Coordination of Several Robots based on Temporal Synchronization

Andrés Montaño and Raúl Suárez

Institut d'Organització i Control de Sistemes Industrials (IOC)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain

Abstract

This paper proposes an approach to deal with the problem of coordinating multi-robot systems, in which each robot executes individually planned tasks in a shared workspace. The approach is a decoupled method that can coordinate the participating robots in on-line mode. The coordination is achieved through the adjustment of the time evolution of each robot along its original planned geometric path according to the movements of the other robots to assure a collision-free execution of their respective tasks. To assess the proposed approach different tests were performed in graphical simulations and real experiments.

Keywords: Multi-Robot Systems, Motion Planning, Temporal Coordination, Motion Synchronization.

1. Introduction

The efficient coordination of several robot arms in order to avoid collisions while they carry out some independent given tasks in a common workspace is a frequent problem of relevance in several robotic fields, both in industrial and service applications. This work proposes a practical approach to solve this problem modifying the temporal evolution of the robots along their precomputed geometrical paths, as it was initially presented in [1].

The problem of coordinating the movements of several robots working in a common workspace is an important issue in robotics and manufacturing as described in several recent works [2, 3, 4]. This problem can be solve following two different strategies, which lead to the centralized and decoupled approaches [5]. In the centralized approaches multiple robots operating in a shared workspace are considered as a single multi-body robot operating in a composite configuration space including the Degrees Of Freedom (*DOF*) of each robot, and then classical planning algorithms are applied to simultaneously find coordinated collision-free paths for all the robots. In the decoupled approaches each robot is treated as a single independent system and the motion planning process is divided into two phases; in the first phase an independent search for each robot path is performed considering only static obstacles and ignoring the presence of other robots in the environment, whereas the second phase (either

off-line or on-line) applies coordination methods to avoid potential collisions when the robots are executing the movements simultaneously in the shared workspace.

The advantages and drawbacks of the centralized and decoupled approaches are presented in [5] and a comparative study of both approaches using a PRM planner is presented in [6]. The conclusion was that in applications that require critical coordinations (small clearances) the use of a centralized planner is more desirable. The centralized approach is complete but it involves a higher number of *DOF* and therefore it is computationally much more expensive than the decoupled approach, which could then be a valid option from the practical point of view.

Many approaches has been proposed to solve the trajectory coordination problem for manipulator robots using the decoupled strategy. The use of priorities was one of the first tools used to search for the robot coordination, by assigning priorities to the robots and sequentially searching for collision-free paths for the robots in order of priority in the configuration-time space [7]. Another approach proposed the use of a prioritization scheme to determine the robots that must adapt their movements in order to avoid collisions with the other robots using attractive elastic forces and repulsive potential field forces to modify the robot paths [8]. Prioritization has also been used with control techniques to coordinate industrial robots [4], in this case task-priorities and sliding control theories are combined to achieve the robot coordination. The main idea of this approach is to define constraints for the multirobot system in order to satisfy them using sliding control and a coordination supervisor, which generates the commanded joint accelerations for the robots. Priority schemes used in industrial applications are usually static, but service applications imply scenarios where priorities may change while the tasks are being executed. The approach proposed here can also use priorities to select the rules of motion in order to avoid collisions. The coordination is achieved modifying only the time evolution along the robot paths, while the geometric trajectory defined by the each robot path is not modified at all. Besides, in service applications the planned motions are likely executed only once because, in general, service tasks are always different and if they have to be repeated it is under different conditions, and the motion planning has to be done on-line; therefore, if there are several robots in the workspace, in order to avoid collisions their motion coordination has to be done also on-line. Broadly speaking, in off-line approaches the objective is to plan time or energy optimal motion trajectories because the computation time is not an important factor, but, in on-line approaches, this optimization cannot be satisfactorily achieved because the complete robot plan may be unknown and the computational time of the motion optimization is usually too large.

An analysis and classification of multiple robot coordination methods was presented in [9], showing that the motion coordination algorithms can be applied on different representations of the workspace (e.g. physical space, composite configuration space, composite configuration-time space, path-time space or coordination space). In all cases, the main goal is to find a coordination curve in the corresponding space that avoids collisions between the robots [10]. There are different approaches to find this curve, like for instance adding a precomputed time delay at the beginning of the movement executions guaranteeing the collision avoidance between the robots [11, 12, 13]. On-line approaches has been also proposed. An event-based approach for on-line and off-line collision-free trajectory planning for dual-arm assembly systems was proposed

in [14], the approach is based on a fast geometric collision detection algorithm, but the robot paths are fully known a priori and the obstacles in the coordination space are discovered by checking the collision between all the robot configurations. Another real-time approach has been proposed for a dual-arm system using a heuristic searching method in the configuration space of the robot [3]. All these methods require a priori knowledge of the robot paths in order to build an entire representation of the coordination space, this is a time expensive procedure, and it is valid only if the robot paths do not change. The approach proposed here does not require a priori analysis of the coordination space, instead of this, the coordination space is explored while the robots execute their tasks, this allows to work with partially known paths.

Dynamic programming has been also used to find a coordination curve [15, 16], in this case, the main goal is the minimization of the execution time of the tasks, considering the dynamics of the robots and the torque restrictions in the robot joints. The obtained coordination curve is used to design the velocity profile for each robot so that collisions are avoided. The robot coordination can be also achieved introducing an adjustment in the geometric paths identifying the regions of the physical space swept by the robots and then modifying the paths planned a priori so that the robots do not occupy these regions simultaneously, if it is not possible to modify the robot paths then their execution time is modified so that the conflictive regions are occupied by only one robot at a time [17]. The problem of multirobot coordination in pick-and-place tasks on a conveyor band has been addressed in [18], presenting an approach based on non cooperative game theory where each robot uses local observations of the conveyor band and their neighbor robots to decide its actions. Each robot chooses the actions that are optimal for it, minimizing a cost function that depends of the relative positions of the robots and the products on the conveyor band. This approaches are valid for applications where the task is repetitive and can be optimized off-line. The approach proposed here intends to be useful for service applications, which are not repetitive and, besides, the paths to accomplish the tasks may be not completely known a priori, which does not allow the use of off-line optimization methods.

A method that solves the robot conflicts based on a path modification sequence was introduced in [2]. The coordination is achieved by re-planning of the paths of the robots in collision. The paths are ordered in a dynamically computed path modification sequence, which selects the path that must be re-planned. On the contrary, in our proposed approach the robot paths are not modified at all, and the coordination is achieved modifying the time evolution along the robot paths, which requires less computation.

The differences between the approach proposed in this paper and other coordination approaches can be summarized as follows. Most of the coordination methods require a priori knowledge of the robot paths to build the coordination space (off-line). The proposed approach just needs knowledge of a limited set of intended movements of the robots since the coordination space is explored at the same time as the robots execute their paths (on-line). The coordination is achieved by the modification of the time evolution along the robot paths, thus, the geometric robot paths are not changed at all. The proposed approach can use priorities to select the proper set of rules used to decide the time evolution along each robot path.

The paper is organized as follows. Section 2 presents an overview of the proposed approach, describing the main features and, specially, the advantages and drawbacks.

Section 3 formally describes the proposed approach, it includes a subsection dealing with the problem modeling and another one describing the coordination procedure itself. Section 4 describes the application of the proposed approach to the case of two robots, including simulated and real experimentation, in such a way that different aspects can be illustrated in detail. Section 5.1 discusses the extension to the case of more than two robots, using simulated examples with three robots to illustrate the concepts. Finally, Section 6 presents a summary of the proposed approach, a brief discussion regarding its application, and expected future work.

2. Overview of the Proposed approach

According to the categories describes in previous section, the robot coordination approach proposed in this paper is a decoupled one that can be applied on-line. Basically, it is assumed that several robots have to work in a shared workspace and that their paths have been determined independently of each other (either off-line or on-line), so each robot path does not have collisions with the objects in the workspace but nothing can be guarantee with respect to collisions with the other robots. Then, the coordination is performed by controlling the evolution of the robots along the planned geometrical paths, without producing any change on their geometry. Since the robot paths are described as a discretized sequence of robot configurations, it is assumed that if there are no collisions at two consecutive robot configurations in the sequence then there are no collisions at any intermediate ones (i.e. the path discretization is fine enough).

To illustrate the addressed problem consider the two robots shown in Fig. 1, one of them has to remove the red cans from the table and the other has to remove the white cans (partially occluded by the red ones in the picture). The motion planning is independently done for each robot (either because they are real independent systems or just in order to reduce the complexity and running time of the planning process), so none of the robots will collide with the table or the cans if it is moved alone, but, if the two robots work at the same time collisions between them may actually occur. In order to avoid these potential collisions the proposed approach adjusts the time evolution of each robot along its path according to the movements of the other robot to assure a collision-free execution of their tasks, and this is done while the robots are already executing their movements. Then, the approach requires that each robot knows the sequence of the expected future configurations of the other robots. This information can be exchanged when the robots have already planned it, which is the case in our current implementation, or it could be determined by the mutual observation of their movement evolution complemented with a prediction of the next positions for a close future. In the second case there may be some uncertainty in the actual configurations of the robots which must be considered as a security margin in the collision check.

The main advantages of the proposed approach are: a) being decoupled, the independent planning of the robot paths strongly reduce the computational cost of the path determination; b) the complexity of the coordination is small enough to allow on-line application. c) the robots can advance in their task while the coordination procedure is executed. On the other side, the main drawbacks are: a) there may be no solution to the coordination problem using only time adjustments of the path (like in any other

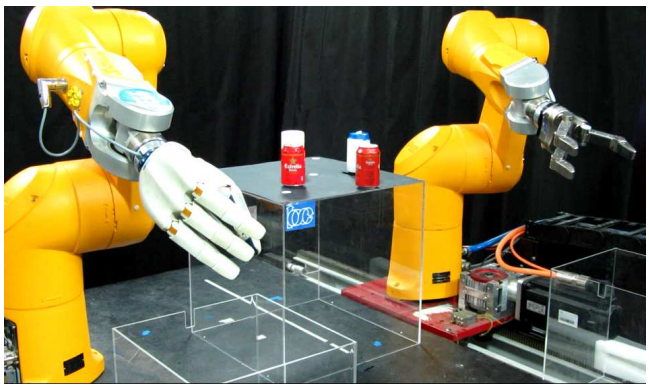


Figure 1: Two robots in a shared workspace. Each robot must grasp and remove from the table cans of different color. The individually computed paths produce collision between the robots, consequently it is necessary a motion coordination in order to avoid them.

approach based only on time adjustments), but if this actually happens the proposed approach can detect it; b) to be of practical application the system must be able to perform more than one collision check between the robots while the robots advance one step in their geometric paths (the influence of the number of collision checks for practical applications is discussed later).

3. Formal Description of the Proposed Approach

3.1. Problem Modeling

Consider n robots R_i, $i \in \{1, ..., n\}$ which have to execute their tasks in a shared workspace following some assigned geometric paths path_i computed independently, i.e. path_i is a set of sequential configurations q_i to be followed by R_i. The geometric path for each robot can be expressed using a path parameter that uniquely identifies the robot configuration along the path as $q_i = \mathrm{path}_i(s_i)$, where s_i denotes the traveling length along the path, with $s_{i_{\max}}$ being the entire path length. The space defined by the points $P = (s_1, ..., s_i, ..., s_n)$, with $0 \leq s_i \leq s_{i_{\max}}$, is called Coordination Space (CS) [19], i.e. CS is the n-dimensional space determined by the n path parameters s_i of the n robots. CS can be discretized considering only a finite set of points $P_k = (s_{1_k}, ..., s_{i_k}, ..., s_{n_k})$, with $0 \leq s_{i_k} \leq s_{i_{k\max}}$, giving as result the Discretized Coordination Space (DCS). The resolution of DCS is given by the composition of the resolution of each path_i, which in practice is determined such that it guarantees collision-free paths, i.e. the movement of a robot between two consecutive collision-free configurations is assumed to be also collision-free. Following this approach, here it is assumed that the movement between two consecutive collision-free points of DCS is also collision-free. The origin of DCS is the point $P_0 = (0, ..., 0)$ and the point at which the robots complete their tasks is $P_{\mathrm{end}} = (s_{1_{k\max}}, ..., s_{n_{k\max}})$. The set of points in DCS representing collision configurations of the robots is called Collision Region (CR). The relative motion between the robots is described by a Coordination Curve (CC) in DCS; a CC may allow robots to move backward, which may

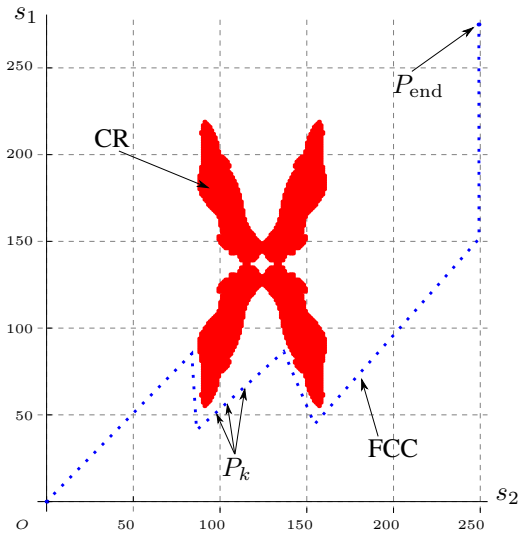


Figure 2: Discretized Coordination Space DCS and Collision Region CR (in red) for two robots. In a real problem CR is not known a priori, but here, for illustrative purpose, it was computed making an exhaustive collision check over all the points in DCS.

be necessary for on-line collision avoidance [14]. If a CC does not pass through CR it is called a Collision-free Coordination Curve (FCC), i.e. a FCC is a set of sequential points $P_k \in$ DCS such that $\forall k \;\; P_k \notin$ CR. Fig. 2 illustrates the DCS for two robots, a Collision Region CR and a Collision-free Coordination Curve FCC.

From a point P_k there are different possible movement directions in DCS, each of them is represented by a Motion Direction (MD). For n robots the number of possible MDs is $\text{N}_{\text{md}} = 3^n - 1$. Fig. 3 shows a piece of DCS for two robots, at any generic point P_k there are eight different possible MDs to move to another point P_{k+1} in DCS (obviously, with the exception of points with coordinates s_{i_0} or $s_{i_{k_{\max}}}$). In this 2-dimensional DCS a horizontal or vertical MD in DCS is equivalent to stop one of the robots while moving the other, i.e. directions (0,+1), (+1,0), (0,-1) and (-1,0). A diagonal MD indicates that both robots are moved, i.e. directions (+1,+1), (+1,-1), (-1,+1) and (-1,-1), either forward or backward depending on the sign. In the general case, the default desired motion direction is $(+1, ..., +1)$, which moves forward all the robots to $P_{k+1} = (s_{1_{k+1}}, ..., s_{n_{k+1}})$, maximizing the overall advance of the tasks that the robots are executing.

The coordination problem can be formulated as: "Given the geometric paths path_i for n robots, find a FCC $\subset$ DCS from the origin P_0 of DCS to P_{end}", i.e. find a sequence of points $P_k = (s_{1_k}, ..., s_{i_k}, ..., s_{n_k})$ from P_0 to P_{end} without passing through the Collision Region CR. When path_i is generated on-line P_{end} may be not explicitly known a priori and then P_{end} has to be replaced by the current "final" point known from each robot path; besides, the Collision Region CR $\subset$ DCS has to be discovered and avoided on-line while the robots are moved along their computed paths.

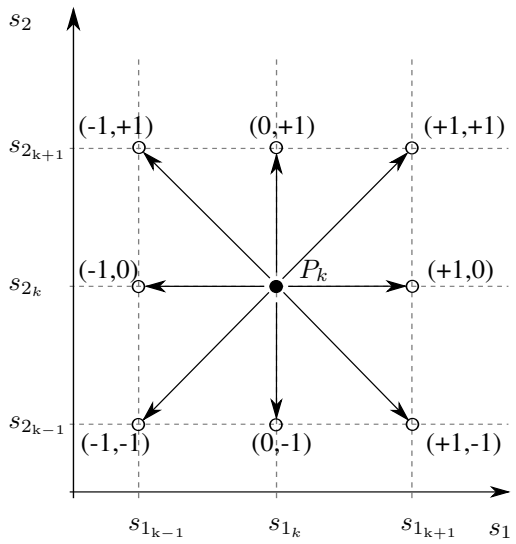


Figure 3: The eight possible motion directions in a Discretized Coordination Space DCS for two robots.

3.2. Coordination Procedure

In order to find a FCC, starting from a given point P_k the next point in FCC is selected using a MD and a collision check is performed in order to detect whether it describes a collision configuration of the robots, i.e. whether it belongs to CR. The tested points of DCS that do not belong to CR are stored in a sequence describing a FCC for the robots.

Assuming that a number $N_{cc} > 1$ of collision checks can be done while each robot advances one step on its geometric path (i.e a transition from one point P_k to P_{k+1} in DCS), during the coordination process the number of points P_k ahead of the robots in FCC increases when the tested points belong to the free space of DCS and thus they can be added to FCC. On the other hand, the number of points P_k ahead of the robots decreases when the selected points belong to CR, since they cannot be added to FCC but the robots continue advancing along the already determined portion of FCC. As it will be discussed later, in critical cases this situation may make the robots stop when many points of CR are checked while the coordination procedure is looking for a feasible solution, i.e. new configurations are not added to FCC while the robots are advancing along it.

Each robot must execute the coordination algorithm to obtain its own trajectory in the physical space, i.e. the evolution in time of the predefined geometric path. As mentioned in Section 2, it is assumed that each robot has information about the next (possible few) movements of the other robots, but there is no general supervisor and therefore each robot must locally decide its next movement according to some predetermined and accepted rules. The algorithms and data used to do this are the same for all the robots so that the global result will be consistent for all of them. On the other hand, priority rules must be established before hand in order to guarantee that all the

Algorithm 1 Main

Require: $\mathrm{path}_i, i = 1, ..., n$
1 FCC$\leftarrow \emptyset$, MD$_k \leftarrow (+1, ..., +1)$, $P_k \leftarrow O$
2 **while** Task is not finished **do**
3 **for** $i = 1$ **to** N$_{\mathrm{cc}}$ **do**
4 **if** $P_{k+1} \neq P_{\mathrm{goal}}$ **then**
5 Determine P_{k+1} using MD$_k$
6 **if** P_{k+1} does not imply collision **then**
7 Add P_{k+1} to FCC
8 $P_k \leftarrow P_{k+1}$
9 **else**
10 Select a new MD$_k$ (using the state diagram)
11 **end if**
12 **else**
13 break
14 **end if**
15 **end for**
16 Move $R_i, i = 1, ..., n$ from its current position to the next one according to FCC
17 **end while**

robots take consistent decisions.

Algorithm 1 shows the main procedure of the proposed approach, which must be executed by each robot R_i. As input it requires information about the next positions of the robots, information that is included in the geometric paths, $\mathrm{path}_i, i = 1, ..., n$. P_{goal} is the point in DCS at which the coordination process is completed and there are no more movements to coordinate, this point can be an intermediate point depending on the information available at a particular time or it can be an absolute final point if the geometric paths are completely known. In the algorithm there are two main actions, the coordination of movements and the execution of them. The coordination implies the exploration of DCS, selecting points P_k, checking them for collisions, and adding them to a FCC if they are collision-free. Since all the robots are running this algorithm the execution of the robot movements implies moving the robots from a point P_k to P_{k+1} in DCS. Both actions must be executed while the goal of each robot is not reached.

In order to determine the next point P_k of a FCC, a state diagram is used with the nodes representing the MDs and the transitions defined according to whether the result of using a given MD produces or not a collision configuration. The state diagram can be designed following different strategies, like, for instance, giving always priority to one of the robots or trying to optimize the overall advance of the whole set of robots. Examples of different state diagrams used to select a new MD are presented and discussed in Sections 4 and 5.1 for the case of two and three robots respectively.

4. Application to the Case of Two Robots

4.1. Particular developments

The approach formulated above for n robots is particularized here for a cell with two robots R_1 and R_2. In this case DCS is a 2-dimensional space and, even when

the coordination is done on-line, path_1 and path_2 are computed off-line for the desired tasks assigned to each robot, thus $s_{1_{k_{\max}}}$ and $s_{2_{k_{\max}}}$ are known, and the condition "Task is not finished" in Algorithm 1 can be formulated as "$s_{i_k} < s_{i_{k_{\max}}}, i = 1, 2$". As mentioned above, the default desired motion direction MD is $(+1, +1)$, and the starting point in DCS is $P_0 = (0, 0)$. It is assumed that two collision checks are executed per cycle, i.e. collisions in two points of DCS can be checked during the movements of the robots between two consecutive points P_k and P_{k+1}.

In order to select the motion direction MD at each transition, two heuristics were implemented. The first heuristic is based on the wall follower, the best-known rule for traversing mazes, also known as either the left- or right-hand rule. The second heuristic is based on the maximization of the overall advance of the robots in each transition in DCS. A state diagram representation is used to determine the selection of the motion directions, where each state represents a MD.

The state diagram in Fig. 4 shows the wall follower heuristic with priority for the robot R_2. The diagram has $\text{N}_{\text{md}} = 8$ states resulting from $\text{N}_{\text{md}} = 3^n - 1$ for $n = 2$. The transitions between states are marked with "C" when the resulting next point is a collision point and with "F" when it is a collision-free point. The initial state (default) is always $(+1, +1)$. For instance, if using $(+1, +1)$ the destination point P_{k+1} in DCS belongs to CR the next MD to be checked is $(0, +1)$, indicating that R_2 moves forward one position and R_1 is stopped. Note that with these conditions when there are collisions configurations the transitions are counterclockwise in the graphical representation of the state diagram; by analogy, if the priority is given to R_1 the transitions would be graphically clockwise. In the state diagram with priority for R_2 shown in Fig. 4, when the state $(+1, 0)$ is reached and the destination is a collision point, a special condition must be considered in order to avoid a closed loop in the graph state. This special condition is marked as the transition C^* in the state diagram, meaning that if the state $(+1, +1)$ is reached through C^* and this MD leads to a collision-free point the next state is determined by F^* instead of F.

The state diagram in Fig. 5 shows the overall impact heuristic with priority for the robot R_2. In this case the eight states are grouped according to the overall impact of motion: the state $(+1, +1)$ has an impact of $+2$, since both robots move forward one position according to their plans, the states $(0, +1)$ and $(+1, 0)$ have a impact of $+1$, etc. The state with the minimum impact is $(-1, -1)$, whose impact is -2, in which both robots move back one position. The transitions between states go from the maximum overall impact to the minimum overall impact, selecting first the states that favours the robot with highest priority. When a collision-free point is reached (marked with a F transition in the states diagram), the next state to be checked is always the state $(+1, +1)$. In this strategy, the points already added to FCC are considered for further explorations as collision points in order to avoid oscillations between two consecutive points in FCC, but as a consequence the system returns an error when the only movement option is coming back to the previous point in FCC (this happens when the flow in the state diagram arrives to the state $(-1, -1)$ and it produces a collision transition; if desired, a specific strategy could be implemented for this case).

The robot priorities can be selected applying different criteria. In the current implementation, the robot with the highest number of intermediate configurations in path_i has the priority (i.e that with largest $s_{i_{k_{\max}}}$). Nevertheless, this criterion is an arbitrary

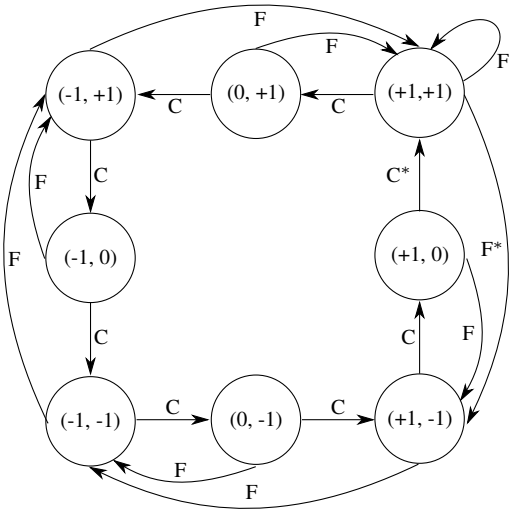


Figure 4: State diagram representing the wall follower heuristic with priority for the robot R_2. The transitions between states are marked with "C" when the resulting next point in DCS is a collision point and "F" when it is a collision-free point.

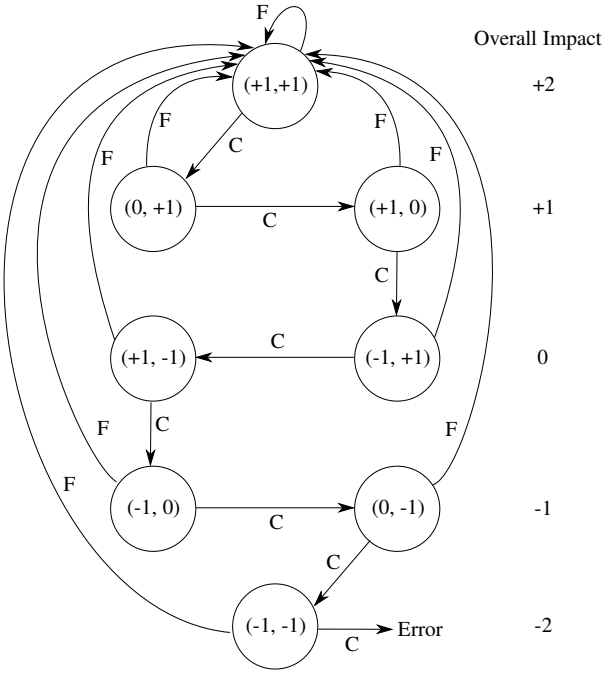


Figure 5: State diagram representing the overall impact heuristic with priority for the robot R_2. The states are ordered by the overall impact of motion, from the highest impact state $(+1, +1)$ to lowest impact state $(-1, -1)$.

choice and since the collision region is unknown it does not assure an optimal solution.

In order to illustrate how the proposed approach works, Fig. 6 shows a simple example of the algorithm generating a FCC and discovering the CR for two robots using the overall impact strategy with priority for R_2. Fig. 6a show the initial situation with the robots in their initial configurations represented by the point $\mathrm{P}_0 = (0, 0)$. Fig. 6b shows the results of the first step, two points of DCS were explored, they were collision-free and therefore added to FCC (both explorations following the default MD $(+1, +1)$). Fig. 6c shows the results of the second step, the robots moved forward one position along FCC while two new points of DCS were explored and, being collision-free, added to FCC, again using the default MD. In Fig. 6d the robots moved forward another position along FCC while two new points of DCS were explored and, in this case, the first checked position using $(+1, +1)$ belongs to CR and therefore the second exploration was done using $(+1, 0)$. In Fig. 6e the same has happened, the robots advanced one position, the first checked position using $(+1, +1)$ belongs to CR and then the second exploration was done using $(+1, 0)$. In Fig. 6f while the robots advanced one step two explorations were done along $(+1, +1)$ without finding collisions and the FCC has surrounded the obstacle. Following this procedure and assuming no more collisions were found, Fig. 6g shows the step in which FCC was completely defined, i.e. FCC reached P_{end}. From this step, it is not necessary to do more explorations and the robots just advance following FCC until reaching P_{end}, as shown in Fig. 6h.

4.2. Experimental Results

The proposed approach has been fully implemented for the case of a real cell with two robots. The code implementation is based on ROS [20] for the communications layer, which is in charge of exchanging the information about the planned movements of each robot, Qt libraries [21] for the user interface, Coin3D for the graphical rendering and PQP [22] for the collision detection. The path planning is computed using the home-developed path planning framework called *the Kautham Project* [23]. This framework provides the developer with several tools needed for the development of planners, like, for instance, direct and inverse kinematic models of the robots and hands, random and deterministic sampling methods [24], metrics to evaluate the performance of planners (number of generated samples, collision check callings, number of nodes in the graph solution, connected components) and simulation tools. For the graphical simulations the robots were modeled using triangular meshes. The robots in the cell are two Stäubli TX-90 with 6 *DOF* equipped with a Schunk Anthropomorphic Hand (SAH) [25] with 13 *DOF*, and a Schunk Dexterous Hand (SDH2) [26] with 7 *DOF*. A PRM planner [27] has been used to obtain the geometric path for each robot, the samples for the path planning are generated in a cloud around the direct linear path in the physical space from the initial to the final configuration.

The synchronization of the real robots is achieved applying event-based control, monitoring the current robot configurations and waiting until each robot reaches its commanded configuration. A simple example of this event-based synchronization scheme is the following: when a robot R_i starts a movement from the current configuration $q_{i_k} = \mathrm{path}_i(s_{i_k})$ toward the next one in the path $q_{i_{k+1}} = \mathrm{path}_i(s_{i_{k+1}})$, a signal WAIT_i is activated, and it is active until R_i reaches $q_{i_{k+1}}$. In order for the robots

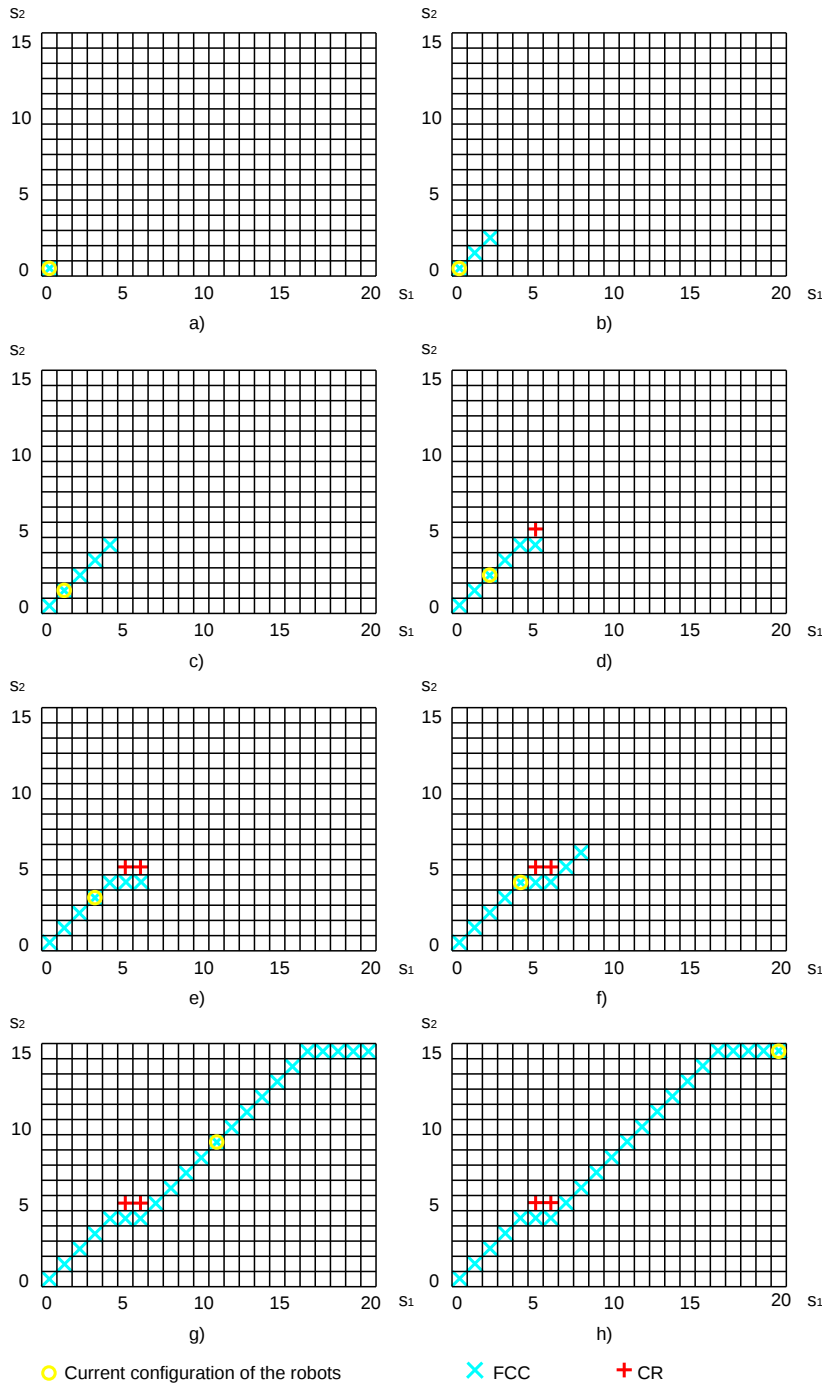
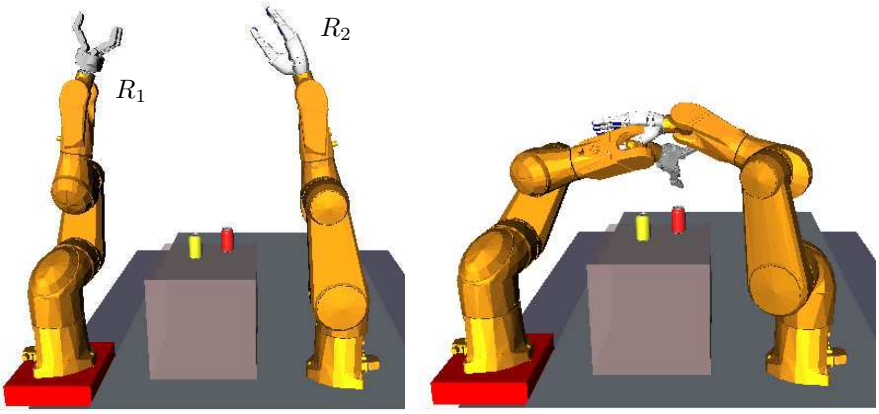


Figure 6: Example of time evolution of the robots along their paths following the overall impact strategy with priority for R_1 as it is shown in the state diagram in Fig. 5.



a) Setup for Example 1. b) Collision without coordination.

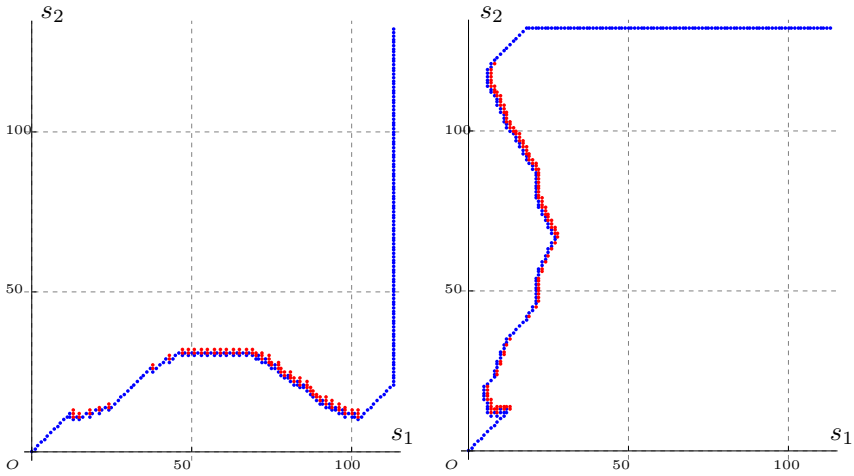
Figure 7: a) Setup for Example 1. The robot R_1 is in charge of remove the red can and R_2 is in charge of the yellow one; b) Collision configuration during a simulated execution without coordination.

to proceed to a new desired configuration $q_{i_{k+2}}$, both signals WAIT_1 and WAIT_2 must be off.

The following two examples illustrate the ability of the proposed approach to coordinate the independently computed paths for the robots.

Fig. 7a shows the setup for the first example, the robot R_1 is in charge of taking off the red can from the table and R_2 the yellow one. Fig. 7b shows a snapshot where the robots are in collision during a task simulation without coordination. The computed paths for R_1 and R_2 have, respectively, $s_{1_{k_{\max}}} = 114$ and $s_{2_{k_{\max}}} = 133$ configurations. Fig. 8a and 8b show the FCC found using the wall follower heuristic and giving priority to R_1 and to R_2, respectively. In the case of priority given to R_1, the search of the FCC required 329 collision checks, the whole FCC has 237 steps and was completely defined when the robots were executing the step 165, R_1 needed 126 steps to finish its task and R_2 needed 237. When the priority was given to R_2 the search of the FCC required 358 collision checks, FCC has 236 steps and it was completed when the robots were executing the step 178, R_1 needed 236 steps to finish its task and R_2 needed 141. In both cases the robot with priority completes the task before the other (which can not be always guaranteed since it depends on the shape of CR). Fig. 9 shows the complete CR computed only for illustrative purpose; in order to find the complete CR it was necessary to execute $s_{1_{k_{\max}}} \times s_{2_{k_{\max}}} = 114 \times 133 = 15,162$ collision checks.

Fig. 10 shows the setup for the second example. The robot R_1 is in charge of removing the red cans, C_1 and C_3, and R_2 is in charge of the yellow ones, C_2 and C_4. The computed path for R_1 has $s_{1_{k_{\max}}} = 426$ configurations, and the path for R_2 has $s_{2_{k_{\max}}} = 289$, thus the priority was given to R_1. The search of a FCC in the coordination process required 728 collision checks using the wall follower heuristic,



a) FCC with priority for R_1.

b) FCC with priority for R_2.

Figure 8: DCSs for the two robot problem in Fig. 7. a) FCC using priority for robot R_1; b) FCC using priority for robot R_2. In both cases, the robot with priority reaches the $s_{i_{\max}}$ before the other one.

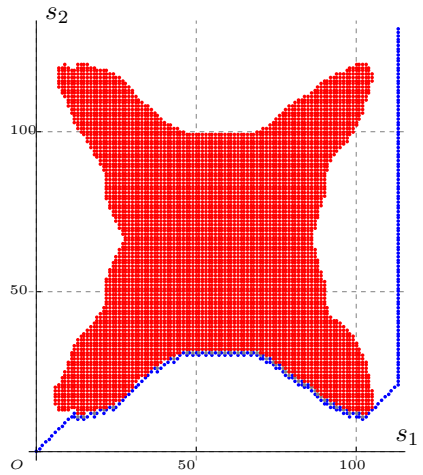


Figure 9: DCS for the two robot problem in Fig. 7 and the complete CR computed for illustrative purpose.

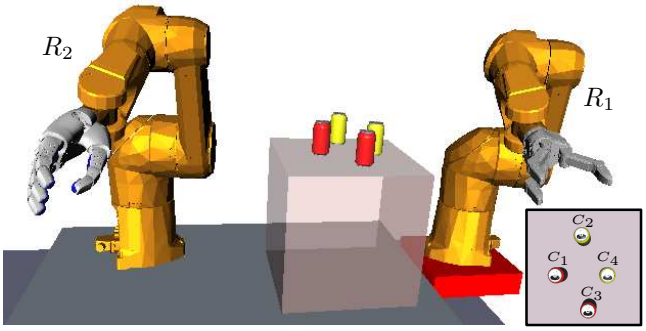


Figure 10: Setup for Example 2. The robot R_1 is in charge of removing the red cans C_1 and C_3, and R_2 is in charge of the yellow cans C_2 and C_4. The square in the bottom-right corner shows a top view of the table.

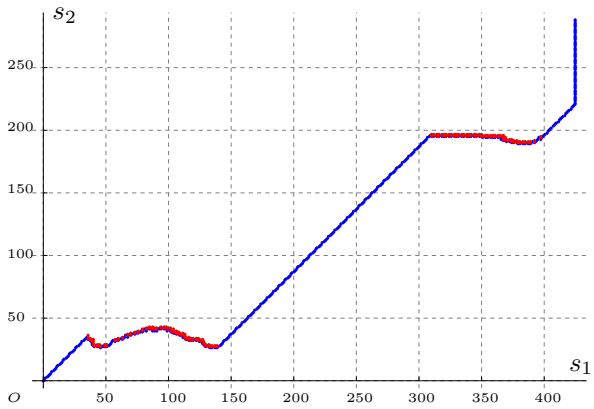
FCC has 506 steps and was completed when the robots were executing the step 364. R_1 and R_2 needed, respectively, 440 and 506 steps to finish their tasks. Fig. 11a shows the computed FCC (in blue) and the checked points of CR (in red). Fig. 11b shows, only for illustrative purpose, the FCC (in blue) and the complete CR (in red). In order to find the complete CR it was necessary to execute $s_{1_{k_{\max}}} \times s_{2_{k_{\max}}} = 426 \times 270 = 115,020$ collision checks. The execution of this example using the real robots is illustrated in Fig. 12 by snapshots of the coordinated movements to perform the tasks of each robot without collisions among them, and a video showing the complete real execution is available following the link in [28]; besides, another video showing the application to a different setup with different robot arms is available following the link in [29], which shows that the approach can be applied to different robotic systems. All the coordination information is the exactly that mentioned in the simulation case, and the total time required in the real execution was 149,2 s. using robot velocities and accelerations of 10% of the maximum one.

In the current implementation, the average execution time of a collision check for two robots was 501.2 μs with a standard deviation of 0.38 μs (this time strongly depends on the particular software implementation and the complexity of the used robot models). Table 1 summarizes the relevant information for the two coordination examples.

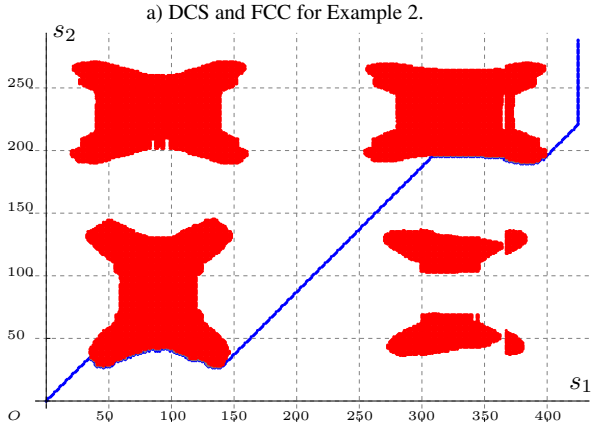
5. Discussion

5.1. Application to more than two robots

In this section we discuss the application of the approach to more than two robots, showing examples with three robot arms to illustrate the concepts. The approach presented in a generic way in Section 3 and described in detail for two robots in Section 4 can be applied to any number of robots, although, as it is expected, the collision check requires more time and the number of possible movements in DCS increases producing the effect described below.



a) DCS and FCC for Example 2.



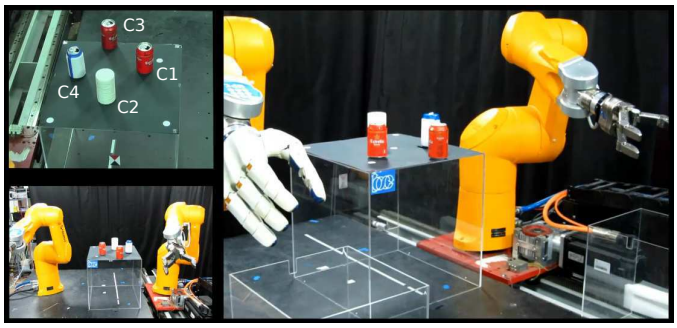
b) Complete CR.

Figure 11: DCSs for the problem presented in Fig. 10. a) FCC and explored CR using priority for robot R_1; b) FCC and complete CR computed for illustrative purpose.

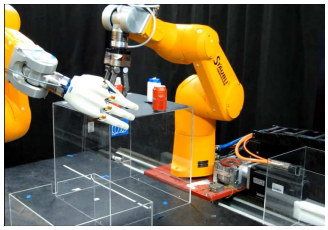
Table 1: Results for the two coordination examples.

	$s_{1_{k_{\max}}}$	$s_{2_{k_{\max}}}$	CC	FCC steps	TS-R_1	TS-R_2
Example 1 Priority R_1	114	133	329	165	126	237
Example 1 Priority R_2	114	133	358	178	236	141
Example 2 Priority R_1	426	289	728	364	440	506

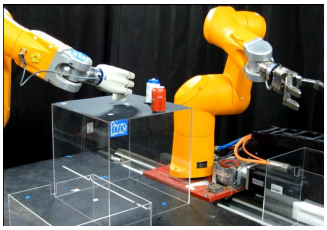
$s_{i_{k_{\max}}}$: Number of points in the geometric path for robot R_i.
CC: Number of collision checks done during the computation of FCC.
FCC Steps: Number of steps executed by the robots when FCC is completely defined.
TS-R_i: Total number of steps finally done by R_i.



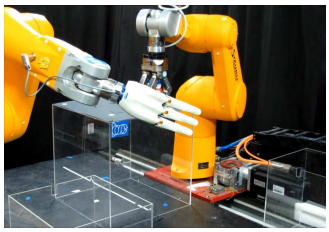
a) Initial configuration.



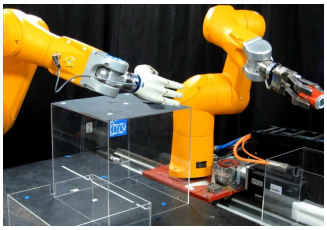
b) R_1 taking C_1.



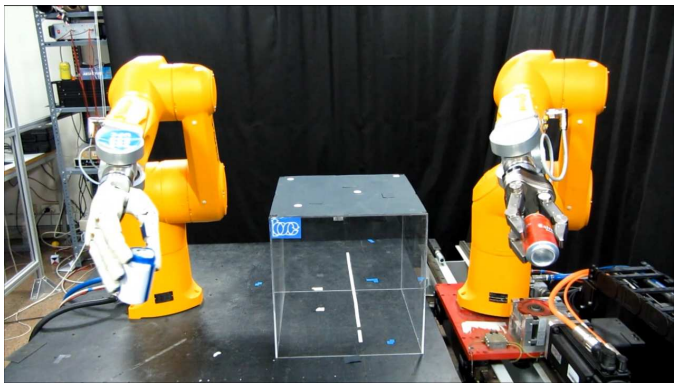
c) R_2 taking C_2.



d) R_1 taking C_3.



e) R_2 taking C_4.



f) Final configuration.

Figure 12: Snapshots of the real execution of the second example.

The computational cost for the case of n robots is a direct function of the dimensionality n of DCS, which, as has been previously stated, means a number of possible motion directions $N_{md} = 3^n - 1$ (i.e. the number of states in the state diagram). N_{md} is then an upper bound for the number of collision checks necessary to decide the next movement in a FCC (it could be reached in very tight relative configurations of the robots). Then, if tight relative configurations of the robots appear frequently during the execution of the tasks, it may happen that the robots advance steps along their paths faster than the generation of new steps in the FCC, and the robots may have to stop at some point and wait for the system to find and add new points to FCC, making the whole system having a poor efficiency. The evident solution to this problem is the increasing of the number of collision checks N_{cc} that the system is able to perform during the evolution of the robots along one step in their paths (see Step 3 of Algorithm 1). Of course the largest N_{cc} the better, but assuring a high value of N_{cc} may impose constraints on the robot velocities, limiting them. On the other side, the complexity of the Collision Region CR depends on the particular tasks and paths assigned to the robots so the required number of points of DCS to be explored is unknown, although it could be really low in many practical cases and therefore increasing N_{cc} may just produce slower robot movements, being then useless.

In order to illustrate the effect of N_{cc} on the coordination procedure we applied the proposed approach to the case of a simulated cell with three industrial robots working in a common workspace (shown in Fig. 13a) using different values of $N_{cc}$and the overall impact heuristic. We present here the results of one particular example that clearly illustrates the concept. As in the examples for two robots, each of the three robots is in charge of grasping a can of a specific color and take it off from the workspace, R_1 is going for the red can, R_2 is going for the blue can and R_3 is going for the yellow can. If the robots execute their respective paths without any synchronization there will be collisions among them (see Fig. 13b). The coordination procedure was applied in this example considering the overall impact heuristic, which for three robots is represented by the state diagram shown in Fig. 14. The independent robot paths have 206, 170 and 102 configurations for R_1, R_2 and R_3 respectively. 1,324 collision checks were required to solve the coordination problem under these conditions. R_1 needed 544 steps to finish its task, R_2 needed 371 steps, and R_3 needed 197. The number of times that each node of the graph state was visited during the search of the FCC is given in the histogram shown in Fig. 15 (it must be remarked that this histogram depends on the geometric paths assigned to the robots and it is independent of N_{cc}). The search of a complete FCC was done using $N_{cc} = 2$, $N_{cc} = 4$ and $N_{cc} = 8$. Fig. 16 shows the number of coordinate points already determined in FCC ahead of the point describing the current position of the robots for the three different values of N_{cc} (i.e. the number of steps that the robots can still advance with guaranty of no collision). For $N_{cc} = 2$ this number is zero during a significant part of the activity of the robots, meaning that all the robots have to stop and wait for the system to find the next collision-free point in FCC. Under this condition, the FCC was completed when the robots were executing the step 449 and 757 steps were necessary for the three robots finishing their respective tasks (544 steps from the largest coordinated robot path for R_1 plus 213 steps while the robots were stopped). In the case of $N_{cc} = 4$ the three robots still have to stop and wait but only in a reduced number of cases, FCC is completely defined when the robot

Table 2: Results for the three-robot coordination example.

N_{cc}	CC	FCC steps	R_1 STC	R_2 STC	R_3 STC	TSTC
2	1324	449	544	371	197	757
4		317				558
8		166				544

$s_{i_{k_{\max}}}$: Number of points in the geometric path for robot R_i.
CC: Number of collision checks done during the computation of FCC.
FCC Steps: Number of steps executed by the robots when FCC is completely defined.
TS-R_i: Total number of steps finally done by R_i.
TTS: Number total of steps executed to complete all tasks.

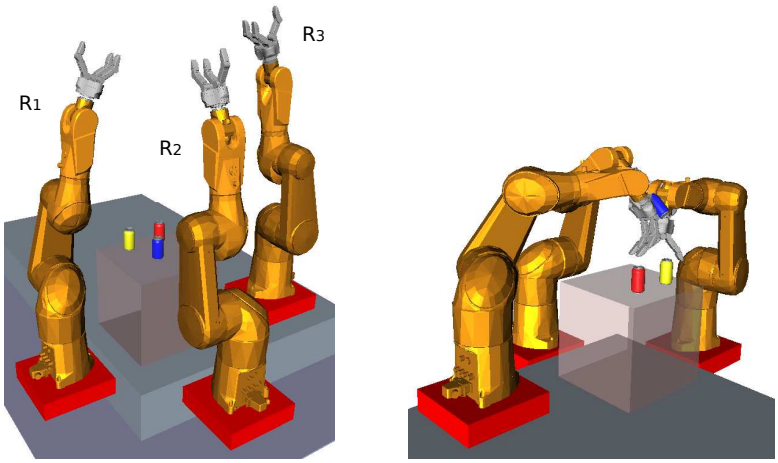
were in the step 317 and all the robots finished their tasks after 558 steps (544 steps from the largest coordinated robot path plus 14 steps while the robots were stopped). Finally, using $N_{cc} = 8$ the number of coordinate points in FCC ahead of the current position of the robots does never fall to zero, even more, it grows monotonically and FCC was completed when the robots were in the step 166, meaning that the three robots were never blocked, and they finished all the tasks in only 544 steps. After executing a number of experiments, we see that in the described experimental setup with three industrial robots manipulating objects in the same workspace, $N_{cc} = 8$ is enough to avoid the robots arriving to a halt, which is significantly smaller than $N_{md} = 26$. Nevertheless, this may be not always true and there is no rule to determine the minimum value of N_{cc} that avoids the robot halt. Reducing the velocity of the robots to allow an increasing of N_{cc} may help in avoiding the stopping of the robots, but in general it will increase the total time required to finish their tasks. Table 2 summarizes the relevant information for the three-robot coordination examples.

One additional advantage of a large enough N_{cc} is that once the whole FCC was already determined (or even before it), it is possible to do additional explorations in DCS looking for an optimization of the FCC ahead of the robots, avoiding backward robot movements and reducing the total time needed by the robots to finish their tasks.

5.2. Optimization of FCC

The FCC can be optimized to prevent that the robots move backward while they follow the FCC. The number of points in the portion of FCC between the last added point to FCC P_k and the point representing the current position of the robots is called Explored Window EW. The FCC can be optimized exploiting the size of EW. Since the size of EW limits the number of points in DCS that can be analyzed in the optimization process before the robots reach the first point P_{inter} in the portion of FCC being optimized. The size of EW increases when the explored point belongs to the free space of DCS and it decreases when the point belongs to CR, and this size also depends on the number of N_{cc} per movement of the robots, as illustrated in the examples in Fig. 16. Therefore, the available time for the optimization also changes in function of EW. A first propose for this optimization was already presented in [30].

The point P_k in which one or more robots must perform a backward movement is determined looking the current motion direction MD. Once a point P_k involving a backward movement of a robot R_i is added to FCC, it is necessary to determine the



a) b)

Figure 13: a) Cell with three robots, each of them has to remove a particular can from the workspace following independently planned paths; b) Collision configuration when the robots follow their paths without coordination.

set of points OPT that contains possible new points of FCC that would replace some points in the current FCC. OPT is composed by the points in DCS linking P_k with a point $P_{inter} \in$ FCC that do not imply a movement of R_i (yellow points in Fig. 17). P_{inter} is selected to avoid the backward movement of the robot with higher priority, in case that more than one robot move backward. Then, it is necessary to check whether the current robot configuration in FCC has not exceed P_{inter}, if this condition is true the optimization can be done, otherwise the optimization of this portion of FCC is not feasible.

The optimization begins from the point $P_{opt} \in$ OPT closest to P_{inter}, if $P_{opt} \notin$ CR then it is added to FCC replacing the point P_{check} that follows P_{inter} in the original FCC (see Fig. 17). The process is repeated until OPT was completely included in FCC or the robots reach the current P_{inter}.

6. Conclusions

This paper has proposed a method for the on-line temporal coordination of multiple robots in a shared workspace whose paths were computed independently. The approach is based on the on-line exploration of the Discretized Coordination Space (DCS) that represents the relative positions of the robots along their corresponding paths in order to find a Collision-free Coordination Curve (FCC). Following this FCC the robots are moved in a coordinated way avoiding collisions between them. The approach has been implemented and successfully applied, in simulations for two and three robots and in real executions for the case two robots.

The approach can be applied to any type of robots, being the only requirements that each robot must able to know the future positions of the other robots (not necessarily the

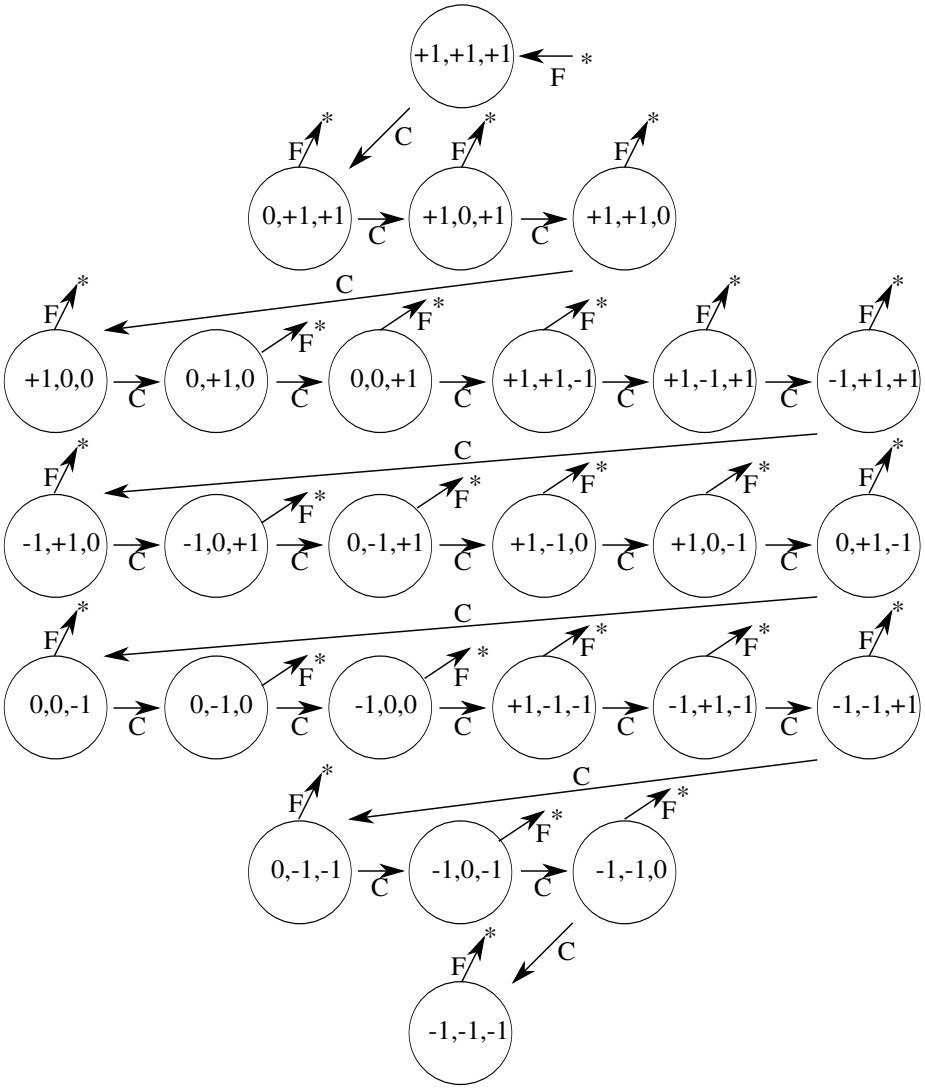


Figure 14: State diagram for three robots using the overall impact strategy.

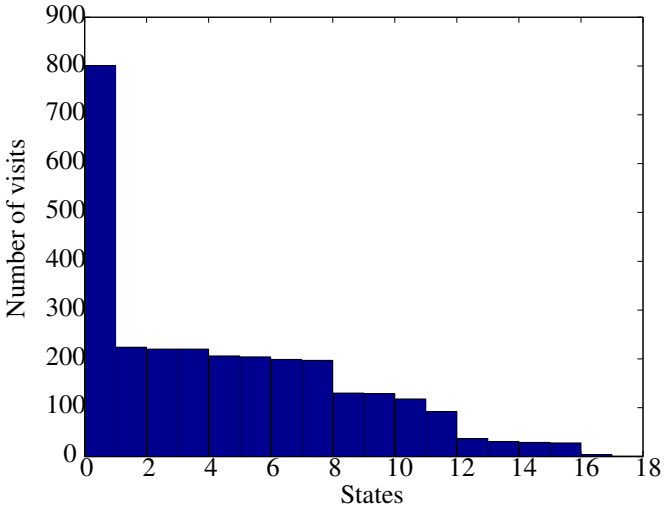


Figure 15: Histogram showing the number of times that each node of the graph state was visited during the search of a complete FCC for the example given in Fig. 13. The axis of abscissa indicates the first 18 states starting from $(+1, +1, +1)$ in the state graph in Fig. 14. and going left to right and top to bottom.

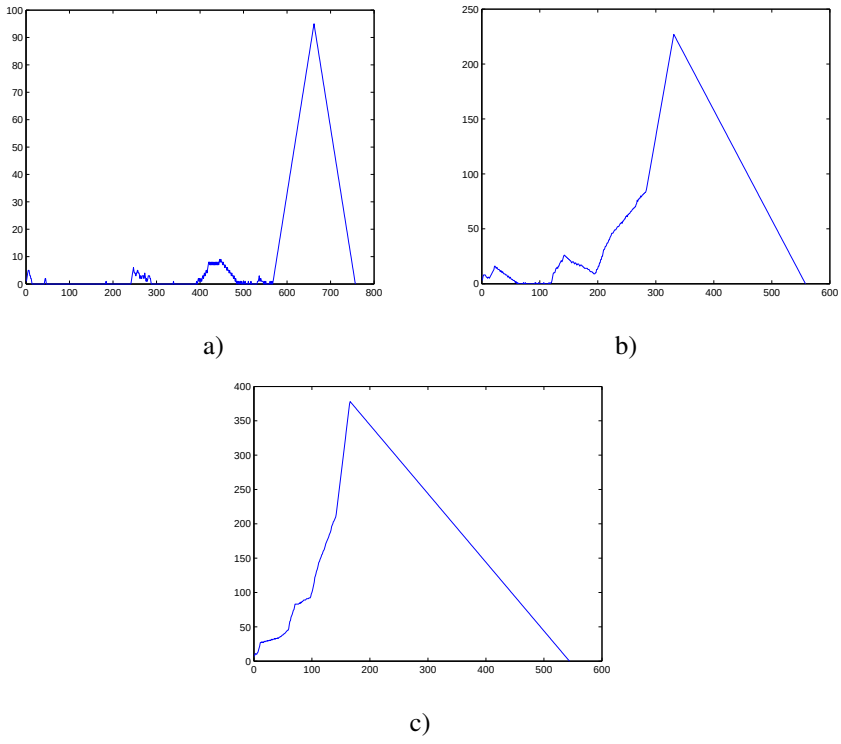


Figure 16: Number of coordinate points in FCC ahead of the point describing the current position of the robots during the whole execution from P_0 to $P_{\rm end}$ for: a) $\mathrm{N_{cc}} = 2$; b) $\mathrm{N_{cc}} = 4$; c) $\mathrm{N_{cc}} = 8$.

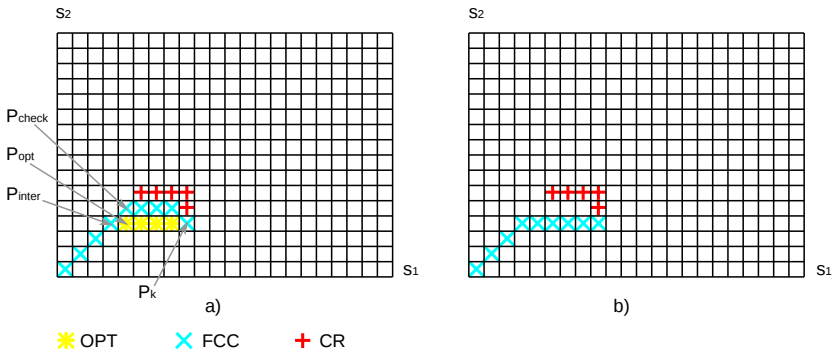


Figure 17: Elements involved in the optimization process in DCS. a) FCC without optimization; b) FCC already optimized.

whole path, but at least a set of next configurations) and that the robots must advance one step along FCC in a synchronized way, i.e. once the coordinate evolution along their paths has been fixed in the FCC they have to be able to follow their paths with the fixed time evolution.

7. Acknowledgment

This work was partially supported by the Spanish Government through projects DPI2013-40882-P and DPI2014-57757-R. The authors would like to thank to Carlos Rodriguez for his contributions in the implementation of the examples and the optimization procedure.

References

[1] A. Montaño, R. Suárez, An On-Line Coordination Algorithm for Multi-Robot Systems, in: 18th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA, 2013, pp. 0 – 6.

[2] S. S. Chiddarwar, N. Ramesh Babu, Conflict free coordinated path planning for multiple robots using a dynamic path modification sequence, Robot. Auton. Syst. 59 (7-8) (2011) 508–518.

[3] Y. Fei, D. Fuqiang, Z. Xifang, Collision-free motion planning of dual-arm reconfigurable robots, Robotics and Computer-Integrated Manufacturing 20 (4) (2004) 351 – 357.

[4] L. Gracia, A. Sala, F. Garelli, Robot coordination using task-priority and sliding-mode techniques, Robotics and Computer-Integrated Manufacturing 30 (1) (2014) 74 – 89.

[5] J.-C. Latombe, Robot motion planning, Kluwer Academic Publishers, 1991.

[6] G. Sanchez, J.-C. Latombe, Using a PRM planner to compare centralized and decoupled planning for multi-robot systems, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 2, 2002, pp. 2112 – 2119.

[7] M. Erdmann, T. Lozano-Perez, On multiple moving objects, in: IEEE Int. Conf. on Robotics and Automation, Vol. 3, 1986, pp. 1419–1424.

[8] K. NakYong, S. DongJin, R. Simmons, Collision-free motion coordination of heterogeneous robots, Journal of Mechanical Science and Technology 22 (11) (2008) 2090–2098.

[9] E. Todt, G. Rausch, R. Suárez, Analysis and classification of multiple robot coordination methods, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 4, 2000, pp. 3158–3163.

[10] P. O'Donnell, T. Lozano-Peréz, Deadlock-free and collision-free coordination of two robot manipulators, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 1, 1989, pp. 484 – 489.

[11] K. Shin, Z. Zheng, Minimum-time collision-free trajectory planning for dual-robot systems, IEEE J. Robotics and Automation 8 (5) (1992) 641–644.

[12] Z. Bien, J. Lee, A minimum-time trajectory planning method for two robots, IEEE Trans. on Robotics and Automation 8 (3) (1992) 414–418.

[13] C. Chang, M.-J. Chung, B. H. Lee, Collision avoidance of two general robot manipulators by minimum delay time, IEEE Trans. on Systems, Man and Cybernetics 24 (3) (1994) 517–522.

[14] S. Lee, H. Moradi, C. Y., A real-time dual-arm collision avoidance algorithm for assembly, in: Proc. IEEE Int. Symp. on Assembly and Task Planning, 1997, pp. 7–12.

[15] A. Mohri, M. Yamamoto, S. Marushima, Collision-free trajectory planning for two manipulators using virtual coordination space, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 2, 1993, pp. 674 –679.

[16] J. Lee, H. S. Nam, J. Lyou, A practical collision-free trajectory planning for two robot systems, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 3, 1995, pp. 2439–2444.

[17] X. Cheng, On-line collision-free path planning for service and assembly tasks by a two-arm robot, in: Proc. IEEE Int. Conf. Robotics and Automation, Vol. 2, 1995, pp. 1523 –1528.

[18] H. I. Bozma, M. Kalalolu, Multirobot coordination in pick-and-place tasks on a moving conveyor, Robotics and Computer-Integrated Manufacturing 28 (4) (2012) 530 – 538.

[19] Y. Shin, Z. Bien, Collisionfree trajectory planning for two robot arms, Robotica 7 (1989) 205–212.

[20] M. Quigley, B. Gekey, K. Cnley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, Ros: an open-source robot operating system, in: Workshop on Open Source Robotics in IEEE Intl. Conf. on Robotics and Automation, 2009, pp. 0 – 6.

[21] J. Blanchette, M. Summerfield, C++ GUI Programming with Qt 4, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[22] E. Larsen, S. Gottschalk, M. C. Lin, D. Manocha, Fast proximity queries with swept sphere volumes, in: Proc. of Int. Conf. on Robotics and Automation, 2000, pp. 3719–3726.

[23] J. Rosell, A. Pérez, A. Akbari, Muhayyuddin, L. Palomo, N. García, The Kautham Project: A teaching and research tool for robot motion planning, in: 19th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA, 2014, pp. 0 – 6.

[24] J. Rosell, M. Roa, A. Pérez, F. García, A general deterministic sequence for sampling d-dimensional configuration spaces, J. of Intelligent and Robotic Systems 50 (4) (2007) 361–373.

[25] J. Butterfass, M. Fischer, M. Grebenstein, S. Haidacher, G. Hirzinger, Design and experiences with DLR hand II, in: Proc. of the World Automation Congress, Vol. 15, 2004, pp. 105–110.

[26] Schunk, SCHUNK GmbH & Co. KG – Shunck Dexterous Hand - SDH2, Site: www.schunk.com (Sep 2011).

[27] J. Rosell, R. Suárez, C. Rosales, A. Pérez, Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures, Autonomous Robots 31 (2011) 87–102.

[28] IOC, On-line Coordination of two Staubli robots based on temporal synchronization. (2014). URL `https://goo.gl/y7JQi2`

[29] IOC, On-line Coordination of two UR5 robots based on temporal synchronization. (2016). URL `https://goo.gl/Sf8Fxr`

[30] C. Rodríguez, A. Montaño, R. Suárez, Optimization of robot coordination using temporal synchronization, in: 19th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA, 2014, pp. 0 – 6.