Lab: The Robot Motion Kernel

Overview

In this lab, you will not just *use* a robot controller; you will **build** one. You are provided with a simulation environment, but the robot's ability to move (`MoveJ` and `MoveL`) has been stripped away. Your task is to implement the mathematical logic for these motions and verify them, first in a test environment, and then in a matlab version of an industrial RAPID program.

Phase 1: The Motion Logic (Warm-up)

Objective: Implement the core interpolation algorithms in a simplified environment.

1. The Setup

Create a new MATLAB script named `Lab_Simple_Motion.m` and copy the code below. This script sets up a robot and generates random start/target positions.

> **Note:** The script calls `MoveJ` and `MoveL` , but these functions are empty. The script will fail or do nothing until you write the logic.

```
close all
clear
clc
robot = importrobot('test.urdf');
showdetails(robot);
randConfig = robot.randomConfiguration;
tform = getTransform(robot,randConfig,"link6");

show(robot,randConfig);
%%
target = randomConfiguration(robot);
home=homeConfiguration(robot);

MoveJ(home, target, robot)
MoveL(home, target, robot)
```

2. Your Tasks

1. **Implement** `MoveJ` **:**
   - Write a loop that calculates intermediate joint angles.
   - Use linear interpolation: $q_{curr} = q_{start} \times (1 - t) + q_{end} \times t$.
   - Visualize the motion using `show(robot, config)` .
2. **Implement** `MoveL` **:**

- Calculate the **Start Matrix** and **Target Matrix** using `getTransform`.
- Interpolate the X, Y, Z coordinates linearly.
- For each step, calculate the required transformation matrix and use `inverseKinematics` to find the joint angles.
  - https://www.google.com/search?q=https://se.mathworks.com/help/releases/R2025a/robotics/ref/inversekinematics-system-object.html
- **Challenge:** How do you handle rotation? (Simplest: Use Quaternion Nlerp. Advanced: Use Quaternion Slerp).

Phase 2: The RAPID code (Integration)

Objective: Apply your motion kernel to execute a real RAPID program structure.

1. The Scenario

You have received a snippet of RAPID code defining a path for a robot (p10 $\rightarrow$ p20 $\rightarrow$ ... $\rightarrow$ p60). We have parsed this code into a MATLAB simulation for you.

2. The Setup

Create a new script `Lab_Simulation.m` and copy the **Starter Kit** below.

```
close all
clear
clc

% MODULE MainModule
%       TASK PERS tooldata t4:=[TRUE,[[-105.513,2.40649,246.356],[1,0,0,0]],[0.5,[50,0,
%       TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[559.804,5.50957,-3.63248],[0.999987,
%       CONST robtarget p10:=[[-46.86,-7.90,235.64],[0.0498083,-0.0133606,-0.998594,-0.
%       CONST robtarget p20:=[[-21.39,-34.91,-0.63],[0.0497861,-0.0134405,-0.998589,-0.
%       CONST robtarget p30:=[[19.16,-34.92,-0.53],[0.0498128,-0.0133747,-0.998593,-0.0
%       CONST robtarget p40:=[[19.17,35.63,-0.49],[0.0498143,-0.0133815,-0.998593,-0.01
%       CONST robtarget p50:=[[-21.65,35.64,-0.22],[0.0498108,-0.0133915,-0.998593,-0.0
%       CONST robtarget p60:=[[-21.65,35.64,-0.22],[0.0498136,-0.0133915,-0.998593,-0.0
%       VAR num xpos:=0;
%       VAR num ypos:=0;
%       VAR num zrot:=0;
%     VAR robtarget current_pos;
%     VAR intnum timer;
%     VAR num x;
%     VAR num y;
%     VAR num z;
%     VAR iodev logfile;
%       PROC main()
%         CONNECT timer WITH Read;
%         Open "HOME:" \File:= "LOGFILE2.DOC", logfile \Write;
%         ITimer 0.5, timer;
%               draw;
%               TPReadNum xpos, "Enter the value of x ";
```

```
%               TPReadNum ypos, "Enter the value of y ";
%               TPReadNum zrot, "Enter the angle of z ";
%               wobj1.oframe.trans.x := xpos;
%               wobj1.oframe.trans.y := ypos;
%               wobj1.oframe.rot := OrientZYX(zrot,0,0);
%               draw;
%       ENDPROC
%       PROC draw()
%               MoveJ p10, v200, fine, t4\WObj:=wobj1;
%               MoveL p20, v200, z10, t4\WObj:=wobj1;
%               MoveL p30, v200, z10, t4\WObj:=wobj1;
%               MoveL p40, v200, z10, t4\WObj:=wobj1;
%               MoveL p50, v200, z10, t4\WObj:=wobj1;
%               MoveL p20, v200, fine, t4\WObj:=wobj1;
%       ENDPROC
%       TRAP Read
%               current_pos := CRobT();
%         x:=current_pos.trans.x;
%         y:=current_pos.trans.y;
%         z:=current_pos.trans.z;
%         Write logfile,"X;"\num:=x;
%         Write logfile,"Y;"\num:=y;
%         Write logfile,"Z;"\num:=z;
%       ENDTRAP
% ENDMODULE


robot = importrobot('abbIrb1600.urdf');
config = robot.randomConfiguration;
tform = getTransform(robot,config,"tool0");
robot = addFrame([-105.513,2.40649,246.356],[1,0,0,0],robot,'t4','t4j','tool0');
robot = addFrame([559.804,5.50957,-3.63248],[0.999987,-0.00156359,-0.00487101,7.47128E-
robot = addFrame([5,4,0],[0.67559,0,0,-0.737277],robot,'oframe','oframej','uframe');

robot = addFrame([-46.86,-7.90,235.64],[0.0498083,-0.0133606,-0.998594,-0.0123139],robo
robot = addFrame([-21.39,-34.91,-0.63],[0.0497861,-0.0134405,-0.998589,-0.0127018],robo
robot = addFrame([19.16,-34.92,-0.53],[0.0498128,-0.0133747,-0.998593,-0.0123192],robot
robot = addFrame([19.17,35.63,-0.49],[0.0498143,-0.0133815,-0.998593,-0.0123221],robot,
robot = addFrame([-21.65,35.64,-0.22],[0.0498108,-0.0133915,-0.998593,-0.0123259],robot
robot = addFrame([-21.65,35.64,-0.22],[0.0498136,-0.0133915,-0.998593,-0.0123257],robot

show(robot,config,Visuals="on");

%%


MoveL(getTransform(robot,robot.homeConfiguration,"tool0"),getTransform(robot,config,"p1
MoveL(getTransform(robot,config,"p10"),getTransform(robot,config,"p20"),robot,'t4');
MoveL(getTransform(robot,config,"p20"),getTransform(robot,config,"p30"),robot,'t4');
MoveL(getTransform(robot,config,"p30"),getTransform(robot,config,"p40"),robot,'t4');
MoveL(getTransform(robot,config,"p40"),getTransform(robot,config,"p50"),robot,'t4');
MoveL(getTransform(robot,config,"p50"),getTransform(robot,config,"p60"),robot,'t4');
MoveL(getTransform(robot,config,"p60"),getTransform(robot,config,"p20"),robot,'t4');
MoveL(getTransform(robot,config,"p20"),getTransform(robot,config,"p10"),robot,'t4');
```

```
function robot = addFrame(Trans,q,robot,name,jointname,parentname)
    R=quat2rotMatrix(q)
    T = [[R;[0 0 0]],[(Trans./1000)';1]]
    frame = rigidBody(name);
    jnt1 = rigidBodyJoint(jointname,'fixed');
    setFixedTransform(jnt1,T);
    frame.Joint = jnt1;
    addBody(robot,frame,parentname)
end
```

3. Your Tasks

1. **Migrate Logic:** Copy your working `MoveJ` and `MoveL` functions from Phase 1 into `Lab_Simulation.m`.
2. **Adapt for Named Targets:**
   - In Phase 1, the target was a *Transformation matrix*.
   - In Phase 2, the targets are *Frame names* (e.g., `'p10'`, `'p20'`).
   - **Update your code:** You must modify `MoveL` (and `MoveJ`) to accept a string name (like `'p10'`) and use `getTransform(robot, homeConfig, 'p10')` to find the Cartesian target matrix.
3. **Use your own RAPID code**

Deliverables

Submit a single report containing:

1. **The "Kernel":** Your final implementation code for `MoveJ` and `MoveL`.
2. **Meeting notes and responsibilities.**
3. **The Presentation:** You will need to present your work either to the teacher or as a recording.