ZHAO Yuntian - 11811715 - Homework 6

Proof Problems

a)

What above is previous dicussion.

关于$\Delta R = I$的讨论

问题的提出

数值方法解机器人逆运动学问题的时候，当已知一组关节变量的值（记做$\tilde{q}$，是一个向量）时候，需要判断这一组关节变量在经过正运动学映射到末端执行器的位姿（记做$(\tilde{T}(\tilde{R}, \tilde{p}) = fk(\tilde{q}))$），是否与期望的位姿相(记做$T_{Ref}$)重合，如果不，则需要衡量二者之间的距离。

问题由此提出，当两个旋转变换矩阵R_1, R_2不想等的时候，是否可以表示同一种姿态?

需要证明的内容

存在旋转变换矩阵R_1, R_2，满足$\Delta R = R_1 - R_2 = E$，且R_1, R_2表示同一种姿态。

证明结果

所有$R \in SO(3)$，都是旋转变换矩阵，即所有满足$R^T R = E$的3×3的矩阵都是旋转变换矩阵。

则对于任意的$R \in SO(3)$，若满足$(R + E)^T (R + E) = R^T R + R^T + R + E = 2E + R^T + R = E$，则必须满足$R^T + R + E = 0$，但同时满足$R^T R = E$的矩阵$R$不存在（解方程过程略，存在自相矛盾），故这样的矩阵不存在。

其他思考

课上提出的误差计算公式是$err = ||\Delta p||_2 + ||\Delta \theta||_2$，其中$(\Delta\theta, k) = ln\Delta R$，而考虑从当前$\tilde{R}$到$R_{Ref}$的姿态变换，应有$\Delta R = R_{ref}\tilde{R}^{-1}$，这里$\Delta R$是两个矩阵的“商”。那么需要证明的$\Delta R =$

$OP(R_1, R_2)$是否也是“商”呢？如果是的话，则无需证明所有旋转矩阵都满足这一点；如果不是的话，是否这里err的计算存在某些问题?

后记

个人理解，可能有疏漏，希望老师指出或给予此问题更详细的解释；同时欢迎其他同学一起探讨。

After the clarification of $R_0 \Delta R = R_{Ref}$, by what above, we will certainly have that **if and ony if** $R_0 = R_{Ref}$ **we have the same orientation, so we must have** $\Delta R = I$**.**

b)

To pove $R_k^T(q) = R_k(-q)$ is to prove $R_k^T(-q) = R_k(q)$. We will use Rodrigues' formula to prove:
$LHS = R_k^T(-q) = [e^{\hat{k} \cdot (-q)}]^T = [E + \hat{k}sin(-q) + (\hat{a})^2(1 - cos(-q))]^T = E + (\hat{-k}) \cdot (-sin(q)) + [(\hat{a})^2]^T(1 - cos(q)) = E + \hat{k}sin(q) + (\hat{a})^2(1 - cos(q)) = R_k(q) = RHS$ □

Code

Pseduo Code in a 'stack' form will be preformed to illustrate.

`SetupBipedRobot.m`

pseudo code

```
Define Rad-Deg convertion and axis direction
Define uLINK, include name, mass, sister, child, vector b,a and angle q
        Call [MakeBox]
        % MakeBox generate 8 vertexes, each expressed 'local' frame and 6 faces, each expressed wi
        % input is dimension of x,y,z and coordinate of origin
        % output is vertexes and faces
        return
        Call [FindMother]
        % FindMother is recursively 'assign' mother to sister and child nodes
        % input the root of the tree
        % output nothing
        if input is NULL:return
        if input is ROOT:input->mother=NULL
        if input has child:
                input->child->mother=input
                call [FindMother],input=input->child
        if input has sister:
                input->sister->mother=input->mother
                call [FindMother],input=input->sister
        return
Label All LINK with number for convenience
Set BODY LINK's pose
        Call [ForwardKinematics]
        % ForwardKinematics is also a recursive function
        % input is the root of the tree
        % output is null, but update pose of all link via fk
        if input is NULL:return
        if input is not ROOT:
                relative position of this link = R * b + p
                Call [Rodrigues]
                % Rodrigues is like an inline function, input is axis and angle, output is rotatio
                Using Rodrigues formula to calculate
                return
                relative oriantation of this link = R_mother * R_self calculated by Rodrigues
        call [FordwardKinematics],input=input->sister
        call [FordwardKinematics],input=input->child
Define linear and angular velocity of BODY in world frame
Set all joint velocity to zero
```

scripts involved

```
MakeBox
FindMother
ForwardKinematics
        Rodrigues
```

IK_leg.m

pseudo code

```
% this part is the MATLAB implementation of what is in Lecture Kinematics IV Page13-17
% IK_leg is a function whose input and output are:
% input uLINK part BODY, FOOT and distance of joints D, A and B
% output is the joint angle vector q
calculate vector r from p7 to p2 in world frame
calculate cos q5 using cosine rule with A,B and length of r
calculate q5 by in different cases of cos q5
calculate part of q6 denoted as alpha, using sine rule, with A,D,length of r and cos q5
calculate q7 with r transformed in FOOT's frame
calculate another part fo q6 with r transformed in FOOT's Frame
calculate q6 with its teo parts
calculate sphere HIP's three joint angles by matching up two matraces
```

$$R_z(q_2)R_x(q_3)R_y(q_4) = R_1^T R_7 R_x(-q_7)R_y(-q_5 - q_6)$$

```
return vector q of q2 q3 q4 q5 q6 q7
```

scripts involved

```
none
```

InverseKinematics.m

pseudo code

```
% This is the implementation of the numerical method for inverse kinematics
% input is the target link and its pose
% output is numeric error
get links between target link and root by
        call [FindRoute]
        % FindRoute is recursive get the mother of a link and add it to a list, then return it
        % input is target link
        % output is a ordered list of link id from target to root
        if input is ROOT:
                return list{1}
        else:
                return list{input [FindRoute], input=input->mother}
        call [FordwardKinematics]
        % This has been detailed analysed in [SetupBipedRobot.m]
calculate error now
        Call [CalcVWerr]
        calculate position error just by misusing one another
        calculate theta error using Rotation matrix error
                call [rot2omega]
                % This is the logarithm of matrix
                return angular velocity vector
        return the error combining position error and angle error
% The program using for loop, but actually do-while loop is more suitable
for counter not reach maximum ierations:
        if error is less than thershold:
                break
        % use Newton method to minimum error, Jacobian is second order derivative
                call [CalcJacobian]
                % This is the calculation of Jacobean matrix, but may be a different from the most
                % The most simple ways is to do multi variable differential, and maybe another way
                % But this seems like using the so called screw theory. Each column of the Jacobea
                return the Jacobean of now pose
        get revise dq = step_length * Jacobean / error
        % This numerical algorithm update the position of the end-of-effector in time
                call [MoveJoints]
                % This function transversely update the joints of all LINKs
                for all link in idx:
                        update joint angle
                return
                call [ForwardKinematics]
                % This has been detailed discussed in [SetupBipedRobot.m]
                return the new end of effect pose
                call [CalcVWerr]
                % this has been detailed discussed in [InverseKinematics.m]
                return new error
% Then is some output match (The nargout part)
```

scripts involved

```
FindRoute
ForwardKinematics
        Rodrigues
CalcVWerr
        rot2omega
        CalcJacobian
        MoveJoints
```

fk_random.m

pseudo code

```
% This script set up a biped robot and then random set joint angles, then perform forward kinemati
% The program loop forever to generate robot configure until user define signal SIGINT is sent.
Clear workspace
Define uLINK
Call [SetupBipedRobot.m] to create a robot
Set random seed
Create figure
While true:
        random left and right joint angles
        assign angles to robot
        set BODY pose
                call [ForwardKinematics],input is root
                % This function has been detailed discussed in [SetupBipedRobot.m]
                return
        clean figure
                call [DrawALLJoints]
                % This function recursively draw all LINK of the robot
                % input is root of the tree
                % output is nothing
                if input is not null:
                        if input has word field vertex:
                                convert its vertex to the world frame
                                        call [DrawPolygon]:
                                        % This function uses patch to draw polygon with input para
                                        % input is vertexes and faces, the same word field as in [
                                        draw polygon
                        ****            return
                        if input->mother is not null:
                                        call [Connect3D]
                                        % This function just connect input points
                                        % input is mother link and child link
                                        % output is nothing
                                        return
                                call [DrawCylinder]
                                % This function just draw cylinder with input parameters
                                % inputs are a reference point position, z axis direction , radius
                                % no output
                                draw cylinder with input parameters
                                return
                                call [DrawAllJoints], input=input->sister
                                call [DrawAllJoints], input=input->child
                return
        change view and axises of the plot, set z limits and open grid
        print abort message
        pause
```

scripts involved

```
SetupBipedRobot
        MakeBox
        FindMother
        ForwardKinematics
                Rodrigues
DrawAllJoints
        DrawPolygon
        Connect3D
        DrawCylinder
```

ik_random.m

pseudo cpde

```
% This script uses analytic inverse kinematics to caluclate a random given end-of-effector pose an
clear and set up robot
        call [SetupBipedRobot]
        return
set robot joints to non-singular angle
set random seed
open figure
% loop forever until SIGINT signal
while true:
        set BODY pose
        set left foot and right foot to random pose
                call [IK_leg]
                % IK_leg is analytic inverse kinematics of robot leg
                % It has been detialed discussed in [IK_Leg]
                return joint angles of left and right legs
        for all joints:
                set joint angle as what inverse kinematics calculated
                call [ForwardKinematics]
                % This has been detailed discussed in [SetupBipedRobot]
                return
        clean figure
                call [DrawAllJoints]
                % This has been detailed discussed in [fk_random.m]
                return
        change view, set axis and limits, open grid
        print abort message
        pause
```

scripts involved

```
SetupBipedRobot
        MakeBox
        FindMother
        ForwardKinematics
                Rodrigues
IK_Leg
DrawAllJoints
        DrawPolygon
        Connect3D
        DrawCylinder
```

ik_random2.m

pseudo code

```
% This script uses numerical inverse kinematics to caluclate a random given end-of-effector pose a
clear and set up robot
        call [SetupBipedRobot]
        return
set robot joints to non-singular angle
set random seed
open figure
% loop forever until SIGINT signal
while true:
        set BODY pose
        set left foot and right foot to random pose
                call [InverseKinematics]
                % InverseKinematics is numerical inverse kinematics of robot end-of-effector
                % It has been detialed discussed in [InverseKinematics]
                return joint angles of left and right legs and errors
        for all joints:
                set joint angle as what inverse kinematics calculated
                call [ForwardKinematics]
                % This has been detailed discussed in [SetupBipedRobot]
                return
        clean figure
                call [DrawAllJoints]
                % This has been detailed discussed in [fk_random.m]
                return
        change view, set axis and limits, open grid
        print numerically inverse errors
        print abort message
        pause
```

scripts involved

```
SetupBipedRobot
        MakeBox
        FindMother
        ForwardKinematics
                Rodrigues
InverseKinematics
        FindRoute
        CalcVWerr
                rot2omega
                CalcJacobian
                MoveJoints
DrawAllJoints
        DrawPolygon
        Connect3D
        DrawCylinder
```