An On-Line Coordination Algorithm for Multi-Robot Systems

Andrés Montaño Raúl Suárez *

Abstract

This paper proposes a solution to the problem of coordinating multi-robot systems, which execute individually planned tasks in a shared workspace. The presented approach is a decoupled method that can coordinate the participants robots in on-line mode. The coordination is achieved through the adjustment of the time evolution of each robot along its original planned path according to the movements of the other robots to assure a collision free execution of their tasks. To assess the proposed approach a two robot system was used, and different tests were performed in graphical simulations as well as in real executions. Some examples are presented in the paper.

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 still an open problem of relevance in several robotic fields, both in industrial and service applications. Several approaches have been proposed to deal with this problem, as it is discussed in the next section. From a global point of view, we highlight here that the problem can be solved by simultaneously planning the trajectories of all the robots in the shared workspace, or by independently planning the trajectories of each robot and then applying an additional coordination phase (either off-line or on-line) to prevent potential collisions between them. The first approach is complete but it involves a higher number of Degrees Of Freedom (*dof*) and therefore it is computationally much more expensive than the second one, which is then considered from the practical point of view. From another point of view, 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 also to be done on-line.

*The authors are with the Institute of Industrial and Control Engineering (IOC) - Polytechnic University of Catalonia (UPC), Barcelona, Spain (andres.felipe.montano, raul.suarez@upc.edu). This work was partially supported by the Spanish Government through the projects DPI2010-15446 and DPI2011-22471.

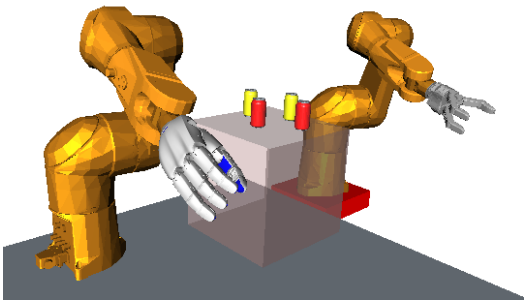


Figure 1. Two robots in a shared work space. Each robot must grasp and remove from the table cans of different color. The individually computed paths produce collision between the robots, consequently it is necessary a motion coordination in order to avoid them.

Dealing with this last problem, this paper proposes an approach to on-line coordinate two robot arms considering that they 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 guaranteed with respect to collisions with the other robot. To illustrate the 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 yellow cans. The motion planning is independently done for each robot (either because they are independent systems or 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 collision free executions.

The paper is organized as follows. Related work is presented in Section 2. Then, Section 3 formally introduces the proposed approach and Section 4 describes the particularization for the case of two robots. Experimental results are described in Section 5 and, finally, a summary and future work are given in Section 6.

2 Related Work

The approaches for multi-robot motion coordination are classified into centralized and decoupled approaches [8]. In the centralized approaches multiple robots operating in a shared workspace are considered as a single multi-bodied robot operating in a composite and multiple *dof* configuration space, and classical planning algorithms are applied to simultaneously find a collisions free path and coordinate 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 individual 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 applies coordination methods to avoid potential collisions when the robots are executing the movements simultaneously. Sanchez and Latombe [16] presented a comparative study between centralized and decoupled planning for multi-robot systems using a PRM planner, which concluded that in applications with a rather tight robot coordination the use of a centralized planner is more desirable. As discussed previously, centralized methods are not practical for on-line motion planning because they involve a large number of *dof*, and therefore a decoupled approach should be used.

An analysis and classification of multiple robot coordination methods was presented by Todt et al. [18], 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).

O'Donnell and Lozano-Peréz [12] addressed the motion coordination problem adding a precomputed time delay at the beginning of the movement execution that guarantees the collision avoidance between the robots. Lee et al. [9] and Yamamoto and Marushima [11] found an optimized coordination curve using dynamic programming. Their main goal is the minimization of the execution time of the tasks, considering the dynamics of the robots and the torque restrictions. The obtained coordination curve is used to design the velocity profile for each robot so that collisions are avoided. Cheng [3] introduced an adjustment in the geometric paths identifying the regions of the 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 one robot at a time. Lee et al. [10] proposed an event-based approach for on-line and off-line collision-free trajectory planning for dual-arm assembly systems based on a fast geometric collision detection algorithm. More recently, Chiddarwar and Babu [4] introduced a method that solves the robot conflicts based on a path modification approach. The conflictive paths are modified based on the robot positions in a dynamically computed path modification sequence.

In off-line approaches, as those mentioned above, 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 achieved because the complete robot plan is unknown and the computational time of the motion optimization is usually too large. The proposed approach works in on-line mode, searching to reduce the *dof* of the problem in order to minimize the computation time for the motion planning and the number of required collision checks for coordinate the robots.

3 Proposed Approach

This paper proposes a decentralized on-line motion coordination approach for multi-robot systems. Consider n robots R_i, $i \in \{1, ..., n\}$, in a shared workspace. In order to do its task each robot has an assigned geometric path $q_i(t)$ computed independently, where q_i is a configuration of R_i. For the coordination process we will adjust the evolution of the robot along the original planned path. In order to do this, we will express the path as a function of a parameter s_i that represents the travelling length along the path with $s_{i_{\max}}$ being the entire path length, and we will determine the evolution of s_i syncronized with the other robots (i.e. s_i can decrease along time, meaning that the robot is moving back along its geometric path). We use the following definitions:

Definition 1: Coordination space (CS) [17] of n robots is the space defined by the points $P = (s_1, ..., s_i, ..., s_n)$, with $0 \leq s_i \leq s_{i_{\max}}$.

Definition 2: Collision region (CR) is the set of points in CS representing collision configurations of the robots.

Definition 3: Discretized coordination space (DCS) is a discretized representation of the coordination space CS.

In the DCS the goal is to go from the origin point, $(0, ..., 0)$, to the end point, $(s_{1_{k_{\max}}}, ..., s_{n_{k_{\max}}})$, following a sequence of points $P_k = (s_{1_k}, ..., s_{i_k}, ..., s_{n_k})$, with $0 \leq s_{i_k} \leq s_{i_{k_{\max}}}$ without passing through the collision region CR. (Fig. 2 illustrates the DCS for two robots and the collision region CR).

Definition 4: Coordination curve (CC) is any continuous curve in CS describing a relative motion between the robots.

A CC may allow robots to move backward, which may be necessary for on-line collision avoidance [10]. A CC that does not pass through CR is called a collision-free coordination curve (FCC).

Definition 5: Motion direction (MD) is each of the possible movement direction in DCS. For n robots the number of possible MDs is $3^n - 1$.

To illustrate the MDs, consider the piece of DCS for two robots shown in Fig. 3. At each given point P_k there are eight different possible MDs to move to another point P_{k+1} in DCS. A horizontal or vertical MD in DCS is equivalent to stop one of the robots while moving the other. A diagonal MD indicates that both robots are

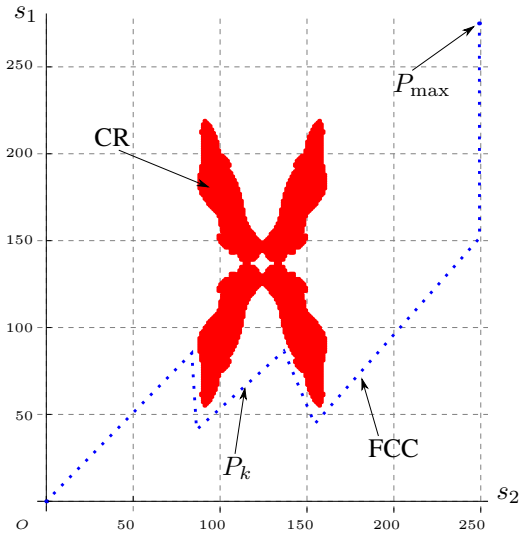


Figure 2. Discretized coordination space DCS and collision region CR for a particular case with two robots. In a real problem the collision region is not known a priori, but here, for illustrative purposes, it was computed making an exhaustive collision check over all the coordination space.

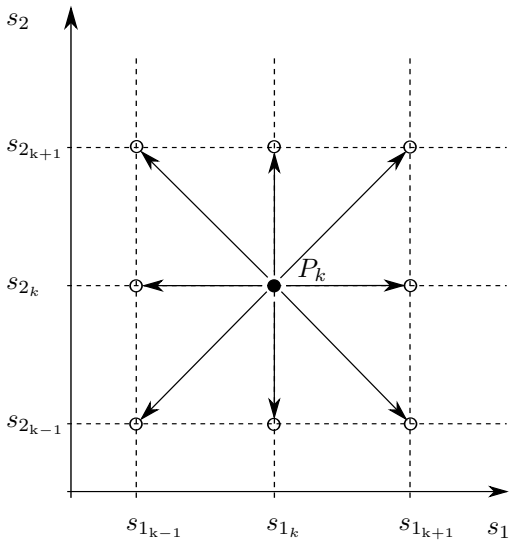


Figure 3. The eight possible motion directions MDs in a discretized coordination space DCS for two robots.

moved. The default desired motion direction is $(+1, +1)$ that moves the two robots to $P_{k+1} = (s_{1_{k+1}}, s_{2_{k+1}})$.

Now, the problem to be solved can be formulated as: "Given the geometric paths $q_i(s_{i_k})$ for n robots, find a FCC $\subset$ DCS from the origin to $P_{\max} = (s_{1_{k_{\max}}}, ..., s_{n_{k_{\max}}})$". When the problem is solved on-line $P_{\max}$ is not explicitly known a priori and the collision region CR $\subset$ DCS will be discovered and avoided on-line while the robots are moved along their paths.

Starting from any point in DCS, using a MD a new point of DCS is selected and a collision check is performed in order to detect whether it describes a potential collision configuration of the robots, i.e. whether it belongs to CR. The tested points that do not belong to CR are stored in a sequence describing a FCC for the robots. It is assumed that during each movement executed by robots (i.e a transition from one point P_k to P_{k+1}) it is possible to carry out at least two collision checks (i.e., verify the possible existence of collisions in two other points of DCS). This allows the generation of a FCC from the current robot position with a number of future points P_k that will grow when the tested point belongs to the free space of DCS and thus can be added to the FCC, and will decrease when the selected point belongs to the CR and cannot be added to the FCC.

The proposed approach is a decentralized method because each robot must compute the coordination algorithm to obtain its own trajectory in the physical space, i.e. the evolution in time of the predefined geometric path $q_i(s_{i_k})$. It is assumed that each robot has information about the next (possible few) movements of the other robots (it can be an on-line extrapolation of their already "seen" movements), 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 robots take consistent decisions.

Algorithm 1 shows the main procedure of the proposed approach, which must be executed for each robot R_i. It requires as input the set of the geometric paths of each robot R_i, Q $= \{q_i(s_{i_k}), i = 1, ..., n; k = 1, ..., s_{i_{k_{\max}}}\}$. The algorithm consists of two main actions, the planning of coordinated 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 implies moving them from a point P_k to P_{k+1}. 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 a collision configuration or not. The procedure used to select a new MD is discussed in detail in Section 4 for the case of two robots.

4 Case of Two Robots

The approach has been formulated above for n robots, and it was fully implemented for a cell with two robots.

Algorithm 1 Main

Require: Q

FCC$\leftarrow \emptyset$, MD$_k \leftarrow (+1, ..., +1)$, $P_k \leftarrow O$
while Task is not finished **do**
  for $i = 1$ **to** 2 **do**
    if $P_{k+1} \neq P_{\text{goal}}$ **then**
      Determine P_{k+1} using MD$_k$
      if P_{k+1} does not imply collision **then**
        Add P_{k+1} to FCC
        $P_k \leftarrow P_{k+1}$
      else
        Select a new MD$_k$ (using the state diagram)
      end if
    end if
  end for
  Move R_i from its current position to the next one according to FCC
end while

In this case, the DCS is a 2-dimensional space. For a given task each robot path $q_i(s_{i_k})$ is computed off-line, thus $s_{i_{k_{\max}}}$ is known, and the condition "Task is not finished" in Algorithm 1 can be formulated as $s_{i_k} < s_{i_{k_{\max}}}$. As mentioned above, the default desired motion direction MD is $(+1, +1)$, and the starting point in DCS is $P_k = (0, 0)$.

It is assumed that it is possible to execute two collision checks per cycle, i.e. check collisions in two points in DCS 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, a heuristic was implemented based on the wall follower, the best-known rule for traversing mazes, also known as either the left- or right-hand rule. A state diagram representation is used to explain the selection of the motion directions, where each state represents a motion direction MD.

The state diagram in Fig. 4 shows the wall follower heuristic with priority for the robot R_2. The diagram has eight states resulting from $3^n - 1$ for $n = 2$. At each state there is an ordered couple to indicate the move to be done by of each robot, $+1$ means that the robot moves forward one position, 0 means that the robot remains stopped, and -1 means that the robot moves backward one position. Depending on the result of this movement a transition to another state is done. 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 this conditions if there are collisions the transitions are counter-clockwise in the graphical representation of the state diagram; by analogy, if the priority is given to R_1 the transitions would be graphically clockwise (Fig. 5). In the state diagram with priority for R_2 (Fig. 4), when the state $(+1, 0)$ is reached, and the

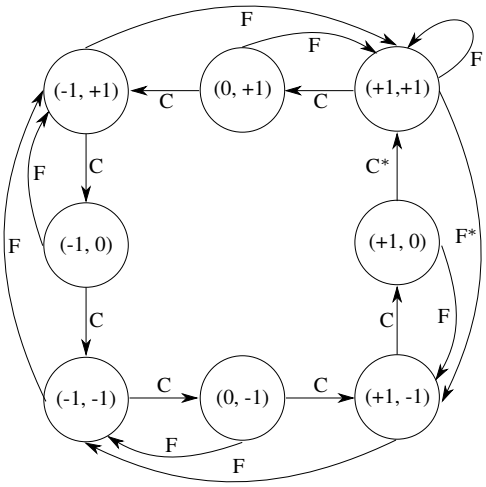


Figure 4. State diagram representing the wall follower heuristic with priority for the robot R_2. The transitions between states are marked with "C" when the resulting next point is a collision point and "F" when it is a collision free point.

destination is a collision point, a special condition must be considered in order to avoid a close 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 $(+1, +1)$ leads to a collision free point the next state is determined by "F*" instead of "F". This consideration is valid for the state diagram with priority for R_1 (Fig. 5) when state $(0, +1)$ is reached.

The robot priorities can be selected applying different criteria and can be changed if a solution is not found with the current selection. In the current implementation, the robot with the highest number of configurations in its planned path has the priority (i.e that with largest $s_{i_{\max}}$).

5 Experimental Results

The proposed approach has been fully implemented for the case of two robots. The code implementation is based on ROS [14] for the communications layer, Qt libraries [1] for the user interface, Coin3D [6] for the graphical rendering and PQP [7] for the collision detection. The path planning is computed using the home-developed path planning framework called *the Kautham Project* [13]. 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, metrics to evaluate the performance of planners (number of: generated samples, collision check callings, nodes in the graph solution and connected components) and simulation tools. For the graphical simulation the robots were modelled using triangular meshes.

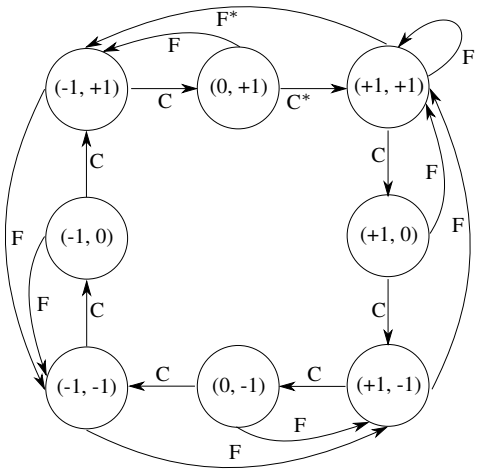


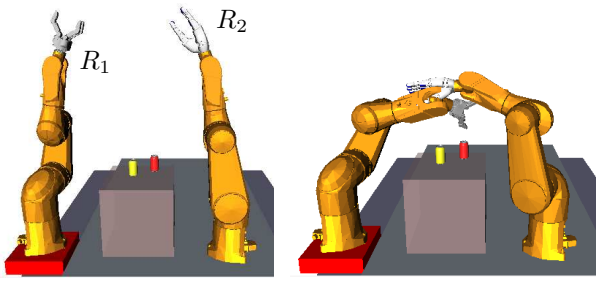
Figure 5. State diagram representing the wall follower heuristic with priority for the robot R_1.

The robots are two Stäubli TX-90 with 6 *dof* equipped with a Schunk Anthropomorphic Hand (SAH) [2] with 13 *dof*, and a Schunk Dexterous Hand (SDH2) [5] with 7 *dof*.

A PRM planner has been used to generate the geometric paths for each robot, with the samples generated in a cloud around the direct linear path from the initial to the final configuration. The planner also uses the concept of Principal Motion Directions (PMD) [15], which are directions in the working space resulting from a Principal Component Analysis (PCA), such that, properly ordered, the first PMD indicates the most representative direction of the robot workspace, the second PMD indicates the second most representative direction and so on. By choosing only a reduced number of the first PMDs the dimension of the search space can be significantly reduce keeping an acceptable approximation of the complete workspace. In our particular set up this was used to reduce the search space of the hand from 13 *dof* for the SAH and 6 *dof* for the SDH2, to only 2 in both cases. The discretization of the paths must be small enough in order to guarantee collision free movements between two configurations of the robots.

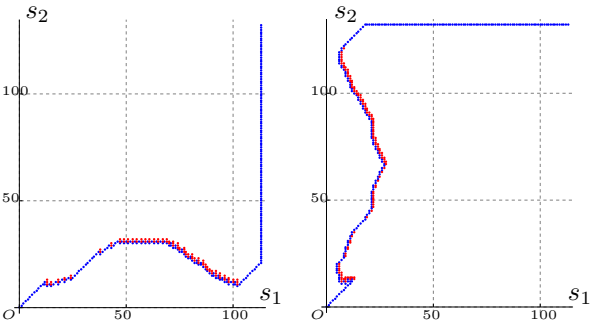
The synchronization of the robots is achieved applying event-based control, monitoring the current robot configurations and waiting until each robot reaches its desired configuration. A simple example of this event-based synchronization scheme is the following: when a robot R_i starts a movement from the current configuration $q_i(s_{i_k})$ toward the next one in the path $q_i(s_{i_{k+1}})$, a signal WAIT_i is activated, and it is active until R_i reaches $q_i(s_{i_{k+1}})$. In order to proceed to a new desired configuration $q_i(s_{i_{k+2}})$, all the signals WAIT_i from all robots must be off, for our case with two robots, WAIT_1 and WAIT_2 must be off to allow the robots proceed on their paths.

The following two examples illustrate the ability of the proposed approach to coordinate the independently computed paths for the robots. In the examples, the robots have to grasp and remove several cans that lie on a table.



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

Figure 6. 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.



(a) FCC with priority for R_1. (b) FCC with priority for R_2.

Figure 7. DCSs for the two robots in Fig. 6. FCC (in blue), CR (in red). a) Using priority for robot R_1; b) Using priority for robot R_2.

In the first example the robot R_1 is in charge of removing a red can, and the R_2 is in charge of remove a yellow one. Fig. 6 shows the setup for this example (Fig. 6a) and a snapshot where the robots are in collision during a task simulation without coordination (Fig. 6b). The computed robot paths have 133 and 114 configurations for R_1 and R_2 respectively. Fig. 7 shows the FCCs found using priority for robot R_1 (Fig. 7a) and priority for robot R_2 (Fig. 7b). In both executions the robot with priority completes the task before the other.

Fig. 8 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 426 configurations, and the path for R_2 has 270 configurations, thus the priority was given to R_1. The coordination process, i.e. the search of a FCC, required 728 collision checks, and the FCC was

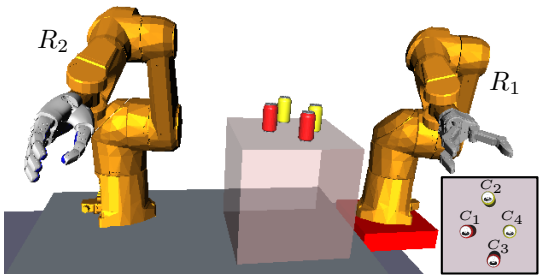
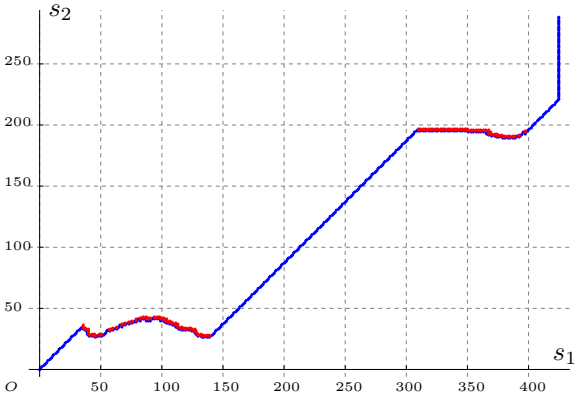
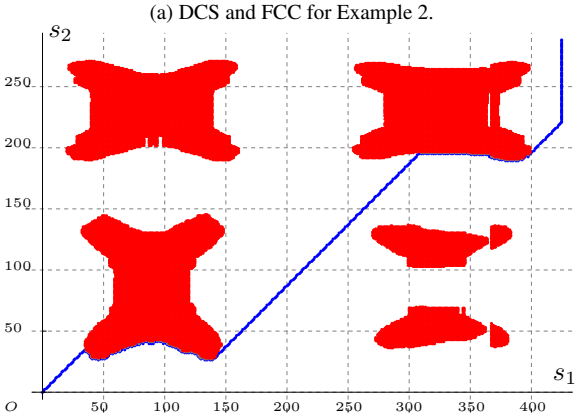


Figure 8. Setup for Example 2. The robot R_1 is in charge of removing the red cans C_1 and C_3, and R_2 is in charge of removing the yellow cans C_2 and C_4.



(a) DCS and FCC for Example 2.



(b) Complete CR.

Figure 9. DCSs for the two robots in Fig. 8. a) FCC (in blue) and explored CR (in red) using priority for robot R_1; b) FCC and complete CR computed for illustrative purposes.

fully determined when the robots were executing the move 364. The robot R_1 needed 440 moves in order to finish its task, meanwhile R_2 needed 506 moves. Fig. 9a shows the computed FCC (in blue) and the founded points which belong to CR (in red). Fig. 9b, for illustrative purposes, shows the FCC (in blue) and the complete CR (in red). In order to find the complete CR it was necessary to execute $s_{1_{k_{\max}}} \times s_{2_{k_{\max}}} = 115,020$ collision checks, a larger number than the 728 collision checks used in the coordination process. The real execution is illustrated in Fig. 10, where snapshots of the coordinated moves are shown.

6 Summary and Future Work

This paper has proposed a new on-line method for temporal coordination of multiple robots in a shared workspace with paths computed independently. The approach is based on the on-line exploration of the discretized coordination space (DCS) 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 and real executions, for a two-robot system.

As it was shown in the examples, the computation of the complete collision region CR in DCS in order to find a FCC is an expensive procedure, in terms of collision check tests, compared with the on-line search just checking the points determined by the motion directions MDs.

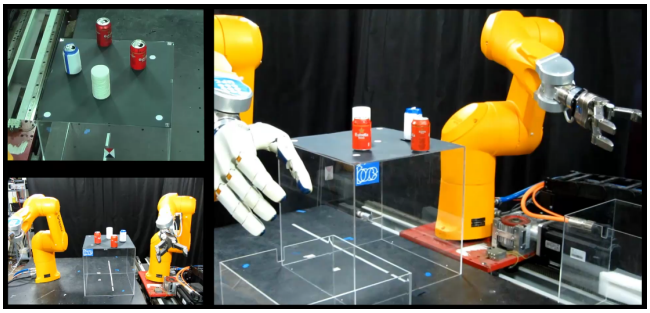
An extension of the implemented work is to develop new sequences of states used to select the MDs and to explore the DCS. Furthermore, a local optimization of the found FCC exploiting the difference between points added to FCC and the current point being executed by the robots could be performed. This difference grows when the explored sample belongs to the free space of DCS and it decreases when the sample belongs to CR, increasing or reducing, respectively, the available time for the optimization while the robots proceed with their tasks. This optimization could help to avoid the chattering due to the continuous move/wait actions used in the event-based control for the synchronization. Finally, another future work is the implementation of the proposed approach for more than two robots, which implies determining new state diagrams for the MDs with a larger number of states, which, as state above, is given by $3^n - 1$ and therefore grows exponentially with the number n of robots. The scalability of the approach to $n > 2$ robots depends on the capability of the system to perform the appropriate number of collision test during the robot movements.

References

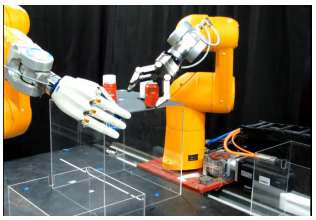
[1] J. Blanchette and M. Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[2] J. Butterfass, M. Fischer, M. Grebenstein, S. Haidacher, and G. Hirzinger. Design and experiences with DLR hand II. In *Proc. of the World Automation Congress*, volume 15, pages 105–110, 2004.

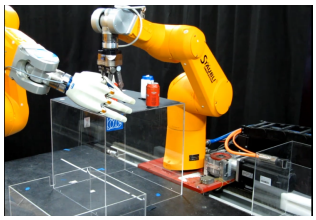
[3] X. Cheng. On-line collision-free path planning for service and assembly tasks by a two-arm robot. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 2, pages 1523 – 1528 vol.2, 1995.



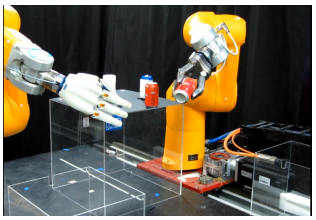
(a) Initial configuration.



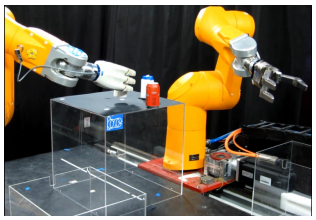
(b) R_1 and R_2 moving.



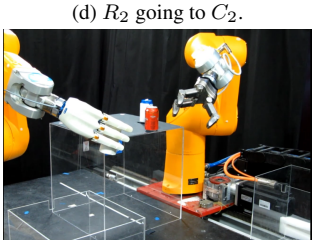
(c) R_1 taking C_1.



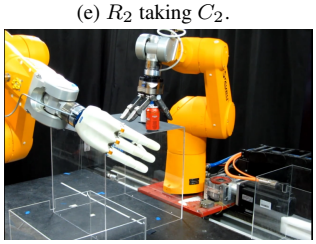
(d) R_2 going to C_2.



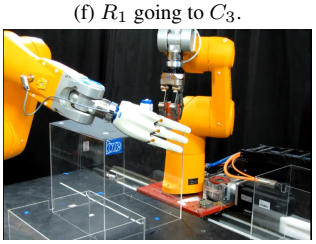
(e) R_2 taking C_2.



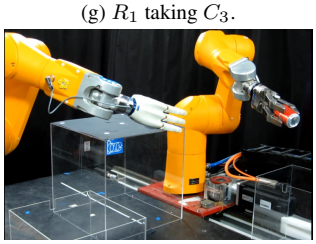
(f) R_1 going to C_3.



(g) R_1 taking C_3.



(h) R_1 taking C_3.



(i) R_2 taking C_4.

Figure 10. Snapshots of the real execution for the example 2.

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

[5] S. GmbH & Co. KG. Shunck Dexterous Hand - SDH2. www.schunk.com, Sep 2011.

[6] Kongsberg Oil & Gas Technologies. Coin3D - 3d graphics development tools. www.coin3d.org, December 2010.

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

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

[9] J. Lee, H. S. Nam, and J. Lyou. A practical collision-free trajectory planning for two robot systems. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 3, pages 2439–2444, 1995.

[10] S. Lee, H. Moradi, and C. Yi. A real-time dual-arm collision avoidance algorithm for assembly. In *Proc. IEEE Int. Symp. on Assembly and Task Planning*, pages 7–12, aug 1997.

[11] A. Mohri, M. Yamamoto, and S. Marushima. Collision-free trajectory planning for two manipulators using virtual coordination space. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 674 –679 vol.2, 1993.

[12] P. O'Donnell and T. Lozano-Peréz. Deadlock-free and collision-free coordination of two robot manipulators. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 484 –489 vol.1, 1989.

[13] A. Pérez and J. Rosell. A Roadmap to Robot Motion Planning Software Development. *Computer Applications in Engineering Education*, September 2009.

[14] M. Quigley, B. Gekey, K. Cnley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. *Workshop on Open Source Robotics in IEEE Intl. Conf. on Robotics and Automation (ICRA), Kobe, Japan*, May 2009.

[15] J. Rosell, R. Suárez, C. Rosales, and A. Pérez. Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures. *Autonomous Robots*, 31:87–102, 2011. 10.1007/s10514-011-9232-5.

[16] G. Sanchez and 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*, volume 2, pages 2112 – 2119, 2002.

[17] Y. Shin and Z. Bien. Collisionfree trajectory planning for two robot arms. *Robotica*, 7:205–212, 6 1989.

[18] E. Todt, G. Rausch, and R. Suárez. Analysis and classification of multiple robot coordination methods. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 4, pages 3158–3163, 2000.