Shared Autonomy for Intuitive Teleoperation

Simon Manschitz, Dirk Ruiken

2022

Preprint:

This is an accepted article published in ICRA Workshop: Shared Autonomy in Physical Human-Robot Interaction: Adaptability and Trust. The final authenticated version is available online at: https://doi.org/[DOI not available]

Shared Autonomy for Intuitive Teleoperation

Simon Manschitz[1] and Dirk Ruiken[1]

***Abstract*— Shared autonomy can be a means for combining the strengths and alleviating the weaknesses of two heterogeneous agents. In a teleoperation system with shared autonomy, a robotic system can take over the control from the human operator in certain situations, for instance when the operator has issues controlling the robot due to diminished depth perception. However, it is often difficult to decide when and how to take over control. In this paper, we introduce a virtual reality based shared autonomy teleoperation framework that provides support for the human operator during object interactions, e.g., for pick and place tasks. The framework allows for a better understanding of the current situation by providing visual and haptic cues to the operator. Additionally, based on the estimated intention of the operator, the system can autocorrect the operator's commands to semi-autonomously guide the robot towards the current goal when appropriate. By retaining the operator's sense of agency, the system increases the operator's trust in the system.**

I. Introduction

Teleoperation can be utilized for remotely controlling a robot, for instance in environments that are too dangerous for a human. The main downside is that teleoperating a robotic arm is difficult due to the difference in the embodiment of the robot and that of the human operator. The degrees of freedom, the limb lengths, the work range and physical limits of the robotic arm are usually different from that of the human operator and therefore controlling the robot requires extensive training of the operator and often knowledge about robotics in general and/or knowledge about the internal structure of the robotic arm. An alternative to teleoperation is the autonomous execution of a task by a robot, which requires little to no effort by an operator. However, due to the unpredictability of the real world, autonomous task execution is usually restricted to a very limited set of simple tasks and requires a lot of programming. Shared autonomy is a teleoperation paradigm which aims at finding a compromise between the two extremes of autonomous execution and pure teleoperation [1]. The idea is that in phases of a task which can be easily automatized the robot takes over the control, while in other phases the human operator fully or partially controls the robot. Taking over the control can be done automatically or based on human input. In this paper, we present a framework for such a shared autonomy teleoperation scenario. The human operator is wearing a head mounted display (HMD). The operator's hand is tracked with a Valve Index controller. The state of the fingers is tracked with a continuous button of the Index controller. Hence, a continuous value $[0, 1]$ indicates the finger state.

[1]Simon Manschitz and Dirk Ruiken are with Honda Research Institute Europe GmbH, Carl-Legien-Straße 30, 63073 Offenbach/Main, Germany simon.manschitz@honda-ri.de

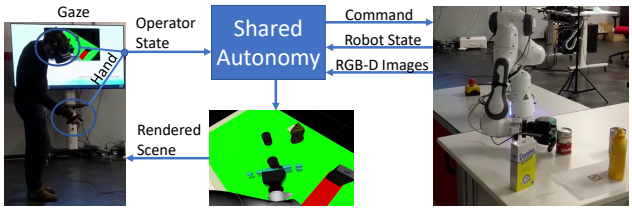

Fig. 1: The system extracts a scene from raw RGB-D images and displays it to the operator. It processes the scene together with the remaining input (operator hand state, gaze and the robot state) and generates a command for the robot. In contrast to a pure teleoperation setup, the system estimates the operator's intention and may correct the operator's input commands in certain situations.

A rendered scene is shown to the operator, who can freely walk around the scene in VR. Instead of sending the current hand state of the operator as a target to the robot as in normal teleoperation, the hand state might be processed first by an internal logic. Based on the current behavioral state (e.g., grasping or approaching an object), the hand state is altered in different ways with the aim of supporting the human operator in achieving his or her goal. An overview of our system is depicted in Figure 1 while the system itself is presented in Section III.

II. Related Work

The presented framework is comprised of different modules that are responsible for inferring the operator's intention, generating target commands for the robot based on the current behavioral state as well as providing feedback for the operator. In the following, we will discuss the related work in those areas.

Intention estimation: The aim of intention estimation is to infer the operator's intention by tracking his or her hand pose(s) and/or the gaze [1], [2], [3]. One can either infer the expected trajectory the operators wants the robot to track [4] or the intention is only estimated on a more abstract level where for instance only the object of interest or the intended action (e.g., grasp, push, etc.) is estimated. The intention estimation module we use has recently been presented in [5]. In this approach, the authors train a Hidden Markov Model to learn to infer the most likely object of interest from the gaze pattern and hand motion trajectory of an operator in a VR setup.

Haptic and Visual Feedback: In a teleoperation setting, a feedback connection of any modality establishes a bi-directional communication channel between the operator and

the robotic system. The feedback connection allows the operator to not only send commands to the system but rather to interact with it. In [6], the authors showed that haptic feedback can improve the Human-Robot agreement and user satisfaction in a shared-autonomy teleoperation setting. In [4], models are proposed for hand movement prediction to estimate future events (e.g., collisions) in virtual environments and enable real-time multi-modal feedback. Such feedback could include vibrotactile and pneumatic feedback, which can increase the accuracy of sketching in VR [7]. Similar principles could be included in teleoperation scenarios in the future. A review of haptic wearable haptics can be found in [8].

Supported Teleoperation: In [9] an approach is presented where the operator's intention is utilized for autocorrecting the commands that are send to the robot. A similar approach is presented in [10]. The approach presented in [11] is similar, but the approach is applied to a drone setting.

III. Supported Teleoperation Framework

The aim of our framework is to support the human operator in achieving his/her manipulation target. In this paper, we concentrate on pick and place tasks. One design target was that the human operator should always have the feeling of being in control of the system—retain the sense of agency. Ideally, the operator is not even aware that a support system is active. Hence, the support should always be very subtle and the system should not take over control for larger phases of a task. In the following section, we discuss the overall framework, its individual modules and some details which we think are important aspects of the system. An overview of the system is depicted in Figure 2.

A. Scene Understanding & Intention Estimation

Since manipulation tasks require physical interaction with objects in the environment, we first have to be able to detect those objects in the scene. In addition, we must guess the operator's intention for providing support for those objects. In this paper, we assume to have a database of known object models. Understanding the scene therefore reduces to detecting known objects in the scene and matching their shapes with the ones known from the database. Please note that a single object model (e.g., a coffee mug) can appear multiple times in a scene. Currently, we use the object models from the YCB-video dataset [12]. Each object is accompanied with its mesh and a primitive geometry representation (cylinder, sphere, box, or frustum). The primitive geometries are used for collision detection, since collision checks with the meshes are too costly.

For object detection and pose estimation, the scene is recorded with two static, calibrated Intel RealSense RGB-D cameras. Images are segmented with one instance of MASK R-CNN [13] per camera. Point clouds are generated for each segment and fused across cameras. Feature are extracted [14] followed by global registration [15] and fine registration [16]. Finally, pose confidence estimation (similar to visible surface discrepancy in [17]) is used to filter the results.

For detecting the operator's intention, we use the approach presented in [5]. By tracking the hand motion of the operator and his or her gaze, the system infers which action (pick, place, or moving) the operator wants to perform with which object. For picking, the system will infer the object to be grasped as well as the most likely grasp on this object the operator wants to perform. For placing, the system infers the most likely placing location as well as some parameters required for placing the object. Parameters can be an exact placing location, a placement area and/or some details about how the operator wants to place the object (e.g., upside down, on the side, etc.).

B. Grasp- & Placement Pose Estimation

A grasp pose determines how an object is grasped by the robot, while a placement pose determines how a given object is placed onto another object. For computing such poses, we decompose the given objects into sets of manifolds. For instance, a cylindrical object can be decomposed into two circular manifolds which represent the bottom and top of the object as well as a cylindrical manifold which represents the side of the object. We derived functions for mapping the different manifold types onto each other. Those functions can be utilized for computing grasp- and placement poses. For instance, by mapping the bottom circular manifold of a cylindrical object onto a rectangular manifold representing the top of a table, one can generate a placement pose which determines how the cylindrical object would be placed on the table. We model the palm and tool center point of the robot's hand with a manifold as well. Therefore, the same approach can be used for computing grasp poses and placement poses. In the context of shared autonomy teleoperation the current grasp pose is determined by the current (virtual) position of the operator's hand and the object of interest as indicated by the intention estimator. The placement pose is computed if an object has been grasped and depends on the relative pose between the grasped object and the placement location as indicated by the intention estimator.

Even if the system has already committed to a grasp or placement pose, the operator's hand movement are used to find suitable new poses on the corresponding manifolds. Hence, the operator has fine control to adjust the grasp and placement poses without needing to worry about their integrity. This strategy provides sense of agency even when the system controls part of the robot movement.

C. Behaviors

We decompose the overall task of picking and placing an object into different (sub-)tasks. The support the system provides for the operator depends and varies on the current sub-task. We switch between the different sub-tasks based on the operator's intention and the distance of the robot's hand to the objects of interest, as well as the feasibility of the current motions of the operator.

1) Approach Object: When at least one object has been detected in the scene and a pick intention for this object has been received by the system, the "approach object" behavior

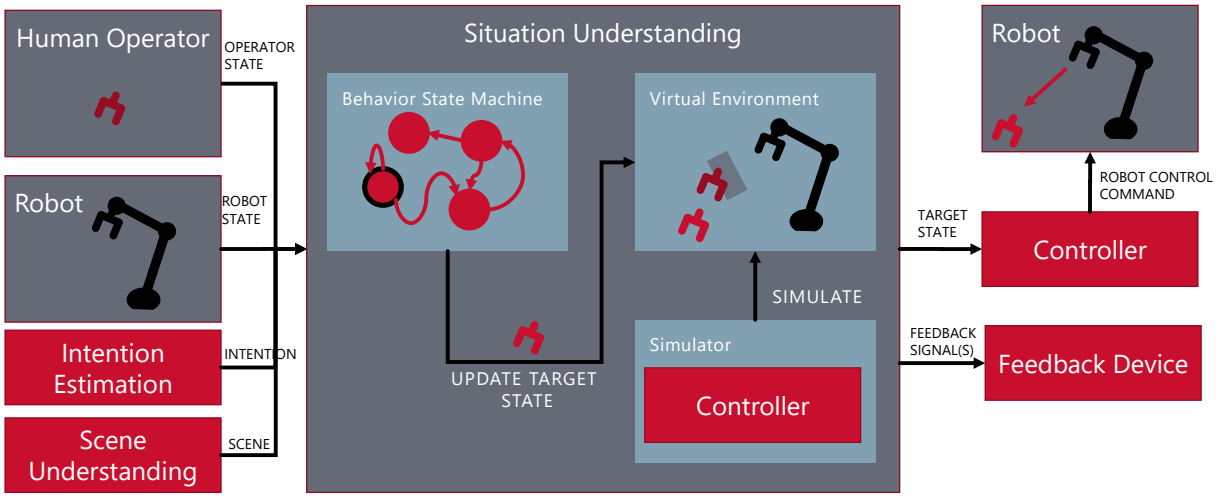

Fig. 2: Overview of our system. Based on the current input, a behavior is activated which determines how the operator's input is converted into a target hand state for the robot. Subsequently, a simple simulator uses a virtual copy of the robot controller to check if the target can be safely reached by the robot. If the target is considered safe, it is sent to the actual robot controller which then moves to the target pose. Additionally, feedback signals are generated for the operator, for instance when the robot is in a collision or the target can't be reached by the robot.

is activated. Here, the operator can freely move the robot's hand in any direction. Only in cases where the robot is about to reach its joint or speed limits the system may override those commands. In that case, the system may either stop the robot or may slightly adapt the robot's trajectory to prevent it from reaching those limits. In the background, the system computes a grasp pose for the given object, given the current robot's hand pose. The grasp pose is updated at roughly 40 Hz.

2) Snap to Object: When the distance between the robot's hand and the computed grasp pose falls below a predefined threshold, the system will take over control and automatically moves the robot's end-effector to the grasp pose. During that small period of time (usually less than a second), the operator's hand motion does not influence the robot's motion. The reason for this behavior is that we have observed that for an operator it is difficult to accurately judge relative distances between hand and objects due to the lack of depth perception.

3) Align with Object: Once the end-effector has reached the grasp pose, the operator can still move the end-effector around the object in the nullspace of stable grasp poses provided by the manifolds used for calculating the grasp poses. For instance when being snapped onto the side of a cylinder, the end-effector can move up and down the vertical axis. It can also move around the cylinder, but motions around the roll angle are blocked in order to avoid collisions with the object. Thus, the operator can move the end-effector to finely adjust the desired grasp pose and retains the sense of agency despite being helped with positioning. Moving around the object works by continuing to update the grasp pose in every step. The robot will then move to the computed grasp pose without colliding with the object. Please note that a grasp pose also contains a desired finger joint configuration. However, until the operator actually initiates grasping the object, this finger configuration is ignored.

4) Grasp Object: Once the operator is satisfied with the pose of the end-effector and presses a button on the VR controller, the fingers of the robot are closed. In this behavior, the end-effector is not allowed to move. Since we use a feed forward controller, we can simply assume the object is grasped when the fingers are virtually fully closed.

5) Align with Surface: After grasping the object, the operator can move the object around on its current support plane (e.g., table). Small movements in normal direction of the surface and rotations other than around the surface normal are ignored to keep the object in contact with the surface. Like for *Align with Object*, the system helps keeping the object in contact with the surface while offering all freedom to refine the pose on the surface. Once the operator moves the controller further away from the surface, the system will lift the object up and switch to a different behavior.

6) Unsnap from Surface: When the operator moves the hand virtually away from the support surface, the robot's hand will autonomously lift the object without colliding with other objects.

7) Approach Surface: After lifting the object, the system is in a state where the operator can again freely move around the object in the scene. Hence, this mode is similar to *Approach Object* with the only difference that the robot now holds an object in its hand.

8) Snap to Surface: When being close to the inferred placing location of an object, the system will again take over control and will automatically place down the held object such that it is positioned stably on the surface. After snapping, the system switches again to *Align with Surface* and the operator can either move the object around on the support plane to fine-adjust the placement pose or can open

the robot's fingers and move away from the object (or align the hand with it).

9) Release Object: In this state, the robot will simply open its fingers and the end-effector is not allowed to move.

D. Forward Simulator

Before sending a target pose to the real robot, we first simulate if the robot is actually able to reach that pose. Internally, we use the resolved motion rate controller to compute joint velocities, multiply them with a step size and add them to the current robot joint angles. For the newly computed robot pose, we check if it is in collision or in joint limits and if the desired target pose has already been reached. We keep on performing those steps until the motion is considered successful, infeasible, or a time limit has been reached (e.g., $500\,ms$). The advantage of this simple simulation is that it is fast enough to be computed in real-time. We also observed that it is accurate enough since the simulation is continuously executed. For the simulation we use the same controller that is used for computing the real robot's control commands.

E. Feedback Signals

In the presented framework, visual feedback is used in the form of rendering the scene and the robot in VR, while haptic feedback is utilized to signal warnings to the operator when the robot movement is altered to avoid physical limits (e.g., joint limits) and collisions of the robot, such as self-collisions or collisions with objects in the scene.

IV. Conclusion and Future Work

We presented a framework for teleoperation with shared autonomy that aims at retaining the operator's sense of agency while helping with task execution. Our framework has been used with two different robot setups: a Franka Emika Panda robot with a ReFlex gripper (see Figure 1) and with the new bi-manual Honda Avatar robot, as shown in Figure 3. Initial tests indicate improved task execution because of the assistance while sense of agency remained high. We are currently designing experiments to show the benefits of shared autonomy interaction as it combines the strengths of the robot (sensing and accurate control) with that of the human operator (logical reasoning). The aim of the experiments is to evaluate in a user study whether the presented framework improves task success and/or execution speed. At the same time, we want to investigate how the framework affects the trust of the operator into the system.

Since the system relies on object detection and the predicted user intent, it is likely that in some situations the system's support will prevent the operator from achieving the task goal. Even though the operator can turn off the support system in this situation and can try to perform the task with a purely teleoperated robot arm, the trust of the operator into the system might be reduced by this experience. It is therefore crucial to provide the right amount of support to the operator in the right situations without affecting his or her sense of agency. In future work, we want to investigate this further. Furthermore, we think it might be a good idea to control the amount of support the system provides based on the estimated trust of the operator into the system.

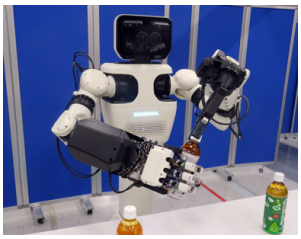

Fig. 3: The bimanual Honda Avatar Robot.

References

[1] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *IEEE International Conference on Robotics and Automation*, 1997.

[2] A. Borji and L. Itti, "Defending yarbus: Eye movements reveal observers' task," *Journal of vision*, vol. 14, no. 3, pp. 29–29, 2014.

[3] A. Haji-Abolhassani and J. J. Clark, "An inverse yarbus process: Predicting observers' task from eye movement patterns," *Vision research*, vol. 103, pp. 127–142, 2014.

[4] N. M. Gamage, D. Ishtaweera, M. Weigel, and A. Withana, "So predictable! continuous 3d hand trajectory prediction in virtual reality," in *In Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2021.

[5] S. Fuchs and A. Belardinelli, "Gaze-based intention estimation for shared autonomy in pick-and-place tasks," *Frontiers in Neurorobotics*, vol. 15, p. 33, 2021.

[6] D. Zhang, R. Tron, and R. P. Khurshid, "Haptic feedback improves human-robot agreement and user satisfaction in shared-autonomy teleoperation," in *IEEE International Conference on Robotics and Automation*, 2021.

[7] H. Elsayed, M. D. Barrera Machuca, C. Schaarschmidt, K. Marky, F. Müller, J. Riemann, A. Matviienko, M. Schmitz, M. Weigel, and M. Mühlhäuser, "Vrsketchpen: Unconstrained haptic assistance for sketching in virtual 3d environments," in *26th ACM Symposium on Virtual Reality Software and Technology*, 2020.

[8] C. Pacchierotti, S. Sinclair, M. Solazzi, A. Frisoli, V. Hayward, and D. Prattichizzo, "Wearable haptic systems for the fingertip and the hand: taxonomy, review, and perspectives," *IEEE Transactions on Haptics*, vol. 10, no. 4, pp. 580–600, 2017.

[9] C. Wang, S. Huber, S. Coros, and R. Poranne, "Task autocorrection for immersive teleoperation," in *IEEE International Conference on Robotics and Automation*, 2021.

[10] C. González, J. E. Solanes, A. Muñoza, L. Gracia, V. Girbés-Juan, and J. Tornero, "Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback," *Journal of Manufacturing Systems*, vol. 59, pp. 283–298, 2021.

[11] M. K. Zein, M. Al Aawar, D. Asmar, and I. H. Elhajj, "Deep learning and mixed reality to autocomplete teleoperation," in *IEEE International Conference on Robotics and Automation*, 2021.

[12] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS)*, 2018.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE international Conference on Computer Vision*, 2017.

[14] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[15] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*, 2016.

[16] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[17] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *European Conference on Computer Vision*, 2016.