AI of WeChat's Jump

E-Neo

January 5, 2018

Outline

Outline

工具

- Python (3.6.4)
- Pillow (pip install Pillow)
- adb (sudo pacman -S android-tools)

```
import subprocess
import io
from PIL import Image, ImageFilter
```

前端代码

```
def get_screen():
    t = subprocess.run("adb shell screencap -p | sed 's/\r$//'",
                       stdout=subprocess.PIPE, shell=True)
    img = Image.open(io.BytesIO(t.stdout))
    return img


def jump(duration):
    subprocess.run("adb shell input swipe 0 0 0 0 " + str(duration),
                   shell=True)
```

Outline

思路分析

Figure: 游戏画面

两个方向：

1. 直接对原图进行操作
2. 先对原图进行图像处理再进行操作

图像特征

1. 我手机的屏幕分辨率是 720*1028
2. i 的（左下，右下）颜色是 (2b2b49, 3b3651) 基本不变
3. 背景色竖直方向存在渐变效果
4. 影子颜色不唯一
5. 目标都是规则几何图形（矩形或圆）

找出 i 的坐标

根据图像特征 2：i 的（左下，右下）颜色是 (2b2b49, 3b3651) 基本不变，可以按由左到右，由下到上的顺序扫描图像寻找 i 的底部最左端，然后向右偏移 25 个像素得到 i 的坐标。

```
def find_i(img):
    w, h = img.size
    pixel = img.load()
    for i in range(0, w):
        for j in reversed(range(0, h)):
            r, g, b, a = pixel[i, j]
            if r in range(40, 50)\
               and g in range(40, 50)\
               and b in range(70, 80):
                return i + 25, j
```

找出 d 的坐标

1. 图像处理边缘检测
2. 找出目标的最高点 (Pt)
3. 找出目标最右端的点 (Pr)
4. 对于中心对称的目标图形，(Pt.x, Pr.y) 即为所求的 d

图像处理

```
def process(img):
    img = img.convert('L').filter(ImageFilter.CONTOUR)
    threshold = 230
    img = img.point(lambda x: 0 if x < threshold else 255, mode='1')
    return img
```

图像处理

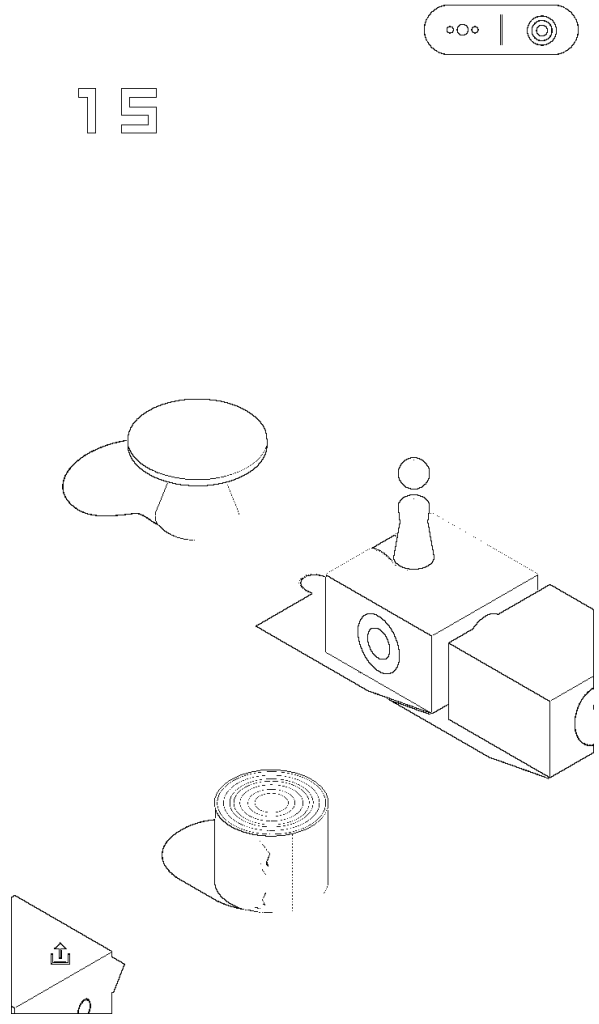

Figure: 处理后的图像

find_d_top

```
def find_d_top(pixel, img_size):
    w, h = img_size
    for j in range(300, h):
        for i in range(20, w - 20):
            if pixel[i, j] == 0:
                for k in range(i, w - 20):
                    if pixel[k, j] != 0:
                        return (i + k) >> 1, j
```

check_right, check_down

```
def check_right(pixel, i, j, steps=5):
    for k in range(i, i + steps):
        if pixel[k, j] == 0:
            return True
    return False

def check_down(pixel, i, j, steps=10):
    for k in range(j, j + steps):
        if pixel[i, k] == 0:
            return True
    return False
```

find_d_right

```
def find_d_right(pixel, img_size, d_top):
    w, h = img_size
    i, j = d_top
    while True:
        if check_right(pixel, i, j):
            i = i + 1
        elif check_down(pixel, i, j):
            j = j + 1
        else:
            while pixel[i, j] != 0:
                i = i - 1
            k = j
            while pixel[i, k] == 0:
                k = k + 1
            return i, (j + k) >> 1
```

find_d

```
def find_d(img):
    pixel = process(img).load()
    d_top = find_d_top(pixel, img.size)
    d_right = find_d_right(pixel, img.size, d_top)
    return d_top[0], d_right[1]
```

计算长按屏幕的时间

```
def calc_duration(img):
    i = find_i(img)
    d = find_d(img)
    distance = math.sqrt((i[0] - d[0])**2 + (i[1] - d[1])**2)
    return int(60 / 29 * distance)
```

单步运行

```
def step():
    img = get_screen()
    duration = calc_duration(img)
    jump(duration)
```

Outline

当前效果

Figure: 测试效果

效果不是很理想：

1. 分数不是很高
2. 分数不稳定

可能原因

- 目标物体不都是中心对称图形
- 唱片会蹦出音符
- 时间距离关系不是很准确

接下来编写调试函数，获取游戏过程中产生的数据，进一步分析原因并改进。

End

Thank you for you attention!