

pybullet Inverse Kinematics Writing Letters

Di AI – Mines ParisTech - AIMove 2020

Introduction

In this mini project, I use pybullet simulator to control a robot in order to write Latin letters (A - Z). The control method is inverse kinematics.

Methods

1. Letter trajectory regression

The letter trajectories of robot are not pre-fixed, but are generated by learning (regression) from people's hand writing trajectories (9 hand writing samples of each letter).

There are 3 regression methods to generate letter trajectories:

1. Locally weighted regression (LWR)
2. Gaussian Mixture Regression (GMR)
3. Gaussian Mixture Regression (GMR) with Dynamical movement primitives (DMP)

Thus each execution of the script will generate slightly different letter trajectories for robot.

Please see the source code as well as detailed comments in github repository:

https://github.com/eddieai/pybullet_inverse_kinematics_write_letter/blob/master/trajectory.py

2. Inverse Kinematics control to write letters

After obtaining letter trajectories, the robot will begin to write letters by using inverse kinematics control.

A function ***IK_write*** is defined to implement this, it has 7 parameters:

- *letter*: Latin letter for robot to write
- *regression*: Letter trajectory regression method
- *n_states*: Number of states used in trajectory regression method
- *plan*: Robot writing letter in vertical or horizontal plan
- *orientation*: Robot's end effector point orientation. 'front' should be used when writing in vertical plan, 'down' should be used when writing in horizontal plan
- *useNullSpace*: True to use null-space inverse kinematics control; False to use inverse kinematics with joint damping coefficients
- *resample*: Resample rate of letter trajectory from regression method, use it for value between 0 and 1 to slow down the robot writing

Some other important parameters are globally predefined in the script :

- lower limits, upper limits, joint ranges, rest-poses (for null space)
- joint damping coefficients (when not using null space)

Please see the source code as well as detailed comments in github repository:

https://github.com/eddieai/pybullet_inverse_kinematics_write_letter/blob/master/mini_projet.py

Results

By using the function *IK_write* (inverse kinematics with null-space control) as presented above, I controlled the robot to write firstly each letter in “HELLO” in a vertical plan in front of the robot, and then each letter in “WORLD” in a horizontal plan, with the robot’s end effector pointing down.

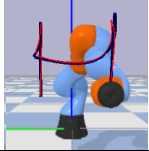
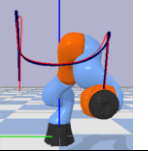
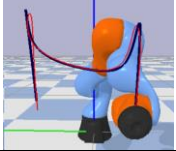
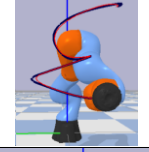
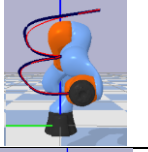
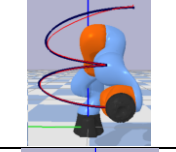
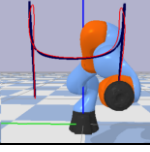
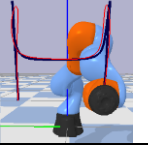
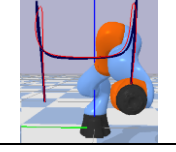
Please see the demo video:

https://drive.google.com/file/d/1eJQeoz_thZ34yjBygyWoDStb73FIE313

Discussion and conclusions

When testing different parameters of script, I have made two major discoveries:

1. Comparison of different regression methods

	LWR	GMR	DMP-GMR
letter = 'H' n_states = 5			
letter = 'E' n_states = 5			
letter = 'H' n_states = 20			

Conclusion: For all three regression methods, number of states is an important parameter. If number of states is too small, the regression trajectory risks to be inaccurate, especially for more complicate trajectories (such as ‘H’). That’s why I finally I choose to use $n_states = 20$. I also observe that LWR (2 degree polynomial) has better performance over GMR or DMP-GMR under small number of states.

2. Comparison of inverse kinematics with and without null-space control

I observe that when using null-space control, the robot can always adjust the optimal joint poses. While not using null-space control (only use joint damping coefficients), the robot sometimes can not adjust well the joint poses, and thus some joints might even get blocked because they arrive at their limits.

This is especially obvious when the robot switch from writing letters in vertical plan to horizontal plan. In fact, when using null-space control, the robot can switch successfully from vertical plan to horizontal plan and write all letters correctly. However while not using null-space control, after switching to horizontal plan, the robot can not perform correctly to write the second letter (‘O’).

Please refer to the recorded videos for detailed comparison:

- With null-space control:
https://drive.google.com/open?id=1SsmM1_2NqPfX2Pn5B15AzrtchuOsWLk
- Without null-space control:
https://drive.google.com/open?id=1HnWofc_FUosQPmrclzvizmKPM5vDEd4e