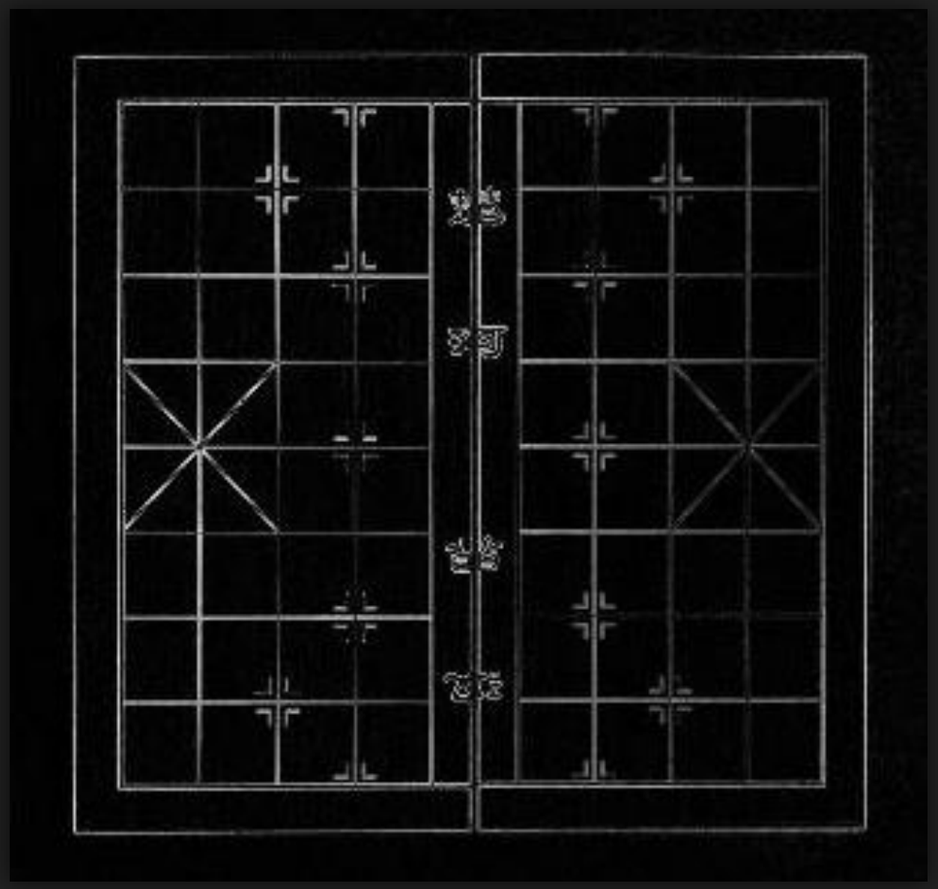







STEP.1 EDGE DETECTION



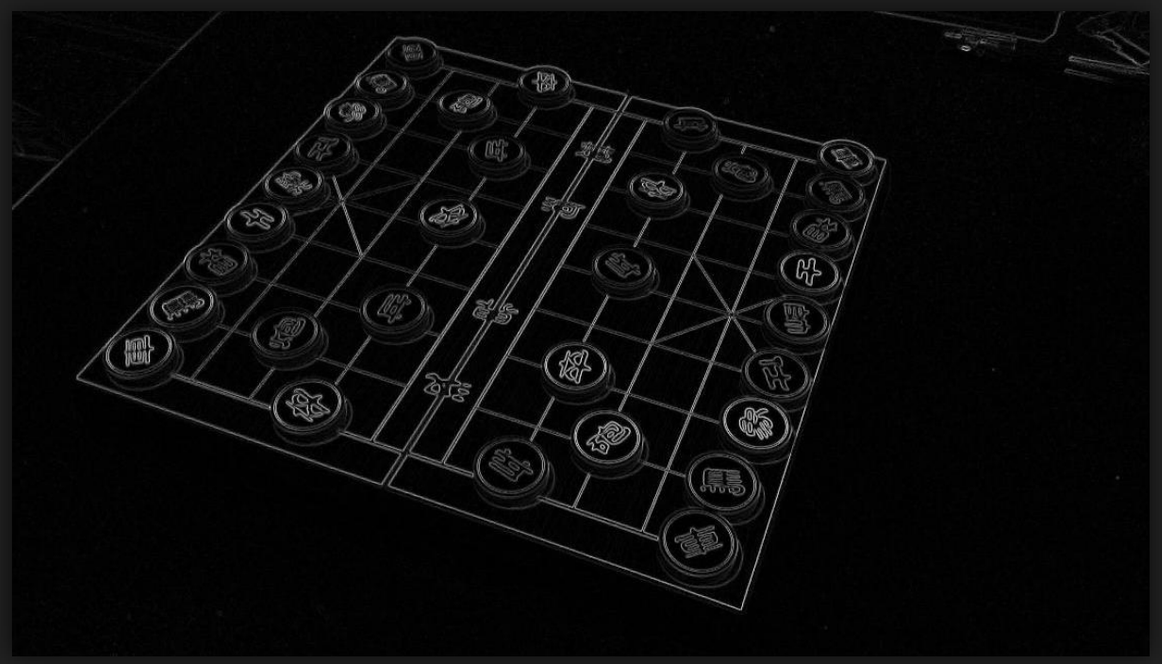

```
sobelKernelY = [ 1 2 1; 0 0 0; -1 -2 -1]

sobelKernelX = [-1 0 1; -2 0 2; -1 0 1]
```

STEP.1 EDGE DETECTION



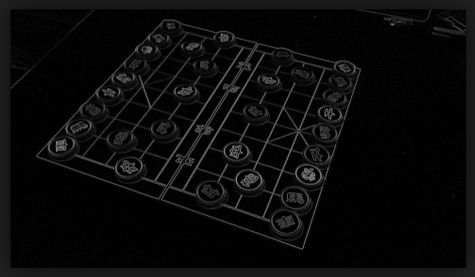
Reference chess board

Test input chess board

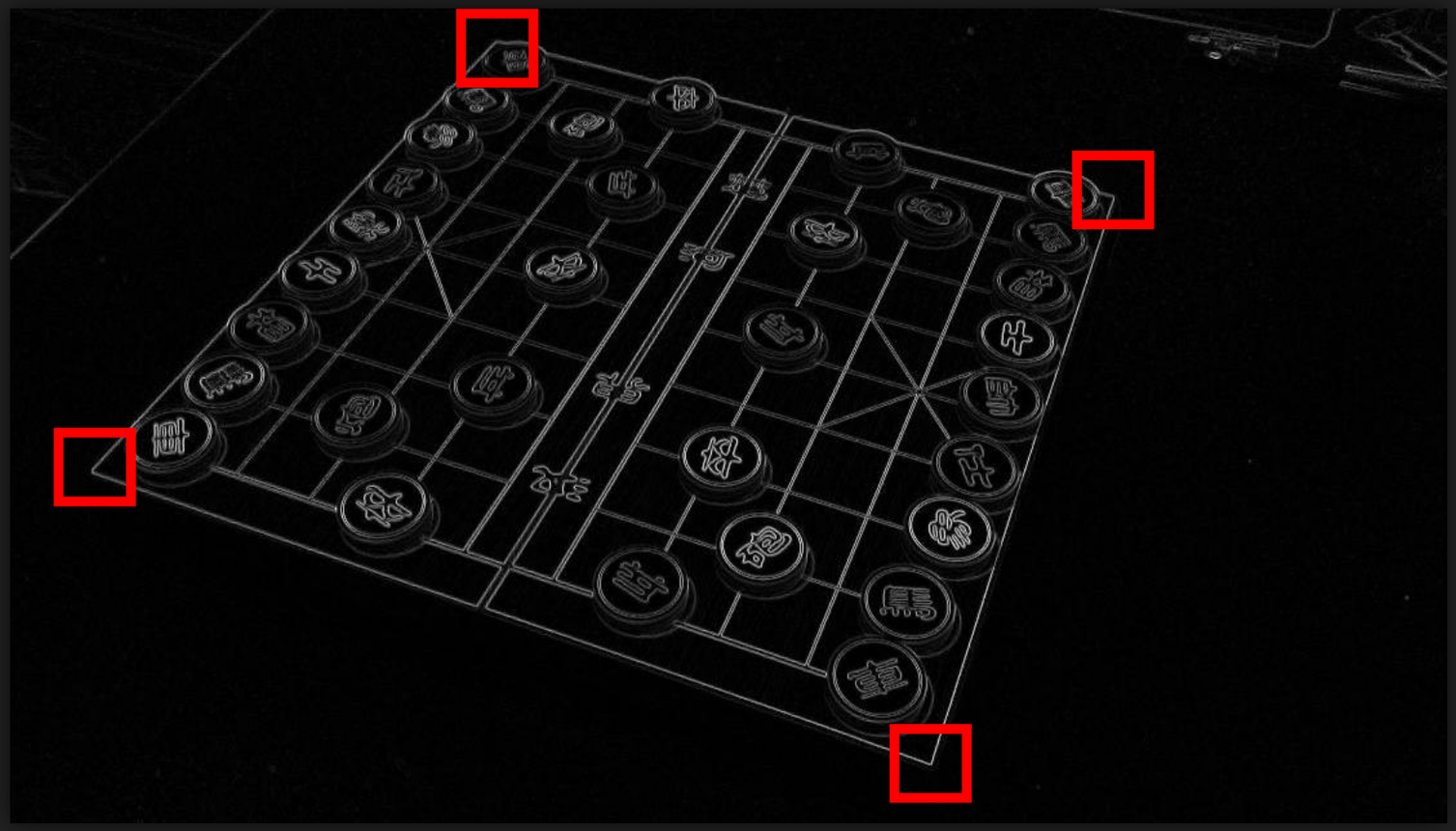


STEP.2 PERSPECTIVE TRANSFORMATION

Auto-locate the 4 corners of test image



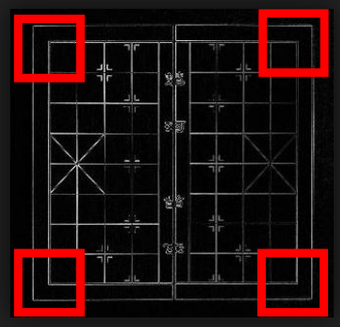
STEP.2 PERSPECTIVE TRANSFORMATION

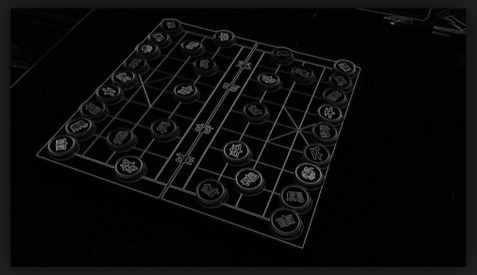


Line Detection → Find the top 4 longest lines → Their endpoints are 4 corners of board

STEP.2 PERSPECTIVE TRANSFORMATION

Auto-locate the 4 corners of test image

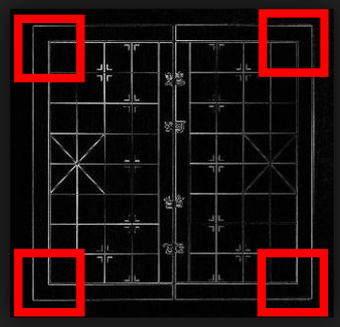




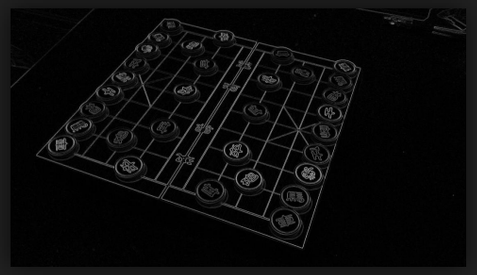
Using 8 corners from Ref and Test image as fixed points

STEP.2 PERSPECTIVE TRANSFORMATION

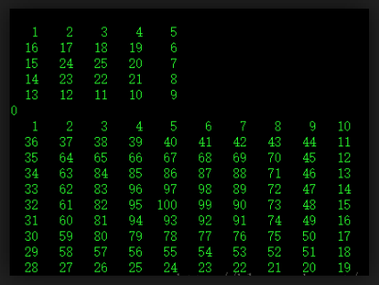
Auto-locate the 4 corners of test image



Calculate transform matrix with the 8 fixed points



Using 8 corners from Ref and Test image as fixed points



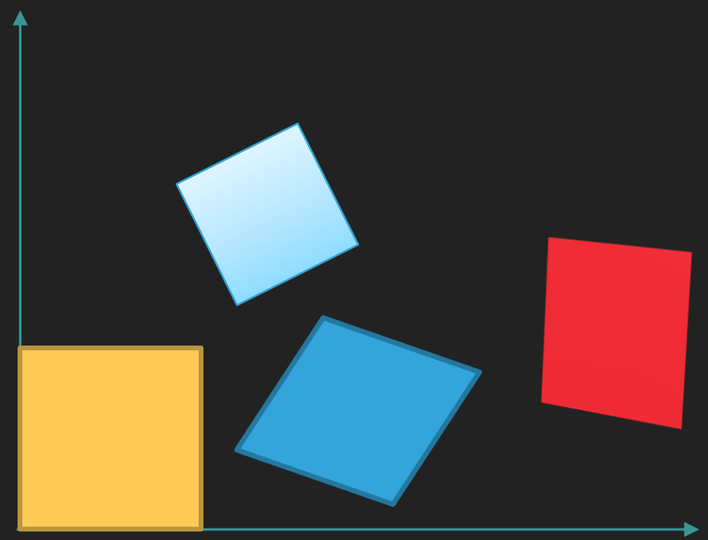
STEP.2 PERSPECTIVE TRANSFORMATION

$$(x'\ y'\ w') = (u\ v\ w)\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

(u, v) refers to the original coordinate, and
(x, y) refers to the transformed coordinate.
Where $x = \frac{x'}{w'}$, $y = \frac{y'}{w'}$.

We can split this transform matrix $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ into 4 parts:

$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ denotes the linear transform,
e.g. scaling, shearing and rotation.



STEP.2 PERSPECTIVE TRANSFORMATION

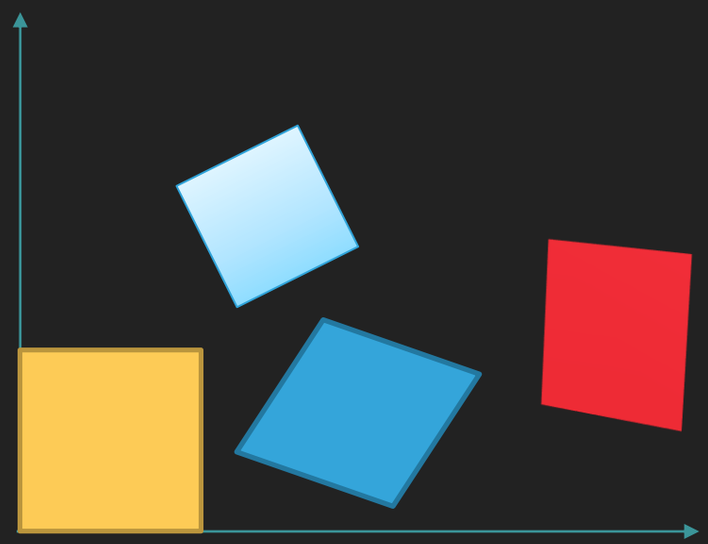
$$(x'\ y'\ w') = (u\ v\ w)\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$(a_{31} \quad a_{32})$ is applied for translation.
$(a_{13} \quad a_{23})^T$ is applied for affine transform.
Then we can obtain,

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}}$$

$$y = \frac{y'}{w'} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}}$$

Also if we know the original and transformed coordinates, it is easy to calculate the transform matrix.



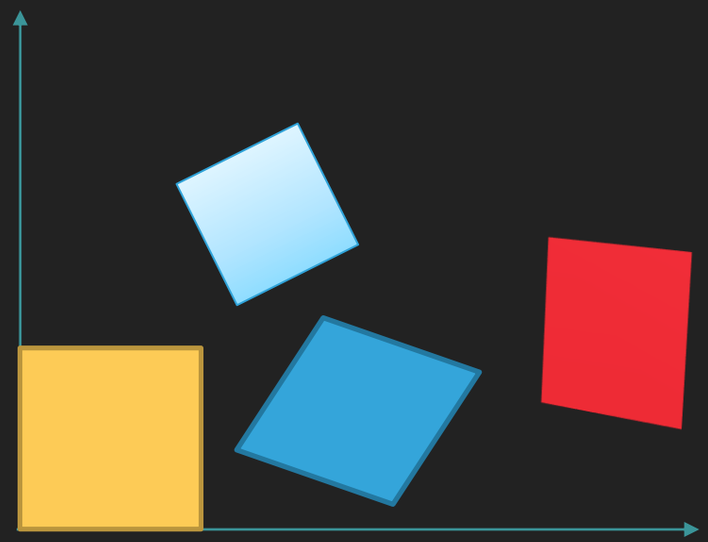
STEP.2 PERSPECTIVE TRANSFORMATION

Assuming the coordinates before and after the transformation like

$$(CornerRefX_1, CornerRefY_1)$$
$$-> (CornerTestX_1, CornerTestY_1)$$
$$\dots$$
$$(CornerRefX_4, CornerRefY_4)$$
$$-> (CornerTestX_4, CornerTestY_4)$$

According to transform formula, we have

$$a_{31} = x_1$$
$$a_{11} + a_{31} - a_{13}x_2 = x_2$$
$$a_{11} + a_{21} + a_{31} - a_{13}x_3 - a_{23}x_3 = x_3$$
$$a_{12} + a_{31} - a_{23}x_4 = x_4$$
$$\dots$$
$$a_{22} + a_{32} - a_{23}y_4 = y_4$$



Therefore,

$$a_{11} = x_2 - x_1 + a_{12}x_2 \quad a_{12} = y_2 - y_1 + a_{13}y_2$$
$$a_{21} = x_4 - x_1 + a_{12}x_3 \quad a_{22} = y_4 - y_1 + a_{13}y_3$$
$$a_{31} = x_0 \quad a_{32} = y_0$$

$$a_{12} = \frac{\begin{vmatrix} \Delta x_1 & \Delta x_3 \\ \Delta y_1 & \Delta y_3 \end{vmatrix}}{\begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix}}$$

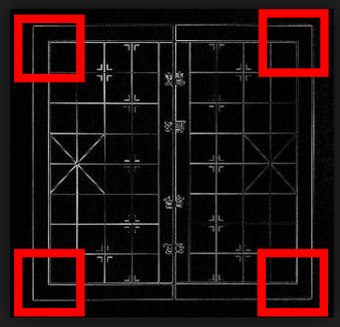
$$a_{13} = \frac{\begin{vmatrix} \Delta x_3 & \Delta x_2 \\ \Delta y_3 & \Delta y_2 \end{vmatrix}}{\begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix}}$$

Where,

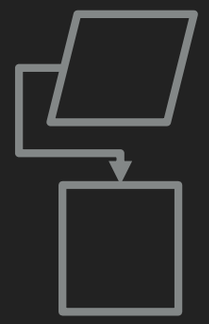
$$\Delta x_1 = x_2 - x_3 \; \Delta x_2 = x_4 - x_3 \; \Delta x_3 = x_1 - x_2 + x_3 - x_4$$
$$\Delta y_1 = y_2 - y_3 \; \Delta y_2 = y_4 - y_3 \; \Delta y_3 = y_1 - y_2 + y_3 - y_4$$

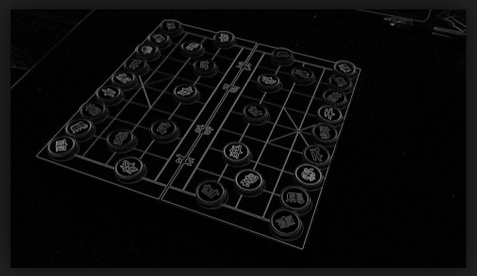
STEP.2 PERSPECTIVE TRANSFORMATION

Auto-locate the 4 corners of test image

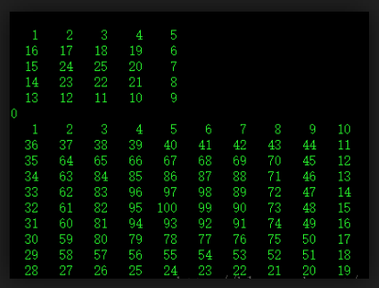


Calculate transform matrix with the 8 fixed points





Using 8 corners from Ref and Test image as fixed points

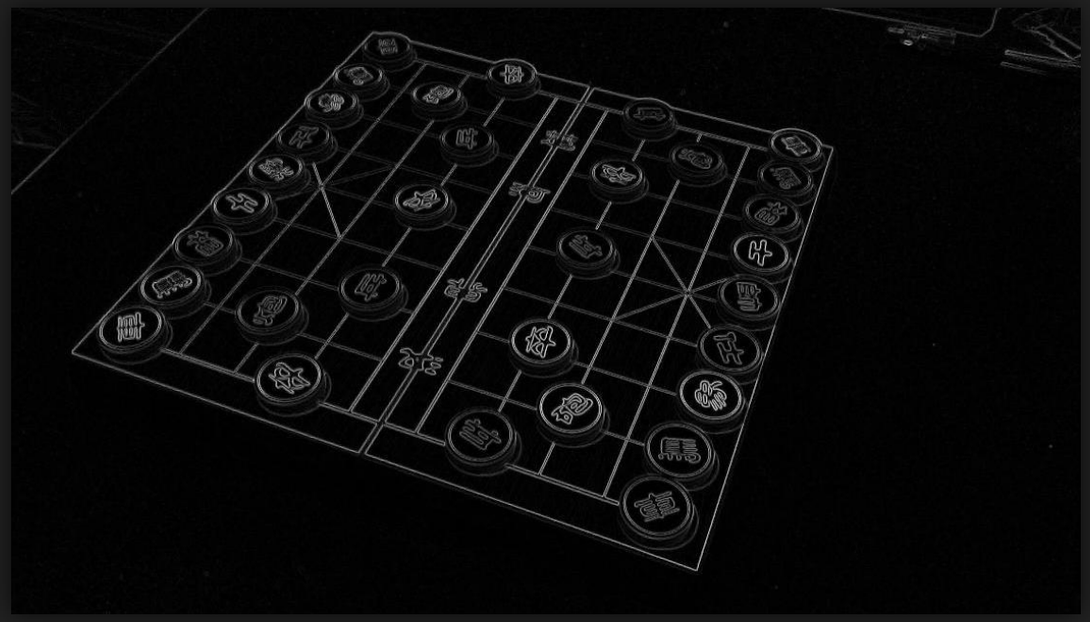


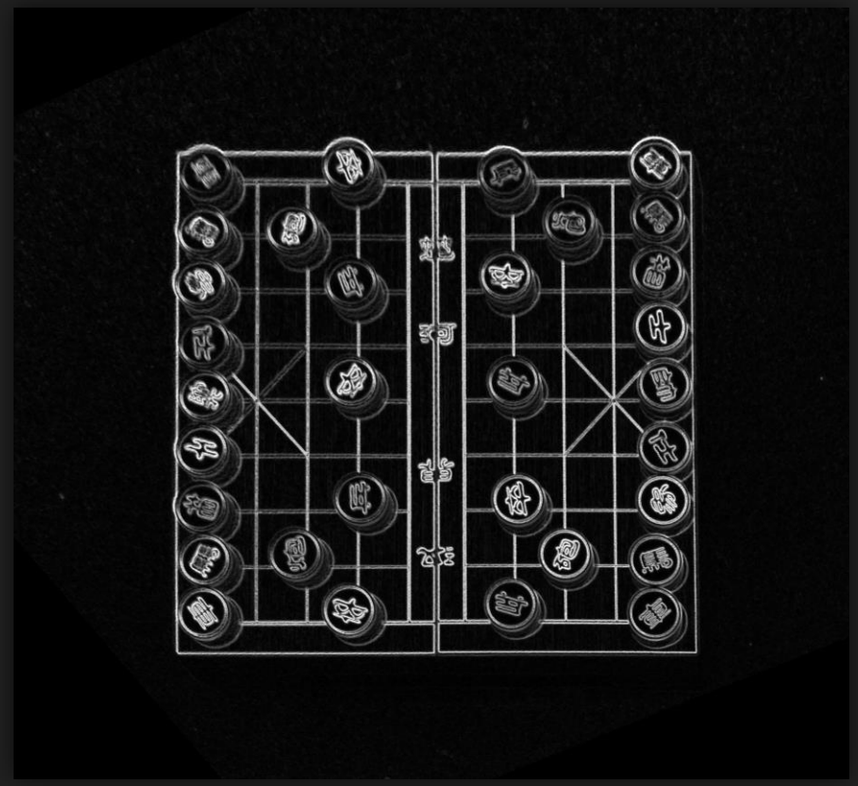
Transform a leaning image into a straight one

STEP.2 PERSPECTIVE TRANSFORMATION

$$(x'\ y'\ w') = (u\ v\ w)\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

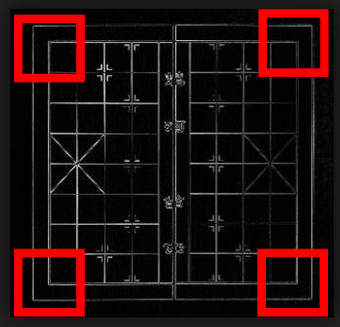
$$x = \frac{x'}{w'}\ ,\ y = \frac{y'}{w'}\ .$$



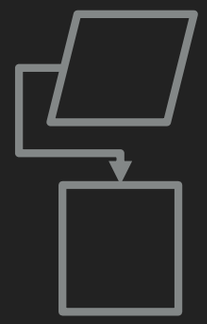


STEP.2 PERSPECTIVE TRANSFORMATION

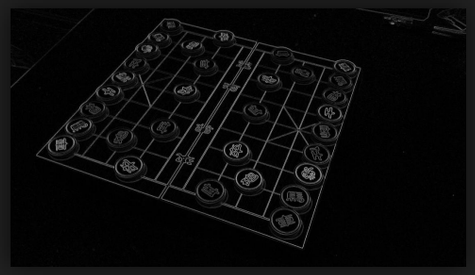
Auto-locate the 4 corners of test image



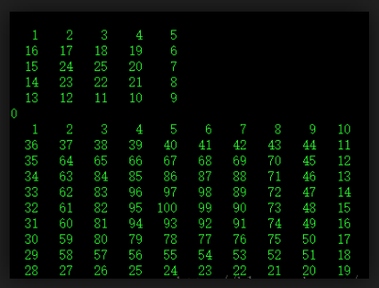
Calculate transform matrix with the 8 fixed points



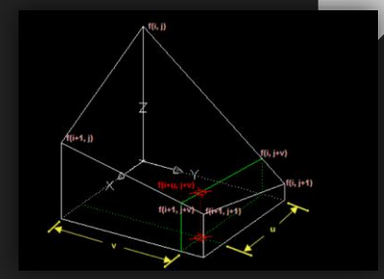
Bilinear inter-polation



Using 8 corners from Ref and Test image as fixed points

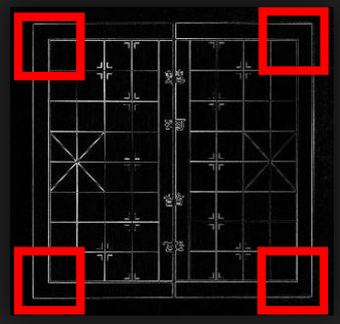


Transform a leaning image into a straight one

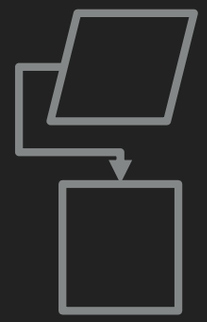


STEP.2 PERSPECTIVE TRANSFORMATION

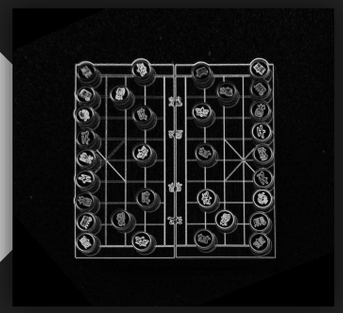
Auto-locate the 4 corners of test image

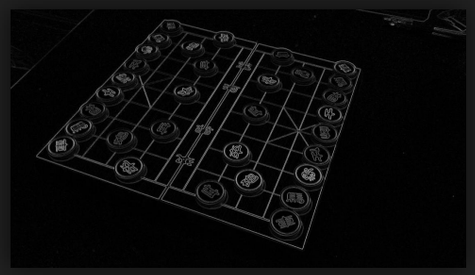


Calculate transform matrix with the 8 fixed points

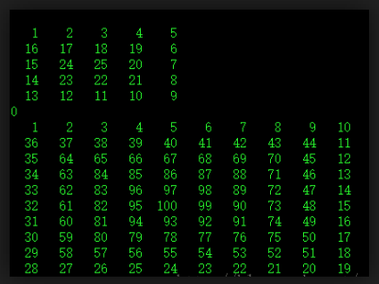


Bilinear inter-polation



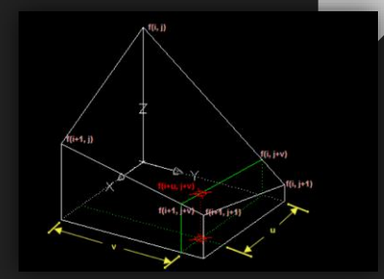


Using 8 corners from Ref and Test image as fixed points

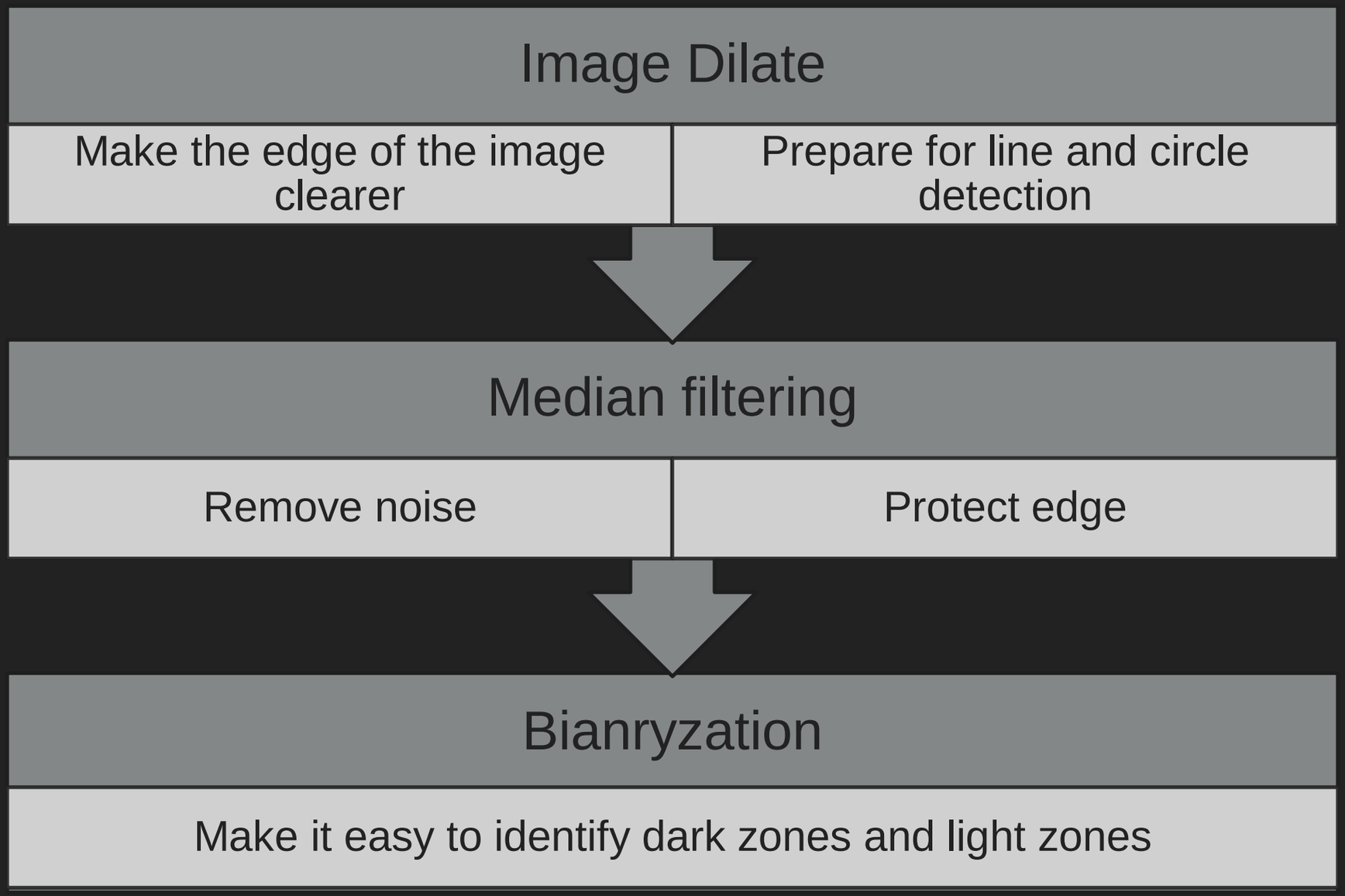


	1	2	3	4	5					
	16	17	18	19	6					
	15	24	25	20	7					
	14	23	22	21	8					
	13	12	11	10	9					
0										
	1	2	3	4	5	6	7	8	9	10
	36	37	38	39	40	41	42	43	44	11
	35	64	65	66	67	68	69	70	45	12
	34	63	84	85	86	87	88	71	46	13
	33	62	83	96	97	98	89	72	47	14
	32	61	82	95	100	99	90	73	48	15
	31	60	81	94	93	92	91	74	49	16
	30	59	80	79	78	77	76	75	50	17
	29	58	57	56	55	54	53	52	51	18
	28	27	26	25	24	23	22	21	20	19

Transform a leaning image into a straight one

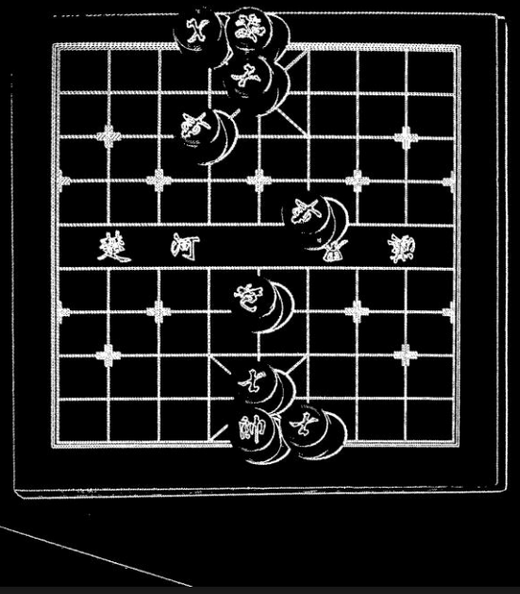


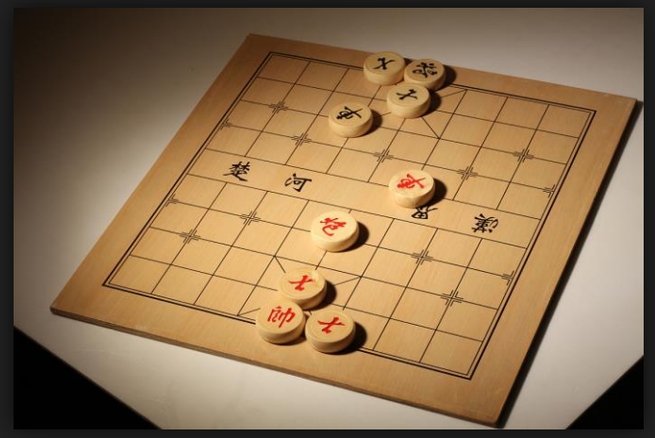
STEP.3 IMAGE QUALITY OPTIMIZE



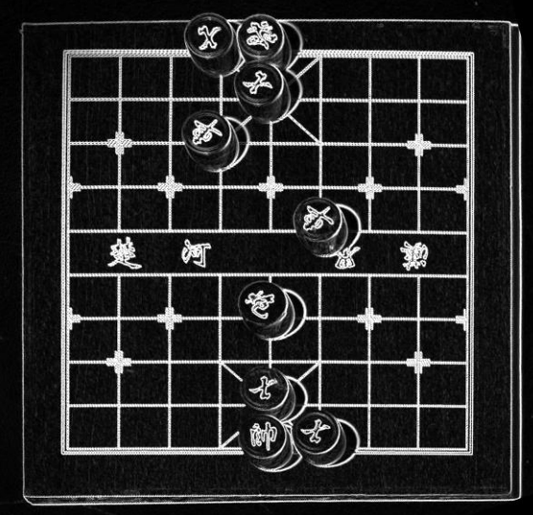
STEP.3 IMAGE QUALITY OPTIMIZE

Before optimizing

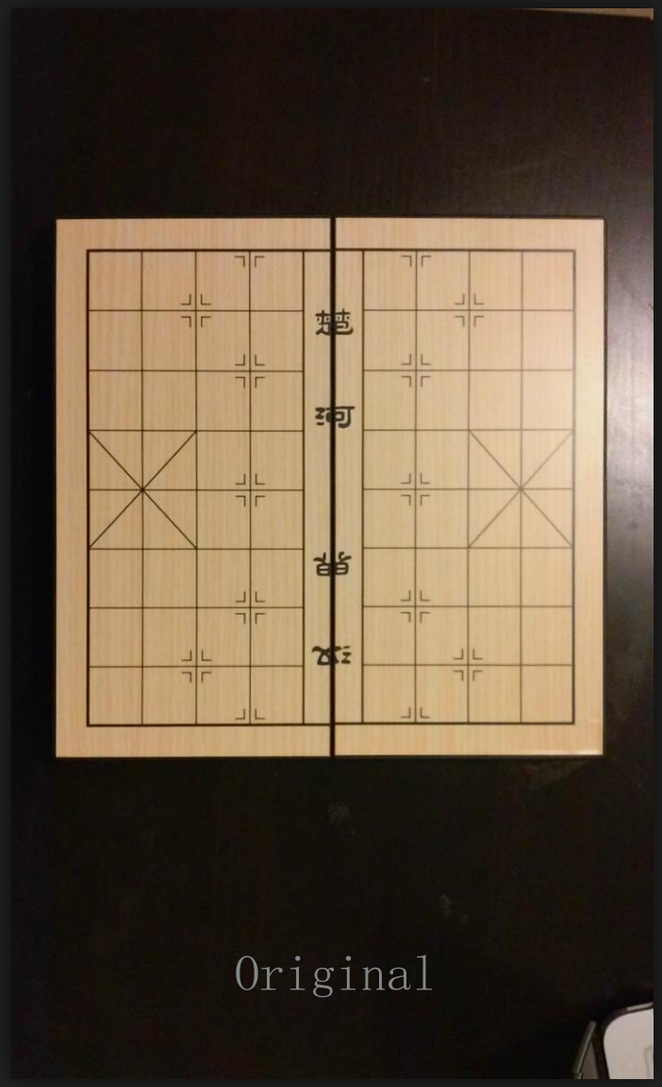




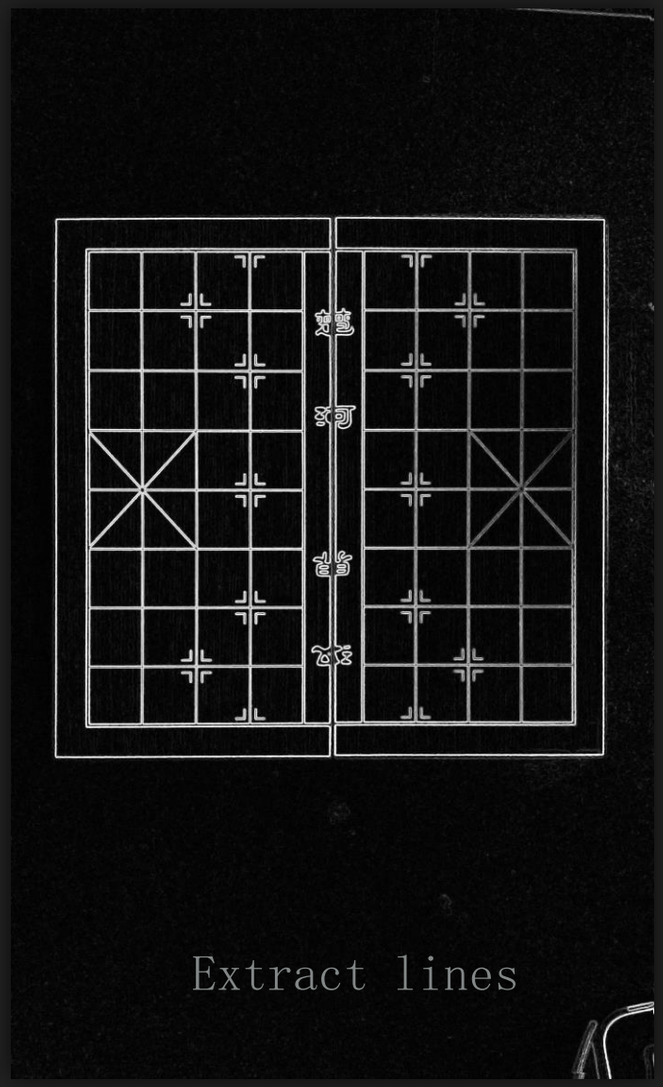
After optimizing



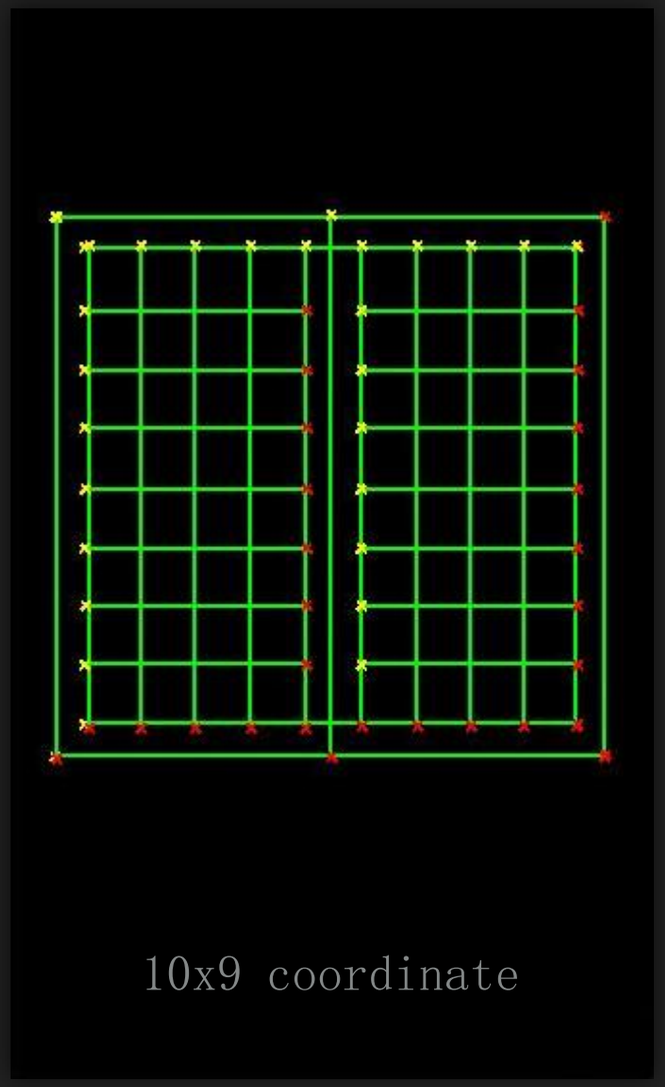
STEP.4 LINE DETECTION



Original



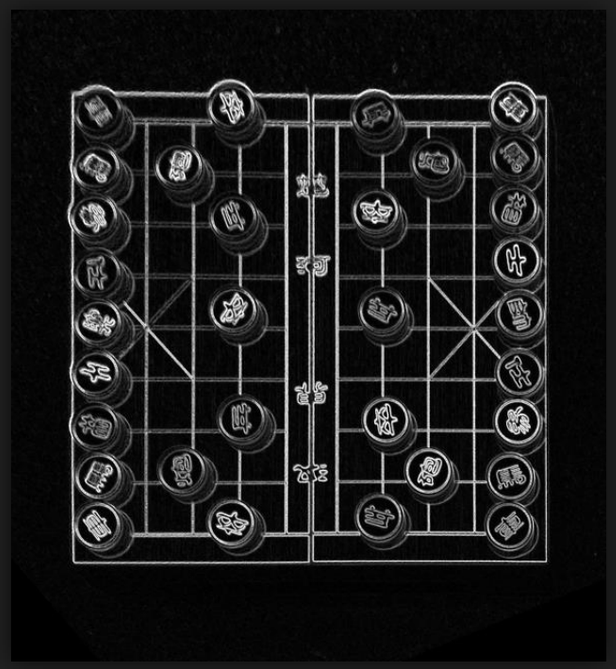
Extract lines

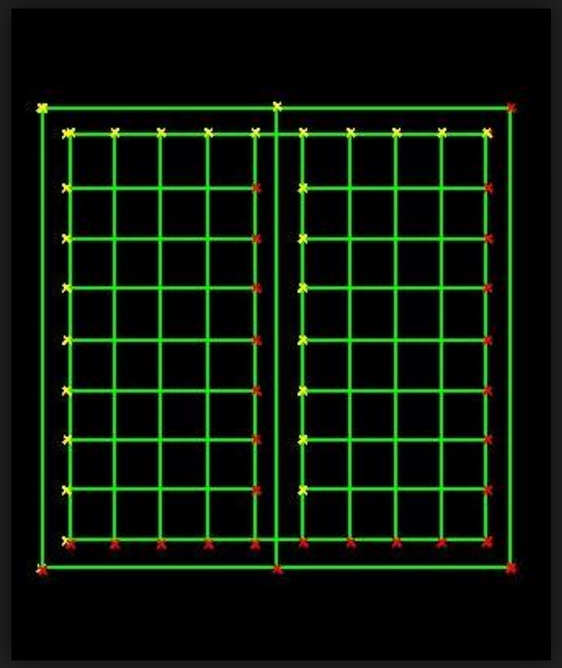


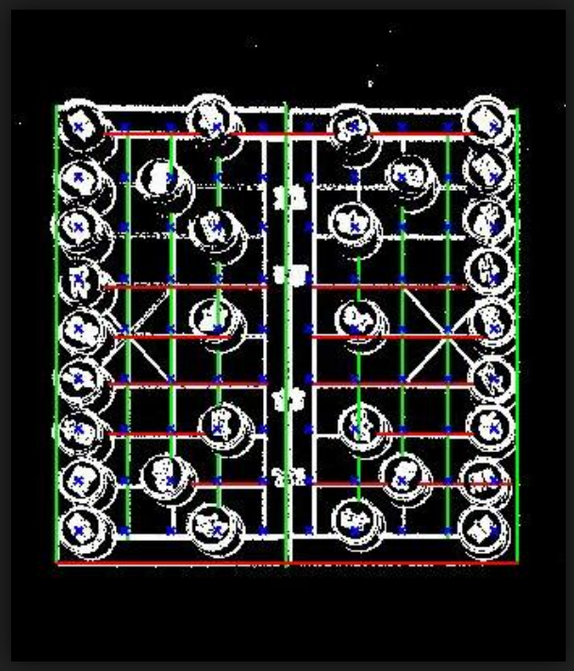
10x9 coordinate

Detecting lines makes it possible to build a “Chess Board Coordinate”!

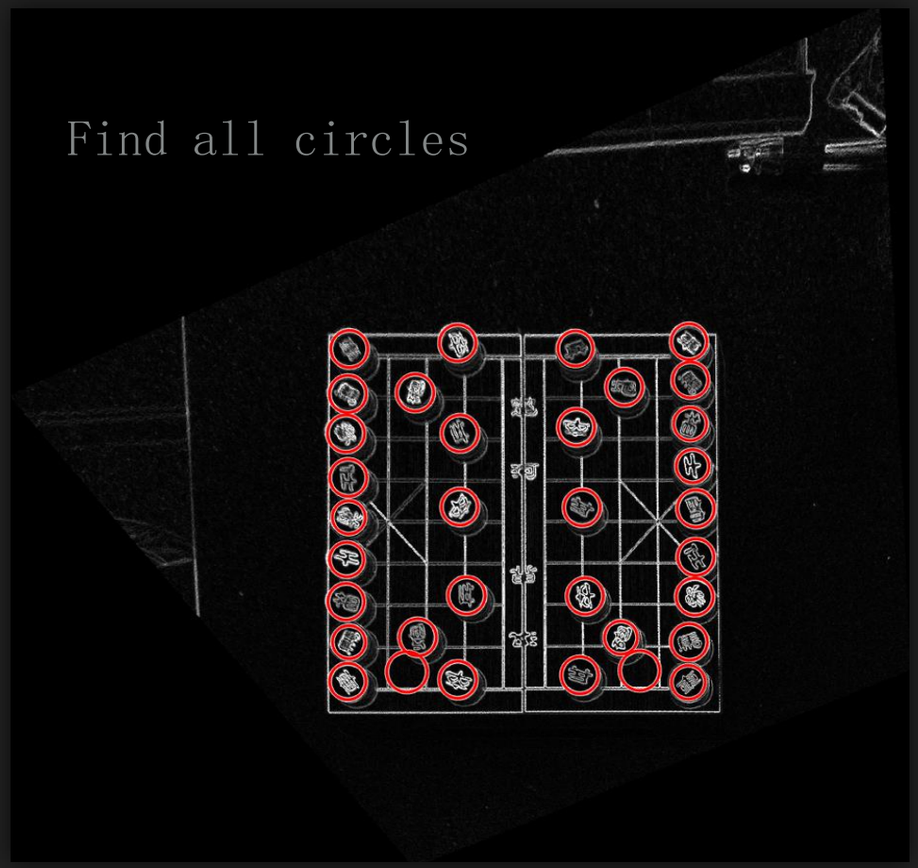
STEP.4 LINE DETECTION

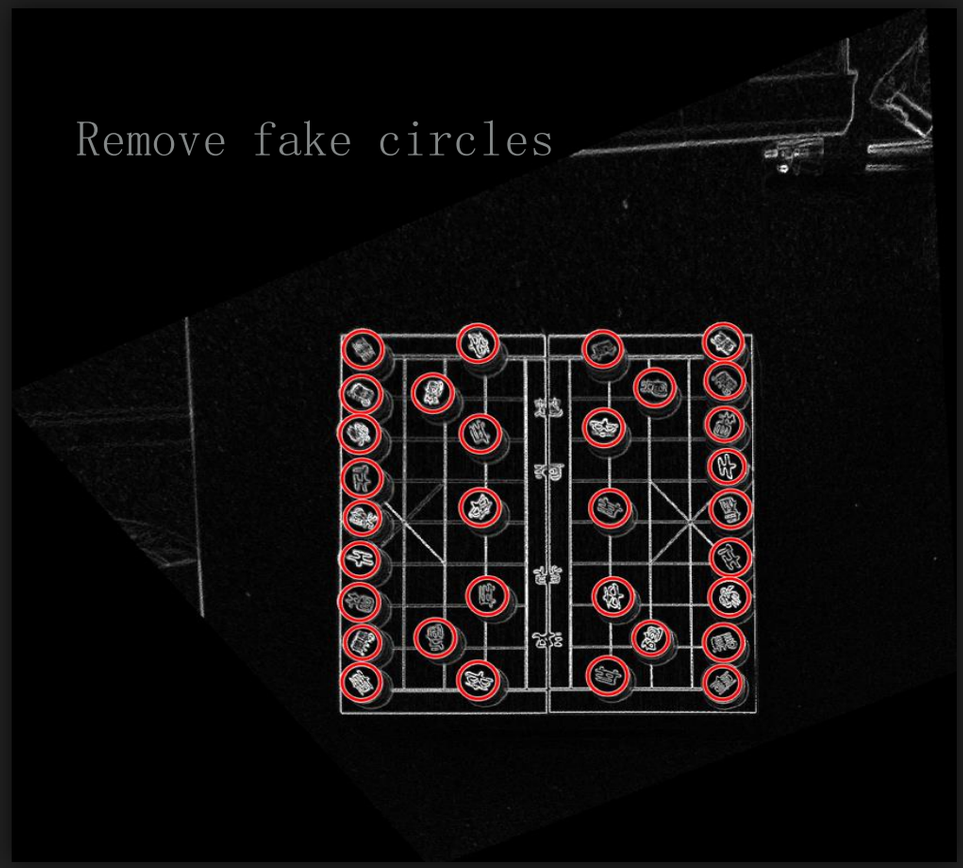






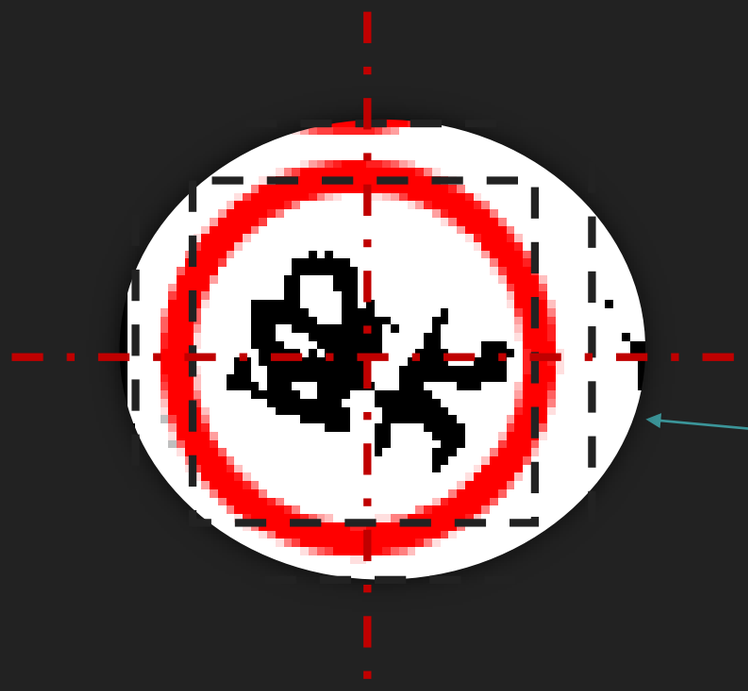
STEP.5 POSITION CHESS PIECES

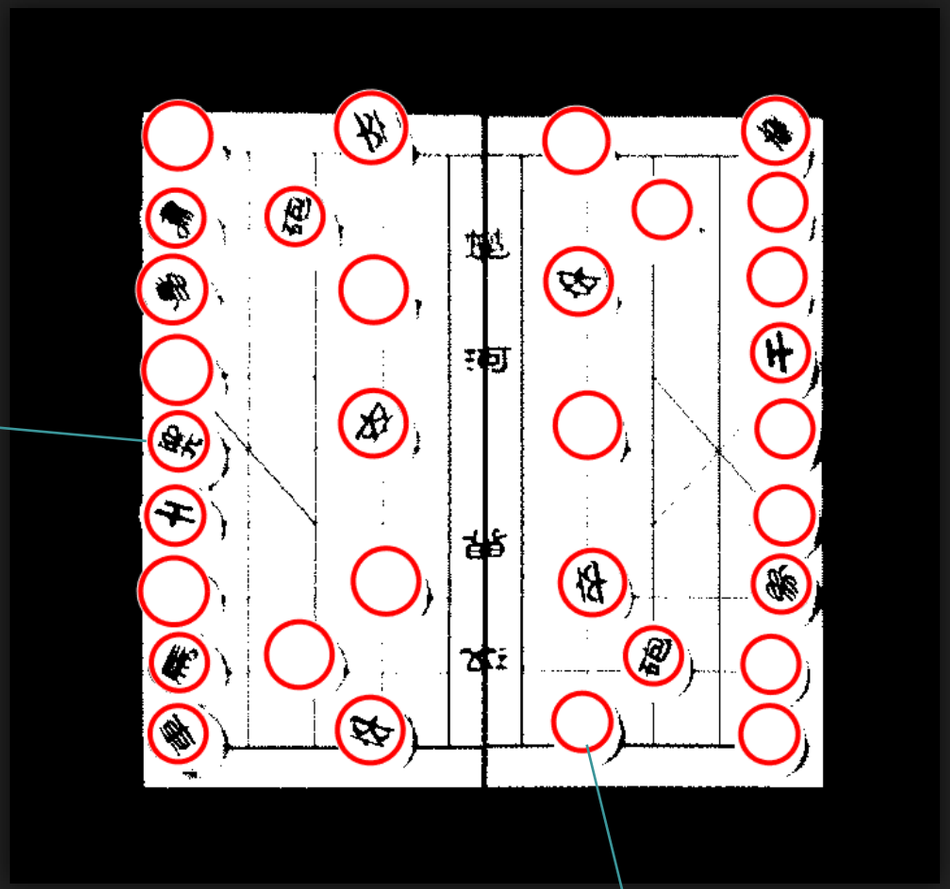


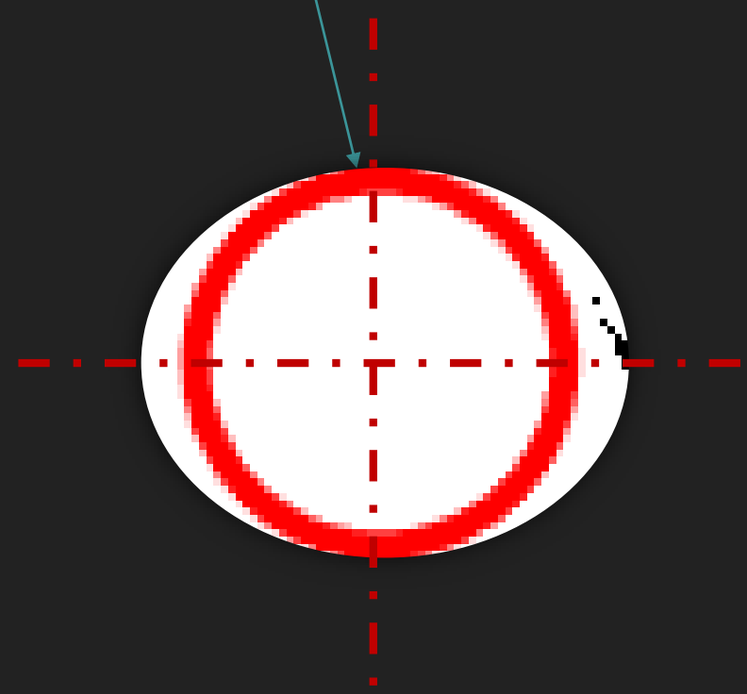



```
for i=1:10
    for j=1:9
        for p=1:m
            if((centers(p,1)<point(i,j,1)+min_lines/4)&&(centers(p,1)>point(i,j,1)-min_lines/2.5))
                if((centers(p,2)<point(i,j,2)+min_columns/4)&&(centers(p,2)>point(i,j,2)-min_columns/2.5))
                    h = viscircles(centers(p,:),radii(p,:));
                    s=sprintf('(%d,%d)',i,j);
                    disp(s);
                    break;
                end
            end
        end
    end
end
```

STEP.6 IDENTIFY CHESS PIECES' COLORS





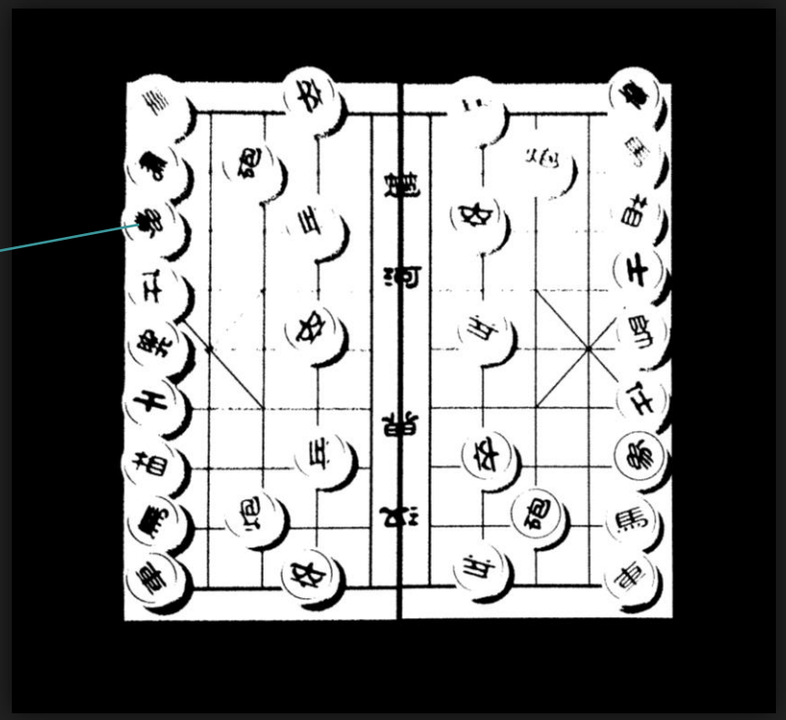


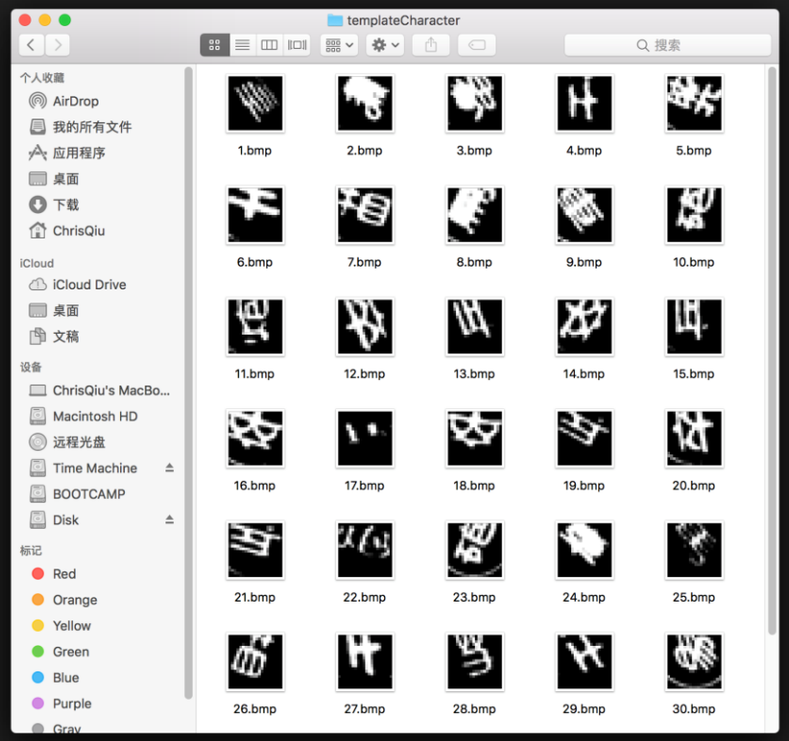
Assuming center coordinate of circle is (x, y) and its radius is r.

$$Pixels = \sum_{t=-\frac{r}{2}}^{t=\frac{r}{2}} P_{(x,y+t)}$$

If $Pixels > r - pixels$,
we consider this chess piece is black.

STEP.7 IDENTIFY CHESS PIECES' CHARACTERS





$$\begin{cases} Y_i = \dfrac{V_i}{\dfrac{W_i}{T_i} \times \dfrac{X_i}{U} \sqrt{\dfrac{(T_i - TUV_i)^2 + (U - TUV_i)^2 + (V_i - TUV_i)^2}{2}}} \\ \mathrm{TUV} = \dfrac{T + U + V}{3} \end{cases}$$

PERSONNEL ALLOCATION

张凌霄

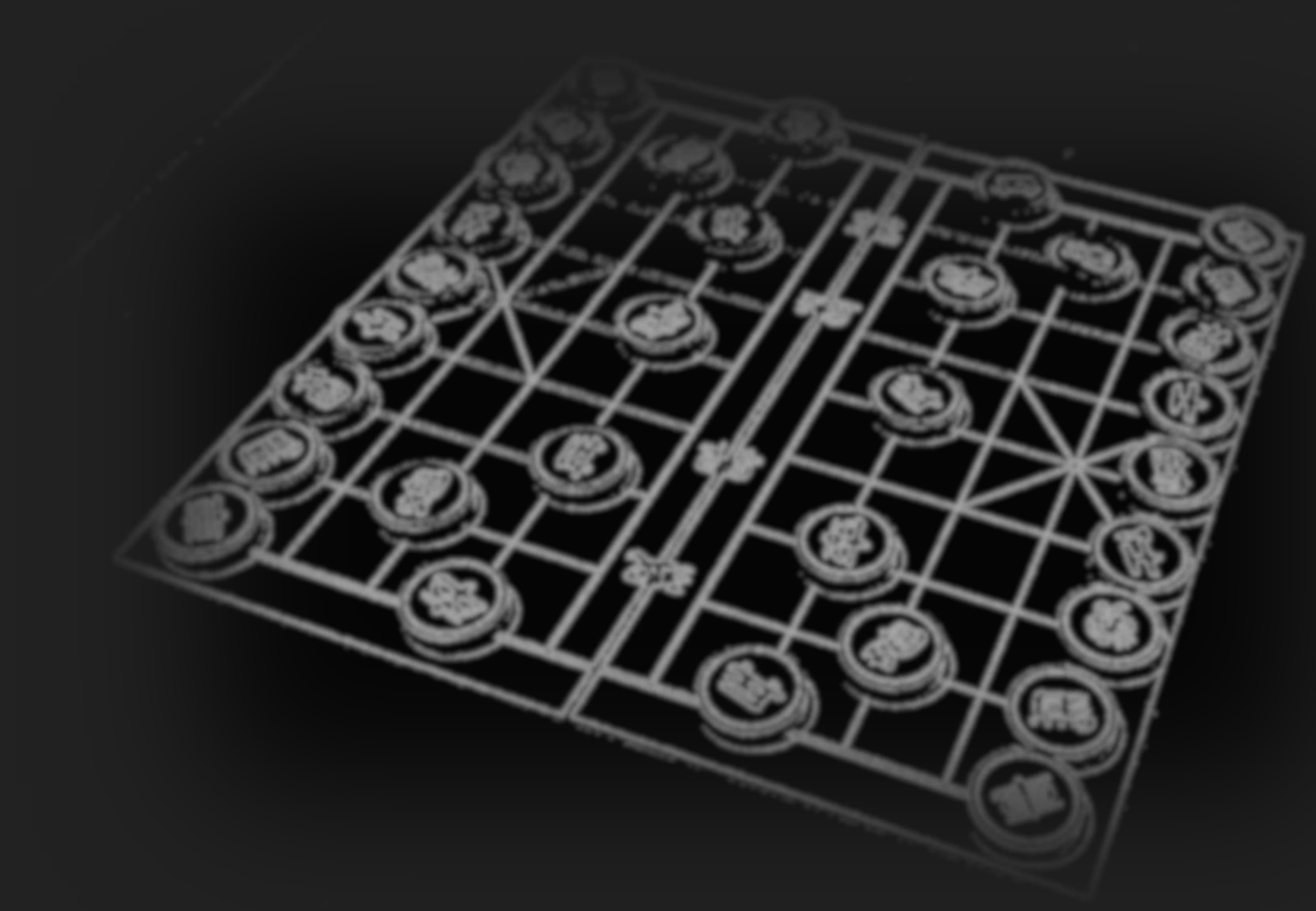
- 编写边缘提取、透视变换、直线检测、圆形识别等部分程序
- 安排项目进度

邱子濛

- 编写棋子颜色识别、棋子字符识别等部分程序，建立棋子字符模板
- 制作PPT并展示

郑睦炜

- 编写图像优化部分程序
- 撰写小组报告



基于MATLAB的

象棋棋盘及棋子识别

张凌霄 邱子濛 郑睦炜