智能技术与系统综合实验
实验手册

任务 1：youBot 麦克纳姆轮智能车底盘运动学仿真

1. 实验背景

编写算法，根据输入，计算并控制 youBot 麦克纳姆轮智能车的运动。使用仿真软件观察智能车的运动动画（示例如图 1）。麦克纳姆轮智能车通过配备滚轮的特殊构造轮胎，使得智能车不仅可以前进后退外，还可以进行横向移动，其典型几何结构见图 2 所示。本次实验的算法主函数为 nextconf=NextState(current_conf, control, dt, max_speed)，其输入输出定义如下：

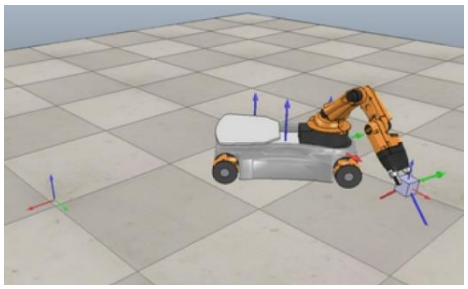


图 1. 配备机械臂的麦克纳姆轮智能车

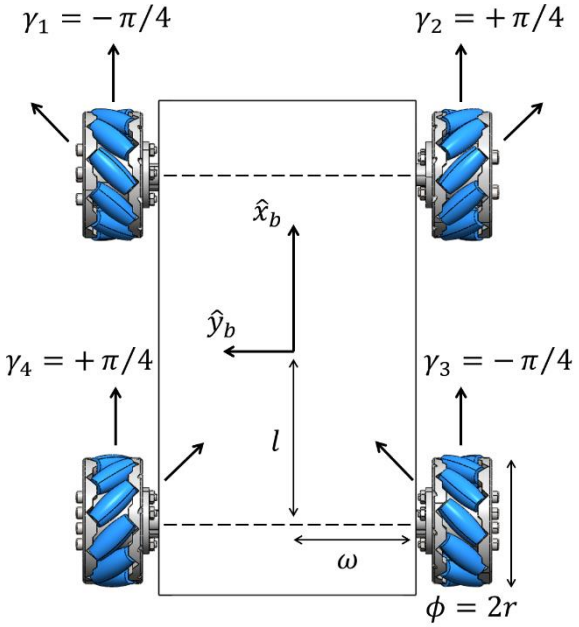


图 2. 麦克纳姆轮车几何参数

2. 输入

1) 一个 12 维的状态向量，表达当前机器人的位形(3 维表示智能车壳体位形，5 维表示机械臂位形，剩下 4 维表示智能车轮子的转角)：$q = [\phi, x, y, J1, J2, J3, J4, J5, \theta1, \theta2, \theta3, \theta4]$，初始都为 0。

2) 一个 9 维的输入向量，控制机械臂关节速度$\dot{J}$(5 维向量)，和控制智能车轮子转速$\dot{\theta}$(4 维向量)：$u = [\dot{J}1, \dot{J}2, \dot{J}3, \dot{J}4, \dot{J}5, \dot{\theta}1, \dot{\theta}2, \dot{\theta}3, \dot{\theta}4]$。

3) 步长Δt（例如$0.01sec$），总运行时间t。

4) 限制机械臂关节速度的最大值，和限制麦克纳姆轮转速的最大值：超过极值时按照饱和值计算。

3. 输出

一个 12 维的状态向量，表达Δt后机器人的位形。函数 NextState 基于简单的一阶欧拉步长方法：

1) 新时刻的机械臂关节角度=上一时刻机械臂关节角度J+关节速度$\dot{J}\times$步长Δt（即$J_{k+1}=J_k+\dot{J}\times\Delta t=J_k+\Delta J$）；

2) 新时刻的智能车轮子角度=上一时刻轮子的角度θ+轮子转速$\dot{\theta}\times$步长Δt（即$\theta_{k+1}=\theta_k+\dot{\theta}\times\Delta t=\theta_k+\Delta\theta$）；

3) 新时刻的智能车壳体的位形由下列公式定义：

$$\mathcal{V}_b=F\Delta\theta=\frac{r}{4}\cdot\begin{bmatrix}-1/(l+w) & 1/(l+w) & 1/(l+w) & -1/(l+w)\\ 1 & 1 & 1 & 1\\ -1 & 1 & -1 & 1\end{bmatrix}\cdot\Delta\theta \tag{1-1}$$

$$\mathcal{V}_b=[\omega_{bz}\quad v_{bx}\quad v_{by}]^T \tag{1-2}$$

$$If\ \omega_{bz}=0,\Delta q_b=\begin{bmatrix}\Delta\phi_b\\ \Delta x_b\\ \Delta y_b\end{bmatrix}=\begin{bmatrix}0\\ v_{bx}\\ v_{by}\end{bmatrix} \tag{1-3}$$

$$If\ \omega_{bz}\neq 0,\Delta q_b=\begin{bmatrix}\Delta\phi_b\\ \Delta x_b\\ \Delta y_b\end{bmatrix}=\begin{bmatrix}\omega_{bz}\\ (v_{bx}sin\omega_{bz}+v_{by}(cos\omega_{bz}-1))/\omega_{bz}\\ (v_{by}sin\omega_{bz}-v_{bx}(cos\omega_{bz}-1))/\omega_{bz}\end{bmatrix}$$

$$\Delta q=\begin{bmatrix}1 & 0 & 0\\ 0 & cos\phi_k & -sin\phi_k\\ 0 & sin\phi_k & cos\phi_k\end{bmatrix}\Delta q_b \tag{1-4}$$

$$q=[\phi,x,y,J1,J2,J3,J4,J5,\theta1,\theta2,\theta3,\theta4] \tag{1-5}$$

$$q_{k+1}(1:3)=q_k(1:3)+\Delta q^T \tag{1-6}$$

$$q_{k+1}(4:8)=q_k(4:8)+0 \tag{1-7}$$

$$q_{k+1}(9:12)=q_k(9:12)+[\dot{\theta}1,\dot{\theta}2,\dot{\theta}3,\dot{\theta}4]\cdot\Delta t \tag{1-8}$$

其中$l=0.47/2$为麦克纳姆轮轴到智能车中心的轴向距离，$w=0.3/2$为麦克纳姆轮到智能车中心的径向距离，$r=0.0475$为麦克纳姆轮半径，$\Delta\theta=\dot{\theta}\times\Delta t$为 4 个轮子经过步长$\Delta t$后转过的角度，$\mathcal{V}_b$为智能车车体运动旋量，$\omega_{bz}$为车体绕$z$轴旋转的角速度，$v_{bx}$和$v_{by}$分别为车体沿$x$和$y$方向的线速度，$\phi$为车体角度。

4. 测试主函数 NextState

1) 运行 CoppeliaSim（**附录 1** 展示了如何安装并运行仿真软件），打开 scene->Scene3_youBot，移动滑块或输入值以控制智能车车体的位形及机械臂的角度，熟悉智能车的状态$q=[\phi,x,y,J1,J2,J3,J4,J5,\theta1,\theta2,\theta3,\theta4]$。

2) 设置步长$\Delta t=0.01$，总时间$t=1$。编写并运行一个循环函数如 runNextState，每次使用相同的常值输入$u=$

$[\dot{j}1, \dot{j}2, \dot{j}3, \dot{j}4, \dot{j}5, \dot{\theta}1, \dot{\theta}2, \dot{\theta}3, \dot{\theta}4]$，因此将反复调用 NextState 函数如 100 次。

3) 记录每次的输出状态(12 维状态变量)，并生成一个 csv 文件(csv：comma separated values（**附录 2** 展示了 Matlab 中如何记录并生成 csv 文件)。每行包含 12 维状态变量，最后第 13 位写 0 或 1，表示机械臂末端执行器的开或者闭)，并拷贝到 V-REP_scenes 文件夹下。
4) 运行 CoppeliaSim，打开 scene->Scene4_youBot_csv，输入 csv 文件（完整路径，例如 C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\scenes\V-REP_scenes\myTest.csv），观察仿真模拟。

5. 实验目标

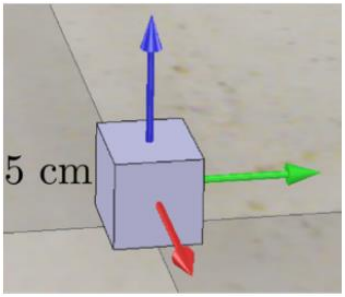
编写算法并仿真模拟如下场景（不控制机械臂，因此 csv 文件相应的位置填写 0 即可(公式 1-7 所示))：

1) $u = (0,0,0,0,0,10,10,10,10)$，时间 1 秒。麦克纳姆轮智能车将向前行驶约 0.475 米。
2) $u = (0,0,0,0,0,-10,10,-10,10)$，时间 1 秒。麦克纳姆轮智能车将向左平移约 0.475 米。
3) $u = (0,0,0,0,0,-10,10,10,-10)$，时间 1 秒。麦克纳姆轮智能车将沿$z$轴正向原地旋转约 1.234 弧度。
4) 限制轮子转速最大值为$[-5,5]$，再次测试 1)~3)，观察结果。
5) 编写控制量u，控制麦克纳姆轮智能车沿着一个正方形轨迹进行移动(尝试多种控制：如正常行驶、平移、侧移、旋转等)。

任务 2：youBot 麦克纳姆轮智能车+机械臂物体抓取轨迹生成

1. 实验背景

编写算法，为配备机械臂的麦克纳姆轮智能车生成一条抓取物体的轨迹，共 8 端。每个轨迹位形T_{se}代表末端执行器坐标系$\{e\}$相对于空间固定坐标系$\{s\}$的位形。仿真的主函数为 result_traj=TrajectoryGenerator(Tseinitial, Tscinitial, Tscfinal, Tcegrasp, Tcestandoff, k)，其输入输出为如下定义所示：



2. 输入

1) 状末端执行器（机械臂的爪子）在生成轨迹中的初始位形$T_{se,initial}$，（s 代表 space，e 代表 end-effector，即 e 相对于 s 的初始位形）。

$$T_{se,initial} = \begin{bmatrix} 0 & 0 & 1 & 0.5518 \\ 0 & 1 & 0 & 0.0000 \\ -1 & 0 & 0 & 0.4012 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-1}$$

2) 物块初始位形$T_{sc,initial}$，（c 代表 cube，即 c 相对于 s 的初始位形）。

$$T_{sc,initial} = \begin{bmatrix} \cos(0) & -\sin(0) & 0 & 1.0000 \\ \sin(0) & \cos(0) & 0 & 0.0000 \\ 0 & 0 & 1 & 0.0250 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-2}$$

3) 物块期望的终点位形$T_{sc,final}$，即 c 相对于 s 的最终位形。

$$T_{sc,final} = \begin{bmatrix} \cos(-\pi/2) & -\sin(-\pi/2) & 0 & 0.0000 \\ \sin(-\pi/2) & \cos(-\pi/2) & 0 & -1.000 \\ 0 & 0 & 1 & 0.0250 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-3}$$

4) 末端执行器即将开始抓取（爪子开始闭合时），相对于物块的位形$T_{ce,grasp}$。

$$T_{ce,grasp} = \begin{bmatrix} \cos(3\pi/4) & 0 & \sin(3\pi/4) & 0.0050 \\ 0 & 1 & 0.0000 & 0.0000 \\ -\sin(3\pi/4) & 0 & \cos(3\pi/4) & -0.005 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-4}$$

5) 末端执行器相对于物块、位于物块正上方的一个待命位形$T_{ce,standoff}$。末端执行机构在抓取物体前、抓取物体后、放置物体前、放置物体后都会处于这个位形。例如从$T_{ce,standoff}$垂直往下移动 0.05 米可达到$T_{ce,grasp}$。

$$T_{ce,standoff} = T_{ce,grasp},\ T_{ce,standoff}(3,4) = 0.05 \tag{2-5}$$

6) 每秒的参考轨迹位形数目k。例如$k = 1$时意味着轨迹中每秒有 100 个位形。

3. 输出

包含所有部分的完整轨迹，共 8 段（每段轨迹的生成方式参考**附录 3**）：

1) $T_{se,initial}$到$T_{sc,initial} \cdot T_{ce,standoff}$（矩阵相乘）；

2) $T_{sc,initial} \cdot T_{ce,standoff}$到$T_{sc,initial} \cdot T_{ce,grasp}$；

3) 关闭爪子（13 维位形向量中前 12 维等于上一段轨迹最后的位形，并保持不变，第 13 维写 1）；

4) $T_{sc,initial} \cdot T_{ce,grasp}$到$T_{sc,initial} \cdot T_{ce,standoff}$；

5) $T_{sc,initial} \cdot T_{ce,standoff}$到$T_{sc,final} \cdot T_{ce,standoff}$；

6) $T_{sc,final} \cdot T_{ce,standoff}$到$T_{sc,final} \cdot T_{ce,grasp}$；

7) 打开爪子（13 维位形向量中前 12 维等于上一段轨迹最后的位形，并保持不变，第 13 维写 0）；

8) $T_{sc,final} \cdot T_{ce,grasp}$到$T_{sc,final} \cdot T_{ce,standoff}$。

将得到的每时刻的轨迹系列改写为行向量的形式，进行 csv 文件格式输出，如：

$$T_{se} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow [r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}, p_x, p_y, p_z, 0/1] \tag{2-6}$$

4. 实验目标

编写算法并检验生成的轨迹：

1) 将得到的轨迹转换成如公式 2-6 所示的形式，并生成一个 csv 文件。每行为 13 维向量。将 csv 文件拷贝到 V-REP_scenes 文件夹下。

2) 运行 CoppeliaSim，打开 scene-> Scene8_gripper_csv，输入 csv 文件，观察仿真模拟。

附录 1

安装仿真模拟软件

1) 复制并粘贴 CoppeliaSim_Edu_V4_0_0_Setup 到选定的文件夹；
2) 双击并安装；
3) 程序将被安装在 C 盘(C:\Program Files\CoppeliaRobotics)；
4) 解压 V-REP_scenes-Nov2019.zip 到 C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\scenes
5) 双击 CoppeliaSim 运行(位于桌面快捷方式或 C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu)；
6) 进入后点击 File->Open scene。双击 V-REP_scenes，选择 Scene4_youBot-csv
7) 点击“Play”按钮(一个蓝色的三角形按钮，位于屏幕中央上方；不需要进行仿真时请点击“Stop”按钮，即蓝色的正方形，以降低计算机负荷)；
8) 此时需要输入一个 csv 文件的路径
9) 一旦输入可行的 csv 文件路径，软件将按照 csv 进行机器人运动模拟。

附录 2

Matlab 环境中书写 csv 文件并保存

```
q_k = [1,2,3,4,5,6]
csvwrite('q_k_csv_file.csv',q_k);
```

附录 3

Matlab 代码：旋量轨迹生成

```
function traj = ScrewTrajectory(Xstart, Xend, Tf, N, method)
% Takes Xstart: The initial end-effector configuration,
%       Xend: The final end-effector configuration,
%       Tf: Total time of the motion in seconds from rest to rest,
%       N: The number of points N > 1 (Start and stop) in the discrete
%          representation of the trajectory,
%       method: The time-scaling method, where 3 indicates cubic
%               (third-order polynomial) time scaling and 5 indicates
%               quintic (fifth-order polynomial) time scaling.
% Returns traj: The discretized trajectory as a list of N matrices in
SE(3)
%               separated in time by Tf/(N-1). The first in the list is
%               Xstart and the Nth is Xend .
% This function calculates a trajectory corresponding to the screw
motion
% about a space screw axis.
% Example Input:
%
% clear; clc;
% Xstart = [[1 ,0, 0, 1]; [0, 1, 0, 0]; [0, 0, 1, 1]; [0, 0, 0, 1]];
```

```
% Xend = [[0, 0, 1, 0.1]; [1, 0, 0, 0]; [0, 1, 0, 4.1]; [0, 0, 0,
1]];
% Tf = 5;
% N = 4;
% method = 3;
% traj = ScrewTrajectory(Xstart, Xend, Tf, N, method)
%
% Output:
% traj =
%    1.0000         0         0    1.0000
%         0    1.0000         0         0
%         0         0    1.0000    1.0000
%         0         0         0    1.0000
%
%    0.9041   -0.2504    0.3463    0.4410
%    0.3463    0.9041   -0.2504    0.5287
%   -0.2504    0.3463    0.9041    1.6007
%         0         0         0    1.0000
%
%    0.3463   -0.2504    0.9041   -0.1171
%    0.9041    0.3463   -0.2504    0.4727
%   -0.2504    0.9041    0.3463    3.2740
%         0         0         0    1.0000
%
%   -0.0000    0.0000    1.0000    0.1000
%    1.0000   -0.0000    0.0000   -0.0000
%    0.0000    1.0000   -0.0000    4.1000
%         0         0         0    1.0000

timegap = Tf / (N - 1);
traj = cell(1, N);
for i = 1: N
    if method == 3
        s = CubicTimeScaling(Tf, timegap * (i - 1));
    else
        s = QuinticTimeScaling(Tf, timegap * (i - 1));
    end
    traj{i} = Xstart * MatrixExp6(MatrixLog6(TransInv(Xstart) * Xend)
* s);
end
end
```

附录 4

Matlab 中添加 mrlib 库到默认路径：mrlib 包含诸多有用的函数，将其添加到默

认路径后，每次启动 Matlab 将不需要手动添加。具体方法为拷贝 mrlib 文件夹到任意路径下（推荐到 C:\Users\Admin\Documents\MATLAB），点击“Set Path（添加路径）”-> “Add with Subfolders”->选择 mrlib 并确认。