2 视觉处理模块

2.1总体流程

项目采用opencv-python轮廓识别与轮廓近似法确定方形轮廓，通过分析轮廓关系实现矫正点定位，使用透视变换得到校正后图像。再使用开闭运算以及最小内接正矩形框定地图，最后使用霍夫圆检测确定各个宝藏点的坐标。

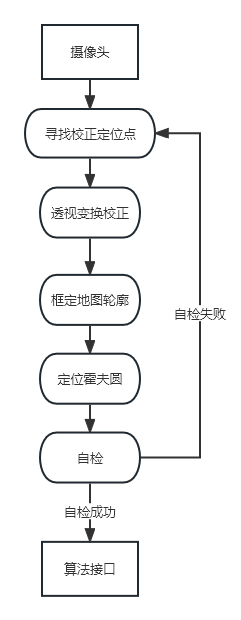


图2.1.1 算法流程

2.2 寻找校正定位点

首先对原始图像进行灰度处理和高斯滤波，并进行二值化，去除高频噪声点和阴影。再采用Opencv轮廓识别方法findContours提取出图中的轮廓，本项目采用轮廓近似法实现各种轮廓的识别，即一条曲线可以由一系列短线段近似，进而可以生成近似曲线，该曲线由原始曲线定义的点子集组成，调用cv2.approxPolyDP即可实现。

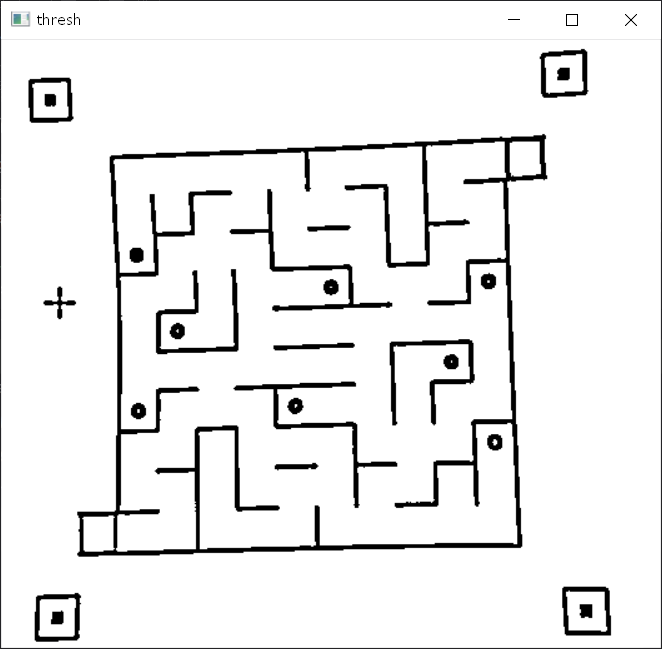


图2.2.1 二值化图像

轮廓是由一系列顶点组成的；如果是三角形，将拥有3个向量，如果有4个顶点，那么是矩形或者正方形，进而计算轮廓的边界框 并且计算宽高比，如果正方形的宽高比为1 ，否则是矩形；否则假设其为圆形以便于后续的圆点识别。

对于提取后的轮廓，使用上述的轮廓检测器提取出满足正方形或长方形条件的轮廓。对于位于四角的定位点，可以发现其内部含有方形小轮廓，故筛选出含有子对象的方形轮廓。又发现其具有并列关系，即都属于某一个较大的父级轮廓，所以通过分析轮廓继承关系树中的父子关系，可以筛选出满足含有子对象且同属于同一对象子对象的轮廓，以上过程使用findContours返回值hierarchy分析即可。

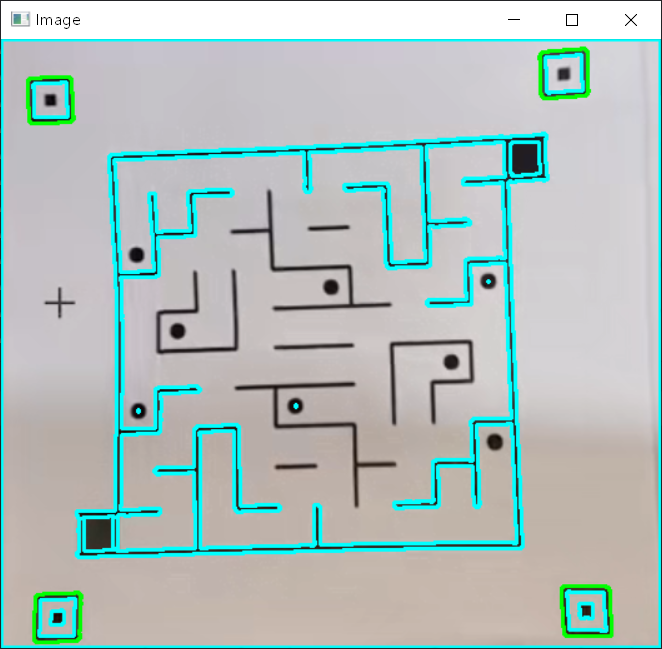


图2.2.2 轮廓搜索

满足以上关系的还有中央的地图轮廓，此时计算五个轮廓面积并去除面积最大的轮廓即可得到四角定位点轮廓，进而使用从cv2.moments计算图像的不变矩，通过不变矩的信息可以计算出图形的质心，依次方法的得到是四个质心即为图像进行透视变换的四个定位点。

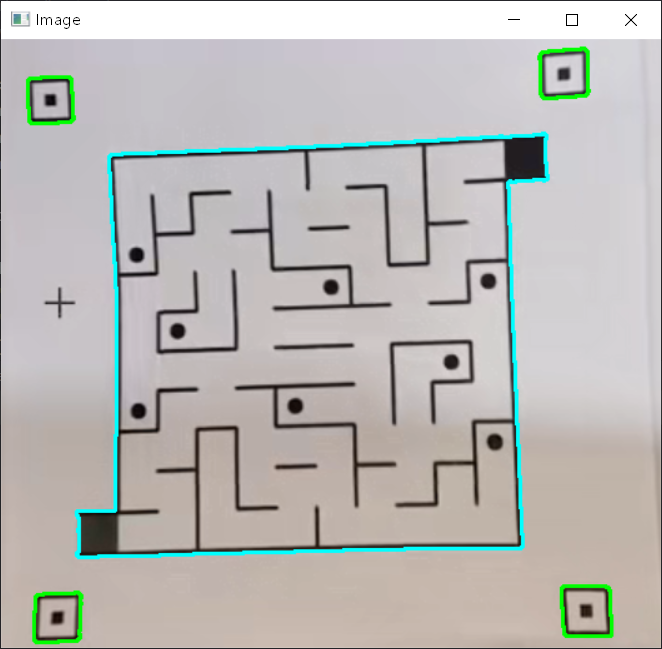


图2.2.3 计算定位点

2.3 透视变换校正

透视校正算法通过求解相机成像的逆变换，将图像恢复到其在三维空间中的原貌。 具体而言，需要确定四个点分别代表图像中一个矩形的四个顶点，然后根据这四个点在真实世界中的位置计算出相机成像矩阵的逆矩阵，对图像进行变换即可得到去除透视畸变的结果。

变换公式为：

由于分析对象为二维图像，令在*z*、*Z*、*a33*值为1，得到方程：

求解上述上面含有8个未知量（a）的方程,我们需要8个像素点，原图像4个，即原图像四个顶点；新图像4个，即上文分析出的四个标志点。

变换后图像如图所示：

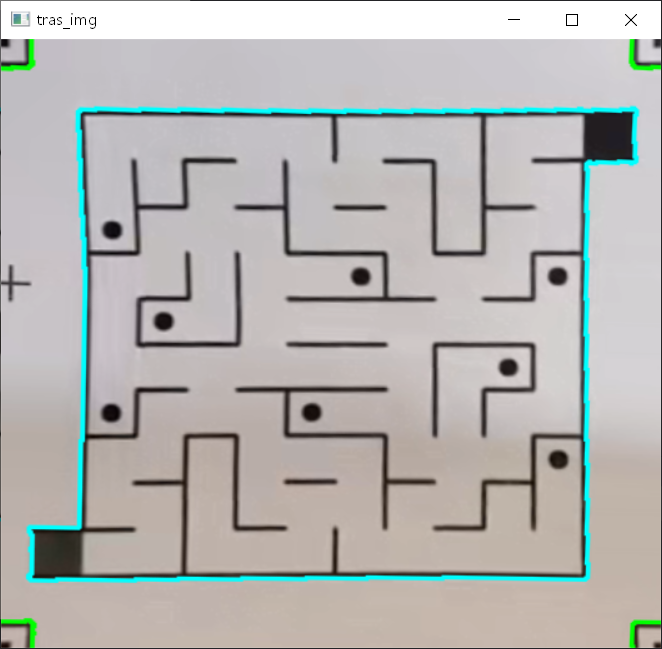


图2.3.1 校正后图像

2.4 框定地图轮廓

得到透视变换校正畸变后的图像后，需要框定地图轮廓并裁剪图像至地图大小方便进行宝藏点坐标相对位置的确定。

本项目采用开闭运算法实现地图轮廓的提取。首先使用numpy定义运算核矩阵，由于地图内黑色较为密集，采用先膨胀再腐蚀的操作使地图形成整体，再使用先服饰再膨胀的操作去除不属于地图的分支。最后沿用上文的方案，即先采用灰度处理和高斯滤波，并进行二值化，定义轮廓检测器筛选出唯一的方形轮廓。

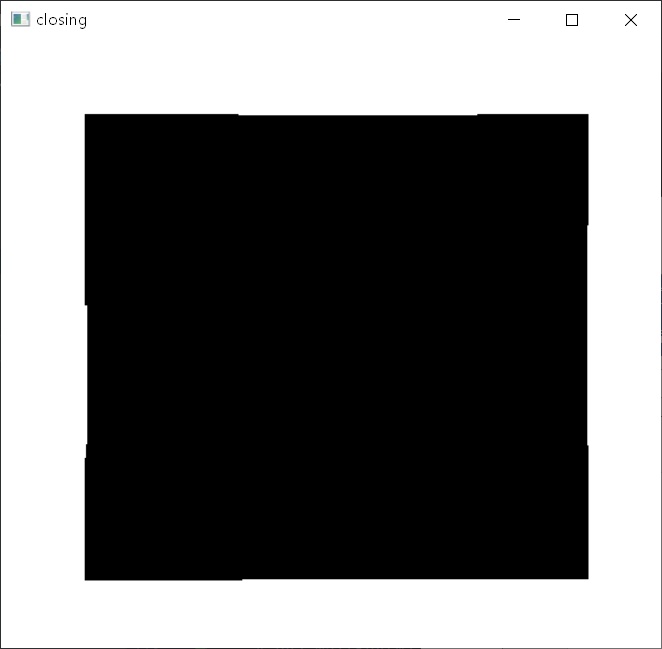


图2.4.1 开闭运算效果

由于获取的方形轮廓不一定为标准的方形，故计算图形的最大正内接矩形，并对矩形的四个顶点进行排序，获取轮廓坐标。最后对图形裁剪进行裁剪即可。

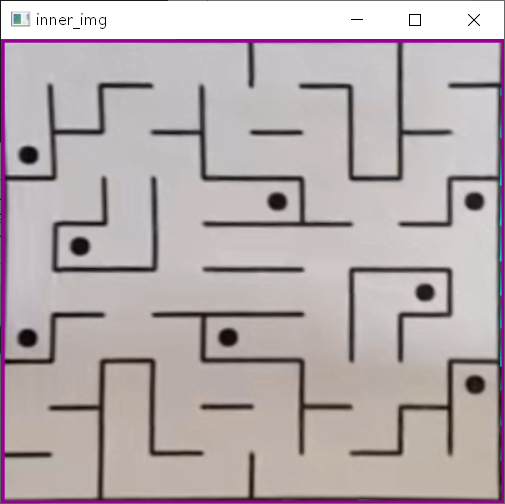
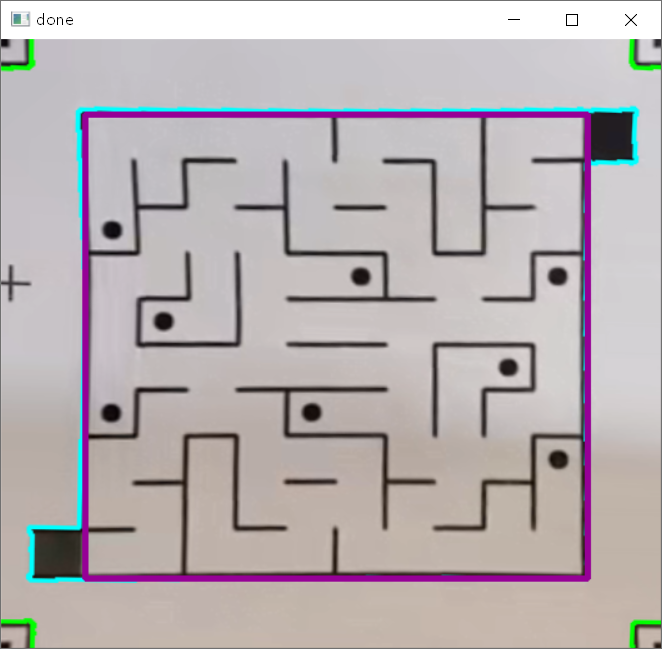


图2.4.2 框定地图效果

2.5 霍夫圆检测

霍夫圆检测法是早期的一种以投票方案进行图形拟合的算法建立霍夫参数三维空间，并对空间内各个单元进行投票，设置阈值从投票结果中筛选合适的圆，并做非极大化抑制，进行调参规定霍夫圆的大小范围后得到8个圆心坐标，此时将圆心坐标与方格区间进行对应基于得到各个宝藏点的坐标。

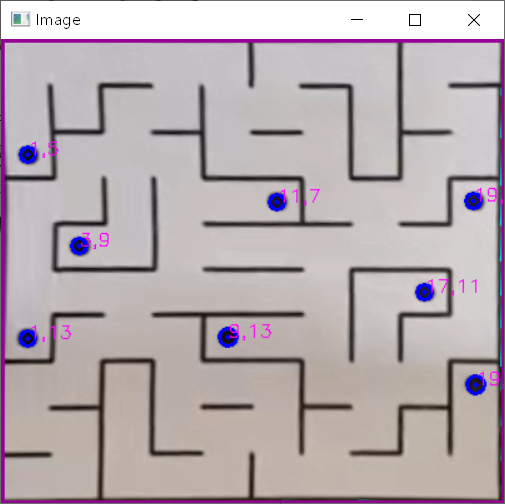


图2.5.1 宝藏点定位效果

3 路径规划算法模块

3.1 地图建模

将迷宫问题模型化，整个地图可以看作中心对称的矩阵，将边界核墙设为1，可以通行的区域设为0，依据opencv的坐标定位方法，将左上角定为(0，0)，右下角定为(20，20)。

首先需要使从地图一点到达地图另一点时间最短，在理想机械设计条件下，横竖运动速度相等，即路径最短。其次需要满足机器人遍历所有宝藏点以及起止点的路径为所有可能情况下的最优解，即旅行商问题（TSP），TSP问题无法采用常规数学方法求解，若采用穷举法，耗费时间非常长，故需要设计一种较为智能的搜索算法。

通过以上分析，机器人迷宫寻宝可以看作内层的深度搜索问题，外层的旅行商问题。

3.2 内层点对点寻路：深度搜索

迷宫是深度搜索算法的典型应用。在深度优先搜索中，对于最新发现的顶点，如果它还有以此为顶点而未探测到的边，就沿此边继续探测下去，当顶点*v*的所有边都已被探寻过后，搜索将回溯到发现顶点*v*有起始点的那些边。这一过程一直进行到已发现从源顶点可达的所有顶点为止。如果还存在未被发现的顶点，则选择其中一个作为源顶点，并重复上述过程。整个过程反复进行，直到所有的顶点都被发现时为止。

1）把初始状态放入数组中，设为当前状态；

2）扩展当前的状态，从合法状态中寻找一个新的状态放入数组中，同时把新产生的状态设为当前状态；

3）判断当前状态是否和前面的状态重复，如果重复则回到上一个状态，产生它的另一状态；

4）判断当前状态是否为目标状态，如果是目标状态，则找到一个解答，根据实际问题需求，选择继续寻找答案或是直接返回。

5）如果数组为空，说明对于该问题无解。

通过以上思想编写算法，依次对可能路径搜索，碰到墙后搜索回溯，直到获取合适的路径。

3.2 外层路径规划：模拟退火

模拟退火算法作为一种启发式算法，有以下两个特点：

1）如果新状态的解更优则修改答案，否则以一定概率接受新状态;

2）有助于加快求解速度。

模拟退火算法的流程图如下：

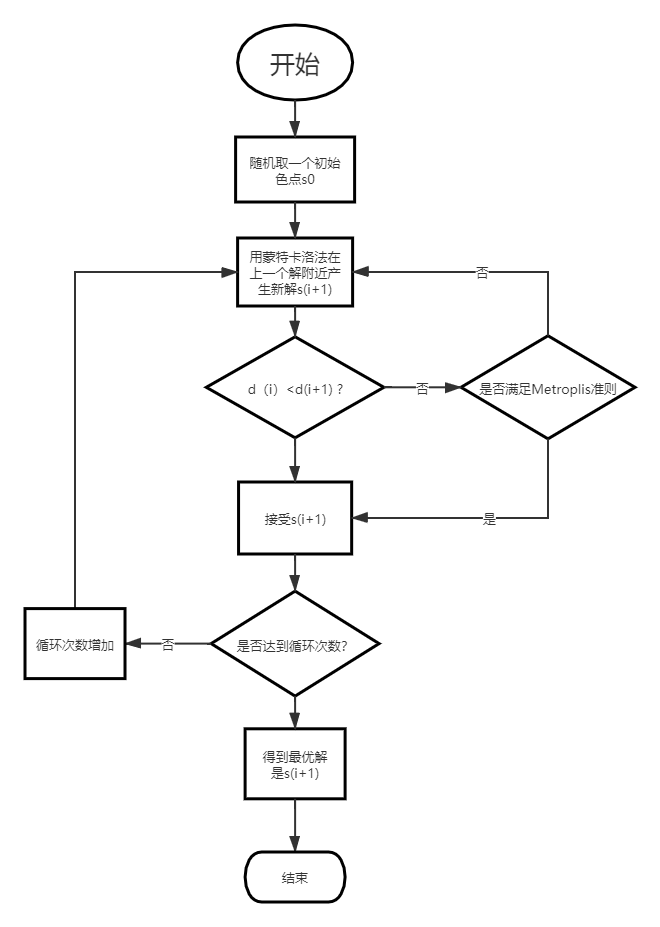


图3.2.1 模拟退火算法流程图

首先，随机生成一个初始顺序，计算每一步的路径步长，将距离储存在一个列表中，并求出距离之和。其次，利用模拟退火过程以一定规则更新顺序并重复计算（比如随机调换路径两点顺序），求出距离之和，如果大于，那么接受；如果小于，那么有概率接受，有1-的概率不接受。

概率的计算公式如下：

在程序中代表退火温度。观察可知，在程序运行的前期，较小更有可能接受新解，在解空间的搜索范围更大。在程序运行的后期，较大使得接受新解的概率更小，解空间的搜索范围局限在最优解附近,这样就避免了算法陷入局部最优解的现象。

当循环次数达到给定次数时，程序退出循环，此时得到模拟退火算法得到的最优解。算法迭代温度图以及路径规划效果如图所示：

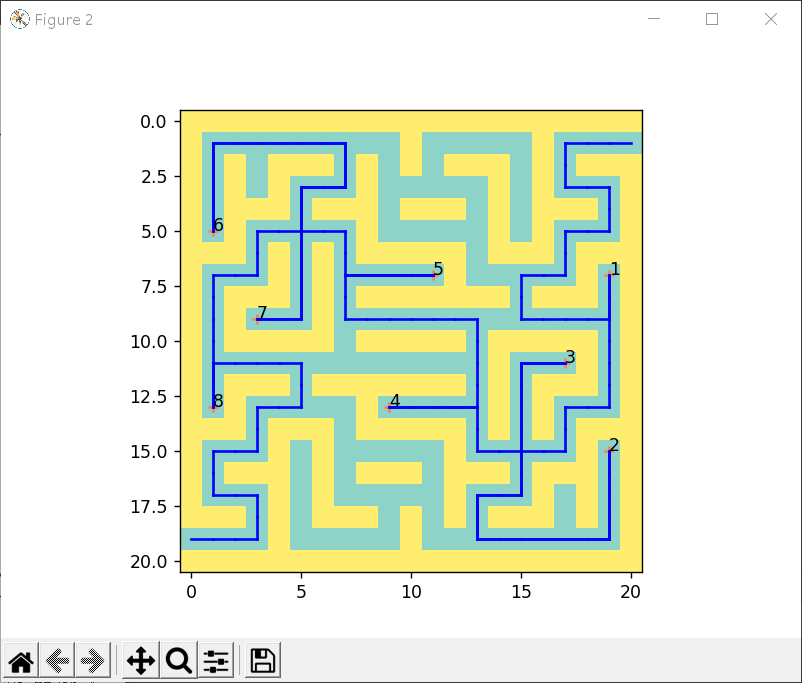
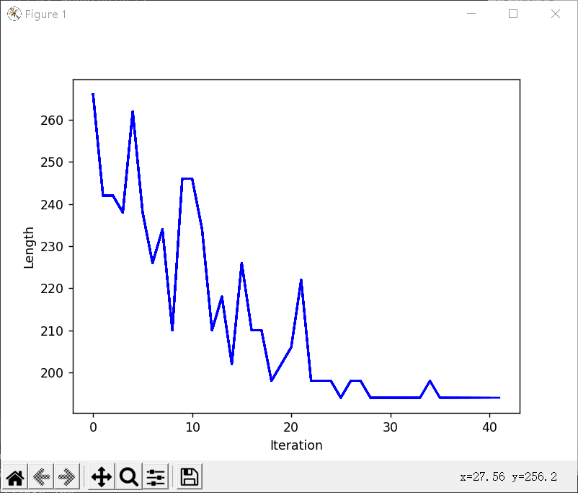


图3.2.2 模拟退火算法迭代图和效果图

可见路径规划算法可以极大的减小移动路程，提升搜索效率。输出的每一步路经通过串口发送至机器人，写入机器人主控flash中。