任务书

摘要

目录

# 绪论

## 研究背景及意义

2016年3月进行的围棋人机大战中， AlphaGo最终以4:1战胜了韩国九段名将李世石，引起了全世界的广泛关注，也让人工智能、机器人等概念进入了公众的视野。近年来，越来越多的机器人被应用到生产和生活中，家里的扫地机器人可以让你在结束一天的劳累工作后免去做家务的烦恼；高级轿车的自动驾驶技术可以让你有个免费安全的24小时司机。

现代社会高速发展，我们的日常生产和生活越来越离不开自动化和人工智能技术，越来越多的工场使用机器人等高科技设备来代替人工完成一些重复性高，精确度高，任务繁重的生产任务。创新工场的CEO李开复先生在2016年WISE大会上说：“在未来十年，世界上50%的工作，都会被人工智能所取代”。人工智能及自动化技术将在今后的人类社会中扮演至关重要的角色。绝大部分的机器人等智能设备在工作过程中，都要涉及到自动寻路这一环节，即机器人需要像人类一样在复杂的环境中实现从起始点到目标点的过程。机器人完成自动寻路需要依靠程序人员事先为其设计好的自动寻路程序，而程序如何在各种复杂的真实情况下快速、稳定地寻找出最佳路径就需要程序人员在众多已有的寻路算法中选择出最适用的算法，或者设计出更高效的寻路算法。

迷宫可以模拟复杂的工作环境，通过设计和实现一个可以自动生成复杂有解迷宫并自动寻路的系统，可以了解和掌握多种路径规划算法，比较各种算法之间的区别和应用范围，并根据实际情况对算法进行优化，从而在解决问题时提出更加高效、合理的路径规划算法，提高人工智能水平。

## 相关理论

路径规划问题指的是：在一个充满障碍物的环境中，如何避开障碍物，规划一条从起始点到达目标点的最优路径。路径规划问题涉及到迷宫的表示和自动生成、路径规划算法和搜索策略。迷宫自动生成算法负责如何生成一个复杂有解的迷宫并有效地表达出来；路径规划和搜索算法研究的是如何在已有迷宫的基础上求解出最优路径的过程。

### 1.2.1 复杂有解迷宫生成算法

常见的迷宫在程序中被描述为一个M\*N大小的矩阵，每个单元分别用0和1表示能否通过。迷宫自动生成算法就是通过一定的策略将初始全为0的M\*N矩阵变为一个任意两点都有路径可以到达的由0和1组成的M\*N矩阵。

递归回溯法是常见的生成复杂有解迷宫的算法。它的基本流程是：初始一个M\*N的矩阵，任意两个单元之间的四个方向的墙壁都是存在的。选择一个单元为开始单元，从当前单元四个方向的临界单元中选择一个合法的没有访问过的单元，并打通与它之间的墙壁，然后将当前单元压入栈中，并将打通的单元作为当前单元，接着继续寻找没有访问过的临界单元。如果当前单元已经没有未访问过的临界单元时，则从退回到栈顶单元作为当前单元，继续寻找它的临界单元是否可以访问。当栈为空（初始单元被退回）的时候，算法结束，M\*N的矩阵单元变为了一个有解的迷宫。

### 1.2.2 路径规划算法

路径规划算法是在之前生成的迷宫矩阵基础上，找到一条可以从初始点到达目标点的最优路径。常见的路径规划算法有宽度优先搜索（BFS）、深度优先搜索（DFS）、启发式搜索劫遗传算法等。

1. 宽度优先搜索（Breadth First Search）

宽度优先搜索算法（又称广度优先搜索）是最简便的图的搜索算法之一，这一算法也是很多重要的图的算法的原型。Dijkstra单源最短路径算法和Prim最小生成树算法都采用了和宽度优先搜索类似的思想。其别名又叫BFS，属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。

宽度优先搜索算法的具体描述为：已知图G=(V,E)和一个源顶点s，宽度优先搜索以一种系统的方式探寻G的边，从而“发现”s所能到达的所有顶点，并计算s到所有这些顶点的距离(最少边数)，该算法同时能生成一棵根为s且包括所有可达顶点的宽度优先树。对从s可达的任意顶点v，宽度优先树中从s到v的路径对应于图G中从s到v的最短路径，即包含最小边数的路径。该算法对有向图和无向图同样适用。

之所以称之为宽度优先算法，是因为算法自始至终一直通过已找到和未找到顶点之间的边界向外扩展，就是说，算法首先搜索和s距离为k的所有顶点，然后再去搜索和S距离为k+l的其他顶点。

广度优先搜索一般使用队列（queue）来实现，整个过程可以看做一个倒立的树形：

1、把根节点放到队列的末尾。

2、每次从队列的头部取出一个元素，查看这个元素所有的下一级元素，把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。

3、找到所要找的元素时结束程序。

4、如果遍历整个树还没有找到，结束程序

1. 深度优先搜索（Depth First Search）

深度优先搜索是一种在开发爬虫早期使用较多的方法。它的目的是要达到被搜索结构的叶结点(即那些不包含任何超链的HTML文件) 。在一个HTML文件中，当一个超链被选择后，被链接的HTML文件将执行深度优先搜索，即在搜索其余的超链结果之前必须先完整地搜索单独的一条链。深度优先搜索沿着HTML文件上的超链走到不能再深入为止，然后返回到某一个HTML文件，再继续选择该HTML文件中的其他超链。当不再有其他超链可选择时，说明搜索已经结束。

深度优先搜索算法是一种用于遍历或搜索树或图的算法。沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都己被探寻过，搜索将回溯到发现节点v的那条边的起始节点。这一过程一直进行到已发现从源节点可达的所有节点为止。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被访问为止，也属于盲目搜索。

深度优先搜索一般使用栈（stack）来实现，整个过程也可以想象成一个倒立的树形：

1、把根节点压入栈中。

2、每次从栈中弹出一个元素，搜索所有在它下一级的元素，把这些元素压入栈中。并把这个元素记为它下一级元素的前驱。

3、找到所要找的元素时结束程序。

4、如果遍历整个树还没有找到，结束程序。

1. 启发式搜索（Heuristically Search）

启发式搜索又称为有信息搜索(Informed Search)，它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的，这种利用启发信息的搜索过程称为启发式搜索。启发式策略可以通过指导搜索向最有希望的方向前进，降低了复杂性。通过删除某些状态及其延伸，启发式算法可以消除组合爆炸，并得到令人能接受的解(通常并不一定是最佳解)。

然而，启发式策略是极易出错的。在解决问题的过程中启发仅仅是下一步将要采取措施的一个猜想，常常根据经验和直觉来判断。由于启发式搜索只有有限的信息(比如当前状态的描述)，要想预测进一步搜索过程中状态空间的具体行为则很难。一个启发式搜索可能得到一个次最佳解，也可能一无所获。这是启发式搜索固有的局限性。这种局限性不可能由所谓更好的启发式策略或更有效的搜索算法来消除。一般说来，启发信息越强，扩展的无用节点就越少。引入强的启发信息，有可能大大降低搜索工作量，但不能保证找到最小耗散值的解路径(最佳路径)。因此，在实际应用中，最好能引入降低搜索工作量的启发信息而不牺牲找到最佳路径的保证。

用于评价节点重要性的函数称为估价函数，其一般形式为：

f(x) = g(x) + h(x)

式中：g(x)为从初始节点到节点x付出的实际代价；h(x)为从节点x到目标节点的最优路径的估计代价。启发性信息主要体现在h(x)中，其形式要根据问题的特性来确定。

1. 遗传算法（Genetic Algorithm

遗传算法是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

## 主要内容

智能迷宫游戏系统是一个可以自动生成随机复杂有解迷宫，然后选择不同的路径规划算法寻找一条从起始点到目标点的最优路径，并且可以将迷宫生成过程和寻路过程动态地展示出来的系统。

本文的研究工作主要包括：输入迷宫的宽度、高度和起始点位置，然后生成随机、复杂、有解的迷宫；选择多种不同的路径算法寻找从起始点到目标点的最优路径并对不同算法进行分析和比较；动态可控地展示迷宫生成和路径规划过程；总结不同路径规划算法的优势和适用范围，给出根据具体情况的改进等。

## 结构安排

本文一共包括五个章节，详细叙述了论文中所涉及到的相关理论算法以及作者所做的分析和总结工作，并且阐述了相应的智能迷宫游戏系统的设计及实现，最后对人工智能路径规划的未来发展做出展望。

第一章，绪论。主要介绍本课题的研究背景及其意义，对涉及到的相关路径规划算法理论做了简单的介绍，并说明了本课题的研究工作内容及本文结构。

第二章，系统需求。这一章主要阐述智能迷宫游戏系统的系统需求，包括需求概述，系统结构，各个模块的功能需求图解和非功能需求。

第三章，系统设计。这一章主要阐述智能迷宫系统的系统设计，包括概要设计，各个功能模块的详细设计、接口设计、流程分析和图解，模块之间的交互设计等。

第四章，算法设计。这一章主要阐述本文所研究的迷宫生成算法和路径规划算法，包括数据结构设计、算法流程图解和具体实现等。

第五章，系统实现。这一章主要阐述智能迷宫游戏系统及论文研究的总体情况，包括系统运行环境、运行截图展示，不同路径规划算法的执行数据及其分析、比较和总结。

总结和展望对整篇论文进行了总结，提出了研究过程的不足之处，并对未来的人工智能研究进行展望。

# 第二章 系统需求

## 2.1 需求概述

### 2.1.1 需求分析目标

为了明确本系统的需求，合理地安排软件的开发计划与进度；保证软件开发人员有目的、有安排地编写程序；使测试人员的测试工作完整、有据可循。

进行智能迷宫游戏系统进行需求分析的主要目的：

1)对智能迷宫系统的功能做一个全面的分析和阐述，帮助作者判断该系统实现功能的正确性、一致性和完整性，促使作者在对该系统进行概要设计之前周密地、全面地思考系统的需求。

2)全面地了解和描述实现该系统所需要的全部信息，为系统的设计、实现和测试提供一个标准

需求分析的具体内容可以总结为以下四个方面：系统的功能需求，系统的主要接口，系统的非功能性需求和技术难点等。

### 2.1.2 需求概述

智能迷宫游戏系统是一个可以让用户能够生动、直观地了解、学习复杂有解迷宫自动生成算法和各种路径规划算法的学习工具程序，它内置了几种常用的自动寻路算法和复杂有解迷宫生成算法，用户可以自定义迷宫的大小、起始位置等参数，选择不同的算法来自动生成复杂有解迷宫，能看到选择不同的自动寻路算法让游戏角色从起点走到迷宫终点的动态过程，并且能比较不同算法在不同维度的差别，从而能够直观地了解和学习各种寻路算法。

本系统的主要功能包括迷宫角色管理、复杂有解迷宫的绘制和生成、自动寻路算法的实现及不同算法的多维度比较等。用户输入自定义的迷宫参数，选择迷宫生成算法，然后点击生成迷宫按钮，接着在迷宫主界面可以看到迷宫的动态生成过程，并能够在右侧输出栏看到算法的耗时情况等信息。生成完迷宫之后，用户可以选择不同的自动寻路算法，然后点击寻路按钮，可以看到迷宫主角从起到一步一步寻路到终点的动态过程，并且右侧信息输出栏会显示算法的耗时情况等信息。用户还可以选择两种不同的寻路算法点击比较按钮，可以看到所选算法的稍微快一点的动态寻路过程，两种算法寻路都完成后，可以在右侧看到两种算法的比较图表。

### 2.1.3 系统结构

用户层

界面交互

游戏角色管理模块

智能迷宫自动生成模块

自动寻路模块

信息栏模块

用户通过与界面进行交互，输入一些操作命令，进入下层的逻辑模块。界面交互主要负责给用户提供美观、友好的操作界面，使得用户不仅可以通过方便、简洁、易于理解的操作步骤完成想要达到的目的，而且可以直观、清晰地看到操作带来的反馈和迷宫的处理过程及结果。

游戏角色管理模块主要负责对智能迷宫游戏系统的账号管理，登录和注销功能。智能迷宫自动生成模块主要负责根据用户选择的不同复杂有解迷宫自动生成算法来自动创建迷宫，绘制迷宫等逻辑。自动寻路模块主要负责根据用户选择的不同自动寻路算法动态地完成游戏角色的自动寻路过程，其中还要负责控制游戏角色的移动，寻路的动态过程的可视化绘制等逻辑，并把寻路算法执行的信息输出到信息栏和比较图表模块。信息栏和比较图表模块主要负责将寻路算法执行的信息有序、完整地输出到输出栏中，并且能够将不同寻路算法的数据以图表的形式显示出来。

## 2.2 游戏角色管理

游戏角色管理模块主要负责智能迷宫游戏系统的账号管理，账号登录和账号注销功能。该模块可以保证用户只有通过账号验证登录后才可以使用本系统。

### 2.2.1 账号管理

账号管理主要负责智能迷宫游戏系统的账号注册、密码修改等功能。用户在第一次打开本系统时需要登录账号才能使用，如果之前没有在该系统注册过账号，用户需要进行账号注册。用户需要输入符合要求的用户名、密码及重复密码，然后点击账号注册按钮来成功注册账号。成功注册账号后，会自动登录当前注册的账号。

用户登录过一个账号后，可以进入账户管理界面，进行账号信息修改。可以修改的信息包括用户的昵称、个性签名、密码等。修改密码时需要用旧密码做验证，验证通过后才可修改新的密码。成功修改密码后，用户需要用新密码重新登录一遍账号。

### 2.2.2 登录与注销

用户在第一次打开智能迷宫游戏的时候，需要进行账号登录验证。用户需要输入注册过的用户名、密码来验证身份，验证通过后才可以使用本系统。登录界面应该包括账号注册按钮，方便没有账号的用户进行注册操作。另外，登录界面还应该包扩自动登录选项，默认打开，即如果本次登录成功后，下次再打开系统时将默认登录本次的账号。

如果用户想要切换当前登录的账号或者退出当前账户时，可以通过账号注销功能，点击账号注销时，应该弹出二级窗口提醒用户是否确定退出注销当前账号，用户点击“确认”按钮后才注销账号并退回到登录界面。如果用户点击“取消”按钮，则取消注销操作。

## 2.3 迷宫生成

迷宫自动生成功能模块包括的主要功能有：用户自定义生成迷宫的大小、起始位置等信息，生成随机、复杂、有解的迷宫；将迷宫的动态生成过程动态地展现出来；将迷宫生成过程的算法执行信息发送给信息栏模块。

### 2.3.1 自定义迷宫

自定义迷宫功能允许用户输入预生成迷宫的高度、宽度、起始X坐标、起始Y坐标等信息。系统需要设计出人性化的设置界面：包含输入信息的合法性限制说明，键盘输入、鼠标点击上下箭头等多种输入方式，系统默认输入信息等。系统需要对用户输入的不合法信息进行处理，比如用户输入了超出宽度或者高度限制的起始位置坐标，系统需要自动将输入值改为限制的相应界值；用户输入了非奇数的宽度或者高度，系统需要自动将输入值改为最近似的合法奇数值。

另外，因为系统生成迷宫的过程是动态的，需要一定时间，如果在生成迷宫的过程中用户修改了迷宫的初始宽度、高度等信息，界面应该能够正确响应，并且系统应该将新的输入信息缓存，应用于下一次迷宫的生成，而不能影响当前正在生成的迷宫。

### 2.3.2 生成迷宫

当用户输入了预生成迷宫的初始信息后，点击生成迷宫按钮，开始生成迷宫。迷宫界面需要先根据用户输入的宽度和高度绘制一个相应大小的矩阵，初始时矩阵单元的背景颜色都是深灰色，表示不可通行。接着从用户设置的起始位置开始，系统逐步将当前算法执行结果对应的矩阵单元的背景颜色设置为白色，表示可以通行。整个迷宫生成过程是逐步、动态地展现出来的，系统会用绿色来表示算法递归的当前单元，用红色表示回溯时的当前单元。在整个迷宫生成期间，用户都可以设置单步的速度，即可以动态地控制迷宫的生成速度。

用户设置了不同的初始信息后，再次点击生成迷宫按钮后要清除上一次的迷宫，根据新的初始信息重新绘制新的迷宫，上一次的迷宫结果不能对新的迷宫生成过程有影响。即使没有改变迷宫初始信息，用户再次点击生成迷宫时，系统也应该清除上一次的迷宫，并生成相同大小、初始位置，但路径与上次不同的复杂有解迷宫，即有随机性。

如果用户在迷宫的生成过程中点击了生成迷宫或者其他功能模块的互斥按钮，系统应该通过弹出消息窗的方式提醒用户，等待本次生成迷宫过程完成后再进行操作，并无视掉相应的程序处理，以免出现不可控制的错误。

### 2.3.3 执行信息

在迷宫的动态生成过程中，迷宫生成模块应该将算法执行的每一步信息发送给信息栏模块，包括算法开始和结束提示、当前执行单元的位置信息、试探方向、是递归还是回溯和堆栈的深度等等。

发送的执行信息应该与用户看到的迷宫生成过程保持同步，不能出现用户看到的单元状态和输出信息不一致的情况。用户再次点击“生成迷宫”按钮时，程序应该清除上一次的执行信息。

## 2.4 自动寻路

自动寻路功能模块包含的主要功能有：用户可以选择不同的算法，迷宫路径规划，将寻路过程逐步地、动态地展现出来，将算法执行信息同步地发送给信息栏模块。

### 2.4.1 算法选择

本系统应该提供深度优先搜索、宽度优先搜索、启发式搜索、遗传算法等多种路径规划算法供用户选择。所有选择框应该为单选框，选中某一个算法时，其他算法选择框都应该是非选中状态，并且必选有一个算法是选中状态。默认状态下，系统选中深度优先搜索供路径规划使用。

如果在自动寻路过程用户切换了不同的算法，并不能影响本次的寻路过程，而是对下一次的寻路生效。

### 2.4.2 自动寻路

用户点击“自动寻路”按钮后，系统开始自动寻路。系统会从迷宫左上角的点开始，逐步将寻路算法执行结果对应的矩阵单元背景颜色设置为黄色，当算法寻路到迷宫右下角的终点时，会逐步回退重现最终路径，系统应该从终点开始逐步地将最终路径对应的矩阵单元的背景颜色设置绿色，表示系统最终寻找到的路径。最后要清除无效的试探信息，即把黄色的单元格设置为白色。整个过程是动态的，逐步的，用户可以随时设置寻路的单步速度以便于直观地观察算法的执行过程。

如果用户事先没有生成迷宫而直接点击“自动寻路”按钮，系统应该弹出窗口提示用户需要先生成迷宫。如果在自动寻路的过程中再次点击“自动寻路”按钮，系统应该弹出窗口提示用户当前正在寻路中，请等待本次寻路完成后再试。

一次寻路完成后，用户选中另一种路径规划算法或者继续使用当前的算法，然后点击自动寻路时，迷宫界面应该清除上一次的寻路结果，根据当前选中的算法开始新的寻路过程，并且上一次的寻路结果不能对本次的寻路造成任何影响。

### 2.4.3 执行信息

在路径规划的动态过程中，自动寻路模块应该将算法执行的每一步信息发送给信息栏模块，包括算法开始和结束提示、当前寻路结果单元的位置信息、试探方向、是递归还是回溯和堆栈的深度等等。

发送的执行信息应该与用户看到的自动寻路过程保持同步，不能出现用户看到的单元状态和输出信息不一致的情况。用户再次点击“自动寻路”按钮时，程序应该清除上一次的执行信息。

## 2.5 信息栏

信息栏模块包含的主要功能有：处理迷宫生成过程发来的信息，处理自动寻路过程发来的信息。信息栏界面应该以类似命令行模式的方式输出信息，当输出信息过多时，界面应该自动生成滚动条并滚动到最底部。另外，输出的算法信息应该简单易懂，与算法执行过程保持同步，便于用户理解算法执行过程。

### 2.5.1 迷宫生成

在迷宫生成模块开始生成迷宫时，信息栏界面需要清空上一次的输出信息，并输出算法开始执行的提示信息。在单步递归生成迷宫的过程中，信息栏界面需要输出单步算法执行的单元格位置、本次随机方向等信息。在回溯程序栈的过程中，信息栏界面需要指出回溯过程及其位置，算法执行深度等信息。生成迷宫算法执行完成后，信息栏界面需要输出算法完成的提示信息。

### 2.5.2 自动寻路

在自动寻路模块开始寻路时，信息栏模块需要清除上一次的执行信息，并输出当前选择的路径规划算法和算法开始的提示信息。在自动寻路的单步执行过程中，信息栏界面需要输出本次执行结果对应的矩阵单元的位置信息。在自动寻路回溯确认最终路径的过程中，信息栏界面需要输出最终路径对应的矩阵单元的位置信息。当自动寻路完成时，信息栏界面需要输出算法结束的提示信息。

## 2.6 速度设置

速度设置模块主要负责设置整个系统的单步执行速度。该模块应该由一个可以拖动的滑竿组成，让用户设置速度更加便捷。速度范围应该保证当用户不想重视过程时很快的完成，当用户需要仔细观察每一步时又可以有充足时间可以理解和反应。

系统默认的单步速度为50ms，保证用户的初次体验效。另外，用户可以随时地设置单步速度并立即应用到当前的过程中，做到想慢就慢，想快就快。

## 2.7 其他非功能需求

### 2.7.1 性能需求

对于两种复杂随机有解迷宫的自动生成算法，执行过程都不能出现可以感觉到的卡顿；对于自动寻路算法，算法的单步执行过程都不能感觉到明显的卡顿，整个寻路过程的动态展示过程都要流畅，不能有明显的卡顿。

### 2.7.2 扩展需求

对于两种复杂随机有解迷宫的自动生成算法，执行过程都不能出现可以感觉到的卡顿；对于自动寻路算法，算法的单步执行过程都不能感觉到明显的卡顿，整个寻路过程的动态展示过程都要流畅，不能有明显的卡顿。

### 2.7.3 界面和交互需求

界面主要包含三个部分，上方是操作栏窗口，又细分为三个部分：迷宫生成部分，自动寻路部分和寻路比较部分。下方左侧是整个迷宫游戏的窗口，该系统的主要窗口，下方右侧是信息输出栏，可以切换为文字信息输出和比较图表两种页签。

## 2.8 本章小结

总结

# 第三章 系统设计

## 3.1 概要设计

### 3.1.1 系统结构

### 3.1.2 设计要点

### 3.1.3 主要类图及关系

## 3.2 游戏角色管理

### 3.2.1 功能描述

### 3.2.2 接口设计

### 3.2.3 流程分析

### 3.2.4 时序图

## 3.3 智能迷宫自动生成

### 3.3.1 功能描述

### 3.3.2 接口设计

### 3.3.3 流程分析

### 3.3.4 时序图

## 3.4 自动寻路

### 3.4.1 功能描述

### 3.4.2 接口设计

### 3.4.3 流程分析

### 3.4.4 时序图

## 3.5 信息栏和比较图表

### 3.5.1 功能描述

### 3.5.2 接口设计

### 3.5.3 流程分析

### 3.5.4 时序图

## 3.6 功能模块之间的交互

### 3.6.1 交互关系

### 3.6.2 设计难点

## 3.7 本章小结

# 第四章 算法设计

## 4.1 迷宫自动生产算法

### 4.1.1 递归回溯

### 4.1.2 XXXXXX

## 4.2 自动寻路算法

### 4.2.1 宽度优先搜索

### 4.2.2 深度优先搜索

### 4.2.3 启发式搜索

### 4.2.4 遗传算法

## 4.3 本章小结

# 第五章 系统实现

## 5.1 运行环境

## 5.2 数据分析与比较

### 5.2.1 执行数据

### 5.2.2 分析与比较

## 5.3 寻路算法总结

## 5.4 系统展示

# 总结与展望

# 致谢

# 参考文献