```
#   _*_ coding:utf-8 _*_
__author__ = 'admin'

from PIL import Image, ImageDraw, ImageFont

im = Image.open(r'C:\Users\admin\Desktop\1515415644.png')
im.show()
im_pixel = im.load()
width, height = im.size
draw = ImageDraw.Draw(im)

#   变量初始化
#   棋盘重心的横纵坐标
target_x = 0
target_y = 0
#   棋子的匹配域内像素点个数、横纵坐标的和统计
sum_piece_x = 0
count_piece_x = 0
sum_piece_y = 0
count_piece_y = 0
#   棋盘的上顶点/上边缘的像素点个数、横纵坐标的和统计
sum_target_x = 0
count_target_x = 0
sum_target_y = 0
count_target_y = 0
#   棋盘上顶点、下顶点初始化
oberer_pol_x = 0
oberer_pol_y = 0
lower_vertex_x = 0
lower_vertex_y = 0

#   已知棋子重心的色域，扫描匹配项，公式求重心
for i in range(width):
    for j in range(height):
        pixel = im_pixel[i, j]
        if (50 < pixel[0] < 60) and (53 < pixel[1] < 63) and (95 < pixel[2] < 110):
            #   匹配的像素值都换为黄色，方便观察
            draw.ellipse((i, j, i, j), fill=(255, 255, 0))
            sum_piece_x += i
            count_piece_x += 1
            sum_piece_y += j
            count_piece_y += 1

# 使用公式锁定棋子重心坐标
piece_x = sum_piece_x // count_piece_x
piece_y = sum_piece_y // count_piece_y

#   缩减棋盘的扫描范围，以棋子的中中心线为分割，向左或向右扫描
```

```
if piece_x < width / 2:
    board_x_start = piece_x
    board_x_end = width
else:
    board_x_start = 0
    board_x_end = piece_x

for i in range(int(height / 3), int(height * 2 / 3)):
    #   先确定一个参照像素，当扫描到棋盘上边界时，好做判断(不然我怎么知道扫描到的这个像素就是属于棋盘的了呢)
    refer_pixel = im_pixel[0, i]
    #   经过下面搜寻后获得上顶点
    if target_x != 0 or target_y != 0:
        break
    for j in range(board_x_start, board_x_end):
        # 获取坐标点的像素值
        pixel = im_pixel[j, i]
        #   将搜索的和坐标区域刨除棋子的宽度(由于不确定piece_x就是棋子的正中心，所以直接刨除棋子的完整宽度102，为了稳妥)
        if abs(j - piece_x) < 102:
            continue
        if abs(pixel[0] - refer_pixel[0]) + abs(pixel[1] - refer_pixel[1]) + abs(pixel[2] - refer_pixel[2]) > 10:
            #   游戏内的棋盘有菱形和圆形两种形状
            #   ①菱形的上顶点就是一个单纯的像素点
            #   ②圆形无上顶点，它的上边缘是一行相同的像素点
            sum_target_x += j
            count_target_x += 1
    # sum_target_x不为0时，获取棋盘近似的上顶点的横坐标(不要放在for循环中了，没必要浪费计算时间)
    if sum_target_x:
        #   上顶点的近似横坐标
        oberer_pol_x = int(sum_target_x / count_target_x)
        #   上顶点的纵坐标
        oberer_pol_y = i
        #   打印上顶点坐标（可通过windows自带的画图，查看确定的是否准确)
        # print(oberer_pol_x, oberer_pol_y, "UP")
        # 上顶点的像素值(不然该值要global)
        oberer_pol_pixel = im_pixel[oberer_pol_x, oberer_pol_y]
        #   下面来确定棋盘的下定点
        #   ①菱形的上顶点和下顶点在一条对角线上，目前游戏内的此类对角线与Y轴垂直
        #   ②圆形无上顶点和下顶点在一条直径上，该直径与Y轴也垂直
        #   依照已经确定的上顶点纵坐标，向下+347的像素，即下顶点必然在[oberer_pol_y, oberer_pol_y + 347]上
        #   注意：这里的347需要调整，必须是所有棋盘的对角线、直径的最大值，不然是搜不到下顶点的
        for m in range(oberer_pol_y + 347, oberer_pol_y, -1):
            temp_pixel = im_pixel[oberer_pol_x, m]
```

```
            if (temp_pixel[0] == oberer_pol_pixel[0]) and (temp_pixel[1] ==
oberer_pol_pixel[1]) and \
                    (temp_pixel[2] == oberer_pol_pixel[2]):
                lower_vertex_y = m
                lower_vertex_x = oberer_pol_x
                #   打印下顶点坐标
                # print(lower_vertex_x, lower_vertex_y, "down")
                break
        # 不论直径还是对角线，(上顶点 + 下顶点) / 2便是棋盘中心点
        target_x = (oberer_pol_x + lower_vertex_x) / 2  # 这步有些多余
        target_y = (oberer_pol_y + lower_vertex_y) / 2

#   标记出图片的height的三分之一和三分之二间区域
draw.line(((0, int(height / 3)), (width, int(height / 3))), (0, 0, 0))
draw.line(((0, int(height * 2 / 3)), (width, int(height * 2 / 3))), (0, 0, 0))
#   在已知棋子的重心坐标时，划分棋子的左右区域
draw.line(((piece_x, int(height * 1 / 3)), (piece_x, int(height * 2 / 3))), (0,
128, 64))
#   在已知棋子的重心坐标时，划分棋子的上下区域
draw.line(((0, piece_y), (width, piece_y)), (0, 128, 64))
#   标记棋子重心
draw.ellipse((piece_x - 3, piece_y - 3, piece_x + 3, piece_y + 3), fill=(255, 0,
0))
#   在已知棋盘的重心坐标时，划分棋盘的左右区域
draw.line(((target_x, int(height / 3)), (target_x, int(height * 2 / 3))), (255, 0,
128))
#   在已知棋盘的重心坐标时，划分棋盘的上下区域
draw.line(((0, target_y), (width, target_y)), (255, 0, 128))
#   标记出棋盘的上顶点
draw.ellipse((oberer_pol_x - 3, oberer_pol_y - 3, oberer_pol_x + 3, oberer_pol_y +
3), fill=(0, 0, 0))
#   标记出棋盘的下顶点
draw.ellipse((lower_vertex_x - 3, lower_vertex_y - 3, lower_vertex_x + 3,
lower_vertex_y + 3), fill=(0, 0, 0))
#   标记出棋盘的重心
draw.ellipse((target_x - 3, target_y - 3, target_x + 3, target_y + 3), fill=(0, 0,
0))
#   连接棋子重心和棋盘重心
draw.line(((piece_x, piece_y), (target_x, target_y)), (255, 128, 64))

im.show()
```