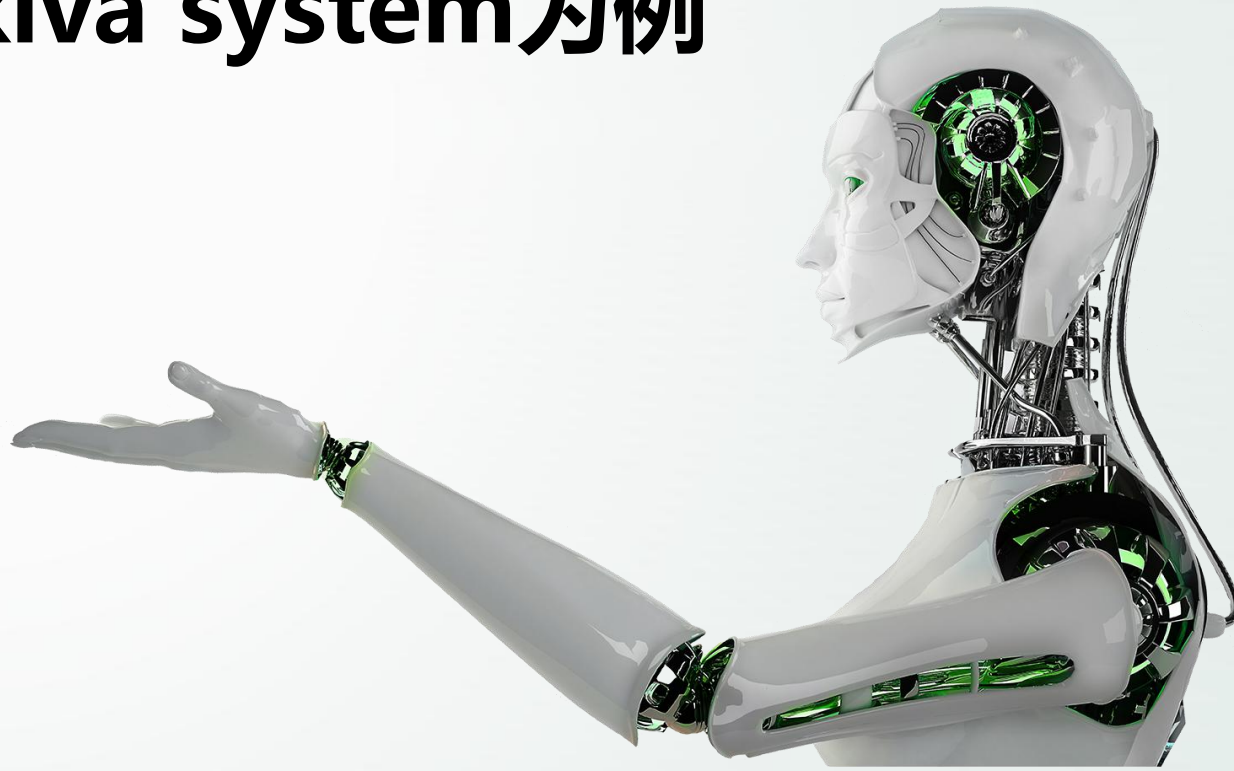
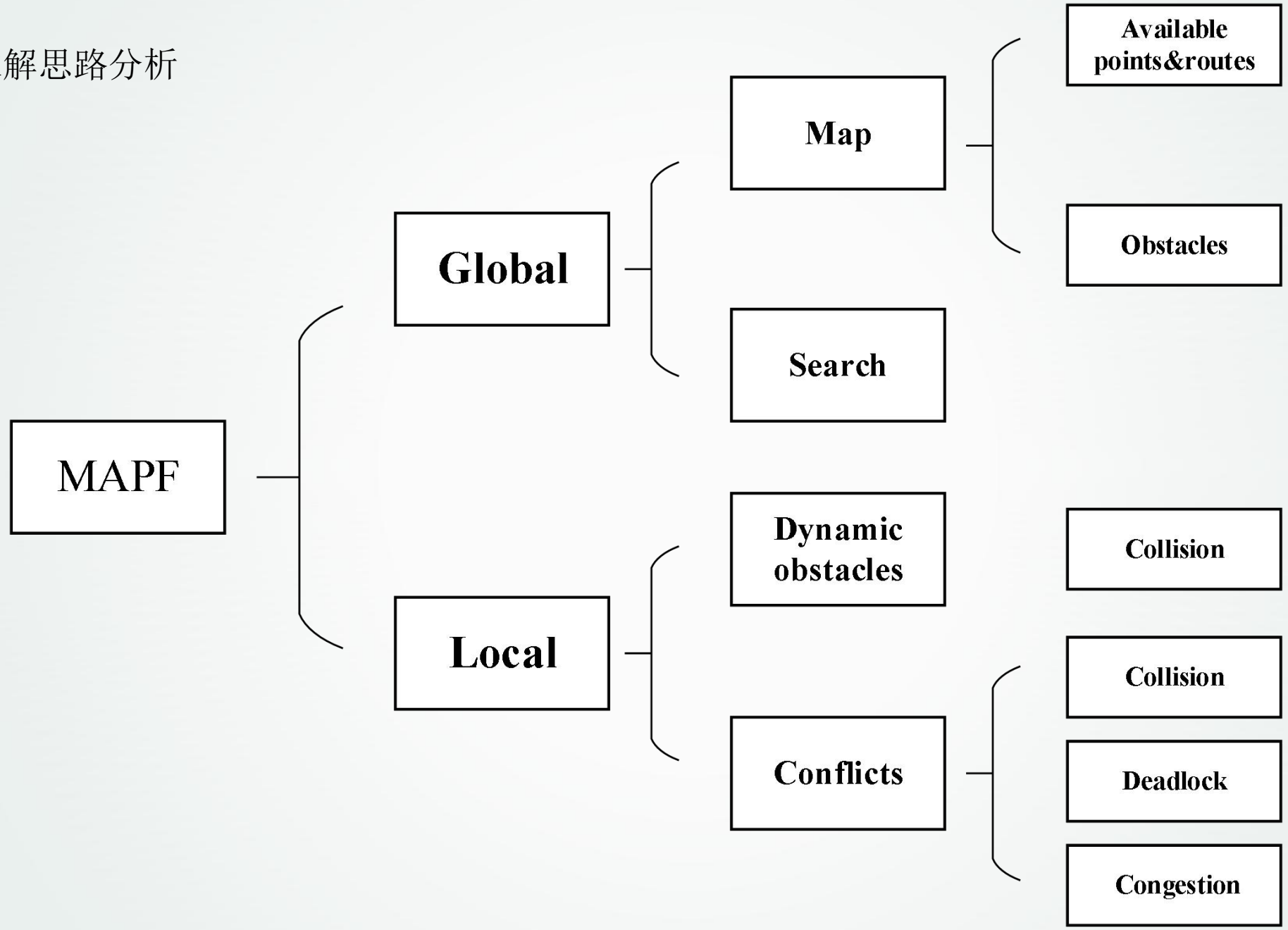
第八章

仓储系统中的MAPF-以kiva system为例

1. Kiva系统简介
2. Kiva picking路径规划思路
3. Kiva picking实现



MAPF求解思路分析



什么是Kiva?

为什么是Kiva?

1. Kiva系统简介

Kiva系统是一种针对拣选过程的自动化物料处理系统。

Kiva系统应用了廉价的机器人，能够提升和搬运三英尺见方的货架单元，称为库存吊舱。

这些被称为驱动装置的机器人将库存吊舱从储存地点运送到工作站，工人可以在那里从货架上拣下物品并将其放入装运箱中。

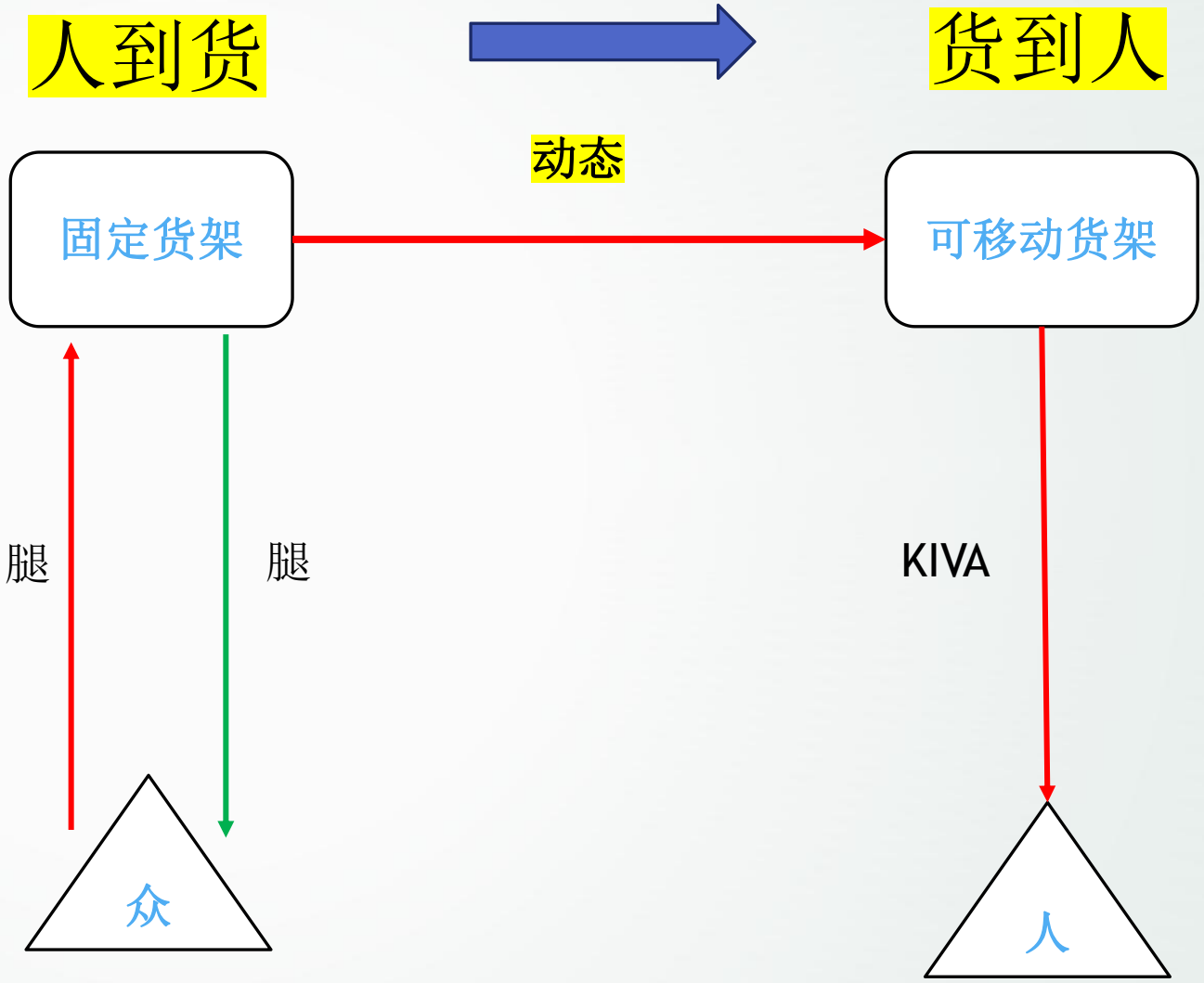
一整天，工作者都呆在自己的位置上，而机器人承担连续不断的拣选任务。

现在有了机械臂以及先进的识别算法，最后一环的人工也可被解放

1. Kiva系统简介

1. **Picker to parts 到 parts to picker**
2. 效率高、成本低
3. 灵活性、可控性
4. 柔性（模块化，动态）

Kiva系统由几十上百台机器人组成。这就映射到了MAPF问题



2. Kiva picking路径规划思路

1. 智慧物流的一般工作流程

订单到达—

任务分配、资源分配—

kiva拣货—

kiva送至传送带—

传输至拣选台—

人工或机械臂拣选—

打包出库—

运输—

快递配送--

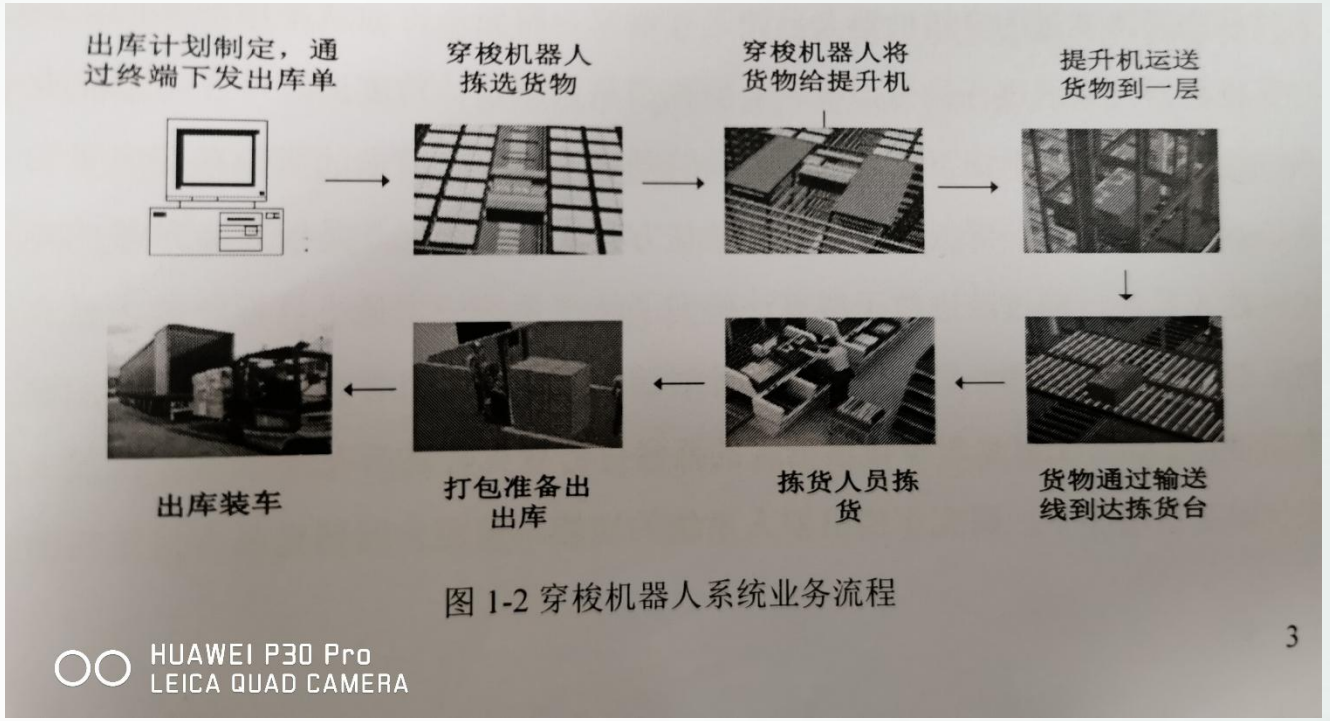
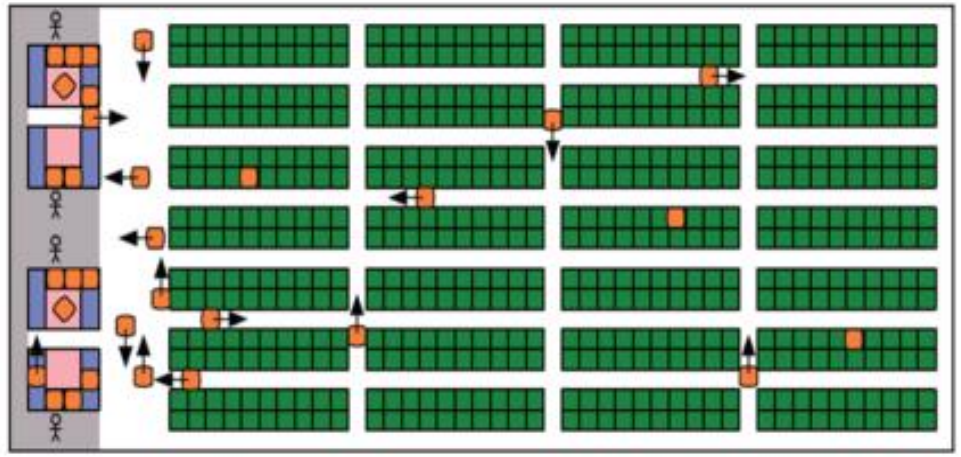


图 1-2 穿梭机器人系统业务流程



2. Kiva picking路径规划思路

1. 智慧物流的一般工作流程

订单到达—

任务分配、资源分配—

kiva拣货—

kiva送至传送带—

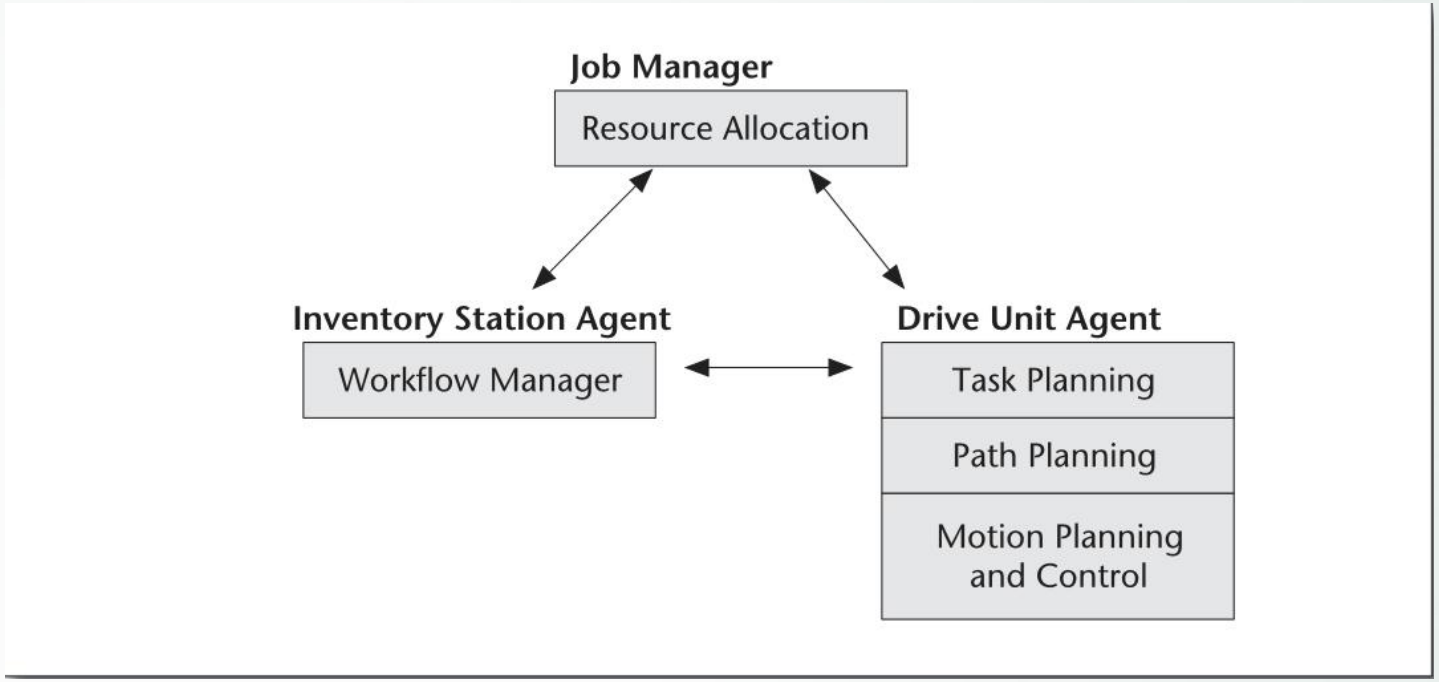
传输至拣选台—

人工或机械臂拣选—

打包出库—

运输—

快递配送--



这个过程与云计算的任务调度过程高度相似

Job—workflow—task scheduling- resource allocation

这也是第十章云大脑控制智能仓储的原理之一

2. Kiva picking路径规划思路

1. 智慧物流的一般工作流程

订单到达—

任务分配、资源分配—

kiva拣货—

kiva送至传送带—

传输至拣选台—

人工或机械臂拣选—

打包出库—

运输—

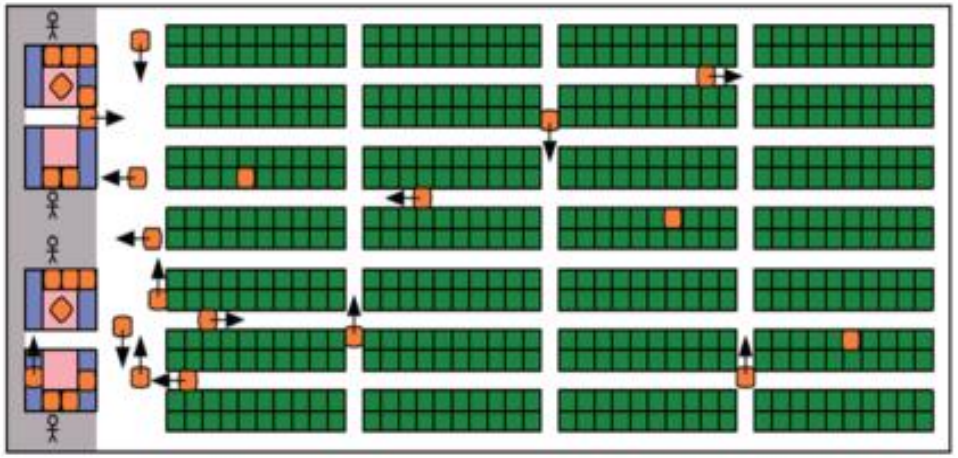
快递配送--

2. Kiva picking路径规划思路

系统建模

需求分析（约束）

算法设计及测试



2. Kiva picking路径规划思路

Kiva 系统的相关介绍参考如下文章

AI Magazine Volume 29 Number 1 (2008) (© AAAI)

Articles

Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses

Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz

3. Kiva picking实现

Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

A Coordinated Path Planning Algorithm for Multi-Robot in Intelligent Warehouse *

Xin Chen, Yanjie Li, Lintao Liu
School of Mechanical Engineering and Automation
Harbin Institute of Technology, Shenzhen
Shenzhen, Guangdong Province, China
autolyj@hit.edu.cn

思路清晰
结构简单
可复现性

3. Kiva picking实现

In this paper, we study the coordinated path planning problem of multi-robot in intelligent warehouse. We **first** describe the problem of the multi-robot path planning and analyze the basic structure of the intelligent warehouse system. **Then** we discuss the adjustment of multi-robot path planning algorithm from two aspects. We reserve the path information of the continuous time period in the reservation table of the **Windowed Hierarchical Cooperative A* (WHCA*) algorithm**, and analyze whether to reuse the path information each time the algorithm is invoked. Then on the basis of this, we introduce the **turning factor** as a measure when expanding a path, and hope to find the path with fewer turns to reduce the travel time of the robot. **Finally**, we do some experiments and make the analysis. The results show that the methods we used can improve the path planning efficiency of the robot and increase the order throughput of the intelligent warehouse system.

1. 描述问题
2. 分析仓储系统特征
3. 讨论算法的调整/改进
4. 具体的调整/改进策略
5. 实验

3. Kiva picking实现

问题描述

The problem of the multi-robot path planning can be expressed as $\{Map, ROB, S, G\}$. The Map is the environment that robots are located in, and there are one or more obstacles in it. We assume every location of the Map is marked by a QR code. The set ROB contains N robots, which is expressed as $ROB=\{r_1, r_2, ..., r_N\}$. S is a set of $\{s_1, s_2, ..., s_N\}$, which denotes the starting points of each robot. G is a set of $\{g_1, g_2, ..., g_N\}$, which denotes the target points of each robot. The input of every robot is $< Map, s_i, t_i >$, and the output of every robot is a path that leads the robot to complete their tasks. The robot needs to plan a collision-free path from the starting point to the target point. If a feasible path is found for all robots, then such a solution is called a feasible solution. While finding the feasible solution, we hope that some of the evaluation indexes will be optimal. If necessary, we will add constraint to the robot.

系统特征

In the intelligent warehouse system, it generally consists of storage area, picking stations, replenishment stations, pods, and robots. The intelligent warehouse system is shown in the Figure 2. The items are stored in the movable pod, and the pods are randomly placed in the storage area. There are to the station according to the order tasks. In general, we want pods that are often executed for picking and replenishment operations to be placed closer to the station.

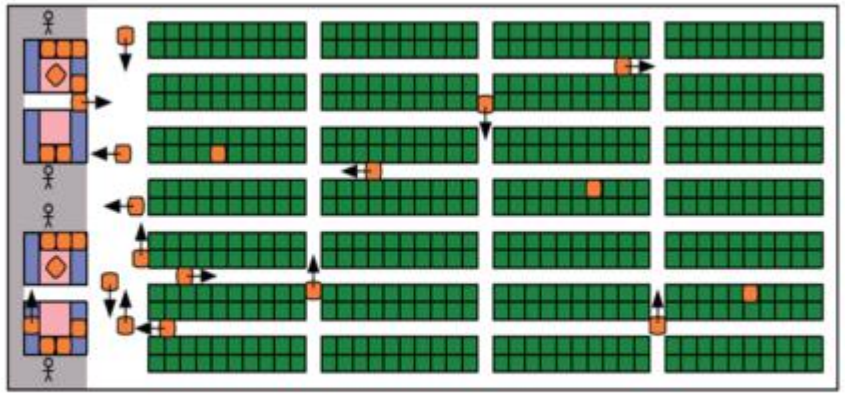


Fig. 2 Intelligent warehouse system.

系统特征

There are generally multiple decision problems in intelligent warehouse system. For example, task assignment, path planning, resource allocation, etc. In the task assignment problem, the central controller transmits the received order task information to the task assignment controller. The task assignment controller assigns a new task to the robot for execution according to the location of the robot in the warehouse and the completion of the task through a certain assignment method, for example an auction-based method [16]. The path planning problem is roughly the same as that discussed earlier. In the resource allocation problem, it involves the sub-problems of pod selection, pod storage, robot allocation and so on [17]. If a robot has just performed the picking task and the pod needs to be replenished immediately, then the robot can simply carry the pod from the picking station to the replenishing station. Therefore, the order of execution of the replenishment and picking tasks, and the selection of the required pods will affect the completion of the final task to some extent. The control block diagram of the intelligent warehouse system is shown in the Figure 3.

During the actual driving process, the robot will be given some constraints such as maximum driving speed, whether it can drive in both directions, and so on. The robot cannot occupy the same position at the same time. For an aisle with a fixed direction of travel, the robot can only travel in the specified direction. In the actual warehouse environment, we also consider the acceleration rate and deceleration rate of the robot. The output of the system is the travel path of the robot and the completion of the order task. The objective function for evaluating path planning usually include *makespan* [11], *planning time*, *travel distance*, etc.

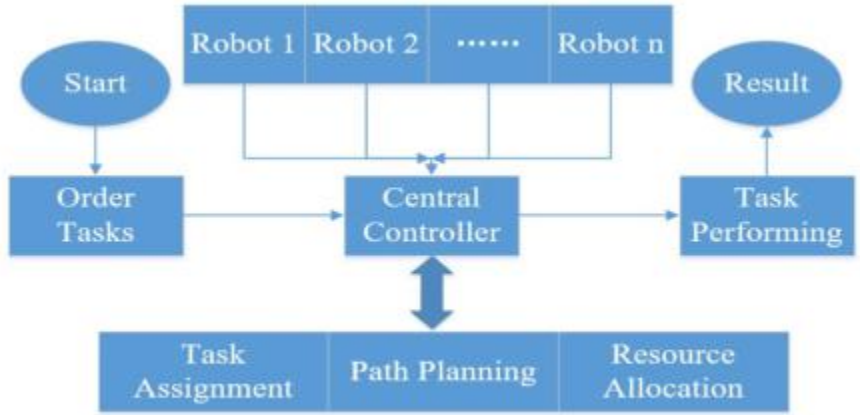
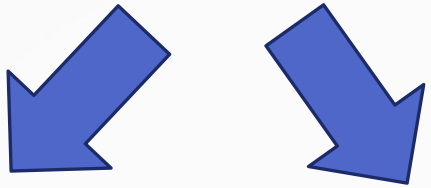


Fig. 3 Control block diagram of the intelligent warehouse system.

3. Kiva picking实现

算法设计及调整/改进思路



A 考虑多机并行的冲突问题

A*算法适用于单机路径规划

导致多机并行冲突问题显著

改进

具体方法是WHCA*

WHCA*的原理

B 考虑转向次数对cost的影响

路径长度相同或相近时，转弯过多会增加时间成本

改进

减少转向次数

具体方法是修改cost函数，即fitness

原理是每次转向增加一个惩罚

3. Kiva picking实现

A 考虑多机并行的冲突问题

A. Consider Actual Movement

In the previous problems of multi-robot path planning research, most methods are on the grid map environment, and the robot moves in a single step, and the cost per step is the same. The A* algorithm is suitable for the path planning problem of a single robot. When multiple robots perform path planning, collisions and conflicts will occur between multiple paths. Therefore, based on the basic A* algorithm, [6] proposed the WHCA* method, which has achieved very good results in the path planning problem in the game field. We can also apply this algorithm to multi-robot path planning problems in intelligent warehouse.

The WHCA* algorithm takes into account the paths that other robots have planned during the search. The main idea of this method is to expand the two-dimensional space map into a three-dimensional space-time map by increasing a time variable. The algorithm records the path information of the robot through a global reservation table, so as to avoid occupying the same location at the same time in the subsequent robot planning path process. In order to improve the efficiency of path planning and reduce long-term reservations for a certain location point, the depth of the space-time search is limited by adding a time window.

1. 指出以往研究和应用的特点和问题：

a)栅格地图/网格地图

b) 时间离散化，步进式，计算cost

c) 采用A*等

d) single--- multi产生conflict

2. 提出改进策略：

a) 采用WHCA*（源自游戏NPC的路径规划）

b) 思想是考虑其他agent的路径信息

3. 根据思路设计具体的改进策略：

a) XY二维地图---加入时间变量，变为三维

b) 设置global reservation table（用于实时记录所有agent的路径信息）

c) 加速策略：设置时间窗，作用有两个，一是强约束，进一步规避conflict；二是避免table信息溢出，也就是搜索空间过大

3. Kiva picking实现

A 考虑多机并行的冲突问题

In an actual warehouse environment, the robot tends to run continuously during operation and also satisfies some kinematic conditions. For example, if a robot travels a long distance, it will experience acceleration, uniform speed, and deceleration. If a robot goes to the picking station to perform the picking task immediately after completing the replenishment task at the replenishment station, it may only have the acceleration and deceleration process. In these cases, the path planning process of the robot is often measured in continuous time. A safe interval path planning (SIPP) algorithm is proposed in [18], which is a single robot continuous time path planning algorithm. SIPP is a path planning algorithm based on A* algorithm. It combines multiple collision-free continuous time steps into a safe interval and searches for the state represented by (configuration, safe interval) pair in the search space. SIPP theoretically provides guarantee of optimality and completeness. Therefore, under the condition of continuous time, we can make a certain adjustment to the reservation table, and record the continuous time interval reserved for each position. Each interval has a start time and an end time. Only robots with the correct number in this interval can go through this position. The adjusted reservation table is shown in the Figure 4.

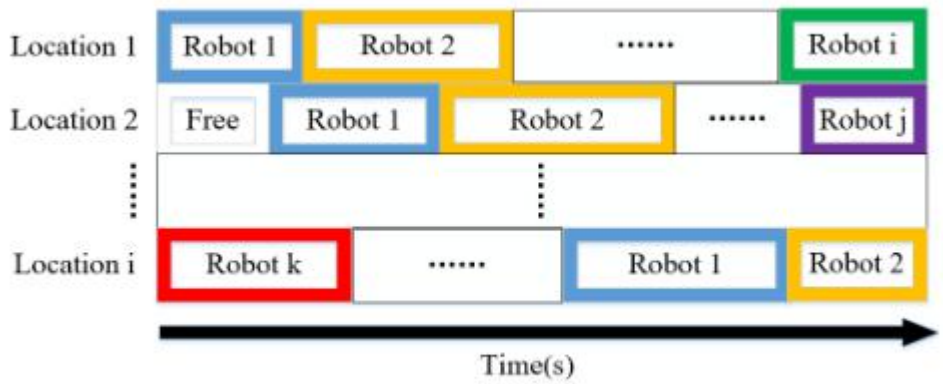


Fig. 4 Adjusted reservation table.

1. 解释机器人实际运行状态：

a) 包括加速，匀速，减速，停止状态

b）实际是连续状态，为方便优化做离散处理

2. SIPP：安全区间路径规划

a) 实际上就是前面讲到的预约机制

b） 预约表的本质就是时间窗

3. Kiva picking实现

A 考虑多机并行的冲突问题

After modifying the reservation table in the WHCA* algorithm, we discuss the use of existing path information when invoking the algorithm. The algorithm each time it is invoked will plan paths from start point to target point for each robot that has been assigned task. In the process of other robots performing tasks, if a robot in an idle state is assigned a new task, the algorithm is invoked again to plan the path for the robots. Whether the algorithm reuses the path information in the reservation table at each call is related to the execution efficiency of the algorithm. Intuitively, reusing existing path information will help us reduce the time it takes to calculate a new path. However, the path of the previous robot cannot be updated in real time. It is possible that the robot will miss a chance to drive a more perfect path in the remaining travel process before reaching the target point.

In general, the method of reusing path information is more suitable for the case where the obstacle in the environment is fixed, and the method of not reusing the path information is more suitable for the case where the obstacle in the environment is dynamically changed. Using different methods can have different effects on the overall efficiency of the intelligent warehouse system.

进一步解释预约表的使用

1. 每次表更新后，根据表的变动，对每条path进行重规划
2. 如果有新的agent从idle变为busy，即有新的机器人加入执行任务，立即对其进行路径规划
3. 关于信息保留：是否保留已有路径信息？
   a) 全保留：计算快，路径可能非最优
   b) 不保留：计算慢，路径优
   c) 选择保留：介于两者之间

3. Kiva picking实现

B 考虑转向次数对cost的影响

B. Consider Turning Factor

As shown in the Figure 5, when the starting point and the target point of a robot are determined, it can have multiple paths with the same distance to reach the target point on the grid warehouse map. When the path of the robot is searched by the A* algorithm, it is expanded randomly. When the robot travels along path 1, there are only 2 times of turning, there are 4 times of turning when traveling along path 2, and there are 5 times of turning when traveling along path 3. Different driving paths will take different driving time. If you choose a path with more turns, it will undoubtedly increase the driving time of the robot. For this reason, we need to improve the evaluation function in the A* algorithm and take the turning factor into account to find a path with fewer turns.

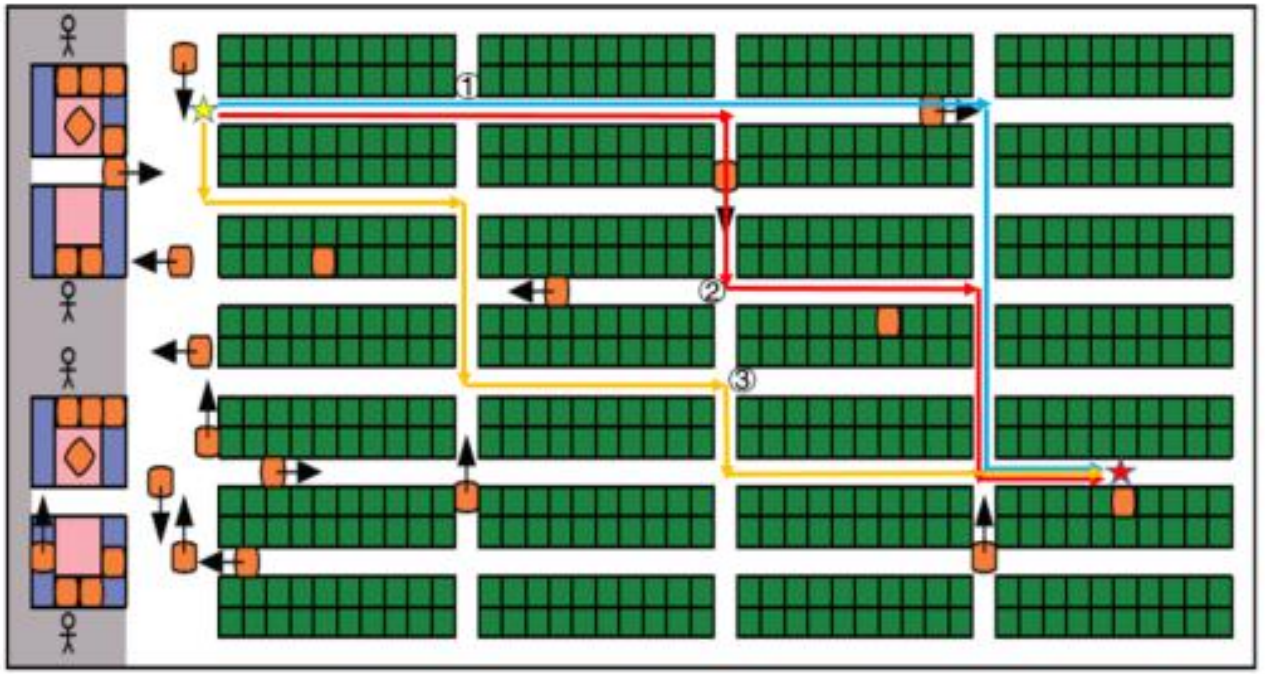


Fig. 5 Paths of the same length with different turns.

首先讨论了转向次数对cost的影响

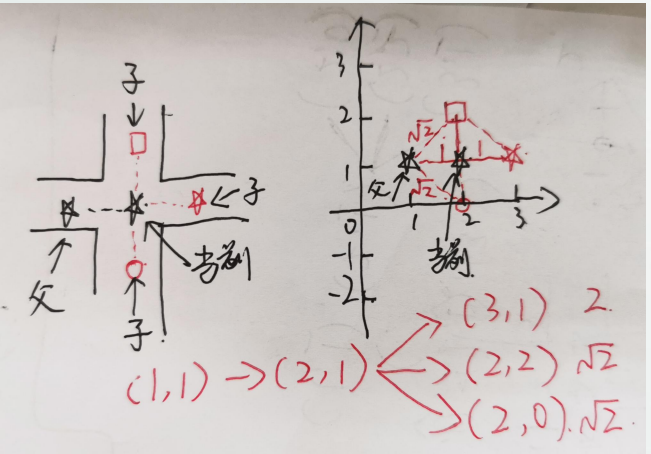
3. Kiva picking实现

B 考虑转向次数对cost的影响

When judging whether a turn is generated, we judge by the positional relationship between the current node and its parent node, and the positional relationship between the current node and its neighboring nodes. Two situations in which a path generates a turn are: 1) if the current node has the same x-direction coordinate as its parent but the x-direction coordinates of its neighbor are not the same, or 2) the current node has the same y-direction coordinates as its parent, but the y-direction coordinates of its neighbor are not the same. A turn is generated by satisfying one of the conditions.

如何判定是否发生了转向

通过计算下一节点与父节点的距离来判定



在f函数中加入转向因子

本质就是加惩罚

We can introduce a turning factor τ as a measure. The coordinate of the current node C is expressed as $(current.x, current.y)$. The coordinate of the parent node P is expressed as $(parent.x, parent.y)$. The coordinate of the neighbor node N is expressed as $(neighbor.x, neighbor.y)$. In the first situation, the schematic diagrams of turning and no turning is shown in the Figure 6, and the turning factor in this case is expressed by the equation (1). In the second situation, the schematic diagrams of turning and no turning is shown in the Figure 7, and the turning factor in this case is expressed by the equation (2). We add a turning factor to the original evaluation function of A* algorithm and expand the path by selecting the node with the smallest evaluation function. The evaluation function that considers the turning factor is shown in the equation (3), where $g(n)$ is the cost function, $h(n)$ is the heuristic function.

$$\tau = \begin{cases} 1 & if \ \tan \dfrac{|neighbor.x - current.x|}{|current.y - parent.y|} \neq 0 \\ 0 & otherwise \end{cases}. \quad (1)$$

$$\tau = \begin{cases} 1 & if \ \tan \dfrac{|neighbor.y - current.y|}{|current.x - parent.x|} \neq 0 \\ 0 & otherwise \end{cases}. \quad (2)$$

$$f(n) = g(n) + h(n) + \tau . \quad (3)$$

3. Kiva picking实现

B 考虑转向次数对cost的影响

We can introduce a turning factor τ as a measure. The coordinate of the current node C is expressed as $(current.x, current.y)$. The coordinate of the parent node P is expressed as $(parent.x, parent.y)$. The coordinate of the neighbor node N is expressed as $(neighbor.x, neighbor.y)$. In the first situation, the schematic diagrams of turning and no turning is shown in the Figure 6, and the turning factor in this case is expressed by the equation (1). In the second situation, the schematic diagrams of turning and no turning is shown in the Figure 7, and the turning factor in this case is expressed by the equation (2). We add a turning factor to the original evaluation function of A* algorithm and expand the path by selecting the node with the smallest evaluation function. The evaluation function that considers the turning factor is shown in the equation (3), where $g(n)$ is the cost function, $h(n)$ is the heuristic function.

$$\tau = \begin{cases} 1 & if \ \tan\dfrac{|neighbor.x - current.x|}{|current.y - parent.y|} \neq 0 \\ 0 & otherwise \end{cases} . \quad (1)$$

$$\tau = \begin{cases} 1 & if \ \tan\dfrac{|neighbor.y - current.y|}{|current.x - parent.x|} \neq 0 \\ 0 & otherwise \end{cases} . \quad (2)$$

$$f(n) = g(n) + h(n) + \tau . \quad (3)$$

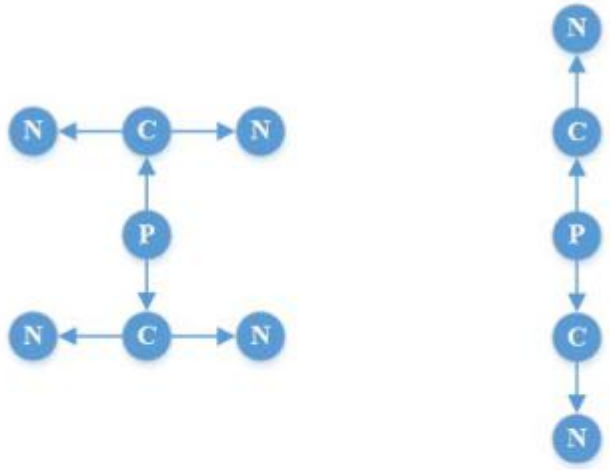


Fig. 6 The schematic diagrams of turning and no turning in situation one.

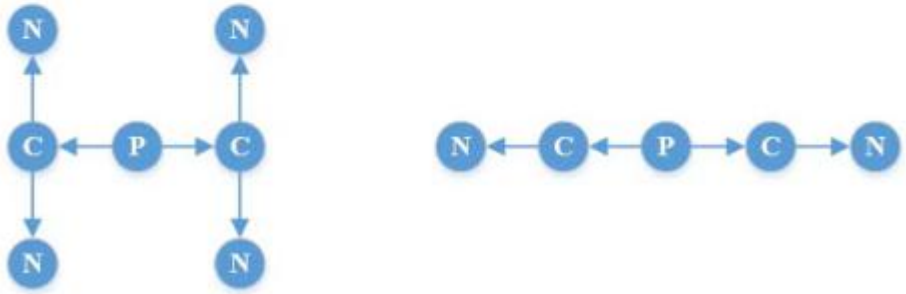


Fig. 7 The schematic diagrams of turning and no turning in situation two.

1.判定是否转向的公式，稍显笨重，原理还是通过距离判定

2. 转向的类型

F=g+h
F=g+h+t

3. Kiva picking实现

算法验证及实验分析

A. *The Usage of Path Information*

We do the experiment for whether reusing path information. We set up different numbers of robots in the warehouse, and observe the impact of the two methods on the overall path planning effect. The time window of the algorithm is set to 30 seconds. In each simulation, the maximum travel speed of the robot is 1.5m/s, the acceleration rate and deceleration rate are both 1m/s^2, and the robot performs 24 hours of work. The relevant experimental data is shown in the table I and table II. The comparison of the two methods in average path planning time is shown in the Figure 8. From the experimental data, we can see that as the number of robots increases, the average time required to plan the path will increase, and the overall driving distance will also increase. The method of reusing path information is less in overall planning time, planning count, average planning time than the method of not reusing path information. However, the two methods show different effects in the overall driving distance due to the number of robots. When the number of robots is between 5 and 19, the method of reusing the table path information allows the robot to travel a smaller distance than the method of not reusing the table path information in the overall travel distance. When the number of robots is between 20 and 40, the method of reusing the table path information allows the robot to travel a longer distance than the method of not reusing the table path information in the overall travel distance. When the number of robots is between 41 and 50, it may be due to the influence of other decision controllers, and the two methods will alternately lead.

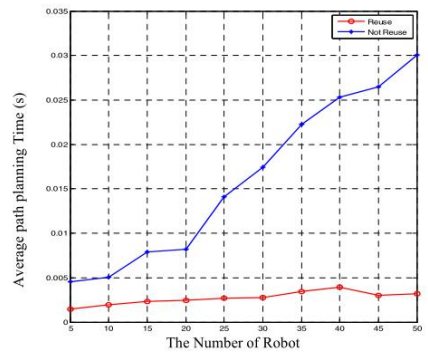


Fig. 8 Comparison of average path planning time between the two methods.

TABLE I
TEST OF REUSING PATH INFORMATION

Robot Number	Index			
	Overall Planning Time(s)	Planning Count	Average Planning Time(s)	Overall Distance Traveled(m)
5	23.87785430	16221	0.0014720334	310470.835
10	55.21246469	28488	0.0019380955	505970.434
15	84.33417589	36222	0.0023282584	628361.679
20	94.05264989	38164	0.0024644338	691357.138
25	106.24174929	39489	0.0026904138	731619.736
30	116.19998649	42275	0.0027486691	781488.137
35	162.54191819	47067	0.0034534157	865781.978
40	193.06734340	49065	0.0039349300	900024.569
45	150.22180419	49623	0.0030272616	905905.698
50	163.53304039	51113	0.0031994412	937513.703

TABLE II
TEST OF NOT REUSING PATH INFORMATION

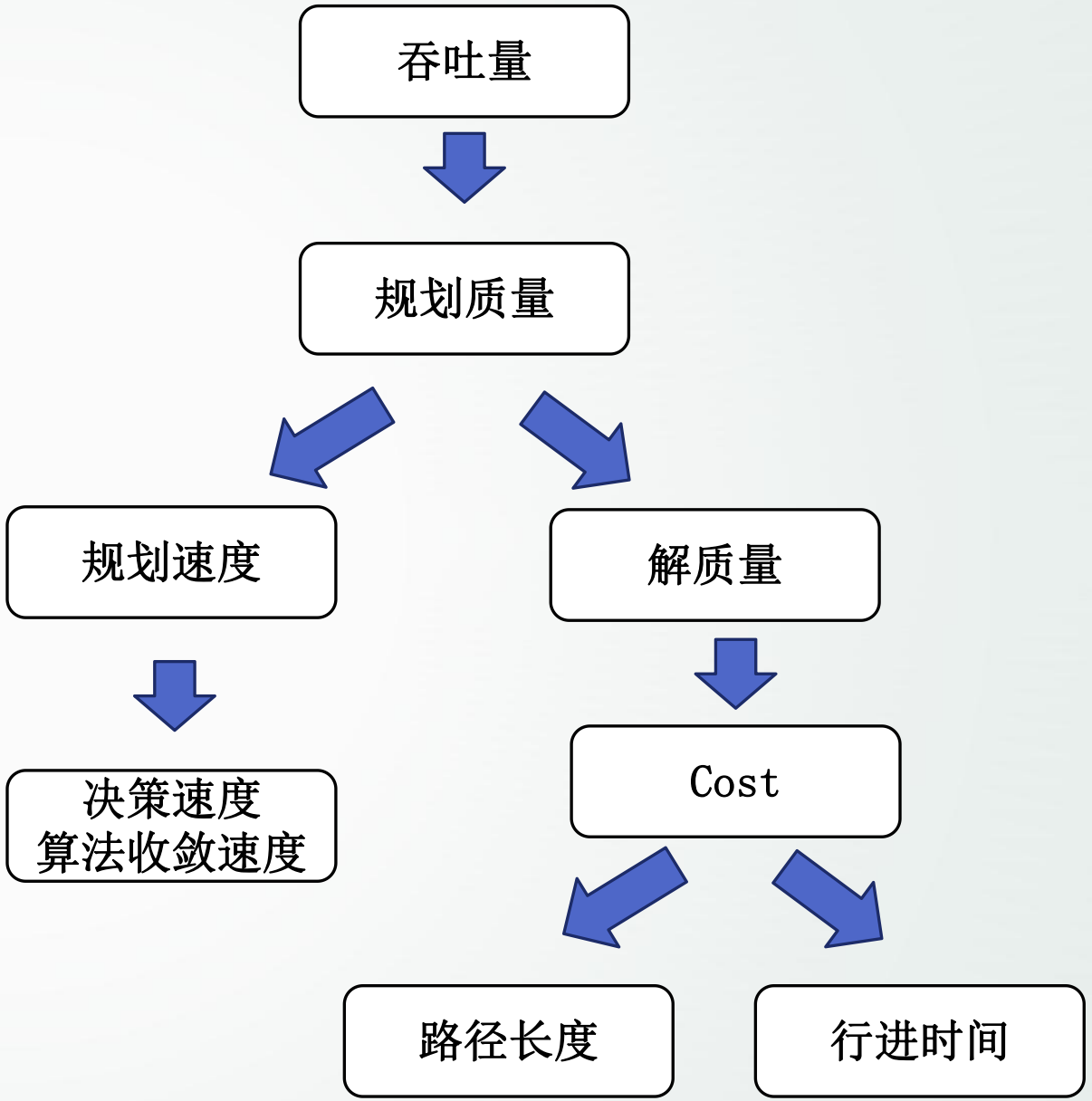
Robot Number	Index			
	Overall Planning Time(s)	Planning Count	Average Planning Time(s)	Overall Distance Traveled(m)
5	35.91374130	15377	0.0045749033	312893.777
10	138.71865579	27541	0.0050368053	514582.302
15	279.40013110	35513	0.0078675452	635010.643
20	315.57937120	38429	0.0082120110	688489.948
25	556.81646550	39584	0.0140667054	724631.682
30	755.32981119	43244	0.0174666962	776654.984
35	1100.78493229	49393	0.0222862538	859806.599
40	1311.25279369	51818	0.0253049673	896680.720
45	1364.82839769	51585	0.0264578539	909073.240
50	1618.20704609	53818	0.0300681379	935459.071

1. 比较是否resume path information的性能
   1) 参数设置
      a) time window 30s
      b) Vmax=1.5m/s
      c) a= 1.2m/s^2
      d) T= 24h
   2) 实验结果
      a) 平均规划时间对比
      b) 解质量对比：5-19，resume好；20-40个，不resume好；41-50个，交替领先

3. Kiva picking实现

算法验证及实验分析

Generally, we want to obtain a larger order throughput in the intelligent warehouse system, so a faster speed of algorithm is important. But according to the computing power of the modern computer, the calculation increment of the microsecond level is not a big problem. The shorter driving distance in the actual warehouse is the goal we are pursuing, therefore we will adopt different methods according to the number of robots in the warehouse. From the comparison between Table I and Table II we know, in most cases, the method that does not reuse path information when performing WHCA* algorithm will get a shorter distance. However, the method that reuse path information is faster. There is a trade-off between travel distance and planning time.



3. Kiva picking实现

算法验证及实验分析

B. The Impact of Turning Factor

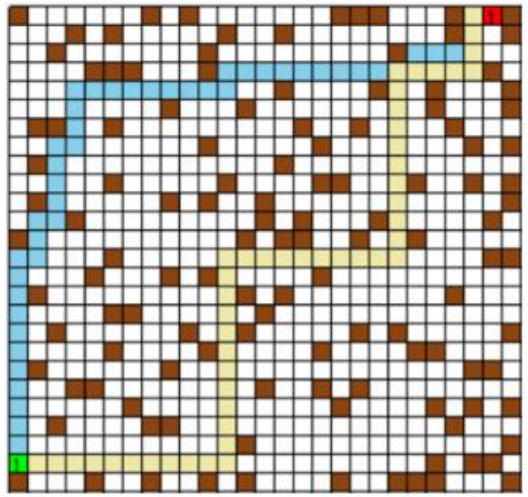


Fig. 9 Path comparison in random obstacle environment.

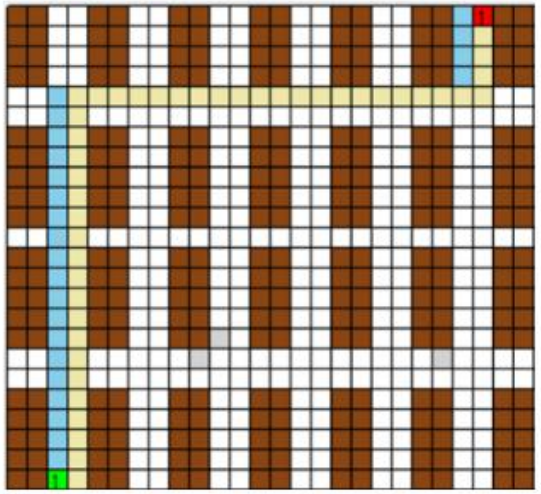


Fig. 10 Path comparison in a regular warehouse environment one.

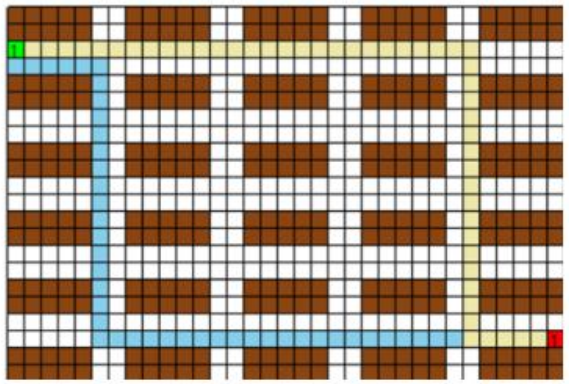


Fig. 11 Path comparison in a regular warehouse environment two.

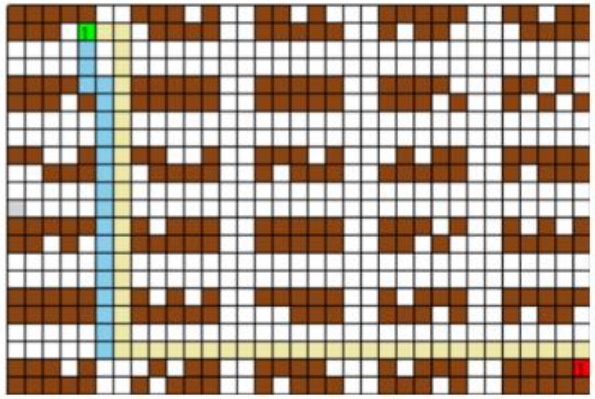


Fig. 12 Path comparison in a regular warehouse environment three.

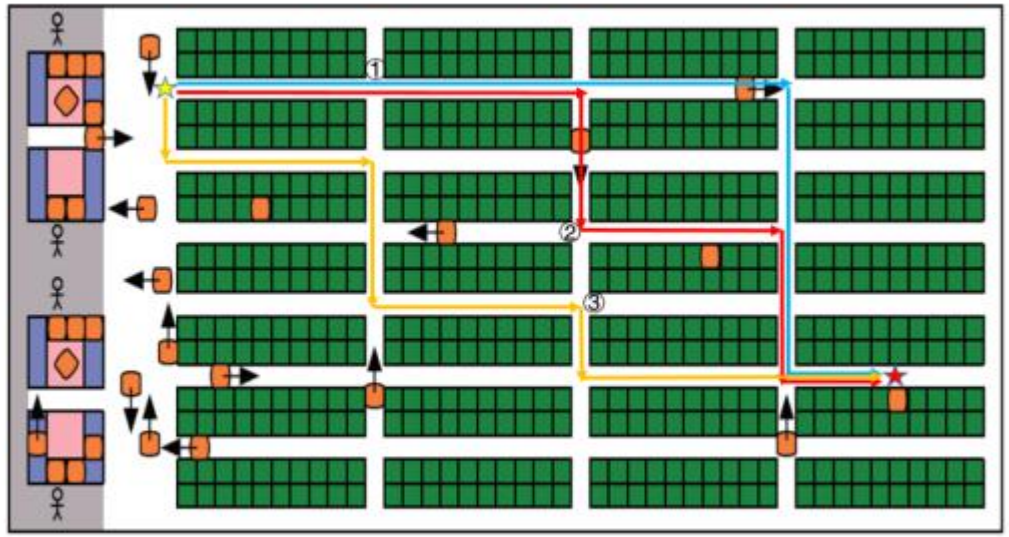


Fig. 5 Paths of the same length with different turns.