Universitat de Girona

IFRoS MIRS

HANDS-ON INTERVENTION:
Vehicle-Manipulator Systems

Lab 6: Task-Priority kinematic control (2B)

Patryk Cieślak
patryk.cieslak@udg.edu

Exercise 1 (4 points)

The goal of the exercise is to implement a new class, representing a mobile manipulator, that will work with the already developed Task-Priority algorithm. Its kinematics should represent the kinematics of a differential drive robot with a manipulator mounted on top of it. The simulated robot should include a 3-link planar manipulator.

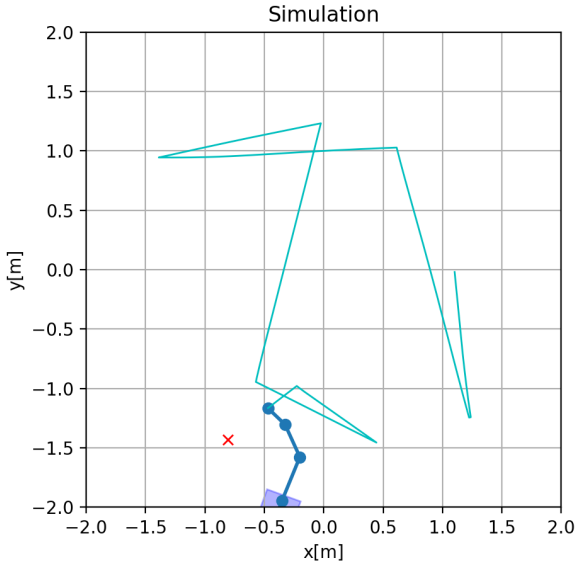

Fig 1. Simulation of the mobile manipulator, including end-effector goal.

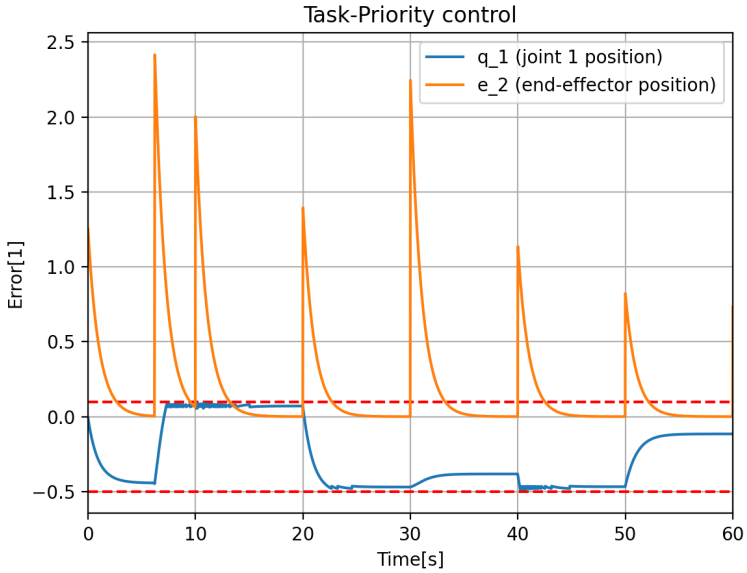

Fig 2. Evolution of the TP control error and first manipulator joint position, over time.

The following elements have to be included in the simulation program:

1) Import of necessary libraries.
2) Definition of Denavit-Hartenberg parameters of a 3-link planar manipulator.
3) Definition of parameters of the simulation.
4) Implementation of class MobileManipulator, compatible with class Manipulator (Lab5), with following modifications:
   a) Field to store the pose of the mobile base (position and orientation).
   b) Field to store distance between robot centre and manipulator base position.
   c) List representing all degrees of freedom of the robot.
   d) Rewritten *update()* method including: update of robot joint positions, update of mobile base position and orientation (linear), calculation of the chain of kinematic transformations.

   e) Updated implementation of all other methods, where necessary.
   f) Accessor returning the pose of the robot mobile base.
5) Modification of the *kinematics()* function, to be able to pass the base transformation as a parameter.
6) Definition of a task hierarchy including joint limits of the first manipulator joint and end-effector position task.
7) Visualisation of the robot structure using the animation functionality of *Matplotlib*.
8) Visualisation of the desired end-effector position on the plane.
9) Second, separate plot, displayed after the simulation is finished, presenting evolution of end-effector position task error and position of the first manipulator joint, over time.
10) All plots with titles, labeled axes, proper axis limits, grid.
11) Random desired end-effector position, initialised on simulation start (inside *init()* function).
12) Simulation repeating every 10s.

The following elements have to be included in the report:
1) Drawing of the robot model, including DH parameters and coordinate systems.
2) Code of the simulation program (well formatted and commented in detail).
3) Plot presenting the visualisation of the robot structure in motion (see Fig. 1).
4) Plot presenting the evolution of task error and joint position over time (see Fig. 2).

Exercise 2 (3 points)

The goal of this exercise is to implement the weighted DLS algorithm and show how it impacts the resulting motion of the system when the weights are changed.

The following changes have to be introduced to the code of Exercise 1:
1) Definition of a task hierarchy including only one task - the end-effector configuration.
2) Implementation of the weighted DLS algorithms.
3) Random initialisation of the desired end-effector configuration with every simulation repeat.
4) Plot with two subplots, displayed after the simulation is finished:
   a) The first subplot presenting evolution of the end-effector configuration task error (separately, norm of position error and norm of orientation error), over time.
   b) The second subplot presenting the evolution of velocity output from TP algorithm, for all controlled DOF of the robot, over time.

The following elements have to be included in the report:
1) Code of the program (well formatted and commented in detail).
2) For 3 different values of the weighting matrix:
   a) Plot presenting the evolution of end-effector configuration task error over time.
   b) Plot presenting the evolution of the velocity output from TP algorithm, for all DOF, over time.

Exercise 3 (3 points)

The goal of this exercise is to evaluate the error introduced in the simulation, by the simplified, linear update of the mobile base kinematics, compared to the correct update, taking into account that the robot is moving along an arc.

The following changes have to be introduced to the code of Exercise 2:
1) Three implementations of the mobile base kinematics integration:
   a) First move forward, then rotate.

   b) First rotate, then move forward.
   c) Move forward and rotate at the same time.
2) Desired end-effector configuration defined in a vector (not random). To be able to run multiple times and get the same results.
3) Plot displayed after the simulation is finished, presenting evolution of the end-effector configuration task error (separately, norm of position error and norm of orientation error), over time.
4) Plot displayed after the simulation is finished, presenting evolution of the mobile base position and the end-effector position, on the X-Y plane.

The following elements have to be included in the report:

1) Equations used to implement each of the three integration ideas.
2) Code of the program (well formatted and commented in detail).
3) For each of the integration methods: plot presenting the evolution of the end-effector configuration task error over time.
4) For all integration methods together: plot presenting the evolution of the mobile base position and the end-effector position, on the X-Y plane. Use clearly distinguishable colours and a legend for easy identification of corresponding results.