A case study of automated dual-arm manipulation in industrial applications

Yoann Solana, Hector Herrero Cueva, Alvaro Rubio Garcia, Sergio Martinez Calvo, Urko Esnaola Campos, Damien Sallé, Juan Cortés

A case study of automated dual-arm manipulation in industrial applications

Yoann Solana
LAAS-CNRS
Université de Toulouse, CNRS
Toulouse, France

Hector Herrero Cueva
Tecnalia Research and Innovation
Industry and Transport Division
San Sebastian, Spain

Alvaro Rubio García
AIRBUS Operations
Puerto Real, Spain

Sergio Martínez Calvo
AIRBUS Operations
Puerto Real, Spain
sergio.martinez-calvo@airbus.com

Urko Esnaola Campos
Tecnalia Research and Innovation
Industry and Transport Division
San Sebastian, Spain
urko.esnaola@tecnalia.com

Damien Sallé
Tecnalia Research and Innovation
Industry and Transport Division
San Sebastian, Spain

Juan Cortés
LAAS-CNRS
Université de Toulouse, CNRS
Toulouse, France
juan.cortes@laas.fr

***Abstract*—Nowadays, factories are required to increase production flexibility in order to manufacture small-lot variants, rapidly adapting to customer demands. Furthermore, manufacturing may involve complex manipulation tasks, usually performed by human workers. In such a context, traditional robotic systems are not competitive due to the huge costs of installation, maintenance and adaptation. A new generation of robots, equipped with multiple arms, is appearing as an attractive alternative because of their potential versatility and ability to execute intricate manipulation tasks. To facilitate the integration of these robots in a work-cell and a rapid adaptation to different tasks, easy-to-use programming interfaces and a high degree of autonomy are mandatory. Autonomous task and motion planning are particularly relevant in this context. In this paper, we present our recent progress in this direction. Hardware and software developments are explained in the context of a pilot dual-arm robot station that is being integrated in the production line of a big airplane manufacturer. First experimental results are also presented.**

***Index Terms*—Dual-arm manipulation, coordinated manipulation, motion planning, flexible task programming software.**

Fig. 1: The Nextage robot is manipulating a rib with both arms to place it on the deburring support.

I. Introduction

The recent trends of mass customization of products and lean approaches impact production by a drastic reduction of lot sizes. In this context, traditional automation and robotics fail to be competitive since all individual product variants would require a new complete automation project. In addition, keeping up with the introduction of robots outside of the traditional sectors requires to automate much more complex manipulation tasks, where traditional robotics fails to provide a good ratio of cost vs. robustness, mainly due to the rigidity of existing production equipment in terms of programming and tools. There is thus a great opportunity in the manufacturing assembly sector (aeronautics, car industry, electronics, SMEs...) for dual-arm robots, able to execute complex manipulation/assembly tasks that are traditionally assigned to humans. Many projects and developments are targeting this objective, most of which involve academic and industrial partners. For instance: THOMAS [1] focuses on creating a dynamically reconfigurable shop floor with mobile dual-arm robots; VERSATILE [2], in turn, applies dual-arm capabilities in executing complex tasks that are traditionally assigned to humans due to their manipulation requirements; and SHERLOCK [3], which focuses on human-robot collaboration, and specifically, co-manipulation of large parts by a dual-arm mobile manipulator.

Several examples of dual-arm robots have been showcased in recent years. However, apart from a very few examples (such as those exhibited at the DARPA robotics challenge [1]), most applications demonstrate high-level sequence-coordinated individual motions of each arm. To move from coordinated individual arm motions to complex dual-arm motions, one must solve planning and control problems for

[1]THOMAS : http://www.thomas-project.eu/
[2]VERSATILE : https://versatile-project.eu/
[3]SHERLOCK : http://www.sherlock-project.eu/home/

the closed kinematic chain formed by the two arms and the manipulated part. Some interesting approaches have been developed at a lab scale (e.g. [2], [3], [4]), but have not yet been transferred to or tested in industrial pilot stations.

To make the jump from lab scale to industry, flexibility is mandatory, in addition to robustness. Moreover, it is very important to facilitate the use of the system by moderately qualified operators. Easiness of use is therefore an important aspect, even more when one wants to adopt the paradigm of Industry 4.0 [5], where all the systems in a factory are connected and must provide the tools to flexibly adapt to the continuous changes in production.

This paper presents ongoing work on the development of a software platform to facilitate the integration of versatile dual-arm robots in the manufacturing industry. The developments are illustrated in a real use-case: a pilot work-cell located at the AIRBUS factory in Puerto Real (Spain), where a dual-arm robot is being deployed to debur parts of an AIRBUS A380 aircraft. This use-case, together with the specific hardware, are presented in Section II. Then, Section III presents the software architecture being used. In parallel to the work presented in this paper, we are developing several approaches (skill-based programming, CAD-based programming, ...) to enable moderately qualified operators to easily specify a manipulation task [6]. Here, we focus on dual-arm motion planning, which is one of the key issues to be solved in the context of this case study. Motion planning is required to automatically generate trajectories of the robot to perform manipulation tasks while satisfying intrinsic constrains of the system as well as collision avoidance with obstacles in the workspace. Section IV describes the motion planning problems that need to be solved in the present use-case, and explains the adopted state-of-the-art methods. Details about the implementation of these motion planning methods in our software platform are provided in Section V. Results presented in Section VI show the performance of the implemented solution. Finally, Section VII presents conclusions and directions for future work.

II. Use-case and Hardware

A. The scenario and its technical particularity

The pilot case presented in this paper is a work-cell located at AIRBUS Puerto Real. This work-cell is being developed for a dual-arm robot to work instead of a human for deburring the 120 different ribs of the vertical and horizontal tail planes of an AIRBUS A380 aircraft. The possibility to replace human workers by robots in this context is important due to the potential toxicity of the task.

Using a Nextage dual-arm robot as the one shown in Fig. 1 (see Section II-D for explanations about the robot), the smallest ribs can be manipulated with one arm while using the other arm for the deburring task. However, 35% of the ribs cannot be processed in this way because of their larger size and weight. Coordinated dual-arm capabilities are required in this case to move the ribs using both arms. This technological enhancement, unavailable in the software packages used as a basis for this work, had to be implemented for a full-scale validation before production deployment.

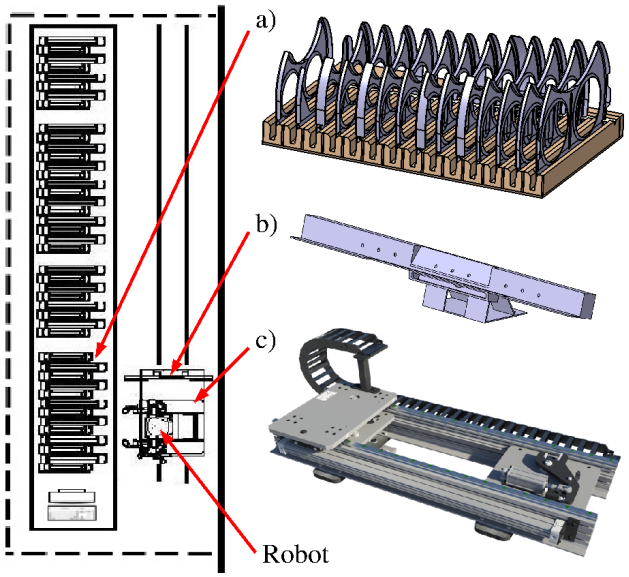


Fig. 2: Lay-out and main elements of the work-cell: a) rib container rack; b) deburring support; c) lineal track.

B. Cell design

Different parts were designed for the work-cell:

(a) Rib container rack: This rack (Fig. 2.a) allows to store the ribs in an accurate position, easy to reach from both sides by the robot and by a human operator. The robot will be able to reposition each rib using one arm, by sliding motion of the rib on the rack, if needed previously to grasping it with the two arms.

(b) Deburring support: This support allows the robot to place each rib on it, and to debur its holes from both sides. The robot slides the part along the support, using one arm, to make the holes reachable for the deburring tool mounted on the other end-effector.

(c) Lineal track: The lineal track (Fig. 2.c), on which the robot is mounted, enables the robot to move alongside the container rack to reach all the ribs.

C. Lay-out

The integration of all the elements of the working cell can be seen in the lay-out represented on the left side of Fig. 2. The operator will only have to go inside the work-cell to get processed ribs from the rack and to furnish new ribs to be deburred, the rest of the process being fully automated.

D. The robot

The Kawada Nextage Open robot[4] (Fig. 1) was chosen for the pilot work-cell described in this paper. It has two arms with 6 degrees of freedom (dof) each attached to a rotary torso (1 dof) and a stereo-vision-equipped head with 2 dof; 15 dof altogether managed by a single controller.

[4]Kawada Nextage : http://nextage.kawada.jp/en/

For the application described in this work, the robot was equipped with specific grippers and an RGB-D camera for workspace monitoring. Combined with the CAD models of the work-cell, the data provided by the RGB-D camera is used to build a virtual representation of the workspace, which is used for motion planning as explained below. The left arm has a pneumatic gripper designed for grasping carbon fiber parts. The right arm has a multipurpose gripper that incorporates: (i) a suction cup for holding the parts; (ii) a deburring tool; and (iii) a pair of cameras to detect the holes of the carbon fiber parts to enable a high-precision deburring operation.

III. Software Architecture

The software architecture was developed to allow full control of the executed task. In is based on the Robot Operating System (ROS)[5] [7]. The robot is connected to ROS through a bridge developed by JSK (Jouhou System Kougaku) Laboratory at The University of Tokyo[6], in collaboration with Tokyo Opensource Robotics Kyokai Association (TORK)[7] and Tecnalia[8]. OpenRTM[9], developed by AIST, was used as a middleware. This combination of components allows using all ROS capabilities. At this point, it should be emphasized that, thanks to the use of ROS, the software architecture is hardware vendor independent: different robotic hardwares can be used through appropriate interfaces between ROS and the robot controller.

In addition, the proposed software architecture is aimed to facilitate the implementation of new applications by increasing the re-usability of the developed modules. Through an easy-to-use graphical user interface (see [6] for additional details on the software architecture and interface), operators are able to create, modify, reuse and maintain industrial processes, thus increasing the flexibility of the cell.

As shown in the Fig. 3, the architecture is composed of three layers, namely: (i) application development layer, which allows to create, load and modify processes; (ii) execution engine layer, to execute a broad portfolio of primitives and skills; and (iii) robotic interface layer, providing the necessary interfaces and controllers for different robots [6]. Regarding the aforementioned reusability and flexibility, the application development layer allows the selection of previously developed operations. These operations are the mentioned skills and range from generic tasks such as dual-arm pick and place to more specific tasks as dual-arm deburring. These skills can be reused in similar processes, requiring a configuration to which the operators are familiar.

Among the ROS packages and tools, MoveIt![10] was mostly used in this work for motion planning. The extension of the

[5]Robot Operating System (ROS) : http://www.ros.org/

[6]Jouhou System Kougaku Laboratory, Tokyo University : http://www.jsk.t.u-tokyo.ac.jp/

[7]Tokyo Opensource Robotics Kyokai Association (TORK) : http://opensource-robotics.tokyo.jp/

[8]Tecnalia : http://www.tecnalia.com/en/

[9]OpenRTM-AIST middleware : http://openrtm.org/

[10]MoveIt! : http://moveit.ros.org/

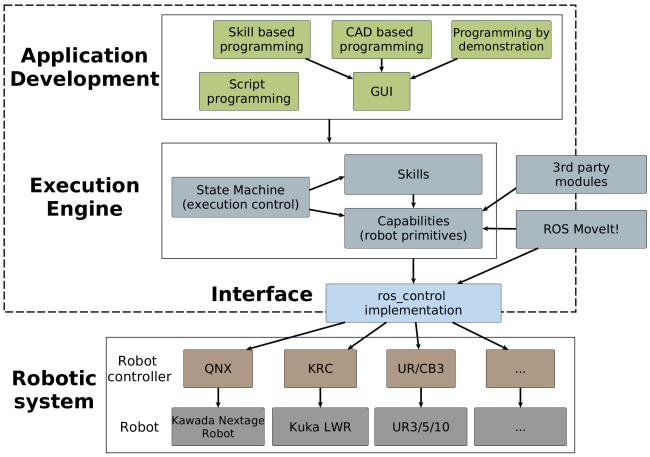


Fig. 3: Three-layer-based software architecture: application development through diverse approaches, execution engine for task execution and capability managment, and interface with the actual robotic systems.

MoveIt! framework to enable dual-arm manipulation is described in SectionV. The point-cloud obtained from the RGB-D camera was integrated into MoveIt! using the Octomap plugin. Thanks to this, the 3D occupancy grid implemented in the Octomap library can be added to the MoveIt! Planning Scene, and thus taken into account for motion planning in addition to the static objects directly considered in the model.

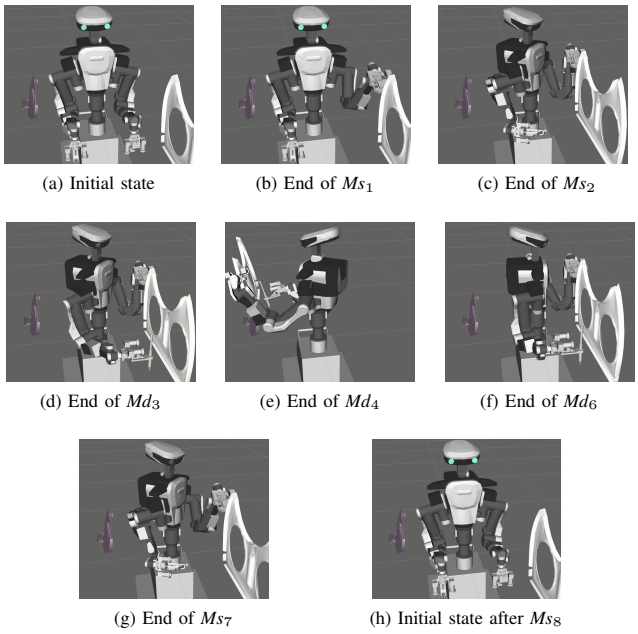
IV. Dual-arm Motion Planning

A. Problem Formulation

The whole deburring process of a part requires different types of motions of the robot, involving one or two arms:

Ms_1 : pick the part on the rack with the left arm.
Ms_2 : extract the part from the rack (sliding motion) using the left arm and the torso.
Md_3 : pick the part with the right arm.
Md_4 : place the part on the deburring support using both arms.
M_5 : a sequence of motions, involving mainly the right arm, to perform the deburring process.
Md_6 : place the part in front of the rack using both arms.
Ms_7 : put the part back on the rack (sliding motion) using the left arm and the torso.
Ms_8 : ungrasp and move back to the initial configuration.

Fig. 4 shows some snapshots corresponding to intermediate states that illustrate these elementary motions. Each motion (with the exception of M_5, which requires an ad-hoc vision-based technique not described here) is obtained by solving a planning problem for a given start q_s and goal q_g states of the robot, and considering the current state of the objects in the workspace. In other words, a motion planning algorithm is required to find a path between each pair of intermediate configurations satisfying feasibility constraints, in particular collision avoidance. A more formal definition of the motion planning problem can be found in related literature, such as [8], [9]. Note that, when one part is being manipulated, its



(a) Initial state (b) End of Ms_1 (c) End of Ms_2

(d) End of Md_3 (e) End of Md_4 (f) End of Md_6

(g) End of Ms_7 (h) Initial state after Ms_8

Fig. 4: Snapshots of the motions described in Section IV-A

pose is directly determined from the configuration of the robot, and collisions have to be avoided as for the robot's bodies. It is also important to mention that, in the coordinated dual-arm manipulation case, feasible configurations of the robot must satisfy a kinematic loop-closure constraint induced by the object grasped with the two end effectors (E_i). The kinematic loop-closure constraint is defined by a fixed transformation ${}^{E_1}T_{E_2}$, which can be determined from the grasping points in the manipulated object.

In the current version of the software, the initial and final configurations of each elementary motion are defined by the user via the application development layer. The user can also provide additional information to facilitate the manipulation of the part, such as specifications about start/approach/retreat translations for grasping/ungrasping. An automatic decomposition of the overall manipulation task into the sequence of elementary motions is an interesting direction for future work, as will be mentioned in Section VII.

B. Methods

The motion planning problems to be solved in the present context are complex due to the cluttered industrial environment and the high dimensionality of the configuration space (up to 13 for both arms and the torso of the Nextage robot). Among numerous types of planners, sampling-based algorithms have been proved to be efficient methods to solve such complex problems [10], [9]. Several of these planners, such as the Probabilistic Roadmap (PRM) [11], the Rapidly-exploring Random Tree (RRT) [12], and their variants, are commonly used in motion planning software. In particular, these algorithms are implemented in OMPL[11] [13], a motion planning library used within the MoveIt! framework.

For planning single-arm motions, basic versions of the algorithms implemented in OMPL can be directly applied. However, dual-arm motions require dealing with kinematic loop-closure constraints, which must be considered at the motion planning level, significantly increasing the difficulty of the problem. Several approaches have been proposed for extending sampling-based planners to treat this type of constraints. Some of them apply numerical methods to project sampled configurations on the constraint manifold [14], [15], [16], whereas others are based on a kinematic decomposition of the mobile system and the application of closed-form (analytical) inverse kinematics solvers [17], [18]. It has been shown that kinematics-based decomposition approaches are preferable in environments cluttered with obstacles [19]. Therefore, in this work, we implemented state-of-the-art methods of this latter class described in previous work [18], [20]. Note however that other approaches for motion planning under kinematic loop-closure constraints could be used with minor changes in the code. The main idea behind this approach is that some degrees of freedom are directly managed by the planner, whereas some others are computed using closed-form inverse kinematics. Furthermore, some planning operations are performed reasoning in the robot's workspace, instead of the configuration space. Although this requires introducing additional variables, reasoning in the workspace allows a better control of the manipulated object, and may lead to more human-predictable motions when the robot works alongside humans. Further details about the extension of MoveIt! to treat dual-arm manipulation are provided below.

V. Implementation Details

A. Functional view

At the highest level, the robot programming framework manages the tasks to be performed by the robot. To process a specific part, a sequence of motions is defined, as described in the previous section. Then, for each motion M_i, a planning request is sent to MoveIt!. Following the type of the task and the *group* (i.e. set of joints) involved, the request is performed by MoveIt! using the dedicated action/service. We implemented specific action/service servers to compute pick and place motions with one or two arms.

B. Developed components

Several components were implemented[12] to add the dual-arm capability to MoveIt!. All the components are ROS packages and most of them are plugins. Fig. 5 shows the interconnection of these components to solve a dual-arm planning request (coming from the Application Development layer). Each developed component is described below.

[11]Open Motion Planning Library (OMPL) : http://ompl.kavrakilab.org/

[12]The implemented MoveIt! components will be distributed as open-source code in the short future.

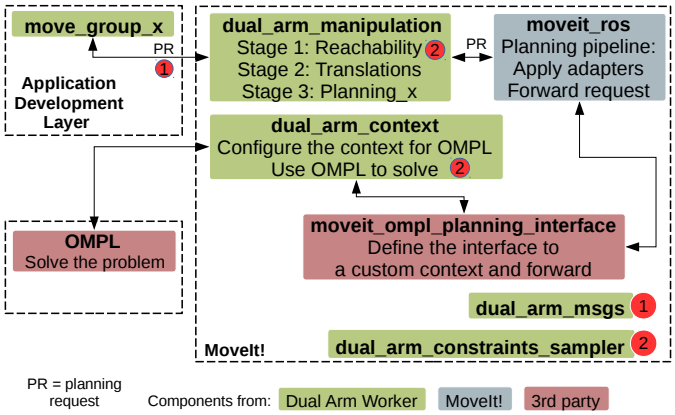


Fig. 5: Functional view of the developed packages to solve a dual-arm planning request. The numbers 1, 2 indicate where the two unconnected components at the bottom are used.

1) move group x package: Overrides the MoveGroup object provided by MoveIt! to interact with a robot. It adds the dual-arm pick/place functions and eases the work for a user to build a motion planning request.

2) dual-arm msgs package: Holds the definition of actions and messages used to communicate a dual-arm pick/place request to one of the dual-arm servers.

3) dual-arm manipulation package: Defines a plugin to add the dual-arm manipulation capability to MoveIt!. It contains two action servers that process dual-arm pick and place requests by decomposing them into three stages. For instance, a "place a part" query is decomposed as:

- *goal reachability*: It is aimed to generate a valid state of the robot to place the object at a given location. For this, the dual-arm constraint sampler is used to generate the valid state for both arms and the torso from the pose of the object and the two end-effectors.
- *translations*: Generates the start/approach/retreat linear trajectories following the planning request.
- *plan*: Computes the trajectory from the last state of a start trajectory to the first state of the approach trajectory. This stage uses the dual-arm context to treat the problem with dual-arm constraints.

4) dual-arm constraint sampler package: This sampler generates valid configurations of a dual-arm robot for a given pose of the two end-effectors, which is defined by the pose of the manipulated object and the relative reference frames for grasping it. This sampler requires inverse kinematics solvers for the arms, but is general-enough to be used with any torso (1, 2, ... dofs). Using ideas from related work [21], [20], the algorithm incrementally samples the joint values of the manipulation group (torso+arms) following a 3-stage procedure. First, the configuration of the torso is sampled based on the pose of the two end-effectors. Then, inverse kinematics (IK) problems are solved to determine the configuration of both arms satisfying the pose constraint. The analytical solver IKFast[13] [22] was used in this work. Finally, the validity of the whole configuration with respect to collisions is tested.

5) dual-arm planning context: Inside MoveIt!, a planning context is used to set up objects of the planning library that are then applied to solve a request. Among the OMPL objects, the state-space defines the set of state variables seen by OMPL, and provides methods to process states. A new context was developed to perform dual-arm motion planning directly considering the pose of the manipulated object. More precisely, this context involves the set of joint values for the torso and the arms, plus the pose of the object. The implemented OMPL state-space incorporates methods to manage the compound states and to perform actions (distance, interpolate, sample, ...) required by the planner that satisfy the dual-arm constraint. It can be plugged to MoveIt! using the *moveit_ompl_planning_interface* package.

VI. Results

This section presents the results of an evaluation of the computational performance of the motion planning algorithms, and experimental results about the quality of the trajectories executed by the robot.

A. Dual-arm motion planner

Solving the dual-arm motion planning problem, corresponding to Md_4, is the most computationally time-consuming part of the process. For this, we apply the single-query bi-directional RRT planner extended to closed kinematic chains, as explained in the previous sections. We evaluated the performances of the planner to solve 100 queries. A time-out of 5 seconds was set, after which the planner returns failure. CPU time presented in this section corresponds to an Intel® Core™ i7-6820HQ processor at 3.60 GHz with 16 GB of DDR4 SDRAM.

Table I presents a summary of the results for the planner. The success rate of 100% indicates that all the planning queries were solved in less than 5 seconds, which is an indicator of the good performance of the planner given the complexity of the constrained dual-arm motion. The average time including path planning and post-processing (path simplification, trajectory generation and verification) is around 1.5 seconds. This computational efficiency is in part explained by the high sampling speed provided by the *dual-arm constraint sampler* (around 200 configurations per second), which allows to rapidly construct the exploration tree.

[13]IKFast plugin : http://docs.ros.org/indigo/api/moveit_tutorials/html/doc/ikfast_tutorial.html

TABLE I: Dual-arm planning results averaged over 100 attempts to generate the motion Md_4.

success	100%
time to find a path	0.91s
time to post-process path	0.51s
total time to solve request	**1.42s**

B. Experiment

The developments presented in this paper were validated in simulation as well as on the real robot. A video showing experimental results in different situations, including complicated motions due to a very cluttered workspace, is available[14]. All the planned motions were accurately executed using basic methods implemented in ROS/MoveIt! and the standard controller of the Nextage robot, based on a velocity control following a trapezoidal profile. The available ROS driver for commanding the robot accepts, as a command, a message specifying the desired position, velocity and acceleration of each joint with a given time resolution. Afterwards, the robot QNX controller acts as a black-box for following the planned trajectory.

The implementation of the AddTimeParameterization *adapter* from MoveIt! was used to generate trajectories from the planned paths considering velocity and acceleration limits. Although our experiments show that this method provides good results in most cases, we envision the implementation of a more sophisticated trajectory generation algorithm [23], [24]. Table II shows the time for the planned elementary motions considering the maximum speed values in Table III, and a maximum acceleration value of 1 rad/s^2 for all joints. The duration of the motions was measured at the Application Development level, including the time to solve planning requests, which is very small compared to the execution time. For the coordinated dual-arm motions (Md_4 and Md_6), the time is higher than for the other motions due to the length of the trajectory to move the part between the rack and the deburring support.

The coordinated dual-arm motions are also the most challenging ones for execution. The constraints imposed by the dual-arm grasping make the configuration of the arms to be dependent on each other. If the controller is not able to track the trajectory provided by MoveIt! accurately, then large forces/torques can be exerted on the arms and the part, leading to a failure and possible break. Fig. 6 presents a comparison of the joint values for both arms between the planned trajectory provided by MoveIt! and the executed one corresponding to the coordinated motion Md_4. The plots show that the deviation is small (less than 0.012 radians on average and less than 0.1 radians in the worst case). The largest errors appear in the last part of the trajectory for certain joints. This is particularly visible for joints 1 and 2 of the left arm. The detected errors can be explained by slight inaccuracies of the models. Nevertheless, these errors do not produce any appreciable jump during the execution of the planned trajectory. Note also that no warning message due to an excessive effort during the manipulation was returned by the robot. These results show that, thanks to the accuracy of the dual-arm motion planner, complicated trajectories can be executed using simple controllers. Nevertheless, as further discussed in the next section, the implementation of more sophisticated control methods is essential to enhance safety and robustness, and would be a valuable complement to the work presented in this paper.

[14]Video of experiments:
https://homepages.laas.fr/jcortes/ETFA2019/ETFA2019-DAW-video.mp4

TABLE II: Execution time of the planned elementary motions to process a part (averaged over 10 tests)

motion	time (s)	motion	time (s)	motion	time (s)
Ms_1	7.74	Ms_2	3.89	Md_3	4.21
Md_4	15.5	Md_6	9.23	Ms_7	3.08

TABLE III: Maximum speed values (rad/s)

joints	speed	joints	speed	joints	speed
Torso-J$_0$	1.91986	Arms-J$_0$	3.00197	Arms-J$_1$	2.32129
Arms-J$_2$	3.99680	Arms-J$_3$	4.60767	Arms-J$_4$	3.90954
Arms-J$_5$	5.23599				

VII. Conclusion and Future Work

In this paper, we have presented ongoing work on the development of flexible and easy-to-use software packages to facilitate the integration of autonomous dual-arm manipulators in manufacturing industry. Such software is an essential ingredient to enable rapid reconfiguration of a robot in a versatile work-cell.

The current capabilities of the implemented methods were illustrated through an application scenario in the context of aerospace industry. Using models of the robot and the work-cell, the operator can easily specify a manipulation task, involving single-arm and dual-arm motions. The robot automatically computes collision-free trajectories in very short time, and then executes the motions with high accuracy.

Our first goal for future work is to improve the robustness of motion planning and execution methods by integrating more sophisticated dynamic collision avoidance algorithms. The idea is to equip the work-cell with other sensors, in addition to the RGB-D camera, able to detect and localize the presence of unexpected obstacles, and to use this information for on-line trajectory adaptation or re-planning. This is required for safety when human operators are allowed to enter the work-cell while the robot is working. As mentioned at the end of the results section, safety and robustness can also be enhanced using more advanced control methods. In particular, and depending on the accessible control inputs and on the embedded force/torque sensors, force control methods (Chapter 9 in [25]) would be a perfect complement to the dual-arm motion planner.

Aiming to further simplify robot programming, another objective for future work is the implementation of methods for the automatic decomposition of the high-level manipulation task into the sequence of elementary motions and actions. Several methods have been proposed to solve such a task decomposition problem based on extensions of sampling-based motion planning algorithms [26], [27], [28], or on a combination of motion planning and symbolic task planning algorithms [29].

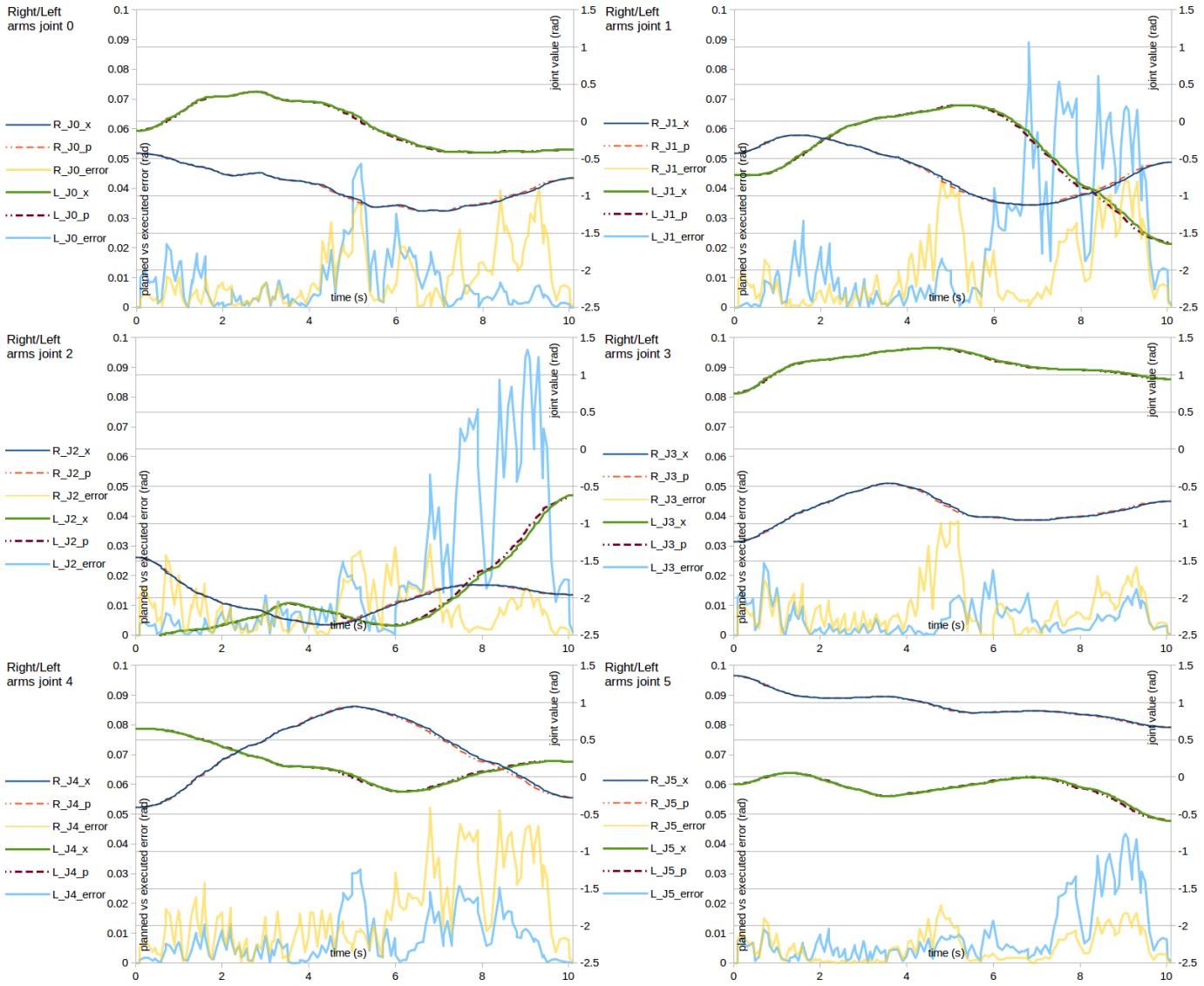


Fig. 6: Comparison of the right and left arms joint values for motion Md_4 between a **p**lanned trajectory and the e**x**ecuted one.

ACKNOWLEDGMENT

This work was conducted in the context of the DualArm-Worker experiment, which received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration, as part of the project ECHORD++ under grant agreement no. 601116.

REFERENCES

[1] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The DARPA robotics challenge finals: Results and perspectives," *Journal of Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017.

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

[3] N. Hudson, J. Ma, P. Hebert, A. Jain, M. Bajracharya, T. Allen, R. Sharan, M. Horowitz, C. Kuo, T. Howard, L. Matthies, P. Backes, and J. Burdick, "Model-based autonomous system for performing dexterous, human-level manipulation tasks," *Autonomous Robots*, vol. 36, no. 1, pp. 31–49, 2014.

[4] B. Cohen, S. Chitta, and M. Likhachev, "Single- and dual-arm motion planning with heuristic search," *International Journal of Robotics Research*, vol. 33, no. 2, pp. 305–320, 2014.

[5] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0 – securing the future of german manufacturing industry," Acatech – National Academy of Science and Engineering," Final Report of the Industrie 4.0 Working Group, 2013. [Online]. Available: http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf

[6] H. Herrero, J. L. Outón, M. Puerto, D. Sallé, and K. López de Ipiña, "Enhanced flexibility and reusability through state machine-based architectures for multisensor intelligent robotics," *Sensors*, vol. 17, no. 6, 2017.

[7] 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 *ICRA Workshop on Open Source Software*, 2009.

[8] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E.

Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees : Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. Donald, K. Lynch, and D. Rus, Eds. A.K. Peters, 2001, pp. 293–308.

[13] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[14] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 17(6), pp. 951–958, 2001.

[15] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[16] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 105–117, 2013.

[17] L. Han and N. M. Amato, "A kinematics-based probabilistic roadmap method for closed kinematic chains," in *Algorithmic and Computational Robotics: New Directions (WAFR2000)*, B. Donald, K. Lynch, and D. Rus, Eds. A.K. Peters, 2001, pp. 233–245.

[18] J. Cortés and T. Siméon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Algorithmic Foundations of Robotics VI*, M. Erdmann, D. Hsu, M. Overmars, and F. van der Stappen, Eds. Springer-Verlag, 2005, pp. 75–90.

[19] A. Sintov, A. Borum, and T. Bretl, "Motion planning of fully actuated closed kinematic chains with revolute joints: A comparative analysis," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2886–2893, 2018.

[20] M. Gharbi, J. Cortés, and T. Siméon, "A sampling-based path planner for dual-arm manipulation," in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2008, pp. 383–388.

[21] J. Cortés, T. Siméon, and J.-P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using prm methods," in *Proc. IEEE Int. Conf. Robotics and Automatation*, 2002, pp. 2141–2146.

[22] R. Diankov, "Automated construction of robotic manipulation programs," PhD Dissertation, Carnegie Mellon University, Robotics Institute, 2010.

[23] X. Broquère, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2808–2813.

[24] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2010.

[25] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[26] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, pp. 729–746, 2004.

[27] K. Harada, T. Tsuji, and J. P. Laumond, "A manipulation motion planner for dual-arm industrial manipulators," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 928–934.

[28] P. Lertkultanon and Q.-C. Pham, "A certified-complete bimanual manipulation planner," *IEEE Transactions on Automation Science and Engineering*, vol. 15(3), pp. 1355–1368, 2018.

[29] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *International Journal of Robotics Research*, vol. 28, no. 1, 2009.