Simulation on 2-DOF Waterjet with PUMA560 Robot
Processed by MATLAB Robotics Toolbox

Xiaoxuan SHAO

Southeast University

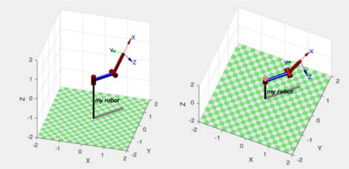
January 24, 2021

Table of Contents

Introduction to Robotics Toolbox
Helpful Tool for simulation

As an important toolbox of MATLAB, Robotics Toolbox is tactful to handle problems such as build robot model, trajectory planning, joint torque control. To create a robot in toolbox is like:

```
Robotics Toolbox          version 10.1.0

startup rvc

L1 = Link('d',0,'a',1,'alpha',pi/2)

L2 = Link('d',0,'a',1,'alpha',0)

bot = SerialLink([L1,L2], 'name', 'my robot

bot.fkine([0.1,0.2])

bot.plot([pi/4,pi/3])
```



[1]https://petercorke.com/toolboxes/robotics-toolbox/#Downloading_the_Toolbox

Figure: A very simple example

As an important industrial robot, PUMA560 is already set up in Robotics Toolbox. We can build a PUMA560 by "mdl_puma560"

By applying consoled functions, we can

- Solve forward kinematics by "fkine"
- Solve inverse kinematics by "ikine"
- Output simulation images by "robot.plot(q)"

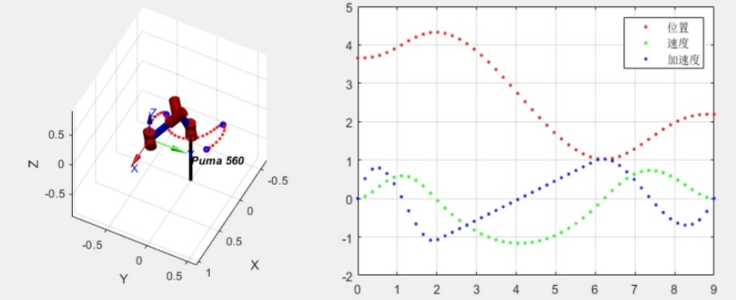
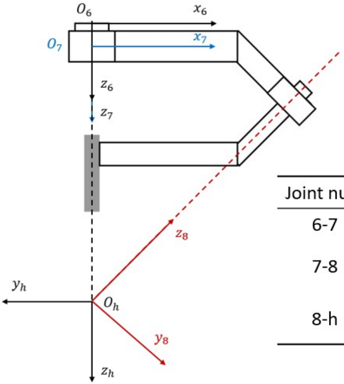


Figure: trajectory planning and control

Simulate a robot that travels at the contour of a specific image.

- Based on image recognition
- Obtain the required cutting figure contour
- Perform path planning to determine the pose of the end joint point



Joint num	a_i	d_i	α_i	θ_i
6-7	0	0.01	0	0
7-8	0	0.13	$\frac{3\pi}{4}$	$\frac{\pi}{2} + \beta$
8-h	0	0	$-\frac{3\pi}{4}$	θ

Figure: waterjet model

Calculate the tangent line of the position of discrete points:
z-axis is always perpendicular to the cutting plane
x-axis is always tangent to the cutting curve

Advantage:

- When the cutting material changes, do not need to change the posture or inverse kenematics
- Arm in charge of curve, waterjet in charge of quality

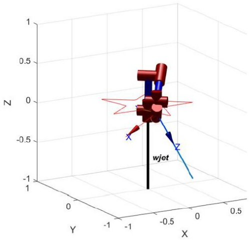


Table of Contents

To solve the problem, I divided the codes into following parts:

1) Build a PUMA 560
2) Initialize all parameters(for visualization usage)
3) Import image and obtain contour
4) Trajectory planning and output
5) Postprocessing and visualization

And following I will introduce part 3), 4) in detail.

How Do I Achieve My Goal

Import image and obtain contour

```
center = [0 0 0];
II = imread('star.jpg');
BW = im2bw(II);
B = bwboundaries(BW);
imshow(II);
hold on;
targetBoundary = B{2};
plot(targetBoundary(:, 2), targetBoundary(:, 1), 'm', 'LineWidth', 2);
points = ([targetBoundary(:, 2), targetBoundary(:, 1)])';
height = linspace(center(3),center(3),length(points));
points = [points;height];
imgsize=size(II);
points(1,:)=(points(1,:)/imgsize(1)-0.5)+center(1);%长
points(2,:)=(points(2,:)/imgsize(2)-0.5)+center(2);%宽
```

- Read image "star"
- Get image boundary
- Plot contour to screen

How Do I Achieve My Goal

Trajectory planning and output

```
figure
leap = fix(size(points,2)/PointCount) %取点个数
for i = 1:leap:size(points,2)
    pause(0.01)
    bx = points(1,i);
    by = points(2,i);
    bz = points(3,i);
    targetPos = [bx by bz];
    TR=transl(targetPos);
    if i-leap>1 %后序点用切线做角度旋转
        dx = points(1,i) - points(1,i-leap);
        dy = points(2,i) - points(2,i-leap);
        dz = points(3,i) - points(3,i-leap);
        TR = TR*trotz(-atan2(dx,dy));
    end
    hold on
    grid on
    plot3(bx,by,bz,'r.','linewidth',10);    %%红色点点
    q=p560.ikine6s(TR)
    if isempty(q)
        warning("point not reachable");
    else
        p560.plot(q);
    end
```

- Grasp contour points
- Compute rotate angle
- Inverse kinematics
- Output on screen

Table of Contents

Simulation on Different Shapes

A Cross Shape Picture

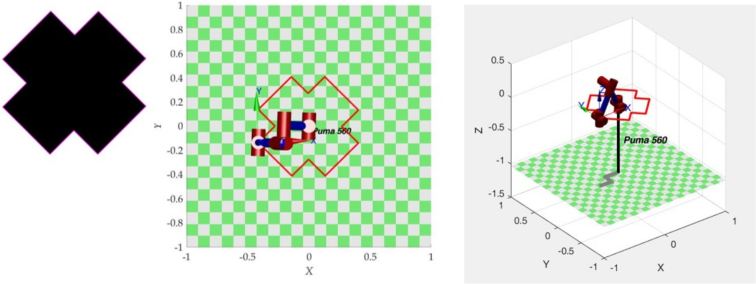
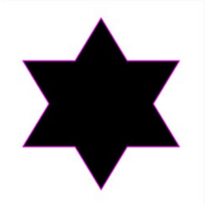


Figure: To simulate a cross



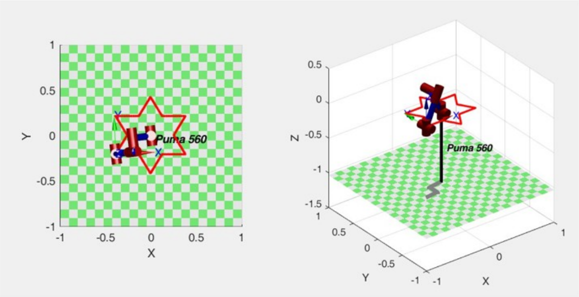


FIGURE: To simulate a star

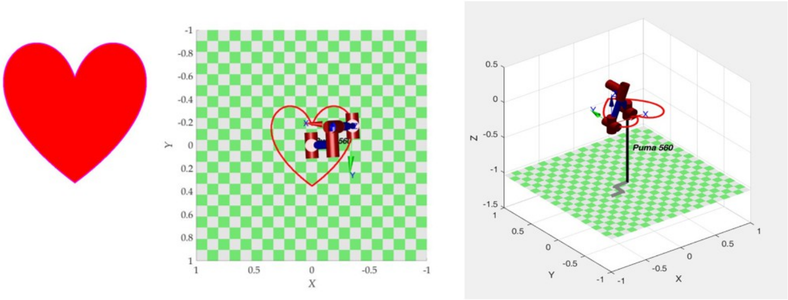


Figure: To simulate a heart

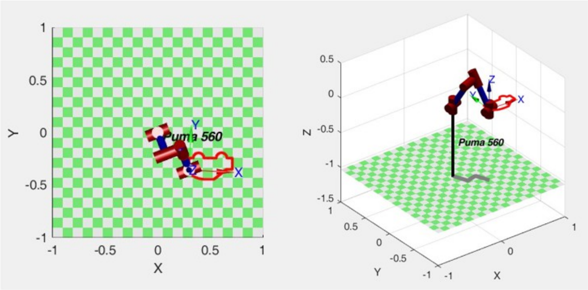


FIGURE: To simulate a car

CROSS

STAR

HEART

CAR

Thanks for Listening

Xiaoxuan SHAO

SyrinxSHAW@outlook.com