

Task priority assignment with collision avoidance

Stefano De Filippis & Marco Menchetti

Sapienza - University of Rome



Why priority?

- Decomposition of problems in many tasks.
- Most problems **can't** be solved by just one task.
- Error is kept on the tasks that **can't** be executed **EXACTLY**
- More natural and smoother behavior.



Collision avoidance. How?

1. Control points

We push constantly the control point away from the obstacle so as to be sure they will never get in contact!

How do we handle priority?

- When the control point is too near the priority lowers (i.e. becomes more important)
- As soon as the distance exit the *dangerous* region the priority rise again

TODO: Figure of the KUKA and its control points



Collision avoidance. How?

2. How do we push?

We change approach whether the control point is on the e-e or on the structure:

- For the end-effector we add a cartesian velocity pointing away from the obstacle
- For the ones on the structure we add assign a cartesian velocity projected along the distance from the obstacle (1 DOF).

TODO: figure of the repulsive velocity



Tasks

We know why to prioritize Tasks, but which are the ones we are going to use?

- 1 A cartesian positioning task (i.e. we want our e-e to behave in a certain way)
 - 2 DOFs
- 2 An orientation task used to simulate any kind of auxiliary task
 - 2 DOFs
- 3,4 Two collision avoidance task, each one on 1 DOF
 - 2 DOFs

In the end we saturated all the 7 DOFs of the manipulator.



Task 1: Cartesian positioning

Cartesian positioning means we want the end effector to execute a given trajectory $\{\text{TODO:check timing law}\}$ in \mathbb{R}^3 .

Path used:

- A linear path
- A circular path

TODO:pic

The associated jacobian J_1 is the analytical jacobian of the direct kinematics.



Task 1: Collision avoidance

- We have **7** tasks and we can't add more. How is it performed?

IDEA:

Add another "safety" velocity to the desired one, pointing away from the obstacle!

TODO:pic

- In this way the *Flacco Matrix* will handle the **exact** joint velocities so as to execute the **sum** of the two.
- This won't change the jacobian of the task.



Safety velocity

How do we choose? **TODO:** **TODO:**pic



Task 2: Link orientation

The orientation task tries to keep constant the elevation and azimuth of the **TODO**:link axis and it can be defined as follows **TODO**:check:



Tasks 3,4: Control points

We chose the following:

- **TODO:**
- **TODO:**

TODO:pic



Tasks 3,4: Collision avoidance

As already introduced we can't perform collision avoidance for the control points using all the three component of the distance vector.

PROJECT

Using the same safety velocity we can project it on the distance from the obstacle, obtaining a "safety speed" we will call v .

$$v = \eta^T \dot{r}_o = \eta^T J_i \dot{q} = J_{c,i} \dot{q}$$

J_i is the jacobian at the i -th control point.



Code



Results

