Bi-Manual Articulated Robot Teleoperation using an External RGB-D Range Sensor

Emily-Jane Rolley-Parnell[1], Dimitrios Kanoulas[2], Arturo Laurenzi[2], Brian Delhaisse[2], Leonel Rozo[2], Darwin G. Caldwell[2], Nikos G. Tsagarakis[2]

Abstract— **In this paper, we present an implementation of a bi-manual teleoperation system, controlled by a human through three-dimensional (3D) skeleton extraction. The input data is given from a cheap RGB-D range sensor, such as the ASUS Xtion PRO. To achieve this, we have implemented a 3D version of the impressive *OpenPose* package, which was recently developed. The first stage of our method contains the execution of the OpenPose Convolutional Neural Network (CNN), using a sequence of RGB images as input. The extracted human skeleton pose localisation in two-dimensions (2D) is followed by the mapping of the extracted joint location estimations into their 3D pose in the camera frame. The output of this process is then used as input to drive the end-pose of the robotic hands relative to the human hand movements, through a whole-body inverse kinematics process in the Cartesian space. Finally, we implement the method as a ROS wrapper package and we test it on the centaur-like CENTAURO robot. Our demonstrated task is of a box and lever manipulation in real-time, as a result of a human task demonstration.**

I. Introduction

Throughout the past few decades, robots with limbs have started to emerge from labs and into the real world. Their role in completing tasks that may be harmful or difficult for humans, makes them important for the future of humanity. Unfortunately, their autonomy, which could be the key-aspect in completing these tasks, is still under development; especially in cases where the environment is unknown and uncertain. For this reason, there is great potential in the development of semi-autonomous ways to control and plan robot motions with the aid of humans; particularly by exploring the action application necessary for completing various tasks. One of the possible control methods is teleoperation, in which a human is controlling directly either the joints/torques of the robot or it's end-effectors. In this work, we are interested in tasks that include manipulation of objects by humanoid robots [1], controlling their end-effectors through whole-body motions [2].

The state-of-the-art in manual robot teleoperation is perpetually moving forward, closer to more reliable and robust methods of control. Whether for on-site control or telepresence, industrial or human facing, the number of reliable robotic systems, articulated systems in particular, is increasing. Thus it is necessary to focus on designing real-time control methods that are intuitive and comprehensive, especially in the case of humanoid robots.

More specifically, we face advances in popularity for applications of robots controlled manually to perform tasks such as construction, assisted assembly, or human-robot interaction. Yet in each of these situations, the on-site physical presence of a human may be hazardous. While teleoperation would be an invaluable tool for controlling robots that perform dangerous tasks, the current state of teleoperation is particularly costly, or may require a long and arduous calibration process; for instance, using a body suit of IMU sensors [3], an exoskeleton [4], or a specific array of motion-capture cameras [5] to generate a 3D model of the operating human.

In this paper we focus on providing a method that uses a cheap external RGB-D sensor (ASUS Xtion PRO) to control a complex articulated robot for realistic manipulation tasks. Our intention is to provide a three-dimensional (3D) visual system that can mimic human actions closely, by tracking human hands and controlling the robot's corresponding end-effectors. In particular, the implementation presented in this paper uses the OpenPose algorithm [6]—an impressive system that was recently introduced and is based on deep learning, which is able to detect in real-time ($8-10$Hz) human joint poses, using 2D images from an RGB camera. We use this system as input to transform the poses in 3D, using the depth information of our range sensor. The 3D human joint coordinates that are acquired from the system are given, after filtering, as references to control the end-effectors of the humanoid robot. These references are processed by a Cartesian Interface that we have implemented, which controls the whole-body of the robot in the Cartesian space, maintaining balancing through the real-time communication middleware XBotCore [7] and the Stack-of-Task (SoT) package OpenSoT [8] that is used on our robot to maintain the robot's balance, while driving bi-manual trajectory plans for manipulation. The implemented method was selected specifically for its cost effectiveness and accessibility, due to the cheap RGB-D cameras that are easily accessible nowadays. An overview of our system is visualised in Fig. 1.

The paper is organised as follows. First, we review the related work, followed by the implemented system overview, including the sensing, data processing, the hand pose tracking in 3D, and the robot trajectory planning control. Then, we present a set of demonstrations on our articulated robot

[1]University of Plymouth, Drake Circus, Plymouth, Devon PL4 8AA, United Kingdom. erolleyp@gmail.com

[2]Humanoid and Human-Centered Mechatronics Department, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163, Genova, Italy. {Dimitrios.Kanoulas, Arturo.Laurenzi, Brian.Delhaisse, Leonel.Rozo, Darwin.Caldwell, Nikos.Tsagarakis}@iit.it

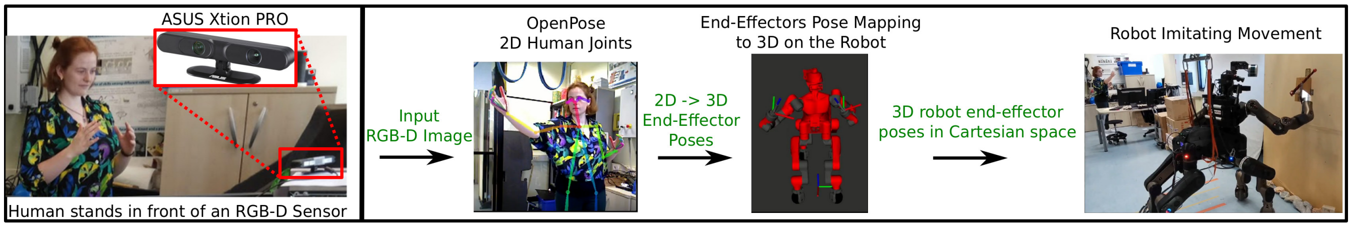

Fig. 1: System Overview.

CENTAURO in simulation and real-world, and finally we conclude with some future work.

A. Related Work

Teleoperation is a well studied field in robotics [9] and over the last decade it has started developing also for humanoid robots [10], [11]. As mentioned before, several systems use high accuracy tracking methods, such as motion capture cameras, markers or suits [12], [13], [14], or exoskeletons [15] to control humanoid robots from human demonstrations. There is also a variety of works that teleoperate humanoids through a GUI [16], [17], [18] or joystick-like tools [19], [20]. In this paper however, we are focused on cheaper methods of human mimicking-based teleoperation, using range cameras. This is a challenging problem due to various reasons, such as inaccurate or uncertain input data and occlusions of the human-body parts. Thus, few works have used visual-driven methods on real humanoid robots. In [21], a mini-humanoid robot is controlled using an RGB-D sensor, without providing any real-world testing on the robotic platform. In [22], a classic approach of using the skeletal tracking system NiTE to map human joints on mini-humanoids has been studied. In the direction of mini-humanoids, the same concept of teleoperation has been tried on a NAO robot [23], using the second version of Kinect. Most of these works were tried on mini-humanoids for tracking and imitating human movements. In this work, we intend to provide an integrated system that works on a full-size humanoid robot and can achieve bi-manual manipulation tasks, such as pushing and grasping. In that direction, the half-size humanoid robot iCub [24] used partially the Kinect sensor in combination with an Oculus Rift and a SensorGlove system to drive simple toy grasping movements on the robot.

Human-pose estimation, based on visual sensors is also an area that has been extensively studied, mainly in computer vision [25]. In this work, we focus on RGB-D data as input to our algorithm for human joint or pose estimation. As mentioned above, we use the impressive OpenPose deep learning based system [6] to extract 2D poses from the RGB sensor and then we map to the registered 3D space using the corresponding depth image, as we will describe in the next section. Recently, in [26] 2D pose estimates predict 3D joint positions with relatively low error, using a deep feed-forward network. In [27] the depth image is used to assign joint positions within the human hand with a residual network. However, this is done only related to human hands, and not the entire body. Detecting poses of multiple people present is achieved in [28] by using bottom-up proposals for body parts. Even though the system performs well, it does not outperform the 2017 OpenPose network. Before the release of OpenPose, promising results were reached by human action recognition networks [29] for skeleton localization and convolution pose machines [30] for pose estimation based on image features learning and image-dependent spatial models. Last but not least, in [31] articulated human detection and deformable part model templates are used to recognize human limbs. OpenPose outperforms most of these methods and will be used in this work for 2D human joints localization, using RGB data.

II. System Overview

In this section, we describe in detail the stages of the total integrated system as it is visualised in Fig. 1, that is used to turn RGB-D human pose data to end-effector poses that control a humanoid robot for bi-manual tasks in the whole-body Cartesian space.

A. RGB-D Sensing

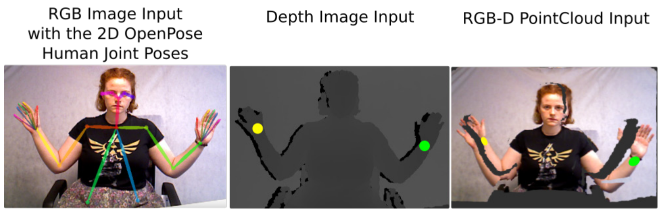

Fig. 2: An RGB input image with the localised 2D human joint poses (OpenPose), and the corresponding depth image and the coloured organised point-cloud.

The source of input data plays an important role for our algorithm. The most common and reliable human body-parts detection or localisation methods, such as OpenPose [6], rely on RGB data. In addition, the corresponding registered depth or pointcloud data are required for the 2D-to-3D pose mapping from the human to the robot. Between different types of sensors [32], we decided to use an RGB-D structured light camera sensor (ASUS Xtion PRO), which is cheap, but reliable enough and works at a frame rate of 30Hz. It's depth resolution is 640×480, while the RGB camera can reach the value of 1280×1024 in a lower frame rate. In our experimental setup, we keep both depth and RGB resolution to 640×480 (acquired as grid-organised, with the non-valid depth pixel values be set to NaNs) working in 30Hz, so that

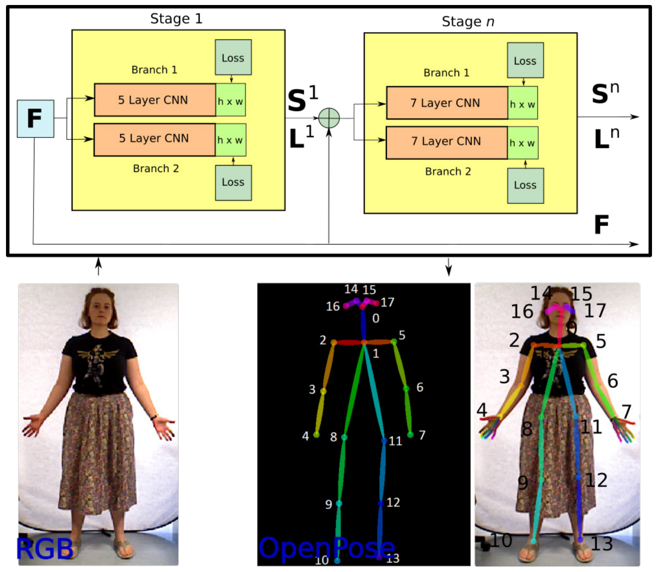

Fig. 3: The OpenPose architecture, with two-branch multi-stage Convolutional Neural Network. The first branch predicts confidence maps $\mathbf{S}^n$, and the second predicts Part Affinity Fields $\mathbf{L}^n$. After each stage, the predictions from the two branches and the image features, are concatenated for the next stage. The input data is RGB images, and the outputs are keypoint markers that display the human joint poses (numbered from 0 to 17; we use only the upper body parts).

we also have a one-to-one correspondence between depth and colour data, while keeping the performance of the OpenPose neural network computationally efficient. Keeping the data organised is important to access simultaneously the colour and depth values for each pixel of the input RGB-D data. A visualisation of the range sensing system (RGB, depth, and coloured point cloud data) is found in Fig. 2. One can notice that due to calibration errors between the colour and depth sensors on the camera, there is a mismatch between some colour and depth pixels around depth discontinuities in the real world. This may cause errors when mapping 2D to 3D positions and will be handled by using real-time points nearest neighbourhoods of organised pointclouds, for the human joint localisation and mapping, instead of a single pose pixel.

B. Extracting 2D Human Positions with OpenPose

OpenPose [6] (Fig. 3) is a feed-forward deep neural network trained on the CMU Panoptic Dataset [33] with 65 human sequences (5.5 hours) and 1.5 million two-dimensional corresponding skeletons. The method is able to efficiently detect 2D positions of multiple people within an RGB image in real-time ($8-10$Hz). This is done by bodypart detection and association, using Part Affinity Fields and Confidence Maps. Each of these make OpenPose an innovative network with state-of-the-art success rate. The bodypart detection takes place in a sequential style, performing bottom-up prediction by using learned spatial context. The extracted information is subsequently used to create the starting structure for the human skeleton in the current frame. Each body part is enumerated accordingly, as it is visualised in Fig. 3. We use this method for the robust tracking of both body and hands parts of humans in 2D. The method is reliable in preventing occlusions and misidentification by proximity, making it an invaluable tool for accurate pose estimation.

C. Mapping 3D Human Positions

The original OpenPose network outputs 2D positions of human joints in the RGB image. To be able to use these to control the robot end-effectors, it is required to map these positions in 3D. For this purpose we wrote a Robotic Operating System (ROS) wrapper package, that transforms the 2D output keypoint coordinates to 3D positions. Since the associated point cloud is also organised, one can directly extract the corresponding point (X,Y,Z) in the 3D Cartesian space for each 2D pixel (u,v) in the colour image. Fig. 2 visualises this process.

There may be two important issues with this direct mapping. First, there is a case that a pixel may have unregistered (NaN) depth value in the corresponding associate point cloud. To handle this, we use the nearest neighbourhood pixels and average their values, assuming that the central pixel of the human 2D joint position, is mapped in the correct registered depth pixel from the sensor itself. In this way, we avoid having a lot of invalid RGB-D pixels and we make our mapping more reliable to inaccurate depth errors. Secondly, when there are occlusions OpenPose sets the location of the human part in occlusion to NaN. In this case, we also do not map this body part to any 3D corresponding position. We then handle it's absence when controlling the robot, as we will see below.

D. Robotic Hand 6-DoF Pose Control

In order to make use of the 3D position information provided by the preceding system, for controlling the robotic hands (end-effectors), the 6 Degree-of-Freedom (DoF) need to be generated for each extracted human hand position. In particular, each extracted X,Y,Z position in the 3D camera frame needs to be represented as a pose in the robot world frame. We handle separately the position (Cartesian coordinates represented by X,Y,Z) and orientation (rotation represented by a quaternion q_w,q_x,q_y,q_z), as explained in the following subsections.

1) Robotic Hand Position: To control the hand end-effectors of the bi-manual robot, we must provide a Cartesian position of each wrist relative to the human world frame. Throughout the process of implementation, three different methods were attempted to provide these Cartesian positions. The first method gave wrist positions relative to the previous image frame. While this method was easy to implement, it was prone to drifting and losing the true location of the wrists relative to the human operator. The second method provided the coordinates relative to the hip joints of the human and mapped this appropriately to the humanoid robot. However,

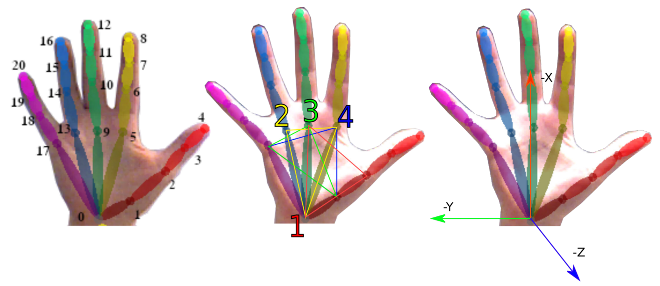

Fig. 4: Left: the labels of the hand keypoints $1-20$, as assigned by OpenPose. Middle: the points 0, 1, 2, 5, 9, 13, and 17 are used to create a series of planar triangles to calculate the $\hat{Z}$-axis of the hand-frame. Right: the calculated hand-frame.

this required a calibration sequence in order to address the difference in size between the human and the robot, and to scale the displacement in limb from human to robot. The disadvantage of this method is that as the position of the hands was given relative to the position of the hips, the robot functionality of changing height was removed, and thus a robot could not change elevation to reach down and pick up an object. Last, we have implemented a simple but robust method, which we finally used for our system. In the third method the reference point is the initial point of the hands when the tracking system runs. In this way, the movement of the robot, as measured in the real world, is the same as the difference between the frames that moved in the robot frame. This method is simple yet effective in maintaining reasonable accuracy and preventing drift of coordinates.

To remove unneeded system complexity and confusion, we consider and filter the OpenPose output data in such a way that only one person is tracked at a time. Another layer of filtering is added with the intent of removing any frame that is unsuccessful in identifying the location of the wrist joints, or is outlying, so that the physical robot's wrist jumps (i.e., fast movements from one point to an other) between reference points cannot be so big as to be hazardous for the robot, thus smoothing the output.

2) Robotic Hand Orientation: The OpenPose method also returns the hand joints as long as they are visible. The orientation of the robotic hand can be found by creating three hand-frame axes, determined by keypoints recognised on the hand of the human, as seen in Fig. 4. Our initial task is to calculate a 3×3 rotation matrix:

$$M = \begin{bmatrix} \hat{X}_x & \hat{Y}_x & \hat{Z}_x \\ \hat{X}_y & \hat{Y}_y & \hat{Z}_y \\ \hat{X}_z & \hat{Y}_z & \hat{Z}_z \end{bmatrix} \tag{1}$$

where $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ are each unit vectors of the hand-frame to be calculated. Their subscripts are the the individual x, y and z components of each unit vector. The following demonstrates how we calculate each one.

We let $\hat{Z}$ be the surface normal of the planes formed by 4 different triangles between palm keypoints. The four normal vectors that are formed from each triangle are given by:

$$\vec{K}_\ell = \overrightarrow{P_iP_j} \times \overrightarrow{P_iP_k} \tag{2}$$

where $\vec{K}_\ell$, $\ell \in \{1,2,3,4\}$ denotes the four normals extracted from each triangle, and P_i, P_j, and P_k are the 3D hand keypoints that form the triangles. We use the triangles that correspond to the (P_i,P_j,P_k) point tuples: $(1,17,5)$, $(1,17,9)$, $(0,13,5)$, and $(0,9,2)$, as visualised in Fig. 4. The cross product is denoted by $\times$. To calculate the final $\hat{Z}$ normal vector, we average over the four $\vec{K}_\ell$ vectors:

$$\hat{Z} = \frac{1}{4}\sum_{\ell=1}^{4} \vec{K}_\ell. \tag{3}$$

Notice, that the $\hat{Z}$ vector needs to be normalised to give a unit vector for the hand frame. Averaging over four triangles helps to avoid issues when hand keypoints may be misidentified.

The $\hat{X}$ vector is calculated as the the connecting vector from the centre of the palm towards the fingers. This is calculated by connecting the 0-labeled 3D keypoint (P_o) on the hand and the the mean M_p, of the keypoints labelled as $P_i \in \{17,13,9,5\}$ that are in the beginning of each finger:

$$M_p = \frac{1}{4}\sum_{i=1}^{4} P_i \qquad \hat{X} = \overrightarrow{P_oM_p}. \tag{4}$$

Finally, the $\hat{Y}$ vector is given by the cross product of the normalised $\hat{Z}$ and $\hat{X}$.

$$\hat{Y} = \hat{Z} \times \hat{X} \qquad \hat{X} = \hat{Y} \times \hat{Z}. \tag{5}$$

The recalculation of $\hat{X}$ is due to the fact that Eq. 4 does not give the axis perpendicular to the $\hat{Z}$ one. We let the negative vectors $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ to form the final rotation matrix M. The rotation matrix can be easily converted into a quaternion q_w, q_x, q_y, q_z, as used by our Cartesian Interface module. We used the *pyquaternion* library in Python to convert from rotation matrices to quaternions.

3) Cartesian Interface on the CENTAURO Robot: The robot that we use in this work is the CENTAURO [34]. The 42-DoF robot is loosely designed on the structure of a centaur and has 4 wheeled legs and two manipulator arms with 7-DoF each. It's purpose is to have a design that is capable of conquering rough terrain that is challenging to bipedal robots, while leaving two bi-manual manipulators available for obstacle removal and tools.

CENTAURO is controlled using the hard real-time middleware XBotCore [7]. To control the robot's end effectors in the Cartesian space, given hand goal reference poses, a Cartesian Interface module[1] has been developed on the top of the XBotCore. These use the OpenSoT [8] kinematic solver to update the joints accordingly. OpenSoT is a library dedicated to hierarchical whole-body robot control. This is done subject to constraints, such as joint limits, joint velocities, balancing, or Cartesian constraints. In this way we can control the robot end-effectors according to the calculated and mapped 3D human hand poses in real-time. Along with

[1] github.com/ADVRHumanoids/CartesianInterface

linear interpolation applied to the rotations independently, the Cartesian Interface applies smoothing and path interpolation for both hand end-effectors.

III. Robot Demonstrations

In this section, we present all the experimental results of the total system on the CENTAURO robot, including three different real-world manipulation tasks through teleoperation and one simulated movement task. Videos of the experimental results can be found under our webpage: `https://sites.google.com/view/telepose`

A. *Runtime and Disparity Measurements*

We first measure the runtime speed of our system and then we present the disparity movement of the robot hand compared to the human hand.

1) Runtime Speed: We measured the runtime speed of our network at many stages. Our available computer is equiped with two NVIDIA Titan Xp GPUs and the frequency of messages is shown in Table I.

Frequency of Message Published						
Stage	**OpenNI2**		**OP**	**OPToROS**		**Pt2Xbot**
ROS Msg	I_R	PCL2	keypts	r_hand	r_arm	r_robot
Freq (Hz)	29.82	29.98	8.07	6.65	6.75	7.13

TABLE I: A summary of the frequency at which the data messages are published from ROS. The Cartesian Interface node receives messages at a rate of approximately 7.13Hz.

The frequency output of the RGB-D sensor's colour image and point cloud is approximately 30Hz. This drops to 8.07Hz, when tracking both human body and hands, matching the OpenPose benchmark. OPToROS is an executable that republishes only the wrist and hand keypoint messages needed for our system. The speed of publishing drops here, but the final speed is 7.13Hz, which is still high enough speed to provide smooth robot movements.

2) Disparity in Movement: While the mapping of human to robot is intended to be at 1:1 ratio, there may be slight errors in tracking. To test this, we sampled a number of movements in a horizontal direction by 0.5m. From a small sample we found that if the movement is repeated, the variance in X, Y and Z coordinates is small, i.e., 0.001m, 0.0001m, and 0.0002m, respectively (0.001m Euclidean distance variance). The average Euclidian distance error was calculated to be approximately 0.06m.

B. *Task Control*

In order to demonstrate the capabilities of our system, three manipulation tasks on the real robot and one on the simulated robot were designed for using the robotic hands. The goal is to mimic a human teleoperator through an external ASUS Xtion PRO sensor that is set in front of her. Below, we give the summary of each task, for which the wheels/feet of CENTAURO are fixed in place. Notice that in the first three tasks the hand orientation is fixed to show the positioning capabilities, while for the fourth task we allow also hand orientation movements.

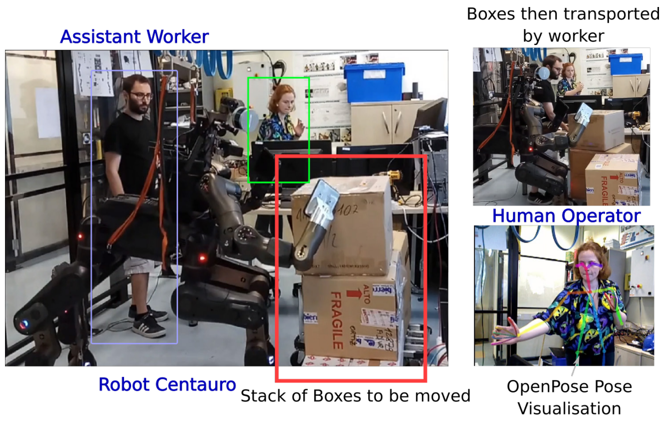
Assistant Worker
Boxes then transported by worker
Human Operator
Robot Centauro
Stack of Boxes to be moved
OpenPose Pose Visualisation

Fig. 5: Task 1: demonstrating a box push from one stack to another, for the purpose of being transported by a worker.

Task 1 - Box Push In the experiment illustrated in Fig. 5, a pile of boxes is arranged in front of the robot CENTAURO. A human worker waits nearby to transport a trolley of boxes from one location to another. To move the top box efficiently from one stack to the worker's stack, a simple push from the right hand is required. While only one hand is manipulating, the other hand, being controlled by the demonstrator moves in a way as to maintain balance of the robot and allow a bigger range of moves for the other hand, thus implementing bi-manual control.

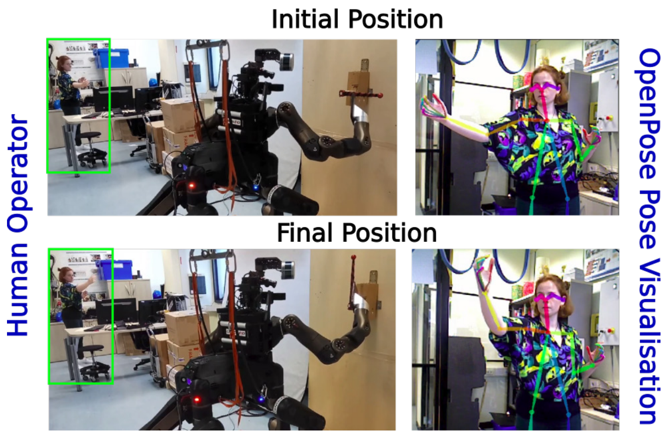

Fig. 6: Task 2: the initial and final positions of a lever being moved. The lever is rotated 90 degrees counter-clockwise.

Task 2 - Lever Move: In Fig. 6, a rotational bar lever is situated in front of the robot. Instead of this task being performed by the human, this is easily carried out by our robot. The robot, following the movement of the human operator, lifts the bar in an counter-clockwise direction approximately 90 degrees.

Task 3 - Box Lift: This task requires that a worker may access a box that is in the middle of a stack of other boxes. To offer assistance, the robot CENTAURO carefully lifts the top box to reveal the middle box. In this lifting action, individual control of each hand allows an end-effector to be positioned on either side of the top box. As the manipulators move

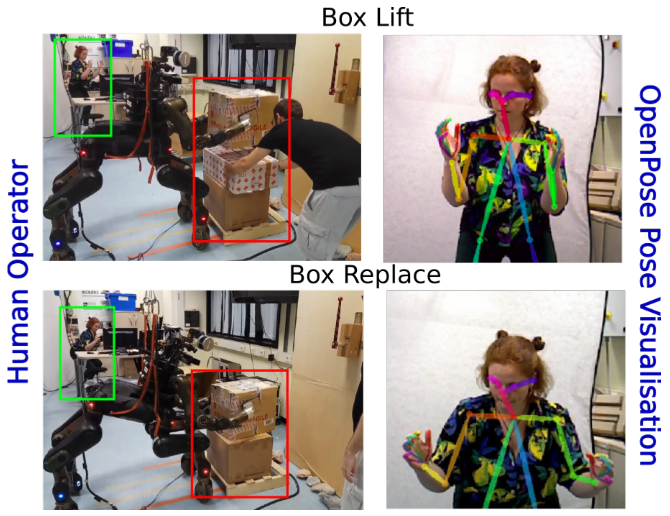

Fig. 7: Task 3: two stages of a box lift task. In the lifting stage, the box is held in the air so that the worker can reach for the box in between. Last, the robot is placing the box back on the pile, on top of the lower box.

together, pressure is applied to the box. Whole-body control is exemplified in this movement as a change in height of the legs and body is used to raise the box. The whole-body control is a feature of the Cartesian Interface module and OpenSoT that allows a much greater range of height and width when referring to an area of manipulability.

Task 4 - Orientation Example: In order to demonstrate the ability of hand orientation, Task 4 is a performance of range on only one wrist. It can be seen in the image that the palm of the operator is aligned with the direction of the end effector on CENTAURO. The axes of the hand effector are orientated in this fashion so that when using the flat paddle manipulator hand, like in Tasks 1-3, the orientation of the palm may match the orientation of the paddle.

1) Analysis: In the experimental tasks described above, small jumps may be observed. These are caused by a possible number of reasons. Occasionally, the OpenPose network may misidentify the true location of a joint; if the joint is correctly identified, there is no guarantee that the depth cloud has an available depth for that pixel location; or an occlusion of the joint returns an inaccurate depth as it reads the closest depth rather than the true position. In order to prevent these jumps from being problematic to operation, we apply a filtering that limits the Euclidean distance of robot hand movement to be a maximum of a threshold (10cm has been used throughout the experiments): if the distance between positions in the frame at time t and at $t+1$ is greater than the threshold, the movement is ignored.

The method we use to track position becomes challenging to implement when measuring the rotation of the hands. Many orientations create occlusions, as such the orientation cannot be predicted with as much accuracy. We attempted to solve this problem with projected path of movement, and this reduces some of the jumps, along with the simulation movement being smoothed by the Cartesian Interface.

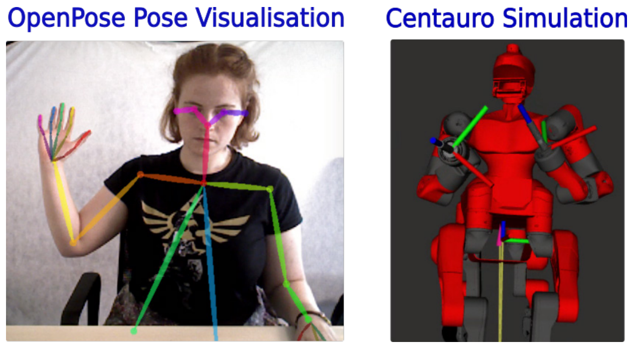

Fig. 8: A visualisation of the comparison between the angle of the human hand and the robot end effector on the simulated CENTAURO robot.

2) Reach Differences: Due to the human proportions being different from those of CENTAURO, the maximum reach of the arms is different. This difference signifies that even if the arms of the human are fully extended, this is not necessarily true on the robot. At the beginning of testing, the experiments were run on the robot COMAN. The differences in reach were more obvious as its size was closer to an average adult human. However to circumvent this scale issue, we implemented the method found in [35]. In order to map from the given keypoints of the wrists to the robot, a sequence of calibration frames must be taken. These frames were then used to calculate the required scales in each of the directional axes. First position measurements are taken from the OpenPose network output with arms down in a neutral position; then with arms straight towards the camera; and finally in a ”T” position with the arms straight out to either side. This method was useful for our initial attempts at controlling COMAN, as scaling of movements was required, but was unnecessary and time consuming for the robot CENTAURO inside the system that was described in this paper.

IV. Conclusions and Future Work

In this paper, we have demonstrated that our system implementation and integration of an RGB-D sensor, a pose recognition network, and an Inverse Kinematic system, is effective in controlling a bi-manual robot, through human visual teleoperation. We demonstrated the method on a real robot (CENTAURO) to complete various bi-manual tasks.

We envision this system to be used for simple and quick control of humanoid robots. For instance in recording demonstration training data of arm movements that can be later used for reinforcement learning. This data may be collected in an easy manner rather than having to manually manipulate potentially heavy and unwieldy robot arms. Moreover, we plan to use the developed system with other types of sensors, such as stereo cameras, that can work also under sun light. Another opportunity for improvement would be to develop further on the hands; incorporating not only orientation, but also tracking whether the hand is open or closed. A particularly useful tool to be used with grippers that are frequently affixed to bi-manual robots. These are both interesting avenues for future work.

ACKNOWLEDGMENT

This work is supported by the CENTAURO (grant agreements no 644839) and CogIMon (grant agreements no 644727) EU projects. The two Titan Xp GPUs used for this research were donated by the NVIDIA Corporation. The authors would also like to thank Enrico Mingo Hoffman, Luca Muratore, and Giuseppe Rigano for their help with the OpenSoT and XBotCore packages on the robot.

REFERENCES

[1] D. Kanoulas, J. Lee, D. G. Caldwell, and N. G. Tsagarakis, "Center-of-Mass-Based Grasp Pose Adaptation Using 3D Range and Force/Torque Sensing," *International Journal of Humanoid Robotics (IJHR)*, p. 1850013, 2018.

[2] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and Beyond for Assistive Humanoid Robots," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 175–226, 2013.

[3] N. Miller, O. C. Jenkins, M. Kallmann, and M. J. Mataric, "Motion Capture from Inertial Sensing for Untethered Humanoid Teleoperation," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, vol. 2, 2004, pp. 547–565.

[4] I. Sarakoglou, A. Brygo, D. Mazzanti, N. G. Hernandez, D. G. Caldwell, and N. G. Tsagarakis, "HEXOTRAC: A Highly Under-Actuated Hand Exoskeleton for Finger Tracking and Force Feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1033–1040.

[5] J. Silvério, S. Calinon, L. D. Rozo, and D. G. Caldwell, "Learning competing constraints and task priorities from demonstrations of bimanual skills," *CoRR*, vol. abs/1707.06791, 2017.

[6] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] L. Muratore, A. Laurenzi, E. Mingo Hoffman, A. Rocchi, D. G Caldwell, and N. Tsagarakis, "XBotCore: A Real-Time Cross-Robot Software Platform," in *IEEE International Conference on Robotic Computing (IRC)*, 2017, pp. 77–80.

[8] E. Mingo Hoffman, A. Rocchi, A. Laurenzi, and N. G. Tsagarakis, "Robot Control for Dummies: Insights and Examples using OpenSoT," in *17th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2017, pp. 736–741.

[9] S. Lichiardopol, *A Survey on Teleoperation*, ser. DCT rapporten. Technische Universiteit Eindhoven, 2007, dCT 2007.155.

[10] M. Stilman, K. Nishiwaki, and S. Kagami, "Humanoid Teleoperation for Whole Body Manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 3175–3180.

[11] C. Fok, F. Sun, M. Mangum, A. K. Mok, B. He, and L. Sentis, "Web Based Teleoperation of a Humanoid Robot," *CoRR*, vol. abs/1607.05402, 2016.

[12] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a Humanoid Robot using Full-Body Motion Capture, Example Movements, and Machine Learning," in *Australasian Conference on Robotics and Automation (ACRA)*, 2012.

[13] R. O'Flaherty, P. Vieira, M. X. Grey, P. Oh, A. Bobick, M. Egerstedt, and M. Stilman, "Humanoid Robot Teleoperation for Tasks with Power Tools," in *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2013, pp. 1–6.

[14] F. Negrello *et al.*, "The WALK-MAN Robot in a Postearthquake Scenario," *RAM*, vol. PP, no. 99, 2018.

[15] "Toyota Unveils Third Generation Humanoid Robot T-HR3," 2017. [Online]. Available: goo.gl/aaztz1

[16] P. Kaiser *et al.*, "An Affordance-Based Pilot Interface for High-Level Control of Humanoid Robots in Supervised Autonomy," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 621–628.

[17] C. Phillips-Grafflin, N. Alunni, H. B. Suay, J. Mainprice, D. Lofaro, D. Berenson, S. Chernova, R. W. Lindeman, and P. Oh, "Toward a User-Guided Manipulation Framework for High-DOF Robots with Limited Communication," *Intelligent Service Robotics (ISR)*, vol. 7, no. 3, pp. 121–131, 2014.

[18] P. Balatti, D. Kanoulas, G. F. Rigano, L. Muratore, N. G. Tsagarakis, and A. Ajoudani, "A Self-tuning Impedance Controller for Autonomous Robotic Manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[19] T. Rodehutskors, M. Schwarz, and S. Behnke, "Intuitive Bimanual Telemanipulation under Communication Restrictions by Immersive 3D Visualization and Motion Tracking," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 276–283.

[20] M. Ogawa, K. Honda, Y. Sato, S. Kudoh, T. Oishi, and K. Ikeuchi, "Motion Generation of the Humanoid Robot for Teleoperation by Task Model," in *24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2015, pp. 71–76.

[21] W. Song, X. Guo, F. Jiang, S. Yang, G. Jiang, and Y. Shi, "Teleoperation Humanoid Robot Control System Based on Kinect Sensor," in *4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, 2012, pp. 264–267.

[22] S. Michieletto, E. Tosello, E. Pagello, and E. Menegatti, "Teaching Humanoid Robotics by Means of Human Teleoperation through RGB-D Sensors," *Robotics and Autonomous Systems (RAS)*, vol. 75, pp. 671–678, 2016.

[23] C. Li, C. Yang, P. Liang, A. Cangelosi, and J. Wan, "Development of Kinect Based Teleoperation of NAO Robot," in *International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2016, pp. 133–138.

[24] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra, "First-Person Tele-Operation of a Humanoid Robot," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 997–1002.

[25] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, "RGB-D-based Human Motion Recognition with Deep Learning: A Survey," *CoRR*, vol. abs/1711.08362, 2017.

[26] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A Simple Yet Effective Baseline for 3D Human Pose Estimation," *CoRR*, vol. abs/1705.03098, 2017.

[27] M. Oberweger and V. Lepetit, "DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation," *CoRR*, vol. abs/1708.08325, 2017.

[28] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," *CoRR*, vol. abs/1605.03170, 2016.

[29] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, "A Human Activity Recognition System Using Skeleton Data from RGBD Sensors," *Intell. Neuroscience*, pp. 21–35, 2016.

[30] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional Pose Machines," *CoRR*, vol. abs/1602.00134, 2016.

[31] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 35, no. 12, pp. 2878–2890, 2013.

[32] G. Rauscher, D. Dube, and A. Zell, "A Comparison of 3D Sensors for Wheeled Mobile Robots," *Intelligent Autonomous Systems (IAS)*, vol. 302, pp. 29–41, 09 2016.

[33] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic Studio: A Massively Multiview System for Social Interaction Capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[34] L. Baccelliere *et al.*, "Development of a Human Size and Strength Compliant Bi-Manual Platform for Realistic Heavy Manipulation Tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5594–5601.

[35] B. Delhaisse, D. Esteban, L. Rozo, and D. Caldwell, "Transfer Learning of Shared Latent Spaces Between Robots with Similar Kinematic Structure," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4142–4149.