Dual-Arm framework for cooperative applications

Carlos Rodríguez, Abiud Rojas-de-Silva and Raúl Suárez
Institute of Industrial and Control Engineering (IOC)
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

***Abstract*—This paper presents a framework schema that allows to bring simulation results obtained from different dual-arm robotic applications to executions in real environments. The framework uses the Robot Operating System (ROS) to communicate each component of the dual-arm, and the robotic software tool *The Kautham Project* for the computation of motion paths, inverse kinematics validations, collision checking and the graphical visualization of the simulated environment. The Anthropomorphic Dual Arm Robotic System (ADARS) was used in this work which is composed of two robotic arms of 6 degree of freedom, two anthropomorphic hands, and tactile sensors in the fingertips. The real execution of two different applications are presented to show the robustness of the framework. The resulting framework is general enough, allowing the reimplementation with minimum changes in any other dual-arm system.**

I. Introduction

Object manipulation using robotic systems is an activity that requires solving different types of problems, such as the computation of the grasp for a given object, the motion planning to move the robot from an initial to a goal configuration, and the computation of a path to move the object from its initial configuration to a new goal configuration, among others. In the particular case of the dual-arm systems, the complexity of the problems is higher, since a task-distribution planning and the coordination of each set hand-arm is required.

These issues has motivated to the robotics community to develop several approaches for multi-robot systems, focusing in the coordination of the system, and the task-distribution for cooperative tasks. Nevertheless, to translate the results of such approaches to real movements of the system, it is necessary to have an integration framework that allows the motion of the robotic system in real environments.

In this work we present a framework for synchronize and coordinate a dual-arm robotic system, in order to use it as a dual serial kinematic chain (e.g. move and manipulate objects in the workspace) or as a close kinematic chain (e.g. grasping bulky objects). The framework presented in this work uses the Robot Operating System (ROS) [1] for the communication layer of the dual-arm system.

After this introduction the paper is organized as follows. Section II presents a review of related works, Section III presents the hardware and software setup and Section IV presents the proposed framework schema. In Section V it is detailed the communication layer for framework for cooperative applications. Then, Section VI explains the dual-arm applications and section VII presents two real experimentations using the proposed framework. Finally, Section VIII summarizes the work and presents some topics deserving future work.

This work was partially supported by the Spanish Government through the projects DPI2013-40882-P and DPI2014-57757-R. A. Rojas-de-Silva was partially supported by the Mexican CONACyT doctoral grant 313768.

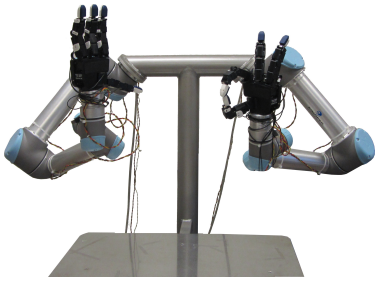


Fig. 1: ADARS: Athropomorphic Dual-Arm Robotic System.

II. Related Work

The introduction of dual-arm robotic systems to perform manipulation tasks has yielded a great variety of approaches, mainly focused to deal either with tasks that involve the manipulation of a single object using both arms [2], [3], or manipulation tasks of multiple objects using each arm in an independent way [4], [5].

Combining motion and task planning is still an open problem when objects are manipulated using each arm in independent way [6]. Nevertheless, different algorithms to grasp and manipulate objects in cluttered environments have been already proposed [7], [8], [9], and one of the applications presented in this work is in this line.

On the other hand, the problem of finding reachable grasps for bulky objects using two hands has become an important branch of research [2]. The problem has been tackled for 2D objects [10] and 3D objects using cost functions [11] and using motion planners with special features [12], [13], [3].

Despite most of the algorithms proposed for motion or grasp planning were applied in real experimentation to show their robustness, usually the framework used for their execution in the real physical system is not detailed. Examples of works dealing with this topic are related with the teleoperation of robotic systems [14], [15] and with the development of generic platforms for the low level integration of robotic components [16], [17].

In this work we present a framework to synchronize and coordinate robotic dual arm systems. It uses a communication layer based on ROS to transmit and synchronize the results of planning algorithms to the real system. We use two robotic applications to illustrate the approach performance. In the first application the arms work in a independent but cooperative

way to reach some desired objects, removing potential removable obstacles and properly distributing the tasks between the robot arms. The second application deals with the problem of grasping bulky objects using two anthropomorphic hands simultaneously.

III. Experimental setup

The hardware used in this work is the Anthropomorphic Dual Arm Robotic System (ADARS), whose components are:

- **Arms.** These are two robot manipulators UR5 from Universal Robots with 6 *DOF*, a payload capacity of 5 kg, a workspace radius of 850 mm and a repeatability of mbox+/- 0.1 mm.
- **Hands.** Two Allegro Hands (left and right) from Simlab are the end-effectors of the UR5, they are lightweight and anthropomorphic robotic hands with 4 fingers and 4 independent current-controlled joints per finge (16 *DOF*) and can hold up to 1.5 kg.
- **Tactile Sensors.** Each fingertip of the hands is equipped with a tactile sensor that has a sensing matrix of 4 x 8 sensing cells with a spatial resolution of 3.8 mm and works with a sample rate of 400 frames per second.
- **Simulation and planning software.** We use *The Kautham Project* [18], which is a home-developed open source environment that provides several tools to compute motion paths, perform collision checks, evaluate the performance of planners, among others.

IV. Framework Schema

The main goal of the proposed approach is to facilitate real experimentation with a dual-arm system, including, independent and cooperative tasks, and it is considered the possibility of performing an online coordination to avoid possible collisions between the robots.

The framework schema is presented in Figure 2, and its main modules are:

- **Applications.** This module hosts the planning algorithms to solve an application. The output of these algorithms is a set of robot joint configurations that defines the paths that will be followed by the dual-arm system.
- **Coordination.** This module coordinates the trajectories of each arm in order to avoid possible collisions during the task execution. It is used only when the arms execute independent tasks, otherwise it is assumed that the arms paths have been previously coordinated by the motion planner.
- **Synchronization.** This module synchronizes the movements of the components of the dual-arm system; it divides each global configuration into the corresponding parts of each component, i.e. the configuration corresponding to of each of the arms and hands.
- **The Kautham Project.** This module is used to accomplish the tasks assigned in each application, as well as to perform the online collision checking when the coordination module is active. Optionally, a graphical visualization of the task execution can be displayed during or prior to the real execution.
- **Controllers.** This module sends the corresponding joints values to the controller of each component (arms and hands) of the dual-arm system.

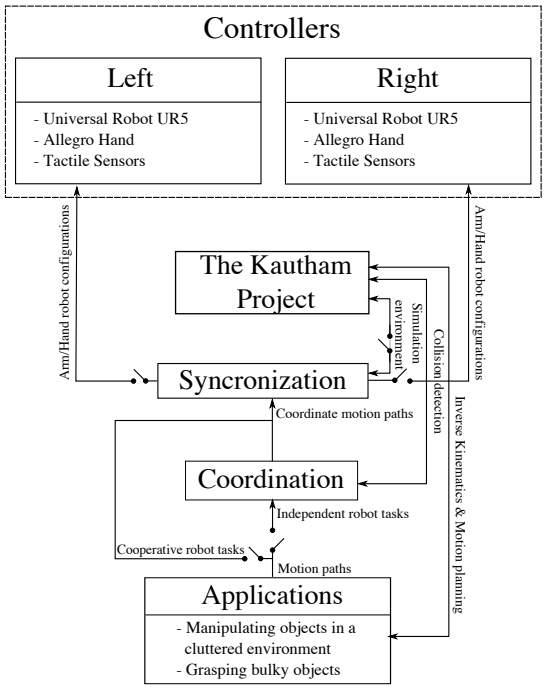


Fig. 2: Framework schema.

V. Communication Layer

The communication layer of the framework is based on the Robot Operating System (ROS), which allows the communication between the different processes involved in a robot application, based on nodes of a peer-to-peer networks. The communications between nodes are classified, according to how they exchange the information, as: a) Topics. Communication based on the "publish/subscribe" structure. Any node publishes information by sending a message to a given topic, and if another node needs this information it must be subscribed to the same topic to get the information. b) Services. Communication based on "client/server" structure. A client node sends a request to a server node and it awaits for the reply. c) Actions. Communication based on the client/server structure, but with the particularity that actions need a feedback about whether the request was properly delivered, in the case that this feedback is negative the next actions are not executed.

Figure 3 shows the diagram of the communication layer based on ROS. The ovals correspond to the nodes (described below), the rectangles with continuous line correspond to the topics, services, and actions, and the rectangles with the dashed line correspond to "namespaces", which are containers that allow to handle the same nodes and topics for different robots. In this case we have defined two namespaces, Left and Right, corresponding to each set hand-arm.

The communication layer works as follow. Node #1 is used for the applications (examples are presented in Seccion VI). Each application generates the paths to be followed by the dual-arm system, which are then sent by the node #1 to the node #2 for a graphical visualization of the task (including a simulated synchronization), and to node #3 to do the synchronization in the real execution.

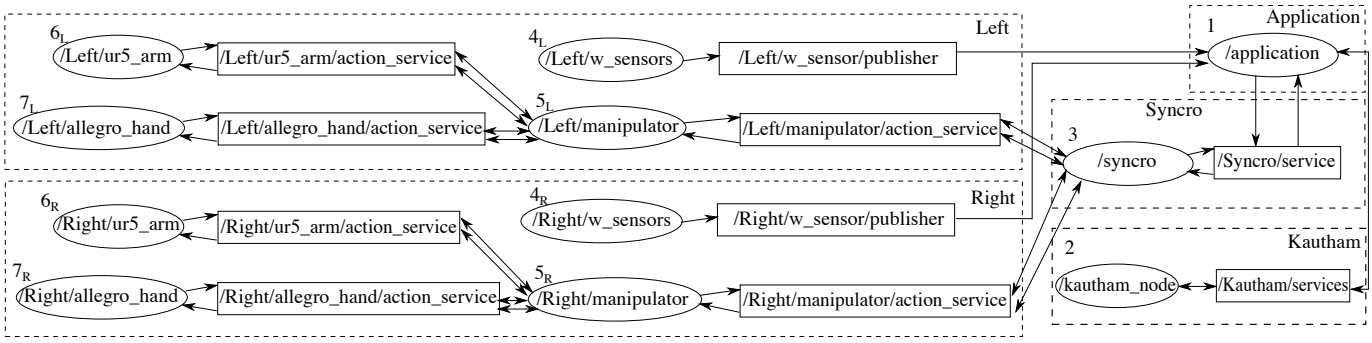


Fig. 3: Diagram of the communication layer based on ROS.

Node #2 is inside of the robotics software tool *The Kautham Project*, which allows the use of many of its tools by the following services.

- **Collision_check**. Detects collisions between: the arms and the environment, between the arms, and self-collisions of each arm.
- **Solve_query**. Receives the initial and goal configurations and generates a path between them. This service can use a variety of motion planners from the OMPL library.
- **Move_robots**. Moves the robots to a given configuration.
- **Coordinate_robots**. Receives a path for each set hand-arm and coordinates their motions in order to avoid the collisions between them. This service is used when each set hand-arm performs independent tasks in a shared workspace.
- **Simulation**. Simulates the real workspace. This service receives the model of the workspace and the paths for the dual-arm system. The simulation node can also be used independently of the framework.

Node #2 is used in three different processes. In the first one, the applications uses node #1 to call functions of *The Kautham Project* using the services provided by node #2 to solve inverse kinematics, find solution paths and perform collision checks, in the second process the node is used to perform the online coordination and in the last process the node is used to display graphically the execution of the task in a simulated environment.

Node #3 uses a service that receives from node #1 the paths of both robots as only one message, then it divides and assigns to each robot R_i the corresponding part of the message, with $i = L, R$, indicating Left and Right (the same subscript will be used to indicate nodes assigned to the Left and Right components). After that, each message is sent to node #5_i, where again is divided into the arm and hand configurations.

The node #4_i obtains the information about the state of the tactile sensors of R_i through a periodical request to a internal library, which, in case of contact, gives the coordinates of the contact point on the sensor pad and the applied contact force. Then, this information is published in a topic to be used by any application.

The node #6_i provides actions to control the UR5 arms. The arms controllers are connected through Ethernet using a low-level server/client software, that allows to calibrate the arms, send values of positions, velocities, and accelerations, and get the actual position of the arm.

The #7_i provides actions to control the Allegro Hands. The hands are controlled through a Controller Area Network (CAN) interface provided by the manufacturer. In this CAN interface we have included a position controller that, given the current and the desired position of each joint of the hand, computes the required torque to reach the desired position.

The actions nodes for arms and hands use three messages to exchange information: the request, the result, and the feedback. When a change in the configuration of the robot position is required, the nodes #6_i and #7_i generate the request message that contains the new desired configuration. This message is sent to the hand and arm controllers that internally execute the functions that move the robots and, when the movement of the robot finishes, a result message is returned to the nodes to inform whether the new configuration has been reached or not. The feedback message reports if the request message has been received, otherwise any other message if rejected.

VI. Dual-arm applications

The coordinated manipulation using dual-arm systems can be classified into goal-coordinated and bimanual manipulation [2]. In the first class, the robots work in the same task but without a direct interaction between them (e.g. place different objects in the same box). In the second class, both manipulators must interact with each other in order to accomplish a given task (e.g. lift a bulky object). The next sections present one application of each class to show how they are implemented with the proposed framework. The first application deals with object manipulation in cluttered environments and the second one with the bimanual grasp of bulky objects.

A. Manipulating objects in a cluttered environment

Consider a cluttered environment with fixed and removable objects. The goal of this application is to grasp two target objects with the dual-arm system. The objects can be grasped with either robot hand or can be assigned in advance, depending on the task to be performed. The objects of interest may be blocked by other removable objects, which therefore must be removed. It is assumed that environment is known, including the positions and the models of each object, a set of pre-grasp configurations of each object, and the initial configurations of the robots.

A detailed description of this application can be found in [4], which presents a variation of a probabilistic roadmap planner to compute the robot paths and uses a *precedence graph* G to represent the tasks to be executed by each robot arm. The root nodes of G represent the target objects and the rest of the nodes represent the removable objects (O_j, $j = 1, ..., n$), and the arcs represent the use of each robot arm R_i, $i = 1, 2$.

A motion planner is used to compute a path $P_{i,j}$ from the initial configuration of R_i to the grasp configuration of any object O_j, but, as a difference with the typical use of these planners, a sample of the robot configuration that implies a collision of the robot with removable objects O_k in the environment is not rejected; instead, these samples are added to the roadmap as valid samples and the objects O_k are stored in a set of removable obstacles $SO_{i,j}$. Then, a robot path $P_{i,j}$ is generated knowing the involved set of obstacles $SO_{i,j}$. The motion planner is used in a recursive way to find paths to remove each obstacle $O_k \in SO_{i,j}$ following a minimum cost strategy until finding collision-free paths that allows grasping the target objects. Note that the collision check performed for the configurations of a robot path must be done considering the arm and the hand when the robot is going toward the object to be grasped, and considering also the grasped object when this is removed from the environment. The computation of the motion paths as well as the collision checks are performed using the functions of The Kautham Project by means of the node #2, which, after receiving a request from node #1, provides access to the functions using the services **Solve_query** and **Collision_check**.

Finally, the paths of each arm must be executed by the dual-arm system, but since they are not coordinated, an online coordination method is run in order to avoid collisions between each set hand-arm robot during the real execution [19]. This coordination is performed with the service **Coordinate_robots** provided by the node #2.

Figure 4(a) shows a 2D hypothetical example, a cluttered environment composed by square objects that represent the removable objects O_j in front of the dual-arm system represented by the robot arms R_1 and R_2. The blue (light gray) squares represent the target objects O_1 and O_2 and the red (darker gray) squares the obstacles. Figure 4(b) shows the resulting graph G to grasp O_1 and O_2 with each R_i: to grasp O_1 with R_1 resulted $SO_{1,1} = \{O_4\}$ and for R_2 resulted $SO_{2,1} = \{O_5, O_7\}$. The branch of the precedence graph with the lower number of obstacles is chosen, in this case the branch that contains O_4. In order to remove O_4 collision-free paths were found for both robot arms (i.e. $SO_{1,4} = \emptyset$, $SO_{2,4} = \emptyset$. On the other hand, to grasp O_2 with R_1 resulted $SO_{1,2} = \{O_6\}$ and for R_2 resulted $SO_{2,2} = \{O_4\}$, since O_4 has a collision-free path this branch is chosen. Figure 4(c) shows the resulting sequence of actions for this example.

B. Grasping bulky objects

Most of the applications of dual-arm systems are focused on the manipulation and regrasping of small objects and only a few of them are focused on the problem of determining grasps of bulky objects using two hands. We have developed an application that deals with this problem.

The object surfaces are represented by point clouds, and only precision grasp with frictional contacts are considered, using 3 fingers per hand (6 contacts in total). The goal of the application is finding three reachable contact points for each hand. First, the point cloud (PC) that describes the object surface is divided into two sets of slices SS_h, with $h = L, R$ representing the right and left hand respectively. This is done considering equidistant planes orthogonal to the main inertial axis of the object, the points lying between two consecutive planes belong to the same slice (see Fig. 5).

In each slice S_i^h a set of triplets $ST_i^h = \{T_{i,j}^h, j = 1, ..., n\}$ is computed, with each triplet composed of three points (each one with the associate normal direction), $T_{i,j}^h = (p_1, p_2, p_3)$. The triplet selection is based on three conditions: a) The area of the triangle formed by the three points of the triplet must be smaller than the maximum reachable area of the triangle formed by the center point of the three fingertips; b) The distance between the points must be smaller than the maximum allowed distance between the fingertips; c) An index indicating how close the triangle defined by the three contact points is to be an equilateral one, which must be above a given threshold. The index is equal one for equilateral triangles and 0 for degenerate triangle).

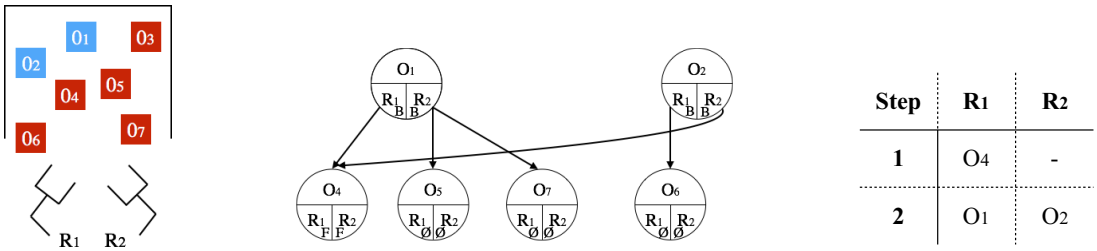
Then each triplet of ST_i^R is tested together with each triplet of ST_i^L in order to find a couple of triplets $G = (T_{i,j}^R, T_{i,j}^L)$ that satisfies the force-closure (FC) condition with a quality above a given threshold.

The grasp quality is evaluated as the largest perturbation that a grasp can resist in any direction [20]. Finally, a reachability analysis for the couple of triplets is performed by computing the inverse kinematics of the whole system, if there is a kinematic solution a motion planing and a collisions check are performed to ensure that the configuration of the whole system is suitable. The motion planning and collision-cheking are performed using the functions of The Kautham Project by means of the node #2, which, after receiving a request from node #1, provides access to the functions using the services **Solve_query** and **Collision_check**. In this case, the paths of both arms are computed by the planner considering them as only one system, therefore the online coordination is not required.

VII. Experimentation

The first experiment presented below deals with the manipulation of objects in cluttered environments. Several cylinders lie on a table in front of the dual-arm system. The target objects are the cylinders O_1 and O_2 and there are a set of removable objects (cylinders $O_j, j \geq 3$) that may act as obstacles that block the direct access to the target objects. The goal is that each robot arm grasps a target cylinder. A set of 35 pre-grasp hand configurations has been computed in advance to wrap the object around the Z axis (i.e. grasping from the top side is not allowed). Figure 6(a) shows the setup of the workspace used for the experiment. The resulting precedence graph is shown in Figure 6(b).

The result of the experiment was as follows. In order to grasp O_1 a motion path colliding with O_2 and O_6 was found for R_1, and a motion path colliding with O_4 was found for R_2. Then, a collision-free path was found to remove O_4 with



Step	R1	R2
1	O4	-
2	O1	O2

Fig. 4: Conceptual 2D example: a) 2D workspace, b) Resulting precedence graph G, c) Resulting sequence of actions.

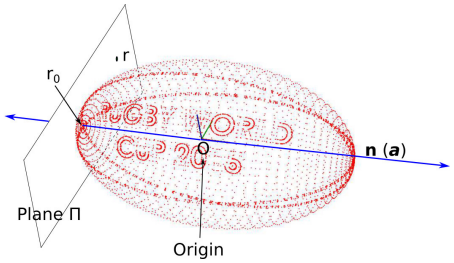


Fig. 5: a) Rugby ball and a plane orthogonal to the main inertial axis used to compute the slices.

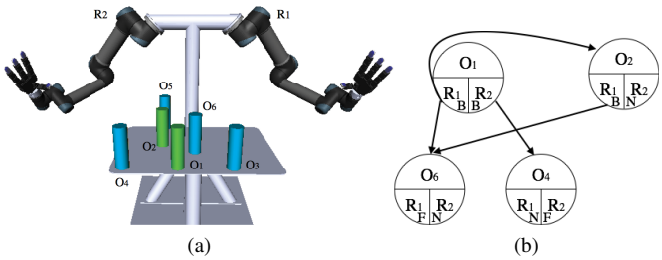


Fig. 6: Experiment 1, (a) Setup of the workspace, (b)The resulting manipulation task graph.

R_2. On the other hand, in order to grasp O_2 with R_1 a path that collides with O_6 was found, and in order to remove O_6 a collision-free path was found for R_1 while R_2 cannot reach O_6 because the path has a collision with O_2, generating a loop in G. Figure 7(a) shows the resulting sequence of actions, and Figures 7(b-d) show snapshots of the real execution of experiment 1.

The second experiment deals with the grasp of 3 bulky objects: a helmet, a rugby ball and a detergent bottle. It was considered a friction coefficient of 0.4 (the range of mechanizable plastics is between 0.08 and 0.6), and a FC grasp is considered acceptable if its quality is higher than 0.03 using the quality measure defined in [20].

30 trials per objects were performed, obtaining grasps with a quality higher than 0.03 in the 92.22% of the cases, from which 86.5% has inverse kinematics solution and, from these, 88.4% has a free-collision path solution. In the real experimentation the grasping successful rate was 65%.

Figure 8(a) shows different sets of contact points of the three objects used in this experiment and Figures 8(b-d) show snapshots of the real execution of experiment 2.

VIII. Summary

The paper has presented a framework for the execution of cooperative actions using dual-arm robotic systems, including a detailed description of the framework's modules and how they are interconnected using ROS. The approach has been implemented in a real dual-arm systems composed by two UR5 robot arms and two anthropomorphic Allegro hands equipped with tactile sensors in the fingertips. Two different real applications were used to show the use of the framework. Future work includes the integration of new modules, for instance, a perception module using cameras and a grasping module able to deal with uncertainty.

References

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2009.

[2] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation - A survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[3] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous Grasp and Motion Planning," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 43–57, 2012.

[4] C. Rodríguez, A. Montaño, and R. Suárez, "Planning manipulation movements of a dual-arm system considering obstacle removing," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1816 – 1826, 2014.

[5] R. Suarez, J. Rosell, and N. Garcia, "Using synergies in dual-arm manipulation tasks," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2015, pp. 5655–5661.

[6] K. Hauser and J. C. Latombe, "Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries," in *ICAPS Workshop on Bridging the Gap between Task and Motion Planning*, 2009.

[7] M. Stilman, J.-u. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2007.

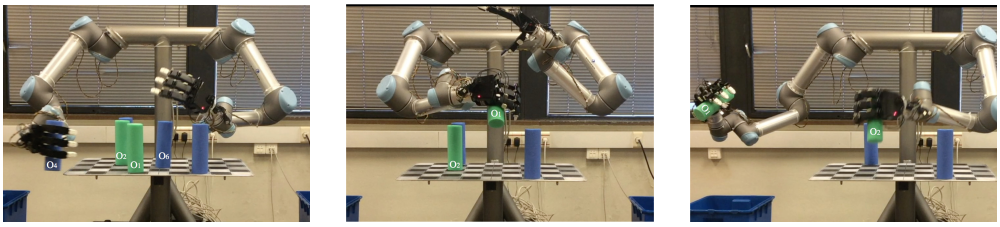
[8] M. Dogar and S. Srinivasa.., "A framework for push-grasping in clutter," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[9] L. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2011, pp. 1470–1477.

[10] J. Caraza and X. Yun, "Two-handed grasping with two-fingered hands," in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, June 1991, pp. 597–602 vol.1.

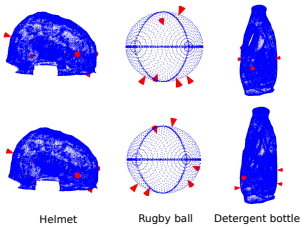
Step	R_1	R_2
1	O_6	O_4
2	O_2	O_1

(a)

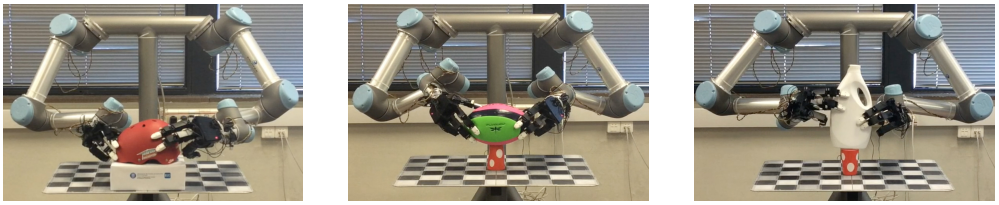


(b) (c) (d)

Fig. 7: Experiment 2: a) Resulting sequence of actions, b) R_1 going to grasp O_6 and R_2 grasping O_4, c) R_1 waiting to grasp O_2 and R_2 moving O_1, d) R_1 moving O_2 and R_2 in the final configuration.



(a)



(b) (c) (d)

Fig. 8: Experiment 2: a) sets of contact points, b) Grasping a helmet, c) Grasping a rugby ball, d) Grasping a detergent bottle.

[11] D. Berenson and S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, 2008.

[12] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, oct. 2009, pp. 2464–2470.

[13] N. Vahrenkamp, M. Przybylski, T. Asfour, and R. Dillmann, "Bimanual grasp planning." in *Humanoids*. IEEE, 2011, pp. 493–499.

[14] J. Rosell, R. Suarez, and A. Pérez., "Safe teleoperation of a dual hand-arm robotic system," in *Robot2013: First Iberian Robotics Conference, Advances in Intelligent Systems and Computing*, Springer, Ed., 2014.

[15] D. Kruse, J. T. Wen, and R. J. Radke, "A sensor-based dual-arm tele-robotic system," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, p. 4.18, 2015.

[16] H. M. Do, T. Y. Choi, D. I. Park, G. J. Chung, and J. H. Kyung, "Design of software framework for system integration of dual-arm robot," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, 2012.

[17] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, "The robot software framework armarx," *Information Technology*, vol. 57, pp. 99–111, 2015.

[18] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García, "The kautham project: A teaching and research tool for robot motion planning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA'14*, 2014. [Online]. Available: sir.upc.edu/kautham

[19] A. Montaño and R. Suárez, "An On-Line Coordination Algorithm for Multi-Robot Systems," in *18th Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA*, September 2013.

[20] C. Ferrari and J. Cany, "Planning optimal grasps," in *IEEE International Conference on Robotics and Automation*, 1992.