Reactive Planning on a Collaborative Robot for Industrial Applications

Gautier Dumonteil, Guido Manfredi, Michel Devy, Ambroise Confetti, Daniel Sidobre

Reactive Planning on a Collaborative Robot for Industrial Applications

Gautier Dumonteil[1], Guido Manfredi[2], Michel Devy[2], Ambroise Confetti[1] and Daniel Sidobre[2,3]
[1]*Siemens Industry Software SAS, Miniparc 2, 478 rue de la Decouverte, 31670 Labege, France*
[2] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[3] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*
{*gautier.dumonteil,ambroise.confetti*}*@siemens.com,* {*gmanfredi,michel.devy,daniel.sidobre*}*@laas.fr*

Keywords: Industrial robotics, cobot, reactive planning, obstacle detection, ROS architecture

Abstract: A challenge for roboticists consists in promoting collaborative robotics for industrial applications, i.e. allowing robots to be used close to humans, without barriers. Safety becomes the key issue. For manipulation tasks, a part of the problem is solved using new arms like the KUKA LWR, which is able to physically detect a collision from torque measurements on each joint. Nevertheless it is better to avoid collisions, especially if the obstacle is the arm or the head of an operator. This paper describes how obstacles could be detected and avoided, using a single Kinect sensor for the monitoring of the workspace and the reactive planner of the KineoWorks^{TM} software library for the real-time selection of an avoidance trajectory. Experimental results are provided as a proof that this dynamic obstacle avoidance strategy works properly.

1 INTRODUCTION

An important challenge for industrial countries concerns the renovation of industrial sites, with several objectives, such as higher productivity, improved versatility and flexibility of the production system, and improved safety for the workers, especially the reduction of injury and musculoskeletal disorders.

For this reason new robots appeared 10 years ago, such as the KUKA Light-Weight-Robot (LWR) or the Neuronics Katana arms, among others. These robots have been designed with respect to the ISO standard 10218-1 defining the safety rules to be respected by collaborative robots, i.e. robots which can be deployed close to humans. Figure 1 shows some contexts for the deployment of collaborative robots as coworkers for industial applications. We study these use cases in the context of the *Industrial Cooperative Assistant Robotics* (ICARO) project.

In the use case considered in this paper, it is mainly a manipulator executing motions close to an operator. The complete experiment involves a physical interaction between the robot and the operator, i.e. the operator executes an insertion on an object grasped by the manipulator. In the Factory of the Future, such collaborative tasks will be authorized only if safety is guaranteed for the operator, i.e. by the use of intrinsic and extrinsic behaviours to minimize collision risks, especially with the operator.

These robots are endowed with intrinsic safety behaviours; they are able to detect a collision from torque measurements made in real time on every joint. This is not sufficient, since an operator could be injured by collisions before the robot stops effectively. Therefore it is necessary to detect from extrinsic sensing, the presence of obstacles along a planned trajectory, and then to modify the trajectory online, so that these obstacles are avoided. First, using the modules available on the ROS web site (ros, 2010), it is possible to perform an experiment with the robot stopping as soon as an obstacle is detected on the trajectory, and restarting as soon as the path is free.

This paper presents a more reactive approach, integrating the ROS middleware with the reactive planner developped by Siemens PLM Software (formerly Kineo CAM), and evaluating using a dedicated setup with a KUKA arm available at LAAS-CNRS. It is also proposed to integrate the SoftMotion controller developped by LAAS-CNRS for collaborative manipulators interacting with humans, in place of the Reflexxes module available with ROS. Presented here is the principle of the planner with initial experimental results.

Section II proposes a very synthetic state of the art. Section III recalls how to use the OctoMap module in order to monitor the robot path during motion execution. It has been improved to be used efficiently for obstacle detection. Section IV describes the reac-

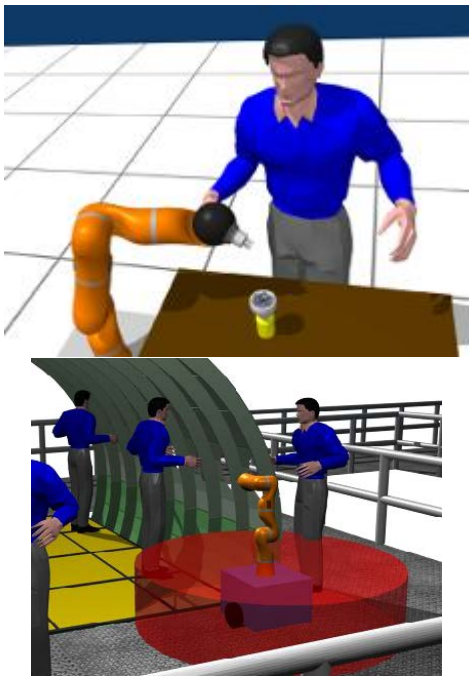

Figure 1: Some examples where co-workers execute collaborative tasks with humans, either during the learning step of an action, or for executing assembly operations in cluttered environments.

tive planner, while Section V shows how it has been integrated with the other ROS modules. Section VI proposes some experimental results. Finally section VII summarizes our contribution and speaks about our on-going works.

2 STATE OF THE ART FOR OBSTACLE DETECTION AND AVOIDANCE

Obstacle detection and avoidance is a classical topic in robotics, that has been overall studied for mobile robots navigating in cluttered environments. Concerning manipulators, generally, it is considered that the robot is alone, knows perfectly its workspace, so that paths are always generated in the free space; these hypothesis are no more possible for collaborative manipulation tasks.

Several recent works address dynamic obstacle detection and avoidance while an arm executes a planned trajectory. Flacco et al (Flacco et al., 2012) present a simple variant of a classical potential field method for safe human-robot cooperation. It enables real-time generation of repulsive commands for the robot to avoid collisions. First the concept of depth space is introduced in order to evaluate distances between the robot and possibly moving obstacles, and to estimate their velocities, directly from depth images acquired by a Kinect sensor. These distances and velocities are exploited to generate repulsive vectors that are integrated in the control law applied on the robot while executing a generic motion task. The complete collision avoidance framework, has been validated on a KUKA LWR IV executing tasks in a dynamic environments with obstacles and a human.

Yang and Brock (Yang and Brock, 2010) presented an original solution, though lacking convincing experimental validation and generic obstacle detection capabilities. It is based on a novel motion generation technique, called elastic roadmap, proposed to generate robust and globally task-consistent motion in dynamic environments. Building and following a reactive path in real time, the motion must satisfy different classes of constraints for a redundant mobile manipulator for which the end effector follows a path.

Considering our use cases about collaborative robotics, these two approaches have important drawbacks. Flacco et al reacts only at control level by trying to increase distance to obstacles; Yang and Brock provide a solution taking advantage of a mobile arm, but they do not take into account human constraints.

In our applications, the set-up is simpler, with only a fixed manipulator (without considering that this arm is mounted on a mobile platform by now); this arm has to execute a plan, and to reactively adapt this plan if an obstacle is detected. The path is directly replanned in real time.

The ROS-based robot motion control architectures make use of the Reflexxes framework (Kröger and Wahl, 2010) (T.Kröger, 2011) which allows to transform a path in an on-line time-defined trajectory. This intermediate layer allows to generate jerk-limited and continuous motions, taking into account constraints on the dynamic robot capabilities with low latencies (1ms for the low-level control loop). Our reactive planner has been firstly integrated with the Reflexxes framework.

Even if the Reflexxes framework is very efficient, it does not guarantee an error bound between the initial path and the executed trajectory. The current version of our system takes profit of the SoftMotion framework as the robot controller (Broquere et al., 2008) (Broquère and Sidobre, 2010). The SoftMotion controller has been independantly developped in order to generate motions for collaborative manipulators, i.e. soft motions more tolerable by an operator; this module was previously validated and adapted for a KUKA LWR-IV arm, as it was presented in (Zhao

et al., 2014). Our reactive planner produces a path to be executed by the robot as a polygonal line defined by a sequence of via-points. Using smoothing techniques, SoftMotion computes firstly a trajectory stopping at via-points and then smooth the trajectory near each vertex. The smoothed area is managed to maintain the moving parts inside a pre-defined tube, so respecting error bounds specified by the user. The system can adapt the kinematic bounds (velocity, acceleration and jerk) in real time making the trajectory more acceptable to humans.

3 OBSTACLE DETECTION

The Kineo^{TM} Collision Detector (KCD) provides fast and reliable collision detection, based on minimal distance computations. This module is synchronized with the OctoMap module (Hornung et al., 2013). The OctoMap is updated at 30Hz with the point clouds acquired by an Xtion PRO LIVE RGB-D camera.

In the OctoMap updating process there is a compromise between updating speed and noise: the faster the update, the easier noise appears. Has we are doing reactive planning, we want the OctoMap to be updated as quickly as possible. In order to avoid the introduction of noise in the OctoMap, we must use robust noise filters. In fact, before a point cloud is added to the OctoMap, it undergoes three filtering steps. The first step removes the points corresponding to NaN depth values, typically points at infinity or on specular surfaces, and points out of reach of the robot's arm, with which there is no possible collision. The second step filters out parts of the point cloud corresponding to robot joints. During this step, a correct hand-eye calibration is crucial as it will determine the quality of this filtering. Though most of the noise sources are now removed, a third step is necessary. Indeed, the sensor can generate spurious points at random positions which impair the reactive planning by blocking free paths.

In order to limit the apparition of such sparse outliers, we use a statistical filter (Rusu and Cousins, 2011) available in the PCL library (pcl, 2010). For all points, the statistical filter measures the distance between every point and its closest N neighbors. These data are fitted to a Gaussian. A threshold is set in function of the mean and variance of the Gaussian. Every point for which the mean distance to its neighbours is higher than the threshold, is considered as an outlier and is removed.

4 SIEMENS REACTIVE PATH PLANNING ROS PACKAGE

As part of the ICARO project, we have developed a reactive path planning package over the ROS middleware. This kws_ros_interface package (hereafter kws_ros), is built on top of the KineoWorks^{TM} software component. KineoWorks addresses all aspects of motion processes including collision-free automated path planning. It includes the most common motion types, such as joint motion, and features an advanced algorithm for detecting collisions along a trajectory, which is both fast and exact, regardless of kinematical complexity. The reactive path planning involves two packages: the kws_ros package dedicated to trajectory planning, interacts with the Kineo^{TM} Collision Detector (hereafter, KCD).

4.1 General Structure

The kws_ros package contains two nodes, one in charge of all collision-free path planning tasks and the other performing reactive control (Yoshida and Kanehiro, 2011) over the executing trajectory. Both nodes embed a representation of the scene into KCD. This one has to be defined offline in a dedicated CAD modeller, the Kite module, by providing CAD models of static obstacles and parts of the robot, together with its kinematics model. The representation is loaded at node launch and it is then enriched, for reactive purpose, by the dynamic point cloud data such as those obtained through laser scanning or optical vision sensors. KCD enables collision detection between point clouds and the rest of the environment. Each one of these nodes is implemented using a robust state machine fully driven by the ROS actionlib protocol allowing multi process communication. Both nodes update the robot position by a ROS topic and provide convenient topics to add, move, remove, attach and detach geometric parts on the fly. This allows covering a wide variety of scenarios from simple pick and place tasks to complex human collaboration tasks.

4.2 Reactive path planning

Reactive path planning is mainly performed by the controlling node which basically interacts with the path planning node and some critical nodes executing the global task, here grouped in the ICARO stack, such as the trajectory execution node. The reactive strategy is implemented in its state machine by four states described in figure 2.

WAITING state: A simple state waiting for user input.

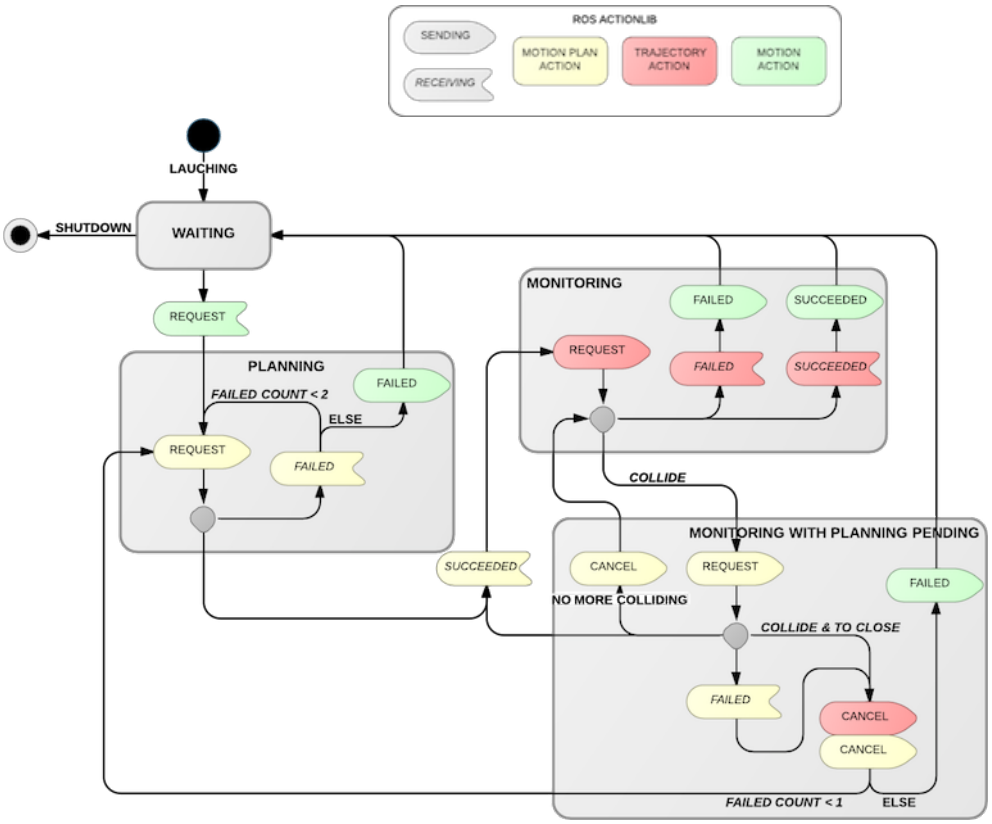

Figure 2: Controlling node state machine. Colors correspond to different actionlib types. Arrows indicate sending or receiving events.

PLANNING state: The user has requested a robot motion. The controller requests a path plan via the path planner node and waits for the result. For robustness, a fallback strategy is implemented to retry the path planning in case of failure if a predefined timeout has not yet been reached. If the path planner node is still failing, then the robot motion is canceled. Oterwise, the controller retrieves a collision free trajectory and sends it for execution via the trajectory executor node.

MONITORING state: This state is entered when a trajectory is being executed by the robot. The controller monitors the robot position to extract the remaining trajectory to follow and test it for collision in a fast way, without distance computation. If no collision is detected, it continues monitoring until full trajectory execution. If a collision is detected then it switches to the next state.

MONITORING_WITH_PLANNING_PENDING state: In this state, the controller has already detected a collision over the trajectory. Therefore, it sends a new path plan request to the path planning node with the current robot position as start configuration and the initial user goal configuration. Then it continues to monitor the remaining trajectory. Collision detection in this state is applied more accurately; it includes distance computation. If the minimal distance allowed between the robot and the environment is not yet reached, it continues to monitor and waits for the response of the path planning node. A fallback strategy is also implemented in this state in case of path planning failure. Several exit events are taken into account: if the trajectory is no longer colliding, the controller cancels the path plan request and switches back to the monitoring state; if the path planning node fails to find a new collision free trajectory, or reaches the timeout, then the robot motion is stopped and the controller returns to the initial waiting state. And finally, if the path plan succeeds on time, the current executing trajectory is cancelled and the controller switches to the monitoring state which sends the new trajectory for execution.

Since a small amount of time can be spent by the path planning node to find a new collision free tra-

jectory, the robot position will not be the same as the start configuration of the new trajectory. To avoid the robot to go backwards during the new trajectory execution, a fast collision free optimization is done on the first four configurations.

5 INTEGRATION USING THE ROS MIDDLEWARE

The ICARO project has performed meaningful experiments, related to the use cases illustrated on figure 1. These experiments involve numerous modules (planning, perception on the operator or on objects, gesture recognition, object recognition, object grasping, control...) which are integrated using the ROS middle-ware (ros, 2010). This paper is only concerned by the reactive motion control modules; so we do not present modules related to perception or sensor-based control.

The set-up can be seen on figure 6, with a Kinect sensor mounted on a pan and tilt platform above the KUKA LWR ARM. The arm is mounted on a Neobotix mobile robot, but here neither the pan and tilt platform or the robot are exploited for this paper. The calibration process has been executed offline, using (opencv, 2010) for the Kinect calibration and (pcl, 2010) in order to estimate the Kinect reference frame with respect to the arm's frame, executing known motions of the arm, while matching the arm bodies in the depth images acquired by the Kinect sensor.

A first robot motion control architecture presented on Figure 3 has been integrated and validated. It uses the standard OMPL module for path planning, and the Reflexxes framework (T.Kröger, 2011) for motion generation. The experiment presented in the next section, have been performed using the architecture presented on Figure 4, where the kws_ros_interface for the planning function and the Kineo^{TM} Collision Detector (KCD), replace the standard ROS modules. The planned trajectory has to be pre-processed for feeding the Reflexxes module in order to generate motions.

In the last version, we have replaced Reflexxes by the SoftMotion module, which generates better motions with respect to the operator acceptability; see Figure 5.

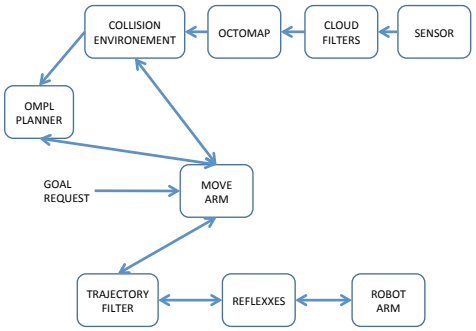

Figure 3: Architecture using available modules on (ros, 2010)

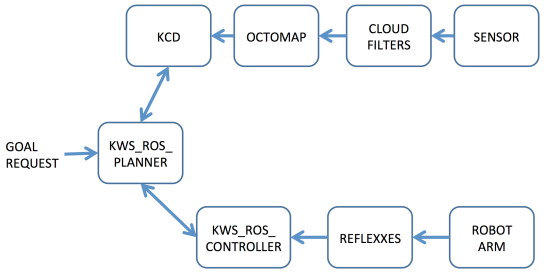

Figure 4: Architecture integrating the KCD and Kws modules with Reflexxes

6 EXPERIMENTAL RESULTS

The reactive path planning has been successfully tested for hours in different simulated environments and real scenarios, on both single computer and multi computers configuration. A video (video, 2014) illustrates some experimental results; the following figures are extracted from this video.

The KUKA arm is programmed to execute a simple trajectory: a loop between two positions A and B. An operator perturbs this process, introducing an obstacle on the path. Figure 6 presents the initial A and final B positions of this nominal trajectory.

Only two situations are illustrated here. Figure 7 shows a situation where the planner fails in finding a new path that avoids the obstacle. Therefore it is blocked until the obstacle is removed and disappears fromthe OctoMap. Nevertheless due to the latency on the OctoMap, it modifies the nominal trajectory going under a virtual obstacle, created by isolated points remaining in the map.

Figure 8 shows a replanning process. The planner finds a new path, going above the detected obstacle. Many other situations have been tested.

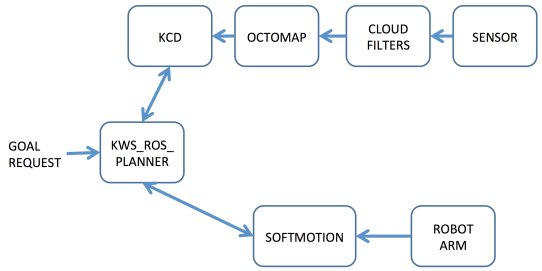

Figure 5: Architecture integrating the SoftMotion framework for motion generation

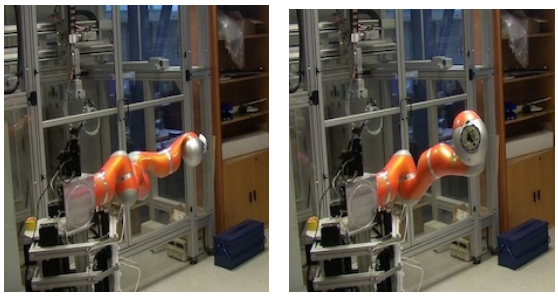

Figure 6: Experiments: execution of a nominal trajectory from an initial point (left) to a final one (right).

7 CONCLUSIONS

This paper presented a reactive planification algorithm, that has been integrated in a ROS architecture devoted to the execution of collaborative manipulation tasks between a KUKA LWR arm and an operator. Such a planner, associated with a module dealing with obstacle detection, guarantees that a robot arm could detect and avoid a human before any collision.

The detection module is built upon the OctoMap algorithm already integrated under ROS. It has been necessary to adapt OctoMap and to apply specific tunings so that (1) detected obstacles disappear from the map as soon as the corresponding actual objects are no longer on the trajectory the robot is executing, and (2) the robot is not itself detected as an obstacle.

As soon as an obstacle detection occurs, the planner module tries to find another path, continuous with the initial ones in order to reach the goal. If it fails, the robot performs as it would do by now with the OMPL module under ROS: it waits until the obstacle disappears from the map.

As a future work, the OctoMap module has to be modified in order to minimize the latency when an obstacle appears or disappears in the map. Moreover the link between the kws_ros reactive planner and the SoftMotion controller has to be refined in order to make more dynamic the obstacle avoidance.

ACKNOWLEDGEMENTS

This work has been funded by the ICARO project from the ANR CONTINT program (french Research Ministry) and by the CAAMVIS project from the AEROSAT program (french Industry Ministry and Midi-Pyrénées region).

REFERENCES

Video (2014): homepages.laas.fr/michel/documents/ReactivePlanningJido2LAAS_HD.mp4.

Open CV (2008). OpenCV: Open Computer Vision library. opencv.willowgarage.com/.

PCL (2010). PCL: Point Cloud Library. www.pointclouds.org.

ROS (2010). ROS: Robot Operating System. www.ros.org.

Broquère, X. and Sidobre, D. (2010). From motion planning to trajectory control with bounded jerk for service manipulator robots. In *IEEE Int. Conf. Robotics and Automation, Anchorage (USA)*, pages 4505–4510.

Broquere, X., Sidobre, D., and Herrera-Aguilar, I. (2008). Soft motion trajectory planner for service manipulator robot. *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) Nice (France)*, pages 2808–2813.

Flacco, F., Kroger, T., De Luca, A., and Khatib, O. (2012). A depth space approach to human-robot collision avoidance. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 338–345. IEEE.

Hornung, A., Wurm, K., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots Journal*.

Kröger, T. and Wahl, F. (2010). On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, pages 94–111.

Rusu, R. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1–4.

T.Kröger (2011). Opening the door to new sensor-based robot applicationsthe reflexxes motion libraries. In *Proc. IEEE Int. Conf. on Robotics and Automation, Shanghai, China*.

Yang, Y. and Brock, O. (2010). Elastic roadmapsmotion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130.

Yoshida, E. and Kanehiro, F. (2011). Reactive robot motion using path replanning and deformation. In *Proc. IEEE Int. Conf. on Robotics and Automation, IEEE press, Shanghai, China*.

Zhao, R., Sidobre, D., and He, W. (2014). Online viapoints trajectory generation for reactive manipulations. In *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (submitted to AIM 2014), Besanon, France*.

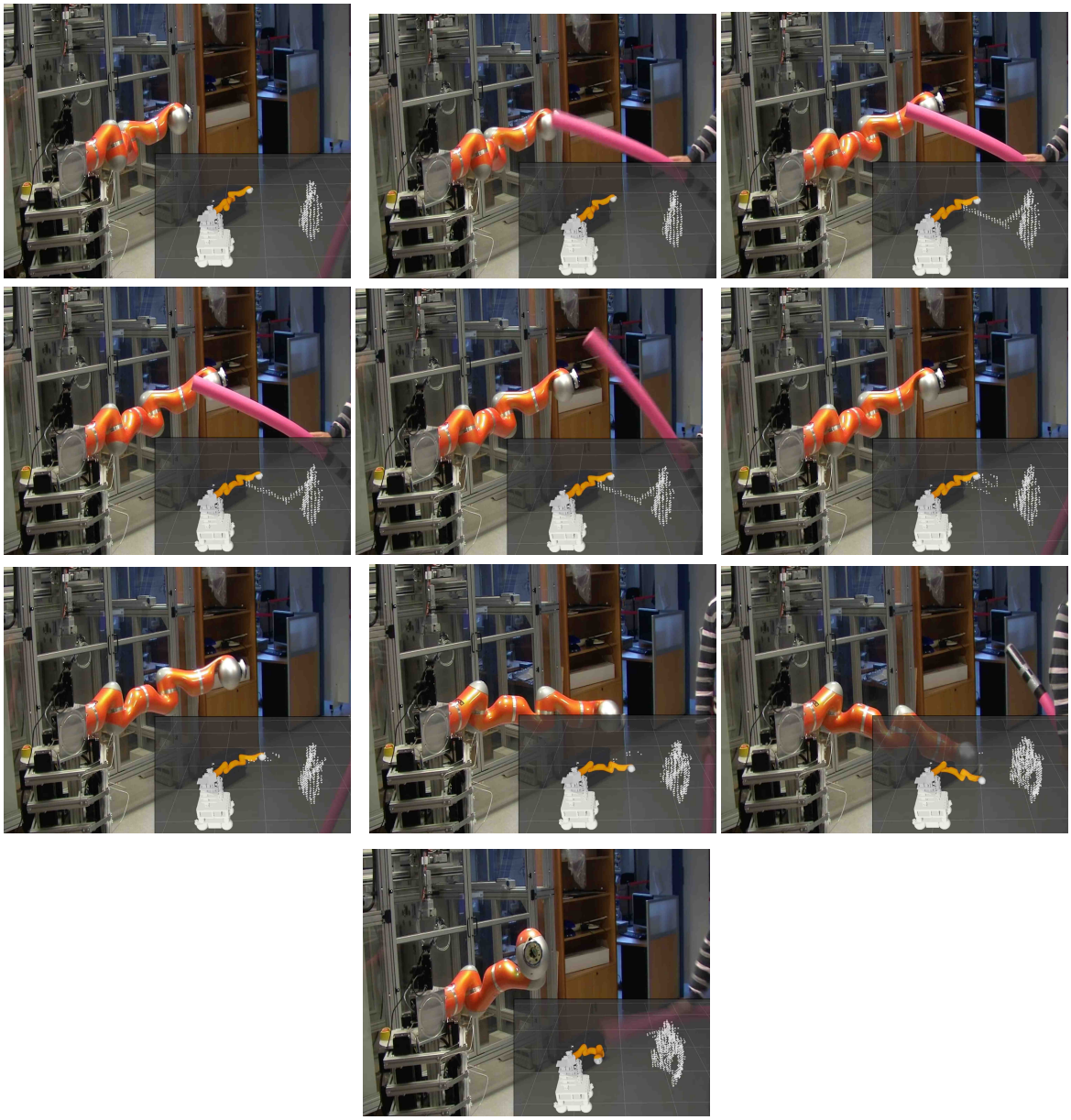

Figure 7: Experiments: the operator has blocked the arm which stops as soon as the obstacle apperas in the map. Here the generation of a new path failed; so the arm restarts as soon as the obstacle disappears from the map; nevertheless, due to a latency in the map, it avoids a virtual obstacle going under virtual points.

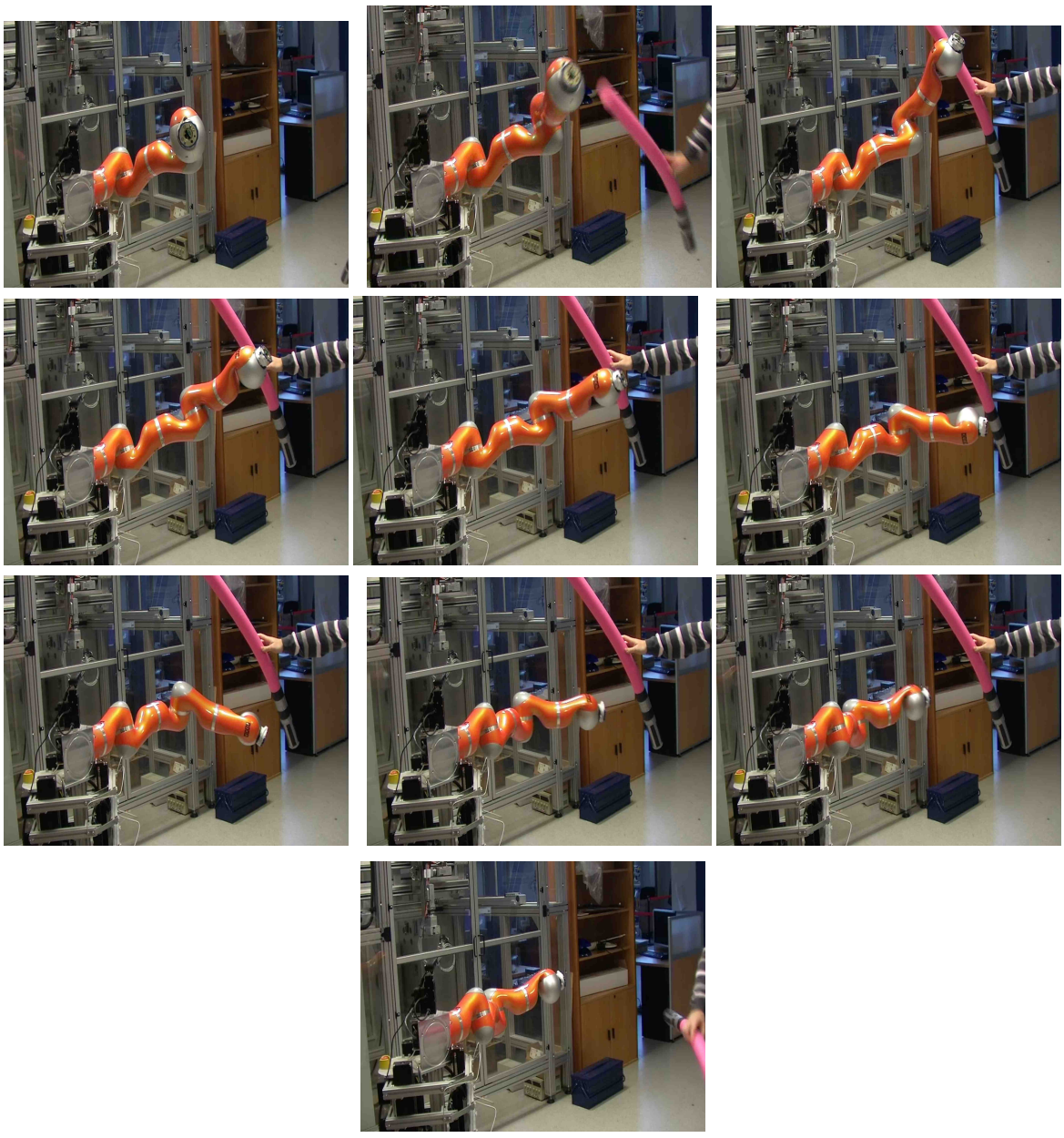

Figure 8: Experiments: a dynamic obstacle avoidance. The operator tries to block the arm, which generates a trajectory towards the goal, through a subgoal under the obstacle.