所需工具：

- Android Debug Bridge (adb)
- Python-pillow
- 一台 Android 设备

程序主要思路：

1. 首先通过 adb 获取安卓设备截图，并传输到电脑端。
2. 通过棋子最上端的颜色(#34353B)并加上一定的偏移(+5, +190)进行定位**初始点**：getChessPos()
3. 接下来获取终止点的位置，主要通过确定终止块的最上方顶点和最右方顶点进行确定。
4. 通过尝试，得出起始点与终止点的距离与时间的线性相关系数。

程序结构：

getScreenshot：获取手机目前截图

getChessPos：获取当前棋子位置并计算初始点位置

getDestinationUp：获取终止块最上方顶点

getDestinationRight：获取终止块最右方顶点

goDist：模拟安卓手机操作按压指定时间

getDist：通过给定图片计算距离

* 对于每一张图片，为了方便进行调试，标注了起始点，终止块上顶点，终止块右顶点，终止点的位置。

main：主程序

辅助函数：

blend(c1, c2, p1)：以 p1 比例混合两个颜色

similar(c1, c2)：判断两个颜色是否相似

程序编写/改进路线：

1. 在写 getDestinationUp 的时候需要注意，棋子可能会比最高块高，因此不能单纯

的从上至下寻找最高边缘点。

解决方案：在 x 轴上刨除起始点所在的左右区域。

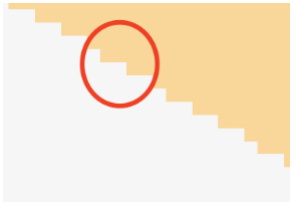
2. 在写 getDestinationRight 的时候遇到的问题比较多，前后想了大概 3 个版本：
   a) 从最简单的开始，在向右跳的时候，从右至左，从上至下扫描到第一个边缘点，即为最右侧点。这种方法最为简单粗暴，但是 handle 不了向左跳的情况。

      同时也有一些问题需要考虑：

      Q：从右至左的时候，下方已经跳过的块会进行干扰。

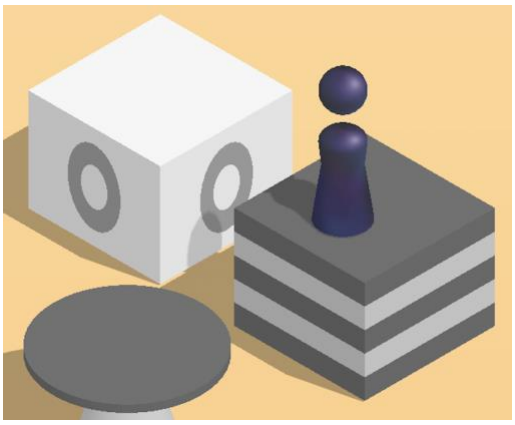
      A：可以使用起始点的 y 坐标进行一次过滤：终止点的 y 一定小于起始点的 y。

   b) 如果考虑到向左的跳情况，上述方法明显不适用，因此可以使用从起始点 x 坐标开始向右扫描，找到第一个不包含边缘点的列，返回之前的列。但是这样有一个问题因为图片可能会有一个不包含边缘点的列出现在块的内部：

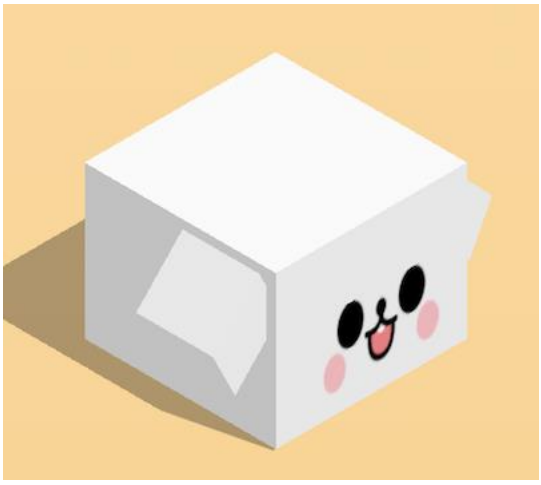
      

      在这种情况下，中间列因为边缘没有下降，没有水平方向的边缘点，因此我们需要添加一个变量来累计出现多少连续没有边缘点的量，在超过一个阈值之后将之前的列作为边缘点（最右点）返回。

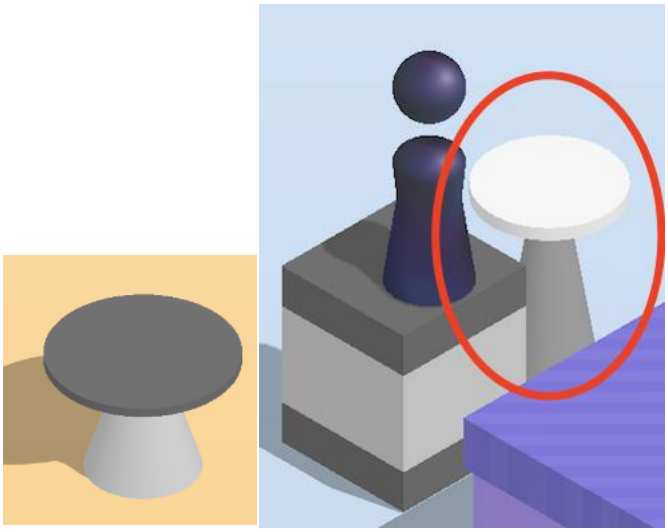
   c) 然而上述版本无法解决如下问题：

      在这种情况下将会把灰色块的右顶点识别为终止块右顶点，因为中间每一列都包含边缘点，这种情况我们需要统计每一列上真正的边缘点：相邻的点是背景色。因为背景是渐变的所以通过顶端颜色，底端颜色，当前 y 值可以计算出当前高度应该的背景色。而一个边缘应该有一定数量的相邻是背景的边缘点。同时也可以解决如下的盒子的识别问题：



然而经过测试发现在这种圆台上会出现问题：



如下两种圆台无法再单纯调整阈值的情况下同时识别右侧边界，左侧圆台的下方台柱造成了一定的干扰使得减小阈值将会使台柱成为边界；而右侧圆台的边界又太小使得需要一个小的阈值。为了解决这个问题。在判断边界的时候进行两次判断，第二次判断将前一列和本列的边缘点数量加并和新的阈值进行比较以判断出第二种圆台的边界。