深 圳 大 学 实 验 报 告

课程名称：机器人学导论

实验项目名称：Time and Motion

学院：电子与信息工程学院

专业：电子信息工程

指导教师：郑琪

报告人：陈闻天 学号：2023280259

班级：04

实验时间：2024 年 9 月 24 日

实验报告提交时间：2024 年 10 月 7 日

教务处制

Aim of Experiment:

1. Learn time-varying position and how to compute relative velocity and angular velocity.
2. Master how to generate a temporal sequence of poses, a trajectory, that smoothly changes from an initial pose to a final pose.

Experiment Content:

(1)3.1.4 Incremental Rotation

(2)3.3.1 Smooth One-Dimensional Trajectories

(3)3.3.2 Multi-Axis Trajectories

(4)3.3.3 Multi-Segment Trajectories

(5)3.3.4 Interpolation of Orientation in 3D

(6)3.3.4.1 Direction of Rotation

(7)3.3.5 Cartesian Motion in 3D

Experiment Process：

(1)3.1.4 Incremental Rotation: compute approximate the angular velocity vector and compare the speed of different ways of computing them.
(2)3.3.1 Smooth One-Dimensional Trajectories: generate trajectory and plot some points with time interval.
(3)3.3.2 Multi-Axis Trajectories: plot x- and y- axis on the same 2D coordinate frame.
(4)3.3.3 Multi-Segment Trajectories: move smoothly along a path and avoid obstacles.
(5)3.3.4 Interpolation of Orientation in 3D: interpolate and smoothly change from orientation with some function.
(6)3.3.4.1 Direction of Rotation: try different direction of rotation and compare the cost of them.
(7)3.3.5 Cartesian Motion in 3D: find a smooth path between two 3D poses referred to as Cartesian motion.

Data Logging and Processing:

3.1.4 Incremental Rotation

```
rotmx(0.001)
Rexact = eye(3); Rapprox = eye(3);  % null rotation
w = [1 0 0]';   % rotation of 1rad/s about x-axis
dt = 0.01;      % time step
tic
for i = 1:100     % exact integration over 100 time steps
  Rexact = Rexact*expm(vec2skew(w*dt));       % update by composition
end
toc  % display the execution time
tic
for i = 1:100     % approximate integration over 100 time steps
  Rapprox = Rapprox + Rapprox*vec2skew(w*dt); % update by addition
end
toc  % display the execution time
det(Rapprox)-1
det(Rexact)-1
rotm2axang(tformnorm(Rexact))
rotm2axang(tformnorm(Rapprox))
```

函数解释：

1) eye
   a) 函数解释：创建单位矩阵
   b) 调用方法：eye(n)，n 是维数
2) tic
   a) 函数解释：启动秒表计时器
   b) 调用方法：直接调用即可
3) toc
   a) 函数解释：从秒表读取已用时间
   b) 调用方法：直接调用即可
4) tformnorm
   a) 函数解释：将 SO(3)或 SE(3)矩阵归一化
   b) 调用方法：tformnorm(T)，T 是 SO(3)或 SE(3)矩阵

3.3.1 Smooth One-Dimensional Trajectories

```
t = linspace(0,1,50); % 0 to 1 in 50 steps
[q,qd,qdd] = quinticpolytraj([0 1],[0 1],t);
clf; stackedplot(t,[q' qd' qdd'])
[q2,qd2,qdd2] = quinticpolytraj([0 1],[0 1],t, ...
  VelocityBoundaryCondition=[10 0]);
mean(qd)/max(qd)
[q,qd,qdd] = trapveltraj([0 1],50);
stackedplot(t,[q' qd' qdd'])
max(qd)
[q2,qd2,qdd2] = trapveltraj([0 1],50,EndTime=1,PeakVelocity=1.2);
[q3,qd3,qdd3] = trapveltraj([0 1],50,EndTime=1,PeakVelocity=2);
```

函数解释：

1) linspace
   a) 函数解释：生成线性间距向量
   b) 调用方法：y = linspace(x1,x2,n)，x1，x2 为区间左右端点，n 为数量
2) quinticcpolytraj
   a) 函数解释：生成五阶轨迹
   b) 调用方法：[q,qd,qdd,pp] = quinticpolytraj(wayPoints,timePoints,tSamples)，wayPoints 为轨迹航点，timePoints 为轨迹航点的时间点，tSamples 为轨迹的时间样本
3) mean

a) 函数解释：返回数组的均值
b) 调用方法：M = mean(A)，A 为数组

4) max
a) 函数解释：返回数组的最大元素
b) 调用方法：M=max(A)，A 为数组

5) trapveltraj
a) 函数解释：生成具有梯形速度曲线的轨迹
b) 调用方法：[q,qd,qdd,tSamples,pp] = trapveltraj(wayPoints,numSamples)，wayPoints 为轨迹航点，numSamples 为轨迹的时间样本的数量

6) stackedplot
a) 函数解释：在堆叠图中绘制表或时间表的变量
b) 调用方法：stackedplot(tbl1,...,tblN)，tb11 等为输入表或时间表

3.3.2 Multi-Axis Trajectories

```
q0 = [0 2]; qf = [1 -1];
q = trapveltraj([q0' qf'],50,EndTime=1);
plot(q')
```

函数解释：
无

3.3.3 Multi-Segment Trajectories

```
cornerPoints = [-1 1; 1 1; 1 -1; -1 -1; -1 1];
R = so2(rotm2d(deg2rad(30)));
via = R.transform(cornerPoints)';
[q,qd,qdd,t] = trapveltraj(via,100,EndTime=5);
plot(q(1,:),q(2,:),"b.-")
plot(q(1,:))
q2 = trapveltraj(via,100,EndTime=5,AccelTime=0.5);
plot(q2(1,:),q2(2,:),"r.-")
plot(q2(1,:))
[q,qd,qdd] = minjerkpolytraj(via,[1 2 3 4 5],500);
plot(q(1,:),q(2,:),".-")
vel_lim = [-1 1; -1 1]; accel_lim= [-2 2; -2 2];
[q,qd,qdd] = contopptraj(via,vel_lim,accel_lim,numsamples=100);
plot(q(1,:),q(2,:),".-")
```

函数解释：

1) so2
a) 函数解释：将输入矩阵转换为 SO(2)矩阵
b) 调用方法：rotation = so2(rotation)，rotation 为旋转矩阵

2) minjerkpolytraj
a) 函数解释：生成通过路点的最小抖动轨迹
b) 调用方法：[q,qd,qdd,qddd,pp,tPoints,tSamples]=minjerkpolytraj (waypoints, timePoints, numSamples)，wayPoints 为轨迹航点，timePoints 为轨迹航点的时间点，numSamples 为轨迹的时间样本的数量

3) contopptraj
a) 函数解释：生成受运动学约束的轨迹
b) 调用方法：[q,qd,qdd,t] = contopptraj(waypoints,vellim,accellim)，wayPoints 为轨迹航点，vellim 弹道的最小和最大速度限制，accellim 轨迹的最小和最大加速度限制

3.3.4 Interpolation of Orientation in 3D

```
rpy0 = [-1 -1 0]; rpy1 = [1 1 0];
rpy = quinticpolytraj([rpy0' rpy1'],[0 1],linspace(0,1,50));
animtform(so3(eul2rotm(rpy')))
q0 = quaternion(eul2rotm(rpy0),"rotmat","point");
q1 = quaternion(eul2rotm(rpy1),"rotmat","point");
q = q0.slerp(q1,linspace(0,1,50));
whos q
animtform(q)
```

函数解释：

1) whos
   a) 函数解释：列出工作区中的变量及大小和类型
   b) 调用方法：直接调用即可
2) rottraj
   a) 函数解释：在方向旋转矩阵之间生成轨迹
   b) 调用方法：[R,omega,alpha] = rottraj(r0,rF,tInterval,tSamples)，r0 是开始的方向，rf 是最终的方向， tInterval 是时间间隔， tSamples 是时间样本
3) so2
   a) 函数解释：
   b) 调用方法：
4) so2
   a) 函数解释：
   b) 调用方法：

3.3.4.1 Direction of Rotation

```
q1 = quaternion(rotmz(-2),"rotmat","point");
q2 = quaternion(rotmz(1),"rotmat","point");
animtform(q1.slerp(q2,linspace(0,1,50)))
q2 = quaternion(rotmz(2),"rotmat","point");
animtform(q1.slerp(q2,linspace(0,1,50)))
```

函数解释：

无

3.3.5 Cartesian Motion in 3D

```
T0 = se3(eul2rotm([1.5 0 0]),[0.4 0.2 0]);
T1 = se3(eul2rotm([-pi/2 0 -pi/2]),[-0.4 -0.2 0.3]);
T0.interp(T1,0.5)
tpts = [0 1]; tvec = linspace(tpts(1),tpts(2),50);
%23A only: Ts = transformtraj(T0,T1,tpts,tvec);
% whos Ts
% animtform(Ts)
% Ts(251)
% P = Ts.trvec();
% size(P)
% plot(P);
% plot(rotm2eul(Ts.rotm()));
% Ts = transformtraj(T0,T1,tpts,trapveltraj(tpts,50));
```

函数解释：

1) se3
   a) 函数解释：将矩阵转换为 SE(3)矩阵
   b) 调用方法：transformation = se3(rotation)，rotation 是旋转矩阵

2) se3/interp
   a) 函数解释：在变换之间进行插值
   b) 调用方法：transformation0 = interp(transformation1,transformation2,points)，transformation 是变换矩阵，points 是插入的点

Experimental Results and Analysis:

3.1.4 Incremental Rotation

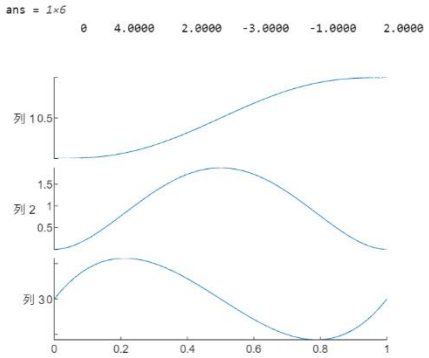
实验结果如图 1 所示，由实验结果可知，验证了近似计算的可行性，并通过多次计算得到近似的计算时间，说明近似算法的效率高。

```
ans = 3×3
    1.0000         0         0
         0    1.0000   -0.0010
         0    0.0010    1.0000

历时 0.020337 秒。


历时 0.002733 秒。
ans = 0.0100
ans = -2.5535e-15
ans = 1×4
     1     0     0     1

ans = 1×4
    1.0000         0         0    1.0000
```

图 1

3.3.1 Smooth One-Dimensional Trajectories

实验结果如图 2、3 所示，由实验结果可知，成功计算时变的参数并且绘制出时变的轨迹。



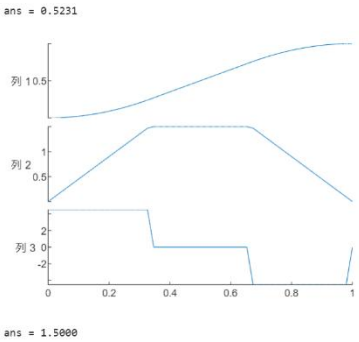


图 2 图 3

3.3.2 Multi-Axis Trajectories

实验结果如图 4 所示，由实验结果可知，成功在同一平面绘制出 x 和 y 轴的数据。

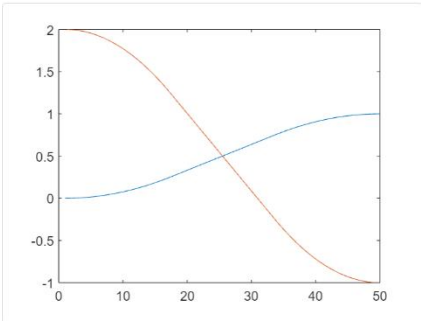


图 4

3.3.3 Multi-Segment Trajectories

实验结果如图 5、6、7 所示，由实验结果可知，成功实现动点沿给定路径行走并实现避障，绘制出路径图，除此，绘制和比较了不同速度的变化情况。

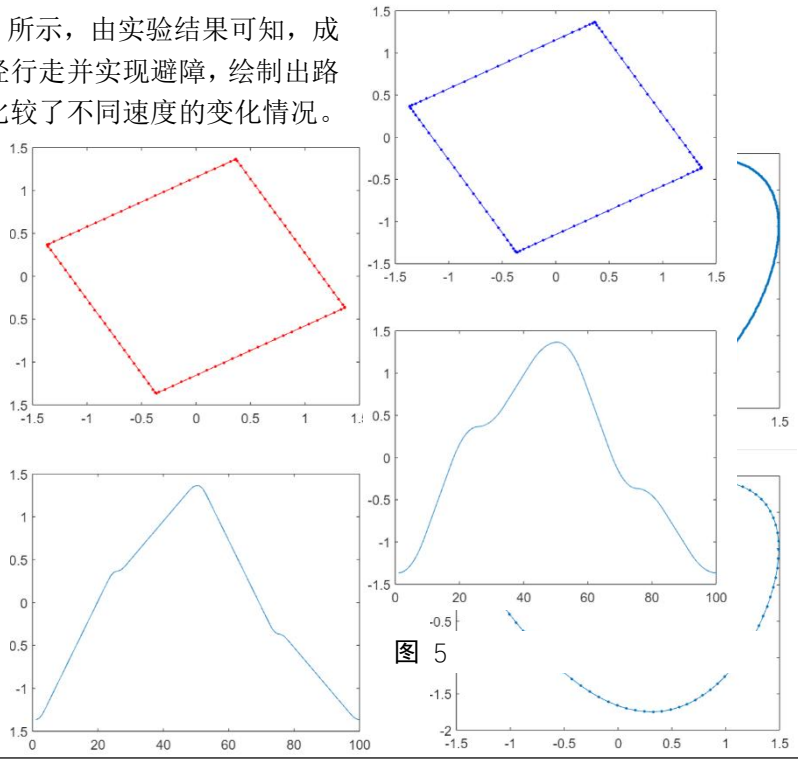


图 5

图 6

图 7

3.3.4 Interpolation of Orientation in 3D

实验结果如图 8、9 所示，由实验结果可知，通过算法成功实现丝滑地更改坐标轴的方向。

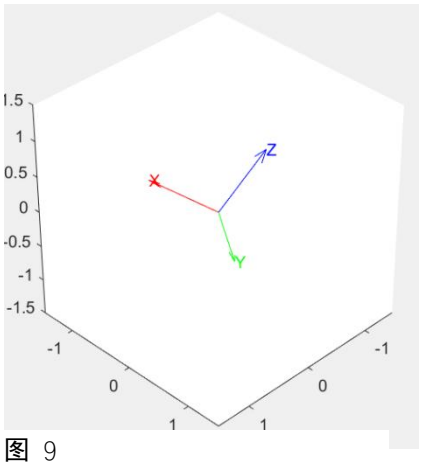


图 9

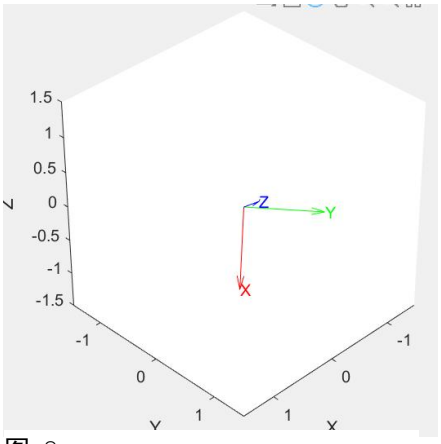


图 8

3.3.4.1 Direction of Rotation

实验结果如图 10、11 所示，由实验结果可知，从同一起点旋转到同一终点，不同的旋转方向会使旋转路径不同和消耗时间不同。

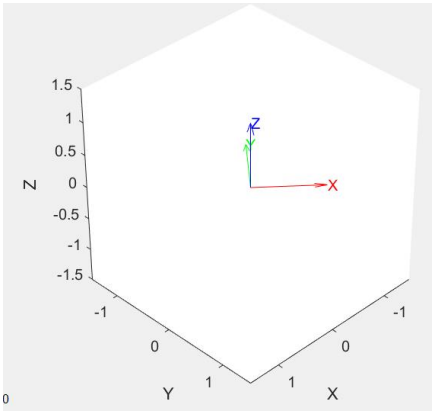


图 11

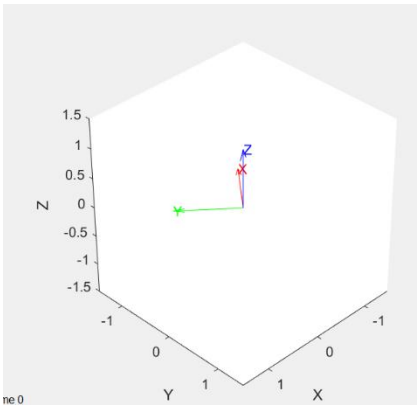


图 10

3.3.5 Cartesian Motion in 3D

实验结果如图 12 所示，由实验结果可知，计算出了变换位姿的最优解。

```
ans = se3
    0.7239   -0.4622    0.5122         0
   -0.0256    0.7239    0.6894         0
   -0.6894   -0.5122    0.5122    0.1500
         0         0         0    1.0000
```

图 12

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。