信号与系统大作业之“跳不停”

无 64 贾宇宽 2016011069

1. 算法思路

目标是需要根据小人所处位置和目标块的中心位置给出距离乘时间系数得到按压时间完成跳跃，所以**核心技术**是根据游戏截图(1920*1080)得到**小人的位置坐标**(这里取小人的底部中心)，和**目标块的中心位置**(这里认为是对角线交点)。

算法中标*为作者非常创新的想法，待与大家讨论。

a) 小人的位置坐标

基本方法：根据小人的颜色（定义为**棋子色**），约为（54 60 102），对接近这种颜色对像素点进行滤波，得到判断是否为棋子色的布尔矩阵，然后从左向右扫描找到最左边的点，由于小人的大小基本不随着游戏的进行而改变，所以从该点到底部中心的位移是一定的，由此得到小人底部中心坐标。

***进阶处理：**

1. 然后由于有些混淆的目标块，例如“嘻哈盒”和“微信猫(狗？)”的侧面，也含有棋子色的像素，因此增加判断条件：

   左边缘点向右 60 个像素点基本应都为棋子色；

   棋子中心横坐标的向上 80 个点应多半为棋子色；

   棋子中心向下 15 个点应为棋子色；

   从而解决混淆的问题。

2. 在底部点的精确测定上，如果当前估计的底部中心坐标点**不是棋子色**，则上移一个点，若下面点坐标**也为棋子色**，则下移一个点，直到达到边界。在边界处测量左右两边棋子色像素点的宽度，取中即可。

b) 目标块的中心位置

矩形块上边缘的检测： 首先将 RGB 图转换为 HSV 图，取饱和度（Saturate）分量，进行 **Canny 边缘检测**[1]，再使用 **Hough 变换检测直线**[2]，由于所有的目标块边的角度均为 60 / -60 度，且目标块的上边缘应为全图中最高的 60 / -60 度的两条直线，由此确定矩形块的两条上边缘。

***排除棋子的影响：**根据棋子中心点水平坐标，观察得知如果棋子中心点处在左半屏幕(x<=540)，则一定往右跳，否则一定往左跳。假设棋子向右跳，考虑到棋子可能比目标块的上顶点高的影响，可以去除边缘检测结果在棋子“脖子处”右边缘以左的所有数值，从而不会将棋子的上半身当作目标块的上顶点。

目标块上顶点的检测：利用边缘检测的结果，检测在高度为 400 以下的最高边缘点，一般即为目标块的上顶点。

*但是偶尔边缘检测图片得到的结果会在背景出现水平横线，需判断排除；

*且有时图片会出现干扰，比如测试图片 10 中的飞起的音乐符号，也需要判断后排除；

判断是否为矩形块（即是否可用直线结果）

由于 Hough 变换检测直线有时并不能很准确定出直线，需判断上边缘两条直线的上顶点是否与目标块上边缘点很近，若距离超过 10 则认为此直线“偏离”。

---

[1] edge(im, ‘canny’) -Matlab Image Processing Toolbox
[2] hough, houghpeaks, houghlines -Matlab Image Processing Toolbox

直线法确定中心：

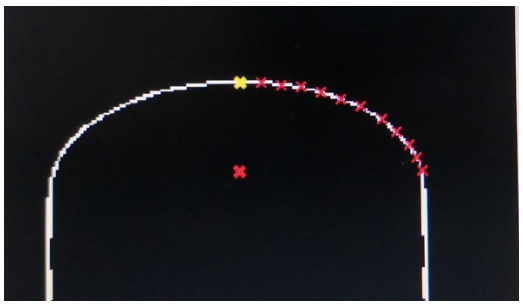
若两条直线均被认定，且长宽比不大于 1.5（不超过长方形 CD 机的长宽比），则取**两条直线的下顶点坐标平均值为中心点**；若长宽比过大，考虑为其中一条直线被过短的识别，将短直线置为“偏离”；

若两条直线均偏离，则采用后续的“边缘下降法”；

若一条偏离，一条边缘被认定，则取**认定直线的下顶点的纵坐标和目标块上顶点的横坐标为目标块坐标**；

*__边缘下降法：__这种方法用于检测圆形目标块以及由于纹理难以识别直线的矩型目标块。如图，从目标块上顶点开始，**观测其右下方是否存在下一个边缘点，若存在，则跳向下一个边缘点**，以此类推，直到达到右边缘（这里取右边缘可以排除影子打向左边的影响），即右下方短距离没有边缘。

	?	?	?	?	?	?	?	?
*	?	?	?	?	?	?	?	?
	?	?	?	?	?	?	?	?
	?	?	?	?	?	?	?	?
	?	?	?	?	?	?	?	?
	?	?	?	?	?	?	?	?
	?	?	?	?	?	?	?	?



*__细节处理__:由于跳的时候**优先大步跳**，优先向右跳，需检测边缘在相同横坐标下的最高点，退回作为右边缘点；

由于边缘点位于两种颜色之间，取左边点作为上表面颜色，若当前点颜色不同，则向上／向下调整不超过两个点，退回到上表面之内；

然后检测上表面内最右点向下有多少点与上表面颜色一致，取平均作为上表面右边缘点。

最终确定目标块中心点坐标为**目标块顶点横坐标**和**上表面右边缘点纵坐标**。

2. 调试问题

1. 在调试的过程中，首先的问题是 Hough 变换对有纹理的矩形块并不能很好的识别出直线，尤其是“木质椅子”、“433 天”等，其边缘图很乱，有时只能识别一条边，有时会识别出很短的直线，甚至能识别出偏离的直线。所以修正了“直线法确定中心”应用的条件，对不满意的直线识别结果采用“边缘下降法”重新进行计算；
2. 另外有些遮挡和意外的因素也需要考虑，例如飞起的音乐符号需要被滤掉，否则会被当成目标块上边缘，并且遮挡住的目标块也会影响直线长度的判断，从而使中心点位置偏离；
3. 最大的问题在于电脑在运行 Matlab 的时候，无论使用虚拟机还是 MuMu 模拟器（相对流畅），均发生较大的卡顿现象，引入跳跃距离的噪声，甚至在截图时发生卡顿导致截图时新的目标块仍在空中；
4. 在电脑状态良好，使用占用资源较少的 MuMu 模拟器时，可以连续跳到+32 分连击，可达到 3000+的分数，如图，可以证明该“跳不停”程序运作良好。

3. 关键代码

1. chess_middle.m

```
function [midpoint] = chess_middle(img)
img_red = img(:,:,1);
img_green = img(:,:,2);
img_blue = img(:,:,3);

%filter for chess color
is_chess = (abs(img_red - 54) + abs (img_green - 60) + abs(img_blue -
102) < 10);

%looking for the leftmost verified point of chess (detail in repo)
chess_column = sum(is_chess);
found = 0;
for left_bound = 150:1000
    if chess_column(left_bound) > 0
        for k = 1620:-1:900
            if  is_chess(k, left_bound) && sum(is_chess(k,
left_bound :1: left_bound+60)) > 55 && is_chess(k+15, left_bound+38)
&& sum(is_chess(k-80:1:k, left_bound+38)) > 40
                left_point_x = left_bound;
                left_point_y = k;
                found = 1;
                break;
            end
        end
        if(found == 1)
            break;
        end
    end
end
%approximately get the bottom point
midpoint = [left_point_x + 37 left_point_y+17];

%modify the height
if is_chess(midpoint(2), midpoint(1) == 0)
    midpoint(2) = midpoint(2) - 1;
end
while is_chess(midpoint(2)+1, midpoint(1))
    midpoint(2) = midpoint(2) + 1;
end
```

```
%modify the width
left = midpoint(1); right = midpoint(1);
while is_chess(midpoint(2), left - 1)
    left = left - 1;
end
while is_chess(midpoint(2), right + 1)
    right = right + 1;
end
midpoint(1) = floor((left+right)/2);
end
```

2. board_middle.m

```
function midpoint = board_middle(img, chess_x)
img_red = img(:,:,1);
img_green = img(:,:,2);
img_blue = img(:,:,3);
img_hsv = rgb2hsv(img);
img_bw = img_hsv(:,:,2);
img_edge = edge(img_bw, 'canny');
%figure;imshow(img_edge); hold on;

%eliminate the influence of chess
if(chess_x <= 540)
    img_edge = img_edge .* [zeros(1,chess_x+33) ones(1, 1080 -
(chess_x+33))];
else
    img_edge = img_edge .* [ones(1,chess_x-33) zeros(1, 1080 -
(chess_x-33))];
end

%detect for lines
[H, T, R] = hough(img_edge);
P = houghpeaks(H, 100, 'threshold', 0);
lines = houghlines(img_edge, T, R, P, 'FillGap', 11, 'MinLength',
80);
tallest_left_line_height = 1920; tallest_left_line = 1;
tallest_right_line_height = 1920; tallest_right_line = 1;

%take tallest left/right lines
for k = 1:length(lines)
    if lines(k).theta >= 58 && lines(k).theta <= 62
```

```
        if max(lines(k).point1(2), lines(k).point2(2)) <
tallest_left_line_height
            tallest_left_line = k;
            tallest_left_line_height = max(lines(k).point1(2),
lines(k).point2(2));
        end
    end
end
for k = 1:length(lines)
    if lines(k).theta >= -62 && lines(k).theta <= -58
        if max(lines(k).point1(2), lines(k).point2(2)) <
tallest_right_line_height
            tallest_right_line = k;
            tallest_right_line_height = max(lines(k).point1(2),
lines(k).point2(2));
        end
    end
end

%check for music notes
is_music = (abs(int16(img_red) - 177) + abs (int16(img_green) - 181)
+ abs(int16(img_blue) - 209) < 5);

%detect the tallest point of target board
sum_raw = sum(img_edge, 2);
for top_corner = 400:1800
    found = 0;
    if(sum_raw(top_corner) > 0)
        for k = 101:1000
            if img_edge(top_corner, k) == 1 && sum(img_edge(top_corner,
k-50:k)) < 45 && sum(img_edge(top_corner, k:k+50)) < 45 &&
sum(sum(is_music(top_corner-3:top_corner+3, k-3:k+3))) == 0
                top_point(1) = k + floor(sum(img_edge(top_corner,
k:k+40)/2));
                top_point(2) = top_corner;
                found = 1;
                break;
            end
        end
        if(found == 1)
            break;
        end
    end
end
```

```
%plot(top_point(1), top_point(2),
'x','LineWidth',2,'Color','yellow');

%verify the lines to check the rectangle
if distance(lines(tallest_left_line).point2, top_point) < 10
    left_verified = 1;
else
    left_verified = 0;
end
if distance(lines(tallest_right_line).point1, top_point) < 10
    right_verified = 1;
else
    right_verified = 0;
end
length_left = distance(lines(tallest_left_line).point1,
lines(tallest_left_line).point2);
length_right = distance(lines(tallest_right_line).point1,
lines(tallest_right_line).point2);
if left_verified && right_verified
    if  length_left * 1.4 < length_right
        left_verified = 0;
    elseif length_left > length_right * 1.4
        right_verified = 0;
    end
end

%check verified
if left_verified
    if right_verified
        midpoint(1) = ceil((lines(tallest_left_line).point1(1) +
lines(tallest_right_line).point2(1))/2);
        midpoint(2) = ceil((lines(tallest_left_line).point1(2) +
lines(tallest_right_line).point2(2))/2);
    else
        midpoint(1) = top_point(1);
        midpoint(2) = lines(tallest_left_line).point1(2);
    end
else
    if right_verified
        midpoint(1) = top_point(1);
        midpoint(2) = lines(tallest_right_line).point2(2);
    else
        midpoint = calc_others(img, top_point);
    end
```

```
end

%plot(midpoint(1), midpoint(2), 'x','LineWidth',2,'Color','red');
end
```

3. calc_others.m

```
function midpoint = calc_others (img, top_point)
img_hsv = rgb2hsv(img);
img_bw = img_hsv(:,:,2);
img_edge = edge(img_bw, 'canny');
%imshow(img_edge);hold on;

%jump down alongside the right edge
cursor(2) = top_point(1);cursor(1) = top_point(2);
while sum(sum(img_edge(cursor(1)-1:cursor(1)+5,
cursor(2)+1:cursor(2)+8))) > 1 && cursor(1) < 1500 && cursor(2) <
1060
    found = 0;
    for right = 8:-1:1
        for down = 5:-1:-1
            if img_edge(cursor(1)+down, cursor(2)+right) &&
right+down > 0
                cursor = cursor + [down right];
                found = 1;
                break;
            end
        end
        if found
            break;
        end
    end
    %plot(cursor(2), cursor(1), 'x','LineWidth',1,'Color','red');
end

%take the highest edgepoint
while img_edge(cursor(1) - 1, cursor(2))
    cursor(1) = cursor(1) - 1;
end

%emboard on the surface
if abs(sum(int16(img(cursor(1), cursor(2), :)) - int16(img(cursor(1),
cursor(2) - 1, :)))) > 5
```

```
    if abs(sum(int16(img(cursor(1) + 2, cursor(2), :)) -
int16(img(cursor(1), cursor(2) - 1, :)))) < 5
        cursor(1) = cursor(1) + 2;
    elseif abs(sum(int16(img(cursor(1) + 1, cursor(2), :)) -
int16(img(cursor(1), cursor(2) - 1, :)))) < 5
        cursor(1) = cursor(1) + 1;
    elseif abs(sum(int16(img(cursor(1) - 1, cursor(2), :)) -
int16(img(cursor(1), cursor(2) - 1, :)))) < 5
        cursor(1) = cursor(1) - 1;
     elseif abs(sum(int16(img(cursor(1) - 2, cursor(2), :)) -
int16(img(cursor(1), cursor(2) - 1, :)))) < 5
        cursor(1) = cursor(1) - 2;
    end
end

%take the midpoint of right edge of surface
same_layer = img(cursor(1):cursor(1) + 50, cursor(2), 1:3);
same_layer_red = same_layer(:,:,1);
same_layer_green = same_layer(:,:,2);
same_layer_blue = same_layer(:,:,3);
same_layer = (abs(int16(same_layer_red) - int16(same_layer_red(1,1)))
+ abs (int16(same_layer_green) - int16(same_layer_green(1,1))) +
abs(int16(same_layer_blue) - int16(same_layer_blue(1,1))) < 5);
layerlength = 1;
while(same_layer(layerlength, 1)  && layerlength < 50)
    layerlength = layerlength + 1;
end

%check points under to choose the output
if sum(sum(same_layer)) < 15
    midpoint = [top_point(1) cursor(1)+floor(layerlength/2)];
else
    midpoint = [top_point(1) cursor(1)];
end
```

4. 总结

这次大作业的内容，制作一个跳一跳的外挂程序是非常令人兴奋的，所以我做的很认真，综合考虑了各种情况进行调试，但受制于电脑配置，总是发生定点很准但连击中断的情况，即使如此，我仍然乐此不疲。

在平台搭建上，协助应睿同学进行测试也是很锻炼能力的过程，这是我首次使用 Matlab 完成一个综合性的任务，又是解决了一个生活中遇到的问题，感到收获很大。

程序完全是在**自主探索**之后写出来的，**除了 Matlab 的工具箱没有利用任何开源工具**，**并未查阅其他人的工作**，并且运行结果远好于给定的 example/matlab 的效果。

感谢老师、助教和开发同学的付出！

5. 参考资料

1. Matlab Documentation