Rule-Based Approach for Constrained Motion Control of a Teleoperated Robot Arm in a Dynamic Environment

Marie Claire Capolei*
Technical University of Denmark, Kgs. Lyngby, Denmark
Email: macca@elektro.dtu.dk

Haiyan Wu
Danish Technological Institute, Taastrup, Denmark
Email: hwu@teknologisk.dk

Adrian Llopart M., Silvia Tolu, Ole Ravn
Technical University of Denmark, Kgs. Lyngby, Denmark
Email: {adllo, stolu, or} @elektro.dtu.dk

***Abstract*—This paper presents a preliminary robotic solution for constrained teleoperation tasks in an uncertain and dynamic environment. The robotic system is supported by a reasoning agent which makes the control action reactive and context-sensitive. The investigation is motivated by the future Human-Robot collaboration, therefore, it focuses on minimizing or avoiding collisions within the robot and the surroundings objects. The report describes the developed control architecture, which, in its modular and hierarchical structure, combines knowledge from different areas such as control theory, path and trajectory planning, computer vision, collision avoidance, and decision-making theory. The software is implemented in a ROS framework, in order to support a clear and modular design, suitable for future extensions and integration on different hardware components. The experiments are run on both real and simulated systems. The results show an autonomous robot capable of continuously adapting its movements despite the external agent interruptions, with a 99% success rate. We can conclude that an adaptive robotic system capable of performing constrained tasks and simultaneously reacting to external stimuli in an uncertain and dynamic environment is potentially obtainable.**

***Index Terms*—Rule-Based System, Decision-Making, Autonomous Robot, Intelligent system, Robot Control, Constrained Motion Control, ROS**

I. Introduction

Traditionally, robots were frequently used in industry in static and known conditions; therefore the autonomous and safe robotic moves were not required in real world applications. Robots were just machine executing pre-computed trajectories in fenced areas.

Only in the Nineties, industrial robots were equipped with exteroceptive sensors in order to perceive the surroundings and the object they were manipulating. This endowment enabled the increase of detection methods, but it was not enough to introduce robots in the human vicinity. In 2005, a new complete generation of robots emerged: the collaborative robots, or cobots. Cobots addressed to most of the tasks which until now could not be fulfilled due to security reasons [1-2]. Lately, it was proved that the human-robot cooperation could bring to a drastic reduction of the workplace accidents and to the optimization of the production process in different commercial sectors [3-5]. In the last decade, this new trend is leading to a significant progress in the design of new types of robots and the development of advanced collision avoidance methods. As a matter of fact, the biggest concern of human-robot collaboration is the safety [6-8]. Human physical safety is assured by avoiding or minimizing collisions within humans and machines; the procedures to resolve the safe human-robot collaboration can be mainly branched into pre-collision safety strategies [9-12] and post-collision safety strategies [13]. Hence, the introduction of robots in a dynamic human populated environment implies the self-awareness of the artificial system boosted by reactive reasoning and decision making skills.

In 2017, during the design of an architecture for controlling a teleoperated industrial manipulator employed in surgical application [14], we faced the problem of safe human-robot interaction. The objective of the investigation was to insert and position accurately and safely a long and slight surgical tool, the laparoscope, throughout a little incision of the patient's skin, which is the fulcrum point of the tool movements. The complexity of the robot control was not only in the constrained movements of the laparoscope, but also in the possible disturbances caused by the surroundings (E.g. a surgeon

Manuscript received August 6, 2018; revised April 5, 2019

doi: 10.18178/ijmerr.8.3.393-400

colliding into the robot could move the system and consequentially provoke dangerous movement of the tool). However, in that introductory analysis, the robot was unconscious of the environment populated by humans and the tool it was manipulating; therefore it could not react or prevent the external disturbances, nor plan a collision free path. This led to the inevitably questioning of the human-robot interaction concept and the design of a system reactive and secure enough while handling several constrains in such delicate and uncertain circumstance.

Assuming the general situation, the robot is in charge of manipulating a known tool always respecting determinate end-effector constrains, in an environment where dynamic obstacles, such humans, can disturb the execution of the action. The overall problem can be separated in two main sub-cases: *waiting state*, the robot is holding the tool waiting for the user command and it has to prevent any occurring collision with the human, due to the fragility or harmfulness of the tool; *moving state*, the robot is manipulating or moving the tool in direction of the working area and has to keep it in a safe position while avoiding collision.

This complex problem demands a well-organized architecture design that can promptly manage different events. Particular interest is given to the work of *Zhou et al.* [15], which, in 2005, presented a new real-time control strategy based on an expert-system for a robot employed in the inspection of a power transmission line. All the operations of the robot were organized into groups and manipulated by a two level expert system. The result was a robot able to crawl along the transmission line and negotiate with various types of obstacles in a reliable and robust manner. Later in 2009, *Wang et al.* [16] proposed a rule-based forward reference approach for the design of a decision subsystem of an intelligent robot. The method avoided complex decision logic, improved the performance of the heuristic direct control and decreased the response time of the machine-learning approach. Wang stated that apart from the system's shorter response, one of the main advantages of the rule-based system is the separation of the knowledge base with the rest, which facilitates the maintenance and boosts the versatility of the control system. The investigation proceeded and the interest moved on methods used to efficiently plan collision free path. In 2016, *Nguyen et al.* [17] designed a fast collision free path planning for a humanoid robot arm based on a hierarchical and modular control architecture that employed existing Cartesian solver and controllers thanks to the implementation in ROS [18]. While in 2011, *Haddadin et al.* [19] boosted the reactivity of their collision prevention using a circular field method. However, all these solutions are not complete for solving the limitations of the tool movements and handling several events at the same time.

Our project idea has merged all these concepts together in a more complete system. What we propose in this introductory research is a robotic control system capable of extracting useful information from the environment, reasoning about it in order to accomplish cautiously the requested task and prevent any occurring collision. We organized the control system in a hierarchical manner to facilitate the information processing at different levels of abstraction. All the components of the system are regulated by a rule-based supervision system. This high-level control system has access to all the current internal and external states, such as the working scene information, the planning and execution status and the user's desires. The low-level information from the environment is collected and processed via specific algorithms that extract only few but important features without overloading the computer memory. This valued data is sent to higher level module that transforms it in collision information. To solve the obstacle detection problem in a computationally cost efficient manner, we introduce a synthetic and fast algorithm based on spherical coordinates system.

Firstly, the report is dedicated to the organization of the overall control architecture in section II. This description introduces the data flow managed by the reasoning system presented in section III. The second part of the paper focuses on the corresponding design and implementation of the software (section IV). We conclude with the description and evaluation of the experiments (section V).

II. Architecture Organization

In the general assessment, the user communicates with a computer which elaborates all the information necessary for controlling the robot, such as the user's command, the robot trajectory state and the perception data. The computer is connected to an industrial manipulator and a stereo vision camera. The design of the control architecture (Fig. 1) is based on the Nashrem model proposed by NASA in the eighties [20] and re-organized by Siciliano [21] for industrial manipulators later on. More in details from the higher to the lower levels the architecture is partitioned as follow.

A. Task Level

The *Supervisor Module* contains the virtual agent which combines and elaborates the robot state, the external events (such as the occurrence of collision) and the requested task in order to guide the robot throughout a safe and reactive execution.

B. Action Level

The *Future Collision Detection Module* detects the possible collision and reports it directly to the *Supervisor* in order to take action. Concurrently, the Command *Interpreter Module* interprets the high-level command from the *Supervisor* and conscious of the current robot state converts it into the most appropriated tool pose.

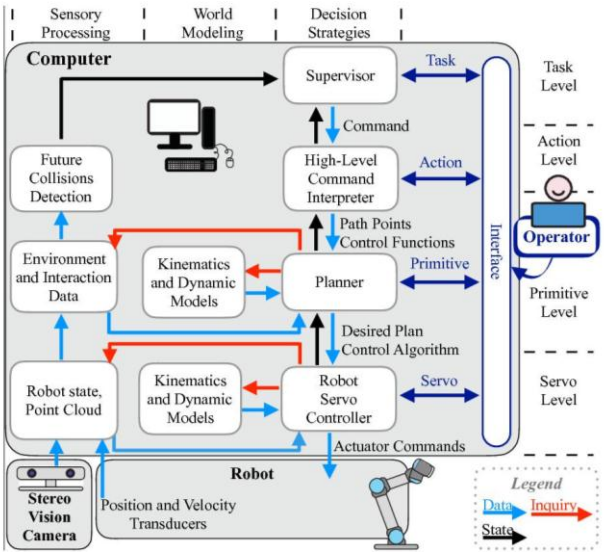

Figure 1. Note Hierarchical levels of the control architecture. The top levels of the control architecture are involved into a logical action planning. The lower level is oriented to physical motion execution. The modules communicate via data flows: those directed bottom-up regard measurements and outcome of actions (E.g. the raw data from the stereo vision camera, the goal status, the encoders data), whilst the top-down information regards the communication of directives (E.g. the high level command from the Supervisor module or the desired path points). In the column of the hierarchy, from right to left: the first leg consists of the Decision Strategies modules which plan and execute the decomposition of high level goals into low level actions; the World Modeling modules where all the kinematics and dynamics models are defined; the Sensory Processing modules filter, correlate, detect, and integrate sensory information over both space and time in order to give them value at different levels of abstraction (E.g. recognize and measure patterns, features, objects, events).

C. Primitive Level

The *Environment and Interaction Data Module* reports all the information about the occurrence of conflicts between motion planning and execution. This module transforms the point-cloud into higher level objects that are included in the planning scene. The *Kinematics and Dynamics Module* shares all the kinematics and dynamics information about the manipulator. Meanwhile, the *Planner Module* designs the servo-controller trajectory when it receives the new tool pose from the *Command Interpreter* and the current joints states from the *Servo level*. To find a collision-free path, the module compares the desired configuration with a model of the robot, accessible in the *Modeling Module*, as well as with the planning scene, available in the *Sensory Module*. Hence, the motion trajectory is interpolated in order to generate the references for the *Servo level*.

D. Servo Level

The *Robot state and Point Cloud Module* provides and processes measurements of the proprioceptive and exteroceptive sensors. In particular, the raw data from the Stereo vision camera is filtered, processed and transformed into point-cloud data. The *Kinematics and Dynamics Module* elaborates the terms of the control law depending on the current state of the robot arm and communicates them to the *Robot Servo Controller Module*. The latter includes the low-level position control, which on the basis of the requested trajectory provides the driving signals to the servo-motors.

III. Decision Making in a Dynamic Domain

Robot autonomy is of high relevance for Human-Robot collaboration, in such uncertain environment, the robot must not only reason about itself but also reason about the surroundings in order to execute appropriate actions. In the considered problem, the robot receives instructions that have to cover different situations, and whose execution is influenced by numerous context-sensitive decisions. For instance, what if the robot is waiting the next command and an obstacle is approaching? What if the robot is executing a constrained task and there is an obstacle in the way or something fails and the execution is aborted? We covered most of the conditions that revealed during the testing, such as failing of the trajectory, abortion of the execution (due for instance to sensors fails) and re-planning if something in the environment changed.

In the general condition, the robot is in a *waiting state*, whenever an event occurs (for instance a user's command or an approaching obstacle) it reacts and changes its status. Imagine the robot is waiting for the next user's command while it is executing a constrained task, for instance positioning a tool, it is important that the robot does not collide with any external rigid body due to the fragility or the harmfulness of the handled object. Hence, the robot should be able to prevent any collisions. The robot is initially in a *waiting state*, if an external rigid body is detected as getting too close to the robot, the robot changes its state into *preventing* and a new movement in the opposite direction with respect to the approaching obstacle is requested. Thus, the program plans the optimal trajectory, if there are any changes in the environment while planning, the program plans again. Once the trajectory is defined, it is sent to the low level position control for the execution. If the trajectory cannot be executed, or rather it is failed or rejected, the robot is back in the *planning state*, if the robot is in the *moving state* and the execution is aborted, it stops and plans again. In case there is an obstacle in the way while the robot is moving, the robot stops and plans. Whenever the robot reaches the goal pose, it will be back to the *waiting state*.

A. Rule-Based Reasoning System

It is evident that this is not a traditional procedural or object-oriented control problem, where the robot knows a priori what to do, how to do it, and in what order. Moreover, the system has to react quickly to changing conditions or failures. That led us to exploit a rule-based methodology, which uses rules to derive conclusions from premises [22]. Specifically, a rule is a command that suits only some subset of problems, then a rule engine regulates which rules to employ at any given time and executes them as opportune. The rule-based approach solved the need of a fast and continuously reactive system suitable in dynamic environments, such system is capable of changing intentions depending on the situation and

dynamic constraints. With a practical example, the working memory contains the fact describing the robot waiting state (blue boxes in Fig. 2), if an approaching rigid body is detected the fact *obstacle* (red boxes) is asserted and appears in the data base, the inference engine fires the rule *preventing-collision* (Fig. 2.b). The rule updates the robot state into *preventing*, asserts the fact *come-back*, and sends the command to move the robot in the opposite direction with respect to the obstacle. Once the robot has prevented the collision, the robot state in the working memory is *waiting*. Whenever, the fact *come-back* is available in the working memory and there are no obstacles (green box) the rule *come-back* fires and the robot is requested to move in direction of the position previous to the prevention.

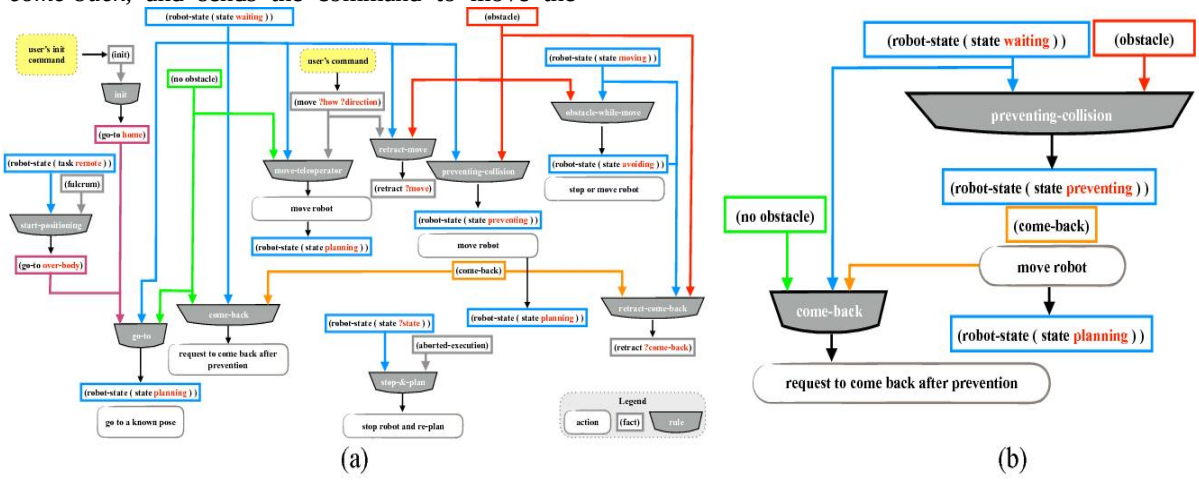

Figure 2. Reasoning system dependencies model: (a) overall structure, (b) preventing a collision example. The diagram gives the guidelines to understand how the main rules are related to each other, the parallelism of their execution, the conditions needed to be activated and how their output affects the performance of the task and the behavior of the robot. For instances, it is evident how the current state of the robot (in blue) and the presence or not of an obstacle (in red and green respectively) influence the decisions flow

IV. Software Components

The software is implemented on a computer running the open source operating systems Ubuntu 14.04 and ROS Indigo [18], in particular, the software ROS-industrial provides all the tools and drivers for industrial manipulators such the UR10. The supervisor node is programmed in Pyclips [23]. This section presents the detailed description of the software elements (Fig. 3). The implementation is based on the control system functional architecture seen in section II.

Teleoperator Node interprets the keyboard input and converts it into command for the robot. The desired action is published on the topic */events/teleoperator*.

Supervisor Node contains the virtual agent "brain". This node subscribes to the main topics in order to have information about: trajectory execution status; goal status; robot status; user's command; obstacle detection. This data is added to the working memory and used by the inference engine in order to quickly react to the observed situation and reach the current goal.

Desired end-effector Node reads the high-level input from the supervisor and transforms it into the next end-effector pose. The node subscribes to the */joint-states* topic to update the current pose of the end-effector and consequently of the tool. The desired end-effector pose is published on the */end-effector-pose/current* topic.

Planner Node subscribes to the topic */end-effector-pose/desired*, and aware of the working scene state, it plans the optimal trajectory that will be sent to the robot servo controller. The planning method is based on the *rapidly exploring random tree star* (RRT*) algorithm in 3D workspace.

Robot Driver Node is responsible for the communication with the hardware, in particular, it interprets the ROS messages into low level position commands.

Stereo Vision Node transforms the pre-filtered data from the Kinect camera into Point Cloud with the world frame as reference frame. The robot model is filtered from the Point Cloud, in this way the system does not see the robot as a collision object and consequentially stop (or plan again).

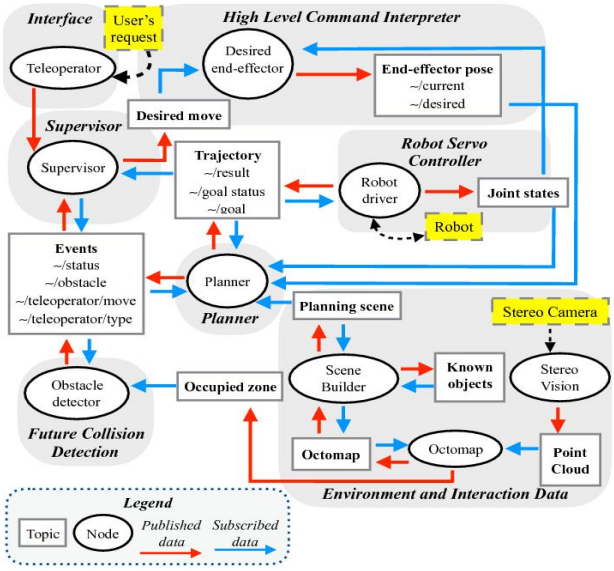

Figure 3. Software components communication.

Octomap Node employs the OctoMap library [24] in order to convert the point-cloud into an efficient real-time 3D occupancy grid map. The node is responsible of the map update whenever a change occurs.

Scene Builder Node collects the Octomap data and other pre-implemented objects, such as the tool, and adds them to the working scene as collision objects. It also creates the interactive markers used to move the collision object in the scene.

Obstacle Detector Node checks if the occupied zone in the working scene published by the Octomap node is too close to the robot. In case of possible collision it alerts the supervisor node publishing the obstacle information on the */events/obstacle* topic.

V. Experiments

The experiments are run both on a real and on a simulated robotic system. In both cases, the robot is teleoperated via keyboard commands and monitored both on the terminal and on the simulator.

A. Experimental Setup

The computer is connected via serial bus to a stereo vision camera, and via Ethernet to an industrial robot arm. More details about the computer employed for the experiments are listed in Table I. The manipulator chosen for the experiments is the UR10 from Universal Robot [25]. This is a six-axis robot arm. The robot is mounted on a table. The maximum speed of each revolute joint is $180\ ^{o}sec^{-1}$. The stereo vision device employed to get the localization of the objects is the Microsoft Kinect [26]. The camera has a pixel resolution of 640 × 480 and a frame rate of 30 *fps*, whereas the IR sensor use a matrix of 320×240 *pixel*. The camera is positioned in front of the robot arm during the experiment. Whenever the robot plans a new trajectory this one is visualized for few seconds before the execution.

Real System. The tests run on the real robot still include the simulator usage, this tool is useful to visualize the occupied working space and the planned trajectory before the possible execution. Fig. 4.a shows the UR10 employed for the experiments with the holder and the tool attached. The box with the black pinhole is used in order to simulate the fulcrum point where the robot inserts and positions the tool around during the constrained task.

TABLE I. Technical Specification of the Computer.

Operating System	Ubuntu 14.04 LTS
Memory	7.7 GiB
Processor	Intel Core i7-6700 CPU @ 3.40 GHz x 8
Graphics	GeForce GT 730/PCIe/SSE2
OS type	64-bit

*Simulated System***.** The software used to visualize the simulated robot and the working scene is Rviz, which is a ROS package. Fig. 4.b shows how the planning scene looks like in the simulator, more specifically: the sky-blue items represent the collision objects implemented offline in the program, such as the table where the robot is placed on; the orange bodies synthesize the attached collision objects that are implemented off-line, such as the tool and its holder; the group of small cubes illustrates the Octomap, thence all the dynamic objects captured by the Kinect.

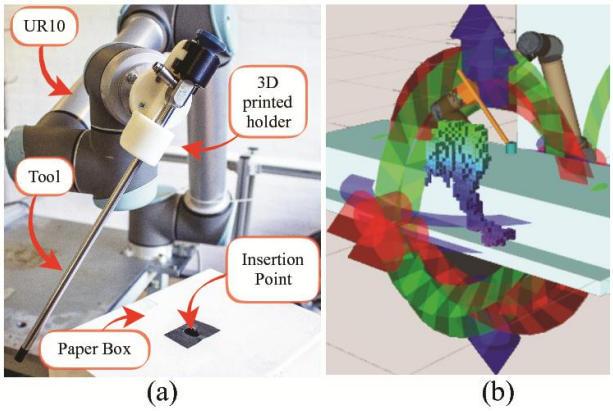

Figure 4. Experimental set-up with the (a) real and (b) simulated system.

B. Experimental Results

The program is run eight times for both systems, for about twenty minutes each time. It is assumed that the robot end-effector is positioned in the same Cartesian quadrants where the table is. Once the program is started, the robot goes directly to the home pose and waits for the user's commands. During the tests, the robot was requested to execute all the teleoperation tasks and was consistently interrupted by external objects captured by the camera. Fig. 5 shows an example of the robot preventing the collision with the human. The executed trajectories are in total 386, whose 63 executions were involved into collision prevention, and 51 into avoidance of collision while the robot was moving. Both the simulated and real system have been able to execute the trajectory the 99% of the time, the 1% failed due to some inaccurate filtering of the robot from the point-cloud data in the real system test, and due to some Kinect noise in the simulations (Table II). It results that most of the trajectories are planned in order to accomplish the user's input (blue areas in Fig. 6) and to re-plan (red areas), the latter includes when the robot plans in order to avoid an obstacle, only a small percentage of the planning is involved into coming back to the previous position (orange areas), while the trajectory execution failed the 1% out of the total (green areas). The real system spent considerably more time re-planning; this is due to the higher percentage of collision avoidance. The trajectories planned to avoid collision are 11% in the simulation and 15% on the real system out of the total planned trajectories. The time that the system employs to transform the user's input into final trajectory is measured or rather the span from the reception of the user's command to the start of the execution.

TABLE II SYSTEM COMPARISON: PLANNED TRAJECTORY EXECUTION.

System	Total Trajectory	Successful Execution	Failed Execution
Simulated	196	99%	1%
Real	193	99%	1%

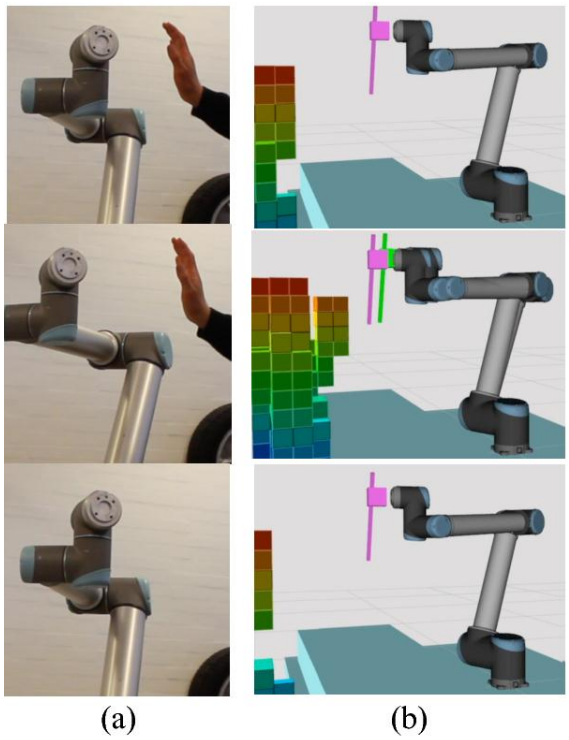

Figure 5. (a) Real and (b) simulated robot preventing collision with human and coming back to previous pose.

As shown in Table III, both systems present similar results. Thus, this value totally depends on the computer performance and on the communication with the robot driver. However, it is possible to observe a slight different duration of the trajectory execution, or rather from the moment the robot servo controller receives the trajectory to the instant the robot is on goal. Assuming that the time set for the execution of the trajectory is 2 *sec* (this parameter has to be tuned in order to perform movements that do not frighten the human collaborator), the real system presents an offset of a few millisecond, with respect to the simulated system (Table IV). This time lag is due to the delay accumulated by the Ethernet communication with the real robot.

Lastly, the responsiveness of the detection algorithm has been investigated, which is the time span from the acquisition of the Octomap data (mean publishing rate 0.021 *sec*, standard deviation 0.012 *sec*) to the actual detection of a possible collision has been recorded.

TABLE III. TIME SPAN FROM USER'S INPUT TO START EXECUTION

System	Mean [s]	Standard Deviation [s]
Simulated	0.052231	0.078428
Real	0.080954	0.110010

Lastly, the responsiveness of the detection algorithm has been investigated, which is the time span from the acquisition of the Octomap data (mean publishing rate 0.021 *sec*, standard deviation 0.012 *sec*) to the actual detection of a possible collision has been recorded. The obstacle detection algorithm results to have a mean frequency of 4*KHz* (mean 0,000247 *sec*, standard deviation 0,000077 *sec)*.

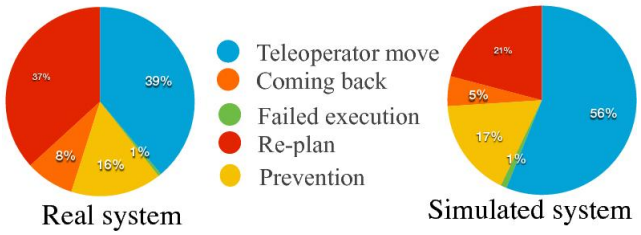

Figure 6. Systems comparison: partition of total planned trajectory.

TABLE IV. TIME SPAN FROM TRAJECTORY EXECUTION TO ROBOT ON GOAL POSE

System	Mean [s]	Standard Deviation [s]
Simulated	2.589558	0.319595
Real	2.814905	0.416457

VI. CONCLUSIONS

The research presents an intelligent and autonomous robotic system capable of moving smoothly in an environment containing dynamic external rigid bodies. The developed system provides a solution for generic teleoperation tasks and constrained actions. In particular, the control system reacts efficiently to external disturbances whilst respecting the imposed constrains. The experimental results exhibit a high success rate of 99% out of the total planned trajectories. The main advantage of the rule-based reasoning system is the fast response to changing conditions or failures. However, the expert system is not able to handle situations that are not present in the database. In future, the work should investigate on additional methods that would make the system more generic and robust to the occurrence of new events; for instance the control flow discovery via event stream. This additional component could enable the robotic tasks extension, for example with pick up and place tasks.

The overall vision system is capable of extracting real-time valuable information from the environment filtering out the robot and rapidly alerting the higher level structure in case of a possible collision. On the other hand, the performance could be improved by enhancing the noise reduction or employing additional stereo vision cameras in order to have multiple viewpoints. Moreover,

it could be of great interest to exploit a human-tracking algorithm to predict future movement and collision or instrument-tracking in order to check the actual state of the tool, as it has been investigated by *Sahu et al.*[27]. The use of an industrial robot arm allows for a large-scale

application. The integration of the software components in ROS supports a clear and modular design, suitable for future extensions and application on different hardware platforms.

REFERENCES

[1] European Parliament, "European parliament resolution of 16 february 2017 with recommendations to the commission on civil law rules on robotics (2015/2103(inl))," 2017.

[2] International Organization for Standardization, "Robotics iso/tc 299," 2017.

[3] M. Hägele, W. Schaaf, and E. Helms, "Robot assistants at manual workplaces: Effective co-operation and safety aspects," in *Proc. the 33rd ISR (International Symposium on Robotics)*, pp. 7–11, 2002.

[4] A. Bauer, D. Wollherr, and M. Buss, "Human–robot collaboration: a survey," *International Journal of Humanoid Robotics*, vol. 5, no. 01, pp. 47–66, 2008.

[5] K. E. Kaplan, K. A. Nichols, and A. M. Okamura, "Toward human-robot collaboration in surgery: performance assessment of human and robotic agents in an inclusion segmentation task," in *Proc. 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 723–729, 2016.

[6] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Safety analysis for a human-friendly manipulator," *International Journal of Social Robotics*, vol. 2, no. 3, pp. 235–252, 2010.

[7] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[8] J. Heinzmann and A. Zelinsky, "Quantitative safety guarantees for physical human-robot interaction," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 479–504, 2003.

[9] L. Balan and G. M. Bone, "Real-time 3d collision avoidance method for safe human and robot coexistence," in *Proc. 2006 IEEE/RSJ International Conference On Intelligent Robots and Systems*, , pp. 276–282, IEEE, 2006.

[10] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. 2012 IEEE International Conference on Robotics and Automation* (ICRA), pp. 338–345, 2012.

[11] D. Kulić and E. Croft, "Pre-collision safety strategies for human-robot interaction," Autonomous Robots, vol. 22, no. 2, pp. 149–164, 2007.

[12] D. Shin, I. Sardellitti, Y.-L. Park, O. Khatib, and M. Cutkosky, "Design and control of a bio-inspired human-friendly robot," The International Journal of Robotics Research, vol. 29, no. 5, pp. 571–584, 2010.

[13] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IROS 2008. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008. pp. 3356–3363, 2008.

[14] M. C. Capolei, H. Wu, N. A. Andersen, and O. Ravn, "Positioning the laparoscopic camera with industrial robot arm," in *Proc. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 138–143, IEEE, 2017.

[15] F. Zhou, J. Wang, Y. Li, J. Wang, and H. Xiao, "Control of an inspection robot for 110kv power transmission lines based on expert system design methods," in *Proc. 2005 IEEE Conference on Control Applications*, 2005. CCA 2005., pp. 1563–1568, 2005.

[16] Y. Wang, Q. Wang, and R. Liu, "A rule-based decision sub-system design approach for intelligent robot," in Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on, vol. 2, pp. 525–528, IEEE, 2009.

[17] P. D. Nguyen, M. Hoffmann, U. Pattacini, and G. Metta, "A fast heuristic cartesian space motion planning algorithm for many-dof robotic manipulators in dynamic environments," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 884–891, IEEE, 2016.

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

[19] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments," IFAC Proceedings Volumes, vol. 44, no. 1, pp. 6842–6850, 2011.

[20] J. S. Albus, H. G. McCain, and R. Lumia, NASA/NBS standard reference model for telerobot control system architecture (NASREM). National Institute of Standards and Technology Gaithersburg, MD, 1989.

[21] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, Springer, 2016.

[22] E. Friedman, Jess in action: rule-based systems in java. Manning Publications Co., 2003.

[23] I. Francesco Garosi, Johan Lindberg, "Pyclips, python module," 2017.

[24] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," Autonomous Robots, 2013. Software available at http://octomap.github.com.

[25] Universal Robots A/S, "Universal robots," 2017.

[26] Microsoft, "Microsoft kinect for windows sensor components

[27] M. Sahu, D. Moerman, P. Mewes, P. Mountney, and G. Rose, "Instrument state recognition and tracking for effective control of robotized laparoscopic systems," *International Journal of Mechanical Engineering and Robotics Research*, vol. 5, no. 1, p. 33, 2016.

Marie Claire Capolei is a PhD student investigating in the field of bio-inspired artificial intelligence applied to robotics in the framework of the H2020 FET Human Brain Project at the Technical University of Denmark (DTU). She received the M.Sc. degree in Automation and Robot Technology Engineering and the B.Sc. degree in Industrial Engineering from the Technical University of Denmark, in 2017, and from Campus Bio-medico University in Rome (UCBM), Italy, in 2015, respectively.

Adrian llopart Maurin received the B.S. degree in Industrial Engineering from Polytechnic University of Catalonia (UPC), Spain, in 2013, and the M.S. degree in Electrical Engineering from The Technical University of Denmark (DTU), Denmark, in 2015, where he is currently pursuing the Ph.D. degree. His current research interests include humanoid robots, autonomous robots, and artificial intelligence.

Haiyan Wu received the B.S. degree in Measurement and Control from the Zhejiang University, China, in 2004; the M.S. degree in Control Engineering from both Tongji University, China, and the Technical University of Munich (TUM), Germany, in 2007; and the Ph.D. degree from TUM in 2011. She finished her PostDoc at the Technical University of Denmark (DTU), Denmark, in 2016 where she worked as a researcher until 2017. At the present time she is working as a senior specialist at the Danish Technological Institute.

Silvia Tolu received the M.Sc. degree in Electronic Engineering from the University of Cagliari, Italy in 2006, the M.Sc. degree in Computing Networks Engineering from the University of Granada, Spain in 2008, and the Ph.D degree (distinction) in Computer Science from the University of Granada, Spain in 2012. She is currently working as a PostDoc at the Technical University of Denmark (DTU), where she is team leader of the DTU neurorobotics group in the Sub-project 10-Neurorobotics of the H2020 FET Human Brain Project.

Ole Ravn received the M.S. degree in Civil Engineering from the Technical University of Denmark (DTU), Denmark, in 1984, and the Ph.D. degree in the same university in 1987. He was an Assistant Professor at DTU Elektro until 1991 when he became Head of Section and later on, in 2007, Head of Studies for Master of Science in Electrical Engineering, for the Automation and Control Group. His current research interests are Industrial Automation Systems and Autonomous Robot Systems.