Supervised Autonomy Interface with Integrated Perception and Motion Planning for Telemanipulation Tasks

Jeffrey Chee Yong Fong+, Minh Khang Pham+, Anil Kürkcü, Jun Li, and Keng Peng Tee*

***Abstract*— In scenarios where the environment and task are known and certain, autonomous robotic manipulation can be performed. However, when an unforeseen situation arises where prior information is unknown, timely human assistance becomes very useful in accomplishing the task. In this paper, we propose a supervisory autonomy system that allows a user to assist the robot in uncertain task scenarios by specifying, through an intuitive user interface, the discrete actions (e.g. grasping) to be performed on the selected objects. The robot has partial knowledge of the environment in that it is able to segment and localize objects. This information is shared with the user to allow easy object and action specification, after which the robot executes the action on the object autonomously while avoiding obstacles. We investigated the system using live perception of real objects with simulated robot and environment.**

I. Introduction

Human-robot collaboration is an active area of research, thanks to the successful applications in areas such as robot tele-operation [1], learning [2], and rehabilitation [3]. In general, humans are superior in reasoning and decision-making, while robots function near perfect in autonomous tasks given the desired trajectory. The conjunction of these capabilities is an important step towards the desired goal of full autonomy in robots.

Existing works on human-robot collaboration systems include lower-level shared control systems and higher-level supervised autonomy systems. Shared control systems combine, at the motion or force control level, the inputs from human and robot, either in a continuous [4], [5] or switched manner [6], [7]. In contrast, higher-level systems are based on discrete commands that are translated into lower-level control by the system itself. Other approaches include learning from demonstration for the purpose of teaching motor trajectories to robots [8] and interactive policy learning [9] where human demonstration not only initializes but also improves the policy.

One of the earlier works on supervised autonomy showed how a mobile robot could autonomously navigate from one room to another with simple instructions given by the supervisor through a graphical user interface which displayed a view of the robot's environment from a camera attached to the robot [10]. More recently, [11] presented a mobile manipulation robot with a visual feedback interface to assist telemanipulation. A virtual reality interface was used to render a third person view of the scene, for the purpose of guiding the human user. In [12], supervised autonomy was combined with reinforcement learning for the purpose of teaching a software-based agent how to make a cake. The human expert guides the agent by treating rewards via supervised autonomy in a simulation framework, while the agent is carrying on its task. In [13], a supervised autonomy framework was used to enable a human teleoperator to assist the robot in localizing an object by positioning a 3D model of the object within a virtual environment of the remote scene. Other applications of supervised autonomy include providing variable ground-based control support for space-deployed robotic systems [14] and robot-assisted therapy for autism spectrum disorders [15]. Systems that are able to dynamically adjust the level of autonomy to optimize the use of resources and capabilities as conditions evolve are termed sliding autonomy systems, and have been applied to human robot coordination systems for space applications [16] and to UAV path planning [17].

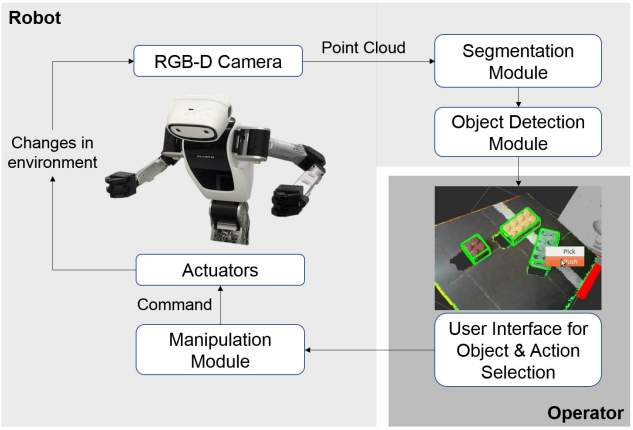


Fig. 1: Supervised autonomy system with integrated perception and motion planning

In this work, we developed a supervised autonomy system using an RGB-D camera on the robot to provide visual feedback to the teleoperator as well as to perceive the robot's environment. Unlike the above mentioned works, which do not have automatic coordination between perception and motion planning, our proposed supervised autonomy system integrates a user interface, a perception module and a manipulation module. The perception module enables automatic segmentation and localization of objects, which are then displayed to the user. Through the user interface, the user can select objects that are in the robot's environment and specify actions to be performed on the objects. The manip-

The authors are with the Institute for Infocomm Research, A*STAR, Singapore 138632.
+Equal contributions.
*Corresponding author (E-mail: kptee@i2r.a-star.edu.sg).

ulation module enhances motion planning by autonomously performing a set of discrete actions to the objects.

Such an integrated system enables a human and robot to collaborate in terms of supervision provided by the human user to the autonomous robot. The user helps the robot by specifying the task in a step-by-step manner for the robot, whereas the robot helps the human user through automatic sensing and management of the environment.

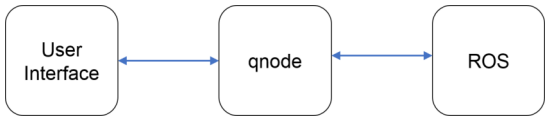


Fig. 2: Operational relationship of the GUI with ROS.

II. Supervised Autonomy Interface for Telemanipulation

An overview of the Supervised Autonomy system is shown in Fig. 1. The full configuration of the system consists of the *Olivia* robot, a customized version of the DRC-Hubo robot comprising a torso (3-DOF), head (2-DOF), and dual articulated arms (7-DOF each), a perception module comprising object detection and segmentation submodules, an intuitive user interface, and a manipulation module. All operations and communications between the modules in the system are controlled using the Robot Operating System (ROS).

During the operation of the system, an RGB-D camera (Asus Xtion Pro) mounted on Olivia provides point cloud data for the system. The segmentation module uses the dominant tabletop surface to segment tabletop objects into individual point cloud clusters. This pre-processed point cloud data is then transferred to the object detection module in which, we establish the orientations of the objects and visualize bounding boxes surrounding the objects. All the essential data generated is then displayed on the user interface, which uses RViz, a 3D visualization tool for ROS, and works with mouse device inputs, thereby providing a user-friendly graphical interface. Subsequently, the user interface relays user-specified pose goals to the manipulation module, which plans and executes collision-free joint trajectories on Olivia, allowing the robot to achieve the tasks indicated by the user.

A. Perception Module for Object Segmentation and Localization

The Perception module allows the robot to segment and localize objects in its field of view while providing visual information for the user to make selections and specify actions to be performed on the objects. This module comprises two submodules, namely a point cloud segmentation module for object and table segmentation, and an object detection module able to create oriented bounding boxes around object clusters. A coverage of these submodules can be found in the following two sections.

1) Point Cloud Segmentation Module: The point cloud segmentation module utilizes the *Organized Multi-Plane Segmentation* pipeline from Point Cloud Library (PCL) to identify all possible planes in the scene. In this pipeline, an organized point cloud normal to the raw point cloud is first estimated using integral images [18]. A connected component analysis [19] is then performed on the input organized normal point cloud. This process extracts and labels disjoint and connected components in the point cloud, allowing for the identification of regions that are planar. The different planar regions are stored in a cluster, with the region of largest size (i.e. dominant plane indicating tabletop) occupying the first index of the cluster.

Object segmentation is performed in a three-dimensional polygonal prism space generated just above the segmented tabletop. Consequently, all points that lie inside the prism will be segmented while the outliers ignored. Euclidean cluster extraction is performed on the inliers to obtain the various object indices needed for further processing in the object detection module.

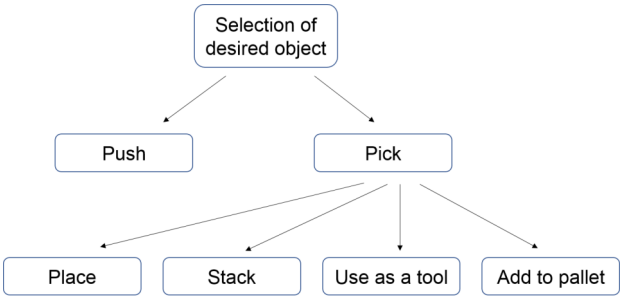


Fig. 3: Action choices when an object is selected.

2) Object Detection Module: The object detection module uses principal component analysis (PCA) to identify the principal components of each object cluster so as to estimate the orientation of the bounding box encapsulating the cluster.

Due to the nature of RGB-D cameras, point cloud clusters are usually not a full representation of the object in its entirety since the camera is limited to a single view and the rear of the object is obscured. Therefore, the estimated principal components tend to deviate from actual ones, resulting in the orientations of the bounding box and the tabletop becoming mutually exclusive, which is not logical in the context of this discussion. To overcome this problem, we project all the points in the point cloud along the normal of the tabletop onto the table. A pseudo two-dimensional PCA is then performed on the projected point cloud. In this way, the third principal component of the object cluster will be identical to the normal of the tabletop (same z-direction). With the principal components determined, the maximum and minimum points in the new x, y and z directions are determined. These points are then used to determine the centroid of the object.

Let $\mathbf{C}$ be a three-dimensional vector representing the center point of the object in a 3D space with respect to the

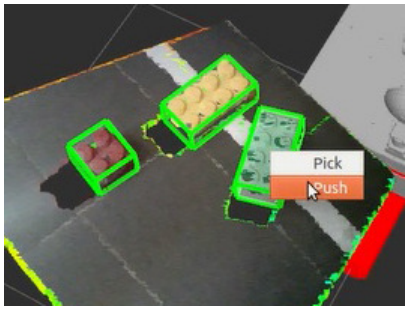


Fig. 4: Snapshot showing dropdown menu for action selection when the user right-clicks on an object on the screen.

Fig. 5: Pop-up window which probes users for sub-action selection with different choices.

world frame. It can be evaluated by utilizing **M** and **N**, the derived minimum and maximum points of the object in the three axes respectively:

$$\mathbf{C}_\star = \mathbf{M}_\star + \frac{(\mathbf{N}_\star - \mathbf{M}_\star)}{2} \tag{1}$$

where $\star$ denotes x, y, or z.

A new `tf` frame is added at the object's center point and with the orientation containing the results of the PCA. All the new data concerning the objects are utilized for visualization using the RViz visualization and interactive markers.

B. Intuitive User Interface for Object and Action Selection

Qt Creator was selected as the Graphical User Interface (GUI) Designer due to its ease of use and high compatibility with ROS and RViz. The GUI is a customized implementation of the ROS package and incorporates a specialized constituent called *qnode* linking the Qt UI with ROS, allowing the adaptation of the GUI to operate in a ROS environment. Fig. 2 summarizes the operational relationship between the frontend UI and ROS.

Using the designed GUI, users select objects to be manipulated by clicking on the interactive marker covering the desired object, prompting two main actions in a menu for the user to select, `Push` and `Pick`. If `Push` is selected, users will be allowed to drag the interactive marker representing the selected object to a desired location to be pushed to. The robot will then proceed with the action after the user confirms the action procedure. If `Pick` is selected, the robot first proceeds with the action and follows up with a list of follow-up actions, such as `Place`, `Stack`, and `Use as a tool`, for users to select. Fig. 3 illustrates the action choices available.

In the event that an error (e.g. object is out of the robot's reach) is encountered during the motion planning or execution process, the system reverts to its previous state, and a separate window appears to warn the user about the detected error.

C. Manipulation Module

The manipulation task is handled by a separate manipulation module to maintain the modularity of the system. The manipulation module expects the object information and a task code containing the desired action indicated by the user to execute the action and feedback the result to the user interface. Object information comprises of the objects bounding box size $\mathbf{S}(s_x, s_y, s_z)$ in the object `tf` frame (OBB_i). All objects in sight have a unique frame handled and managed together with the robots base frame by ROS `tf`-transform. The object pose with respect to the base frame is represented by the transformation matrix:

$$T_{obj} = \begin{bmatrix} R_{obj} & p_{obj} \\ 0 & 1 \end{bmatrix} \tag{2}$$

The task code identifies the action to be performed among the predefined set of actions. This set of action varies upon the current state:

- Action: `pick`, `push`, `undo`
- Action with tool: `push with tool`, `stack`, `place`, `undo`

RRT Connect [20] was used to generate the path. The robot always returns to a ready position after start-up and before listening to a new command, where motion planning and control are triggered by a call-back mechanism. Reversion to the ready position was adopted because it is easier to reach the goal poses from such a position as well as to prevent any obstruction to the vision. In addition, determination of utilizing which end-effector (left_hand or right_hand) is contingent upon the position of the selected object based on the robot's frame.

At the start of each action, a virtual planning scene is set up with objects built from the object information passed in by the detection module. The scene includes the table surface as well as other objects to enable collision avoidance during action planning. Subsequently, the end effector goal pose, with respect to the base frame, is given by:

$$T_{ee} = T_{obj} T_{offset} \tag{3}$$

where T_{offset} is the object-to-hand transformation matrix.

There are two modes of motion, namely the joint space planning and the compute_Cartesian_path modes, that are determined by the nature of action. `Pick`, `Place` and `Stack` uses the joint space planning mode which finds the path minimizing the total joint displacement. `Push` and `Use as a tool` operates near the table, so we use compute_Cartesian_path mode which ensures a straight translational motion of the end-effector. If Moveit fails to generate a plan to reach the goal, the robot will move the end-effector along a minute and random trajectory to create a small disturbance and attempt to generate the motion plan again. If planning continues to fail, no action is executed. In

the case where there are interruptions during the execution of the motion plan, the robot will stop its motion. In both cases, the manipulation module will return the robot to its ready state and publish a Fail message to the perception module which prompts the user. If the motion is planned and executed successfully, a success message will be returned.

The motion plan is visualized using RViz UI and its output is simulated with Gazebo.

III. Experiment Results

Experimental results were successful in the Gazebo simulation environment, with the robot able to perform a wide range of actions with the objects in the scene. The GUI provides a clean interface design in a three-dimensional visual space while RViz interactive markers were used extensively in this experiment, allowing users a comfortable level of involvement in directing the robot to complete its tasks. Dissemination of user instructions through concise methods such as dropdown menus were used. Figure 4 shows an organized dropdown menu after desired object is selected.

The system incorporates a parallel evaluation framework where users can rely on both the error log channel for any error/failure notifications and the RViz UI for visual verification. This framework ensures that the users' experiences are consistent with the effects of their actions.

Executable actions including pushing, picking and using objects as tools among others were developed as part of the manipulation aspect of the experiment.

1) Push: With the GUI, users will provide input by clicking on the desired object and selecting push option in the dropdown menu. The User Interface will provide the user with a draggable interactive marker for determining the final location. This information is used to control the robot end-effector to move directly above the object, grab it, push it along the table and release when finished. The robot then returns to its ready position and update the state to the UI. Figure 6 shows a snapshot of the push action process.

2) Pick: Another action that we have developed is the pick action and it was designed to revolve around the idea of utilization of the picked object by the robot and was developed into further sub-actions such as place and stack, to specify a couple. There are slight differences in the procedure of this action as compared to the push action that we have discussed previously. After the user selects the object to be picked up and confirms from the dropdown menu, object information is sent to the control unit to plan a path for the end-effector to reach directly above the object top surface, grasp it and return to ready position. If the action is successful, an internal message is sent to update the state and enable sub-actions under "Tooled" state. Figure 5 shows the pop-up window which probes users for sub-action selection while Figure 7 shows snapshots of the pick action procedure. The sub-actions that we have developed are discussed in the following sections.

3) Place: The `Place` sub-action allows users to control the robot to place the picked object at any feasible location and orientation on the tabletop in the scene that is within the robot's reach. Through the GUI, users will confirm the end location and orientation of the object to be placed using the draggable interactive marker which appears after the user selects place in the sub-action selection pop-up window. End-location information will be sent to the control unit, which is at the "Tooled" state, to plan the path for the end-effector to reach the desired location and to release the object. Figure 8 contains snapshots which illustrates this process.

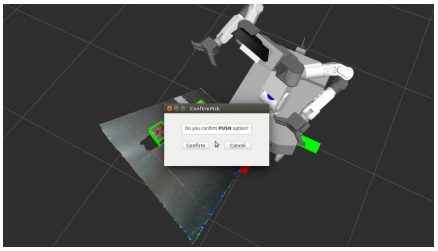
4) Stack: The `Stack` sub-action allows users to stack the picked object onto another object in the scene. Perception of the robot reactivates, allowing for the segmentation of objects in the scene and for the users to select the desired object to be stacked onto. Users can choose between 2 orientations in which the picked object should be stacked onto, each perpendicular to the other. Upon receiving the required information as stated above, the control unit will plan the path for the end-effector to reach the selected object with the respective orientation and release the picked object. Figure 9 contains snapshots of the entire process.

5) Use as a tool: The `Use as a tool` sub-action is a highly expandable sub-action due to its genericness. The tool previously picked up is merged into robot's end-effector for extended reach and interaction. In this experiment, we assumed that the picked object is used to push another object in the scene. The robot is able to push another object to any feasible location on the tabletop within the robot's reach. The highly general nature of this sub-action also provides flexibility and means that it can be further developed in future studies. Figure 10 shows some snapshots of the sub-actions being performed in the User Interface space.

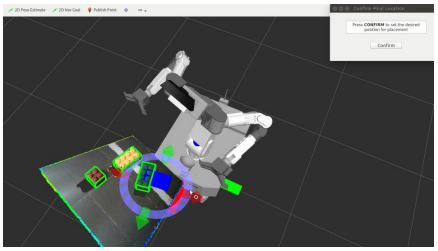
6) Undo: The `Undo` action is a special feature enabling the user to undo any revertible action previously executed. The current reversible action set includes pushing, picking, placing and stacking. During any of such action, the user can stop the process and revert the current action. A successful `Pick` action with `Undo` would result similarly to a `Place` and vise versa. Sometimes the reverse action can fail due to unforeseeable reasons. In this case, since the previous action was stopped, the robot would treat it as a normal fail action and prompt the user for further request. Figure 11 shows some snapshots of the undo performed while executing `place` action in the User Interface space.

IV. Conclusion

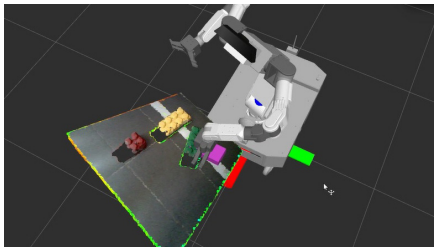
A supervised autonomy system that integrates user inputs, such as placing an object onto another, through a user interface has been studied in this paper. The automatic perception allows a better understanding of the target object that the user is selecting compared to previous studies. The automatic planning removes the bulk of the responsibility of coordinating the robot end effector from the user. These features enable a human-robot collaboration, where the user specifies the task in a sequential manner to the robot and the robot automatically manages its interaction with its environment. As a future work, we would like to present this study on a real robot and to explore elements of Artificial Intelligence through automatic action suggestions on the UI.



(a) Confirmation window which appears after user selects push action on the desired object.

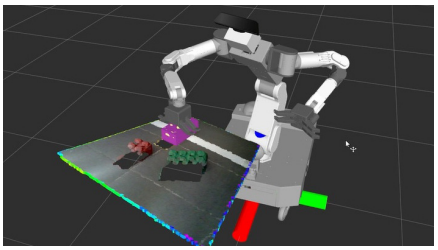


(b) Users are able to drag the interactive marker to select the end location and orientation for the push action.

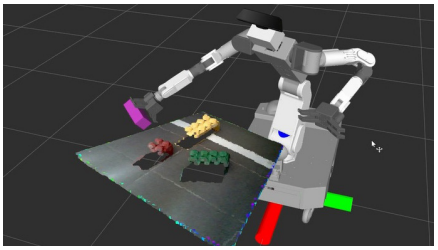


(c) Execution of push action by the robot towards the user-directed end location and orientation.

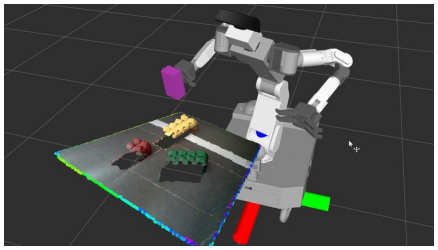
Fig. 6: Snapshots of user-directed `Push` action using the supervised autonomy interface.



(a) Movement of end-effector to reach directly above the object top surface.

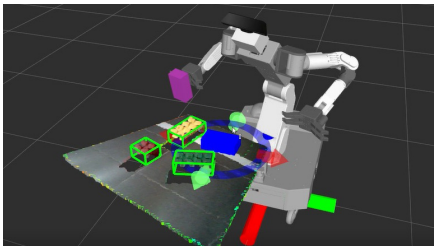


(b) Movement of end-effector back to the ready position while grasping the object.

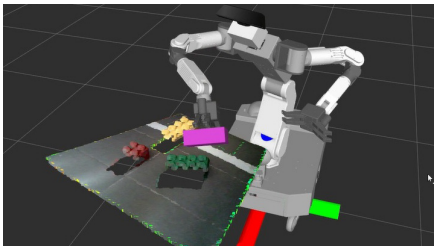


(c) Return of end-effector back at the ready position and activation of "Tooled" state.

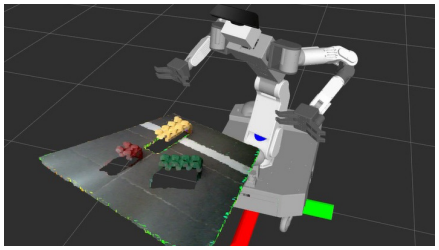
Fig. 7: Snapshots of user-directed `Pick` action using the supervised autonomy interface.



(a) Determination of end location and orientation of the object using interactive markers.

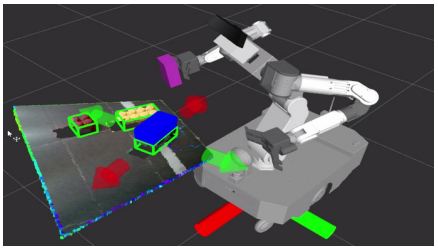


(b) Movement of end-effector towards end location while grasping the picked object.

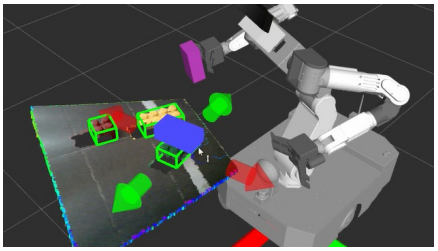


(c) Return of end-effector back at the ready position and activation of "Tooled" state.

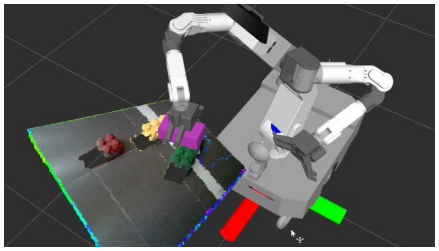
Fig. 8: Snapshots of user-directed `Place` sub-action using the supervised autonomy interface.



(a) Initial suggested configuration for stacking action.

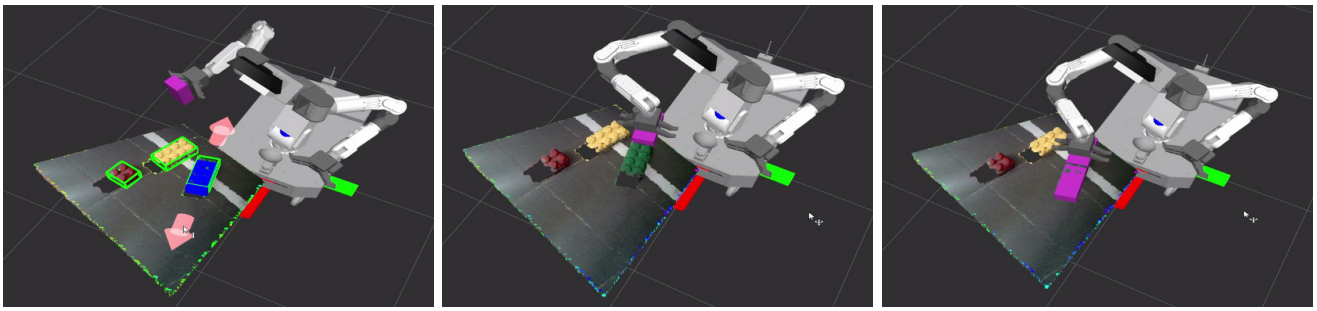


(b) 90-degree-rotated configuration for stack action.



(c) Movement of end-effector towards selected object.

Fig. 9: Snapshots of user-directed `Stack` sub-action using the supervised autonomy interface.

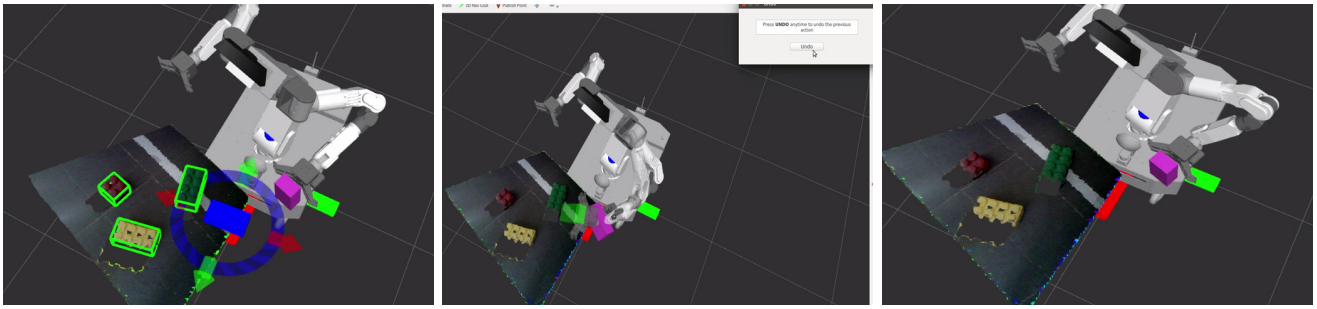


(a) Determination of end location of the object using interactive markers.

(b) Movement of the combined end-effector towards behind the target.

(c) Straight translational motion of the combined end-effector to push the object.

Fig. 10: Snapshots of user-directed `Use_as_a_tool` sub-action using the supervised autonomy interface.



(a) Determination of final pose for placing object.

(b) Human interruption with undo command mid-execution.

(c) Cancellation of place action an return to previous state

Fig. 11: Snapshots of user-directed `Undo` action using the supervised autonomy interface.

REFERENCES

[1] R. Riener, A. Duschau-Wicke, A. König, M. Bolliger, M. Wieser, and H. Vallery, "Automation in rehabilitation: How to include the human into the loop," in *2009 World Congress on Medical Physics and Biomedical Engineering*, Olaf Dössel and Wolfgang C. Schlegel, Eds. 2010, pp. 180–183, Springer Berlin Heidelberg.

[2] A. Ghadirzadeh, J. Btepage, A. Maki, D. Kragic, and M. Bjrkman, "A sensorimotor reinforcement learning framework for physical human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2682–2688.

[3] S. Hirche and M. Buss, "Human-oriented control for haptic teleoperation," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 623–647, March 2012.

[4] Y. Li, K. P. Tee, R. Yan, W. L. Chan, and Y. Wu, "A framework of human-robot coordination based on game theory and policy iteration," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1408–1418, 2016.

[5] K.P. Tee and Y. Wu, "Experimental evaluation of divisible human-robot shared control for teleoperation assistance," in *IEEE Region 10 Conference (TENCON)*, 2018.

[6] J. R. Medina, M. Lawitzky, A. Molin, and S. Hirche, "Dynamic strategy selection for physical robotic assistance in partially known tasks," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1180–1186.

[7] P. Evrard and A. Kheddar, "Homotopy-based controller for physical human-robot interaction," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, 2009, pp. 1–6.

[8] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal, *Robot Programming by Demonstration*, pp. 1371–1394, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[9] Sonia Chernova and Manuela Veloso, "Interactive policy learning through confidence-based autonomy," *J. Artif. Int. Res.*, vol. 34, no. 1, pp. 1–25, Jan. 2009.

[10] G. Cheng and A. Zelinksy, "Supervised autonomy: a framework for human-robot systems development," *Autonomous Robots*, vol. 10, pp. 251–266, 2001.

[11] M. Schwarz, M. Beul, D. Droeschel, S. Schller, A.S. Periyasamy, C. Lenz, M. Schreiber, and S. Behnke, "Supervised autonomy for exploration and mobile manipulation in rough terrain with a centaur-like robot," *Frontiers in Robotics and AI*, vol. 3, pp. 57, 2016.

[12] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, "Supervised autonomy for online learning in human-robot interaction," *Pattern Recognition Letters*, vol. 99, pp. 77–86, 2017.

[13] T. Koolen et al, "Summary of team IHMC's virtual robotics challenge entry," in *The 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct 2013, pp. 307–314.

[14] Ross Gillett, Michael Greenspan, Leo Hartman, Erick Dupuis, and Demetri Terzopoulos, "Remote operation with supervised autonomy (ROSA)," in *6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, 2001.

[15] Serge Thill, Cristina A. Pop, Tony Belpaeme, Tom Ziemke, and Bram Vanderborght, "Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook," *Paladyn*, vol. 3, no. 4, pp. 209–217, Dec 2012.

[16] Liam Pedersen, David Kortenkamp, David Wettergreen, I Nourbakhsh, and David Korsmeyer, "A survey of space robotics," 2003.

[17] Lanny Lin and Michael A. Goodrich, "Sliding autonomy for uav path-planning: Adding new dimensions to autonomy management," in *International Conference on Autonomous Agents and Multiagent Systems*, 2015, AAMAS '15, pp. 1615–1624.

[18] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," *Intelligent Robots and Systems*, pp. 2684–2689, 2012.

[19] Z. Abderaouf, B. Nadjia, and O. K. Saliha, "License plate character segmentation based on horizontal projection and connected component analysis," *World Symposium on Computer Applications & Research (WSCAR)*, pp. 1–5, 2014.

[20] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings IEEE International Conference on Robotics and Automation.*, 2000, pp. 995–1001.