华南理工大学

《智能机器人技术》课程实验报告

实验题目： 机器人避障

姓名： 黄鑫 学号： 202065073276

班级： 20计联 组别：

合作者： 罗胤仪

指导教师： 张平

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：  熟悉机器人仿真软件（例如Simbad、 Webots、TeamBots、Player/Stage/Gazebo、 MotionPlaner等）的使用方法；掌握若干机器人 路径规划算法。  实验要求：  在机器人软件平台上建立一个包含 若干个静止障碍物和运动障碍物的仿真环境，设 定机器人的起始点和终点后，机器人能够规划出 一条从起始点到目标点的安全路径。查阅相关路径规划算法，实现一种以上算法并相互比较。要求给出源代码、试验结果并且进行演示。  【实验环境】  操作系统：Windows 10 |
| **实验内容** |
| 【实验过程】  人员分工：  本人负责DWA算法的设计与实现。罗胤仪负责Bug算法的设计与实现。  **一、实验步骤：**  **步骤一：**  **下载webots。**网址如下：<https://www.cyberbotics.com/> 。也可以到webots的github上的release上下载自己想要的版本。  **步骤二：**  **熟悉webots的操作。**  下面介绍其主要的部分。  1. 节点部分（位于界面的最左边）    图中箭头所指部分就是添加节点的按钮，也可以选择一个节点后右键添加节点。添加后，界面如下。    我们主要是使用Base nodes和PROTO nodes (Webots Projects)。节点根据需要自行选择。  2. 模拟界面（最中间）    这里是模拟机器人和环境的场景。点击某个object后，左边的节点树会显示相应的属性且会在场景中显示改object的坐标系。  3. 代码（controller）编辑器（最右边）    Webots支持多种语言编写的controller。如python，c/c++，java。  4. 控制台（最底部）  这里是程序运行时的控制台。可以打印自己需要的信息，也会有报错信息。  **步骤三：**  **熟悉一些常用的传感器以及API的调用**。下面介绍本实验所用主要的传感器及其API  1. Camera  Camera.enable(samplingPeriod)#启动camera  Cameara.recognitionEnable(samplingPeriod)#允许识别  Camera.getRecognitionObjects()#获取识别的物体  CameraRecognitionObject.get\_position()#获取相对位置 注意这里和官网提供的API不同，因为官网是错的，CameraRecognitionObject的API的命名风格是用下划线而不是驼峰命名  2. GPS  GPS.enable(samplingPeriod)#启动GPS  GPS.getValues()#获取GPS装置在世界坐标中的位置  3.InertialUnit  InertialUnit.enable(samplingPeriod)#启动InertialUnit  InertialUnit. getRollPitchYaw()#获取Roll角、Pitch角、Yaw角，单位是弧度制  4.DisranceSensor  DistanceSensor.enable(samplingPeriod)  DistanceSensor.getValue()    **步骤四：**  **熟悉各种路径规划的算法。**  下面介绍本次实验中本组同学提供的算法。  **（1）全局搜索法：**  本算法的基本思想是预先建立好地图，如一个二维数组。然后利用搜索算法，如广度优先算法，深度优先算法，A\*算法，回溯算法，剪枝算法，双向搜索算法等等，选择最合适的路径。最后控制机器人按照路径移动即可。  本算法优点就是简单易操作易实现。缺点其一是需要预先创建地图，如果地图过大，会耗费很大的内存空间，在实际情况中可能会存在内存不足的情况。其二，这是一种静态的模型，并没有考虑移动障碍物的问题，地图上的障碍物的位置可以改变。针对第二个缺点，我们觉得可以使用GPS获取全局的信息，并且不断发送给当前的机器人。但是问题是GPS是某个物体转载的，获取的数值是当前物体的坐标，要传输GPS的值时需要用到通信。这里就会有两个问题，移动的物体是否有GPS和移动的物体是否能与当前的机器人进行通信。因此此方法存在很多不实际的地方，被我们舍弃。  **（2）遗传算法：**  遗传算法基本思想是模拟生物进化的过程。基本过程就是定义种群，定义基因码，定义适应度函数，随机交换基因码，基因码按照一定概率突变，然后不断重复交换基因码和突变，直到找到某个可行解或者最优解。遗传算法是接近优化问题的算法，所以我们需要把路径规划问题转为优化问题解决。我们如果要实现这个算法，我们需要定义合适的适应度函数，类似A\*的启发式函数，以及路径的基因码表示即某种自定义的数据结构。  遗传算法很强大，可以解决很多NP问题。但是也有一些我们考虑到的缺点。一、算法中的算子即某些超参数的选择很需要经验。二、因为存在随机因子，计算复杂度和时间不可预测，可能会特别长，此时机器人反应可能就会比较迟钝。三、实现过程比较复杂。因此该算法被我们舍弃。  **（3）Bug算法：**  Bug算法的基本思想是模拟一个bug的移动。当bug检测到一个障碍物时，会向前靠近，并且控制自己和障碍物保持一定距离，然后围绕障碍物移动，找到一个最优离开障碍物的点离开，不断向目标移动。若没有检测到障碍物就直接向目标直线运动。  该算法有三种形式：Bug1，Bug2，Tangent bug。  Bug1：    过程：  1. 用一个直线连接起始点和目标点。机器人沿着直线行走直到遇到第一个障碍物，相遇的点命名为(H means Hit)。  2. 机器人沿着障碍物行走一圈直到再次遇到。这时候就可以计算出障碍物的边界距离最近的点(L means Leave)。然后从这个点离开。  3. 机器人沿着起始点和目标点的直线行走，遇到障碍物就执行1和2两步骤。  4. 最终机器人走到目标。  Bug2：  Bug1算法**每次遇到障碍物就要绕着障碍物行走一圈**,这样大大降低了效率,于是在Bug1的算法上加以改进,形成了Bug2算法.    过程：  1.同Bug1一样,机器人沿着起始点和目标点连成的直线行走。  2.遇到障碍物时,机器人沿着障碍物行走直到碰到起始点和目标点连成的直线。  3. 重复上述过程,直到走到目标位置。  Tangent Bug：  Tangent Bug算法是对Bug2算法的改进。它利用机器人上距离传感器的读数对障碍物做出提前规避，可以获得更短更平滑的机器人路径。假设机器人上安装有360°激光雷达(或者红外距离传感器),那么我们可以测得每束光线到达障碍物的距离.下图中x代表机器人的位置,细线代表发出的光线,粗线代表了光线被遮挡(说明机器人无法到达这些位置)。    我们用标记光线与障碍物相交的边界点。      **算法过程：**  1. 机器人直接沿着目标方向按直线行走,直到**激光雷达检测到了障碍物**.  2. 用虚线的圆表示激光雷达的**检测范围**.  3. 标记出，然后机器人向着**启发距离最小**的前进  **关于的选择**  启发距离是人为规定的,比如我们可以用来表示.不同的目标位置会导致机器人对每一步的选择不同. 当然的值是实时更新的,这将导致最后机器人靠近障碍物时行走的轨迹是一条曲线而不是直线。  **关于激光**  使用Tanget Bug可以有效的提升整体效率,这是激光雷达测量半径为R时形成的轨迹.    当测量半径退化为0时(变成了接触传感器),Tangent Bug算法就退化成了Bug2算法。    Bug算法是最早被提出来的路径规划算法，很符合直觉，尤其是在完全一无所知的环境下，Bug算法往往能够取得非常不错的效果。Bug算法易实现，并且是会考虑局部的因素。会考虑到移动的障碍物的问题，不需要地图的构建。在本次实验中，我们采用了这个算法，我们将激光换成了距离传感器。  **（4）DWA(Dynamic Window Approach)算法**  DWA算法（dynamic window approach），其原理主要是在速度空间中**采样多组速度**，并模拟出这些速度在一定时间内的**运动轨迹**，并通过**评价函数**对这些轨迹进行评价，**选取最优轨迹**对应的驱动机器人运动。中v代表线速度 ，w代表角速度。  对于非全向机器人，即不能纵向移动，只能前进和旋转。则运动学模型为：  上面的v(t)是机器人坐标系中x方向的速度。  对于全向机器人，在机器人坐标系中，不仅有x方向的速度，还有y方向的速度，另外还可以旋转。运动学模型如下：  **移动机器人自身受最大速度和最小速度的限制**  为机器人能够到达的所有矢量速度的集合；机器人受到最大最小线速度和角速度影响。  **移动机器人受电机性能的影响**  **移动机器人受障碍的影响**  在上述三条约束条件的限制下，速度空间会有一定的范围，另外随着电机的线加速度、角加速度进行变换，速度空间会动态变化，我们将其称为**动态窗口**。在满足约束条件的情况下，进行采样，可以得到相应的轨迹。  **评价函数**  其中，为方位角评价函数：评价机器人在当前的设定的速度下，轨迹末端朝向与目标点之间的角度差距； 主要的意义为机器人处于预测轨迹末端点位置时与地图上最近障碍物的距离，对于靠近障碍物的采样点进行惩罚，确保机器人的避障能力，降低机器人与障碍物发生碰撞的概率；为当前机器人的线速度，为了促进机器人快速到达目标； 、、 、 为权重。当然，可以对评价函数进行优化，添加更多的评价函数指标。  该算法有很多优点。一、计算复杂度低，考虑到速度和加速度的限制 ，只有安全的轨迹会被考虑，且每次采样的时间较短，因此轨迹空间较小。二、可以实时避障，但是避障效果一般。三、适用于两轮差分和全向移动模型。  该算法的缺点有：一、只是模拟并评价了下一步，在遇到如”C”字形障碍时，不能很好地避障。二、动态避障效果不是很好，对于移动的物体，无法预测该移动物体的轨迹。三、只是局部最优，非全局最优。四、不适用于阿克曼模型。  根据查阅的资料，我们找到的具体实现都是基于静态的模型即所有障碍物时静止的。我们查阅各种资料最终找到该问题的解决方案。我们使用Webots自带的Camera节点来获取机器人当前位置附近的障碍物的相对于机器人的位置信息，然后根据下面的旋转矩阵，将相对坐标转为世界坐标，然后就可以实现动态的避障。  二、实验主要过程：  **Tips**：我们在点击一个节点后会有一个Def，这里我们可以设置别名用来获取该节点，否则就要通过name属性获取该节点。  **DWA part:**  1.搭建场景。新建世界后，通过添加节点，然后改变节点的translation的值就可以控制物体在世界中的位置。然后在children属性下添加Shape属性。在Shape下的appearance下选择PBRApearance。Geometry选择自己想要，这里我们选择Sphere。然后双击recognitionColors。双击的目的是为了相机能够识别该物体。  2.移动障碍物的控制。添加一个base node下的robot节点。我们将这个节点作为Supervisor。将该节点中controller属性选择为我们写的supervisor\_controller.py并且将supervisor属性设置为true。  3.移动机器人的添加。调整translation属性来调整机器人的位置。在children下添加Camera，InertialUnit，Transform，两个HingeJoint，两个solid，一个GPS。  4.编写controller。这里我们有两个controller，一个是supervisor的，一个是移动机器人的。  Bug Algorithm part：  1.搭建场景。新建世界后，通过添加节点，然后改变节点的translation的值就可以控制物体在世界中的位置。然后在children属性下添加Shape属性。在Shape下的appearance下选择PBRApearance。Geometry选择自己想要。然后双击recognitionColors。双击的目的是为了相机能够识别该物体。  2.移动障碍物的控制。添加一个base node下的robot节点。我们将这个节点作为Supervisor。将该节点中controller属性选择为我们写的supervisor\_controller.py并且将supervisor属性设置为true。  3.移动机器人的添加。在PROTO中选择e-puck机器人。  4.编写controller。这里我们有两个controller，一个是supervisor的，一个是移动机器人的。  实验结果  DWA算法中，机器人遇到移动障碍物的转弯：    遇到静态障碍物的移动：    Bug算法  贴障碍物：        参考文献  **（1）Webots官网：**[**https://www.cyberbotics.com/doc/reference**](https://www.cyberbotics.com/doc/reference)  **（2） Zohaib, Muhammad, et al. "IBA: Intelligent Bug Algorithm–A novel strategy to navigate mobile robots autonomously." International Multi Topic Conference. Springer, Cham, 2013.**  **（3） Ogren, Petter, and Naomi Ehrich Leonard. "A convergent dynamic window approach to obstacle avoidance." IEEE Transactions on Robotics 21.2 (2005): 188-195.**  **（4） Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance." IEEE Robotics & Automation Magazine 4.1 (1997): 23-33.**  **（5） Ng, James, and Thomas Bräunl. "Performance comparison of bug navigation algorithms." Journal of Intelligent and Robotic Systems 50.1 (2007): 73-84.** |
| **小结** |
| 本实验我们主要使用Bug算法和DWA算法做避障和路径规划。两个算法有很大不同。Bug类似摸石头过河，DWA类似下棋中的各种假设。本实验中对于移动障碍物的实现使用了Supervisor。这个节点可以控制所有物体的属性。**本次实验的DWA算法相比于一般的静态模型使用了camera和inertialUnit的结合来获取障碍物的绝对位置，解决了地图和动态障碍物避障的问题。**  **本实验中有一点不足的地方就是速度的控制，对于移动障碍物需要判断其速度并且根据真个速度调整机器人的速度，但是我们没这方面经验。对于本实验的模型，我们忽略这个问题。**  Webots使用心得：在使用webots的过程中要随时保存，不然在重新模拟时会丢失所作的工作。自带的编辑器不好用，建议使用Visual Studio Code来debug。建立controller的时候要用菜单栏中向导的新建controller不然webots会报找不到controller的错误。 |
| **指导教师评语及成绩** |
| 评语：  成绩：           指导教师签名：                                                 批阅日期： |