《数字图像处理基础》课程实践

结题报告

项 目 名 称 ：数字图像处理之拼图课程实践

姓　　名 ：谢敬鱼

学　　号 ：516021910125

联 系 电 话 ：1262935410

电 子 信 箱 ：xjy0104@sjtu.edu.cn

合作人姓名 ：吴金柱、孙一鸣

2019 年　 6 月

目录

1. 研究内容综述

1.1 对比开题报告，说明各个研究内容的完成情况

与开题报告中所述目标与研究内容对比，我们完成了如下内容：
（1）拼图块的识别：由于初始时拼图块是随机放置在图片中的，因此要移动拼图碎片，第一步就是要确定拼图碎片的位置
（2）拼图块与原图的匹配：知道了拼图碎片的初始位置，根据初始位置为拼图碎片进行剪裁，并与标准图片对比得到拼图碎片的行列序号，之后根据行列序号及拼图目标区域坐标计算每一块拼图的目标位置
（3）拼图块定点移动：拼图碎片的起始位置与目标位置确定后，采用对应的机械臂控制将拼图碎片从起始位置移动到目标位置

这些任务是完成拼图游戏的关键环节。整个拼图游戏的逻辑处理可见下图。

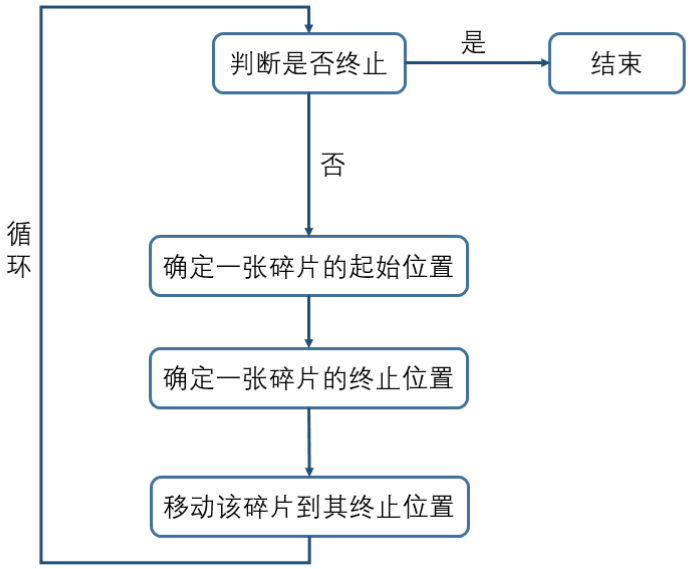


图 1.1 游戏逻辑设计

在实践过程中，我们首先尝试，直接识别拼图碎片。但是在我们的实现过程中由于摄像头拍摄图像质量较低，拼图碎片颜色较为随机等问题，若在一开始直接识别拼图碎片效果很差。因此我们基于我们所采用的拼图块匹配的算法——模板匹配来直接在初始图像中搜索相似度高的图片从而确定拼图碎片的初始位置。这样一来，算法中一个步骤可以逻辑上实现拼图游戏的两个部分，只需增加机械臂控制部分，就可以实现完整的算法处理。

基于上述各任务及算法整体的思路，我们实现了使用机械臂完成拼图游戏任务，同时我们还超额完成了任务，我们不仅设计了 3*3 的拼图算法，同时也完成了 4*4，5*5，6*6 的拼图游戏。由于不同的游戏在相同基本思路基础下，还具有很多特性的东西，因此我们在进行改进的时候对不同的拼图进行了个性化的处理代，使得我们的代码具有对不同规格的拼图游戏都有较好

的应用效果，同时具有较高的机械臂移动精度。综上，我们数图课程实践的整个任务都得到了充分的完成，并实现了较好的演示效果。

1.2 未完成的研究指标应说明原因

如上所述，我们完成了开题报告中所述的所有内容，并且完成度很高。

2. 研究方案

2.1 游戏简介

我们选择的游戏是拼图游戏，其分为自由模式、对换模式、九宫格模式。每个模式下有 3×3、4×4、5×5、6×6 四种难度，在本研究中，我们采用的是自由模式，其对应的四种难度初始化图像如下图（图 2.1.1）所示：

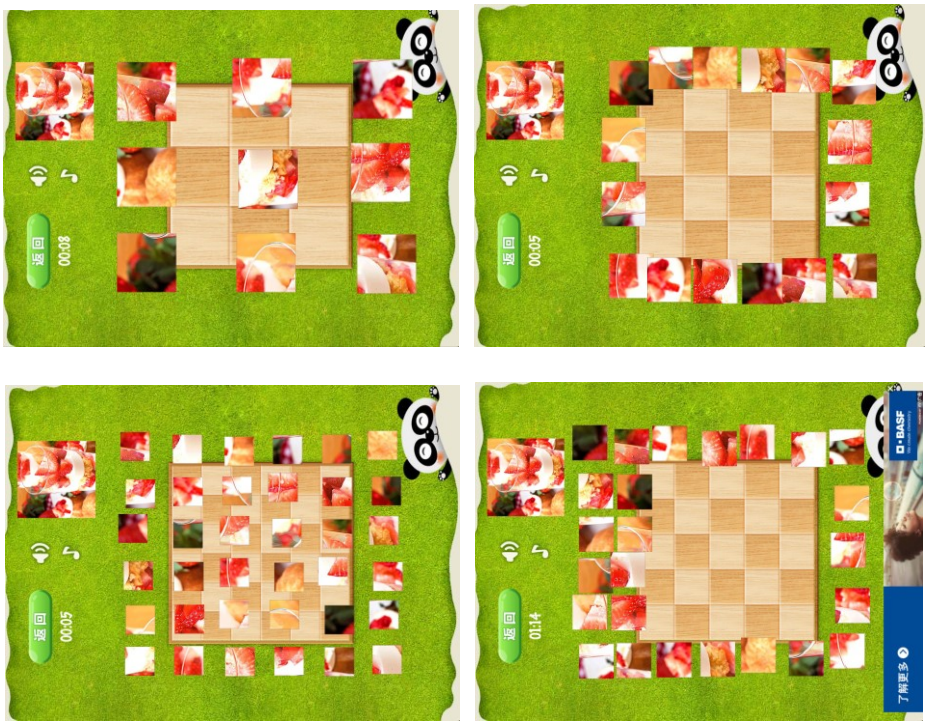


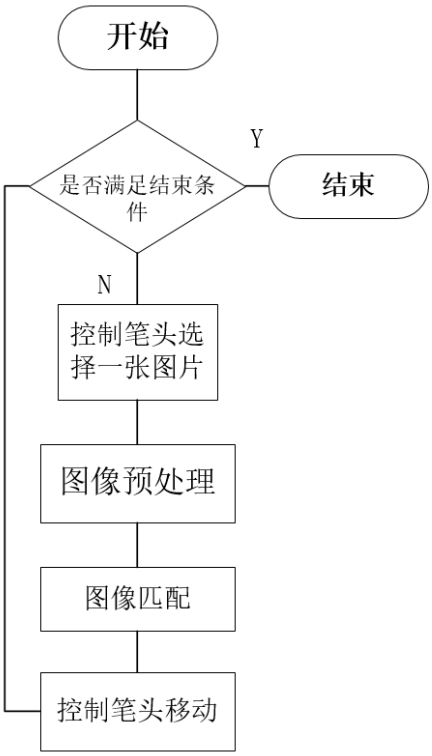
图 2.1.1 拼图游戏介绍

根据图像可以看出不同难度下的拼图游戏有一定的区别

1) 不同难度的拼图中，有的拼图碎片可能存在重叠但是有的不会
2) 不同难度的拼图中，有的拼图碎片位于目标区域上，占据了目标区域的位置；有的占据目标区域的位置不超过一个拼图大小
3) 不同难度拼图块大小不同，对于难度较高的拼图游戏，由于拼图块较小，偶尔会出现识别错误的问题，特别是相对于图像右上角的标准图像，较小的拼图块易发生识别匹配问题

2.2 研究流程概述

对于该研究，基于拼图实现的逻辑以及算法设计的实际情况，我们计划商讨拟按照如下流程进行。首先，我们需要先处理摄像头返回的图像（图像1）。通过图像预处理来降低光照等环境等因素的影响，然后对于处理过的图像（图像 2），我们裁剪获取图片中的参考图像（图像 3），然后根据图像 3 通过匹配找到散落碎片的位置。在图像 2 的基础上通过坐标运算得到碎片目标位置的坐标。最后控制机械臂进行移动来完成拼图。



2.2.1 研究流程图

2.3 研究过程详述

2.3.1 图片处理

首先我们对参考图片进行截取，然后按照相应的格式进行裁剪。截取是根据位置进行的。然后我们对于截取的图像进行 resize 成 240*238 大小的图片，然后进行裁剪。

裁剪图像结果如图 2.3.1 所示（以 6×6 为例）：

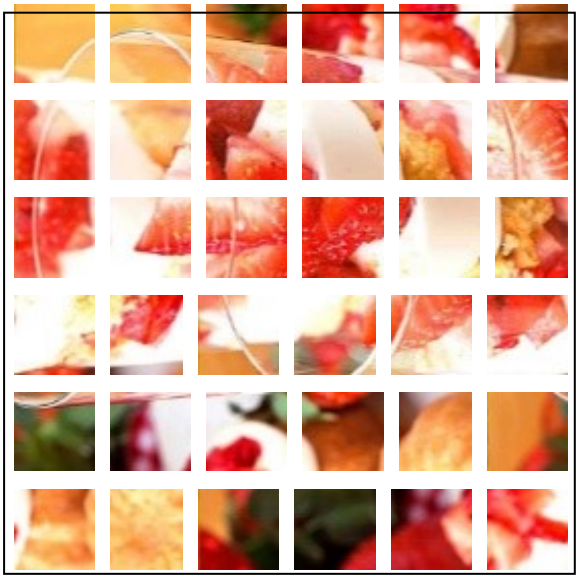


图 2.3.1 裁剪结果图

2.3.2 图片匹配

根据 3.1 中裁剪的图像图 2.3.1，首先进行缩放。将裁减的图片大小缩放成与要匹配碎片的大小一样。这里我们经过测试，缩放比约为 1.3。然后采用 cv.TM_CCOEFF_NORMED 方法进行模板匹配。

模板匹配是在一幅图像中寻找一个特定目标的方法之一。这种方法的原理非常简单，遍历图像中的每一个可能的位置，比较各处与模板是否“相似”，当相似度足够高时，就认为找到了我们的目标。

在 OpenCV 中，提供了相应的函数完成这个操作。

- matchTemplate 函数：在模板和输入图像之间寻找匹配,获得匹配结果图像
- minMaxLoc 函数：在给定的矩阵中寻找最大和最小值，并给出它们的位置

相似度衡量：

a) 差值平方和匹配 CV_TM_SQDIFF
b) 标准化差值平方和匹配 CV_TM_SQDIFF_NORMED
c) 相关匹配 CV_TM_CCORR
d) 标准相关匹配 CV_TM_CCORR_NORMED
e) 相关匹配 CV_TM_CCOEFF
f) 标准相关匹配 CV_TM_CCOEFF_NORMED

我们采用的是：标准相关匹配 CV_TM_CCOEFF_NORMED。这是 OpenCV 支持的最复杂的一种相似度算法。这里的相关运算就是数理统计学科的相关系数计算方法。具体的说，就是在减去了各自的平均值之外，还要各自除以各自的方差。经过减去平均值和除以方差这么两步操作之后，无论是我们的待检图像还是模板都被标准化了，这样可以保证图像和模板分别改变光照亮不影响计算结果。计算出的相关系数被限制在了 -1 到 1 之间，1 表示完全相同，-1 表示两幅图像的亮度正好相反，0 表示两幅图像之间没有线性关系。计算公式如下：

$$T'(x,y)=\frac{T(x,y)-\frac{1}{w\times h}\sum_{x',y'}T(x',y')}{\sqrt{\sum_{x',y'}T(x',y')^2}}$$

$$I'(x,y)=\frac{I(x,y)-\frac{1}{w\times h}\sum_{x',y'}I(x',y')}{\sqrt{\sum_{x',y'}I(x',y')^2}}$$

$$R(x,y)=\sum_{x',y'}(T'(x',y')\times I'(x+x',y+y'))$$

匹配的结果如图 2.3.2 所示：

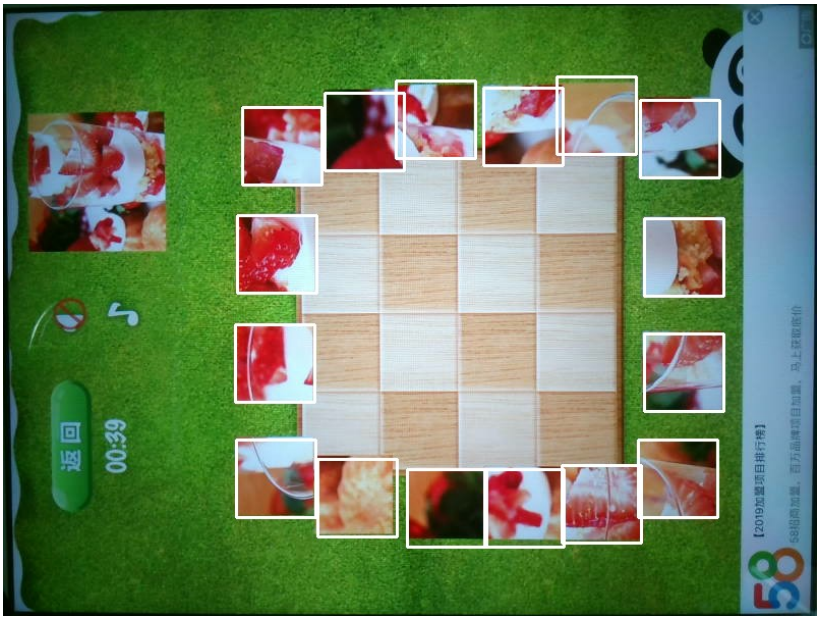


图 2.3.2 匹配结果图

通过匹配得到散落碎片的初始位置。

如果我们先将图像都转换为灰度图，那么计算速度会快很多。

2.3.3 机械臂移动

2.3.2.1 获取目标位置

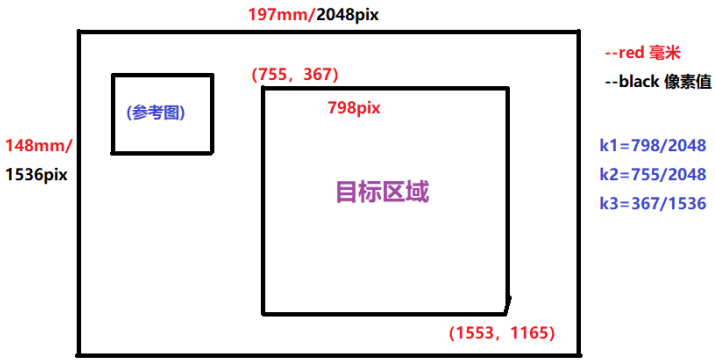


图 2.3.3 图像测量结果图

首先我们根据测量结果（如图 2.3.3 所示）获取目标区域的位置。根据测量得到的 k1、k2、k3，我们经过两层循环可以得到目标区域的位置。其伪代码如下：

```
double k1 = 798.0 / 2048.0;//目标板占整个屏幕的比例
double k2 = 755.0 / 2048.0;//目标板左上角坐标 x 的比例
double k3 = 367.0 / 1536.0;//目标板左上角 y 的比例
double deta = img->width*k1 / num; //碎片的宽度&长度
for (int i = 0; i < num; i++) {
    for (int j = 0; j<num; j++) {
        TarPosMat[i][j].x = (k2 * img->width) + i * deta+0.5*deta;
        TarPosMat[i][j].y = (k3 * img->height) + j * deta+0.5*deta;
   qDebug()<<"targetposition"<<TarPosMat[i][j].x<<""<<TarPosMat[i][j].y;}}
```

通过上述代码，我们可以得到目标坐标矩阵 TarPosMat[][]。

2.3.3.2 机械移动

机械移动，首先需要将像素值坐标转化为毫米坐标单位。如图 2.3.3 所示，在实际检测过程中传回的图像的大小为 800×600，故，相应的比例系数为 m1 = 197.0 / 800;m2 = 148.0 / 600。再根据机械臂的 xy 轴方向与图片 xy 周方向进行坐标转换，得到最终的坐标：init_position[][]，target_position[][]。机械臂按照如下流程执行操作：

1）执行抬起；

2）执行移动到 init_position[][]；

3）执行放下；

4）执行移动到 target_position[][]；

循环上述四个步骤直至所有图片均拼接完成。

拼接效果见第三部分。在这部分，我们在实践过程中发现，若两次移动之间距离差较大，则移动精度下降，为解决这个问题，我们主要进行了如下两个改善：

1）对于相距较远的移动，我们将其拆分成两个动作来完成，中间间隔一个较短的时间。

2）对于相距较远的移动，我们将机械臂移动后停留的时间加大，以给机械臂足够的时间到达目标位置，减小因为时间不够造成的误差。

3）引入初始位置调整，即让机械臂的初始位置为 ipad 的中心，从而让第一次移动的距离变小。

在实践过程中，我们发现拼图时对于靠近 ipad 两段的碎片定位效果不是特别理想，定位均偏向内部。经过思考，我们发现这是由摄像头是弧形的会对像素值相对与距离进行压缩，故我们对于靠近两段的距离进行展开，使之与像素值成线性关系。从而解决了两段定位不准的问题。

2.4 在线调试

我们的算法将基于模板匹配的拼图识别与开环机械臂控制结合在一起，经过理论的尝试及各部分的调整测试之后，我们对整体的拼图过程进行了较长时间的在线调试，以期大大减少识别及移动错误率，得到较好的整体效果。

在调试过程中，由于我们的程序本质上是开环控制，对整个系统的各个参数的精度要求较高，因此我们在后期将大量时间花在了调节摄像头位置、ipad 位置调节以及电容笔初始位置设置等问题上。

2.5 存在的问题及算法改进

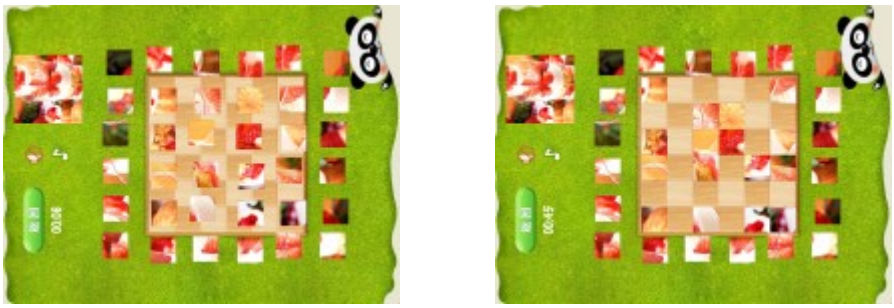
2.5.1 存在的问题

根据上述算法，机械臂可以很高效的完成 4×4 以及 5×5 的拼图，但是对于 3×3、6×6，会出现不能完成的情况，原因是在 3×3、6×6 的界面中，存在原始碎片在目标区域内，在移动过程中可能会造成覆盖。

2.5.2 算法改进

为解决覆盖问题，我们根据图像匹配后的 init_position[][]对碎片进行分类，根据图 2.3.3 中的数据进行判断，若初始位置在目标区域内，则放入 list_in 当中，否则放入 list_out 中。

首先对 list_in 里面的碎片进行处理，首先将碎片移动到其最近的目标区域内的小格中，并更新这些碎片的坐标效果如下：



(a)原始图片　　（b）微移动后的图片

图 2.5.1 目标区域内碎片微调整

然后，再对 list_in 中的碎片进行移动。移动方式为：对 list_in 中每个元素依次出队，若该元素的目标位置为空，则将该元素由当前位置移动到目标位置；若该元素的目标位置不为空，则将该元素加入队尾，并取出队头下一个元素重复上述操作。循环该操作直至队列为空，则结束对初始位置在目标区域内的拼图碎片的操作，开始执行初始位置在目标区域外的拼图碎片的移动。

但是在该循环过程中，可能存在两个或者三个拼图碎片分别占据这几个个碎片的目标位置，导致他们在不断的循环中都无法改变自己的位置。为了避免这种情况，我们建立了一个 flag，flag 用于反映队列 list_in 的长度是否变化。若对队列中每个元素都遍历一次之后，队列长度不变，则说明出现上述死循环过程，flag=1，队列长度即为死循环的拼图碎片数目。此时，随机选取队列中的一个元素将其移动到任意一个空目标位置，之后继续执行上述操作。如此不断循环，最后即可实现队列长度为 0，且初始位置位于目标区域内部的所有拼图碎片都到达自己的目标位置。该算法的流程图见下图。

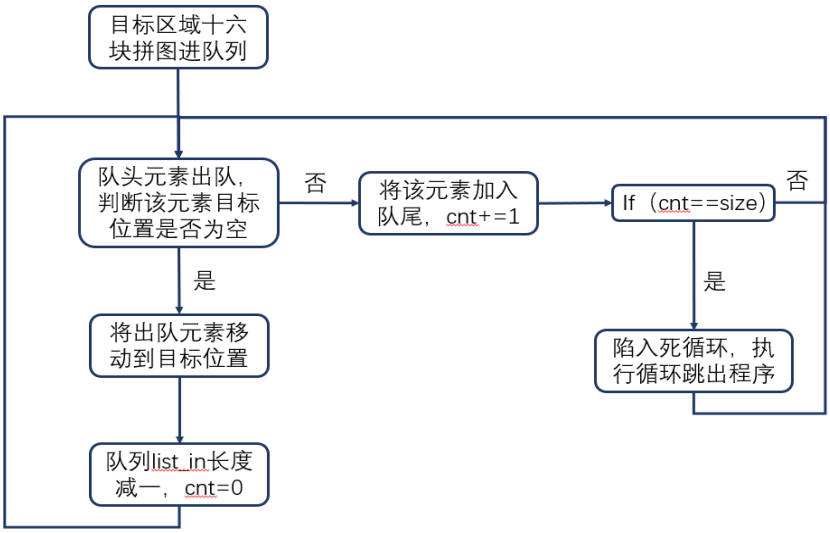


图 2.5.2 算法流程图

2.6 对程序鲁棒性的探索

2.6.1 自动识别屏幕区域算法

为了避免我们的代码对摄像头摆放位置产生过高的依赖，我们设计了自动识别屏幕区域的算法。我们观察到图中的 ipad 屏幕四周有一圈黑框，而黑框外面的白色边框与之产生先明的对比。我们可以利用此进行屏幕的识别。

最简单的想法就是检测黑色边框上的四个角点，然后连接起来，就是一个标准的矩形。而识别角点，我们的算法是，从外向内依次检测各个像素点，检测到的第一个黑色像素点即是角点。具体地讲，加入我们要识别图片中左上角的角点，我们可以在图像左上放选取一个足够大的正方形，使其必定包含角点。然后讲正方形中的点按照从左到右，从上到下的方式遍历，检测到第一个像素点就视为角点。如示意图所示。

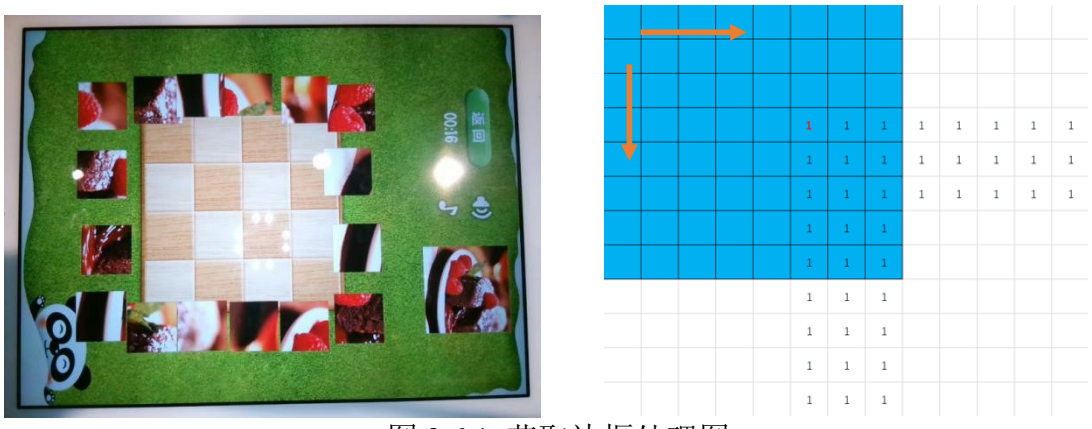


图 2.6.1 获取边框处理图

但这种方式会出现如下问题，算法检测到的角点位置发生了偏移，这显然远远超出了我们误差容忍的范围。最初我们对这种现象发生的原因感到很疑惑，直到我们逐像素点查看数值之后，才发现了问题所在。因为摄像头获取到的图像没

有直接从 ipad 上截图那样精确，难免会存在几个像素的误差。人眼无法分辨如此小的误差，但计算机比较像素点的时候就会发现。以示意图为例，边框本来是标准的直线，但摄像头的误差难免会使某些黑色像素点偏移，就像地面上凸起的小疙瘩。如果不改进我们的算法，这些小疙瘩就会被误以为是角点。

我们想到了一种精妙的改进方法。以示意图检测左上角的角点为例，按照从左到右，从上到下的方式扫描，检测到黑色像素点时，并不立即返回，而是探测该点左下角的点。如果左下角的店仍为黑色，则直接开始扫描下一行，循环上一步骤，即如果发现新的黑色像素点，再检测其左下角的点，直到左下角不为黑色，才认为该点是真正的角点。

这种方法很好的解决了边框识别不准的问题，如有很好的鲁棒性。

但后续过程中我们发现，识别出来的区域并非标准矩形，这并非我们识别算法的原因，而是摄像头本身存在的图像扭曲导致的。其导致我们获取的图像产生一定的弧度弯曲，并且摄像头的高度也会改变其弯曲程度。这样一来，识别屏幕区域这一功能都没有了多大的必要，因为就算区域识别对了，不同高度的弯曲程度也会使我们的参数发生变化。所以我们不如直接固定摄像头高度以确保参数的正确性。

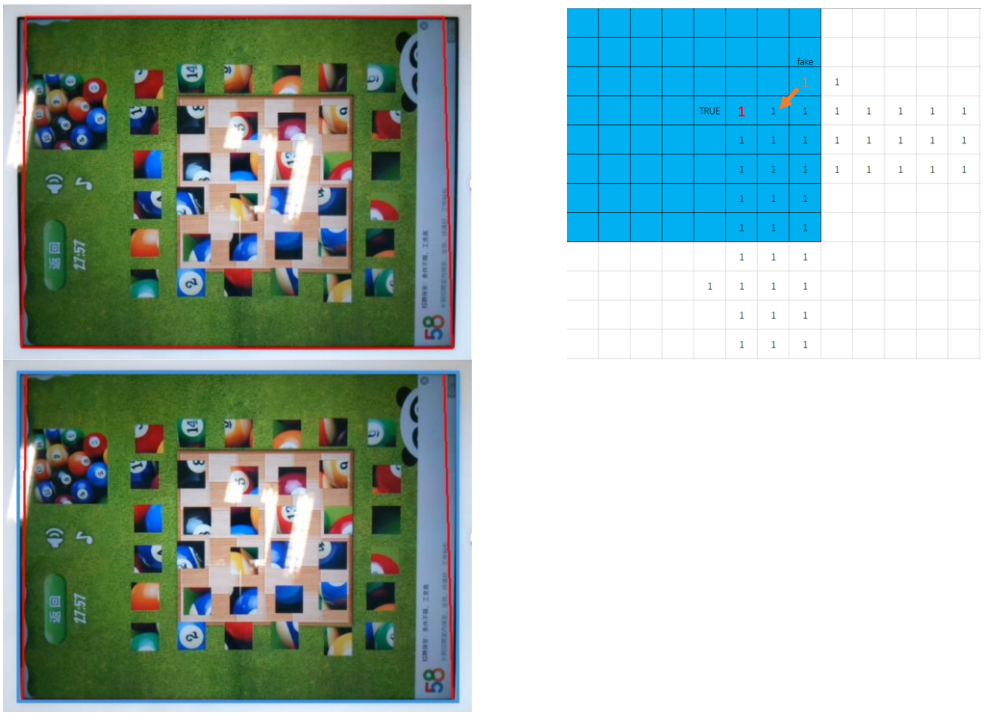


图 2.6.2 改进后的算法（蓝框）与最初算法（红框）对比

2.6.2 闭环控制的改进

经过我们的调整，机械臂的控制精度已经达到了很高的水平，本无需进行闭环控制，但闭环控制是自动化系学生的基本要求和尝试。因此我们后期将精力主要放在程序的闭环控制上。

由于我们的拼图游戏是一块一块进行的，如果我们每拼一块就进行一次闭环校正，整体速度上会严重拖慢。而且摄像头必须在机械臂移开的时候才能完整地看到游戏图像，因此机械臂来来回回之间就浪费了时间，增大了误差。所以我们最终决定，在所有拼图全部拼完之后，再来一次闭环校正。即

先用摄像头读取图像，计算拼图碎片与标准网格的偏差，检查碎片是否挪到最近的框中，如果没有，就执行挪动操作。再用模板匹配的方式检查拼图碎片的位置是否对应，如果不对应，则进行置换。如果上诉两个检查皆无误，则认为拼图完成。如果存在问题，则处理完成后进行下一轮循环，即再执行一遍上述流程，直到检查无误。

3. 研究成果

在选取了合适的拼图软件之后，我们对拼图过程逐步实现了如下成果。

第一， 能够在摄像头拍摄画面内成功截取到 ipad 屏幕部分（以屏幕四周黑色边线为准）。这部分我们写了屏幕截取的代码，但在最终代码中，没有采用，因为我们发现这个截取屏幕的算法会增加成功的不准确度，所以我们最终选择手动调节的方法来降低不确定度。

第二， 根据图像的阈值分割以及矩形分割等操作在截取后屏幕图片的左侧30%找到拼图的对比图。对比图截取如下图，截取后将该图存在本地，用于后续的操作。

第三， 找到标准图像后，将图像分为 n*n 块，依次选取其中的每一块在整个屏幕图像中找到相同的图像块进行裁剪，并确定移动碎片的初始位置。根据标准碎片的位置，就可以确定移动碎片的目标位置。

第四， 操作机械臂移动碎片。

第五， 重复上述第三四步骤，直到所有碎片都被移动，即完成拼图游戏。

按照如上过程，我们完成了 3*3，4*4，5*5，6*6 的拼图游戏。其中在 6*6 的拼图中由于初始时中间会有很多碎片影响后续移动，为此我们设计了用于储存识别到的拼图碎片的两个队列，一个队列用于初始位置放在目标区域外的拼图块，另一个用于初始位置放在目标区域内的拼图块，先对目标区域内的拼图块进行相应的操作。动态演示过程见视频。为检测我们算法的普适性，我们测试了很多个拼图，证明图像模板匹配结果很理想。

图 3.1 实验结果图

4. 研究总结

4.1 研究中存在问题、建议及需要说明的问题

4.1.1 存在的问题

1. 硬件方面：硬件存在针头震荡，且每次下笔深度存在误差
2. 软件方面：没有释放内存，运行时需要消耗大量的内存，以至于我们再后续完成 6×6 拼图时，需要使用大约 9G 内存才能跑完。
3. 镜头方面:镜头存在弧度，所以会造成像素帧在实际距离上的压缩问题，我们通过将平面拓展来解决这个问题。
4. 相对位置固定问题：摄像头支架之间的相对位置对于实践结果的影响还是很大的，如何保证每次两者之间的相对位置是固定的很重要，我们主要通过笔的高度来确定摄像头的高度，并保持摄像头平面与平板平面平行来保证每次相对位置基本固定。

4.1.2 个人建议

1. 建议软件方面能够解决内存问题，这个问题不仅影响到能否运行完全，还影响到闭环的使用。
2. 建议一组提交一份实验报告，因为我们是共同完成课程实践的，每个人都参与了各个环节，所以大家的报告相似度都很高，所以写三份的意义不大。

4.2 在本次课程实践中的感想与体会

本次课程主要考察的是数字图像处理能力和控制算法能力。我们选择的游戏是拼图游戏，主要考察的就是拼图的图像匹配以及控制机械臂。拼图匹配我们采用的是最基础的模板匹配，根据参考图片来对目标区域的图片进行模板匹配，找到最适合的匹配结果，然后再将其通过控制机械臂移动到正确的位置。我们采用的是开环控制逻辑来进行，鉴于视频流的处理问题，我们主要使用第一张图来进行判断。根据一次读图完成所有的操作，然后再根据操作后的结果再次读图进行校正。我们在实践过程中，遇到了内存等诸多问题，在大家的帮助下都得以顺利的解决。

本次课程实践整体进行的还是非常顺利的，项目完成地很高效。对于图像匹配有了实践的认知。加深了对其的理解。十分感谢老师、助教以及同学在实践过程中给予我们的帮助。

4.3 对本次课程实践的意见与建议

首先我认为本次课程安排的内容还是很充实的，考核方式也很多样，既有理论知识的考核，又有实践能力的测试，有助于同学们对于数字图像处理的理解与掌握。但是我认为大作业可以在稍微早些的时 3 候布置，这样可以适当的减缓期末的作业压力，有助于课程实践更加高质量的完成，也能够

让同学通过实践对于理论知识有更深的理解，从而能够更加好的应对理论考试。

参考资料：

[1]. 梁佳楠,李丽丽,周磊.基于图像处理的位置坐标获取方法应用[J].自动化与信息工程,2018,39(03):22-25.
[2]. 贾迪,朱宁丹,杨宁华,吴思,李玉秀,赵明远.图像匹配方法研究综述[J].中国图象图形学报,2019,24(05):677-699.
[3]. 卜飞宇.一种基于边缘直线检测的矩形提取方法[J].电脑知识与技术,2017,13(31):182-184
[4]. 孙晨阳,周廷刚,陈圣波,沈敬伟,王骏飞,杨桦.基于矩形模板匹配的线状地物半自动提取方法研究[J].西南大学学报(自然科学版),2015,37(07):155-160.
[5]. 卢蓉,范勇,陈念年,王俊波.一种提取目标图像最小外接矩形的快速算法[J].计算机工程,2010,36(21):178-180.
[6]. https://blog.csdn.net/liyuanbhu/article/details/49837661
[7]. http://blog.csdn.net/qq_15947787/article/details/51085352
[8].https://blog.csdn.net/wsp_1138886114/article/details/82945328