electronics

MDPI

Article

Human-Robot Motion Control Application with Artificial Intelligence for a Cooperating YuMi Robot

Roman Michalík *, Aleš Janota, Michal Gregor and Marián Hruboš

Department of Control & Information Systems, Faculty of Electrical Engineering and Information Technology, University of Žilina, 010 26 Žilina, Slovakia; ales.janota@feit.uniza.sk (A.J.); michal.gregor@feit.uniza.sk (M.G.); marian.hrubos@feit.uniza.sk (M.H.)

* Correspondence: roman.michalik@feit.uniza.sk

Abstract: This paper deals with the application focused on motion control for a cooperating YuMi robot. The use of sensors on gloves or other controls to control robots may restrict both operation of the robot and the operator's activities. Therefore, research focusing on camera-based monitoring of body parts or hand gestures is becoming increasingly popular. The presented approach is based on using a camera to recognize hand gestures and implementing artificial intelligence to control a YuMi robot with Python using a TCP/IP connection. The program can be expanded by integrating other IoT devices to help control the robot and collect data useful for a needed application.

Keywords: artificial intelligence; cooperative robotics; human-robot interaction

check for updates

Citation: Michalík, R.; Janota, A.; Gregor, M.; Hruboš, M. Human-Robot Motion Control Application with Artificial Intelligence for a Cooperating YuMi Robot. *Electronics* **2021**, *10*, 1976. https://doi.org/10.3390/electronics10161976

Academic Editor: Imre J. Rudas

Received: 8 July 2021
Accepted: 13 August 2021
Published: 17 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

For decades, automation has helped manufacturers to improve continuity of production, flexibility, competitiveness, productivity, and employee safety. In today's world, manufacturers are increasingly turning to collaboration-based automation. Flexible, intuitive, collaborative automation is a natural solution at a time when rapid changes are taking place, reducing the time required for planning, budgets, costs, hiring and training employees to acquire new skills, or the time required to implement large plant configurations.

Although the development of collaborative robots (cobots) persists, cobots are already successfully integrated into various processes. They have been used in a variety of applications from industry through healthcare and to research. There are more differences in how a conventional industrial robot differs from cobots. While a conventional industrial robot is isolated from human operators (by fencing, optical barriers, or other means of stopping the robot's movement), cobots work directly next to humans. They help humans with routine work, handling miniature components, and cooperation.

Robots can be very powerful, so they are subject to very strict safety regulations. The aim is that in the case of contact of the robot with a human, the pressure on the person does not exceed the set limit, i.e., it does not cause pain or even injury. Safety requirements for collaborative robotic applications are addressed in ISO/TS 15066 as mentioned in the work of [1]. Various simulation applications also contribute to a safe workplace. We can simulate the work process using 3D models in a virtual environment that mimics the real environment of a robot. Industrial development toward automation has opened the door to intelligent robots to reduce human labor to a minimum.

In the era of automation, robots have not only eliminated human intervention in repetitive tasks but are also considered a suitable replacement for the human worker for complex jobs where decision-making skills associated with movement planning are essential. Such a futuristic setting may also require sharing workspace between humans and robots. Therefore, robots that can learn skills from humans only by observing how they perform tasks would be quite useful for this purpose. Because modern intelligent

robots come with complex architectures (hardware and software), there are also challenges associated with operating these robots. The demand for a specially trained workforce is growing because robots need to be programmed manually for different applications. The development of intelligent robot control systems that can help people with their daily needs or can be used as a substitute for human labor in various forms of tasks has always been a challenging and interesting problem in the field of robotics.

Advances in collaborative automation can dramatically increase productivity and reduce costs in many manufacturing processes. Collaborative automation is designed to be easy to learn and quick to deploy, so the internal staff can move, change, and redeploy it with minimal assistance from more experienced employees. The events connected with the pandemic have made the problems addressed in this paper even more pressing and the topic more relevant. Its focus aims to expand a comprehensive view of the issue of extension of sensory systems of a collaborative robot, modeling and simulation of a virtual environment for human interaction, and the implementation of artificial intelligence in the decision-making logic of the collaborative robot. There are many ways to detect an object in an image. The idea of this paper is to use the increasingly popular deep learning methods to detect the operator's body parts in the image. The motivation is to control the robot without direct manual programming or the use of other sensors. This eliminates direct contact with the operator, increases mobility, and removes restrictions. We have successfully contributed with an interface between the computer and the YuMi robot; control algorithm for the Astra S camera; algorithm to process data from the AlphaPose and calculate angles for the YuMi robot; algorithm to process data from the Halpe, count fingers, and send commands to the YuMi robot.

2. Human Pose Estimation

Estimating the human pose is an important issue that has enjoyed the attention of the computer vision community for the past few decades. The estimate of the human pose is defined as a problem of locating human joints (also known as "keypoints": elbows, wrists, etc.) in pictures or videos. It is also defined as the search for a specific pose in the space of all joint poses. The 2D pose estimate is based on an estimate of the x, y position coordinates for each joint in the image.

Human pose estimation can be used in several applications. It is often used to recognize actions, animations, games, etc. This is a difficult task because barely visible joints, clothing, and changes in lighting cause problems; furthermore, some parts of the body can be outside of the field of view or occluded by other objects, which makes the task even more challenging.

There are different approaches to estimating 2D human pose, as listed in the work of [2]. The classic approach to estimating the joint position is to use a frame of image structures. The basic idea is to represent a person using a set of parts arranged in a deformable (not rigid) configuration. Significant research effort has been invested in this area, leading to a number of different variants of this main idea, some of which are very sophisticated.

Nevertheless, these classical methods have severe limitations, and in recent years, since the introduction of deep learning, approaches to human pose estimation are starting to take a significantly different route, as exemplified by Toshev and Szegedy's DeepPose system [3]: the first major contribution to deep-learning-based human pose estimation, but also many other more recent works.

A common building block of these deep-learning-based solutions has been the convolutional neural architecture. This architecture encodes useful inductive bias concerning the general characteristics of visual data. In recent years, convolutional networks (and other architectures with useful inductive biases) have largely made it possible to replace all former approaches based on hand-crafted preprocessing, probabilistic graphical models, and such. This new strategy, apart from not relying on complex hand-crafted routines that take decades of research to perfect, has also brought about drastic improvements in terms

of sheer performance over the classical methods. This is not to say that there are no settings in which classical approaches can outperform deep learning; however, they are generally settings where data is very scarce and where deep neural models generalize poorly as a consequence.

To provide a more concrete example, let us turn back to the aforementioned DeepPose system: here, the pose estimation is formulated as a regression problem and solved using a convolutional network. One of the important aspects of this approach is that it reasons about the pose in a holistic fashion, which allows it to provide (reasonable) position estimates even for joints that are occluded.

The model uses an AlexNet backbone. That is to say, the core of the deep network uses the architecture introduced in the work of [4] for image classification and pretrained on the ImageNet data set, which is a large computer vision data set for classification into 1000 different classes. The final layer of the DeepPose model outputs the x, y coordinates of the joints.

Other works in the same vein followed shortly (Table 1), such as the publication on effective object localization using convolutional networks by Tompson et al. [5], the article on convolutional pose automats by Wei et al. [6], human pose estimation with iterative error feedback by Carreira et al. [7], folded hourglass networks for human pose estimation by Newell et al. [8], simple starting points for estimating and monitoring human pose by Xiao et al. [9], in-depth learning of high-resolution representation for estimating human pose by Sun et al. [10] and many more.

Table 1. Characteristics of the related literature on human pose estimation.

Publication	(MPII) PCKh@0.5	Characteristics
Tompson et al. [5]	82	In essence, the model consists of the heat-map-based parts model for coarse localization, a module to sample and crop the convolution features at a specified (x, y) location for each joint, as well as an additional convolutional model for fine-tuning. A critical feature of this method is the joint use of a ConvNet and a graphical model. The graphical model learns typical spatial relationships between joints.
Wei et al. [6]	88.5	Uses a pose machine. A pose machine consists of an image feature computation module followed by a prediction module. Convolutional Pose Machines are completely differentiable, and their multi-stage architecture can be trained end to end. They provide a sequential prediction framework for learning rich implicit spatial models and work very well for human pose.
Carreira et al. [7]	81.3	The overall working is straightforward: Predict what is wrong with the current estimates and correct them iteratively. Instead of directly predicting the outputs in one go, they use a self-correcting model that progressively changes an initial solution by feeding back error predictions, and this process is called iterative error feedback (IEF).
Newell et al. [8]	90.9	The network consists of steps of pooling and upsampling layers, which look like an hourglass, and these are stacked together. The design of the hourglass is motivated by the need to capture information at every scale. While local evidence is essential for identifying features such as faces hands, a final pose estimate requires global context. The person's orientation, the arrangement of their limbs, and the relationships of adjacent joints are among the many cues that are best recognized at different scales in the image (smaller resolutions capture higher-order features and global context).
Xiao et al. [9]	92.3	The network structure is quite simple and consists of a ResNet + few deconvolutional layers at the end. While the hourglass network uses upsampling to increase the feature map resolution and puts convolutional parameters in other blocks, this method combines them as deconvolutional layers in a very simple way.
Sun et al. [10]	92.3	HRNet (high-resolution network) follows a very simple idea. The architecture starts from a high-resolution subnetwork as the first stage and gradually adds high-to-low-resolution subnetworks one by one to form more stages and connect the multi-resolution subnetworks in parallel. Repeated multi-scale fusions are conducted by exchanging information across parallel multi-resolution subnetworks over and over through the whole process. This architecture does not use intermediate heatmap supervision, unlike the stacked hourglass.

2.1. Human Pose Estimation: The Problem Formulation

Under these neural approaches, pose estimation is essentially framed as a regression problem, where the ground truth is derived from an annotated data set. The loss function is usually relatively simple, e.g., the (sometimes weighted) L1/L2 distance between the predicted and annotated position of the joints.

The form of the annotations tends to vary slightly from data set to data set, but they generally contain a rectangular bounding box for each person and then the annotations of the relevant keypoints. To provide an example, the MPII data set [11] annotates the following keypoints: the left/right ankle; the left/right knee; the left/right hip; the pelvis; the thorax; the upper neck; the head top; the left/right wrist; the left/right shoulder; the left/right elbow. It also specifies whether each of them is visible in the image.

Performance is typically evaluated in terms of mean average precision (mAP) based on object keypoint similarity (OKS), which is the metric of choice in COCO data set challenges [12]. OKS is defined as follows [12]:

$$\mathrm{OKS} = \frac{\sum_i \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \tag{1}$$

where d_i is the Euclidean distance between the predicted position of keypoint i and its ground truth position; v_i is the binary visibility flag of keypoint i; s is the object scale (the square root of the object segment area). Symbol k_i denotes a per-keypoint-type constant; the standard deviation of, say, shoulder keypoints w.r.t. object scale is significantly different from that of, say, the head top, so to make sure that OKS is perceptually meaningful, the k_i are set using $k_i = 2\sigma_i$, where $\sigma_i = \mathbb{E}\left(d_i^2/s^2\right)$ evaluated over 5000 redundantly annotated images in the validation set [12].

Predictions are considered correct if the predicted keypoint position is closer to the ground truth position than a certain threshold. The average precision (AP) is then computed for each class of object (if there are multiple classes). Averaging over the classes yields the mean average precision (mAP). The OKS threshold is typically indicated using the @ symbol, e.g., AP@0.5 for AP with the OKS threshold of 0.5 or AP@0.5:0.95 for AP being averaged over multiple OKS thresholds (from 0.5 to 0.95 with the increment of 0.05).

The mAP is also used as an evaluation metric in the visual object detection domain, where; however, the intersection over union (IoU) measure is used in place of OKS to compare the bounding boxes [13].

Other evaluation metrics used in human pose estimation include, e.g., the percentage of correct keypoints (PCK) [14] or the percentage of correct parts (PCP) [14].

2.2. Pose Estimation Using AlphaPose

Estimating the human pose is a major computer vision challenge. In practice, recognizing the pose of several people is much more difficult than recognizing the pose of one person in the image. Recent attempts have approached this problem using either a two-stage structure as in the work of [15] or a parts-based structure presented in the work of [16].

The two-stage structure first detects the boundary boxes of a human and then independently estimates the position within each box. The part-based structure first detects body parts independently and then assembles the detected body parts to create more human poses. Both structures have their advantages and disadvantages. In a two-stage structure, the accuracy of the pose estimation greatly depends on the quality of the bounding boxes detected. In a part-based structure, collected human poses are ambiguous when two or more people are too close together. AlphaPose [17] uses a two-stage structure. Their goal is to reveal accurate human poses, even if they obtain inaccurate bounding boxes. AlphaPose has tested its pose detection approach on the MSCOCO Keypoints Challenge data set [18], where it achieved the mAP@0.5:0.95 of 73.3 and the mAP@0.5 of 89.2 [19]. For tests on the MPII data set [11], they have achieved the mAP@0.5 of 82.1 [19].

3. YuMi Motion Control Application

The YuMi collaborative robot is the first generation of a two-armed robot from ABB with seven degrees of freedom for each arm (Figure 1). The name YuMi originated from the words "you and me". It is designed to suit flexible production for the assembly of small parts. It can work close to a person because its shoulders are wrapped in soft parts and can predict collisions.

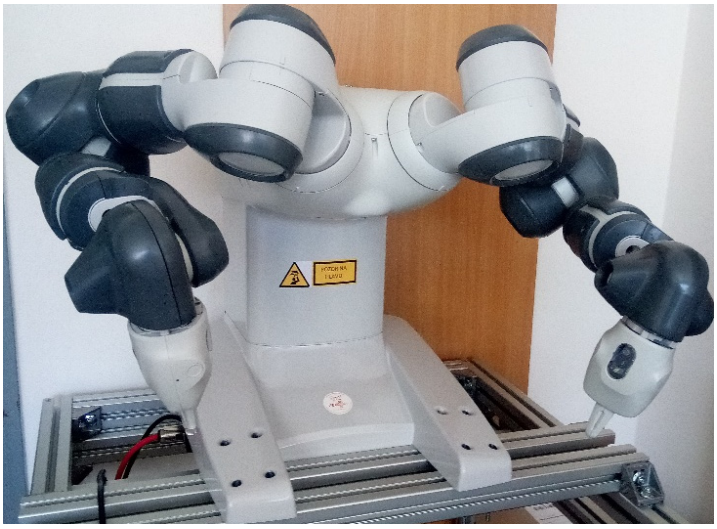


Figure 1. A two-armed YuMi robot.

3.1. Interface between Computer and YuMi Robot

We created a TCP/IP (transmission control protocol/Internet protocol) interface between the computer and the YuMi robot. The standard interface does not support the connection and free control of multiple devices, such as our camera. With our own interface, we can both control the robot and also control and communicate with other devices. To this end, we have created two servers (written in Rapid), one for each arm of the YuMi robot using the RobotStudio software. Each arm also has a separate motion program. Both of these are labeled L for the left arm and R for the right arm in Figure 2. On the client side, we are using Python.

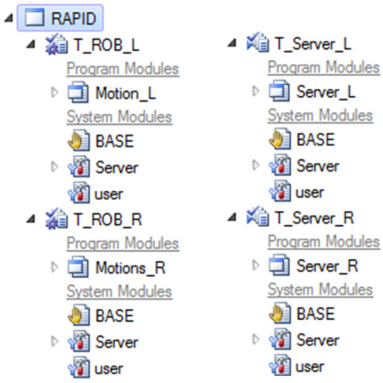


Figure 2. Illustration of modules for each arm of YuMi robot.

Programs in RobotStudio contain the settings of the stack for commands, the IP address on which the robot is located, and the parameters of connection and execution of commands. The Python driver includes functions for calculating angles, robot parameters, and TCP/IP connection.

3.2. Usage of AlphaPose in our Control System

We used AlphaPose to design a YuMi robot control system based on pose detection. AlphaPose extracts the human pose from the visual input, which is then encoded in a JSON (JavaScript Object Notation) file.

To process this data, we created a Python application to calculate the angles between the joints by loading the entire JSON file and calculating the required angles for controlling the YuMi robot. Our approach is to calculate the rotation of the shoulder and elbow for the YuMi robot to imitate the movement of the operator. First, we obtain the coordinates of the relevant keypoints ($\boldsymbol{k}_n$) from the output file. Given three such keypoints $\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3$, we compute the angle between vectors $\boldsymbol{k}_1 - \boldsymbol{k}_2$ and $\boldsymbol{k}_3 - \boldsymbol{k}_2$, resolving the ambiguity regarding the cosine inverse and converting from radians to degrees:

$$\boldsymbol{v}_1 = \boldsymbol{k}_1 - \boldsymbol{k}_2 \tag{2}$$

$$\boldsymbol{v}_2 = \boldsymbol{k}_3 - \boldsymbol{k}_2 \tag{3}$$

$$\alpha' = \cos^{-1}\left(\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\|_2 \cdot \|\boldsymbol{v}_2\|_2}\right) \tag{4}$$

where $\|\cdot\|_2$ denotes the L_2 norm. We then find

$$\boldsymbol{p} \in \mathbb{R}^3 : \boldsymbol{p} \perp (\boldsymbol{k}_2 - \boldsymbol{k}_1);\ \|\boldsymbol{p}\|_2 = 1 \tag{5}$$

and define

$$f = \begin{cases} -1 & \boldsymbol{p} \cdot (\boldsymbol{k}_3 - \boldsymbol{k}_2) < 0 \\ 1 & \text{else} \end{cases} \tag{6}$$

so that the angle α (in degrees) can then be defined as

$$\alpha = f \cdot (180 - \alpha' \cdot 180/\pi). \tag{7}$$

The calculation of these angles is then used in the YuMi robot controller, which sends control commands to the robot server. The robot then moves to the given position, which imitates the pose of the operator. The resulting robot position corresponding to the JSON output file is shown in Figure 3. The system can process all poses of the operator's arms in this way and imitate them, but currently only on a plane and not in the full 3D space.

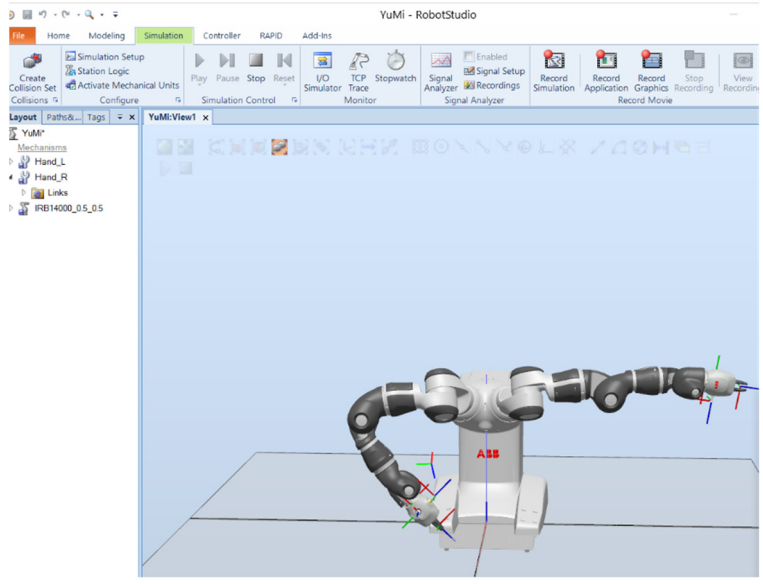


Figure 3. YuMi imitating movement in RobotStudio simulation.

AlphaPose is able to process around 20 frames per second, and the additional cost of the angle computations is negligible, so any additional latency is mostly only due to the TCP/IP connection.

3.3. Estimation of Key Points Using Halpe

Halpe is a joint project by the AlphaPose team and the HAKE (Human Activity Knowledge Engine) team from Jiao Tong University. Its goal is to improve the estimation of human poses. It provides much more detailed annotation of human key points. It describes a total of 136 key points for each person, including the head, the face, the torso, the arms, the hands, the legs, and the feet [20].

Halpe recognizes 136 points on the human body, just as AlphaPose recognizes 17. The output is also a JSON file that contains the position data. Using AlphaPose, we imitated the movement of a person by the YuMi robot, but with Halpe, we used the potential to obtain more detailed information about the key points of the human hand (Figure 4).

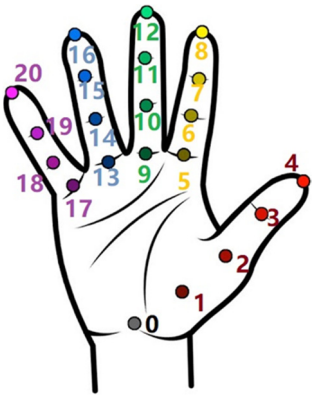


Figure 4. Representation of 21 points on the hand [20].

Halpe [20] uses the YOLOv3 object detector [21] with a ResNet50 [22] network as its backbone. The input size is 256 × 192 pixels. When experimenting with Halpe, we combined knowledge from previous experiments with AlphaPose. The aim was to design an algorithm for recognizing static hand gestures (described mainly through the number and identity of outstretched fingers), to incorporate it into the control system of the YuMi robot, and to verify its functionality. The procedure was as follows:

- Create a program for the camera to acquire the image;
- Create a program to send an image to the server, which will process the image with Halpe;
- Program Halpe image recognition with result return (JSON file);
- Create an algorithm to process the image recognition result and send the motion instruction to the YuMi robot server.

To obtain the image, we used the Astra S camera (Figure 5).

Figure 5. Camera Astra S.

The program is written in Python and consists of three main parts. The first part is the import of the necessary libraries for camera control, which are OpenNI and OpenCV. The

second part is the camera settings: camera initialization, obtaining information about the connected camera via USB, setting the camera format. In our case, we chose the basic RGB format with a resolution of 640 × 480 at 30 fps. The third part is just launching the camera and turning on RGB video for the user.

The image processing was delegated to a Halpe-based server running on a GPU workstation, the specification of which is given in Table 2.

Table 2. Server specification for Halpe.

Processor	Graphics Card	RAM	Motherboard	OS
Intel Core i7-7700K @ 4.2–4.5 GHz, 4 C/8 T	NVIDIA GeForce GTX 1080 Ti 11 GB	Kingston 4 × 16 GB DDR4 KHX3000C15/16GX	ASUS TUF Z270 MARK2	Ubuntu 20.04.2 LTS

The program captures and saves an image from the live video feed provided by the camera, which is then sent to the server using a TCP/IP connection. The server uses a dockerized [23] version of Halpe to produce the resulting JSON file with the annotations and returns them back over the same TCP/IP connection. We use and recommend this Docker-based approach because Halpe's setup is quite fragile and only builds correctly with a relatively restricted set of library versions.

The Python program receives the JSON file and runs a subroutine to process the static gesture data. Visualization of keypoints extracted from a sample image is shown in Figure 6. We propose an algorithm that uses the key points from the JSON file to create a static hand gesture descriptor by classifying the observed hand (left/right) and detecting the number and identity of outstretched fingers. The flowchart of our proposed algorithm is shown in Figure 7.

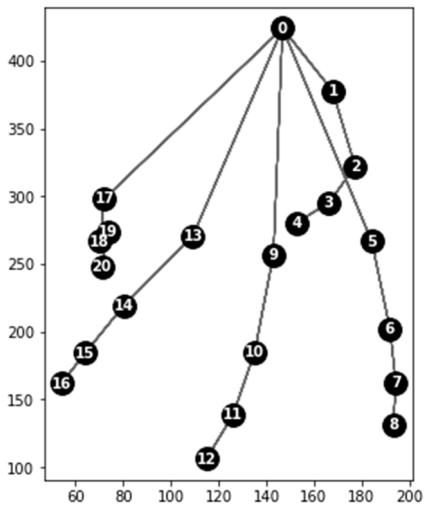


Figure 6. Visualization of hand points.

This gesture descriptor then activates an associated subroutine, which sends a command to the YuMi robot server. If desired, it is possible to assign a certain number/combination of outstretched fingers to a certain path that the robot arm should traverse or simply to set the angles of the individual joints of the robot. If no hand is detected, the robot can be set to either perform no action or to revert to some predefined base pose. An example of this procedure is shown in Figures 8 and 9, where the system is shown two different static hand gestures, and it reacts accordingly by performing the requisite operations. The accuracy of our solution will, of course, depend on the lighting conditions, the distance from the camera, and other factors that may negatively impact the accuracy of the results provided by the Halpe server.

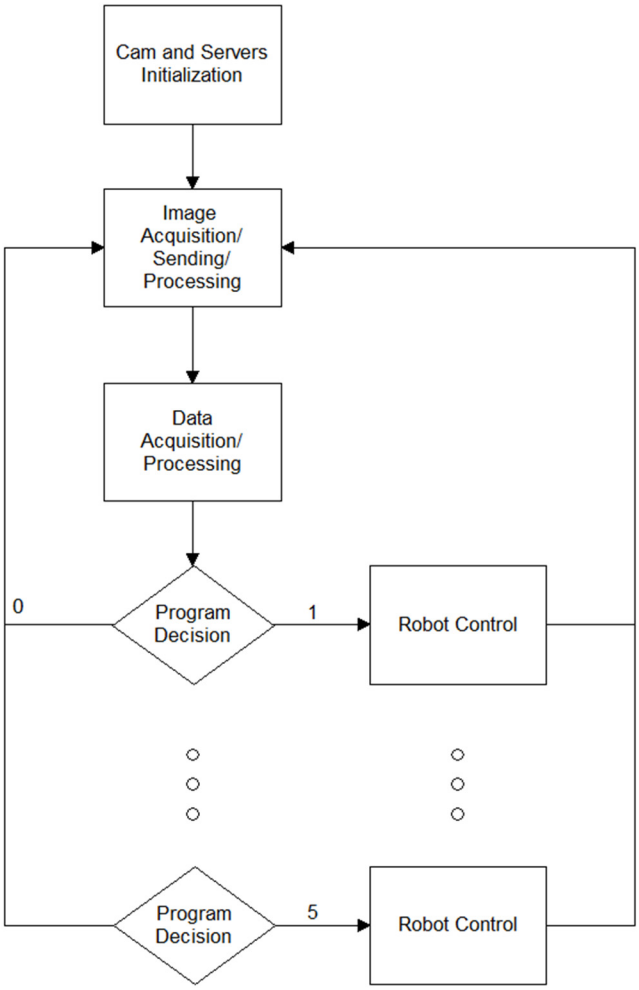
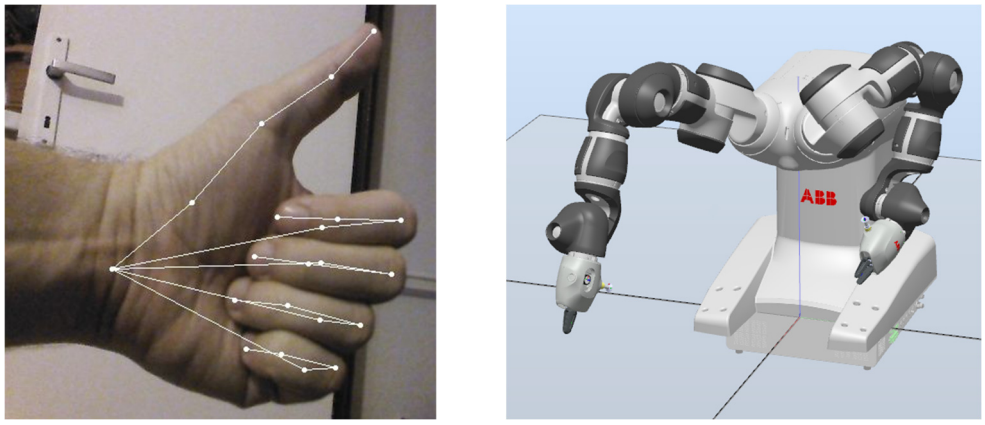
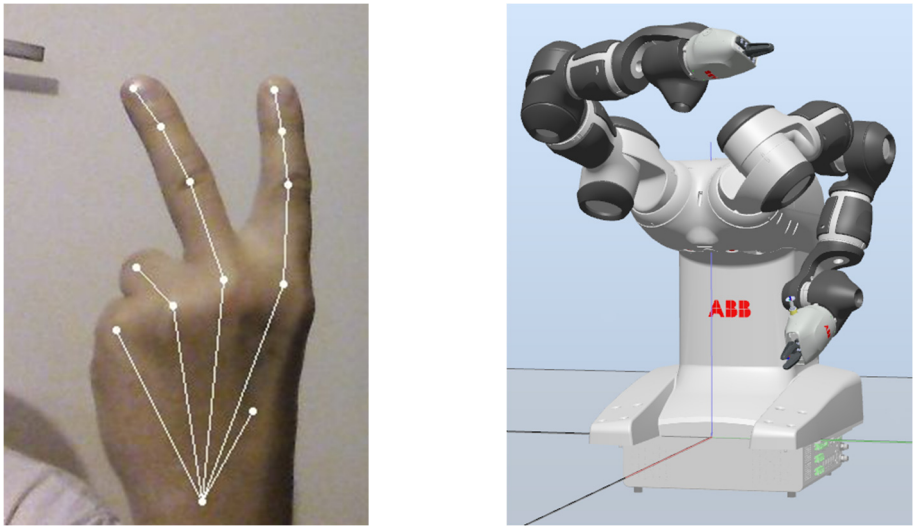


Figure 7. The flowchart of our proposed algorithm.



(**a**) (**b**)

Figure 8. (**a**) Detected points on the hand with the result of 1 extended finger after data processing. (**b**) Movement of the robot arm to the appropriate position according to the number of outstretched fingers.



(**a**) (**b**)

Figure 9. (**a**) Points on the hand with two outstretched fingers detected after data processing. (**b**) Movement of the robot arm to a given position for two outstretched fingers.

We also verified the success by testing at different distances from the camera. The test consisted of creating 40 photos at four different distances (10 photos at each distance) from the camera, namely at:

- 0.5 m;
- 1.5 m;
- 2.5 m;
- 3 m.

The resulting success at different distances (Figure 10) shows that increasing the distance after exceeding 2.5 m from the camera reduces the success of finger detection.

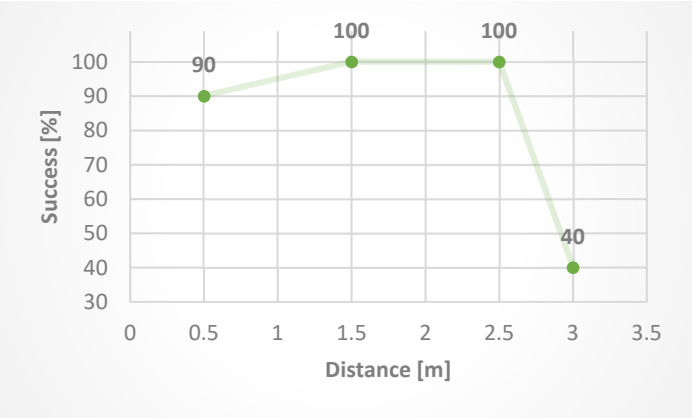


Figure 10. Success of detection at different distances (10 photos at each distance) from the camera for five outstretched fingers.

Success in our case means successfulness of detection of the desired number of outstretched fingers. For example: 10 photos of five outstretched fingers at a distance of 0.5 m have 90% success because 9/10 photos were detected as five outstretched fingers, and 1/10 was detected as four outstretched fingers.

4. Discussion and Conclusions

In this article, we present two distinct systems: the first based on AlphaPose and the second on Halpe. The first represents a smart human-robot interface, which uses AlphaPose

to detect the pose of the human operator, which the robot then mimics. AlphaPose processes the incoming image and creates an output JSON file that indicates the positions of the joints. Our interface then processes these, uses them to compute the target angles for the robot's joints, and sends out the corresponding commands to the two servers that control its arms. As a result, the robot is able to mimic the pose of the operator's arms (currently limited to poses on one plane but easily extensible to full 3D poses).

The second interface uses the Halpe system to gain more detailed information about key points of the human hands. The paper proposes an algorithm that extracts a static hand gesture descriptor consisting of the hand classification (left/right) and the number and identity of outstretched fingers. The solution can bind different sets of commands to different hand gesture descriptors. When a particular static hand gesture is detected, the associated commands are communicated to the servers controlling the robot's arms. The advantage is the control of the robot without direct manual programming or the use of other sensors restricting the movement of the operator. This eliminates direct contact with the operator, increases mobility, and removes restrictions.

Our approach can work with other types of robots if the robot has similar degrees of freedom. Otherwise, the code would have to be modified. The limitation of our algorithm is the number of fingers (one control program per finger) and required hardware equipment. The future directions of research should be aimed at integrating our approach with ROS system, classification, and processing of cloud points of the operator's hands and fusion of data from cameras (like integrated Cognex cameras in the YuMi robot and external RGBD camera).

The success of our algorithm depends mostly on the success of the Halpe subsystem, but under the right lighting conditions and while keeping in the distance range of 1.5–2.5 m from the camera, hand key points are detected reliably. The resulting solution can easily be expanded by integrating other IoT devices to help control the robot and collect data useful for any particular target application. For example, wireless proximity sensors could be used to gather information and help control the robot via LoRaWAN, as mentioned by Mihálik et al. [24].

Author Contributions: Conceptualization, R.M., A.J., M.G. and M.H.; methodology, R.M., A.J. and M.G.; software, R.M. and M.G.; validation, R.M. and M.G.; formal analysis, A.J. and M.G.; investigation, R.M., A.J., M.G. and M.H.; resources, A.J., M.G. and M.H.; data curation, R.M. and M.G.; writing—original draft preparation, R.M. and A.J.; writing—review and editing, M.G. and M.H.; visualization, R.M. and M.G.; supervision, A.J.; project administration, A.J.; funding acquisition, M.G. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by the Cultural and Educational Grant Agency MŠVVaŠ SR, grant number 008ŽU-4/2021: Integrated Teaching for Artificial Intelligence Methods at the University of Žilina.

Acknowledgments: The authors would like to thank Dušan Nemec (University of Žilina) for his major contribution to the TCP/IP and Python interface.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
COCO	Common Objects in Context
HAKE	Human Activity Knowledge Engine
IoT	Internet of Things
ISO/TS	International Organization for Standardization /Technical Specifications
JSON	JavaScript Object Notation
LoRaWAN	Lo(ng) Ra(nge) Wide Area Network
mAP	Mean Average Precision
MPII	Max Planck Institute for Informatics
OpenCV	Open Computer Vision
OpenNI	Open Natural Interaction
OS	Operating System
RGB	Red-Green-Blue
ROS	Robot Operating System
TCP/IP	Transmission Control Protocol/Internet Protocol
USB	Universal Serial Bus
YOLOv3	You Only Look Once, Version 3

References

1. ISO. ISO/TS 15066:2016 Robots and Robotic Devices—Collaborative Robots. Available online: https://www.iso.org/standard/62996.html (accessed on 18 June 2021).
2. Sudharshan Chandra Babu. A 2019 Guide to Human Pose Estimation with Deep Learning. Available online: https://nanonets.com/blog/human-pose-estimation-2d-guide (accessed on 18 June 2021).
3. Toshev, A.; Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1653–1660. [CrossRef]
4. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
5. Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; Bregler, C. Efficient object localization using Convolutional Networks. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 648–656.
6. Wei, S.-E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional Pose Machines. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4724–4732. [CrossRef]
7. Carreira, J.; Agrawal, P.; Fragkiadaki, K.; Malik, J. Human Pose Estimation with Iterative Error Feedback. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4733–4742. [CrossRef]
8. Newell, A.; Yang, K.; Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision—ECCV 2016. ECCV 2016*; Lecture Notes in Computer Science; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9912, ISBN 978-3-319-46483-1.
9. Xiao, B.; Wu, H.; Wei, Y. Simple Baselines for Human Pose Estimation and Tracking. In *Computer Vision—ECCV 2018. ECCV 2018*; Lecture Notes in Computer Science; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11210, ISBN 978-3-030-01230-4.
10. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5686–5696. [CrossRef]
11. Andriluka, M.; Pishchulin, L.; Gehler, P.; Schiele, B. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3686–3693. [CrossRef]
12. COCO. Keypoint Evaluation. Available online: https://cocodataset.org/#keypoints-eval (accessed on 5 August 2021).
13. COCO. Detection Evaluation. Available online: https://cocodataset.org/#detection-eval (accessed on 5 August 2021).
14. Zheng, C.; Wu, W.; Yang, T.; Zhu, S.; Chen, C.; Liu, R.; Shen, J.; Kehtarnavaz, N.; Shah, M. Deep learning-based human pose estimation: A survey. *arXiv* **2020**, arXiv:2012.13392.
15. Pishchulin, L.; Jain, A.; Andriluka, M.; Thormählen, T.; Schiele, B. Articulated people detection and pose estimation: Reshaping the future. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3178–3185. [CrossRef]

16. Insafutdinov, E.; Pishchulin, L.; Andres, B.; Andriluka, M.; Schiele, B. DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model. In *Computer Vision—ECCV 2016. ECCV 2016*; Lecture Notes in Computer Science; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9910, ISBN 978-3-319-46465-7.
17. Fang, H.-S.; Xie, S.; Tai, Y.-W.; Lu, C. RMPE: Regional Multi-person Pose Estimation. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2353–2362. [CrossRef]
18. COCO. COCO 2016 Keypoint Detection Task. Available online: https://cocodataset.org/#keypoints-2016 (accessed on 5 August 2021).
19. Machine Vision and Intelligence Group. AlphaPose. Available online: https://github.com/MVIG-SJTU/AlphaPose (accessed on 24 June 2021).
20. Fang, H. Halpe-FullBody. Available online: https://github.com/Fang-Haoshu/Halpe-FullBody (accessed on 25 June 2021).
21. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
23. Merkel, D. Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2, ISSN 1075-3583.
24. Mihálik, M.; Hruboš, M.; Bubeníková, E. Wireless Proximity Sensor in LoRaWAN Network. In Proceedings of the 2020 International Conference on Applied Electronics (AE), Pilsen, Czech Republic, 8–9 September 2020; pp. 1–4. [CrossRef]