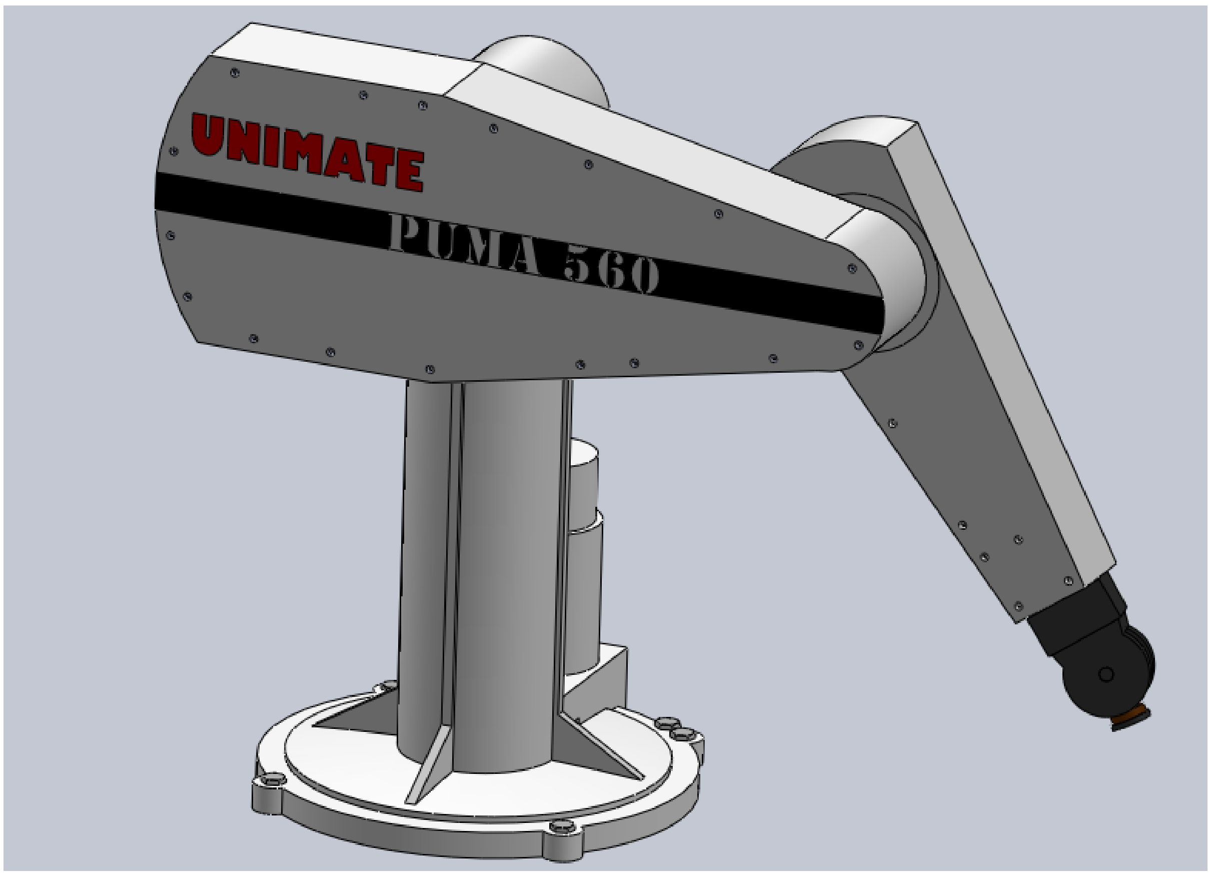
Puma 560 Test 1
Robotics and Automation 282 762

Marc Alexander Sferrazza
12164165 [*][†]

May 29, 2017



Abstract

A brief report of the Puma 560 trajectory demonstration and plots using Peter Corke's Toolbox

[*]This work was not supported by any organization

[†]Faculty of Mechatronics Engineering, Massey University, Albany, Auckland, New Zealand `Progress of project: https://github.com/alex1v1a/Robotics-and-Automation/`

Contents

List of Figures

1 Part A: Code Snippet

Please find Part A in the local directory saved as ”Initials.m” note that this matlab file contains all detailed comments for reference from the lecture slides and Peter Corke’s Toolbox functions.

The snippet contains the described below and a the full version has also been included in the appendix of this document.

2 Part B: Trajectory in 3 Dimensions

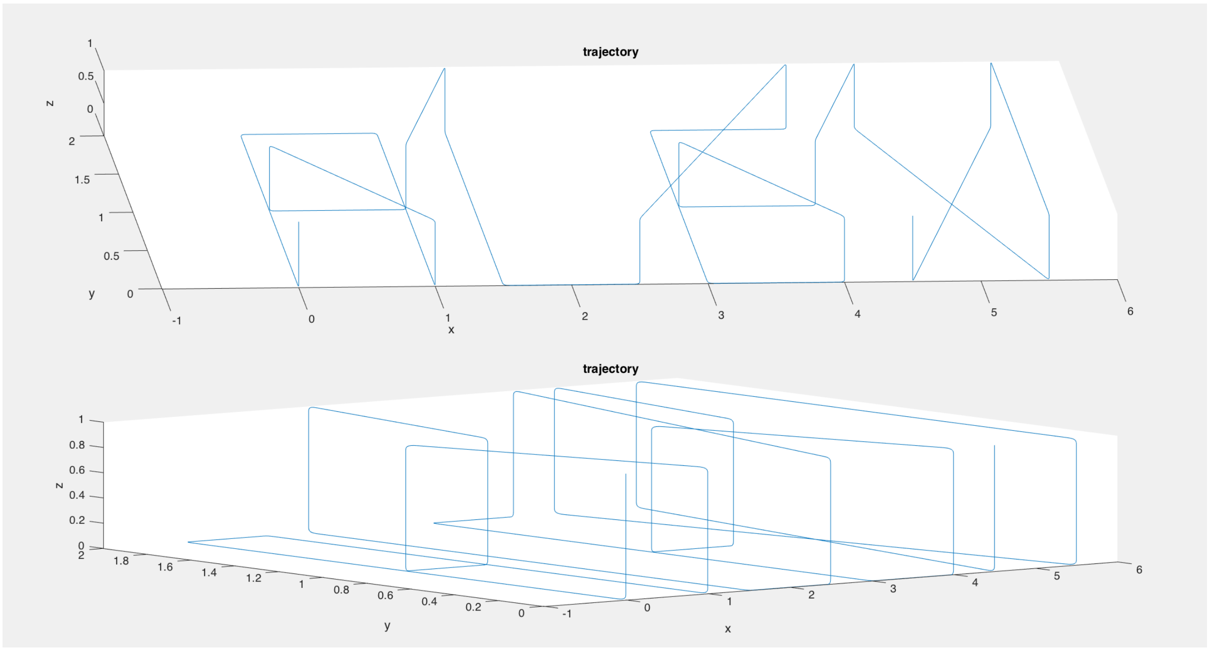


Figure 1: A plot of the end effector name trajectory in 3 Dimensions (X,Y,Z)

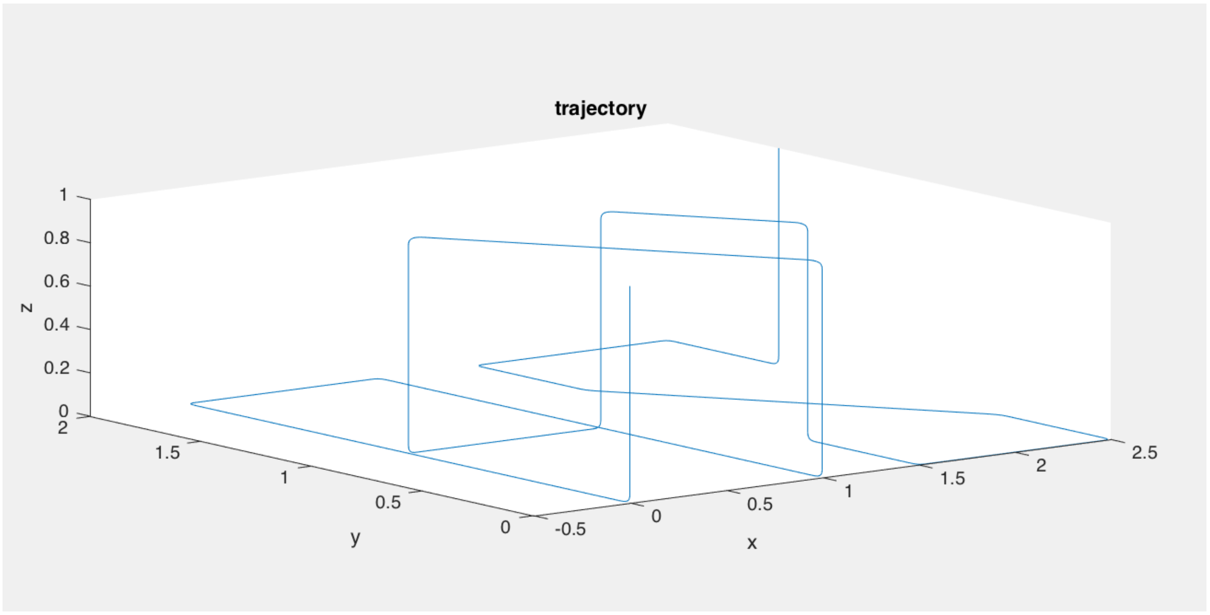


Figure 2: A plot of the end effector initials trajectory in 3 Dimensions (X,Y,Z)

3 Part C: Joint Angles with respect to Time

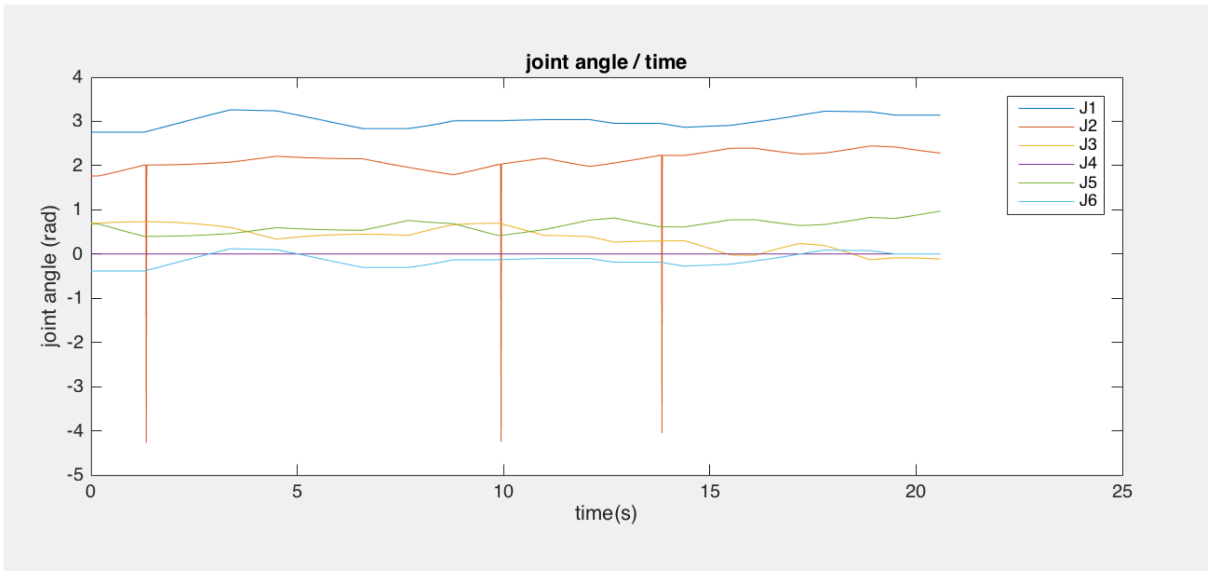


Figure 3: A plot showing each of the joint angles with respect to time

4 Part D: XYZRPY Magnitudes with respect to Time

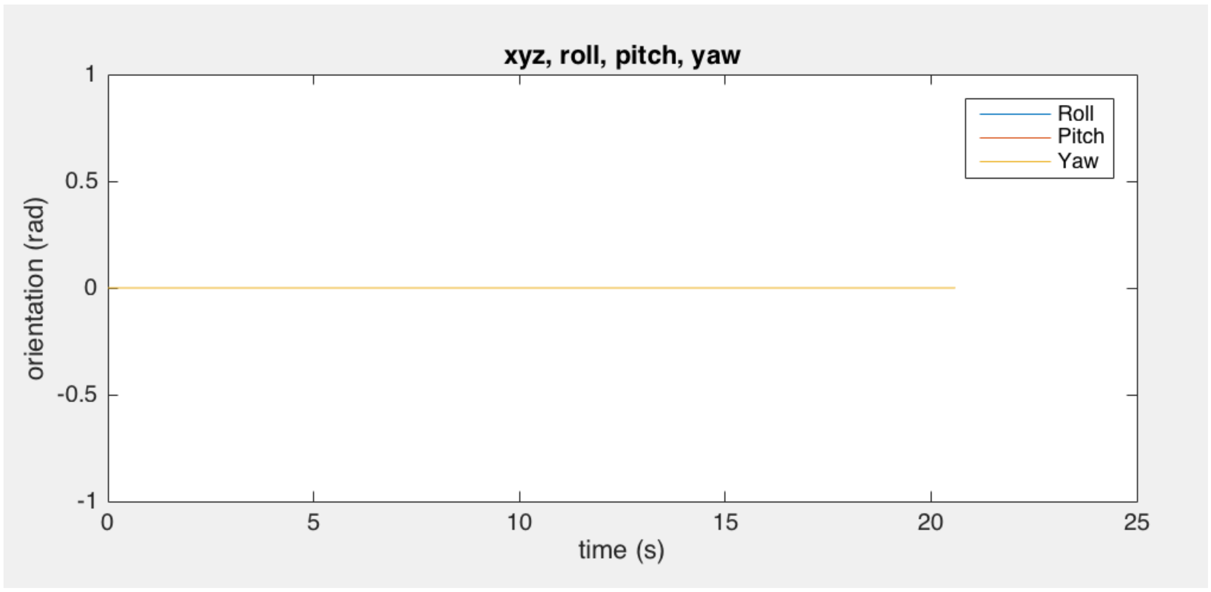


Figure 4: A plot showing XYZRPY magnitudes with respect to time

5 Part E: Trajectory Generation Technique

The implementation of the mstraj (Multi-segment multi-axis trajectory) function has been used when creating the path of the trajectory as it has both the properties of the cartesian and joint space method trajectories. Better results are produced when creating curved and straight lines on the x, y and z axis and when approximating fonts there is a more steady range available, for instance the flat head 'A' could be written as a sharpe head or a curved head while maintaining integrity of the path with direct translation instead of pose motion.

From the information shown on the plots above there are sudden changes in joint angles at 1.5, 10 and 13.5 (rad) which may cause issues with other joints e.g. singularities. In the orientation it is observed on the graph as a straight line, this means that while the joint is moving in the X, Y, and Z axis the roll, pitch, and yaw remains fixed.

6 Part F: Trajectory Outcomes

The mstraj function Produced a good result, and the characters "ALEX" and "AS" have turned out clear and precise. There is room for improvement as producing scaled and more elegant fonts are optional to an extent based on the constraints of the joint angles.

When demonstrating this method on a non-virtual machine, it is necessary to take the trajectory limitations from joint angles into account for things such as singularities and overlapping. If there are no other interferences and disturbances in the trajectory the live demo may be accomplished successfully.

7 Video Demo

Please find A video demonstration of the Puma 560 trajectory for "AS" in the local directory saved as "Demo.avi"

References

$http://petercorke.com/Robotics_Toolbox.html$
Stevens Lecture Notes Part B (Peter Corke - Chap 7 - Second Half)
Stevens Lecture Notes Part A (Peter Corke - Chap 7 - First Half)
Stevens Lecture Notes Part C (Peter Corke - Chap 8)
Stevens Lecture Notes Part D (Peter Corke - Chap 9 - First Half)

APPENDIX

```
% Load in trejectory

% Initials AS
path = [0 0 1;0 0 0;0 2 0;1 2 0;1 0 0;1 0 1;0 1 1;0 1 0;1 1 0;1 1 1;1.5 0.5
     1;1.5 0.5 0;1.5 0 0;2.5 0 0;2.5 0.5 0;1.5 1.5 0;1.5 2 0;2.5 2 0;2.5 1.5
     0;2.5 1.5 1;]

% Name ALEX
% path = [0 0 1;0 0 0;0 2 0;1 2 0;1 0 0;1 0 1;0 1 1;0 1 0;1 1 0;1 1 1;1.5 2
     1;1.5 2 0;1.5 0 0;2.5 0 0; 2.5 0 1;4 2 1;4 2 0;3 2 0;3 0 0;4 0 0;4 0
    1;3 1 1;3 1 0;4 1 0;4 1 1;4.5 2 1;4.5 2 0;5.5 0 0;5.5 0 1;5.5 2 1;5.5 2
    0;4.5 0 0;4.5 0 1;]

% TRAJ = MSTRAJ(P, QDMAX, TSEG, Q0, DT, TACC, OPTIONS)
% trajectory = mstraj(path,[speedx speedy speedz],[],[initx inity initz],
    sampleperiod,acceltime)
point = mstraj(path,[.5 .5 .5],[],[0 0 1],0.02,0.2) % Create a trajectory
    of points

% - P (MxN) is a matrix of via points, 1 row per via point, one column
%   per axis. The last via point is the destination.
% - QDMAX (1xN) are axis speed limits which cannot be exceeded,
% - TSEG (1xM) are the durations for each of the K segments
% - Q0 (1xN) are the initial axis coordinates
% - DT is the time step
% - TACC (1x1) this acceleration time is applied to all segment transitions
% - TACC (1xM) acceleration time for each segment, TACC(i) is the
    acceleration
%   time for the transition from segment i to segment i+1. TACC(1) is also
%   the acceleration time at the start of segment 1.

% Plot 3D trajectory
% Part B: A plot of the end effector trajectory in 3 Dimensions (X,Y,Z)
subplot(3,1,1), plot3(point(:,1),point(:,2),point(:,3))
xlabel('x'); ylabel('y'); zlabel('z');title('trajectory')

% Cartesian Space Method
traj = transl(0.1*point) % Translation of 0.1 Pose
Transform = homtrans(transl(0.4,0,0),traj) % HomoTrans Pose (0.4,0,0) (
    pose1,pose2,time) of the trajectory

% T = transl(0.1*p) % Scale to 10% size
% T_Final = homtrans(transl(0.4,0,0),T)

% Using Peter Corke's␣robotics␣toolbox,␣as␣the␣avaliable␣mdl_puma560␣is␣
    already
%␣done␣for␣us␣/␣traj␣=␣ctraj(pose1,pose2,#steps)
mdl_puma560
p560.tool␣=␣trotx(pi)
angle␣=␣p560.ikine6s(Transform)␣%␣50␣x␣poses␣consisting␣of␣6␣joint␣angles,␣
    inverse␣kinematic␣of␣the␣transform
time␣=␣[0:0.01:20.59]' % 2 Seconds, 50 ms period
% The      causes the vector to be transposed

% total_time = timesteps * step time
% total_time = numrows(p)*0.02
% Total Time = 32 points * 0.02 = 0.64

% p560.tool = trotx(pi)
% q = p560.ikine6s(T_Final)
% p560.plot(q)

% Part C: A plot showing each of the joint angles with respect to time
subplot(3,1,2), plot(time,angle)
xlabel('time(s)'); ylabel('joint␣angle␣(rad)');title('joint␣angle␣/␣time')
legend('J1','J2','J3','J4','J5','J6')

% Part D: A plot showing XYZRPY magnitudes with respect to time.
subplot(3,1,3), plot(time,tr2rpy(Transform)) % Orientation component
xlabel('time␣(s)'); ylabel('orientation␣(rad)');title('xyz,␣roll,␣pitch,␣
   yaw')
legend('Roll','Pitch','Yaw')

% A video file that shows the robot simulation
fig = figure('visible','on');

fps = 100;
n_samples = 5 * fps;

mov.frames = getframe(fig);

filename = 'Demo.avi';
% writerObj.CompressionRatio = 3;
writerObj = VideoWriter('Demo.avi');
open(writerObj);
for i = 1:2060, % 50*0.02 * secconds (0.64)
    p560.plot(gait(angle,i,0,0))
    drawnow
    frames(i) = getframe(fig);
    writeVideo(writerObj,frames)
end
close(writerObj);
```