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)
   - **3 DOFs**
2. An orientation task used to simulate any kind of auxiliary task
   - **1 DOFs**

3,4 Two collision avoidance task, each one on 1 DOF
   - **2 DOFs**

In the end we saturated 6 out of all the 7 DOFs of the manipulator.

Task 1: Cartesian positioning

Cartesian positioning means we want the end effector to execute a given trajectory 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 **4** tasks occupying **6** DOF and we can't another cartesian positioning task. How is it performed?

IDEA:

Add another repulsive 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.

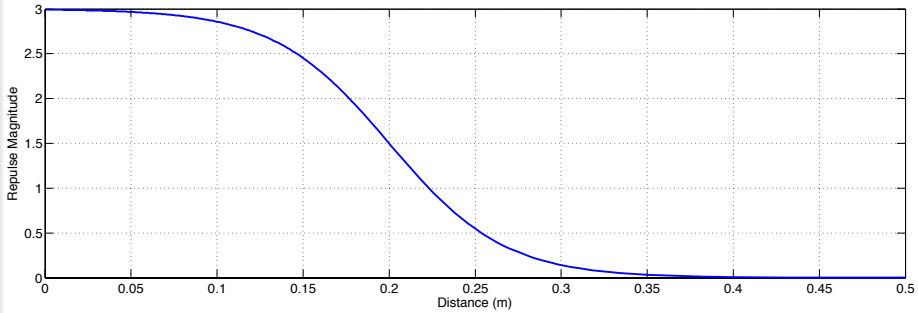
Repulsive velocity

How do we choose?

We want the repulsive velocity to satisfy a certain amount of properties:

- Maximum admissible cartesian velocity at distance $d = 0$ from the obstacle $\rightarrow V_{max}$.
- Smooth descending curve $\rightarrow \alpha$.
- Zero velocity after a given distance from the obstacle $\rightarrow \rho$.

TODO:right v_{max}



Hence:

$$v(P, O) = \frac{V_{max}}{1 + e^{(\|D(P,O)\|(2\rho)-1)\alpha}}$$

Task 2: Link orientation

The "orientation task" tries to keep constant the elevation of the third link axis. We need to define it as a vector in $\mathbb{R}^3$:

$$p_l(q) = p_5(q) - p_4(q)$$

Applying a coordinate transformation into spherical ones we can easily get the expression of the elevation (dropping the dependencies on q):

$$\phi = \arccos\left(\frac{p_{l,z}}{\| p_l \|}\right)$$

Denoting $p = \frac{p_{l,z}}{\|p_l\|}$ we can get the expression of the associated jacobian as:

$$\dot{\phi} = \frac{\partial \phi}{\partial p}\frac{\partial p}{\partial q}\dot{q}$$

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. So what?

PROJECT!

We can project the same repulsive velocity we used on task 1, on the distance from the obstacle, obtaining a "repulsive speed" we will call v

TODO:check on report.

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

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

Code

Results