An affordances based approach to assisted teleoperation

Alessandro Graziano[1], Emanuele Ruffaldi[1], Carlo Alberto Avizzano[1]

Abstract— **The introduction of teleoperated robots in scenarios in which human manipulation dexterity does not have to be limited by the teleoperation system is one of the greatest challenges in the field of telecontrol. In this paper we propose an approach for capturing and transferring human manupulative skills to a robotic manipulator that is able to supervise the teleoperation process by means of a proactive support that introduces elastic motion constraints around specific motion primitives. By exploiting not only information about motion, but also the relationships between the manipulator and the objects, in the form of affordances, the system is capable of assisting the human operator in order to improve the teleoperation task performances. We test the algorithm with a Microsoft Kinect RGBD sensor that allows human body pose tracking and the semi-humanoid Baxter Robot by Rethink Robotics as teleoperated platform.**

I. INTRODUCTION

Teleoperated robotic systems are nowadays widely employed in several application fields ranging from spatial to surgical and assistive applications in general.
The employment of such systems solves issues related to human security, as in the case in which there is the necessity to operate in hazardous environments; teleoperation is also helpful every time there is the need to transfer particular skills of a human to a remote environment, as in the case of DARPA's robotic challenge in 2015. DARPA's challenge demonstrated that tele-guided humanoid robots are the key to tackle the need to transfer human manipulative skill to a remote place. Nevertheless the challenge showed that such systems are very difficult to guide and they require a specifically trained and very skilled operator [1]. In these cases we can use the term expert-in-the-loop and not the classical human-in-the-loop one. The necessity of an expert doesn't only extend the training time to get such a system fully-working but also introduces risks about robustness because the difficulty to independently control all the degrees of freedom of the robot increases the risk of executing a wrong and potentially dangerous motion.

This paper tackles the teleoperation problem from a high level methodological viewpoint and it doesn't address the classical stability and predictive theory to solve problems related to time delays on communication channels [2]. The main objective of this work is thus to develop a high level software supervisor capable of selectively adding constraints that support the human operator in executing a specific teleoperation task. Such additional contraints are helpful in all the cases where the precision of the overall teleoperated system is not enough to guarantee fine grained object manipulation. We demonstrate that free pose control of the remote robot reduces the precision in performing tasks for which fine grained motion is required, like grasping an object whose geometry requires a very good alignment to be successfully picked. The rest of the work is organized as follows: section II reviews the state of the art concerning Kinematics mapping, Virtual Fixtures and Human action recognition; in section III the proposed system model and details about the supervisor architecture are presented; in section IV the tools employed for the experiments are described; in section V experiments and results are presented.

[1]All authors are with Scuola Superiore Sant'Anna, TeCiP Institute, PERCRO. Email: n.lastname@santannapisa.it

II. RELATED WORK

A. Kinematics mapping

A fundamental issue of robot telecontrol is the mapping between the master and the slave kinematic chains. This issue is simply solved if the master and the slave have the same kinematics, in fact in this case a joint-to-joint mapping solves the problem. In the case where the two kinematic chains are different, an inverse kinematics mapping is usually used, as in our previous work [3]. This approach requires that the inverse kinematics is repeatedly solved in order to find the joints configuration of the slave robot that minimizes the error between the relative positions of the master and slave end-effectors with respect to their base frames. Different approaches can be also found in the literature that don't employ explicit Jacobian based inverse kinematics schemes, but they adopt machine learning techniques. Stanton et al. [4] propose a machine learning based method that, by capturing a dataset of corresponding robot-human motions, trains a neural network that allows to generalize human movements and substitute classical inverse kinematics algorithms. However, such approach doesn't allow to predict system performance in terms of mapping error for all human movements and there is no simple way to explain its convergence results.

B. Virtual Fixtures

In the field of teleoperation, additional motion constraints as collaborative control strategies are called "Virtual Fixtures" or "Active contraints" [5]. These strategies are usually implemented by a form of admittance and/or impedance control that simplify the teleoperation task by limiting the controllable robot motion to a particular task-specific pathway. One of the first papers proposing such control concept is [6], in which the authors assert that augmenting the haptic feedback given to the master, and so limiting the motion of the slave according to prescribed performance criteria, substantially improves task performances in terms of execution

time, mental effort and motion errors. Another attempt to introduce constraints in cartesian motion of a telecontrolled robot to improve teleoperation performances is proposed by Funda et al. [7], it uses an augmented Jacobian formulation in the inverse kinematics mapping function between the master and slave of the teleoperation system in order to enforce absolute contraints imposed by physical limitations. A more recent approach is proposed by Casavola et al. [8] that introduces a "Command Governor" in a bilateral teleoperation system that, using a predictive model of the remote process, modifies the reference given to the closed loop controller according to a policy that avoids violation of constraints, like the tracking error and the maximum contact force that can arise between the robot end-effector and the environment.

C. Visual based human action classification

Common approaches tend to preprocess frames to be classified in order to extract relevant features that are afterwards used to classify and label the action [9]. Most of the approaches in the literature are based on algorithms that are trained on big datasets containing relevant cases to classify. An important issue is the intra-class variatibility, indeed different humans do the same action in different manners with different speed and also with different orders of sub-activity that compose the overall activity. An important topic of action classification concerns the relationships existing between humans and the surrounding objects.[10]. A recent approach proposed in [10] models the human sub-activity and its relationship with the objects as a Markov Random Field (MRF), then it proposes a machine learning algorithm formulated as Structural Support Vector Machine (SSVM) that is trained with a big set of features extracted from a dataset of labeled RGB-D videos. Among the used features there are the human joints positions (extracted with the machine learning algorithm proposed in [11]) and the object trajectories (extracted with a visual object pose estimation algorithm). The resulting algorithm is capable to automatically label a video in terms of sub-activity and object affordances. Li et al. [12] employs an action graph that models the dynamics of human motion and the body posture is assumed to be a bag of 3D points. The algorithm is able to classify a set of 20 different actions. Papadopulos et al. [13] uses a Random Markov Model to perform classification starting from motion data provided by skeleton tracker proposed in [11]. Finally [14] proposes the use of dynamic time wrapping to preprocess human motion data and perform actions classification.

III. PROPOSED APPROACH

In this section we present our approach. In particular we first expose the high level system modeling, then we describe the algorithms used to solve issues related sub-tasks like the visual based tracker for both human and objects and the kinematic mapping function.

A. System model

The considered scenario comprises a human operator whose movements are estimated and mapped to a robotic manipulator. The robot is equipped with a gripper that is controllable by the human through a wearable interface. We model the remote environment, comprising the telecontrolled robot and the objects in the surroundings using an oriented graph. Each node represents either the robot or an object. The edges between the robot and the objects represent the *affordances*, a set of labels describing a set of discrete actions the robot can perform interacting with an object. The edges between the objects represent the way the robot can interact with an object through another one, hereinafter mentioned as *object affordances*, e.g. open a lock with a key, fill a glass with a bottle, clean a table with a sponge. Note that the objects that have a state, like the lock, are considered as multiple objects, each for each possible state it can assume. This assumption is made to manage different affordances that an object can have depending on its particular state, e.g. a lock can be openable if it is closed and closable if it is open. Subsequently we implicitly make the assumption that the object recognition algorithm is able to recognize the particular state in which an object can be. The graph is automatically built starting from two sets of predefined labels for each object that can be found in the scene: a set containing the actions that can be done by a particular entity and a set containing the actions that can be performed on an entity. The two sets are here defined $actions_X$ and $slots_X$ where the subscript X indicates a generic object. The graph always contains a node representing the robot and a node for each object in the scene. An edge connects two nodes A and B, going from A to B, if the set $slots_B$ contains an element that matches with an element in $actions_A$. The robot only has the $actions_{ROBOT}$ set because it is assumed that no action can be executed by an object versus the robot.

B. Supervisor state machine

Once the graph explaining the relationship between the entities in the remote environment is built, it is automatically translated in a state machine that will represent the supervisor controller of the teleoperation. In particular such a state machine will have an initial state called *FREE MOTION* that represents the absence of any type of constraint on the motion of the robot. Then for each edge coming from the robot node in the graph, a new state is added to the state machine called $ACTION_X$, representing the presence of a supervised interaction between the robot and the object X that is basically the task of reaching a preferred grasping position. From the $ACTION_X$ state the state machine can go in another state called $FREE\ MOTION_X$ representing an unconstrained motion of the robot with the object X grasped by its end-effector. For each $FREE\ MOTION_X$ state there is an additional state for each edge coming from the object X node. Such state is called $INTERACTION_{X,Y}$, where Y indicates the generic target node of the edge coming from node X. When in $FREE\ MOTION_X$, depending on whether user wishes either to move and release the object

or to interact with another object through object X, the state machine can either return to the initial state *FREE MOTION* or go to one of the possible *INTERACTION* states, after the interaction is completed, it returns to initial *FREE MOTION*$_X$ state.

C. Active constraints

In this section we describe how the supervisor acts in the *ACTION*$_X$ and *INTERACTION*$_{X,Y}$ states. These states represent the situation in which supervisor recognized the intention of the user to interact with a particular object. We implemented active constraints in the form of modification of the reference position given to the robot. Let us introduce a reference frame, named $\{Approach_X\}$ (with X indicating the object to interact with), attached to the object. Such reference frame has its z-axis directed as a predefined preferred approaching direction and the x-y axis generating a generic plane orthogonal to z-axis. Let $\mathbf{x}_H$ be the human hand position with respect to the human torso, regardless of whether it's the left or right hand. Let us assume that the human torso and the robot torso reference frames are coherent in a way that we can consider that the reference position of the robot end-effector, denoted as $\bar{\mathbf{x}}_R$, is equal to $\mathbf{x}_H$ when the supervising state machine is in a $FREEMOTION$ state.

Differently, when in an *ACTION* state, the supervisor guides the motion inside a constrained region that conducts to a preferred grasping position with a tolerance decreasing with the distance from the object along the z-axis of its $\{Approach\}$ frame. We implemented such active constraint as a *virtual spring* that modifies the motion on the plane defined by x-y axis of $\{Approach\}$ frame, while the component along the z-axis is not modified. The modification of the $\{Approach\}$-planar components tends to constraint the motion of the robot end-effector on the z-axis of $\{Approach\}$ frame while it nears to the object.

The complete equations are the following:

$$ {}^{A}\bar{\mathbf{x}}_R = T_{A,B}\, {}^{B}\bar{\mathbf{x}}_R = \begin{pmatrix} {}^{A}\bar{\mathbf{x}}_{R,x} \\ {}^{A}\bar{\mathbf{x}}_{R,y} \\ {}^{A}\bar{\mathbf{x}}_{R,z} \\ 1 \end{pmatrix} \tag{1} $$

where the pre-superscripts in ${}^{A}\bar{\mathbf{x}}_R$ and ${}^{B}\bar{\mathbf{x}}_R$ mean that $\mathbf{x}_R$ is written in coordinates with respect to B (robot $\{Base\}$ frame) and A ($\{Approach\}$ frame) frames. $T_{A,B}$ is thus the homogeneous transformation matrix between $\{Approach\}$ and robot $\{Base\}$ frames. Pedices x, y, z represent the x, y and z components of a vector.

The constrained reference position is obtained as follows:

$$ {}^{A}\tilde{\mathbf{x}}_R = \begin{pmatrix} \alpha\ {}^{A}\bar{\mathbf{x}}_{R,x} \\ \alpha\ {}^{A}\bar{\mathbf{x}}_{R,y} \\ {}^{A}\bar{\mathbf{x}}_{R,z} \\ 1 \end{pmatrix} \tag{2} $$

where ${}^{A}\tilde{\mathbf{x}}_R$ indicates the constrained reference position expressed in coordinates with respect to $\{Approach\}$ frame.

The scaling value α is set equal to:

$$ \alpha = \log\left(1 + {}^{A}\bar{\mathbf{x}}_{R,z}\right) \tag{3} $$

this value of alpha constraints the end-effector position to be on the z-axis of $\{Approach\}$ frame while it is near the object.

The final constrained reference for the end-effector position with respect to robot $\{Base\}$ frame is obtained as:

$$ {}^{B}\tilde{\mathbf{x}}_R = T_{B,A}\, {}^{A}\tilde{\mathbf{x}}_R \tag{4} $$

A graphical illustration of the proposed constrained region is depicted in figure 1.

When in an $INTERACTION$ state instead, either the commanded end-effector position is rigidly constrained to a predefined interaction region of the object to interact with, or if the kind of interaction requires a fine grained alignment it is the same as the $ACTION$ state.

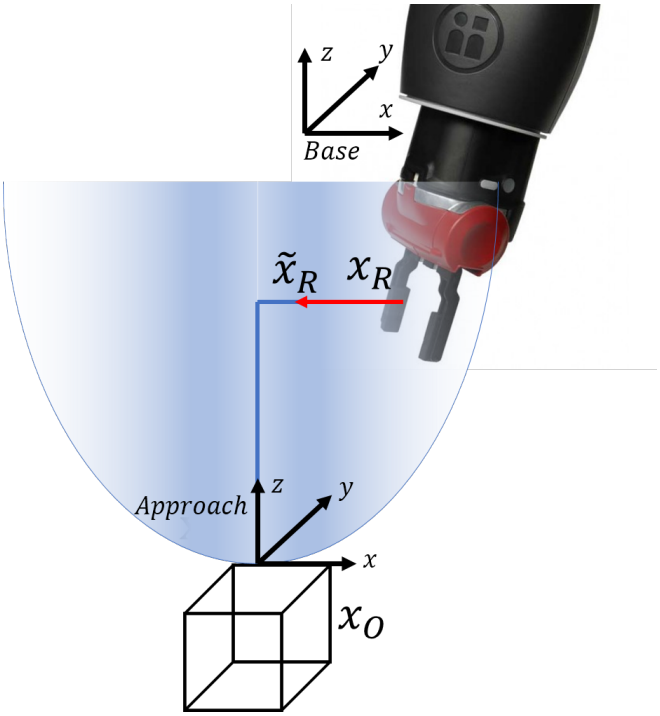

Fig. 1. An illustration of the virtual fixture: for a given grasp point of the object the virtual fixture acts as an force field that eases the vertical approaching motion for the optimal grasp, and provides a lateral force feedback. When the end-effector exits from the lateral side of the region the fixture is disabled and the state is changed.

D. Intention Classification

To tackle intention classification problem, our approach employs machine learning techniques. We propose to collect a large dataset containing relative poses of the objects with respect to an end-effector fixed reference frame and labeled commanded robot end-effector trajectories. Note that the objects can be placed in different positions with respect to the robot, hence the robot can approach an object from different directions. We define a time dependent vector of features $\phi(t)$ that contains the following data:

- the distances between the end-effector and the objects
- the time-derivative of the distance between the end-effector and the objects
- the angles between the line-of-sight from the end-effector to each object and the instantaneous velocity vector of the robot end-effector.

$$\phi(t) = \begin{pmatrix} d\left(_R, \mathbf{x}_{O_1}\right) \\ d\left(_R, \mathbf{x}_{O_2}\right) \\ \dots \\ \dot{d}\left(_R, \mathbf{x}_{O_1}\right) \\ \dot{d}\left(_R, \mathbf{x}_{O_2}\right) \\ \dots \\ \dot{d}\left(_R, \mathbf{x}_{O_N}\right) \\ \cos^{-1}\left(\frac{v_R \cdot s_{R,O_1}}{|v_R|}\right) \\ \cos^{-1}\left(\frac{v_R \cdot s_{R,O_2}}{|v_R|}\right) \\ \dots \\ \cos^{-1}\left(\frac{v_R \cdot s_{R,O_N}}{|v_R|}\right) \end{pmatrix} \quad (5)$$

where N indicates the total number of objects, $s_{R,O_i} = \frac{(\mathbf{x}_{O_i} - \mathbf{x}_R)}{|\mathbf{x}_{O_i} - \mathbf{x}_R|}$ the direction of the line-of-sight from the robot end-effector to a generic object i and v_R the velocity vector of the end-effector. These chosen features are independent from any reference frame and approaching direction. Starting from the collected data we train a neural network to classify the intention from only the first segments of the recorded trajectories, in terms of which object the controlled robot is going to interact with. This approach allows an early classification of the actions, allowing to trigger the supervising controller state transitions. The classification is filtered according to the possibilities expressed by the graph and thus by the state machine introduced in section III-B. The successful classification of an intention triggers the state machine transitions and allows to change the behavior of the system as explained in the section III-C.

E. Human and objects tracking

We employed the algorithm proposed by [11] to track human joints from RGB-D images provided by a Microsoft Kinect sensor. Shotton et al. [11] trained a Random Decision Forest (RDF) on a very huge amount of synthetically generated and labeled data to perform human body pose estimation from features related to local changes in the depth image. We used the OpenNI implementation of such algorithm.
To track objects in the scene we used Aruco fiducial markers [15]. Markers are tracked by an external RGB camera calibrated with the robot base. The calibration allowed to recover the object pose with respect to the robot end-effector useful to calculate the features vector defined in eq. 5. In a more realistic scenario we could employ object tracking algorithm and the current choice of employing fiducial markers is not affecting the concept and realization of the proposed approach.

F. Kinematic mapping

We solve inverse kinematics with an iterative scheme based on pseudo-inverse of robot Jacobian matrix to map the human hand movements to the robot. We project additional subtasks into the Jacobian Null-space in order to satisfy additional constraints like big variations in robot joints angles for little movements of the robot end-effector and to penalize joints values near to physical limits. Details about kinematic mapping are reported in the appendix VI-B.

IV. TOOLS

In this section we will describe the employed tools to conduct the experimental phase of this work.

A. Microsoft Kinect sensor

We used a Microsoft Kinect sensor together with the OpenNI2 library that implements the algorithm of [11]. The Microsoft Kinect sensor provides an RGB image and a Depth image respectively with a resolution 640x480 pixels and 320x240 pixels at a rate of 30Hz. The OpenNI2 library processes these data in order to provide human joints positions with respect to a camera fixed reference frame. Since we need the relative position of the user hands with respect to his torso, we estimate the homogeneous transformation matrix between the camera and the torso of the user by processing the data provided by the OpenNI2 library. We then compute the hands markers coordinates with respect to a torso fixed reference frame. Details about how the transformation matrix is computed are reported in appendix VI-A.

B. Baxter Robot

We used the Baxter Research Robot from Rethink Robotics for the experiment. The Baxter robot is a semi-humanoid robot with two 7-degrees-of-freedom arms. The arms have two electric 1-degree-of-freedom grippers on its end-effectors that allow pick-and-place operations. The robot is interfaced with the ROS operating system allowing rapid-development of complex robotic application. Different arms control modes are offered by the built-in controllers. We used the Joint Position control mode that allows to set reference values for each joint controller. In addition, the Joint Position control mode avoids auto-collision between the arms and the torso. An overview of the chosen control architecture is illustrated in figure 2.

V. EXPERIMENTAL RESULTS

Experiments were conducted to demonstrate the effectiveness of the proposed approach. First, we collected a dataset containing the commanded trajectories of the robot by the teleoperating user tracked with the Kinect sensor. The dataset also contains objects tracked positions and the labeled intentions of the user in terms of what object he is going to interact with. In particular, our scenario comprises three objects: a pot, a lid and a scoop, which are illustrated in figure 3. At this stage these objects have been selected for being easily grasped by the Baxter gripper.
We collected 45 robot end-effector trajectories grasping three

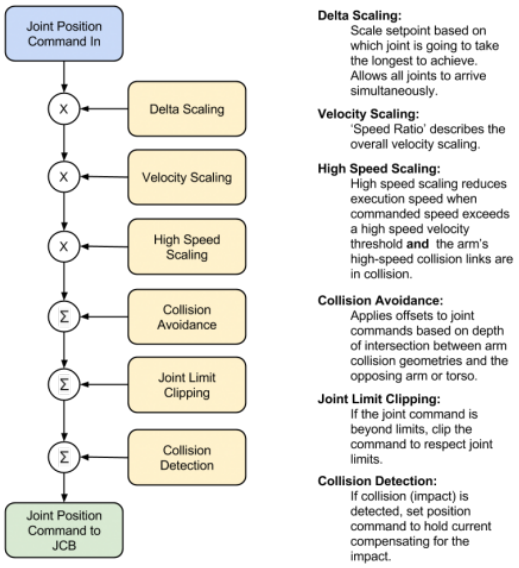

Fig. 2. Joint Position controller overview. (Courtesy Rethink Robotics `http://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes`)

different objects (15 for each object) moving each object in three different positions (5 trajectories for each object position). Then we divided the first part of such trajectories in segments of 1 second of duration extracting 93 labeled segments. For each segment we computed the features as expressed in equation 5 and we trained a motion pattern recognition neural network with the Matlab Neural Networks Toolbox. We set up the neural network to have a single hidden layer with 30 neurons. The result of the optimally trained network is shown as confusion matrix in figure 4.

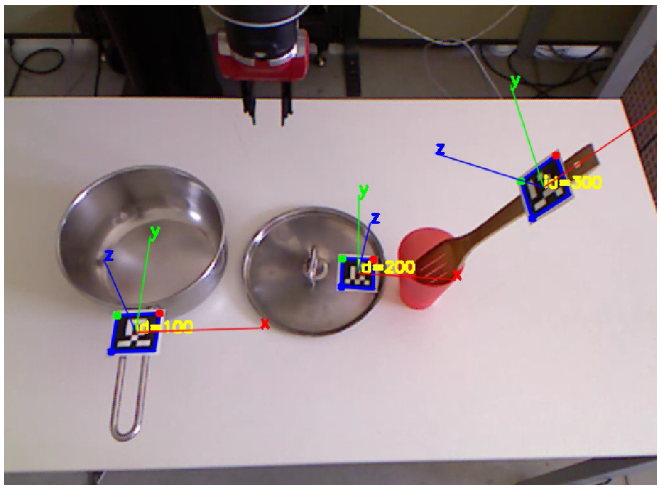

Fig. 3. Objects used for the experiments with the fiducial markers attached for their localization. The image is taken from RViz and shows the reference frames associated to each marker.

The trained network is able to classify the user intention with a very low failure rate. Once the classification network

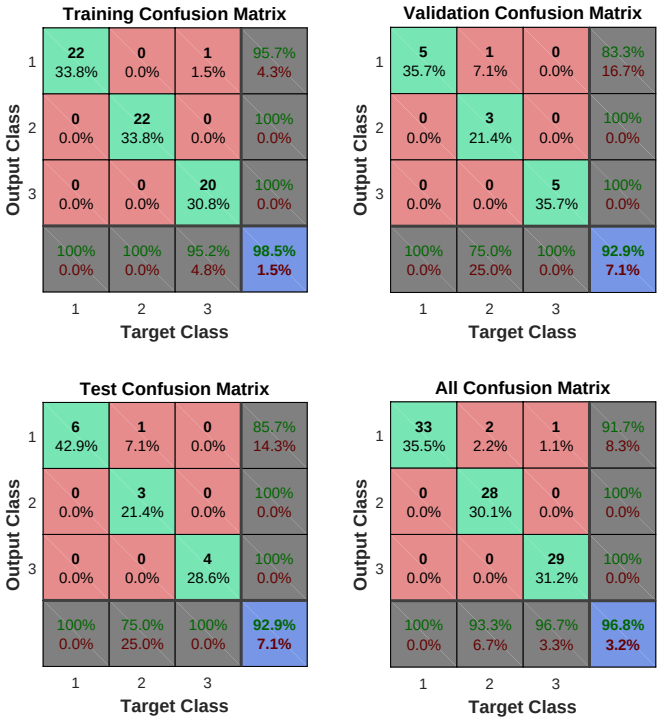

Fig. 4. Confusion matrix of the trained neural network for motion pattern recognition

was trained, we designed and implemented the supervising state machine for the examined case as exposed in section III-A. The properties of the objects gave the relations graph as in figure 5.

The relations graph is translated into the state machine illustrated in figure 6 as exposed in section III-B. With this architecture we tested the system by teleoperating it to pick the lid. The performances of the teleoperation improved in terms of positioning error and elapsed time as illustrated in figure 8 and 9. The elapsed time to complete the task reduced from a mean value of 6 seconds wihout additional constraints to 3 seconds with the virtual fixtures as illustrated by the boxplot in figure 7.

VI. CONCLUSIONS

In this work we developed a model for robot teleoperation that introduces constraints on the robot movements according to the output of an intention classifier and an affordances-based relationship graph. The introduction of virtual fixtures on the robot motion improved teleoperation performance in terms of positioning error and elapsed time to complete a predefined task, although in a preliminary single-subject test. Future work will include a more accurate modelling of user intentions in order to develop a classifier algorithm based on probabilistic modeling of the actions. While preserving the concept introduced in this work we'll need to automatically segment the examples and evaluate different classification techniques. In addition, we also will investigate the generalization of the virtual fixture model that can simplify the teleoperation of robots with various end-effector types having multiple degrees-of-freedom.

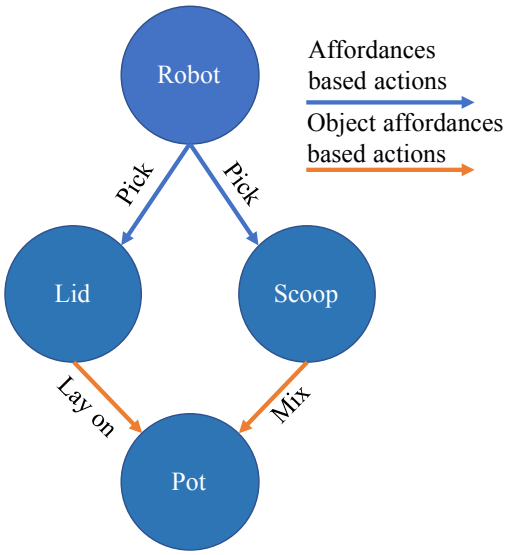

Fig. 5. The graph for the examined case

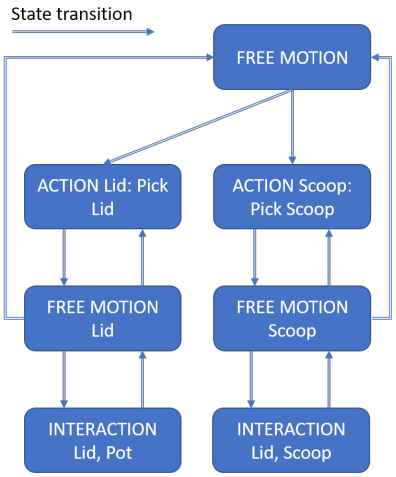

Fig. 6. The state machine derived from graph of figure 5

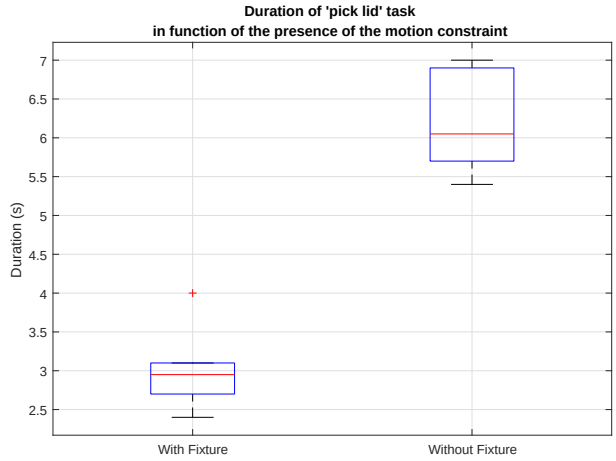

Fig. 7. Variations in task duration in presence and absence of the virtual fixture

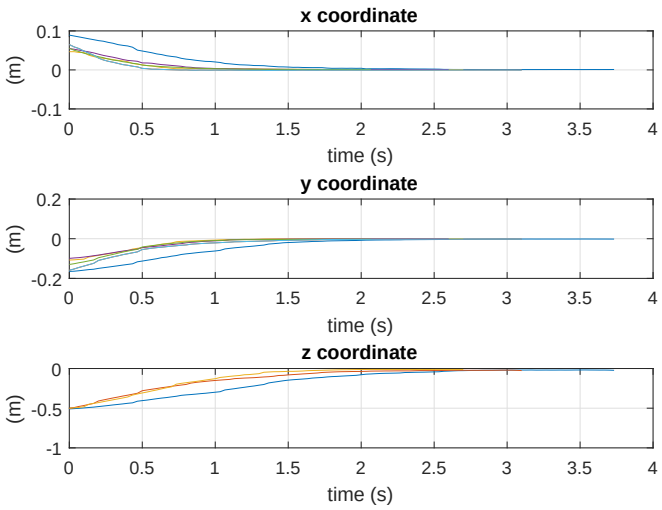

Fig. 8. Difference between robot end-effector trajectories and target object position with active constraint (left hand). Colors encode different trials.

APPENDIX

A. Estimation of Torso Frame

Among the markers provided by OpenNI library, the ones that we used to estimate the transformation matrix between a Torso (T) fixed frame and the Camera frame (C) are: left shoulder $\mathbf{x}^C_{LS}$, right shoulder $\mathbf{x}^C_{RS}$, neck $\mathbf{x}^C_{NE}$ and torso $\mathbf{x}^C_{TO}$. The superscript C indicates that the markers are written in Camera frame coordinates. The rotation matrix of the overall transformation matrix was obtained as follows:

$$\tilde{\mathbf{u}}_y = \frac{{}^C\mathbf{x}_{LS} - {}^C\mathbf{x}_{RS}}{\|{}^C\mathbf{x}_{LS} - {}^C\mathbf{x}_{RS}\|} \tag{6}$$

$$\mathbf{u}_z = \frac{{}^C\mathbf{x}_{NE} - {}^C\mathbf{x}_{TO}}{\|{}^C\mathbf{x}_{NE} - {}^C\mathbf{x}_{TO}\|} \tag{7}$$

$$\mathbf{u}_x = \tilde{\mathbf{u}}_y \times \mathbf{u}_z \tag{8}$$

$$\mathbf{u}_y = \mathbf{u}_z \times \mathbf{u}_x \tag{9}$$

$$R_{T,C} = \begin{pmatrix} \mathbf{u}_x & \mathbf{u}_y & \mathbf{u}_z \end{pmatrix} \tag{10}$$

The overall homogeneous transformation matrix T^T_C between Camera and Torso was obtained by adding the Torso position as translation vector:

$$T_{T,C} = \begin{pmatrix} R_{T,C} & {}^C\mathbf{c}_{T,C} \\ \mathbf{0} & 1 \end{pmatrix} \tag{11}$$

This allows to convert a point from coordinates with respect to Kinect Camera frame to coordinates with respect to Torso frame.

$${}^T\mathbf{x} = T_{T,C}\,{}^C\mathbf{x} \tag{12}$$

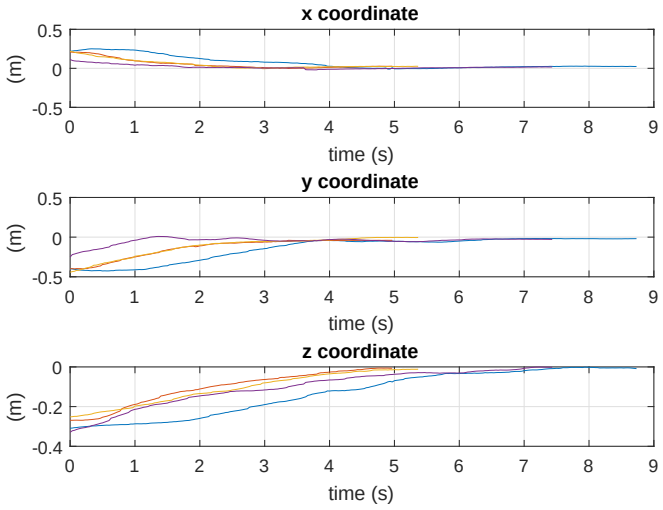

Fig. 9. Difference between robot end-effector trajectories and target object position without active constraint (left hand). Colors encode different trials.

B. Inverse kinematics mapping

To map human movement on the robotic manipulator we implemented an inverse kinematics iterative algorithm. The algorithm uses the pseudoinverse of the robot Jacobian matrix that maps the derivative of joints angles to the end-effector velocity, $\dot{x}_R = J(q)\,\dot{q}$, with q indicating the vector of the joints angles. We added a subtask projected into the null space of the Jacobian matrix in order to minimize the distance of the joints configuration from the middle point of joints limits. We defined the classical objective function:

$$H(q) = \frac{1}{2}\sum_{i=1}^{n_q}\left(\frac{q_i - q_{i,mid}}{q_{i,M} - q_{i,m}}\right)^2 \tag{13}$$

where $q_{i,M}$ and $q_{i,m}$ represent the maximum and minimum value for each joint respectively. $q_{i,mid}$ is the middle value of each joint angle $q_{i,mid} = (q_{i,M} + q_{i,m})/2$. The final inverse kinematics iterative algorithm is:

$$q_{k+1} = q_k + J^+(\mathbf{x}_{H,k} - \mathbf{x}_{R,k}) + \left(I - J^+J\right)\nabla H(q_k) \tag{14}$$

where J indicates the robot Jacobian matrix and the explicit dependence from joints angles vector q is omitted for brevity.

.

References

[1] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman *et al.*, "Team ihmc's lessons learned from the darpa robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[2] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Transactions on robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[3] A. Graziano, P. Tripicchio, C. A. Avizzano, and E. Ruffaldi, "A wireless haptic data suit for controlling humanoid robots," in *ISR 2016: 47st International Symposium on Robotics; Proceedings of.* VDE, 2016, pp. 1–8.

[4] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. Australasian Conference on Robotics and Automation*, 2012.

[5] S. A. Bowyer, B. L. Davies, and F. R. y Baena, "Active constraints/virtual fixtures: A survey," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 138–157, 2014.

[6] L. B. Rosenberg, "The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments." DTIC Document, Tech. Rep., 1992.

[7] J. Funda, R. H. Taylor, B. Eldridge, S. Gomory, and K. G. Gruben, "Constrained cartesian motion control for teleoperated surgical robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 3, pp. 453–465, 1996.

[8] A. Casavola and M. Sorbara, "Towards constrained teleoperation for safe long-distance robotic surgical operations," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 685–690.

[9] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[10] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[11] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[12] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 9–14.

[13] G. T. Papadopoulos, A. Axenopoulos, and P. Daras, "Real-time skeleton-tracking-based human action recognition using kinect data," in *International Conference on Multimedia Modeling*. Springer, 2014, pp. 473–483.

[14] S. Sempena, N. U. Maulidevi, and P. R. Aryan, "Human action recognition using dynamic time warping," in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*. IEEE, 2011, pp. 1–5.

[15] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.