Hands-Free: a robot augmented reality teleoperation system

Cristina Nuzzi*, Stefano Ghidini, Roberto Pagani, Simone Pasinetti and Giovanna Sansoni

***Abstract*— This electronic document is a live template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document.This electronic document is a live template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document.This electronic document is a live template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document.**

I. INTRODUCTION

Despite advances in robotic perception are increasing autonomous capabilities, the human intelligence is still considered a necessity in unstructured or not predictable environments. Typical scenarios concern the detection of random shape objects, manipulation, or custom robot motion. In such context, human and robots must achieve mutual human-robot interaction (HRI) [1].

HRI can be physical (pHRI) or not, depending on the assigned task. For example, when the robot is constrained in a dangerous environment or must handle hazardous materials, pHRI is not recommended. In these cases, robot teleoperation may be necessary. A teloperation system concerns with the exploration and exploitation of spaces that do not allow user presence, thus, the operator acts by remotely move the robot [2]. A plenty of human-machine interfaces for teleoperation have been developed considering a mechanical interface, including exoskeleton [3] or gloves [4]. Such systems are particularly helpful to achieve bilateral teleoperation [5], where they can transmit or reflect back to the user reaction forces from the task being performed. In this case, a high perception with complete haptic feedback [6] is achieved. Other controllers include types of mouse, switchbox, keyboard, touch-screen and joystick, which is usually a better control device than others because the operators can identify better with the task [7]. Among the systems where bilateral teleoperation is not required, a teleoperation system is defined by mean of electromyography (EMG) signals of the muscular activity [8], [9]. However, as reminded recently in [10], EMG could be affected by difficulties in processing EMG signals for amplitude and spectral analysis, reducing their efficiency for many applications. Moreover, all these interfaces still act by contact, hindering the movement of the operator or cause him to act through unnatural movements.

On the other side, if bilater interaction is not required, a vision-based interface is preferable. A vision-based interface does not require physical contact with external devices such as cables, connectors and objects outside of the user working area. This grants a more natural and intuitive interaction, which is reflected on the task performance: as shown in [11], the accuracy of object gripping tasks is improved by mean of a contactless vision-based robot teleoperation method, while in [12] a stereo vision system improved the performance of a mobile robot teleoperation application.

Furthermore, if a vision-interface is integrated with virtual and augmented reality techniques, it translates in a greater level of immersion for the user. Such techniques are used to enhance the feedback information; in fact, the operator feels like being physically present in the remote environment, enhancing the immersion level. The notion of immersion is one of the most important reasons for using virtual and augmented reality [7]. An augmented reality system for teleoperation based on the Leap Motion (LM) controller is presented in [13]. For its application domain, the LM controller is considered an accurate sensor [14], however it is limited to a relatively small measuring distance if compared with other sensors. In this sense, the LM controller introduces spatial constraints that clash with the concept of the high level user immersion.

For these reasons, we present a novel robot augmented reality teleoperation system that exploits RGB cameras, which provide greater measuring distance if compared with the LM controller. A ROS-based framework has been developed to provide hand tracking and hand-gesture recognition features, exploiting the OpenPose software [15], [16] based on the Deep Learning framework Caffe [17]. This, in combination with the ease of availability of an RGB camera, lead the framework to be strongly open-source oriented and highly replicable on all the ROS-based platform. The proposed system includes: neural network for hand−gesture recognition (Section III), a rigorous procedure for robot workspace calibration and a mapping policy between the coordinate system of the user and the robot (Section III). Different experiments were performed on a *Sawyer (Rethink Robotics)* industrial collaborative robot to evaluate repeatability and accuracy of the proposed system. Section IV reports the results.

II. WORKSPACE CALIBRATION AND MAPPING

Our set-up is composed of two workspaces: the *user workspace* and the *robot workspace*. Cartesian points in the

This work was not supported by any organization.

Cristina Nuzzi, Roberto Pagani, Simone Pasinetti and Giovanna Sansoni are members of the Department of Mechanical and Industrial Engineering at University of Brescia, Via Branze 38, 25123 Brescia, Italy.

Stefano Ghidini is a member of the STIIMA-CNR, Via Alfonso Corti 12, 20133 Milan, Italy. He is also a member of the Department of Mechanical and Industrial Engineering at University of Brescia, Via Branze 38, 25123 Brescia, Italy.

* Corresponding author, e-mail: c.nuzzi@unibs.it

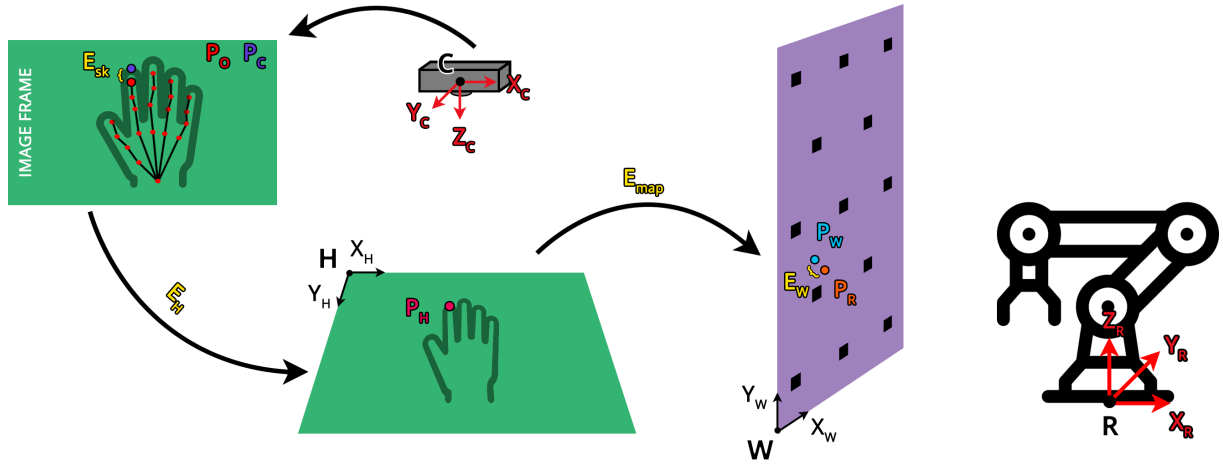

Fig. 1. Scheme of the calibration steps.

user workspace that can be reached by the user hand and are correctly viewed by the camera correspond to precise robot end-effector Cartesian points in the robot workspace. To obtain the mapping between the hand positions and the robot end-effector positions, it is necessary to perform a set of calibration procedures described in detail in the following sections.

A. User Workspace Calibration

In the user workspace an RGB camera is used to recognize the hand skeleton in real-time. Therefore, it is necessary to properly calibrate the camera relative to the user-defined reference system. This procedure is called *camera calibration*, and can be easily realized following standard procedures, such as the one detailed in [18]. The projection mapping for a generic point $\mathbf{P}_C = (u, v)$ in the camera image plane with reference frame C to its corresponding real world coordinate point $\mathbf{P}_H = (x, y, z)$ in reference frame H is defined by the following Equation. Homogeneous coordinates are required:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

However, since we are looking for the point position $\mathbf{P}_H$ in the frame H by back-projecting a 2D point to 3D it is necessary to invert Equation 1:

$$\mathbf{P}_H = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \left(s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \mathbf{K}^{-1} - \mathbf{t} \right) \mathbf{R}^{-1} \tag{2}$$

In the equations above, the scalar $s \in \mathbb{R}$ is the scale factor of the image, $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera matrix containing the intrinsic parameters of the camera such as focal length and optical center obtained through the calibration procedure, $[\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3\times4}$ is the rigid transformation matrix containing the exstrinsic parameters for rotation ($\mathbf{R} \in \mathbb{R}^{3\times3}$) and translation ($\mathbf{t} \in \mathbb{R}^3$) of the camera reference frame C relative to the calibration master reference frame H. To obtain matrix $\mathbf{K}$ it is necessary to perform a calibration procedure. A well-performed calibration procedure allows to obtain a satisfactory estimation of the camera parameters. To correctly map image points to the corresponding real world coordinates, the rigid transformation matrix must be estimated with respect to the user-defined reference system of the calibration master. Thus, if reference frame H changes or the camera frame C moves, it is necessary to estimate again the correct rigid transformation matrix. This procedure has been automatized by our software: a calibration script calculates matrix $\mathbf{K}$ using a set of calibration images acquired by the user, then it takes a new frame from the actual set-up to estimate the position of reference system H.

Images acquired by the camera are processed and the hand skeleton joints coordinates are calculated. By using the abovementioned formula, for each frame it is possible to obtain the real world coordinates of the hand skeleton joints in real time (image frame in Fig. 1).

B. Robot Workspace Calibration

The robot workspace refers to the space in which the robot moves (reference system W of Fig. 1) with respect to the user workspace (reference system H of Fig. 1). In this case, the user hand real-world position in reference system H is mapped to the new reference system W. The mapping between reference system H and reference system W is obtained easily if the two workspaces have the same dimension (matrix $\mathbf{[R|t]}$ is the identity matrix) or if one workspace is a scaled version of the other one (matrix $\mathbf{[R|t]}$ is the identity matrix multiplied by the scale factor).

To correctly move the robot in a cartesian position of reference system W, it is necessary to perform a calibration between reference system W and the robot reference system R. This procedure has been carried out experimentally by moving the robot (using its manual guidance mode) in different Cartesian positions of reference system W. The robot correct positioning on top of each calibration position has been assured by using a 3D-printed centering tool (Fig. 2). The tool must be centered manually on each calibration marker and secured in place, then the robot end effector can

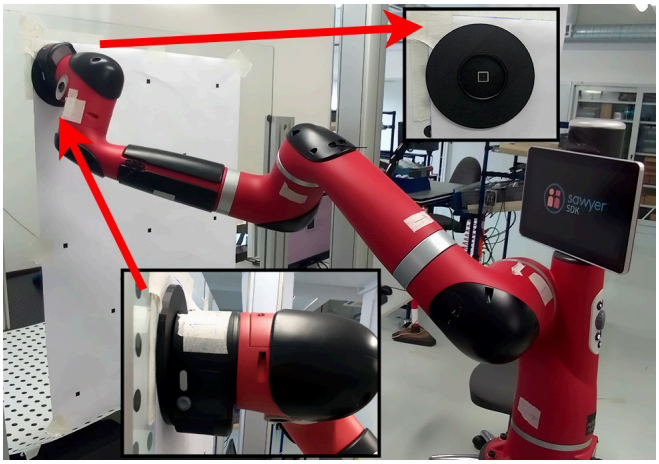

Fig. 2. Example of the robot calibration procedure. The calibration pose tool (top-right) is placed in correspondence of a calibration marker, then the robot end-effector is carefully placed inside the tool (bottom-left). The calibration pose tool cavity has been purposely made to fit the robot end-effector.

be moved on it and carefully positioned inside the purposely made circular cavity of the tool. When the positioning is complete, the robot coordinates (both in the Cartesian space and in the Joints space) corresponding to that particular marker (of which the positioning is known with respect to reference system W) can be extracted using ROS or the robot proprietary software.

When a satisfactory number of calibration positions has been acquired, it is possible to estimate the rigid transformation matrix between workspaces W and R. For the proposed system, the mapping procedure involves a plane motion, thus, the z axis is currently not considered.

Referring to the calibration example in Fig. 3, the plane position of a point $\mathbf{P}_1 \in \mathbb{R}^2$ is calculated with respect to both frame W ($P_{1,W}$) and frame R ($P_{1,R}$). The distance between

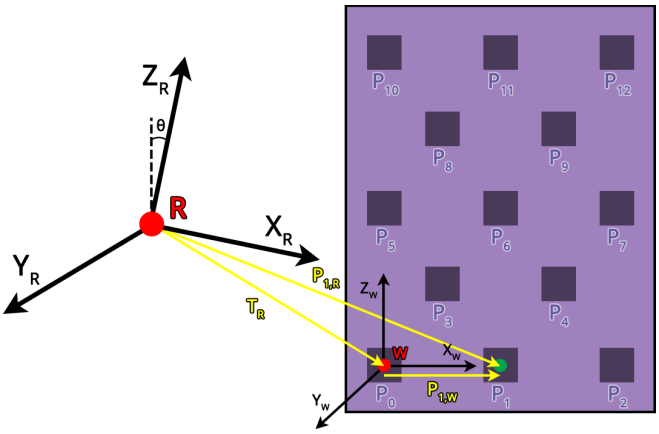

Fig. 3. Master used to calibrate the second user-defined reference system W with the robot reference frame R. The Figure illustrate the position of point $\mathbf{P_1}$ in both reference frames. To properly calibrate the system, the position of each point is required, both for frame W and R. In the procedure, 13 calibration points have been acquired.

the two reference frames is $\mathbf{T}_R \in \mathbb{R}^2$, hence:

$$\mathbf{P}_{1,R} = \mathbf{P}_{1,W} + \mathbf{T}_R \tag{3}$$

Using homogeneous coordinates it is possible to rewrite the previous equation as matrix products:

$$\mathbf{P}_{1,R} = \mathbf{M}_R^W \mathbf{P}_{1,W} \tag{4}$$

Where $\mathbf{M}_R^W \in \mathbb{R}^{3\times3}$ is the rigid transformation matrix between the two reference systems. By outlining equations from Equation 4, we obtain:

$$\begin{aligned} x_{P_{1,W}} &= x_{P_{1,R}} \cos\theta + y_{P_{1,R}} \sin\theta + x_{T_R} \\ y_{P_{1,W}} &= -x_{P_{1,R}} \sin\theta + y_{P_{1,R}} \cos\theta + y_{T_R} \end{aligned} \tag{5}$$

The aim of the calibration procedure is to calculate $\mathbf{M}_R^W$ in order to find the correct position and orientation of reference frame of robot workspace W with respect to the frame R. However, considering only one calibration position point $\mathbf{P}_1$, the system in Equation 5 results underdetermined, hence, a minimum of $n > 2$ calibration points is required to solve the system. To minimize the calibration error, $n = 13$ points have been considered. Thus, the system in Equation 5 becomes an overdetermined system $\mathbf{Ax} = \mathbf{b}$ that has been solved using the least square method, such as:

$$\mathbf{A} = \begin{bmatrix} x_{P_{0,R}} & y_{P_{0,R}} & 1 & 0 \\ -y_{P_{0,R}} & x_{P_{0,R}} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{P_{n-1,R}} & y_{P_{n-1,R}} & 1 & 0 \\ -y_{P_{n-1,R}} & x_{P_{n-1,R}} & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ x_{T_R} \\ y_{T_R} \end{bmatrix} \mathbf{b} = \begin{bmatrix} x_{P_{0,W}} \\ y_{P_{0,W}} \\ \vdots \\ x_{P_{n-1,W}} \\ y_{P_{n-1,W}} \end{bmatrix} \tag{6}$$

The rigid transformation matrix $\mathbf{M}_R^W$ used to identify the reference frame W from R is defined by the components of $\mathbf{x}$. Considering the overall schema in Fig. 1, the generic point $\mathbf{P} \in \mathbb{R}$ in the robot reference frame R with respect to the camera frame C is calculated as follows:

$$\mathbf{P}_R = \mathbf{M}_R^W \mathbf{P}_W \tag{7}$$

The same point in the robot workspace W is:

$$\mathbf{P}_W = K_s \mathbf{P}_H \tag{8}$$

Where $K_s \in \mathbb{R}$ is the a scaling factor between the robot and the user workspaces and $\mathbf{P}_H$ is defined in Equation 2. Finally, considering Equations (7,8) and Equation 2, the resulting point $\mathbf{P}_R$ in the robot reference frame using the camera coordinates is calculated as:

$$\mathbf{P}_R = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_s \mathbf{M}_R^W \left(s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \mathbf{K}^{-1} - \mathbf{t} \right) \mathbf{R}^{-1} \tag{9}$$

The space coordinates (u, v) will be the output of the hand-gesture recognition algorithm, while Cartesian coordinates (x, y, z) are the position set-points for the robot.

III. Hand-Gesture Recognition

The proposed teleoperation method is based on the recognition of the user hands skeleton.

Each frame acquired by the RGB camera (in our set-up, a Kinect v2 camera) is processed by the software, which leverages the OpenPose hand skeleton recognition network to predict the hand skeleton, following the details of [15]. The gesture recognition procedure is based on the position of the reference keypoint (red keypoint 0 in Fig. 4) and on the position of the four knuckles keypoints (blue keypoints $5, 9, 13, 17$ in Fig. 4). We defined two gestures used to carry out basic teleoperation tasks: the **open hand** gesture (Fig. 4, top-right) and the **index** gesture (Fig. 4, bottom-right).

We based the gesture recognition procedure on the recognition of the fingers, that may be opened or closed. To robustly recognize if a finger is opened or closed, we:

1) check if all the keypoints of the considered finger have been correctly predicted by the network (considering a prediction score threshold of 40%);
2) check if, in the case of the considered finger, the fingertip distance from reference keypoint 0 ($D_{0,F}$) and the knuckle distance from reference keypoint 0 ($D_{0,K}$) expressed as a percentage of the fingertip distance from reference keypoint 0 ($\frac{(D_{0,F}-D_{0,K})}{D_{0,F}}$) is less than 10%. If so, the keypoints of the finger are collapsed around the knuckle keypoint, thus the finger is considered closed;
3) check the Euclidean distances between the reference keypoint 0 and the last keypoint of each finger (pink keypoints $8, 12, 16, 20$ in Fig. 4). If the calculated distance of the index finger is greater than the others, the index finger is considered as opened with priority. This requirement has been proved useful to reduce the recognition error of the index gesture due to a wrong prediction of the fingers keypoints.

Considering the calibration procedure detailed in Section II, a certain position $\mathbf{P}_H$ of user workspace H corresponds to a certain robot end-effector position $\mathbf{P}_W$ in workspace W. Hence, to move the robot end-effector in position P_W using the software, users must:

1) place their hand in position $\mathbf{P}_H$ (corresponding to position $\mathbf{P}_W$), using the real-time visualization of the software as guidance (Fig. xx a);
2) perform the open hand gesture to allow the coordinate extraction (Fig. xx b);
3) perform the index gesture, carefully pointing the index finger to position $\mathbf{P}_H$ (Fig. xx c).

It is worth noting that, since the hand skeleton is obtained by a neural network which estimates the joints coordinates frame per frame, their position in consecutive frames may vary. Therefore, our software extracts N different $\mathbf{P}_H$ coordinates from N consecutive index gestures recognized in consecutive frames. The average coordinates are extracted to reduce positioning errors introduced by the hand skeleton recognition network. The higher the value of N, the higher the error reduction, at the cost of a higher delay before the final $\mathbf{P}_H$ coordinates are extracted. In our set-up, we set $N = 7$. After a position $\mathbf{P}_H$ is obtained, the corresponding robot position $\mathbf{P}_R$ is calculated and the robot is moved there using ROS. To perform a new robot movement, the procedure in Fig. xx must be repeated from the start.

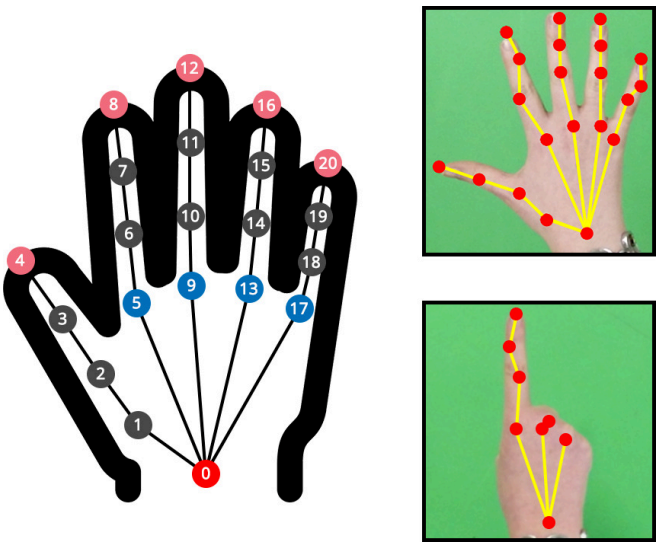

Fig. 4. Scheme of the hand skeleton predicted by OpenPose neural network. The red keypoint is the reference keypoint, the blue keypoints are the knuckles keypoints and the pink keypoints are the fingertips keypoints. Examples of correclty recognized gestures: open hand gesture (top-right) and index gesture (bottom-right).

IV. Experimental Evaluation

A reliable teleoperation system is obtained if the robot correctly moves to the desired position with a low positioning error. In the case of the proposed set-up, the positioning error seen when teleoperating our robot is obtained as a sum of different errors, as shown in Fig. 1.

First, when the user points the index finger to a Cartesian point in workspace H, OpenPose neural network estimates the index position in the image as a point $\mathbf{P}_O \in \mathbb{R}$ (keypoint 8 in Fig. 4). According to the filtering adopted and explained in Section III, the correspoding point is:

$$\mathbf{P}_O = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} \quad \forall\, 0 \leq n \leq N \tag{10}$$

The extracted index position $\mathbf{P}_O$ [px] corresponds to the camera image point $\mathbf{P}_C$ plus an estimation error $\mathbf{E}_{sk}$, obtained as the pixel distance between the real index position ($\mathbf{P}_C$) and the estimaed index keypoint position ($\mathbf{P}_O$). Hence:

$$\mathbf{P}_C = \mathbf{P}_O + \mathbf{E}_{sk} = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} \tag{11}$$

Thanks to the camera calibration procedure, a point in the acquired image frame $\mathbf{P}_C$ [px] corresponds to a Cartesian point $\mathbf{P}'_C$ [m] rototranslated in workspace H. $\mathbf{P}'_C$ corresponds to the real position $\mathbf{P}_H$ of the original $\mathbf{P}_C$ plus an estimation error $\mathbf{E}_H$ which depends on the accuracy of the calibration. In the following, we refer to generic points $\mathbf{P}'$

as the already rototranslated points according to Equations (2,7,8). Thus, $\mathbf{P}_H$ is defined as:

$$\mathbf{P}_H = \mathbf{P}'_C + \mathbf{E}_H = (\mathbf{P}_O + \mathbf{E}_{sk})' + \mathbf{E}_H \tag{12}$$

Since the image point we consider is $\mathbf{P}_O$, we obtain:

$$\mathbf{P}_H = \left(\begin{bmatrix} \frac{1}{N} \sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N} \sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} \right)' + \mathbf{E}_H \tag{13}$$

where the apex represents the conversion from pixels to meters (Section II).

The workspace where we want the robot to move is workspace W, thus, we must obtain the position of $\mathbf{P}_W$, which corresponds to $\mathbf{P}_H$ according to the specific mapping between the two workspaces. We obtain:

$$\mathbf{P}_W = \mathbf{P}'_H + \mathbf{E}_{map} \tag{14}$$

$$\mathbf{P}_W = \left(\begin{bmatrix} \frac{1}{N} \sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N} \sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H \right)' + \mathbf{E}_{map} \tag{15}$$

where $\mathbf{E}_{map}$ is the error caused by this mapping.

Finally, to correctly move the robot end-effector to $\mathbf{P}_W$, we must obtain the corresponding $\mathbf{P}_R$ in the robot reference system R. This correspondence is obtained from the robot calibration procedure detailed in Section II, therefore:

$$\mathbf{P}_R = \mathbf{P}'_W + \mathbf{E}_W \tag{16}$$

$$\mathbf{P}_R = \left(\begin{bmatrix} \frac{1}{N} \sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N} \sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H + \mathbf{E}_{map} \right)' + \mathbf{E}_W \tag{17}$$

It is made evident how moving from one reference frame to another introduces an error. Considering Equation 17, and that in our set-up $\mathbf{E}_{map}$ can be assumed equal to zero (because we kept workspace H and workspace W dimensions except for the scaling factor), we designed two experiments to assess if the positioning error obtained depends (i) on the estimation of the hand skeleton ($\mathbf{E}_{sk}$), (ii) on the camera calibration ($\mathbf{E}_H$) or (iii) on the robot workspace calibration ($\mathbf{E}_W$).

A. Evaluation of the skeleton estimation error

The positioning error due to the estimation of the hand skeleton $\mathbf{E}_{sk}$ made by OpenPose neural network has been evaluated considering the theoretical position of the index in the image $\mathbf{T} \in \mathbb{R}^2$ and the index joint position in the image $\mathbf{A} \in \mathbb{R}^2$ calculated by the software (Fig. xx) such as:

$$\mathbf{E}_{sk} = \mathbf{T} - \mathbf{A} = \begin{bmatrix} T_{x,n} \\ T_{y,n} \end{bmatrix} - \begin{bmatrix} A_{x,n} \\ A_{y,n} \end{bmatrix} \; \forall \, 1 \leq n \leq N \tag{18}$$

When users point to a position, they must keep the index gesture firmly in place until $N = 7$ consecutive index gesture have been successfully detected by the software. Hence, in this experiment the user hand moved to 14 different locations of workspace H, corresponding to $14 * 7 = 98$ couples of image frames and index joint estimations.

Theoretical positions have been manually selected from each acquired frame considering the tip of the index finger, while the actual positioning A of each frame corresponds to the predicted index keypoint obtained from OpenPose neural network. The user hand in the acquired frames is both vertically oriented and left or right oriented. An equal number of left-hand and right-hand frames have been selected for the evaluation.

To evaluate the relevance of $\mathbf{E}_{sk}$ we considered the average value $\overline{\mathbf{E}}_{sk}$ and the standard deviation $\sigma\mathbf{E}_{sk}$ along all 7 software detection for all 14 points (Table I). In each location, the (x, y) components of the average error has been calculated as:

$$\overline{\mathbf{E}}_{sk} = \begin{bmatrix} \overline{E}_{sk,x} = \frac{1}{N} \sum_{n=1}^{N} T_{xn} - A_{xn} \\ \overline{E}_{sk,y} = \frac{1}{N} \sum_{n=1}^{N} T_{yn} - A_{yn} \end{bmatrix} \tag{19}$$

It is worth noting that the mean values of $\overline{\mathbf{E}}_{sk}$ are extremely low: 2.06 [px] along the X-axis and -2.64 [px] along the Y-axis, as well for the mean of standard deviations $\sigma\mathbf{E}_{sk}$: 3.61 [px] along the X-axis and 3.49 [px] along the Y-axis. The negative sign represents the case when the actual positioning A is overstimated with respect to the corresponding theoretical positioning T.

Moreover, by calculating the standard deviation σ_x of the N different pixel positions along both for theoretical positions (σ_{Tx}, σ_{Ty}) and actual positions (σ_{Ax}, σ_{Ay}) (Table II), we know how much the user kept the hand firmly in place for each location. This result is an index useful to understand in which location the user moved the hand too much, thus reducing the accuracy of the estimation of the average index keypoint, which could lead to an incorrect placing of the robot end-effector.

TABLE I
AVERAGES AND STANDARD DEVIATIONS OF ERROR $\mathbf{E}_{sk}$

$\overline{E}_{sk,x}[px]$	$\overline{E}_{sk,y}[px]$	$\sigma E_{sk,x}[px]$	$\sigma E_{sk,y}[px]$
-1.29	-3.29	3.92	4.06
1.71	-2.14	3.06	3.00
-0.29	-4.86	4.49	6.49
1.43	-4.14	2.77	5.19
2.29	0.14	4.43	1.96
0.43	-5.43	2.56	1.18
4.14	-6.14	4.09	2.53
0.29	0.00	3.73	4.07
3.71	-3.00	3.84	3.82
1.29	-0.43	3.92	1.84
4.14	-1.43	4.09	4.03
3.29	-1.71	4.46	2.55
4.71	-3.00	2.37	4.24
3.00	-1.57	2.93	3.96

B. Evaluation of the robot positioning error

In our set-up, reference system H is placed horizontally with the camera mounted still at a 1 m distance (green horizontal square in Fig. 1). Reference system W, however, has been placed vertically on a glass pane (purple vertical square in Fig. 1).

To robustly assess the positioning of the robot end-effector, a 3D printed carrier holding a bright red laser has been

TABLE II

TEST LOCATIONS STANDARD DEVIATIONS

σ_{Tx}[px]	σ_{Ty}[px]	σ_{Ax}[px]	σ_{Ay}[px]
47.83	6.80	48.85	9.05
82.25	67.77	80.35	68.02
41.62	22.46	40.18	28.16
52.52	6.25	53.76	8.34
4.43	3.59	0.00	4.55
2.56	1.18	0.00	0.00
78.10	87.03	78.57	86.21
150.60	26.54	146.98	30.17
97.83	15.08	98.48	15.09
77.88	21.88	76.35	23.49
79.02	20.71	78.26	23.78
8.91	43.75	11.75	44.54
2.26	14.37	4.55	14.04
27.01	98.68	28.53	97.18

mounted on the end effector (*Lasiris laser 635nm, 10mW*) (Fig. 5, bottom-right corner).

When the robot is moved to a certain theoretical position T, the laser will point to its actual positioning A. To correctly visualize and measure the robot workspace and the laser positioning, an RGB camera (*IDS Imaging UI-1460C*) has been mounted behind the glass pane. A measuring software has been developed using LabVIEW to measure the distance (E) between the theoretical position T (calculated as the barycenter of the black square of the experimental master, represented as the green dot in Fig. 5, top-left corner) and the actual positioning A (red dot in Fig. 5, top-left corner).

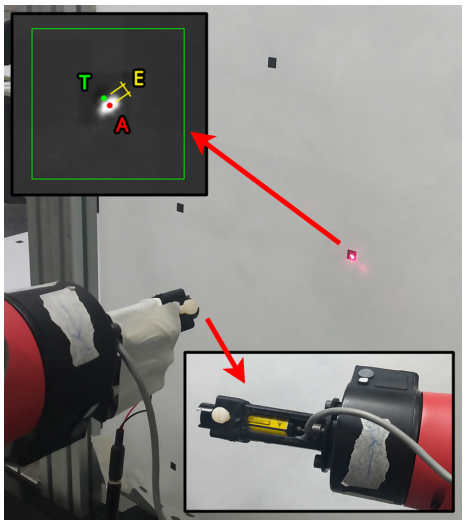

Fig. 5. Example of a robot positioning evaluation. The image shows a close-up of the 3D printed laser carrier used (bottom-right) and the corresponding view of the measuring software developed (top-left).

We moved the robot in 5 theoretical positions, corresponding to markers P3, P4, P6, P8 and P9 in Fig. 3, for a total of 3 times per each theoretical position. To do that, we used the Cartesian positions corresponding to the circular markers barycenters (theoretical positions) to move the robot using ROS. Hence, this procedure avoids considering the hand skeleton estimation errors. The resulting distances in millimeters are reported in Table III.

TABLE III

ROBOT POSITIONING ERRORS

P3 [mm]	**P4 [mm]**	**P6 [mm]**	**P8 [mm]**	**P9 [mm]**
6.27	8.86	7.75	6.84	6.89
6.48	5.64	7.88	6.70	5.64
6.77	5.29	7.87	6.99	5.29

We computed the resulting mean distance and the standard deviation of each marker position, reported in Table IV. The average mean distance and the average standard deviation achieved by the robot are 6.74 [mm] and 0.53 [mm] respectively.

TABLE IV

MEAN AND STANDARD DEVIATION

P3 [mm]	**P4 [mm]**	**P6 [mm]**	**P8 [mm]**	**P9 [mm]**
6.51	6.60	7.84	6.84	5.94
0.21	1.61	0.06	0.12	0.69

V. CONCLUSIONS

Conclusioni sul progetto/esperimenti ottenuti. Problematiche incontrate e come sono state risolte. Future developments.

REFERENCES

[1] H. A. Yanco and J. L. Drury, "A Taxonomy for Human-Robot Interaction," *Engineering*, p. 9, 2002.

[2] J. Vertut and P. C. Coeffet, *Robot Technology; vol. 3A Teleoperation and Robotics Evolution and Development*. 1986.

[3] J. Rebelo, T. Sednaoui, E. B. Den Exter, T. Krueger, and A. Schiele, "Bilateral robot teleoperation: A wearable arm exoskeleton featuring an intuitive user interface," *IEEE Robot. Autom. Mag.*, vol. 21, no. 4, pp. 62–69, 2014.

[4] X. Lv, M. Zhang, F. Cui, and X. Zhang, "Teleoperation of robot based on virtual reality," *Proc. - 16th Int. Conf. Artif. Real. Telexistence - Work. ICAT 2006*, pp. 400–403, 2006.

[5] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[6] C. Glover, B. Russell, A. White, M. Miller, and A. Stoytchev, "An effective and intuitive control interface for remote robot teleoperation with complete haptic feedback," *Proc. 2009 Emerg. Technol. Conf. (ETC), Ames, IA, USA*, 2009.

[7] R. Boboc, H. Moga, and D. TALAB, "A Review of Current Applications in Teleoperation of Mobile Robots," *Bull. Transilv. Univ. Brasov Ser. I Eng. Sci.*, vol. 5, no. 54, pp. 9–16, 2012.

[8] J. Vogel, C. Castellini, and P. van der Smagt, "EMG-based teleoperation and manipulation with the DLR LWR-III," pp. 672–678, 2011.

[9] H. F. Hassan, S. J. Abou-Loukh, and I. K. Ibraheem, "Teleoperated robotic arm movement using electromyography signal with wearable Myo armband," *J. King Saud Univ. - Eng. Sci.*, no. xxxx, 2019.

[10] L. Roveda, S. Haghshenas, A. Prini, T. Dinon, N. Pedrocchi, F. Braghin, and L. M. Tosatti, "Fuzzy Impedance Control for Enhancing Capabilities of Humans in Onerous Tasks Execution," in *2018 15th Int. Conf. Ubiquitous Robot. UR 2018*, pp. 406–411, Institute of Electrical and Electronics Engineers Inc., aug 2018.

[11] J. Kofman, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Trans. Ind. Electron.*, vol. 52, no. 5, pp. 1206–1219, 2005.

[12] S. Livatino, G. Muscato, and F. Privitera, "Stereo viewing and virtual reality technologies in mobile robot teleguide," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1343–1355, 2009.

[13] L. Peppoloni, F. Brizzi, C. A. Avizzano, and E. Ruffaldi, "Immersive ROS-integrated framework for robot teleoperation," *2015 IEEE Symp. 3D User Interfaces, 3DUI 2015 - Proc.*, pp. 177–178, 2015.

[14] H. Hedayati, M. Walker, and D. Szafir, "Improving Collocated Robot Teleoperation with Augmented Reality," *ACM/IEEE Int. Conf. Human-Robot Interact.*, pp. 78–86, 2018.

[15] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.
[16] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.
[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
[18] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2001.