**轮式机器人大作业**

**Sokoban**



组员：项吟沨、宋浩波、黄炯睿、郭家桢

# 1.问题背景：

## 1.1.推箱子问题介绍：

推箱子属于滑块谜题（Sliding Piece Puzzle）的一种，推箱子问题已被证明是NP-hard,进一步的研究表明它比NP问题困难得多，它属于PSPACE-Complete。

推箱子难度很大，它的搜索树深度能达到相当大; 某些级别可以无限延长，每次迭代都需要指数级增加的移动和推动次数。即随着地图维数增加，复杂度将指数级上升。一些推箱子谜题可以通过使用单一代理搜索算法自动解决，如IDA\*。然而，即使对于最好的自动求解器，更复杂的推箱子水平也是遥不可及的。

## 1.2.项目介绍

考虑到全局定位的限制，本项目主要考虑推箱子问题中，较为基础的情况，即单个人在12\*8的范围内完成将3个箱子推动到目标地点。

整个过程大致可以划分为两个部分：**离线路径规划**与**实时轨迹规划。**其中路径规划在Java中完成，轨迹规划则利用全局定位信息完成。

## 1.3 项目前景

本项目的意义主要包括以下两个方向：

（1）本项目给出推箱子游戏的自动求解方法。在一般规模下（如15\*15、5个箱子），能够较快的给出解决路径方案，而对于更小规模的问题，则能将求解时间降至毫秒级。

在封装后若加入可视化模块，则能很好的提供推箱子游戏的参考答案，有助于幼儿益智游戏的开发（检查某种布局的推箱子是否无解）。

（2）本项目在现实中也有较大应用，其中最契合的就是仓储机器人。当前仓储机器人领域用的最多的是自动导引车系统 (AGV) ，即沿着规定轨道运作，这样安排会有较高的稳定性，但是效率方面可能存在欠缺。一个可行的方案就是用自主移动机器人（AMR）来替代AGV。

对于单个自主移动机器人来说，将货物装卸加载的过程事实上就和本项目中差分机器人推动箱子至目标点的行为相近。只是多了与装卸区的配合。

当然对于实际仓储来说，除了自动化，还需要考虑多个机器人之间的协调，以及路径的交叠等问题。这也是当前领域的研究热点。

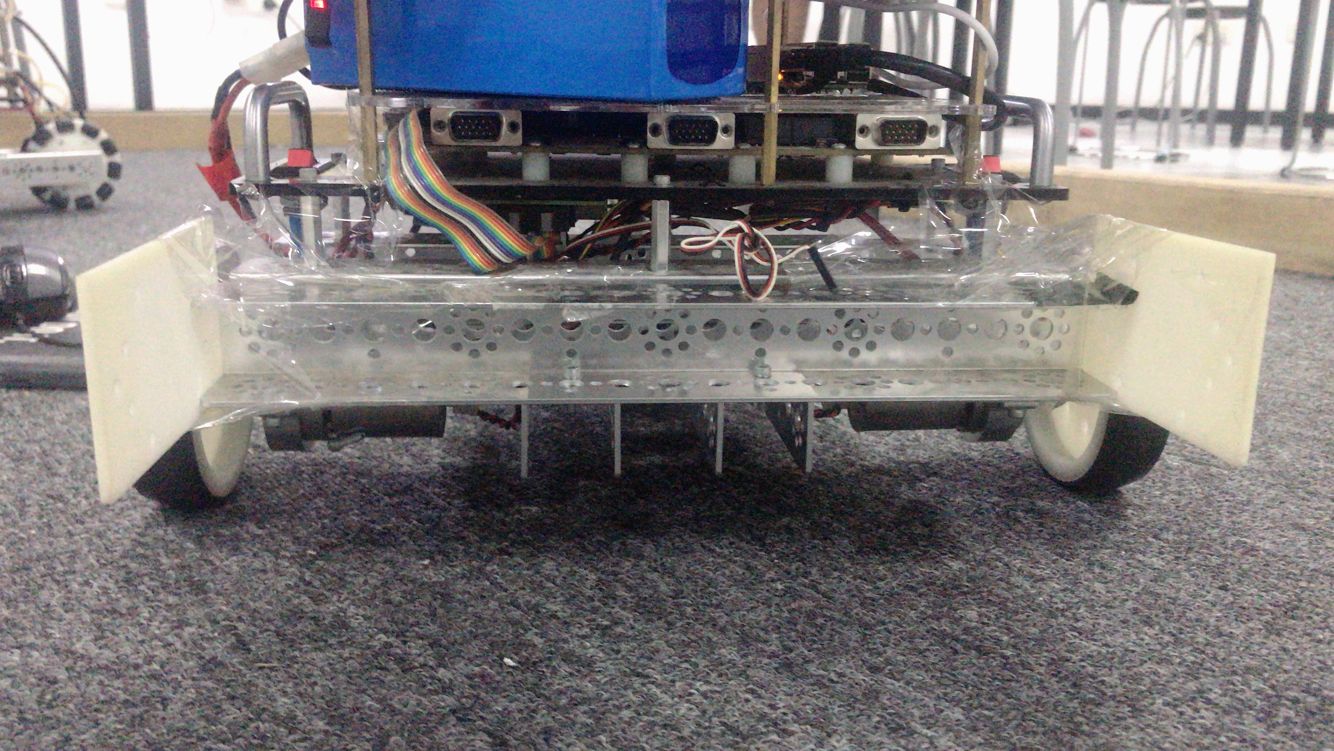
# 2．项目实现：

## 2.1硬件部分：

### 2.2.1 小车部分

为了能够推动箱子，小车需要做适当的改装，即在车身前添加挡板，所需要考虑的部分即是，挡板的长度以及相对距离应当适中：





### 2.2.1 其余部分

除此以外，在前期应当通过地标的形式标记目标格，以及障碍物等信息，在定位完成后，则通过如下的轮滑桩标记边界：

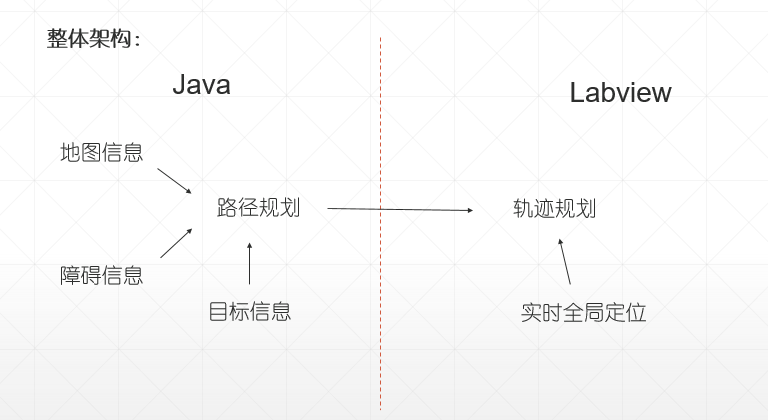


箱子部分则可以利用快递盒子并装载以重物完成：



## 2.2 软件部分：

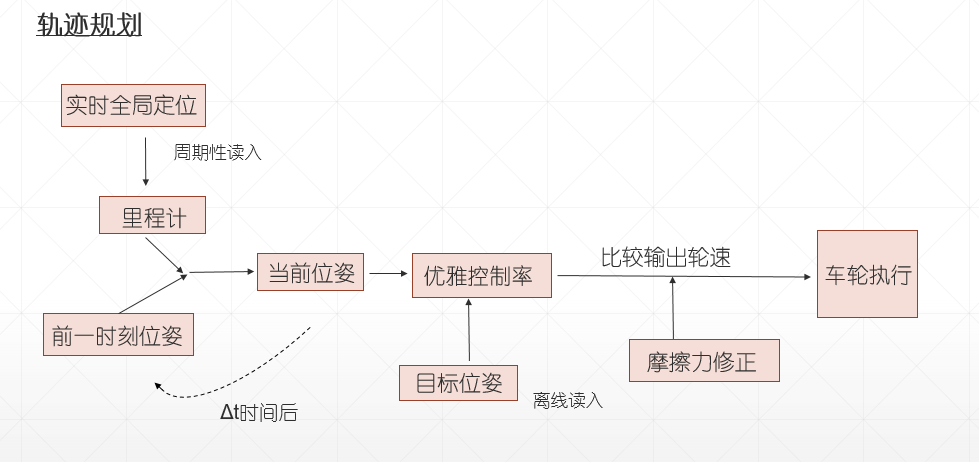
整个程序的架构如下：



### 2.2.1 路径规划

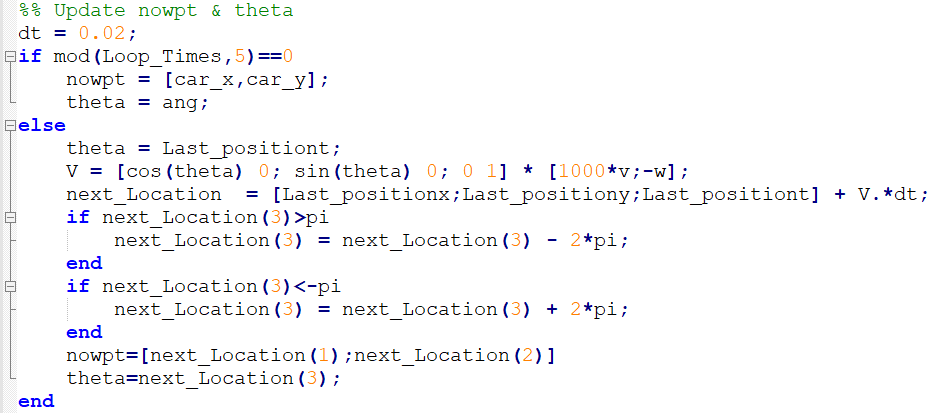
### 2.2.2 轨迹规划

轨迹规划部分流程图如下：



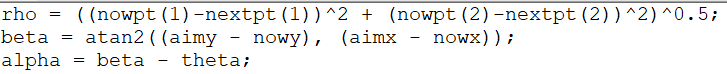
下面分功能模块结合代码阐述具体实现：

1. **全局定位-里程计**



考虑到全局信息存在较大的波动，为了得到更加平滑的位置关系，此处选择以更低的频率（T=0.1s）刷新以此全局信息，而在两次全局信息之间通过里程计的方式计算当前位置。

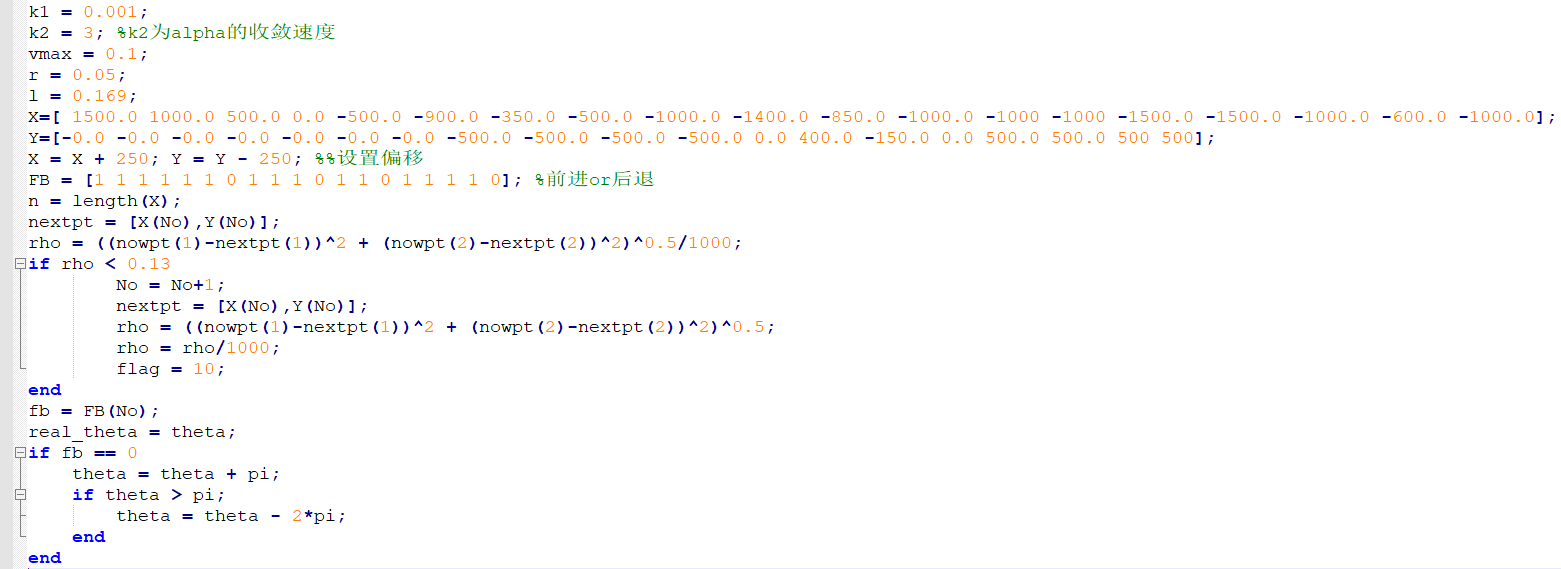
1. **误差向量**

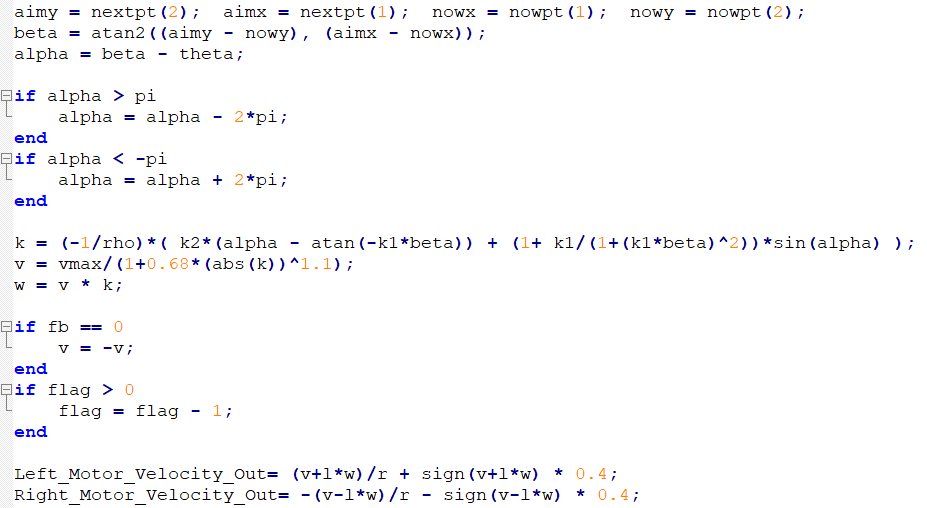


此处通过误差向量 即可描述两个位姿之间的关系。之后只要通过控制函数（控制率），即即可完成位姿的修正。

1. **控制率-摩擦修正**

轨迹规划中最关键的部分即为控制率，部分，此处选用的即是课堂所介绍的一种非线性控制率，具体原理略去不表，代码如下：





需要额外注意，考虑到推箱子过程中不存在带箱子转弯的功能，则在模拟的时候，当一次直线推动举动完成后，需要后退以完成转向的操作，这是代码中FB的具体含义（即在当前点需要前进or后退）。

# 3、效果展示：

## 3.1 路径规划部分：

两版的路径规划效率存在很大的进步。考虑到，不同的地图之间难以用固定量来量化，（例如相同尺寸、目标数复杂度可能相差巨大），以下给出若干种地图，两版算法的耗时：

## 3.2 路径规划部分：

请参照视频信息；需要额外注意的是，两个全局定位源在中间部分存在交叠，故会出现错位的现象。