User-oriented Natural Human-Robot Control with Thin-Plate Splines and LRCN

Bruno Lima$^1$ · Lucas Amaral$^1$ · Givanildo Nascimento-Jr$^1$ · Victor Mafra$^1$ · Bruno Georgevich Ferreira$^1$ · Tiago Vieira$^1$ · Thales Vieira$^1$

Received: 25 May 2021 / Accepted: 7 December 2021 / Published online: 11 March 2022
© The Author(s), under exclusive licence to Springer Nature B.V. 2022

Abstract

We propose a real-time vision-based teleoperation approach for robotic arms that employs a single depth-based camera, exempting the user from the need for any wearable devices. By employing a natural user interface, this novel approach leverages the conventional fine-tuning control, turning it into a direct body pose capture process. The proposed approach is comprised of two main parts. The first is a nonlinear customizable pose mapping based on Thin-Plate Splines (TPS), to directly transfer human body motion to robotic arm motion in a nonlinear fashion, thus allowing matching dissimilar bodies with different workspace shapes and kinematic constraints. The second is a Deep Neural Network hand-state classifier based on Long-term Recurrent Convolutional Networks (LRCNs) that exploits the temporal coherence of the acquired depth data. We validate, evaluate and compare our approach through both classical cross-validation experiments of the proposed hand state classifier; and user studies over a set of practical experiments involving variants of pick-and-place and manufacturing tasks. Results revealed that LRCNs outperform single image Convolutional Neural Networks; and that users’ learning curves were steep, thus allowing the successful completion of the proposed tasks. When compared to a previous approach, the TPS approach revealed no increase in task complexity and similar times of completion, while providing more precise operation in regions closer to workspace boundaries.

Keywords Human-robot interaction · Teleoperation · Natural user interface · Kinematics mapping · Thin-plate spline · Long-term recurrent convolutional networks

1 Introduction

Human-Robot Interaction (HRI) comprises the field that allows humans and robots to communicate with the aim of accomplishing well-defined goals. This interaction can be either close or distant, depending on the aspects of space and time involved. According to [13], two paradigms of human-robot interaction were always present within the robotics context: (1) teleoperation and (2) supervisory control. Teleoperation is the act of remotely operating a robot, either mobile or fixed, without necessarily being physically present in the same environment. Supervisory control is the process of supervising (watching and possibly acting for correction) the behaviour of an autonomous system or process.

A common application example for discussed fields takes place in manufacturing industries with human cognitive inspection [47]. A surveillance mobile robot equipped with a camera and a robotic arm may inspect and take simple actions such as activating an emergency alarm or turning up a crank. Another application would be aiding search and rescue tasks in potentially harmful places [8]. A mobile robot may survey an unknown area while looking for victims and assessing structural stability. In both these applications, advantages reside in the ability to address complex problems from unstructured and possibly unknown environments, combining human cognition with sensory tools and low-level autonomous behavior of robots.

In the context of robotics and automation, Learning from Demonstration (LfD) is the process where a robot acquires new skills by imitating an expert, assuming the existence
of a knowledge transferring procedure, as discussed by Ravinchantar et al. [39]. Within the LfD field, teleoperation can be used as one of such procedures, besides kinesthetic teaching and passive observation. LfD is appropriate for situations where ideal behavior might not be easily scripted or defined as an easy optimization problem. By trying to replicate the expert motions, robots might implicitly learn task constraints. When combined with exploration-based methods and exhaustive simulation, adaptive behavior for unstructured environments can be achieved. Furthermore, in cases where imitator and expert have dissimilar bodies, a kinetic mapping between motions becomes necessary, commonly referred to as motion re-targeting. A similarity metric can be used to evaluate the quality of re-targeting, which can be performed either in joint or Cartesian spaces.

As we face the increasing presence of robots within the human workspace, a concern that naturally emerges is the guarantee of safety for human individuals. As stated in [50], it is necessary to address physical, psychological, and social influence over people who get exposed to robots’ perception, cognition, and action. It is also shown that there was a boom in color-depth sensory devices development starting from 2010, causing a noticeable interest in perception-enabled safety systems. As input data, these systems would use accurate real-time depth information from the observed workspace, which may include the environment, humans, and other robots. Color-depth sensors are also known as RGB-D (red, green and blue color channels along with a depth channel). When compared to ordinary RGB cameras, the main advantage of RGB-D sensors is a higher invariance to color and lighting conditions. One direct implication is to use depth data for human skeleton cognition and pose inference with Deep Neural Networks (DNNs). Vision-based human pose detection allows for high-level interactions, such as controlling the position of a robot based on human body movement. The main advantage resides in direct motion control, exempting the user from additional hardware components such as complex joysticks or wearable devices. Concerning teleoperation, using vision-based control naturally brings into light the re-targeting problem: how to effectively map human motion to robot movement.

In the discussed scenario of HRI, we aim to address the problem of finding a good vision-based position control through a natural user interface that exploits human motion for teleoperating a robotic arm, using only non-intrusive RGB-D data from the user as sensory input, with no additional controllers or wearable devices. As a study case, we implement a complete position control system for a robotic arm and its end-effector using Robot Operating System (ROS) as a communication backend.

The present work directly contributes to those of [23] as an improvement iteration. Here, our main contributions are the following:

1. A human hand state classifier based on Long-term Recurrent Convolutional Networks (LRCNs) that exploits the temporal coherence of the problem, thus allowing a reliable robot gripper state control without the need for temporal filtering;
2. A novel pose mapping approach based on Thin-Plate Splines (TPS) to transfer human body poses to the robotic arm poses. We highlight that the proposed pose mapping can be individually trained to fit each particular user body, addressing the problem of users’ kinematic limitations and comfort concept. This is the main contribution for the present paper.

As corroborated in our experiments, the combination of these parts provides a natural user interface that showed to be an intuitive and responsive alternative for teleoperation, with enhanced precision in critical workspace regions, such as their boundaries.

The organization of the article is as follows. Section 2 discusses about related works within the robotics telemanipulation field. Section 3 gives an overview about the present paper and its goals. Section 4 presents a detailed discussion about the proposed non-linear position mapping between user and robot. Section 5 discusses about the classifier used to infer user’s hand state through Recurrent Neural Networks. Section 6 addresses specifics about the system architecture used to serve as testing environment to this paper. Section 7 discusses about the participants experiment, including definitions and obtained results. At last, Section 8 presents the finishing statements about the present work.

2 Related Work

Natural human-robot control approaches rely on the combination of human pose and gesture recognition methods, and human motion transfer to robots. In the following, we describe previous work related to our method, and also briefly describe natural human-robot control approaches that resemble ours.

2.1 Vision-Based Human Body Pose Recognition

Vision-based human pose estimation research, using rgb images, already resulted in a vast literature [30, 37] before the popularization of depth sensors in the 2010s. Until then, results were not feasible for real-world applications.
The advent of the Kinect sensor and its original skeleton tracking algorithm [43] allowed the development of novel natural user interfaces based on human body poses and gestures [29]. By exploiting geometric information and huge annotated datasets of depth data, real-time human pose recognition finally became a reality. Consequently, the literature in gesture recognition with depth sensors grew quickly. For a comprehensive presentation, we refer the reader to [11, 19, 26, 27].

Nevertheless, depth data-based methods relied on depth sensors, which are still not omnipresent as rgb sensors that are currently available in most mobile devices. This limitation has been overcome by novel deep learning algorithms capable of estimating human poses from rgb data in real-time. To the best of our knowledge, OpenPose [5] has represented the first real-time, multi-person, and deep learning-based system to jointly detect human body, hand, facial, and foot keypoints using monocular rgb images. However, the original algorithm was limited to 2d human poses (skeletons), and thus unsuitable for 3d natural user interfaces. 3d human pose estimation from rgb images is more challenging since it needs to predict the depth information of body joints [6]. Since such methods are still in its infancy, we opt for the popular and accurate depth-based Kinect v2 tracking algorithm, since it achieves state-of-the-art accuracy for 3d human pose recognition [51]. In addition, natural human-robot control applications are expected to be performed in controlled environments, where it is feasible to employ a depth sensor.

### 2.2 Vision-Based Hand Gesture Estimation

Vision-based human hand gesture estimation is also an extensively studied problem in the last few decades due to its potential in many distinct applications, which resulted in many literature reviews and surveys [35, 38, 46]. According to Oudah et al. [35], hand gesture estimation methods may exploit color, appearance, motion, skeleton, depth, 3d models, or be based on Deep Learning algorithms. Similarly to body pose recognition approaches, these vision-based methods are mostly based on rgb or depth images, possibly requiring the use of markers, such as gloves. In what follows, we focus on some Deep Learning approaches, since they are more closely related to our work.

To the best of our knowledge, deep neural networks have been proposed for hand gesture recognition since 2014, when Neverova et al. [33] employed 2D Convolutional Neural Networks to recognize upper-body and hand gestures from RGBD data. They proposed a multi-modal and multi-scale framework for gesture detection, localization and recognition. Next, Molchanov et al. [31] proposed an algorithm for drivers’ hand gesture recognition from depth and intensity data using 3D convolutional neural networks. Their solution combines information from multiple spatial scales for the final prediction of dynamic hand gestures. Differently, our approach requires the recognition of static hand gestures (i.e., hand poses).

Oyedotun and Khashman et al. [36] applied two different deep learning approaches to the task of recognizing 24 American Sign Language (ASL) static hand gestures: 2D convolutional neural networks and stacked denoising autoencoder. As a limitation, their input is a single grayscale image, which consequently does not exploit the spatiotemporal coherence of the problem. Static hand gestures recognition from grayscale single images were also investigated by Peijun et al. [2], but using a deeper convolutional neural network comprised of 9 convolutional layers. Many CNN-based static hand gesture recognition approaches followed similar approaches, including [1, 16, 21]. Recently, object detection approaches have been proposed for static hand gesture recognition, using state-of-the-art deep neural network architectures such as Yolo [32]. None of these works considered more than one image (i.e., a small video of the last frames) to recognize static hand gestures, thus ignoring the spatiotemporal coherence of the problem.

Specifically for control, Lima et al. [23] employed a Convolutional Neural Network that recognizes whether the user hand is opened or closed using depth data. We improve that approach by using small sequences comprised of the last images acquired by the sensor, thus exploiting the spatiotemporal coherence of the problem through LRCNs. Nuzzi et al. [34] presented a Faster R-CNN Object Detector to find the accurate position of the hands in RGB images, with the aim to perform real-time human-robot interaction. Although their application was similar to ours (controlling a robotic arm), they were based on single RGB images. Differently, our hand gesture classifier has two important differences: it uses depth images, and consider not one, but a sequence of several images to perform the classification of the current gesture. Hu and Wang [15] proposed a dynamic hand gesture recognition system based on skeleton data given by a Leap Motion sensor. A few deep neural networks were experimented, with the aim to control unmanned aerial vehicles (UAV). Although providing accurate skeletons, the Leap Motion sensor requires the user’s hands to be at most 60cm from the sensor. Consequently, it is unfeasible to simultaneously capture full body gestures. A Leap Motion sensor was also used by Liu et al. [25] to recognize hand motion in a multimodal control interface for human-robot collaboration that also included body posture and voice recognition. However, only a vanilla Multilayer Perceptron was experimented for hand motion recognition.
2.3 Natural Human-Robot Control

Real-time robotic imitation of human motion is a relevant problem to allow teleoperation and ensuring safe physical human-robot interaction. More importantly, this is a crucial step towards automating complex manipulation tasks in human-shared environments. Due to its importance, many techniques have already been proposed. Specifically, such techniques try to achieve a general solution to the correspondence problem, namely a mapping between the observed human posture and the robot one. This a challenging problem, since the workspace of humans and robots generally have different geometries [23].

The problem of kinematic re-targeting was discussed in [7], while proposing a virtual-joint based similarity criteria for evaluation mapping performance. Two platforms were implemented to experimentally verify the flexibility and the effectiveness of the proposed metric, both using the Matlab framework. Specifically, they used the Robotic Toolbox for model visualization and the Optimization Toolbox for kinematics mapping. Two robots with 6 DOF were used.

Similar to our project, a transformation mapping from the user’s left arm to the robot was the subject of the experiments in three steps: user pose capture, kinematics transformation and, lastly, transfer to the real robot. Differently from them, our work used a metric-less non-linear mapping by directly specifying points from each Cartesian workspace (demonstrator and imitator). They also did not take into consideration the hand state mapping for the robot gripper.

The problem of kinematic re-targeting was discussed in [7], while proposing a virtual-joint based similarity criteria for evaluation mapping performance. Two platforms were implemented to experimentally verify the flexibility and the effectiveness of the proposed metric, both using the Matlab framework. Specifically, they used the Robotic Toolbox for model visualization and the Optimization Toolbox for kinematics mapping. Two robots with 6 DOF were used.

Similarly to our approach, a transformation mapping from the user’s left arm to the robot was the subject of the experiments in three steps: user pose capture, kinematics transformation and, lastly, transfer to the real robot. Differently from them, our work used a metric-less non-linear mapping by directly specifying points from each Cartesian workspace (demonstrator and imitator). They also did not take into consideration the hand state mapping for the robot gripper.

In [14], this problem is addressed by using electromyogram (EMG) signals recorded from the muscles of the elbow. The proposed applications were assisting elderly people, supporting people with physical disabilities, or tele-operation in hazardous sites. They applied the methodology in a 5 degrees of freedom (DOF) anthropomorphic robotic manipulator, by mimicking the exact joint angles from the user’s arm. To accomplish it, they used two accelerometer sensors coupled to the user’s arm alongside the elbow EMG readings as input data. Different from us, they addressed the problem by replicating the full user’s arm joint values in a low-cost robotic arm system, besides using EMG and acceleration sensors. In our case, we simplified the problem to address only the gripper position mapping, using a fixed down-side orientation which is proper for most pick-and-place tasks.

In [20], the problem is addressed from a teleoperated robotic nurse point of view. Their focus was on human-robot collaboration in medical environments, making a study about the acceptability of a teleassistive robotic system among healthcare professionals. Their methodology consisted of testing 3 different control modes for a manipulative goal: haptic control, gesture control, and voice commands. Their gesture control method used a Kinect V1 to capture the user’s upper body skeleton and pose information, alongside two EMG readers (in armband form) to infer the user’s hand state (open or closed). Armbands also provided gyroscope information to detect the roll/pitch motion of the user’s hand to control the robot’s mobile base. They performed the validation experiment with 11 pre-med and bio-engineering students as participants, evaluating two main parameters in the form of a Likert scale survey: (1) difficulty in control and (2) safety. Different from their approach, in our experiments the robot’s base was fixed. Moreover, we did not use any wearable device for the robot manipulator control. Both the user pose and the hand state were inferred directly from the RGB-D data. On the other hand, we also used Likert scales for our participants’ survey, but with different measured aspects (smoothness of movement and difficulty near workspace boundaries).

In [24], the theme of teleoperation is addressed while applied to a manufacturing environment aided by virtual reality (VR) techniques. Using the concept of a Virtual Reality Control Room (VRCR), the user’s state is mapped to the VRCR, and the VRCR is mapped to the robotic system, decoupling the sensory stimuli (which is given from the virtual environment) from the robot communication. They tested the system using the Oculus Rift with Touch controllers, along with a Baxter robot. Baxter provided camera feedback for the user in both the stereo-camera placed at the head of the robot and an end-effector camera for each gripper. They assessed the effect of a high-latency network, through the simulation of a limited system connectivity. Similarly to us, they also tested the system for pick-and-place and assembly tasks, measuring their time of completion. Their sub-tasks (which one could associate to the atomic operations in our work) involved the interaction with two Lego cubes, which were: approaching, grasping,
aligning, and assembling (stacking one on top of another). If the user made an unrecoverable error, mis-aligned the blocks, or took longer than 6 minutes for each sub-task, the attempt was recorded as a failure. Different from us, they assessed the mapping for both human arms, although our system expands naturally for both arms. Also, we did not need any wearable devices like Oculus Rift or Touch Sensors, relying only on the Depth Sensor and visual feedback (i.e.: a common RGB camera).

Finally, Lima et al. [23] proposed to exploit skeletons extracted from a depth sensor to map the user’s hand position (with respect to the shoulder frame) to the robot gripper position (with respect to the base frame), providing a mapping that is invariant to user location and orientation in the camera space. However, the proposed mapping between a human arm pose space to a robotic arm pose space was a naive linear mapping, which is not efficient since both demonstrator and imitator workspaces have different volumes.

Figure 1 illustrates such volume differences. Demonstrator taskspace volume is a semi-sphere while the robot taskspace volume is an approximated semi-cylinder. [23] also developed a hand-state classifier using depth images and Deep Neural Networks. As a depth sensor, they used a Kinect V1, along with a robotic arm (Denso VP6264) and a gripper (Robotiq 2F-85).

Differently from that previous work, we propose a novel hand-state classifier based on LRCNs, and a novel pose mapping approach based on a TPS formulation that learns from each user, through a training session, a customized non-linear pose map to appropriately transfer its body movements to the robotic arm movements. For a more comprehensive presentation on deep learning methods in robotics, we refer the reader to [18].

3 Overview

In this work, we focus on the problem of developing natural user interfaces to control a robotic arm using a depth sensor such as Kinect V2. We face the challenge of developing an accurate and intuitive interface to control the position and the state of a robotic arm manipulator through human motion, without using any wearable device such as haptic gloves.

We take advantage of the similarity between the human superior limbs and the robotic arms, and propose to map the human hand position w.r.t. the shoulder to the robotic gripper position w.r.t its base; and the human hand state to the robot gripper state. With this goal, in addition to the robotic arm, the proposed method relies only on a Kinect V2 sensor, which could eventually be replaced by a similar depth sensor, and an average desktop computer.

The method is comprised of two main parts: a novel customizable motion mapping function based on Thin Plate Splines [3], which relies on a personalized user training session to fit each individual user body characteristics; and a recurrent neural network trained to recognize whether the user hand is opened or closed from depth
images. Figure 2 shows an overview of the method, which we detail in the following sections.

4 Natural Human-Robot Interface

The development of user interfaces for robotic arm teleoperation is challenging due to the high dimension of the robot pose space and the non-intuitive output of varying each joint angle. In this work we propose a novel method based on Thin Plate Splines [3] to smoothly and intuitively map human body poses to the robotic arm pose. We also rely on depth sensor skeletons to estimate human body poses but using a Kinect V2 in our experiments. The proposed mapping is customized for each user in an individual training session, to capture its specific anatomic limitations.

Following, we first describe the invariant human arm representation which was previously presented in [23]. Then, we briefly describe the mapping model formulation and how it is employed to map the human arm pose to the robotic arm pose. Finally, we detail a straightforward training process in which a user may customize the mapping to fit its body characteristics.

4.1 Human Arm Representation

To provide invariance to translations and rotations of the body w.r.t. to the depth sensor, we employ the human arm pose representation presented in [23]. Note that only the relative position of the hand w.r.t. the corresponding shoulder is exploited to control the robot.

First, we define a reference frame centered on the active shoulder, which we consider to be the right shoulder from now on. The following skeleton joints are used: right shoulder (\(j_{rs}\)); right elbow (\(j_{re}\)); right hand (\(j_{rh}\)); left shoulder (\(j_{ls}\)); shoulder center (\(j_{sc}\)); and spine center (\(j_{sp}\)). Used joints and vectors are illustrated in Fig. 3, and discussed at following paragraphs.

The basis of the reference frame is given by three vectors related to the torso plane. Two of them represent the torso
vertical and horizontal directions. More specifically, we compute vertical \( \hat{v} \) and horizontal \( \hat{h} \) unit vectors given by:

\[
\hat{v} = \frac{\mathbf{sp} - \mathbf{sc}}{\| \mathbf{sp} - \mathbf{sc} \|}, \quad \hat{h} = \frac{\mathbf{sr} - \mathbf{sc}}{\| \mathbf{sr} - \mathbf{sc} \|}.
\]

(1)

The last vector of the basis is the normal vector \( \hat{n} \) of the torso plane, which can be easily computed by the normalized cross product:

\[
\hat{n} = \frac{\hat{h} \times \hat{v}}{\| \hat{h} \times \hat{v} \|}.
\]

(2)

A right arm vector \( \mathbf{u} \) is first computed within camera coordinate system, where:

\[
\mathbf{u} = \mathbf{jr} - \mathbf{js}.
\]

(3)

Finally, the aforementioned invariance is achieved through a change of basis of \( \mathbf{u} \) from the camera basis \( \{ \hat{e}_x, \hat{e}_y, \hat{e}_z \} \) to the reference frame basis \( \{ \hat{h}, \hat{v}, \hat{n} \} \) (see Fig. 3):

\[
\mathbf{ur} = \mathbf{u} \cdot \left[ \hat{h}, \hat{v}, \hat{n} \right].
\]

(4)

### 4.2 Human-Robot Pose Mapping

In this subsection, we briefly describe the naive transformation proposed in [23] to map the human arm pose to the robotic arm pose and discuss its limitations. Then, we present the novel position mapping approach.

#### 4.2.1 Naive Affine Pose Mapping

The Naive Affine Pose Mapping (NA-PM) is our base approach. In this straightforward approach, the input is given by a normalized arm vector (\( \mathbf{ur} \)), due to differences in users’ arm lengths. To perform this normalization, the sum of the lengths of the arm and forearm vectors is employed as follows (Fig. 3):

\[
\mathbf{ur} = \frac{\mathbf{ur}}{\| \mathbf{ur} \|} = \left[ u'_x, u'_y, u'_z \right].
\]

(5)

Then, the following affine transformation is applied to mimic the human arm pose to the robotic arm pose:

\[
P_G = \begin{bmatrix}
\eta \omega_x u'_x + \delta_x \\
\eta \omega_y u'_y + \delta_y \\
\omega_y u'_y + \delta_z
\end{bmatrix} \cdot \begin{bmatrix}
\hat{X}_0 \\
\hat{Y}_0 \\
\hat{Z}_0
\end{bmatrix}
\]

(6)

where:

- \( (\omega_x, \omega_y, \omega_z) \) is a constant mapping proportion vector;
- \( (\delta_x, \delta_y, \delta_z) \) is a constant mapping displacement offset;
- \( (\hat{X}_0, \hat{Y}_0, \hat{Z}_0) \) is the orthonormal basis of the robot coordinate system.
- \( \eta \in \{-1, 1\} \) is a constant that represents a mirroring mode, allowing for mirroring horizontal and normal axes of user’s body plane;
- \( P_G \) is the position of the gripper w.r.t. the robot coordinate system, as shown in Fig. 3.

It is worth mentioning that we consider that the gripper orientation always stays pointing downwards since this is the default orientation for simple pick-and-place tasks.

Although fairly simple, this transformation requires a manual tuning of the aforementioned constants, which may vary depending on the user’s preferences and robot motion boundaries. Besides, even after a careful calibration procedure, one of the following cases might stand: (1) the resulting calibration is restrictive and robot movement becomes limited to prevent mapping the robot gripper position outside its valid workspace; or (2) the calibration is more flexible, leading to robot gripper positions that must be projected to the workspace boundary. Both scenarios occur due to the inherently different geometries of the pose spaces (semi-sphere vs. semi-cylinder).

Also, different users might have substantially different body limitations concerning arm movement (body limits that may not be fixed by the arm length normalization discussed in Section 4). For instance, a specific user might not be able to reach his left side with his right arm as far as the others. Such limitations motivated the use of (1) a more flexible non-linear family of mapping functions; and (2) a customized training procedure to fit the mapping function to each specific user body.

#### 4.2.2 Thin Plate Splines Pose Mapping

Thin-Plate Splines (TPS) [3], are a class of spline mapping functions that have been applied to distinct problems such as interpolation [17], non-rigid image [42] and surface registration [4].

Such mapping functions are easily computable, globally smooth, separable into affine and non-affine components, and provide the least possible non-affine warping component in the presence of mapping constraints. Given two corresponding point sets \( X = \{ x_1, ..., x_m \} \) and \( Y = \{ y_1, ..., y_m \} \), Duchon [12] proved that there is a unique function \( f: \mathbb{R}^n \mapsto \mathbb{R} \) that minimizes the bending energy given by

\[
J = \int \left( \sum_{i,j} f_{x_j} \right)^2 \; dx_1 \cdots dx_n,
\]

(7)

constrained to \( f(x_i) = y_i^k \), for \( i = 1 \ldots m \), where \( y_i^k \) is the \( k \)-th coordinate of point \( y_i \).

This unique function takes the form \( x D + K(x) w \), where \( x \) is a row vector containing the homogeneous coordinates of a point of \( X \), \( D \) is an affine transformation; \( K(x) \) is an \( m \)-dimensional row vector in which \( K_i(x) = U_i(x-x_i) \) is a Green’s function; and \( w \) is an \( m \)-dimensional column vector.
of non-affine warping parameters constrained to $M^t w = 0$. Here, $M$ is an $m \times n$ matrix where each row contains the coordinates of a point of $X$.

Note that, given the corresponding point sets $X$ and $Y$, one has to find the TPS parameters $w$ and $D$. Fortunately, such a solution is given in closed-form as the solution of the linear system:

$$\begin{pmatrix} w \\ D \end{pmatrix} = \begin{pmatrix} K & M^t \\ M & 0 \end{pmatrix}^{-1} \begin{pmatrix} Y \\ 0 \end{pmatrix},$$

(8)

where $K_{ij} = U(|x_i - x_j|)$.

In the following, we describe how to train a TPS to map a user arm pose to the robotic arm pose, in a customized manner, to provide a smooth and adaptable Human-Robot Interface. This approach is called TPS Position Mapping (TPS-PM).

4.2.3 Customized TPS Training

As discussed in Section 4.2, the NA-PM proposed in [23] is unsuitable to map a semi-sphere to a semi-cylinder, besides having the necessity of fine-tuning parameters depending on user-specific body constraints.

We propose to compute a non-linear TPS-PM function based on constraints given by key pose correspondences between the user and the robot. Our hypothesis is that, by constraining a few relevant key poses on the boundaries of the human and the robot working spaces, the other poses would be smoothly and intuitively mapped by the interpolation capabilities of the TPS. More specifically, we propose a custom training of corresponding key poses for each specific user.

We empirically defined a set of 16 robotic arm poses that should be mapped to their respective human arm poses. To avoid extrapolation, such key poses were chosen in the boundary of the robotic arm workspace, at two different height levels (8 per level). Figure 4 reveals the 16 proposed positions for the robotic arm, which are represented in the robot base coordinate system.

To assist the users in the training process, an interactive training tool is proposed. For each robotic arm key pose, an image is first rendered from a 3D human avatar performing the suggested corresponding human key poses, as illustrated in Fig. 5. Then, the images are sequentially shown to guide the user in performing the pose. The user pose is collected using the depth sensor, mirrored (so that users think of the avatar as a mirror) and converted to the proposed arm representation (shown in Eq. 4). Since Kinect skeletons are prone to noise, a sequence of skeletons are collected over a specific period of time, and then the resulting skeleton is computed as the median skeleton, by taking the median value for each skeleton coordinate. It is worth mentioning that, differently from the naive affine transformation method, here the arm representation is not required to be normalized, since the TPS mapping is trained for each specific user body.

Finally, after the training process, the sets representing the robotic arm key poses ($Y$) and the collected user poses ($X$) are obtained to compute the TPS mapping, according to Eq. 8. In Section 7.2 we show that the resulting map is capable of performing smoothly while converting user arm poses to robotic arm poses.

5 Hand-state Classification

The second main part of our methodology aims to provide a natural way to control the robot gripper state using the user’s hand, in a binary fashion (open or closed).

We focus on recognizing the user’s hand state using the depth image. The advantage of this approach lies in its resilience against variations in color, texture, and lighting conditions. It is worth mentioning that, although hand skeletons can be estimated by Software Development Kits (SDKs) such as the original Kinect V2 SDK, they are...

---

Fig. 4 Set of 16 robot poses used to train the TPS-PM model. Poses were organized into two different heights. Points were symmetrically chosen, at the boundary of the robot workspace.
still very inaccurate due to the complexity of the hand skeleton and occlusion. Thus, since we aim to solve a binary classification problem, developing a binary classifier that exploits depth images was considered to be the most suitable approach, as already shown in [23].

In that previous work, a Convolutional Neural Network (CNN) was proposed to evaluate the user hand using a single depth image and map the corresponding state to the robot gripper in real-time. The main idea behind CNNs is that they learn to extract relevant features from input data, generating a proper feature map that allows the classification task to be optimally performed. CNNs first perform convolutions on input data through several convolutional layers, possibly using max-pooling and dropout layers, and then flattening the resulting feature map into a single-dimension array. The flattened array is then classified using dense layers.

Differently, in this work we propose to exploit a sequence of depth images collected from a short period, thus taking advantage of the temporal coherence of the problem. With this aim, we adopt a deep neural network architecture known as Long-term Recurrent Convolutional Network (LRCN, [10]). In what follows, we first describe the depth image preprocessing, and then present the adopted LRCN architecture, highlighting its differences from the CNN presented in [23].

5.1 Depth Image Preprocessing

Depth images acquired by a depth sensor are first cropped, according to the hand position in the image space; filtered to remove background objects; and encoded in grayscale or binary image. This process is performed both in dataset acquisition (for classifier training) and during experiment tests.

Fig. 5 All 16 poses used as reference for the TPS model training. For each avatar pose in the collage, a corresponding real user pose is shown below. Both avatar arm and user arm are mirrored. A set of 16 spheres with all avatar hand positions was used. The green sphere means the current hand position while the blue spheres represent all the previous and next ones.

The following data, which is provided by the Kinect V2 SDK in real-time, is exploited in this step:

- raw depth image $d : U \subset \mathbb{R}^2 \mapsto \mathbb{R}$;
- 3D skeleton composed of 3D body joint positions in Euclidian coordinates, where $j_i \in \mathbb{R}^3$ is the position of the $i$-th joint, in the Kinect camera space.
- 2D skeleton, computed by projecting the 3D skeleton joint positions onto the image plane, where $p_i \in U$ denotes the 2D coordinates, in pixels, of the $i$-th joint.

For the hand segmentation phase, we used the same approach as described in [23]. Let $rh$ be the joint index of the user’s right hand. The raw depth image $d$ is first cropped by defining a 2D squared hand region $\mathcal{H}$ centered at $p_{rh}$, with a side of $B$ pixels. Note that, as the hand moves closer or further away, $\mathcal{H}$ must scale accordingly to keep the projected hand on a constant scale. More specifically, $B$ must vary inversely with the distance from the right hand to the sensor, which is given by $d(p_{rh})$. Thus, we set the side of $\mathcal{H}$ according to the function $b : \mathbb{R} \mapsto \mathbb{R}$, which is given by

$$b(d(p_{rh})) = \frac{K}{d(p_{rh})},$$

where $K$ is an empirically tuned proportionality constant that we set to 60.

The raw depth image is cropped to the hand region $\mathcal{H}$, with side given by $b(d(p_{rh}))$, resulting in the cropped hand image $d' : \mathcal{R} \mapsto \mathbb{R}$. However, such an image may include unwanted data like other parts of the user body or background objects. To address this issue, two filtering approaches might be used: segmentation by depth thresholding; and segmentation followed by binarization.
Fig. 6 Process of image acquisition and pre-processing. First, a squared region of size $K/d(p_{bh})$ is extracted from original image $d$, generating the cropped region $d'$. Then, a filtering step is performed in order to remove the background that is further from the threshold distance $D_{\text{min}} + T$, followed by binarization that generates the binary image $b$. For the LRCN input, a sliding window of size 15 is used over the binary images $b$.

Depth thresholding is performed by simply computing a segmented image $g$ as follows:

$$g(x, y) = \begin{cases} d'(x, y), & d'(x, y) \leq D_{\text{min}} + T \\ 0, & d'(x, y) > D_{\text{min}} + T \end{cases},$$

(10)

where $T$ is a threshold whose value was empirically set to 0.2 meters; and $D_{\text{min}} = \min d'(x, y)$ is the closest point from the camera and, therefore, it is assumed to be a point of the hand. Thus, every point that is further than $D_{\text{min}} + T$ should be removed.

Optionally, a binarization procedure may be applied to generate a binary image $b$ according to:

$$b(x, y) = \begin{cases} 1, & d'(x, y) \leq D_{\text{min}} + T \\ 0, & d'(x, y) > D_{\text{min}} + T \end{cases}.$$  

(11)

In our work, we used the binarization approach. After generating $b(x, y)$, each image is resampled to 50 × 50 pixels, which is the input format of the classifiers that we present and compare in this work.

A summary of the discussed pre-processing procedure is available in Fig. 6.

5.2 Deep Neural Network Architectures

In this subsection, we present the LRCN architecture that we adopt in the proposed methodology, and briefly describe the CNN presented in [23] which we use as a baseline to compare to the LRCN. Examples of each architecture are shown in Fig. 7.

It is worth mentioning that the CNN input is a single image, while the LRCN input is a sequence of many continuous images in time. More specifically, the LRCN classifier aims to predict the hand state of the last image of the input sequence.

**Convolutional Neural Network** In this approach, the pre-processed images described in Section 5.1 are given as input to a CNN, which is comprised of two parts: convolutional layers, to extract relevant features; and dense layers, to perform the binary classification. In the following, we describe hyperparameters that we experimented with in order to fairly compare CNNs to the proposed LRCN architecture. The input data is filtered by 1 or 2 convolutional

![CNN: 1x50x50](image1)

![LRCN: 15x150x50](image2)

Fig. 7 Best CNN and LRCN classifiers found in the experiments described in Section 7.1. Images were generated with aid of Netron open-source software [41]
layers, where each layer may apply filters with size \((F)\) \(3 \times 3\) or \(5 \times 5\). For each convolutional layer, \(8, 16, 32, 64,\) and \(128\) filters per layer \((S)\) were examined. After each convolutional layer, a max-pooling layer with \(2 \times 2\) receptive fields may be applied to downsample the feature map. The resulting map is then flattened and follows to 1 or dense layers, with \(80, 120,\) or \(160\) hidden units each \((D)\). Dropout probability \((P)\) defines both probabilities of dropout in convolutional and dense layers, in which case we tested the combinations \([0.5, 0], [0.5, 0.5]\) and \([0, 0.5]\). In all convolutional and dense layers, we use the rectified linear \((ReLU)\) as the activation function.

**Long-term Recurrent Convolutional Network** LRCNs may potentially exploit the temporal coherence in videos (or sequence of images). More specifically, since the hand state of the user continuously varies with time, we hypothesize that a classifier that exploits such coherence using a sequence of continuous images \((LRCN)\), may achieve higher accuracy than a single-image classifier \((CNN)\). Different from the CNNs, the flattened feature map extracted from the convolutional layers are given as input to Long Short-Term Memory \((LSTM)\) cells instead of dense units. It is worth mentioning that, in order to train LRCN classifiers in a supervised manner, training set must comprise one binary label per image. Thus, after defining a fixed sequence length with value \(n\), sequences of \(n\) continuous images in time are compiled from the original training set, through a sliding window approach. Note that each sequence is associated with a single label, which must be the state of the user at the last image of the sequence. To perform a fair comparison, the convolutional layers hyperparameters were experimented as in the previous CNN architecture. In the recurrent part of these networks, a single LSTM-layer was experimented with a varying number of LSTM units: \(50, 100, 250\) and \(500\) units. Dropout probability \((P)\) was also experimented as in the CNN architecture. In all convolutional layers, we adopt the ReLU as activation function and the hyperbolic tangent in all LSTM layers. Finally, a softmax dense layer with 2 neurons outputs the final probability distribution.

A grid-search was performed to evaluate variations of both CNN and LRCN architectures, to both tune and compare whether LRCN outperforms the CNN in the hand state recognition task. Results are revealed in Section 7.

### 6 System Architecture

This section describes implementation details of the proposed methodology. All modules were developed on a Windows 10, Intel Core i5, 16GB RAM DDR3, and NVidia GeForce GTX 2080 TI 12GB computer with ROS [44] in Melodic version. We used a robotic manipulator of model Denso VP6242 [9], along with a Robotiq 2F-85 [40] gripper in our experiments.

The whole system was developed using two computers. One of them was responsible solely for direct communication with gripper and arm controllers through Matlab 2015. The other was responsible for running user data capture, classification and simulation modules.

To collect our dataset, we used a Microsoft Kinect V2 [28] depth camera, including provided SDK. The whole application was programmed in Python 3 language, except for the bridge between ROS and Kinect V2 SDK, which was programmed in C++. Our classification models were developed using TensorFlow GPU and Keras.

Modules from our system communicate in a distributed manner using the publisher/subscriber architecture provided by ROS, with TCP (Transmission Control Protocol) as the underlying communication protocol. In our study case, Windows was chosen because Kinect V2 SDK is unavailable for Linux. There are, however, viable alternatives such as Libfreenect2 from OpenKinect community [49] and OpenNI [48] libraries for Kinect data manipulation.

How the system was developed allowed for decoupling system components. Classification modules are the same for both visual simulation and real-life trials. The complete application comprises the following modules:

- Kinect V2 and ROS interface;
- Database acquisition;
- Hand-state classification model;
- Hand-position transformation model;
- Robot visual simulation;
- Real robot control.

Hand-state classification and hand-position transformation models can be set in run-time, via ROS parameters. Simulation and robot control nodes can be turned on and off without interfering in classification, since both nodes behave as read-only consumers from ROS topics. A detailed flow diagram of the system is shown in Fig. 2. Data is passed from left to right, except when persisting input data is desired. The complete system is capable of running in real-time, at 30 frames per second.

It is important to mention some safety procedures considered in our architecture. Within the robot visual simulation, collision between robot links and the table is taken into consideration by the inverse kinematics algorithm. Joint limits are established in both the simulating and controlling environments. If, for some reason, the connection between simulation and controlling environment is interrupted, the last known joint positions are maintained by the controller.

There are also hardware and software halting mechanisms for the robot. The hardware mechanism is a red
physical button which halts the robot when pressed, stopping the software communication completely. The software mechanism is a switch in the controlling program which toggles between simulation received joint states and home-position (arm up, 90 degrees elbow). Another safety procedure measure is that, during the robot operation, no user is allowed to transit close to the robot workspace. One last procedure that is worthy to mention is a dead-zone applied to the output of the neural binary classifier: the gripper state only toggles if a minimum or maximum state threshold is achieved, removing undesired fluctuations over the gripper state.

7 Experiments

In this section, we describe the experiments performed to both validate and evaluate the proposed methodology and discuss the results achieved. We first describe experiments to compare the novel LRCN hand state classifier and the previously presented CNN (Section 7.1). Then, we describe the setup of user experiments performed to evaluate and compare the proposed pose mapping approach in Section 7.2. Finally, results of the latter experiments are revealed and discussed in Section 7.3.

7.1 Hand State Classifier

We created a novel database from scratch which is comprised of binary images in .pgm format, in order to train the classifiers discussed in the present paper. This database consists of 9668 labeled images from 7 different people and organized them into 14 sequential episodes, publicly available. 1 As the same database would serve both for the CNN and the LRCN methods, it was organized in an episodic way. For each person, two episodes were continuously recorded at 30 FPS while the user performed the grab-and-release task (imagining an object in the air) from one side to another. The difference between each of the two episodes per person was the starting side for the grab-and-release movement.

After collecting the database, the next step was to train the classifiers. We adopted a 2-fold cross-validation setting to split our available data, in which one fold was used for training and the other was used for testing. A grid search was conducted over the hyperparameters previously described for both the CNNs and the LRCNs (Section 5.2). In this section, we present the best configurations found for both variants.

CNN performed better when using dropout probability \( P \) of 0.5 in both convolutional and hidden layers. Table 1 reveal the results for the experimented CNNs. The best CNN architecture showed to be for convolutional kernel sizes \( F = [[5, 5], [5, 5]] \), with \( S = 32 \) filters per layer, and two hidden layers of dimensions \( D = [120, 80] \). It is also worth mentioning the similar accuracy achieved by the top-performing architectures, suggesting low sensitiveness to hyperparameter variations over the best classified ones.

A similar grid search was performed to find the best LRCNs classifiers. Results are shown in Table 2. Dropout showed a weaker influence on LRCN accuracy. Two specific configurations achieved the same top accuracy value of 98.72% and thus outperforming the best CNNs. This is one of the most relevant results of our experiments, suggesting that, by exploiting a sequence of many images, LRCNs are capable of coping with challenging examples.

To better investigate the effects of the sequence length, we experimented with the top-performing LRCN architecture on three distinct sequence lengths. The results are shown in Table 3, revealing higher accuracy for longer sequences. Thus, we conclude that accuracy is improved when more data is available to be evaluated by the LRCNs.

---

1https://drive.google.com/file/d/1bOQfOFtDxgGfp_b9HELAK3WNchBUxGdX
Table 2 Results for the top 10 LRCN-based networks in our grid-search, sorted by accuracy

| F           | S   | D   | P            | Weights | Accuracy | Loss |
|-------------|-----|-----|--------------|---------|----------|------|
| [[3, 3], [3, 3]] | 16  | 250 | [0.5, 0.5]   | 2,189,982 | 98.7225  | 0.0494|
| [[3, 3], [5, 5]] | 128 | 50  | [0, 0]       | 2,981,310 | 98.7225  | 0.0413|
| [[5, 5], [3, 3]] | 64  | 250 | [0.5, 0.5]   | 6,690,094 | 98.7119  | 0.0541|
| [[3, 3], [5, 5]] | 32  | 250 | [0.5, 0.5]   | 3,477,454 | 98.7119  | 0.0503|
| [[3, 3], [5, 5]] | 32  | 50  | [0, 0]       | 676,254  | 98.6908  | 0.0438|
| [[5, 5], [5, 5]] | 128 | 250 | [0, 0]       | 11,032,558 | 98.6908 | 0.0428|
| [[5, 5], [3, 3]] | 32  | 100 | [0.5, 0.5]   | 1,330,682 | 98.6908  | 0.0497|
| [[3, 3], [5, 5]] | 128 | 100 | [0, 0]       | 5,571,610 | 98.6908  | 0.0433|
| [[3, 3], [5, 5]] | 16  | 500 | [0.5, 0.5]   | 4,209,578 | 98.6803  | 0.0564|
| [[5, 5], [3, 3]] | 64  | 50  | [0.5, 0.5]   | 1,328,894 | 98.6803  | 0.0461|

Table 3 Configuration and accuracy of the best-performing LRCN architecture and the sequence length

| Sequence length | Accuracy | Loss |
|----------------|----------|------|
| 15             | 98.7225  | 0.0494|
| 10             | 98.3337  | 0.0575|
| 5              | 98.1377  | 0.0699|

7.2 User Experiments Setup

To validate and compare the proposed TPS-PM (Sections 4.2.2 and 4.2.3) to the previously proposed NA-PM (Section 4.2.1), we performed usability tests to measure the user experience with both compared approaches. We set up a test environment and predetermined a set of five tasks in which users were asked to manipulate objects with the robotic arm through both pose mapping approaches.

For each position mapping approach, the following aspects were evaluated:

- **Training.** Were users able to accomplish the training procedure successfully?
- **Goal achievement.** Were users able to accomplish the required tasks?
- **Comfort.** Which approach feels more comfortable and natural?
- **Learning curve.** Does the learning curve allow rapid progress in such a way that users could cope with increasingly difficult tasks?

In our experiment, the visual feedback was represented by the vision of the robot from the participant’s point of view. The feedback video transmission, which is an important characteristic of a teleoperated environment, was not the scope of the present work. The physical configuration of our reduced teleoperation scenario can be seen at Fig. 8. Participants were placed at 1.2 meters in front of the robot, within a safety distance. The main objective is to make the robotic arm interact with objects from the environment, which are placed at the robot table. A Kinect V2 camera was placed at 0.9 meters towards the right side of the robot, pointing directly to the participant. This camera was responsible to gather the depth images which are used for position mapping and hand state classification.

At the beginning of each session, users were asked to perform the training to calibrate their body to each pose mapping approach (NA-PM and TPS-PM). For the NA-PM, the calibration was effortless, since the only requirement from users was to stretch their right arm in order to measure its length. In contrast, TPS-PM calibration was more time-consuming. The main issue was that TPS-PM required a considerable spatial awareness from users to correctly perform the keyposes during the training process. Thus, it was necessary to reject low-quality TPS-PM customizations. More specifically, a supervisor visually examined the sixteen collected hand positions to check whether they had an unusual spatial distribution, with two or more vertices being too close together or in an unexpected relative position. Figure 9 shows an example of a successful and of a failed TPS-PM customization, respectively. When the training failed, the user was asked to repeat the procedure.

The set of experimental tasks required not only motor skills and spatial awareness from the users but also the ability to adapt to the specific transformation being used. Each experiment has some increased difficulty in relation to the previous one. Before the experiments, users were asked to fill out a compliance form allowing them to be recorded during the sections for analysis and scientific purposes.

Interaction elements employed in the experiments were:

- two cubes of length equal to 5 cm;
- a cup with a 10 cm diameter red-marked open side and a height of 12 cm. The closed side of the cup has 7.5 cm of diameter.

The cup can be used both upwards or downwards. When placed upwards, the cup serve as a hopper where objects
Fig. 8 Configuration of simulated teleoperation environment. In (a), the distances between the testing environment elements are shown. User is placed at 1.2 meters in front of the robotic arm. There is also a Kinect V2 placed at 0.9 meters along the right side of the robot. Participant’s goal is to use their right arm in order to control the robotic arm and interact with objects from the robot table. In (b), a photo of the real environment is provided. User’s visual feedback is provided by their vision of the robot at each moment.

Fig. 9 Three-dimensional visualization of a successful (left) and of a rejected (right) TPS-PM customization. The difference is noticeable in the relative positions between consecutive poses. A low-quality calibration will not follow the relative placement from one pose to another, generating an ill formed model.
can be put in. When place downwards, the cup serve as a platform where objects can be put on top of. Thus, we will refer to upwards cup as hopper from now on, while referring to the downwards cup as platform. We also defined 4 areas within the robot workspace, named from A to D.

To quantify each task, we use the concept of atomic movements. One atomic movement is an indivisible continuous displacement of the robot gripper with well-defined starting and ending times. Translations and gripper state variations (opening/closing, or grab/release) are also considered atomic movements. Thus, each task can be performed using a minimum number of atomic movements.

More specifically, tasks were:

1. Take a cube from region B to region A. It’s an initial task to test the user’s ability to successfully grab and move an object to another location. It can be achieved with a minimum of 4 atomic movements: home-to-B, grab, B-to-A, release.
2. Swap cubes between regions B and A, using region C as auxiliary. The purpose of this task is to measure the increase in user’s skills since the first task. The swap task can be done in a minimum of 12 atomic movements, all of them with the same difficulty as the first task, e.g.: home-to-A, grab, A-to-C, release, C-to-B, grab, B-to-A, release, A-to-C, grab, C-to-B, release.
3. Bring cubes from regions A and B to the hoop. This task introduces the necessity of motion at higher heights, thus increasing task complexity in the vertical axis. Users have to take into consideration avoiding collision between the gripper and the hoop while bringing the cubes to the proper height before releasing it. Skills evaluated here also include ease of use near robot boundaries, since the platform is located at region D. It can also be executed with a minimum of 8 atomic movements, e.g: home-to-A, grab, A-to-hoop, release, hoop-to-B, grab, B-to-hoop, release.
4. Pile cubes in region A. This task introduces a new component in evaluation which is the ability to fine-tune the cube position to successfully pile one cube onto another. Previous tasks had a less strict acceptance with relation to where the cubes might be placed. Can also be performed within 8 atomic movements, e.g.: home-to-C, grab, C-to-A, release, A-to-B, grab, B-to-A, release.
5. Move a cube from platform to A, then from B to platform. Skills needed are boundary regions control and the ability to fine-tune the cube position to place it on top of the platform. Can also be performed on a minimum of 8 atomic movements, e.g.: home-to-hoop, grab, hoop-to-A, release, A-to-B, grab, B-to-hoop, release.

Both initial and target configurations for each of the previously discussed experiments are shown in Fig. 10. In order to measure time completion for each experiment, we defined the following conditions: (1) time counter starts as soon as the user-control mode is triggered, and; (2) time counter finishes as soon as the target is reached and the objects of interaction remain still. Success occurs when the target configuration is reached and objects stay within established areas (not touching the workspace region outside the black markers). A fail occurs when the experiment reaches a state where the target configuration is not feasible without human assistance (i.e. when one cube falls outside the robot workspace).

Besides analyzing the completion time for the tasks, an important measure would be the confidence of users over the classifier during the experiments. For instance, one could ask whether the users were able to trust that grabbed objects by the gripper would not be released unexpectedly. In order to quantify this confidence, we decided to manually count the number and the nature of the errors for each trial, by means of watching the users recorded videos after experiments were finished. Thus, two types of errors were defined: (1) classifier infer open when user hand is closed, and; (2) classifier infer closed when user hand is open. Two values were used in our analysis: the average number of errors per trial atomic movements, and; the total number of errors per error category.

The previously discussed set of five experiments was repeated twice for each user, firstly using NA-PM and then using TPS-PM. Before the very first experiment of each pose mapping method started, a “playground time” of two minutes was given to each user for free operation so that they could acquire familiarity with the method. After the very last experiment of each session, users were asked to answer two multiple choices quantitative Likert-based [22] questions by selecting a value in the range from 1 to 5, about how they felt with relation to the pose mapping method:

- (Q1) How smooth was the movement of the robotic arm during control? This question evaluates how comfortable users were with the smoothness of the robotic arm movement when their right hand move.
- (Q2) How difficult was the movement near the workspace boundaries? This question evaluates the quality of the pose mapping when the robot gripper is closer to its boundaries. A good mapping would present the user with a consistent low difficulty that does not depend on the region of operation, with an overall effort being as minimum as possible.

After both sets of experiments were finished and their respective objective questions were answered, the users were asked to answer a set of 3 discursive questions to
Fig. 10  Initial (left) and target (right) configurations of each task performed by the users. Each row represents one task from 1 to 5, respectively.
Table 4  Demographics about experiment participants. All participants were male computer students or professors. Following data was collected: height \( h \), wingspan \( ws \), hand horizontal span \( hhs \), hand vertical span \( hvs \), body mass interval \( bmi \)

| Participant | \( h \) (m) | \( ws \) (m) | \( hhs \) (m) | \( hvs \) (m) | \( bmi \) (kg) |
|-------------|-------------|-------------|-------------|-------------|-------------|
| p1          | 1.88        | 1.94        | 0.255       | 0.21        | [90, 120]   |
| p2          | 1.70        | 1.71        | 0.225       | 0.18        | [50, 90]    |
| p3          | 1.86        | 2.00        | 0.23        | 0.21        | [90, 120]   |
| p4          | 1.60        | 1.55        | 0.205       | 0.18        | [50, 90]    |
| p5          | 1.89        | 1.86        | 0.24        | 0.25        | [90, 120]   |

evaluate overall opinion about the comparison between both mappings:

– (Q3) In your opinion, which were the advantages of each mapping?
– (Q4) What were the challenges found during the experiment?
– (Q5) Would you have any suggestions to improve the interface?

Besides the experience information we also collected demographic information about all invited participants, Since we rely on body visual information as input to the controlling task. This demographic data included:

– \( h \) height in meters of the participant;
– \( ws \) wingspan in meters, which is defined as the distance between each middle finger when arms are wide open horizontally;
– \( hhs \) hand span in meters over the horizontal axis, which is the maximum distance between the thumb and the little finger when the hand is wide open;
– \( hvs \) hand size in meters over the vertical axis, which is the distance between the wrist and the tip of the middle finger when the fingers are straight;

– \( bmi \) body mass interval in kilograms in which participant’s weight falls into.

All participant answers were collected through an online form for posterior analysis. At this phase, five participants volunteered for testing purposes. A two-sample Student’s T-test [45] was performed over collected duration/atomic values for the two classes (NA-PM and TPS-PM).

7.3 User Experiments Results

After successfully performing the customized training procedure, each individual executed each of the five proposed tasks using NA-PM, before executing the same tasks using TPS-PM. Collected demographics about participants are present on Table 4. All participants were male computer students or professors. Table 5 shows the time duration taken by each participant to complete each task. We did not ask participants to perform the tasks as fast as possible, thus allowing them to take the necessary time without a rush. The number of classification errors accounted for each participant is present in Table 6. All proposed tasks were able to be completed within 3 or fewer trials. A sample of each

Table 5  Timing results for each experiment performed by the volunteers. “F” means fail in the current trial, with each specific task being offered a maximum of 3 trials. Times are represented in seconds from the beginning to the completion. The acronym atm stands for atomic movements

| part. time/task | Duration (secs) for NA-PM | Duration (secs) for TPS-PM |
|-----------------|---------------------------|---------------------------|
|                 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| p1              | 13.3 | 29.2 | 29.1 | 23.0 | 38.2 | 11.0 | 43.1 | 25.2 | 21.2 | F, 26.2 |
| p2              | 14.1 | 33.1 | F, 30.0 | 34.1 | 25.1 | 11.2 | 32.1 | 36.0 | 22.1 | 28.2 |
| p3              | 12.2 | 43.1 | F, 36.1 | 23.2 | 32.2 | 19.1 | 76.0 | 51.1 | F, F, 31.0 | F, 35.0 |
| p4              | 20.1 | 47.2 | 28.1 | 34.2 | 28.2 | 17.3 | 54.0 | 29.1 | 59.3 | 35.0 |
| p5              | 08.1 | 21.0 | 15.1 | 13.2 | 20.1 | 09.0 | F, 15.0 | 23.0 | 16.2 | 22.2 |
| mean            | 13.3 | 33.1 | 29.1 | 23.2 | 28.2 | 11.2 | 43.1 | 29.1 | 22.1 | 28.2 |
| std             | 4.3 | 10.6 | 7.7 | 8.8 | 6.9 | 4.4 | 23.0 | 11.3 | 17.2 | 5.6 |
| atm             | 4 | 12 | 8 | 8 | 8 | 4 | 12 | 8 | 8 | 8 |
| mean/atm       | 3.3 | 2.8 | 3.6 | 2.9 | 3.5 | 2.8 | 3.6 | 3.6 | 2.8 | 3.5 |
Table 6  Number of classification errors visually counted by the authors for each experiment performed by the volunteers. A count of zero means no error was detected within trial

| part. errors/task | # of errors for NA-PM | # of errors for TPS-PM | avg/atm      |
|-------------------|-----------------------|------------------------|--------------|
|                   | 1 2 3 4 5             | 1 2 3 4 5              | mean std     |
| p1                | 0 0 2 0 2             | 0 2 1 0 2             | 0.07 0.11    |
| p2                | 1 1 2.0 2 0           | 1 0 1 1 2             | 0.14 0.11    |
| p3                | 0 0 1.1 0 0           | 1 0 0 2.0 0 1.0       | 0.06 0.09    |
| p4                | 3 4 2 2 1            | 3 1 2 2 0            | 0.3 0.25     |
| p5                | 0 0 0 0 0            | 0 1.1 2 2 1          | 0.07 0.10    |

A number different than zero represents the sum of errors from type 1 (wrongly inferring open) and from type 2 (wrongly inferring closed). At the last two columns, we normalize the count variable over the number of atomic movements before applying the mean and standard deviation.

Task may be seen in a public available video. For each task, there is two simultaneous views of the participant. One is a cropped version of the RGB video captured by the Kinect V2. The other is a video recorded with an ordinary webcam.

By analysing the time intervals of Table 5, it’s possible to infer that the change on the control mode did not have a substantial influence over the total duration of each task. To further investigate this fact, we conducted a two-sample T-test over both classes, for the mean/atomics variable, assuming that classes were independent. The T-test resulted in a p-value of $0.899 > 0.05$, which confirms the null hypothesis in which both mapping modes (NA-PM and TPS-PM) have the same expectation for duration/atomics values. Such fact leads to the assumption that both modes have a similar learning curve, even though each one of them presents its own peculiarities. Such fact can be confirmed visually as shown in Fig. 11.

Some of the participants were able to perform the tasks faster, showing a better spatial awareness, as the participant p5, for instance. Others presented a more cautious and curious manner, trying to experiment with the nuances of the robot movement, as the participant p3, for instance. This heterogeneity on the test participants contributed to a substantial diversity in the final duration of the tasks.

Tasks 1, 2, and 4, which did not require operation in high heights, nor interaction with the cup, presented the best duration per atomic operation. Tasks 1 and 2 were able to be successfully fulfilled within one single trial for all participants, except one. Tasks 3 and 5 failures were due to the collision of the robot gripper with the cup, causing a displacement since it was not fixed to the environment, leading to an unrecoverable state of the current task. Failures in Task 4 were due to a specific displacement in the center of the user’s hand position when the hand state was toggled, i.e. when the hand closes and its center moves closer to the user’s wrist. User p3, with his cautious explorer profile, tried to understand this behavior during the experiment, causing him to take longer to complete the task.

As already stated at the start of the present section, the counting number of classification errors during experiments is present in Table 6. Average values are calculated taking into account the number of errors within each trial, and normalizing it over the task’s minimum number of atomic movements. Since classification errors are deeply related to the user’s hand morphology (instead of the task definition itself), mean and standard deviation were calculated for each specific user.

Analyzing the Table 6, it is possible to notice that three of the users had an average error/atomic of less than 0.07. This leads to a sense of safety during the the tasks execution, since participants were confident while manipulating environment objects. Larger values error values were presented by the participants p2 and p4. Crossling this data with the Table 4, the following facts are evidenced:

1. p2 and p4 have smaller $hhs$ and $hvs$ values. Their hand morphology for open and closed hands are slightly less distinct, if comparing from the same distance from the camera;
2. p2 and p4 have smaller $ws$ values, which means their arms can reach shorter distances. As the distance from the camera was measured by a mark on the floor where they should position their feet, users with smaller $ws$ values would have their hands moving in a further region from the camera.
3. Further and smaller hands, in the camera point of view, present less resolution in the segmented depth image.

Putting facts 1, 2 and 3 together, our hypothesis is that the increased error for participants p1 and p4 would have been decreased if:

- the network training data was augmented with random scale for dealing with different hand sizes;

---

2 Video with experiments: https://youtu.be/Rk3iS_KnaWc
Fig. 11 Comparison between durations for both control modes (NA-PM and TPS-PM). Showing both the mean duration and the mean duration per atomic movement values. Graphs does not show substantial alterations in duration values, which leads to believe that novel method does not increase task difficulty – participants feet positioning were customized accordingly, taking into consideration participants wingspan.

The nature of the classification errors for the whole experiment can be seen in Fig. 12. The majority of the errors is classified as type 2, where the model infer that the hand is closed when it is actually open or partially open. This type of error accounted for more than 90% of the total. For all type 2 errors, if the user opens up his hand a little more, the classifier infers correctly again.

Still referring to Fig. 12, almost no classification error of type 1 was witnessed. Another safety discussion can be added here, since the absence of type 1 error makes users confident that, once they grab a certain object, the gripper would not unexpectedly open and release the object.

Participants’ answers for the subjective questions Q1 and Q2 are presented in Fig. 13. From these plots only, it is suggested that participants agreed more with respect to smoothness of the robot arm movement. On the other hand, they presented more variance within answers related
Fig. 13 Comparison between both control modes (NA-PM and TPS-PM). Both questions Q1 and Q2 were answered in a scale of intensity from 1 to 5. Naive Affine mode performance was better with respect to smoothness, according to users. Thin Plate Spline performance with respect to difficulty was better near to boundary limits of robot arm workspace.

to the difficulty of arm control near the boundaries of the workspace, but suggested a tendency for a decrease in difficulty.

In general, users considered the first control mode to be smoother concerning the robotic arm speed, but found the second control mode to be better near the robot workspace boundaries. Both phenomena are explained by the fact that TPS-PM allows users to define their custom mapping in a non-linear fashion. Thus, while this customization provides a more accurate mapping in the boundary of the workspace, its non-linearity also implies variations in the magnitude of the gradient of the mapping function, which harms the sense of smoothness. Also, it was noted that smaller users’ workspace volumes resulted in more comfortable control of the robot since users do not need to completely stretch their arm to reach robot arm boundaries. In other words, users only need to reach the boundaries of their trained workspace model.

As questions Q3 to Q5 were open and subjective, we will discuss the most frequent statements from the participants. Question Q3 was related to the advantages of each mapping. According to the users, the advantages of the NA-PM model were intuitiveness and smoothness. One participant reported that “The advantage of NA-PM is the displacement of the arm matching the displacement of the robot, in other words, the relative movement is almost the same. The advantage of the TPS-PM is the fact of the movement at boundaries did not degenerate”.

Question Q4 was related to the challenges faced during task executions, where answers presented the higher diversity among the three subjective questions. One participant stated difficulties related to the hand-state classifier. We concluded that this difficulty was due to the hand bounding-box size (defined by Eq. 9) not taking into consideration users’ hand size, but only its distance from the camera. Thus, users with excessive small or bigger hands, which were not present in the hand state classifier training dataset, would face a decreased accuracy in the classifier. This issue can be attenuated by training custom bounding box sizes for each user.

Another challenge for the users was to learn both mappings variants in consecutive sessions. This might have lead to confusion and thus less-than-optimal performance.

Finally, question Q5 was related to improvement suggestions. One suggestion was to have a classifier that would work with any hand orientation. In current work, for the hand-state classifier to be accurate, participants were asked to maintain their hands at a specific orientation (faced to the depth camera). Another valid suggestion was to create a more intuitive calibration for the TPS model, with some kind of online visual feedback, leading to an easier calibration process.

Overall, we conclude that, while the NA-PM is efficient and practical for many simple tasks, the novel TPS-PM approach is more appropriated for tasks where high precision near the workspace boundaries is required. Besides, TPS-PM still provides reliable performance for straightforward tasks.

8 Conclusion

In this work we presented a novel approach for kinematic mapping between human and robotic arm poses that combines a user-customizable pose mapping method based on Thin Plate Splines; and Long-term Recurrent Convolutional Networks for hand state recognition. A complete motion-based teleoperation system using only a depth camera was built and validated through user experiments. The system works in a distributed way, where decoupled modules communicate through Robot Operating System framework. Cross-validation experiments showed higher accuracy of the LRCNs when compared to a previously proposed CNN classifier to recognize human pose state. Besides, user experiments revealed that users adapted successfully to the new approach. Although the timings to accomplish the proposed tasks were similar to the compared straightforward affine mapping, it was perceived by the users that the TPS pose mapping was more accurate in regions closer to the workspace boundaries.
Acknowledgements The authors would like to thank: (1) FAPEAL/CAPES grant 05/2018 for funding and supporting the research; (2) The Instituto de Computação (IC/UFAL) for providing the necessary infrastructure for the development of this project.

Author Contributions Bruno Lima: Methodology, Software, Investigation, Writing - original draft preparation, Writing - review and editing; Lucas Amaral: Methodology, Software, Investigation, Writing - review and editing; Givanildo Nascimento-Jr: Methodology, Software, Investigation, Writing - original draft preparation, Writing - review and editing; Victor Mafra: Methodology, Investigation, Writing - original draft preparation, Writing - review and editing; Bruno Georgievich Ferreira: Methodology, Investigation, Writing - review and editing; Tiago Vieira: Conceptualization, Investigation, Writing - review and editing, Resources, Supervision; Thales Vieira: Conceptualization, Investigation, Writing - review and editing, Resources, Supervision.

Funding This research was supported by FAPEAL/CAPES grant 05/2018.

Availability of data and material Our collected depth images dataset is publicly available through a link provided in Section 7.1.

Code Availability Code used in this article is stored in GitHub and will be made publicly available.

Declarations

Ethics approval Ethics approval was not needed since participants were not explicitly identified, nor asked to perform dangerous, harmful or invasive tasks.

Consent to participate Informed consent was obtained from all individual participants included in the study for research purposes.

Consent for Publication Informed consent was obtained from all individual participants regarding publishing their data and photographs for research purposes.

Conflict of Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Adithya, V., Rajesh, R.: A deep convolutional neural network approach for static hand gesture recognition. Procedia Comput. Sci. 171, 2353–2361 (2020)
2. Bao, P., Maqueda, A.I., del Blanco, C.R., García, N.: Tiny hand gesture recognition without localization via a deep convolutional network. IEEE Trans. Consum. Electron. 63(3), 251–257 (2017)
3. Bookstein, F.L.: Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. 11(6), 567–585 (1989)
4. Brown, B.J., Rusinkiewicz, S.: Non-rigid range-scan alignment using thin-plate splines. In: Proceedings. 2Nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004, 3DPVT 2004., pp. 759–765. IEEE (2004)
5. Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
6. Chen, Y., Tian, Y., He, M.: Monocular human pose estimation: A survey of deep learning-based methods. Comput. Vis. Image Underst. 192, 102897 (2020)
7. Chen, Z., Wang, Z., Liang, R., Liang, B., Zhang, T.: Virtual-joint based motion similarity criteria for human–robot kinematics mapping. Robot. Auton. Syst. 125, 103412 (2020)
8. Delmerico, J., Mintchev, S., Giusti, A., Gromov, B., Melo, K., Horvat, T., Cadena, C., Hutter, M., Ijspeert, A., Floreano, D., et al.: The current state and future outlook of rescue robotics. Journal of Field Robotics 36(7), 1171–1191 (2019)
9. Denso: VP6242 Specs. https://www.denso-wave.com/en/robot/product/five-six/vp.html. Online; Accessed in 28th April, 2020
10. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. CoRR arXiv:abs/1411.4389. http://arxiv.org/abs/1411.4389 (2014)
11. D’Orazio, T., Marani, R., Renò, V., Cicirelli, G.: Recent trends in gesture recognition: How depth data has improved classical approaches. Image and Vision Computing 52, 56–72 (2016). https://doi.org/10.1016/j.imavis.2016.05.007, https://www.scienc e/article/pii/S0262885616300853
12. Duchon, J.: Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces. In: Constructive Theory of Functions of Several Variables, pp. 85–100. Springer (1977)
13. Goodrich, M.A., Schultz, A.C.: Human-robot interaction: A survey Now. Publishers Inc (2008)
14. Gowtham, S., Krishna, K.M.A., Srinivas, T., Raj, R.G.P., Joshuva, A.: Emg-Based control of a 5 Dof robotic manipulator. In: 2020 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), pp. 52–57. https://doi.org/10.1109/WiSPNET48689.2020.9198439 (2020)
15. Hu, B., Wang, J.: Deep learning based hand gesture recognition and uav flight controls. Int. J. Autom. Comput. 17(1), 17–29 (2020)
16. Huang, H., Chong, Y., Nie, C., Pan, S.: Hand gesture recognition with skin detection and deep learning method. In: Journal of Physics: Conference Series, Vol. 1213, pp. 022001. IOP Publishing (2019)
17. Hutchinson, M.F.: Interpolating mean rainfall using thin plate smoothing splines. International Journal of Geographical Information Systems 9(4), 385–403 (1995)
18. Károly, A.I., Galambos, P., Kuti, J., Rudas, I.J.: Deep learning in robotics: Survey on model structures and training strategies. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51(1), 266–279 (2020)
19. LaViola, J.J.: 3D gestural interaction: The state of the field. International Scholarly Research Notices. https://www.hindawi.com/journals/ism/2013/514641/ (2013)
20. Lee, W., Park, J., Park, C.H.: Acceptability of tele-assistive robotic nurse for human-robot collaboration in medical environment. In: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, HRI ’18, pp. 171–172. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3173386.3177084
21. Li, G., Tang, H., Sun, Y., Kong, J., Jiang, G., Jiang, D., Tao, B., Xu, S., Liu, H.: Hand gesture recognition based on convolution neural network. Clust. Comput. 22(2), 2719–2729 (2019)
22. Likert, R.: A technique for the measurement of attitudes. Archives of psychology (1932)
23. Lima, B., Junior, G.N., Amaral, L., Vieira, T., Ferreira, B., Vieira, T.: Real-time hand pose tracking and classification for natural human-robot control. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, pp. 832–839. INSTICC, SciTePress. https://doi.org/10.5220/0007384608320839 (2019)
24. Lipton, J.I., Fay, A.J., Rus, D.: Baxter’s homunculus: Virtual reality spaces for teleoperation in manufacturing. IEEE Robot. Autom. Lett. 3(1), 179–186 (2017)
25. Liu, H., Fang, T., Zhou, T., Wang, Y., Wang, L.: Deep learning-based multimodal control interface for human-robot collaboration. Procedia CIRP 72, 3–8 (2018). https://doi.org/10.1016/j.procir.2018.03.224, https://www.sciencedirect.com/science/article/pii/S221287118303846. 51st CIRP Conference on Manufacturing Systems
26. Lo Presti, L., La Cascia, M.: 3d skeleton-based human action classification. Pattern Recogn. 53, 130–147 (2016)
27. Lun, R., Zhao, W.: A survey of applications and human motion recognition with microsoft kinect. International Journal of Pattern Recognition and Artificial Intelligence 29(05), 1555008 (2015)
28. Microsoft: Kinect - windows app development. https://developer.microsoft.com/en-us/windows/kinect (2020)
29. Miranda, L., Vieira, T., Martinez, D., Lewiner, T., Vieira, A.W., Campos, M.F.: Real-time gesture recognition from depth data through key poses learning and decision forests. In: 2012 25Th SIBGRAPI Conference on Graphics, Patterns and Images. pp. 268–275. IEEE (2012)
30. Moe slund, T.B., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. Computer Vision and Image Understanding 104(2-3), 90–126 (2006)
31. Molchanov, P., Gupta, S., Kim, K., Kautz, J.: Hand gesture recognition with 3d convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1–7 (2015)
32. Mujahid, A., Awan, M.J., Yasin, A., Mohammed, M.A., Damasevičius, R., Maskeliūnas, R., Abdulkareem, K.H.: Real-time hand gesture recognition based on deep learning yoloV3 model. Appl. Sci. 11(9), 4164 (2021)
33. Neverova, N., Wolf, C., Taylor, G.W., Nebout, F.: Multi-scale deep learning for gesture detection and localization. In: European Conference on Computer Vision, pp. 474–490. Springer (2014)
34. Nuzzi, C., Pasinetti, S., Lancini, M., Docchio, F., Sansoni, G.: Deep learning-based hand gesture recognition for collaborative robots. IEEE Instrumentation & Measurement Magazine 22(2), 44–51 (2019)
35. Oudah, M., Al-Naji, A., Chahl, J.: Hand gesture recognition based on computer vision: A review of techniques. Journal of Imaging 6(8), 73 (2020)
36. Oyedotun, O.K., Khashman, A.: Deep learning in vision-based static hand gesture recognition. Neural Comput. Appl. 28(12), 3941–3951 (2017)
37. Poppe, R.: Vision-based human motion analysis: an overview. Computer Vision and Image Understanding 108(1-2), 4–18 (2007)
38. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artificial Intelligence Review 43(1), 1–54 (2015)
39. Ravichandar, H., Polydoros, A.S., Chernova, S., Billard, A.: Recent advances in robot learning from demonstration. Annual Review of Control, Robotics, and Autonomous Systems 3(1), 297–330 (2020). https://doi.org/10.1146/annurev-control-100819-063206
40. Robotiq: 2F-85 Spec. https://robotiq.com/products/adaptive-grippers (2018). Online; Acessado em 26 out. 2018
41. Roeder, L.: Netron https://github.com/lutzroeder/netron (2021)
42. Rohr, K., Steidl, H.S., Sprengel, R., Bel, W., Buzug, T.M., Weese, J., Kuhn, M.: Point-based elastic registration of medical image data using approximating thin-plate splines. In: International Conference on Visualization in Biomedical Computing, pp. 297–306. Springer (1996)
43. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Pinoccchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR 2011, pp. 1297–1304. Ieee (2011)
44. Stanford: Robotic operating system. https://www.ros.org (2020)
45. Student: The probable error of a mean. Biometrika pp. 1–25 (1908)
46. Suarez, J., Murphy, R.R.: Hand gesture recognition with depth images: A Review. In: 2012 IEEE RO-MAN: the 21St IEEE International Symposium on Robot and Human Interactive Communication, pp. 411–417. https://doi.org/10.1109/ROMAN.2012.6343787 (2012)
47. Villani, V., Pini, F., Leali, F., Secchi, C.: Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. Mechatronics 55, 248–266 (2018)
48. Villaroman, N., Rowe, D., Swan, B.: Teaching natural user interaction using openni and the microsoft kinect sensor. In: Proceedings of the 2011 Conference on Information Technology Education, pp. 227–232 (2011)
49. Xiang, L., Echtler, F., Kerl, C., Wiedemeyer, T., Gordon, R., Facioni, F., Wareham, R., Goldhoorn, M., Fuchs, S., Blake, J., et al.: libfreenect2: Release 02 (2016)
50. Zacharakis, A., Kostavelis, I., Gasteratos, A., Dokas, I.: Safety bounds in human robot interaction: A survey. Safety Sci. 127, 104667 (2020)
51. Zennaro, S., Munaro, M., Milani, S., Zanuttilgh, P., Bernardi, A., Ghidoni, S., Menegatti, E.: Performance evaluation of the 1St and 2Nd generation kinect for multimedia applications. In: 2015 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6. https://doi.org/10.1109/ICME.2015.7177380 (2015)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Bruno Lima He has a degree in Electronics from the Federal Institute of Alagoas (IFAL, 2012). Graduated as a Computer Engineer from the Federal University of Alagoas (UFAL, 2018). Also graduated with a Master’s degree in Computer Science (UFAL, 2021), in the subfield of Computer Vision and Machine Learning. Currently, he is a Ph.D. student at the University of Salerno (Unisa). He is mainly interested in robotics and related areas.

Lucas Amaral He completed his high school in Colégio Santíssima Trindade (2010). He has a degree in Computer Science from the Federal University of Alagoas (2019) and a Master’s degree in computer science from the same university. She has experience in the field of Computer Science. Working mainly on the following topics: Brazilian Sign Language, deep learning, dynamic sign language recognition, computer vision.

Givanildo Nascimento-Jr He received a degree in Computer Engineering from the Federal University of Alagoas in 2022. His research interests include image processing, information theory, data reduction, and machine learning applications.

Victor Mafra He is a graduate student in Computer Engineering at the Federal University of Alagoas. Interests involve data science and engineering, artificial intelligence, and backend development. Currently working as a backend software developer.
Bruno Georgevich Ferreira He is a Ph.D. student in computer engineering at the Faculty of Engineering from the University of Porto. He is a Master in Computer Science and also a Computer Engineer from the Federal University of Alagoas. His main area of research is computer vision, more specifically the part of object detection. He is interested in combining object detection with few-shot learning.

Tiago Vieira Born in São Paulo, Brazil, he is currently engaged in R&D projects related to computer vision using deep learning for different applications, such as human-machine interaction and via skeleton estimation, smart cities, industrial inspection for automation, object detection for different purposes such as security monitoring, defect detection on electronic devices, PCB inspection, gun detection, medical image analysis, among others. He is a professor (since 2014) at the Institute of Computing at the Federal University of Alagoas (UFAL) and obtained his Ph.D. in Computer Engineering from the Politécnico di Torino (Italy, 2013). He also holds a Ph.D. in Electrical Engineering from the Federal University of Pernambuco (UFPE - Brazil, 2013).

Thales Vieira He is a Professor at the Federal University of Alagoas (UFAL) since 2010. He holds a Ph.D. degree in Mathematics from Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2010. His current research interests include machine learning, computer vision, natural language processing, and pattern recognition.

**Affiliations**

Bruno Lima1 · Lucas Amaral1 · Givanildo Nascimento-Jr1 · Victor Mafra1 · Bruno Georgevich Ferreira1 · Tiago Vieira1 · Thales Vieira1

Lucas Amaral
lafa@ic.ufal.br

Givanildo Nascimento-Jr
glnj@ic.ufal.br

Victor Mafra
vmhf@ic.ufal.br

Bruno Georgevich Ferreira
bgf@ic.ufal.br

Tiago Vieira
tvieira@ic.ufal.br

Thales Vieira
thales@ic.ufal.br

1 Institute of Computing, Federal University of Alagoas, Maceió, Brazil