2020 17th International Conference on Ubiquitous Robots (UR)
June 22-26, 2020. Kyoto, Japan

Hands-Free: a robot augmented reality teleoperation system

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

Abstract— **In this paper the novel teleoperation method "Hands-Free" is presented. Hands-Free is a vision-based augmented reality system that allows users to teleoperate a robot end-effector with their hands in real time. The system leverages OpenPose neural network to detect the human operator hand in a given workspace, achieving an average inference time of 0.15 s. The user index position is extracted from the image and converted in real world coordinates to move the robot end-effector in a different workspace.**

The user hand skeleton is visualized in real-time moving in the actual robot workspace, allowing the user to teleoperate the robot intuitively, regardless of the differences between the user workspace and the robot workspace.

Since a set of calibration procedures is involved to convert the index position to the robot end-effector position, we designed three experiments to determine the different errors introduced by conversion. A detailed explanation of the mathematical principles adopted in this work is provided in the paper.

Finally, the proposed system has been developed using ROS and is publicly available at the following GitHub repository: https://github.com/Krissy93/hands-free-project.

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 where the user presence is not allowed. Therefore, the operator needs to move the robot remotely [2]. A plenty of human-machine interfaces for teleoperation have been developed considering a mechanical interface, including exoskeletons [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], [7] is achieved. Other controllers may include mouses, switchboxes, keyboards, touch-screens and joysticks. Joysticks in particular are usually a better type of control device than others, because the operators better identify with the task [8]. Among the systems where bi-lateral teleoperation is not required, a teleoperation system is defined by means of electromyography (EMG) signals of the muscular activity [9], [10]. However, as recently highlighted in [11], the electromyography may be affected by difficulties in processing the EMG signals for amplitude and spectral analysis, reducing the efficiency of the method 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 [12], the accuracy of object gripping tasks is improved by mean of a contactless vision-based robot teleoperation method. In [13] a stereo vision system improved the performance of a mobile robot teleoperation application. In [14] a vision and gestures-based communication has been adopted to robustly achieve several task in collaboration with the human operator.

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 improve the feedback information; in fact, the operator feels like being physically present in the remote environment, improving the immersion level. Immersivity is, in fact, one of the most important reasons for using virtual and augmented reality [8]. A successful example of an immersive virtual reality teleoperation system is presented in [15]. Here, the authors developed a whole body multi-modal semiautonomous teleoperation framework which leverages both visual systems and haptic devices, solving the issue of loss of dexterity and information. An augmented reality system for teleoperation based on the Leap Motion (LM) controller is presented in [16]. For its application domain, the LM controller is considered an accurate sensor [17], 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 user immersivity concept.

For these reasons, we present a novel robot augmented reality teleoperation system that exploits RGB cameras, which supply greater measuring distance if compared with

This work was partially funded by the European project H2020-NMBP-FOF-2018-ShareWork (grant agreement 820807).

Cristina Nuzzi, Roberto Pagani, Simone Pasinetti, Gabriele Coffetti 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

the LM controller, are easy to use and commonly available on the market at a reduced price. A ROS-based framework has been developed to supply hand tracking and hand-gesture recognition features, exploiting the OpenPose software [18], [19] based on the Deep Learning framework Caffe [20]. This, in combination with the ease of availability of an RGB camera, leads the framework to be strongly open-source oriented and highly replicable on all the ROS-based platform. Hands-Free is an augmented reality teleoperation tool, in fact the user hand skeleton is visualized in real-time over a representation of the robot workspace, allowing the user to teleoperate the robot intuitively. It is worth noting that the proposed system does not include the Z-axis control in this first version. This is due to the high precision and sensitivity required to robustly control the third axis, a precision that 3D vision systems are not able to provide unless very expensive devices are adopted. Our aim is to further develop the system in order to include the third axis control in a future release.

The proposed system includes: a rigorous procedure for robot workspace calibration and a mapping policy between the coordinate system of the user and the robot (Section II), and a deep learning approach for the hand-gesture recognition by means of hand keypoints estimation (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 user workspace that (i) may be reached by the user hand and (ii) that 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 it may be easily realized following standard procedures (see [21] for a more detailed explaination). 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 formula:

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

where homogeneous coordinates are used. However, since we are looking for the position of point $\mathbf{P}_H$ in frame H by back-projecting a 2D point to 3D, it is necessary to invert equation (1):

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

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

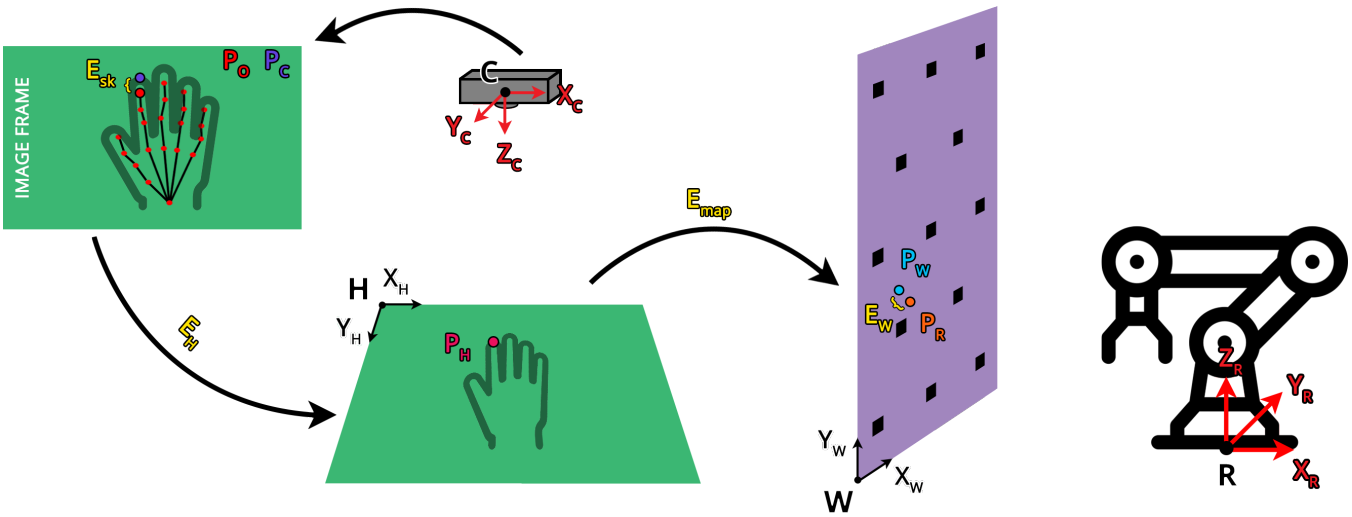

Fig. 1. Scheme of the calibration steps.

the calibration master. Thus, if reference frame H changes or the camera frame C moves, it is necessary to estimate the correct rigid transformation matrix again. 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 formulas, it is possible to obtain the real world coordinates of the hand skeleton joints in real time for each acquired frame (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).

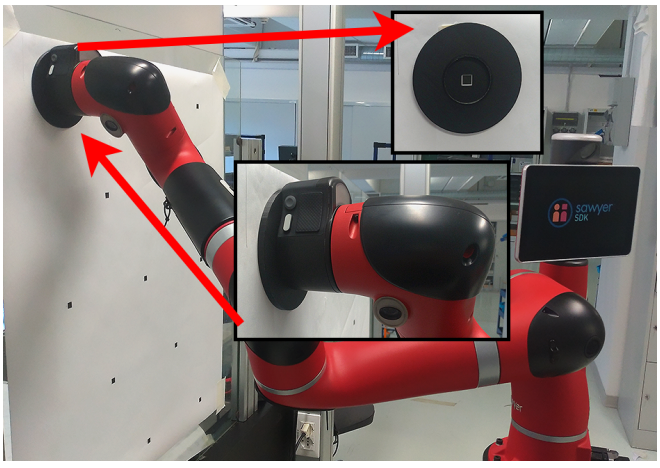

Fig. 2. Example of the robot calibration procedure. The calibration pose tool is placed in correspondence of a calibration marker, then the robot end-effector is carefully placed inside the tool. The calibration pose tool cavity has been purposely made to fit the robot end-effector.

The tool must be centered manually on each calibration marker and secured in place, then the robot end effector can 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. The proposed system currently involves a planar motion, thus, the mapping procedure does not consider the z axis.

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 ($P_{1,R}$).

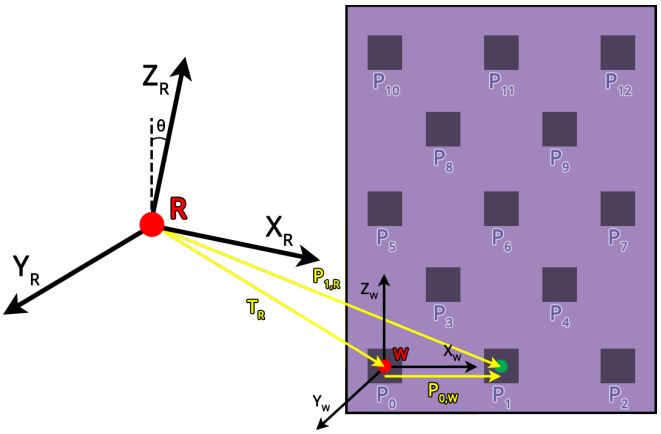

Fig. 3. Master used to calibrate the second user-defined reference system W with the robot reference frame R. The figure illustrate the position of point $\mathbf{P_1}$ in both reference frames. To properly calibrate the system, the position of each point is required, both for frame W and R. In the procedure,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

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

$$\mathbf{P}_{1,R} = \mathbf{P}_{0,W} + \mathbf{T}_R \tag{3}$$

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

$$\mathbf{P}_{1,R} = \mathbf{M}_R^W \mathbf{P}_{0,W} \tag{4}$$

Where $\mathbf{M}_R^W \in \mathbb{R}^{3\times3}$ is the rigid transformation matrix between the two reference systems defined as:

$$\mathbf{M}_R^W = \begin{bmatrix} \cos\theta & \sin\theta & x_{T_R} \\ -\sin\theta & \cos\theta & y_{T_R} \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Hence, by outlining equations from (4), we obtain:

$$\begin{cases} x_{P_{0,W}} = x_{P_{1,R}} \cos\theta + y_{P_{1,R}} \sin\theta + x_{T_R} \\ y_{P_{0,W}} = -x_{P_{1,R}} \sin\theta + y_{P_{1,R}} \cos\theta + y_{T_R} \end{cases} \tag{6}$$

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 (6) 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 (6) becomes an overdetermined system $\mathbf{Ax} = \mathbf{b}$ that has been solved using the least square method, such as:

$$\mathbf{A} = \begin{bmatrix} x_{P_{0,R}} & y_{P_{0,R}} & 1 & 0 \\ -y_{P_{0,R}} & x_{P_{0,R}} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{P_{n-1,R}} & y_{P_{n-1,R}} & 1 & 0 \\ -y_{P_{n-1,R}} & x_{P_{n-1,R}} & 0 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ x_{T_R} \\ y_{T_R} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} x_{P_{0,W}} \\ y_{P_{0,W}} \\ \vdots \\ x_{P_{n-1,W}} \\ y_{P_{n-1,W}} \end{bmatrix} \tag{7}$$

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:

$$\mathbf{P}_R = \mathbf{M}_R^W \mathbf{P}_W \tag{8}$$

The same point in the robot workspace W is:

$$\mathbf{P}_W = K_s \mathbf{P}_H \tag{9}$$

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 (2). Finally, considering equations (8), (9) and equation (2), the resulting point $\mathbf{P}_R$ in the robot reference frame using the camera coordinates is calculated as:

$$\mathbf{P}_R = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_s \mathbf{M}_R^W \mathbf{R}^{-1} \left(\mathbf{K}^{-1} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \mathbf{t} \right) \tag{10}$$

The space coordinates (u, v) are 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 [18]. 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, which may be opened or closed. To robustly recognize if a finger is opened or closed, the procedure is the following:

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) in the case of the considered finger, check 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 ($\frac{(D_{0,F} - D_{0,K})}{D_{0,F}}$). If this distance is less than 10%, the finger keypoints 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.

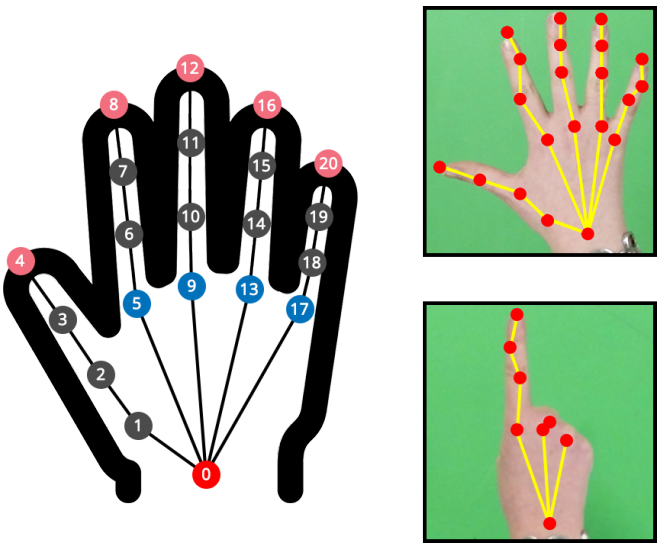

Fig. 4. Scheme of the hand skeleton predicted by OpenPose neural network. The red keypoint is the reference keypoint, the blue keypoints are the knuckles keypoints and the pink keypoints are the fingertips keypoints. Examples of correclty recognized gestures: open hand gesture (top-right) and index gesture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

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, the 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;
2) perform the open hand gesture to allow the coordinate extraction;

3) perform the index gesture by carefully pointing the index finger to position $\mathbf{P}_H$.

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 consectuive 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 the ROS interface. To perform a new robot movement, the procedure must be repeated from the start.

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 proposed set-up, the positioning error 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:

$$\mathbf{P}_O = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} \; \forall\, 0 \leq n \leq N \tag{11}$$

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

$$\mathbf{P}_C = \mathbf{P}_O + \mathbf{E}_{sk} = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} \tag{12}$$

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 $\mathbf{P}_C$ with an estimation error $\mathbf{E}_H$ which depends on the accuracy of the calibration and on the camera resolution. Thus, $\mathbf{P}_H$ is defined as:

$$\mathbf{P}_H = \mathbf{P}'_C + \mathbf{E}_H = \mathbf{P}_O + \mathbf{E}_{sk} + \mathbf{E}_H \tag{13}$$

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

$$\mathbf{P}_H = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H \tag{14}$$

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

The workspace where we want to move the robot is workspace W, therefore 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:

$$\mathbf{P}_W = \mathbf{P}_H + \mathbf{E}_{map} \tag{15}$$

$$\mathbf{P}_W = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H + \mathbf{E}_{map} \tag{16}$$

where $\mathbf{E}_{map}$ is the error caused by the 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:

$$\mathbf{P}_R = \mathbf{P}_W + \mathbf{E}_W \tag{17}$$

$$\mathbf{P}_R = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} x_{P_{On}} \\ \frac{1}{N}\sum_{n=1}^{N} y_{P_{On}} \end{bmatrix} + \mathbf{E}_{sk} + \mathbf{E}_H + \mathbf{E}_{map} + \mathbf{E}_W \tag{18}$$

It is made evident that moving from one reference frame to another introduces errors. Considering (18), and $\mathbf{E}_{map}$ assumed equal to zero (because we kept workspace H and workspace W dimensions equal), we designed two experiments to assess if the positioning error obtained depends (i) on the camera calibration ($\mathbf{E}_H$), (ii) on the estimation of the hand skeleton ($\mathbf{E}_{sk}$) or (iii) on the robot workspace calibration and robot intrinsic characteristics ($\mathbf{E}_W$).

A. Evaluation of the camera error

With the calibration procedure detailed in Section II, a point in the image frame corresponds to a point in workspace H. The conversion from pixel coordinates to real world coordinates introduces an error ($\mathbf{E}_H$), which has been estimated considering the difference between (i) theoretical positions $\mathbf{T} \in \mathbb{R}^2$ [mm], corresponding to the calibration points P_p real world coordinates in workspace $\mathbf{H}$ (Fig. 3) with respect to reference point P_0 and (ii) the converted actual positions $\mathbf{A} \in \mathbb{R}^2$ [mm], corresponding to real world coordinates of the same calibration points, obtained converting their pixel coordinates (taken from the image frame) in real world coordinates using matrices $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ obtained from equation (2). Hence, the total error $\mathbf{E}_H$ is obtained as the sum of P positioning errors, where P is the total number of calibration points considered:

$$\mathbf{E}_H = \sum_{p=0}^{P} (\mathbf{T}_p - \mathbf{A}_p) = \sum_{p=0}^{P} \left(\begin{bmatrix} T_{x,p} \\ T_{y,p} \end{bmatrix} - \begin{bmatrix} A_{x,p} \\ A_{y,p} \end{bmatrix} \right) \tag{19}$$

Table I reports the values obtained for each point P_p. Theoretical positions $\mathbf{T}_p$ are the real world coordinates of the point in workspace $\mathbf{H}$, obtained as the center of each marker of the workspace. It is worth noting that we printed the workspace frame with markers positioned according to theoretical positions in Table I, 1 cm wide each. The actual positions $\mathbf{A}_p$ are the calculated coordinates of the point starting from its corresponding pixel coordinates in the image frame, and the corresponding error is $\mathbf{E}_H^p$. The negative sign represents the case when the actual position $\mathbf{A}_p$ is

overstimated with respect to the corresponding theoretical position $\mathbf{T}_p$.

TABLE I
THEORETICAL POSITIONS $\mathbf{T}_p$, ACTUAL POSITIONS $\mathbf{A}_p$ AND CALIBRATION ERROR $\mathbf{E}_H^p$ OF EACH CALIBRATION POINT

Points	$\mathbf{T}_p$ [mm]		$\mathbf{A}_p$ [mm]		$\mathbf{E}_H^p$ [mm]	
	x	y	x	y	x	y
P_0	0.00	0.00	-0.66	-0.68	0.66	0.68
P_1	225.00	0.00	234.17	0.99	-9.17	-0.99
P_2	450.00	0.00	462.42	4.52	-12.42	-4.52
P_3	112.50	175.00	118.14	180.19	-5.64	-5.19
P_4	337.50	175.00	349.22	180.91	-11.72	-5.91
P_5	0.00	350.00	-0.71	360.32	-0.71	-10.32
P_6	225.00	350.00	234.13	359.17	-9.13	-9.17
P_7	450.00	350.00	464.28	357.07	-14.28	-7.07
P_8	112.50	525.00	118.19	541.19	-5.60	-16.19
P_9	337.50	525.00	351.06	538.16	-13.56	-13.16
P_{10}	0.00	700.00	-2.64	725.08	2.64	-25.08
P_{11}	225.00	700.00	235.02	721.12	-10.02	-21.12
P_{12}	450.00	700.00	469.87	716.21	-19.87	-16.21

The resulting mean error and the standard deviation of values in Table I are -8.26 [mm] and 6.36 [mm] respectively along the X-axis, and -10.33 [mm] and 7.40 [mm] respectively along the Y-axis.

B. Evaluation of the skeleton estimation error

For each considered point P the positioning error $\mathbf{E}_{sk}$ due to the estimation of the hand skeleton made by OpenPose neural network has been evaluated similarly to the estimation procedure of $\mathbf{E}_H$. Theoretical positions $\mathbf{T}_p$ [px] are the index position in the image frame. Actual positions $\mathbf{A}_p$ [px] are the index joint position in the image frame calculated by OpenPose. It is worth noting that when users point their finger 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 both theoretical and actual positions are obtained as the mean value over N consecutive frames. Error $\mathbf{E}_{sk}$ is therefore calculated as:

$$\begin{aligned}\mathbf{E}_{sk} &= \sum_{p=0}^{P}\left(\frac{1}{N}\sum_{n=0}^{N}(\mathbf{T}_p^n - \mathbf{A}_p^n)\right) = \\ &= \sum_{p=0}^{P}\left(\frac{1}{N}\sum_{n=0}^{N}\left(\begin{bmatrix}T_{x,p}^n\\ T_{y,p}^n\end{bmatrix} - \begin{bmatrix}A_{x,p}^n\\ A_{y,p}^n\end{bmatrix}\right)\right)\end{aligned} \tag{20}$$

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 ($\mathbf{H}_p$). Theoretical positions $\mathbf{T}_p$ have been manually selected from each acquired frame considering the tip of the index finger, while the actual positioning $\mathbf{A}_p$ of each frame corresponds to the predicted index keypoint 8 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 $\overline{\mathbf{E}}_{sk}^p$ and the standard deviation $\sigma\mathbf{E}_{sk}^p$ calculated on the 7 consecutive detections of each point P, resulting in the values in Table II. Please note that points P considered in this experiment refer to positions in workspace H chosen randomly by the user.

TABLE II
THEORETICAL POSITIONS $\mathbf{T}_p$, ACTUAL POSITIONS $\mathbf{A}_p$ AND AVERAGES AND STANDARD DEVIATIONS OF ERROR $\mathbf{E}_{sk}$

Points	$\mathbf{T}_p$ [px]		$\mathbf{A}_p$ [px]		$\overline{\mathbf{E}}_{sk}^p$ [px]		$\sigma\mathbf{E}_{sk}^p$[px]	
	x	y	x	y	x	y	x	y
P_1	332	346	333	349	-1.29	-3.29	3.92	4.06
P_2	485	276	483	278	1.71	-2.14	3.06	3.00
P_3	437	264	437	269	-0.29	-4.86	4.49	6.49
P_4	509	278	507	282	1.43	-4.14	2.77	5.19
P_5	554	269	552	269	2.29	0.14	4.43	1.96
P_6	552	275	552	280	0.43	-5.43	2.56	1.18
P_7	614	199	610	205	4.14	-6.14	4.09	2.53
P_8	411	247	411	247	0.29	0.00	3.73	4.07
P_9	587	273	584	276	3.71	-3.00	3.84	3.82
P_{10}	401	255	400	256	1.29	-0.43	3.92	1.84
P_{11}	380	269	376	271	4.14	-1.43	4.09	4.03
P_{12}	446	174	442	176	3.29	-1.71	4.46	2.55
P_{13}	455	94	450	97	4.71	-3.00	2.37	4.24
P_{14}	496	243	493	245	3.00	-1.57	2.93	3.96

In each point P, the (x, y) components of the average error has been calculated as:

$$\overline{\mathbf{E}}_{sk}^p = \begin{bmatrix}\overline{E}_{sk,x}^p = \frac{1}{N}\sum_{n=1}^{N}(T_{x,n} - A_{x,n})\\ \overline{E}_{sk,y}^p = \frac{1}{N}\sum_{n=1}^{N}(T_{y,n} - A_{y,n})\end{bmatrix} \tag{21}$$

It is worth noting that the mean value of $\overline{\mathbf{E}}_{sk}^p$ is 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 $\sigma\mathbf{E}_{sk}^p$: 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 of the N different pixel positions along both theoretical positions (σT_x, σT_y) and actual positions (σA_x, σA_y) (Table II), we know how much the user kept the hand firmly in place for each location. This result is 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.

C. Evaluation of the robot workspace error

In our set-up, reference system H is placed horizontally with the camera mounted 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). The robot workspace error $\mathbf{E}_W$ is composed by (i) an error $\mathbf{E}_W^a$ only related to the calibration procedure and (ii) the robot position error $\mathbf{E}_W^b$ due to physical robot characteristics:

$$\mathbf{E}_W = \mathbf{E}_W^a + \mathbf{E}_W^b \tag{22}$$

To robustly assess the robot positioning error $\mathbf{E}_W^b$ of the end-effector, a 3D printed carrier holding a bright red laser

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

When the robot is moved with its own software 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 centroid 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).

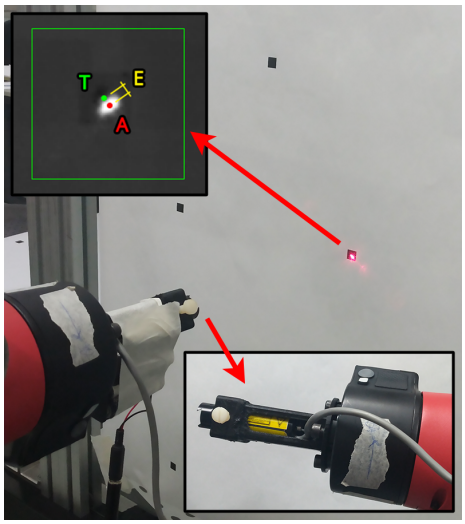

Fig. 5. Example of a robot positioning evaluation. The image shows a close-up of the 3D printed laser carrier used (bottom-right) and the corresponding view of the measuring software developed (top-left).

We moved the robot in 7 theoretical positions, corresponding to points P_3, P_4, P_5, P_6, P_7, P_8 and P_9 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 centroids (theoretical positions) to move the robot using the ROS interface. Hence, this procedure avoids considering the hand skeleton estimation errors.

The error $\mathbf{E}_W^a$ has been calculated by considering the same points with respect to the robot frame, calculated using equation 8. Since the rigid transformation matrix is obtained from the calibration least square method, this procedure allows to isolate the calibration error. $\mathbf{E}_W^a$ is defined as the absolute difference between the robot theoretical position T and the actual position A calculated using equation 8. The resulting distances in millimeters are reported in Table III, where $\mathbf{E}_W^b$ values are reported as the average among the 3 tests.

We computed the resulting mean distance and the standard deviation of each marker position, reported in Table III. Considering $\mathbf{E}_W^a$, the average mean distance is 2.04 [mm] for the X-axis and 0.49 [mm] for the Y-axis. While $\mathbf{E}_W^b$ has an average of 0.66 [mm] for the X-axis and 1.66 [mm] for the Y-axix. The standard deviation among the 3 tests for $\mathbf{E}_W^b$ is 0.13 [mm] for the X-axis and 0.09 [mm] for the Y-axis.

TABLE III
NORM, AVERAGES AND STANDARD DEVIATIONS OF ERROR $\mathbf{E}_W^b$

Points	$\mathbf{E}_W^a$ [mm]		$\mathbf{E}_W^b$ [mm]		$\sigma\mathbf{E}_W^b$ [mm]	
	x	y	x	y	x	y
P_3	0.30	1.10	0.26	1.86	0.01	0.05
P_4	1.70	0.00	0.19	1.59	0.11	0.27
P_5	6.70	0.70	1.63	0.58	0.14	0.05
P_6	0.60	0.40	0.77	1.44	0.04	0.05
P_7	2.60	0.20	0.70	1.15	0.02	0.12
P_8	0.40	0.50	0.20	1.50	0.17	0.06
P_9	2.00	0.50	0.91	1.40	0.40	0.04

It is worth noting that in our experiments we used the end-effector position in the cartesian space obtained using the robot ROS interface. According to the task repeatability reported in the robot datasheet, we have assumed that it can reach the desired position with a minimal error. This assumption has been proved true, in fact the results reported in Table III suggest that a joint-level calibration of the robot was not needed.

V. CONCLUSIONS

In this paper a novel teleoperation method called "Hands-Free" is presented. The proposed system adopts an RGB visual system properly calibrated and OpenPose neural network to acquire the user index finger location in the user workspace H. The estimated position is then converted in the corresponding robot end-effector position with respect to a second workspace W in which the robot operates.

Given the characteristics of Hands-Free, a number of positioning errors sum up and affect the final robot positioning. To correctly determine them, we performed 3 experiments: the first one was aimed at identifying the camera calibration error $\mathbf{E}_H$ which affects the conversion between the pixel coordinates of a point in the image frame to the corresponding real world coordinates in workspace H. The second one was aimed at identifying the skeleton estimation error $\mathbf{E}_{sk}$ which is due to OpenPose neural network. The third error was aimed at identifying the robot workspace error $\mathbf{E}_W$, which depends on both (i) the calibration procedure of the robot with respect to workspace W and (ii) the robot intrinsic characteristics.

Considering the results obtained from the experiments we can conclude that using Hands-Free the final positioning error is due mostly to $\mathbf{E}_H$, and that this error can be reduced if a better RGB camera is used compared to the Kinect v2 camera adopted in this paper.

Hence, Hands-Free is a solid alternative to more expensive teleoperation systems such as the Leap Motion or haptic gloves, providing the user with a wide range of measuring distance according to the RGB camera adopted. The key feature of the proposed system is its simplicity, allowing the user to build its own teleoperation system with just an RGB camera, following the calibration procedure described in the paper. As a further contribution,

we released the code in our public repository on GitHub: https://github.com/Krissy93/hands-free-project.

In future developments we will add the Z-axis spatial control to Hands-Free, considering both 3D visual systems and traditional solutions such as haptic gloves or controllers. Wearable custom-made controllers will also be considered. Considering the results obtained in these experiments, we will reproduce them adopting a different camera to better improve the system performances.

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] M. Tavakoli, A. Aziminejad, R. V. Patel, and M. Moallem, "High-fidelity bilateral teleoperation systems and the effect of multimodal haptics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 6, pp. 1512–1528, 2007.

[8] 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.

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

[10] H. F. Hassan, S. J. Abou-Loukh, and I. K. Ibraheem, "Teleoperated robotic arm movement using electromyography signal with wearable myo armband," *Journal of King Saud University - Engineering Sciences*, 2019.

[11] 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.

[12] 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.

[13] 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.

[14] C. Nuzzi, S. Pasinetti, R. Pagani, F. Docchio, and G. Sansoni, "Hand gesture recognition for collaborative workstations: A smart command system prototype," in *New Trends in Image Analysis and Processing – ICIAP 2019*, pp. 332–342, Springer International Publishing, 2019.

[15] ChangSu Ha, Sangyul Park, Jongbeom Her, Inyoung Jang, Yongseok Lee, Gun Rae Cho, Hyoung Il Son, and Dongjun Lee, "Whole-body multi-modal semi-autonomous teleoperation of mobile manipulator systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 164–170, 2015.

[16] 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.

[17] 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.

[18] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[19] 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.

[20] 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.

[21] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, Nov 2000.