Master in Control and Robotics

OPKID Lab Report

Simulation of a Collaborative Task Process

Authors:
Conti Fabio, Nurmanov Madi, Pagano Francesco

Contents

1 Introduction

The aim of this laboratory work is to create collaborative tasks with two SCARA robots using the DELMIA CAD software. More precisely, the two robots have to show the ability to accomplish *serial* and *parallel* tasks without any kind of collision between the two or the environment.
As shown in Fig 1, the two SCARA robots are settled inside of a working environment composed of two tables on top of which some cubes are placed. Such cubes must be moved around the environment by the two SCARA robots in individual and simultaneous tasks. The tasks' objectives were:

- Creating a trajectory which involves a pick and place task of a cube. Such operation must be executed by the two robots in parallel and independently of one-another. The two SCARA have to pick the cube on a table and drop it on the other table and the coordination between these two tasks must be taken care of during the execution.
- Creating a collaborative task between the two robots such that one of the two is able to pick a cube and drop it inside the other robot's workspace envelop. Then, the latter SCARA should pick the cube and place it on the other side of the working area, unreachable by the first robot.

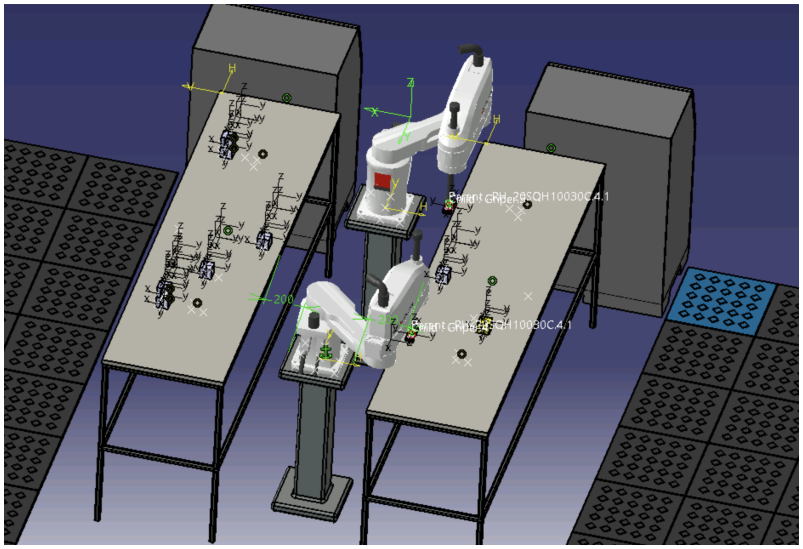

Figure 1: Simulated working environment

2 SCARA Robots Overview

SCARA robots (Fig 2) also known as *Selective Compliant Assembly Robot Arm*, are a fundamental class of open chain robot arms intended for quick pick and place displacements of rather light objects over small distances (food industry, electronic industry). Their general light structure gives them the ability to compute very quick and accurate movements in the workspace. Moreover, it is one of the cheapest category of robots used in the industry. These characteristics make this type of robot the most suitable for our simulated task.

2.1 3T1R Schoenflies motion

The SCARA kind of robots produce **Schoenflies motions**. The acronym **3T-1R** exemplifies the movements the robot structure is capable of executing inside the working environment:

- **3T**: stands for the **three translations** the robot can perform inside the working environment. In the case of the SCARA robots represented inside the simulation (*MISTUBUSHI*

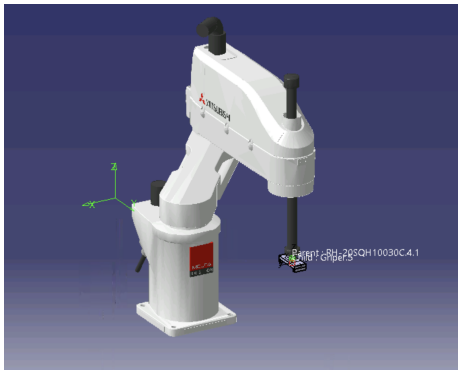

Figure 2: SCARA Robot

RH-20SQH), these three movements are allowed by two rotational joint and a prismatic one. For both the rotational and prismatic joint axis of actuation are the vertical ones. In the case of our robot, the vertical actuation is placed at the end of the chain in correspondence of the end-effector. Some SCARA robots may have the linear actuation in correspondence of the first rotational axis instead of the last one as ours do.

- **1R**: The rotation is given by a rotational joint which axis of rotation coincides with the prismatic joint's movement axis. This joint enables a rotation of the end-effector around the vertical axis adding the ability to the robot to adjust the end-effector position for a better grabbing of certain objects.

2.2 Workspace Shape

2.2.1 Reduced workspace

Even though the link length and the configuration of the SCARA robots' links may vary, the shape of their workspace doesn't undergo major changes. The main workspace alteration a SCARA may encounter has to do with the possibility of designing it with a vertical offset. This change in the design will prevent the robot from self-collisions which would reduce the workspace shape. Since our robot does not provide such feature, the rotational joints will span a 2D workspace (*reduced workspace*) of the following shape 3:

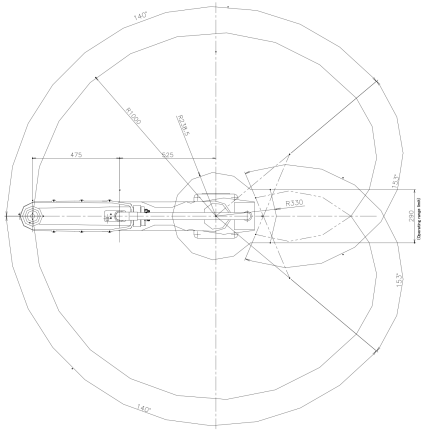

Figure 3: 2D workspace of the SCARA robot

Which clearly shows the impossibility of the robot to compute full rotation around the first joint's rotational axis because of possible self-collisions. However, the spanned area is more than enough for the execution of the *pick and place* required process.

2.2.2 Full workspace

The prismatic joint will give a 3D shape to the work space by adding a vertical translation of the end-effector. as shown in figure 4:

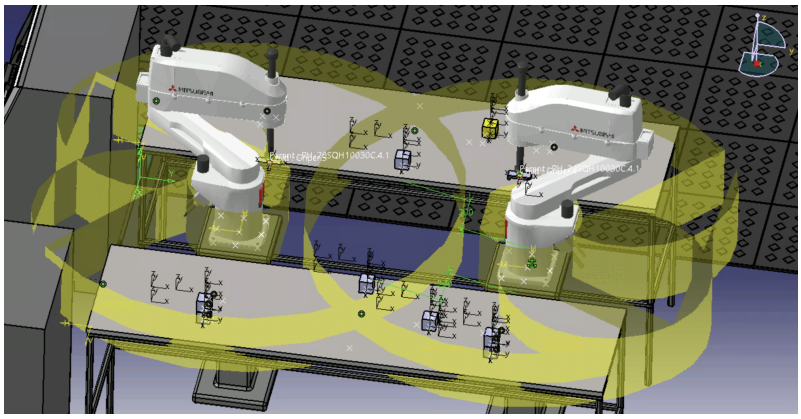

Figure 4: 3D workspace of the SCARA robot

As it is shown in figure 4, the workspace of the two SCARA robots in the working environment overlap. This overlapping condition opens the door to multiple cooperation tasks possibility. As it will be shown the the *methods* section, the two robots will compute a serial task which implies cooperation between the two. However, such configuration may cause collisions between the two arms during parallel task execution. This condition has been encounter during the first simulation of the task. In the following sections we will describe how we prevented this from happening.

2.2.3 N-connected region for pick and place operations

For any kind of robot, the only requirement for pick and place operations is to have an *n-connected workspace*. The n-connected regions of a robot workspace are such regions where the robot's end-effector can move between any two different points. For a pick and place task such as the ones performed in for our simulation the only requirement is to have final and ending point of every *p&p* operation falling under the same n-connected region.
Since the two robots are obstructed by some obstacles inside their workspace, their working volume cannot be considered as *fully n-connected*. The pick and place operation can be successful regardless of what points are chosen to be the starting and ending position as long as they both lay inside the robot's workspace.

2.3 SCARA movements abilities

2.3.1 Dexterity

"Hand agility" is what *dexterity* is defined as. It may be simply described as a measurement of how easily the robot's end-effector can execute small, random movements in its workspace under a certain configuration. The robot's Jacobian matrix is used to calculate dexterity in relation to a certain joint configuration. Thus, the relationship between operational velocities and joint rates and dexterity are tightly associated.
Since our SCARA robot is equipped with three axes of rotation shoulder, elbow, wrist, and a linear axis for vertical movements, it is known to have an high dexterity. Therefore, SCARA robots are typically employed for palletizing processes and quick pick and place applications such as ours. η is a user defined parameter that depends on a_{RDW} (side's length of the regular dextrous workspace) and ρ_{max} (maximum reach of the manipulator).

$$\eta = \frac{a_{RDW}}{\rho_{max}} \tag{1}$$

The RDW is a regular-shaped area (fig.5) of the workplace where the dexterity is higher than a minimally acceptable value, η.

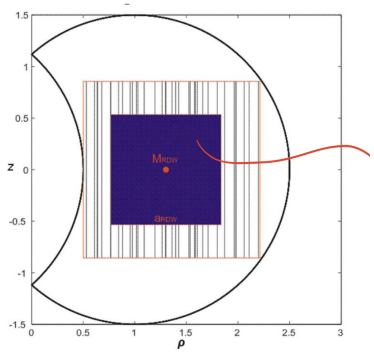

Figure 5: RDW Workspace section

2.3.2 Manipulability

The *Manipulability ellipsoid* is a 2D shape determined in the velocity space $(\dot{x}, \dot{y})$ which characterizes the SCARA robot ability to deal with different kind of movements. The following image shows a possible representation of the ellipsoid for a domain of three joint velocities (Simplified SCARA configuration) fig.6.

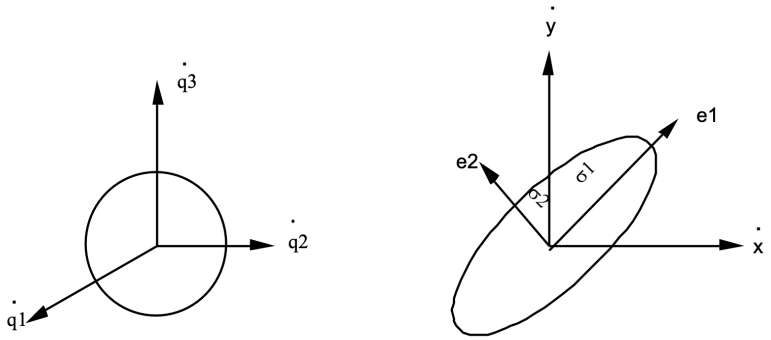

Figure 6: Manipulability ellipsoid for three joint velocities

The two axis of the ellipsoid give some information about the ability of the robot to perform different kind of movements at different operational velocities.

- The direction of the **longest axis** of the ellipsoid is that where:
  - the operational velocity could be the highest;
  - the smoothness of movement (resolution) will be the worst.
- The direction of the **shortest axis** of the ellipsoid is that where:
  - the operational velocity will be the lowest;
  - the smoothness of movement will be the best.

To better characterize such manipulability feature, the *manipulability index* is introduced:

$$w = \sqrt{det(\mathbf{J}\mathbf{J}^T)} \tag{2}$$

For SCARA robots holds:

$$w = l_1 l_2 |sin(\theta_2)| \tag{3}$$

With: l_1 is the length of the first link,l_2 the length of the second link and,θ_2 the angle between the two. The manipulability index measures the volume of the manipulability ellipsoid given a certain $\dot{q}$ configuration.

2.3.3 Isotropy

Any robot in an isotropic condition has the ability to uniformly distribute velocities and forces in all directions. This working state implies that the input and output spaces of the manipulator can be represented as spheres. The *Condition number* evaluates this condition measuring the eccentricity of the manipulability ellipsoid. The *condition number* can be calculated as:

$$k = \frac{\sigma_M}{\sigma_m} \tag{4}$$

Where: σ_M represents the maximum singular value and σ_m the minimum.
If the condition number is equal to 1 ($k = 1$), the robot is in an isotropic configuration. For a SCARA robot to have an isotropic configuration, one must have $Cond(J) = 1$, the *manipulability index* is then twice smaller than the maximum obtained.

3 Methods

Once we got familiar with the DELMIA CAD software we started generating the required tasks in the following order:

1. Loop trajectory for the first robot
2. Loop trajectory for the second robot
3. Cooperation trajectories for the first robot
4. Cooperation trajectories for the second robot

For doing this we used the *Teach* DELMIA tool which is able to create:

- ***Via Points***: which are able to make the robot end-effector reach a desired pose which can be selected directly from the working environment by choosing a frame that corresponds to a desired end-effector position.
- ***Processes***. Processes allow the robot to execute pick and drop operations and must be preceded and followed by two *Via Point*. The time duration of a single process is configurable.This capability turned out to be crucial for the robots timing coordination to avoid collisions.

3.1 Loop Trajectories

The trajectories have to be taught to each robot in order to move the cubes from one table to the other inside the working environment. As aforementioned, we used a set of *Via Point* and *Processes* in order to approach a cube, picking it up, move it on the opposite side of the workspace, and drop it in a specific position. Since there are some obstacles (such as the table), the workspace is not fully n-connected meaning that we cannot state that the robot is able to reach any two points in the workspace. Therefore, before selecting the necessary via points to pick and drop the cubes we had to assure task feasibility. To assure such condition, we made sure that the non operating robot did not interfere with the movement of the operating one. Moreover, we also created the cube approach trajectories such that the end-effector will never collide with the table in either the pick or drop task.
Every task's planning procedure is the same for every trajectory:

1. Select a *Via Point* with a vertical displacement relative to the cube frame. Such *Via Point* will be reached by the robot's end-effector;
2. Select the cube frame as the next *Via Point* to reach;
3. Start the picking/dropping *Process*. The robot will pick/drop the selected cube and close/open its gripper.
4. Select the first *Via Point* so that the robot is able to move and accomplish the next task.

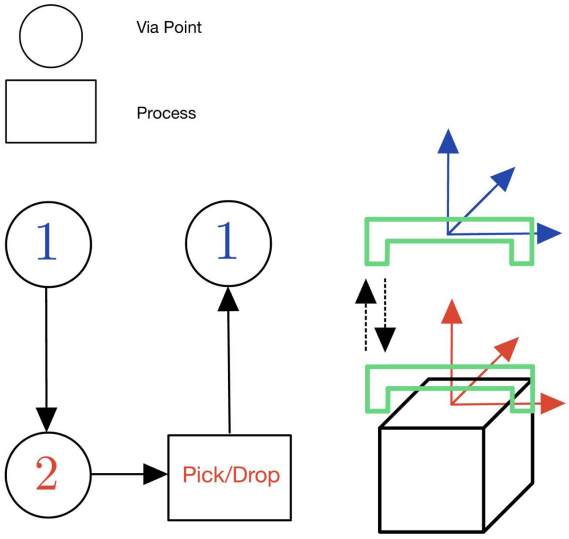

Figure 7: Schematic of the *Via Point* and *Process approach*

3.2 Independent Task Trajectory

The first trajectory development concerns the execution of one independent task for each of the two robots. The task consists of picking a cube up inside the robots' own working area and dropping it at the other side of the table on top of another one (fig.8):

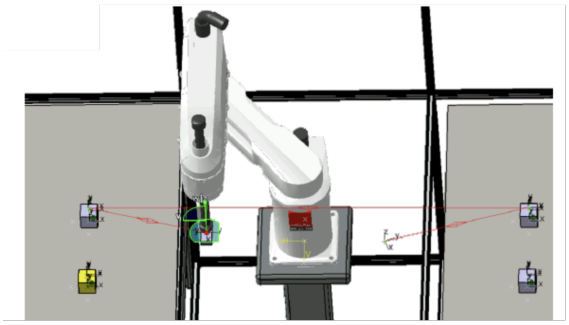

Figure 8: Representation of task 1

In order to ensure continuity of further task execution, is case it will be required, the cubes are placed on top of one-another, allowing to move them back and forth in the workspace. This way, the target trajectories and coordinates are remaining the same for new tasks, which grants reproduction of the results.

If the two identical task are performed in a serial way the two robots will never collide since the non operating one would wait for the moving to finish execution (fig.9):

Figure 9: Serial schedule of the two robot's tasks

Though, this kind of operation may not be very efficient because the two robots are simply waiting for each other and indeed wasting time. Since no collaboration is involved, such task should

be performed in parallel instead of serial approach. This kind of execution will reduce the total operational time of the task accomplishment, but at the same time some collisions may occur.

3.3 Parallel Activities

By using the *Workcell Sequencing* workbench we played the simulation in parallel as shown in Fig 10.

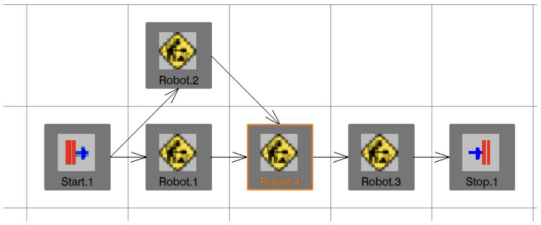

Figure 10: Parallel schedule of the two robot's tasks

During the parallel simulation, we noticed that the two robots were colliding during their loop trajectories motion. To confirm such collision we also used the collisions detection tool disposed by the DELMIA program (fig.11):

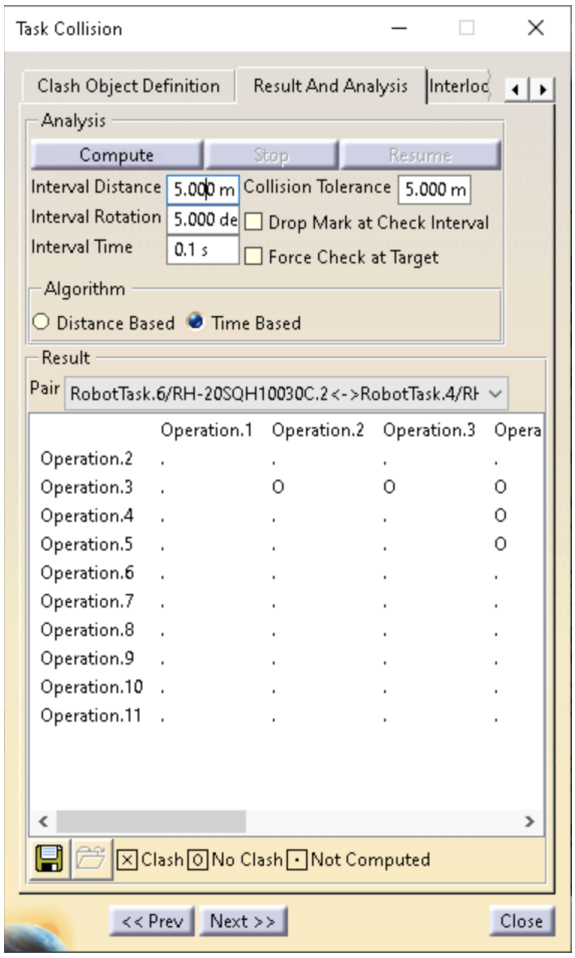

Figure 11: Task collision detection table

The collision between the two robots is due to the overlapping of the robots' workspaces and trajectories required for the accomplishments of the tasks. Therefore, the robots collide during

the execution of the parallel tasks. In order to re-synchronize the two motions and prevent the collision, we incremented the duration of one picking *process* by introducing a **1 second delay**. This change will make the robot "wait" for the other to reach the other side of the working environment avoiding any kind of collision. Thus, the collaborative task was successfully achieved and the overall execution time was shortened by a few seconds.

3.4 Cooperative Task

The second part of the execution concerned the fulfilling of a collaborative robots task. The robots have to show their ability to complete tasks in series without colliding. The objective of this task is to have one of the two robots to pick up a cube and drop it inside the workspace area shared with the other robot. The second SCARA should then pick up the cube and move it out of the shared workspace into its own. To do so, we first made sure that the robot was actually able to drop the cube inside the other robot's workspace. As shown in figure 4, we were able to plot the workspace of each robot and we were able to identify the region of intersection between the two. In correspondence to the table surface, both robots were able to pick up and drop cubes off inside the shared zone. (see Fig.4.) By following the aforementioned procedure, we created all the necessary trajectories to fulfill the serial task.

4 Results & Conclusions

The goal of this work was to teach robots how to accomplish several tasks in a collaborative way between themselves. To do so we simulated a pick-and-place process, taught robots the necessary trajectories and scheduled them both in a serial and parallel framework. Parallel scheduling of the two robots' tasks is to be preferred to minimize robot waiting times and increase process productivity. Therefore, the collaborative task is fulfilled and the overall execution time is shortened.
Serial scheduling of the robots' tasks is safer than Parallel scheduling since the robots start moving only when the other robot finished its tasks and returns to its initial configuration. On the other hand, execution time results are inevitably greater.
The results we obtained in the *DELMIA* simulation software are shown in the following table (Tab. 1):

Table 1: Comparing of the two scheduling methods in terms of time performances

Scheduling Type	**Serial**	**Parallel, with 1s delay**
Time taken	17.858 s	14.002 s

Such time saving is critical within a major supply chain in which such processes are repeatedly performed many times a day. The big industries require not only high precision, but also high reaction and execution speed for moving products by multiple robots on the conveyor, depending on the scenario. However, because neighbouring robot arm moves and creates a new obstacle and risk of collision, parallel scheduling of tasks necessitates a more precise failure detection. To prevent collisions, the robots' motions must be carefully examined and then synchronized to ensure high performance of the system.