高二 集体项目（3人）

**多机器人协作搭七巧板**

**（结题报告）**

2008年3月

摘要：

多机器人协作完成精细的动作，其必要性正日益在人们的生活生产过程中凸现。而多机器人之间进行自动而合理的任务分配，也成为生产过程中一个急需解决的问题。多机器人完成精细的动作，也分不开精确的控制。本项目以七巧板拼图为应用对象，通过视频和无线通讯解决多机器人协作和较高精度控制的问题。

本项目的创新点还体现在，改进轮廓跟踪算法，研究实时化的图形向量化算法和直线搜索算法；改进颜色标识算法；引入负反馈调节和放弃机制；通过无线电数据传输进行控制。

本项目可直接用于机器人演示、教学、人机互动娱乐等用途。

**目录**

[1. 选题目的 3](#_Toc191822018)

[2. 技术与方法 3](#_Toc191822019)

[2.1. 流程 3](#_Toc191822020)

[2.2. 颜色标识和轮廓提取 4](#_Toc191822021)

[2.2.1. Flood fill算法 4](#_Toc191822022)

[2.2.2. 色差判断 5](#_Toc191822023)

[2.3. 使用凸包求多边形顶点 7](#_Toc191822024)

[2.4. 算法复杂度 3](#_Toc191822025)

[2.5. 路线规划算法 9](#_Toc191822026)

[2.5.1. 实时路线规划 9](#_Toc191822027)

[2.5.2. 机器人互相避让 10](#_Toc191822028)

[2.6. 其他描述约定 10](#_Toc191822029)

[2.6.1. 无线通讯 10](#_Toc191822030)

[2.6.2. 目标图形描述 10](#_Toc191822031)

[2.7. 负反馈调节和放弃机制 11](#_Toc191822032)

[3. 机械结构 11](#_Toc191822033)

[4. 创新点陈述 11](#_Toc191822034)

[参考资料 12](#_Toc191822035)

1. 选题目的

多机器人协作完成精细的动作，其必要性正日益在人们的生活生产过程中凸现。而多机器人之间进行自动而合理的任务分配，也成为生产过程中一个急需解决的问题。多机器人完成精细的动作，也分不开精确的控制。以往采用的脱机方式，机器人执行固定的代码或运行固定的时间，这有可能将机械偏差通过多次重复而增大。本项目以七巧板拼图为应用对象，通过视频和无线通讯解决多机器人协作和较高精度控制的问题。

本项目可直接用于机器人演示、教学、人机互动娱乐等用途。

1. 控制端计算机程序
   1. 项目总体流程

本项目使用七个小机器人，各背负七巧板中的一块。操作者在计算机上通过一个游戏型的友好的用户界面，使用鼠标拖拽、旋转七巧板，拼出一个目标图形。其操作的最终状态将被记录，作为机器人拼图的最终效果。计算机通过无线数传模块与七个机器人通讯，控制其按照运算出的路径平移、旋转，最终在场地上拼出目标图形。

* 1. 算法复杂度分析
     1. 空间复杂度分析

考虑每帧图像的分辨率为*M*×*N*，识别过程中的一系列算法从本质上来说不会增加空间复杂度的阶，所以科技发展的为*O(MN)*。相对于目前计算机的内存性能，空间复杂度不会是瓶颈所在。

* + 1. 时间复杂度分析

算法的时间复杂度是本项目的关键之一，考虑到项目的实时性，以下首先估算算法的时间复杂度上界。

将图像的识别过程抽象为对一个*N*阶方阵的操作，考虑一般情况下，目前计算机的每秒指令数为109数量级，简单计算可知，识别算法的复杂度不能超过*O(N3)*。为了保证程序流程的运行，更为理想的时间复杂度为或。

图像预处理的时间复杂度为，Flood fill图像分块为，特征提取算法（凸包，最小覆盖点集）为。

* 1. 颜色标识和轮廓提取（预处理和Flood Fill）
     1. Flood fill算法

与标准的填色Flood fill类似，形状识别所采用的Flood fill的基本思路也是从一个给定的点出发，在一个等大小的数组中作数字填充，将颜色相近的像素点在对应位置上标上相同的标号。

七巧板的七块板用七种颜色的布覆盖，相互之间与背景颜色保持一定的区分度，记作。摄像头拍摄到当前的机器人情况，计算机端的控制程序通过DirectShow获取视频信号，采用C#的非托管代码，以指针的方式直接读取内存中的像素颜色信息。随机选取一个点开始采用Flood fill算法进行近似颜色广度优先搜索，同时记录颜色近似的点的个数。当（最小有效色块面积）时，保存边缘上点的集合为。

为判断当前点是否在边缘上，在Flood fill算法中广度优先搜索扩展状态信息之后，判断与当前点颜色不相近的四联通像素个数，则将当前点加入集合。

Flood fill算法一般采用递归实现，但由于图片上的点数量众多（320×240），很容易导致系统堆栈溢出。因此采用更为节省内存，同时效率也有所提高的队列式Flood fill算法。其步骤是：

1. 将起始点加入队列；
2. 弹出当前队列中的第一个点，读入P中；
3. 依次读取P的四联通像素点的颜色信息Pi，如果Pi与P的颜色相差小于，则将Pi加入队列尾，否则p自加一；
4. 面积计数器自加一；
5. 如果队列非空，回到2，否则退出。

伪代码如下：

int Floodfill(int x0, int y0, Color clrStandard, out **Collection**<point> E)

{

int sze = 0; **Queue** q;

while (q.Count > 0)

{

point P = q.Dequeue();

point Pi;

int p = 0;

foreach (**direction** in **directionList**)

{

Pi.X = P.X + **direction**.X;

Pi.Y = P.Y + **direction**.Y;

if (!**sign**[Pi])

{

if (**ColorDifference**(Pixel[Pi].Color, clrStandard) < ClrDiffMaxS)

{

q.Enqueue(newPt);

**sign**[Pi] = **Code**;

}

else

{ p++; }

}

}

if (p>=1) E+={Pi};

sze++;

}

return sze;

}

由于所有图形的面积都大于，在选取Floodfill起始点的时候可以不必逐个像素地尝试；我们可以将起始点坐标按的步长进行查找。

* + 1. 色差判断

由于摄像头本身的限制和着色时的偏差，以及光线明暗变化，色值总不能保持严格相同。因此需要获取两种颜色之间的差值。建立一个R-G-B坐标系，用有序整数对表示一种颜色的红、绿、蓝色分量数值，其中。则对于某种特定的颜色，必然有且仅有一个点在R-G-B空间中对应，如图所示。两种颜色的色差值则是边长为255的立方体范围内两点之间的距离[1]。

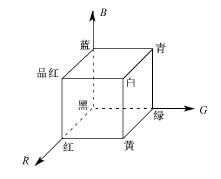


图 1

由颜色数值的意义我们可以知道：



两个色值对应点的距离



将代入，即可算得两个色值的色差。

这样的方法实现简单，但对环境光线的要求比较高。受到12色相环的启发，我们可以通过数学变换的方法，将一种颜色值转换为所在色系的基准颜色。而色彩的HSV空间表示法[1]正好可以满足这种需求，如[2]所示。

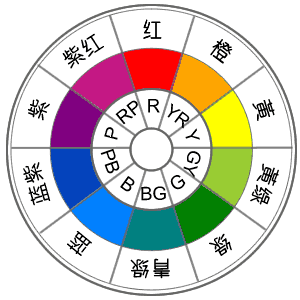


图 2



其中，*H*的取值决定了颜色所处的色系。只需给定，即可确定颜色是否大致属于同一色系（）；同样地，也可以较为直观地确定V的取值。

上述颜色标号和轮廓提取算法的时间复杂度为，其中为图片的宽，为图片的高，下同。

* 1. 使用凸包求多边形顶点[3,5,6]

采用所述方法，可以获得一个连续封闭区域。但是由于色差和其他不可控制的情况，所求得的封闭区域可能存在缺损，这种缺损将极大地影响后续的工作，如所示，右边图像的白色部分是原图（左）经过Flood fill后被填充的区域。

其中，，，



上式再一次证明，使用HSV颜色空间，可以比RGB空间更好地反映颜色的色系。

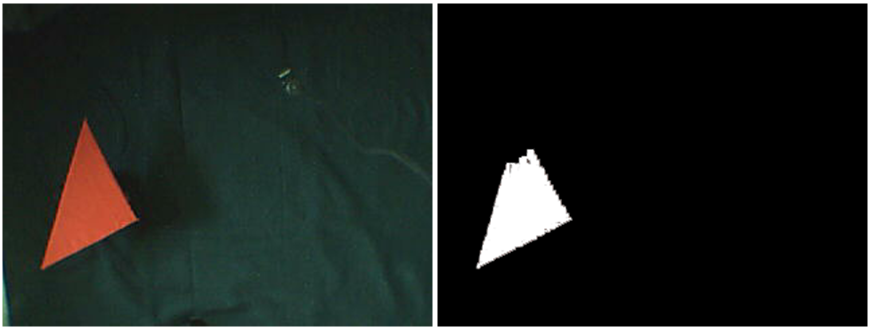


图 3

同时，求得的边缘集合*E*是无序的，它无法直接运用于求多边形顶点的过程。已知点集求得直线，较为成熟的方法有Hough变换[3]等方式。但是经过试验[[1]](#footnote-2)，该方法虽然较为经典，但准确率有较大问题。这与图像的对比度和边缘检测的精度是有关系的。采用Prewitt算子[4]等方法虽然可以有效地获取一条较为理想的轮廓线，但求局部最大值的方法会导致多余直线的产生[[2]](#footnote-3)或（由于Hough变换本身的特性）遗失。

采用Flood fill结合求凸包的方法，就可以方便地获取顶点序列*D*。点集*E*的凸包（convex hull）是指一个最小凸多边形，满足*E*中的点或者在多边形边上或者在其内。下图中由线段表示的多边形就是点集*E*={P0, P1, ... P12}的凸包。

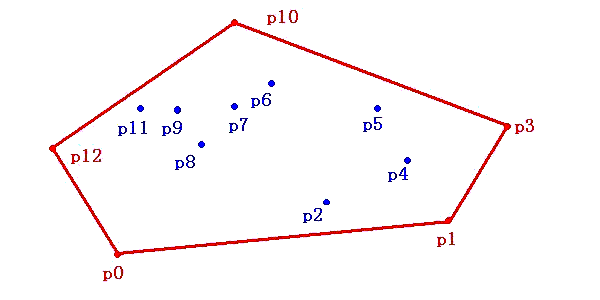


图 4

凸包最常用的凸包算法是Graham扫描法和Jarvis步进法。较为常用的是Graham扫描法。对于一个有三个或以上点的点集Q，过程如下：

计算点集最右边的点为凸包的顶点的起点，如上图的P3点;

//对点集进行排序（先按y坐标顺序，y坐标相同的则按x坐标排序）

Sort(Points);

//凸包集堆栈预先设为空集

Poly.Points=0;

//对每个点进行处理

for (i=1; i<Points; i++)

{

//如果当前凸包上的点的个数≥2且

//凸包点集栈顶前两个点与当前点构成左手螺旋，则该点加入点集堆栈

//否则不断弹出栈顶元素，直到找到或者点个数小于2为之

}

本项目具体在实现时采用的是改进的Graham扫描，原先的标准算法是需要对极角进行排序的，而改进的算法之需要对坐标进行排序，通过一次从上往下的计算便可求得凸包。本质上并没有降低复杂度，但实现起来更为方便。

下面引入对点集求凸包的一个重要性质，这关系到总体识别算法复杂度的有效降低，在第五节讲给与更深入的分析。

性质1 定量研究表明：k维球体中均匀独立分布的N个点，其凸包顶点数。k=2即为平面时，。可见凸包可以大大降低平均情况时空间复杂度。

所示的图像，经过上述求凸包方法，获取的顶点用蓝绿色标示：

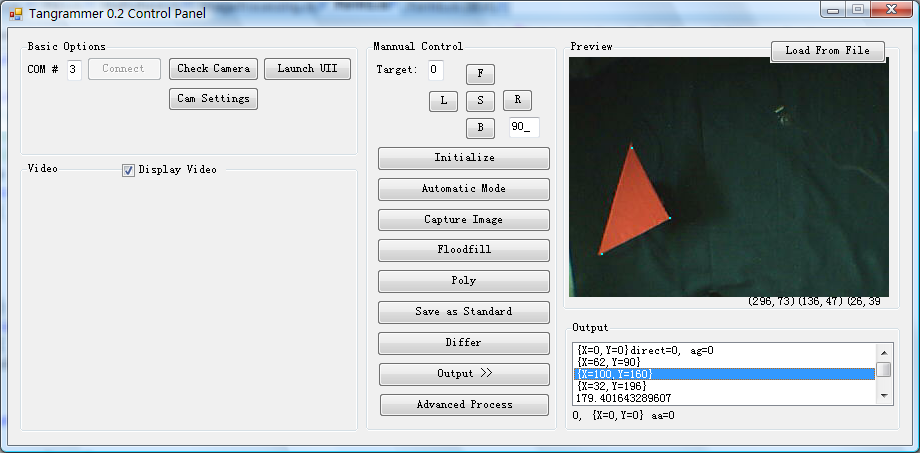


图 5

算法的复杂度为，其中*n*为边缘的个数。经实验验证，一般情况下*n*≤103。

* 1. 顶点合并算法

通过上述方法求得的凸包，只是一个轮廓上的点按照一定顺序的排列。因此可以通过时间复杂度为O(N)的算法对其进行顶点合并，留下图像边缘走向发生明显变化的顶点。

初始化：记起始点*S* = *Q*0。枚举开始。

计算，如果，则说明*Qi*、*Qi*-1、Q0三点共线。考虑到图像精度限制，修改上述平行条件为：



同时维护边集，其中*S*为起始点坐标，为上述求得的。

如果不再成立，则令，重复上述操作。

实际上每次只需针对最新加入队列的点*Qn*进行一次上述操作，维护向量，最终得到边集*VN*，选取其中起讫点连续的两项，即可计算顶角的大小和边长。

将边向量数组*VN*与标准边向量集*VSN*比对，可以计算出标号为*N*的七巧板旋转的角度。将它和标准顶点*S*0的坐标记录为一个四维向量。

* 1. 路线规划算法
     1. 粗略时序规划

综合文献［7，8］关于路线规划的论述，结合是贯穿于整个调节过程中的计算机端行为，路线规划方法如下。

* 获取用户指定的终态图案信息；
* 找出终态图案中的依赖关系，即哪些板块拼接上后，将形成一个包围，使得包围中存在的板块无法从外界进入；
* 优先考虑处于包围中心的板块，在其移位时，其它板块作旋转，准备就位；
* 包围中心的板块到位后，其四周的若干板块再行移位；
* 对当前正在移动的板块，视作以其最长对角线为直径、几何中心为圆心所作的圆；当两圆相切时，随机选择一个板块，先行旋转，就位；
* 重复上述两步，直到所有板块均到位。

进行上述操作时候，得到一个机器人运动顺序和优先级的列表，记作Si。每个元素有两个字段Importance（优先级）和ID（机器人地址）。在按照优先级排列的基础上，对同等优先级的机器人进行下文所述的机器人相互避让方法。

为防止在机器人进行运动前就重复发送指令，规定每个线程在进行一次指令发送后自动暂停40ms。若发送了一个包含延时停止指令的指令序列，则暂停的时间为这些指令执行之间的总和。

* + 1. 机器人互相避让

经过粗略时序规划的机器人，将按照严格优先级的顺序，依次运动。这样，通过时序规划就有效地避免了不同优先级的机器人相互碰撞的可能。然而，对于多个相同优先级的机器人（这样的情况可能而且一定会发生），通过预先的时序规划不一定能解决问题。

与生物学相似，本系统中机器人的相互避让并非两台冲突机器人相互拮抗，最终形成动态平衡时的决策结果，而是在对抗过程起始前就已决定的[9]。这是因为机器人之间的相互通讯不仅会增加系统负担，而且会对其他机器人造成不必要的信号干扰。因此我们在计算机的控制程序中就已经考虑了机器人相互避让的问题，使得结果成为唯一确定而且可控的。其具体方式就是，对于两台机器人，如果它们的最小覆盖圆相交，而且它们都没有到位（或已经放弃），则选取其中标号较小的一台沿着与减小和目标差异的反方向行进一段时间（即，原来是前进的则后退，原来是左转的则右转），直到它们的最小覆盖圆不再重合，让后一台机器人先行通过。

通过凸包的顶点集E求最小覆盖圆方程的算法为：

1. 任意选择两点Pi和Pj。
2. 以Pi和Pj为端点，Pi和Pj的中点为圆心，构造一个圆。如果这个圆包括所有的点，那么他就是最小的圆，中点也就是最小圆的圆心；否则选择圆外的一点Pk。
3. 如果Pi、Pj、Pk这三个点形成的三角形是直角或者钝角三角形，那么改表直角或者钝角所对应的两点的名字为Pi、Pj，然后回到2。否则，这三个点形成一个锐角三角形，构造一个外接圆，如果这个圆包含所有的点，结束；否者进入4。
4. 选择一些不在圆内的点Pl，设点Q为{Pi，Pj，Pk}中离Pi最圆的点.连接并延长点Pl和点Q，将平面分成两个半平面，设点R为{Pi,Pj,Pk}中与Pl不在一个半平面中的点。得到Pl、Q、R三点返回3。
5. 机器人控制
   1. 无线编码

计算机测得七巧板的信息后，对其行进状态作指示。通过无线数据传输模块，向机器人广播一个八比特的数据，各部分数据的含义在此作说明：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位序 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 含义 | 目标机器人编号 | | | 行进状态控制  00000－停止 00111－占位 00001－前进 00010－后退 其他－保留 | | | | |

表1

程序控制计算机的COM口（运用.Net Framework的System.IO.Port命名空间的SerialPort类，直接对COM口进行操作），通过无线数传模块发送一个0～255的整数，机器人收到后对其按照上述规则解码。首先截取前3位，若与自身的编号相符则执行，否则自动忽略。

将机器人控制接口对外封装为一个静态类（Static Class），外部程序直接调用其方法，而隐藏了对端口的操作和具体指令代码。

机器人将所收到的、发给自己的数据保存到一个较小的指令堆栈中。则执行一条指令，弹出当前指令，继续执行下一条。收到指令后，机器人给计算机一个与自己编号有关的回馈信号，提示计算机不必再发送同样指令了。一个指令周期结束。

* 1. 目标图形描述

计算机程序将用户给定的目标图形转化为一个四维向量。其中*i*表示七巧板颜色标号，表示与标准位置顺时针方向的最小夹角。

* 1. 负反馈调节和放弃机制

负反馈机制是指一个系统的输出增加的信息传送到敏感元件引起这个系统的输出减少[9]。由生物体的负反馈调节机制让我们联想到，可以通过一种较为模糊的在线纠偏方式，减小机器人动作误差，提高准确率。它还更为灵活，可以应对突发的变化。

本项目包括一个实时摄像头，计算机程序通过前述算法并行计算七块拼版的位置和方位信息。将两个向量的对应值相减，即可获知其差值。若，则可认为已经达到了目标；若没有达到，则继续向减少差值的方向运动。

如果经过多次尝试，差值始终在0上下摆动，则说明已经达到机械－时间精度的极限。这时程序将放弃调整，保留误差，以防止陷入死循环。

1. 机械结构
   1. 结构

本项目共有7个机器人，每个机器人的结构相同，背有不同的七巧板形状，每台机器人由一个控制器、两台马达、指南针和一块无限数传模块组成。

* 1. 硬件精度

控制器：使用中鸣套件的控制器 八位单片机

马达：1:120的大力马达，位使机器人运动速度减慢，摩擦力增加，提高精度。

无限数传模块：微功率发射，最大发射功率10mW，FSK调制方式，收发一体。项目采用429.0021MHz频率。

* 1. 姿态运动

机器人可以通过直线运动、原地旋转与直线调整以任何方向到达场地内的任何一个位置。

* 1. 运动方案一（无自身调整）

此方案中机器人完全只接受计算机的信号，根据计算机的及时信号作出相应的运动，运动时间为摄像头拍摄一次的周期，即每拍摄一张图像，机器人运动一次，直至运动结束。

* 1. 运动方案二（含自身调整）

此方案由计算机通过一次图像的情况与终态情况的对比，向机器人发送一次运动的信号，使机器人自己完成一次完整的过程，机器人在其中会通过指南针队自身的姿态进行调整，直至计算机发出终止信号。

* 1. 信号接收

因为项目为多机器人合作项目，所以每个机器人都需要一个自己的地址，我们为每个机器人进行了编号，每台机器人会对自己的信号作出反应，屏蔽其他信号，因为机器人与计算机都使用收发一体无线数传模块，详细的信号编制见3.1。

参考文献

1. ［美］Peter Shirley等．计算机图形学（第二版）[M]．人民邮电出版社．2007.6
2. [张旭明](http://hi.baidu.com/zhang%5Fxuming)．十二色相环与二十四色相环[EB/OL]．http://hi.baidu.com/  
   zhang\_xuming/blog/item/bf53d91b6ec1aad4ad6e75e0.html
3. ［美］Wesley E. Snyder、Hairong Qi．机器视觉教程[M]．机械工业出版社．2006.4
4. 樊娜、李晋惠等．图像边缘检测的Prewitt算子的改进算法[J]．西安工业学院学报．(2005)卷25.1
5. 叶其孝、沈永欢．实用数学手册（第二版）[M]．科学出版社．2006.1
6. 钱焕延．计算机数学基础[M]．南京大学出版社．1996.6
7. 谭民，王硕，曹志强．多机器人系统[M]．清华大学出版社．2005.4
8. 张福学．机器人学——智能机器人传感技术[M]．电子工业出版社．1996.9
9. 吴相钰等．陈阅增普通生物学[M]．高等教育出版社．2005.1

1. Hough变换的思想为，将世界坐标系上的点投射到参数空间中，其中表示直线。增开数组a，其中ar,t表示参数空间中点(r,t)对应直线上的点，存在于世界坐标系xOy中的点的个数。运用统计的方法，求得a的局部最大值即可获取对应直线方程，通过解线性方程组的方法获得顶点的坐标。 [↑](#footnote-ref-2)
2. 原创的求局部最大值的方法为，修改Flood fill算法的扩展条件为“新坐标(r,t)对应的a[r,t]值≥当前点的a[r0,t0]”，换言之，程序将不断搜索新的坐标点，直到新的坐标点对应的数值比四联通中所有的点都要大，停止搜索。将停止搜索时的坐标点(r0,t0)加入新队列Q，该点对应的直线即为所求。将队列Q中的参数对应方程两两联立，即可解得交点，也即多边形的顶点。 [↑](#footnote-ref-3)