A Real-time Human-Robot Interaction Framework with Robust Background Invariant Hand Gesture Detection

Osama Mazhar, Benjamin Navarro, Sofiane Ramdani, Robin Passama, Andrea Cherubini

A Real-time Physical Human-Robot Interaction Framework with Robust Background Invariant Hand Gesture Detection

Osama Mazhar*, Benjamin Navarro, Sofiane Ramdani,
Robin Passama, Andrea Cherubini

LIRMM, Université de Montpellier, CNRS,
Montpellier, France

Abstract

In the light of factories of the future, we present a reliable framework for real-time safe physical human-robot collaboration using static hand gestures. To ensure productive and safe interaction between robot and human coworkers, it is imperative that the robot extracts the essential information about the human coworker. We address this by designing a framework for *safe* and *intuitive robot programming* based on hand gesture recognition. First, the OpenPose library is integrated with Microsoft Kinect V2, to obtain a 3D estimation of the human skeleton. With the help of 10 volunteers, we record an image dataset of alpha-numeric static hand gestures, taken from the American Sign Language. We name our dataset as OpenSign and release it to the community for benchmarking. The Inception-v3 convolutional neural network is adapted to train the hand gesture detector. To augment the data for training a hand gesture detector, we use OpenPose to localize the hands in the dataset images and segment the backgrounds of hand images using the Kinect depth map. Then, the backgrounds are substituted with random patterns and indoor architecture templates. Fine-tuning of Inception V3 is performed in three phases, to achieve validation accuracy of 99.1% and test accuracy of 98.9%. An asynchronous

*Corresponding author
Email address: osamazhar@yahoo.com (Osama Mazhar)

integration of image acquisition and hand gesture detection is performed to ensure real-time detection of hand gestures. Finally, the proposed framework is integrated in our physical human-robot interaction library OpenPHRI. Using OpenPHRI, we validate the performance of the proposed framework through a complete teaching by demonstration experiment with a robotic manipulator.

Keywords: Physical Human-Robot Interaction, Hand-Gesture Detection, Convolutional Neural Networks, Skeleton Extraction, Real-time Vision, Transfer Learning, OpenPHRI

1. Introduction

The advent of the Industry 4.0, which is a modern trend of automation and data exchange in the manufacturing industry, has proposed the concept of smart factories of the future [1]. This evolving industry demands a more effective and involved collaboration between humans and robots, where each partner can constructively utilize the strengths of the others to increase productivity and work quality [2].

Safety of the human coworkers and an efficacious interaction between humans and robots are key factors of success in such an industrial setting. To ensure safety, the ability of the robot to detect an external force, differentiate between intended and accidental forces and to adapt to the rapidity of the human coworker is essential [3]. Nevertheless, the sense of vision is also imperative for modern collaborative robots to monitor the behavior and actions of their human coworkers for communicating or preventing accidents [4].

Generally, robots are designed and programmed to perform specialized tasks. Hence, it is difficult for an unskilled worker to reprogram the robot for a new task [5]. The traditional robot teaching methods are tedious, non-intuitive and time consuming. Instead, speech and gestures are natural and intuitive ways to communicate/interact with the robot [6]. In this paper, we propose a real-time robust and background independent hand gesture detection module using the concept of transfer learning in convolutional neural networks [7]. We integrate

the proposed hand gesture detection module with our physical human-robot interaction library OpenPHRI [8] for robot control. This ensures on one hand safety of the human coworker through fast communication and real time depth estimation, and on the other hand an intuitive means for robot programming and reprogramming, through hand gestures.

Background and related work are described in Sect. II. We summarize our contributions in Sect. III, while Skeleton extraction and hand localization are detailed in Sect. IV We describe our convolutional neural network for hand gesture detection in Sec. V, while the robotic framework and example industrial application of the proposed framework are presented in Sect. VI. We conclude in section VII.

2. Background and related work

The authors of [1] present an emerging concept of cyber-physical structure which will employ extensive automation and self-organization of machines and component parts in complex manufacturing scenarios, using different sensor modalities. The primary role of human workers in such a setting will be to dictate a production strategy and to supervise its implementation by the corresponding self-organizing production processes. A detailed review of human-robot collaborative assembly in cyber-physical production is presented in [9]. The authors propose a structured classification and solution framework of human-robot collaboration. Typical requirements for symbiotic human-robot collaboration are summarized and a case study of a super-charger assembly of the car engine is presented to validate the proposed framework. This case study explores the feasibility of transforming conventional industrial robotic cells into collaborative environments.

Although some researchers claim to prefer the use of data gloves or wearable sensors to allow free movement of the user or to deal with the occlusions or varying light conditions [10], these sensors are expensive, non-intuitive and limit the dexterity of the person in his/her routine tasks. Many works in the past

have proposed image-based human-robot interaction schemes with the help of gestures. A task oriented intuitive programming procedure is presented in [11] to demonstrate human-like behavior of a dual-arm robot. The authors decompose complex activities in simpler tasks that are performed through task-oriented programming where the focus is given to "what to be done, rather than how to do it". Moreover, through the development of intuitive human interfaces, high level commands are transferred to a sequence of robot motion and actions. For human-robot interaction, they use Microsoft Kinect V1 [12] to extract human skeletal coordinates for gesture detection, and the built-in microphone array of Kinect V1 to detect the oral commands. Whole body gestures (extended arms) are used to achieve robot motion in a dashboard assembly case. Although the idea of task decomposition and controlling the robot through human gestures is beneficial but the considered gestures, similar to that in [13], are non-intuitive and tiring.

In [14] authors presents methods for obtaining human worker posture in a human-robot collaboration task of abrasive blasting. They compare the performance of three depth cameras namely Microsoft Kinect V1, Microsoft Kinect V2 [15] and Intel RealSense R200 [16]. Kinect V1 uses a structured light approach to estimate the depth map, Kinect V2 is a time-of-flight sensor while RealSense R200 has a stereoscopic infra-red setting to produce depth. In the blasting process, the abrasives are suspended in the air or fill the surrounding environment, and significantly decrease the scene visibility. The use of image-based methods to extract human worker pose is challenging in such environments. The experimental observations suggest that Kinect V1 performs best in the real blasting environment, although no concrete reason could explain this. They also present a novel camera rig with an array of four Kinect V1 to cover 180° horizontal field of view.

In [17], the authors present an online robot teaching method that fuses speech and gesture information using text. Kinect V2 localizes hands position in the scene while an inertial measurement unit (IMU) measures its orientation. The gesture and speech data are first converted into a description text, then a

text understanding method converts the text to robot commands. The proposed method is validated by performing a peg-into-hole experiment, placing wire-cut shapes, and an irregular trajectory following task.

To ensure safe interaction, [18] proposes a virtual reality training system for human-robot collaboration. A virtual game simulation is developed for real-time collaboration between industrial robotic manipulators and humans. A realistic virtual human body, with a simple first person shooter view is included, to intuitively simulate the user's vision. A head mount display and Kinect V1 track the human head and skeleton pose respectively. Several interaction tasks are accomplished including selection of objects, manipulation, navigation and robot control. This technique is useful to establish the acceptability of a collaborative robot among humans in a shared workspace as well as to tackle mental safety issues.

In [5] the authors present a strategy to use speech and a Wii controller to program a Motoman HP6 industrial robot. This helps workers with no knowledge of typical programming languages, in teaching different activities to the robot in an intuitive way. A neural network is trained to recognize hand gestures using features extracted from the accelerometer output of the Wii-controller. In [19], the authors train artificial neural networks to classify 25 static and 10 dynamic gestures to control an industrial 5 degrees-of-freedom robotic arm. A data glove, CyberGlove II, and a magnetic tracker, Polhemus Liberty, are used to extract a total of 26 degrees-of-freedom.

The authors of [3] present a study for measuring trust of human coworkers in fence-less human-robot collaboration in industrial robotic applications. To ensure safety of the human coworkers, it is essential to equip the robot with vision sensors to understand its environment and to adapt to the worker's behavior. They also discuss the use of RGB-D cameras to detect pointing gestures and proximity monitoring for safety using the depth information. In [20] authors use human gestures to navigate a wheeled robot through pointing gestures directed on the floor. The interaction scheme also includes detection of facial gestures which often fails, as stated by the authors, because the untrained users make

those gestures subtly.

In [21], the authors propose object recognition through 3D gestures using Kinect V1. They exploit the depth information from Kinect V1 to subtract background of the objects. This strategy often fails if predefined environmental assumptions are not met. Moreover, a histogram matching algorithm is used to recognize the objects placed on a white color table, and such techniques have recently been outperformed by modern deep learning ones like convolutional neural networks [22]. The authors of [23] propose a human-robot interaction system for the navigation of a mobile robot using Kinect V1. The point cloud acquired from Kinect V1 is fit on a skeleton topology with multiple nodes to extract the pose of human operator. This technique is not reliable to obtain the skeletal pose unless the human body non-linear anatomical constraints are modeled in the design of the skeleton topology.

3. Our Contributions

This paper is an extension of our previous work proposed in [24] which presented a tool handover task between robot and human coworker through static hand gestures. A convolutional neural network, inspired mainly by LeNet [25] was developed, to classify four hand gestures. The aim of the previous work was also to build a robust hand gesture detection system. However, the dataset was small, and the hand images were recorded only by one individual. This could not guarantee correct detection of hand gestures made by other individuals and with backgrounds having rich textures.

We extend our work by training a hand gesture detector on ten gestures instead of four as in [24]. Moreover, the backgrounds are now replaced with random pattern/indoor-architecture images to make the detection robust and background invariant. We propose an intuitive interaction setting where a human coworker can instruct commands to the robot via gestures. The contributions in this paper are summarized as follows:

- Development of a real-time hand gesture detection framework that lo-

calizes hands through asynchronous integration of OpenPose 2D skeleton detector and classify hand-gestures at frame-rate of approximately 20fps.

- Training a background-invariant hand-gesture detection system through transfer learning from Inception V3 convolutional neural network.
- On-line release of hand gestures database of Kinect V2 recordings for benchmarking and comparison.
- Integration of the developed hand gesture detection module with our safe physical human robot interaction framework, namely OpenPHRI.
- Validation of the proposed framework for robot teaching and control of Kuka LWR 4+ arm with the detected hand-gestures.

The overall pipeline of the proposed framework is illustrated in Fig. 1. The dotted lines in the figure represents the asynchronous integration between the modules to ensure real-time execution of the system. Each module is described in the following sections in detail.

4. Skeleton Extraction and Hand Localization

For safe physical Human-Robot Interaction, it is essential for the robot to understand its environment, particularly the human coworker. In this research, we opted for Microsoft Kinect V2 as the main sensor to capture the visual information of the human coworker. Kinect V2 is a time-of-flight sensor and provides a larger field-of-view and higher resolution RGB and depth images than its predecessor Kinect V1. This allows the robot to extract functional information from the scene, like human(s) presence or object/obstacle detection, including depth perception.

4.1. Skeleton Extraction Module

In our work we utilize OpenPose [26, 27], to extract skeletal joint coordinates, as in [28, 29]. This library returns 2D skeletal coordinates (x_i, y_i, c_i), for $i =$

$1, ..., 18$, from a RGB image, using confidence maps and parts affinity fields in a multi-person scene; x_i and y_i are the abscissas and ordinates respectively of 18 COCO body parts [30], while c_i represent their confidence measure.

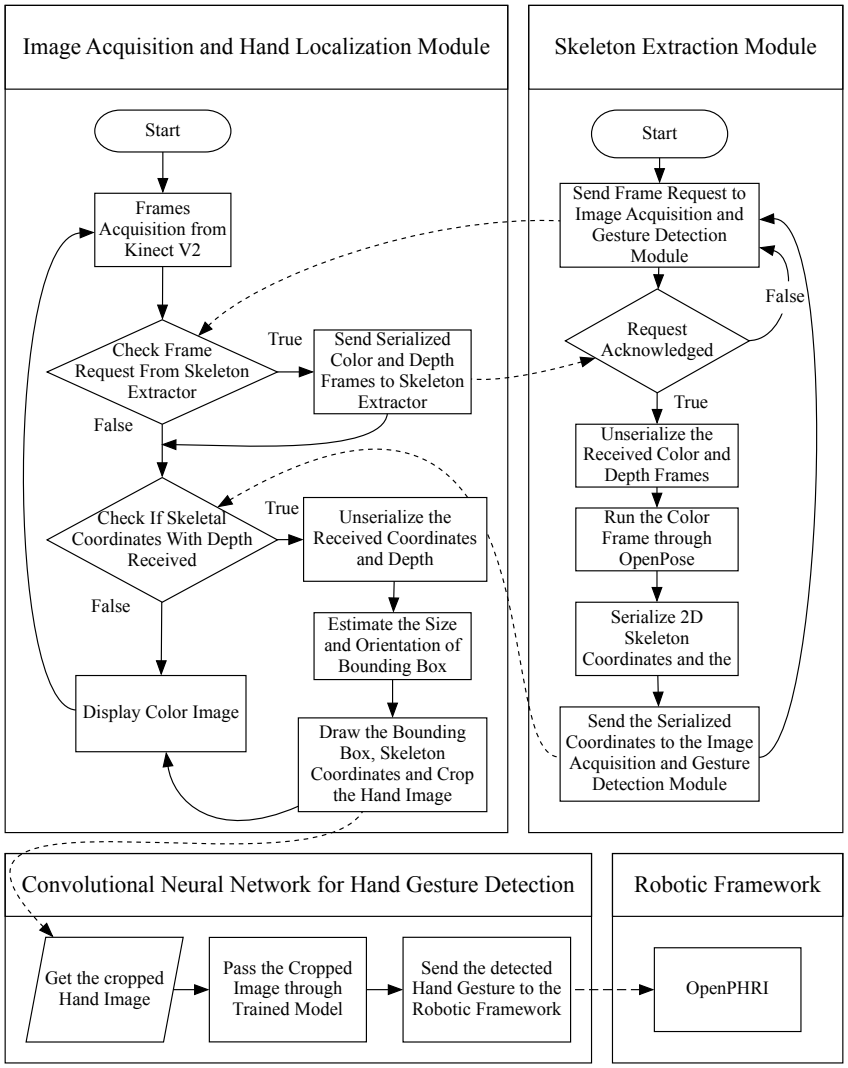


Figure 1: The overall pipeline of our framework for pHRI using hand gestures

OpenPose works on the principle of "convolutional pose machines" described in [27]. OpenPose is a robust skeleton extractor and is not trained on pre-defined body poses. It extracts each joint independently from the overall body pose.

It is therefore preferred over libraries like OpenNI and Microsoft SDK as they are often not accurate in skeleton extraction, they require initialization pose and constraint the user to face the sensor. For real-time skeleton extraction, this method requires a multi-GPU hardware with the output frame-rate mainly dependent on the number of persons appearing in the scene. The average frame rate that is achievable using two Nvidia GeForce GTX 1080 on full-HD Kinect V2 RGB image is approximately 14 fps. Since we currently employ only one GPU in our framework, we obtain 6 fps with 1 person in the scene.

4.2. Image Acquisition and Hand Localization Module

The strategy to localize human body and its sub-parts (i.e., hands or face) depends mainly on the output of the sensor of choice. In [31] the authors use skin color for hand segmentation using a conventional RGB camera, as in [32]. In [33], human body localization is performed using laser sensors, and its sub-parts are obtained through Kinect with the OpenNI library as in [34]. In [23], the authors localize the human body, inspired by [35], by merging clusters of the point cloud obtained from the Kinect V1 after voxel filtering and ground plane removal. Lately, infrared based sensors e.g., Leap Motion, are developed to track fingers of a hand in the near proximity (within 25 to 600 millimeters) of the sensor. However this range is too close for our application. In [36], authors adapt a state-of-the-art object detection deep learning technique namely YOLOv2, adapted to localize hands and head/face of a person in a scene. The authors have utilized OpenPose to first extract hands and face images from recorded videos with human activity, and then used these images to train YOLOv2 to detect hands and the face of the person in the scene in real-time. The face is detected to differentiate left hand from the right one. This is an efficient method to detect hands in the scene in real-time but requires a separate training/adaptation of YOLOv2 for hands and faces.

In our research, since we obtain the skeletal joint coordinates from OpenPose, we do not need to train a separate hand detector to localize hands. To estimate the hands position, we fit a line between the elbow joint and the wrist joint

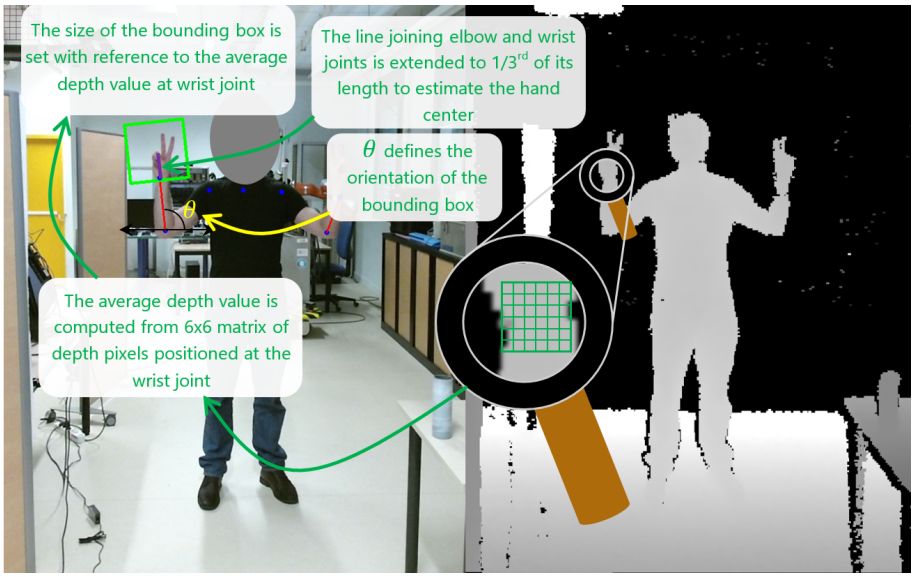


Figure 2: Localization of hand through OpenPose is illustrated. The bounding box is titled with an angle that the forearm makes with horizontal while the size of bounding box is determined by the mean depth value of the wrist joint. The mean depth value is computed by averaging the depth pixel values of a 6 × 6 matrix centered at the wrist joint.

returned by OpenPose and extend this line to one-third of its original length (which is an empirical value) in the direction of hand to approximately reach the center of hand. A bounding box is then centered at this approximated hand center at the angle which the forearm makes with the horizontal. This makes the hand image acquisition rotation invariant. The size of the bounding box is determined by the mean depth value of a 6 × 6 matrix centered at the wrist joint obtained through Kinect V2 depth map as shown in Fig. 2. The hand images are thus cropped with reference to the tilted bounding box, re-scaled to size 224×224 pixels and rotated again such that the cropped image becomes vertical.

4.3. Asynchronous Integration of the Modules

In our previous work [24], we integrated OpenPose with gesture recognition sequentially to obtain an overall temporal resolution of approximately 4 fps.

In this work, to ensure real-time performance, an inter-process distributed system is designed through the nanomsg socket library[1]. The said inter-process distributed system works using a "request-reply" communication pattern, also known as scalability protocol, to ensure that no frames are lost during communication. Figure 1 illustrates this asynchronous communication between the proposed framework modules via dotted lines. The image acquisition and hand localization module retrieves the image stream from Kinect V2 and checks if a frame request has arrived from the skeleton extraction module. When a frame request is received, the current RGB and depth image are first serialized through flatbuffers[2] and then passed to the skeleton extraction module. The skeleton extraction module unserializes the received frames with flatbuffers and then pass the RGB image through the forward-pass of OpenPose which returns a vector of 2D skeleton coordinates (x_i, y_i, c_i). The calculated mean depth values, as described in the previous section, are concatenated with the 2D skeleton coordinates and this 3D vector (x_i, y_i, d_i) is then sent to the image acquisition and hand localization module. The integration of Kinect V2 depth map with the 2D skeleton coordinates from OpenPose however do not represent the actual 3D coordinates of the joints and represent the surface depth value of the joints. There is a possibility that a joint is occluded in the scene by an object or the body itself. To prevent false detection of depth hence preventing potential accident, we use the confidence measure for each joint returned by OpenPose. The depth value of each joint is only updated if $c_i > 0.5$ (this is an empirical value), otherwise the previous depth value is kept. The image acquisition and hand localization module expects to receive coordinates from skeleton extraction module in each execution cycle. Once the coordinates are received, the hand is segmented and cropped image (as described in Section 4.2) runs through the forward-pass of trained convolutional neural network for hand gesture detection. The detected hand gesture label is sent to the robot controller running OpenPHRI to pilot the

[1] `https://nanomsg.org`

[2] `https://google.github.io/flatbuffers`.

experiment. The overall frame rate of our gesture detection pipeline is approximately 20 fps while the skeleton is extracted and the hand location is updated at around 5 fps. This improves the feasibility of pHRI experiments as compared to that in [24].

5. Convolutional Neural Network for Hand Gestures Detection

In recent years, the idea of deep learning has made a concrete impact on computer vision research and has been reported to even surpass human-level performance in image classification [37]. Hence, we chose to exploit convolutional neural network to recognize static hand gestures. Lately in [32], the author proposes a color-independent (using preprocessed binary hand images) hand gesture detector that relies on a convolutional neural network (CNN), inspired by LeNet [25]. The classification accuracy of such system depends largely on the preprocessing steps of image segmentation performed with color or intensity thresholding, while CNNs are inherently able to learn color features robustly as presented in [37]. In our research we aim to develop a robust hand gesture detector that should not require any preprocessing hence, hand tuning of parameters during detection phase. We take a step forward in not only using the color images for CNN training, but substituting the background of the training images with randomly chosen pattern/indoor-architecture images. This adds in the complexity of the learning problem but ensures a background invariant robust detector.

In our previous work [24], we designed the CNN architecture for hand images with relatively plain backgrounds, while the number of gestures were set to 4 and the gestures were recorded by a single person. In this research, 10 static hand gestures are recorded by 10 volunteers of age 22 to 35 (8 males and two females) and the backgrounds of the hand images are substituted as will be explained in Section 5.2. These features combine to make the recognition problem more complex as compared to the one presented in [24], where only 1 volunteer and 4 gestures had been considered. Therefore we opted for transfer

learning for gesture recognition, exploiting state-of-the-art CNNs pre-trained on large image data from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [38]. In particular, Inception-v3 [39] which is state-of-the-art in image classification for 1000 classes, is adapted for our background-independent hand-gesture recognition task. Inception-v3 is available in Keras [40] python library with pre-trained weights.

Figure 3: Samples of the gestures considered for training. The labels represent the letters and the numbers taken from American Sign Language. The last gesture is one of the several **None** gestures included in the training set.

Figure. 3 shows samples of the static gestures we train our framework on. The gestures include 9 letters/numbers in total taken from American Sign Language [41] and a **None** gesture that is not one of the 9 selected gestures. The letters/numbers are chosen such that they resemble with each other (like F, 7 and W; A, L and Y) so as to challenge the training and ensuring robustness of the CNN.

5.1. Preparation of Dataset/Dataset recordings

To create a dataset for gesture recognition and off-line development, RGB and depth image streams from Kinect V2 are saved in the local workstation. The frames are saved with an approximate frame rate of 20 fps. Each gesture is recorded by each volunteer for around 12 seconds with their both hands (see Fig. 4), thrice at distances of 5, 3 and 1.5 meters away from the sensor.

Figure 4: A volunteer recording '7' gesture in the laboratory

The depth information near Kinect V2 is rich and accurate, thus the images recorded at the distance of 1.5 meters are used for the fine-tuning of Inception-V3 as will be discussed in Section 5.3. However, since the network is trained only on RGB images, the hand gestures can also be recognized at other distances. We are releasing our dataset OpenSign[3] online that contains RGB and depth (registered) frames of volunteers recording 10 gestures. The RGB images are saved in *png* format while the float data of the depth images are saved in *bin* files. The total number of images used from our data-set is 20950, and we divide them with a ratio of 3:1:1 giving 12570 train images and the number of cross validation and test images equal to 4190 each. Train images go through extensive preprocessing as will be explained in Section 5.2, while only selective preprocessing operations are applied to cross-validation images to keep the them near to those obtained during recognition phase of the robotic interaction experiments.

5.2. Background substitution and Preprocessing of the Hand Images

Background substitution is performed so the network is trained to detect hand gestures independently from the background. We used nearly 1100 images

[3] `http://bit.do/OpenSign`

of random pattern and indoor architectures which are freely available on the internet[4]. The background substitution process is illustrated in Fig. 5.

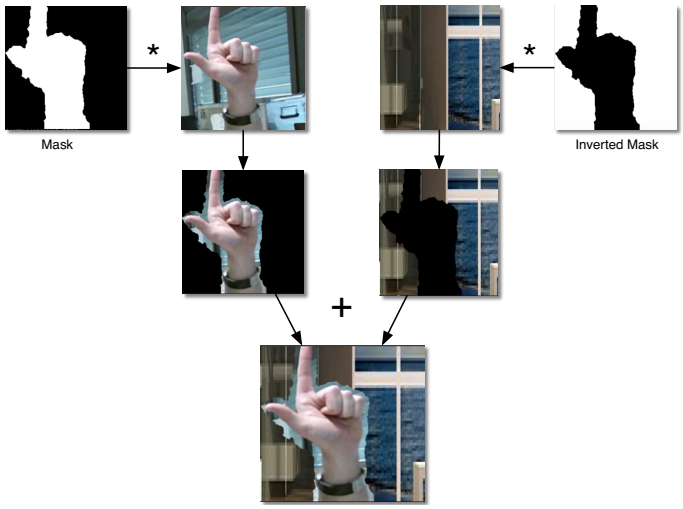


Figure 5: The process of background substitution.

A binary mask for background substitution is created using the depth information from Kinect V2. All the pixels that lie at distance within ±18 % (empirical value) of the mean depth value computed at the wrist joint (obtained through OpenPose) are set to 1, while the rest are zeroed. This binary mask is broadcasted into three channels and then multiplied by the cropped RGB hand image to get a background subtracted hand. An inverted mask is also created by simply applying a "NOT" operation on the mask originally computed. The background pattern images are cropped to multiple 224×224 sized images (as it is the set size of hand images) which are subsequently multiplied by the inverted hand mask. The hand image with subtracted background and the pattern images multiplied with the inverted binary mask are then added in the final step of background substitution. Figure 6 shows the samples of gestures with original and substituted backgrounds.

As discussed in Section 5.1, the training images go through several prepro-

[4] `https://pixabay.com/`

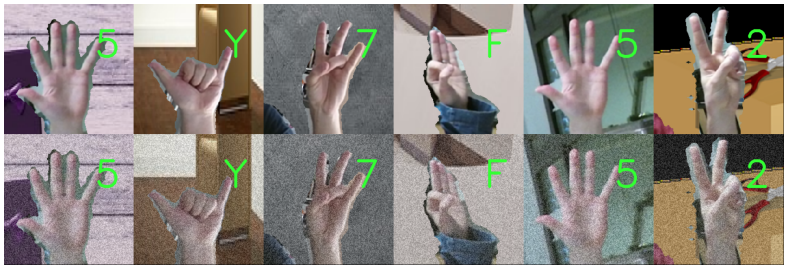
cessing steps. Image processing operations of histogram equalization and introduction of Gaussian and salt and pepper noise are applied on 30% of training images each while the remaining 10% are left unprocessed.

Figure 6: Samples of hand gesture images with original (labeled images) and substituted backgrounds (below originals).

Figure 7 shows random samples of original and processed images after the addition of Gaussian noise and histogram equalization. For robust gesture detection, we also use the real-time data augmentation feature of Keras library. Keras real-time data augmentation is designed to be iterated by the model fitting process, creating augmented image data in defined batch size during training.

This reduces the memory overhead of the computer but adds additional time cost during model training.

(a) Samples of training images after histogram equalization



(b) Samples of training images after the introduction of Gaussian noise

(c) Samples of training images after the introduction of salt and pepper noise.

Figure 7: Image processing operations of histogram equalization, introduction of Gaussian and salt and pepper noise are performed on the training images. First row in each sub-image shows unprocessed image while the processed images are shown in the second rows.

335 The image processing operations that are applied on the training images using the Keras library include channel shift, zoom, shearing, rotation, axes flip and position shift. Samples of processed training images with Keras being passed to the CNN are shown in Fig. 8.

Figure 8: Image processing operations applied to the training images include color-shift, zoom, shear, rotation, axes flip and position shift processes.

5.3. Adapting Inception V3 to gesture recognition

In image classification problems, the input data i.e., an image, is formed by low-level edges, curves and color combinations irrespective of the type of object that the image represents. It is therefore assumed that the early layers in the pre-trained state-of-the-art networks have learned to efficiently extract those features from the images thus they need to be preserved. Inception V3 is trained to recognize 1000 classes of objects as explained in Section 5. To adapt Inception V3 to classify only 10 gestures, the last softmax activation layer of this network with 1000 neurons should be replaced with a new layer of 10 neurons. As implemented in Keras, the Inception V3 has 10 trainable inception blocks. We perform training in three phases. In the first phase all the layers (hence inception blocks) in the network are frozen with the exception of the new layer added and the CNN is trained for 10 epochs only. This fine-tune the weights of the new layer exploiting the knowledge of all pre-trained inception blocks. Then we unfreeze last two inception blocks and train the CNN for 10 epochs, and then we train top four inception blocks so the network is fine-tuned properly

on our dataset. This gradual unfreezing of inception blocks prevents damaging the pre-trained weights and thus avert over-fitting.

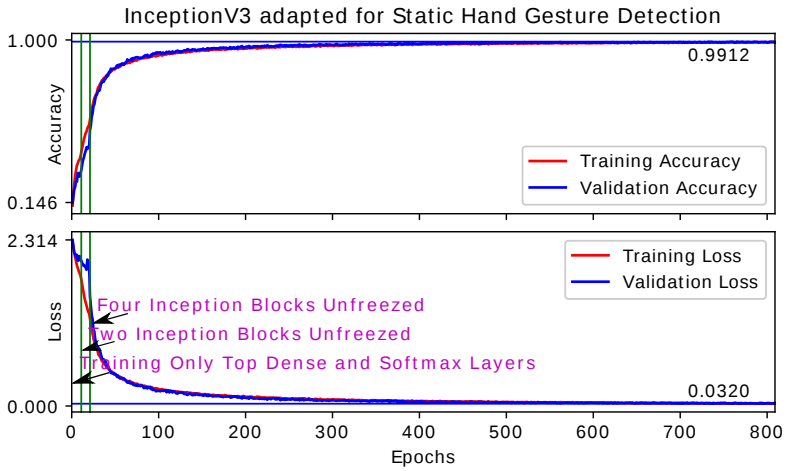


Figure 9: Plot of validation accuracy (top) and validation loss (bottom)

The validation set is used to chose the best performing weights and then the network is tested on the unseen test set to quantify/estimate the accuracy of the selected weights. Figure 9 illustrates the training curve of validation accuracy and loss of our dataset. Each epoch took approximately 130 seconds to pass and the network was able to achieve validation accuracy of 99.12% at 745^{th} epoch taking around 27 hours of training.

5.4. Quantification of the Trained CNN

To validate and quantify the results even further, the accuracy of the trained CNN is tested with a test set of 4190 new images. The overall test accuracy of the trained CNN is found to be 98.9% on test set. The normalized confusion matrix in Fig. 10 shows the accuracy of each gestures and misinterpretation of one gesture against the others. It can be observed that despite 94.3% accuracy of the **None** gesture, it was misinterpreted the most among all. The reason for this lower accuracy is that the **None** gesture defines all gestures that do

not appear like the other 9 as well as all transitional gestures. It is difficult to include all the transitional gesture possible to be classified as **None** gesture.

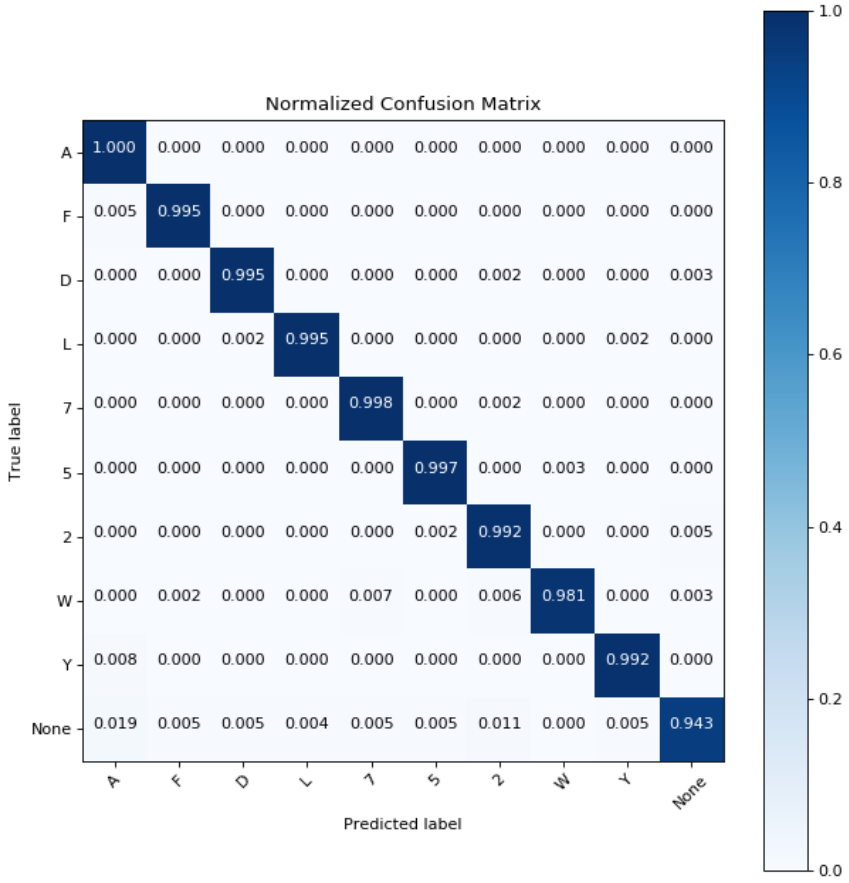


Figure 10: Normalized Confusion Matrix Quantified on Test-Set

Moreover, it can be observed from a close inspection of the test results that the CNN is very accurate in identifying a gesture as **None** when a person is holding an object in his hand. It is inferred that if the CNN is additionally trained on a gesture like "an object in hand", this gesture will be easily distinguished. Meanwhile, this misinterpretation can be dealt by adding a software constraint of not invoking gesture detector until the arm is in the upper two quadrants of the axes centered at the elbow joint of the person, as we did in

[24]. But this requires the user to be instructed on such constraints.

6. Example Industrial Application of The Proposed Framework

To demonstrate the effectiveness of the proposed approached, we set up an industrial-like experiment where multiple operators can safely interact sequentially with the robot using both hand gestures and physical contact. The experiment is decomposed into two phases: 1) a teaching by demonstration phase, where the user manually guides the robot to a set of waypoints and 2) a replay phase, where the robot autonomously goes to every recorded waypoint to perform a given task, here force control.

The BAZAR robot used for the experiments is composed of two Kuka LWR 4+ arms with two Shadow Dexterous Hands attached at the end-effectors. The arms are attached to a Neobotix MP700 omnidirectional mobile platform. In our scenario, the mobile base is kept fixed and only the left arm, without the hand, is used. The communication with the embedded arm controller is done using the FRI library. The external force applied to the arm's end-effector is estimated by the embedded controller (based on joint torque sensing and on knowledge of the robot's dynamic model) and retrieved using FRI. The control rate is set to 5ms. Figure 11 shows the setup, consisting of the BAZAR robot with a Kinect V2 mounted on top of it. To control the robot and to remain safe during human-robot collaboration, we have used the OpenPHRI [8] open-source control library. This library allows to describe the task to perform using force and velocity inputs in both the joint and task spaces while enforcing safety constraints such as velocity limitations, separation distance monitoring or emergency stops.

To orchestrate the experiment, we have designed a finite state machine (FSM), depicted in Figure 12. The transitions between the states are either automatic (no text), depending on sensory information (arrow with text) or triggered by gestures (hand sign with text).

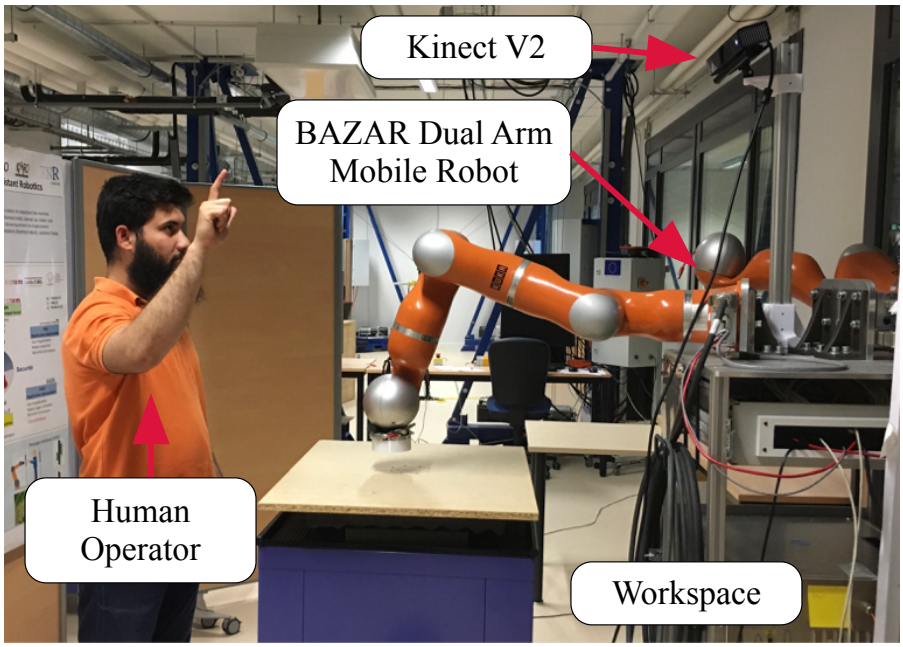


Figure 11: Safe Physical Human Robot Interaction Setup

A video of the experiment is available online[5] and snapshots are given in Figure 13. The experiment goes as follow. First, the robot goes to a predefined initial joint configuration before initializing the *Teach* phase. Once this initialization is performed, the robot is ready to be manually guided and taught the waypoints where the tasks have to be performed during the *Replay* phase. Each time a **Record** gesture (L letter sign) is detected, the current end-effector pose is recorded. When a **Replay** gesture (A letter sign) comes in, the *Teach* phase is ended and the *Replay* phase is initialized. Then, the robot goes to the first recorded waypoint while limiting its velocity thus ensuring safety of the human worker (separation distance monitoring in the FSM) according to the distance of the closest detected body part. This distance is obtained by mapping the depth value given by the Kinect V2 at the 2D joint coordinates obtained from OpenPose as explained in Section 4.3. If the closest body part is occluded by

[5] `http://bit.do/rcim2018phri`

the robotic arm, the depth value (that will then correspond to the depth value of the robot itself) is discarded while the next closest body part visible in the scene is considered a reference for depth.

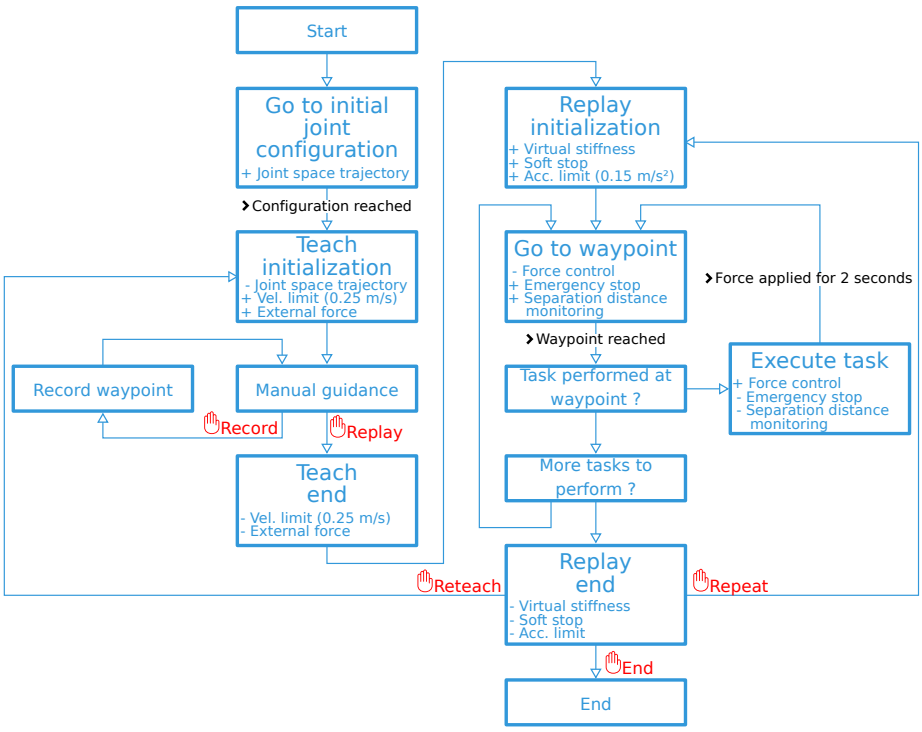


Figure 12: The FSM used for the experiment. A plus sign indicates an addition to the controller (a new constraint or new input) while a minus indicates a removal.

This distance estimation of body parts is not possible with the default output of OpenPose but a feature designed through the integration of Kinect V2 depth map. This amplifies the productivity of OpenPose skeleton extraction while assuring a safe interaction of a human coworker with the robot. While in autonomous motion, the robot can be stopped at any time (Soft Stop constraint in the FSM) using a **Stop** gesture (number 5 sign). Making this gesture will slow down the robot until a full stop is reached. This is useful if an operator must enter the robot workspace without fearing any injury. The **Resume** gesture (Y letter sign) can be made to resume normal operation. When the robot

reaches the waypoint, it switches to the task execution. In this scenario the task is to apply a 30N force for 2s along the vertical axis.

Figure 13: Screenshots from the robotic experiment by operators Op1 and Op2 (a) Op1 manually guiding the robot to a waypoint in the workspace. (b) Op1 records the way-points using Record gesture. (c) Op1 replay the taught waypoints by Replay gesture. (d) Op2 stands far from the robot so it moves with full speed. (e) Op2 stops the robot by applying external force (or accidental touch). (f) Op2 stands near the robot, so it moves slowly ensuring operator's safety. (g) Op2 gives Reteach command to the robot. (h) Op2 sets the new waypoints manually. (i) Op2 gives Record command. (j) Op2 stops the robot by Stop gesture. (k) Op2 resumes the robot operation by Resume gesture. (l) Op1 ends the robot operation by giving End command.

Once the task has been executed, the robot goes back to its waypoint and moves to the next ones to repeat the same operations. If the task has been performed at all the waypoints, the *Replay* phase ends and the next action is determined by the operator. A **Reteach** gesture (number 7 sign) will move the FSM to the *Teach* phase while a **Repeat** gesture (F letter sign) will repeat all

the tasks at the recorded waypoints. If no other operation is needed, an **End** gesture (number 2 sign) will end the experiment.

Experimental results are show in Fig. 14. The time axis has been limited to the 132-185s range for better readability. The top graph displays the result of the hand gesture detection where each vertical dashed line corresponds to the detection of a gesture. To filter out false positives, a gesture is considered valid if it appears in five consecutive frames.

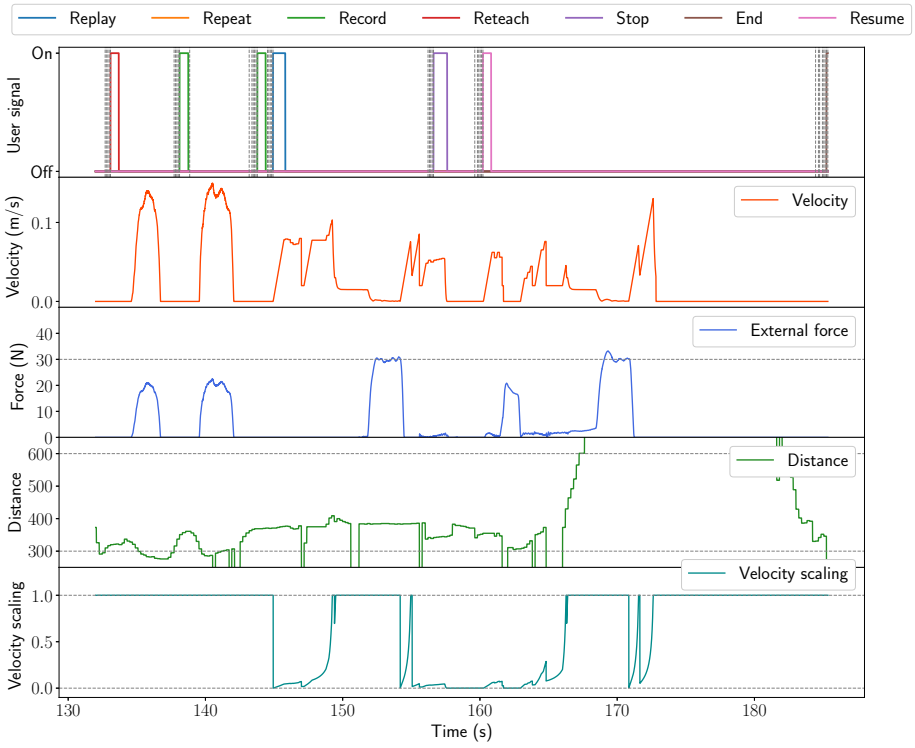


Figure 14: Experimental results. From top to bottom: hand gesture detection (dashed lines correspond to detection instants and plain line to the activation signals), control point translational velocity, external force at the end-effector, distance between the camera and the closest human body part and velocity scaling factor computed by OpenPHRI to slow down the motion.

Considering the hand-gesture detection frame rate of 20Hz, this gives a 250ms delay between the making of the gesture and its detection. This delay

should not impact human-robot interaction since the average human reaction time usually lies within the 200-250ms range[6]. Once the same gesture has been detected five times in a row, the corresponding signal is activated. False positives can be observed, e.g. at t=139 s when the first record signal ends, but thanks to the filtering systems no incorrect signal activation is made.

The two following graphs in Fig. 14 show the end-effector translational velocity and force. It can be seen that through the *Teach* phase, i.e. until t=135 s, the velocity simply follows the force applied to the robot. Then, the *Replay* phase starts and the end-effector velocity is now the result of the motion made to reach the waypoints and also by the force regulation applied at these locations. Between the two task executions (t=153 s and t=170s), one can observe some force applied to the robot at t=162 s. A safety feature is programmed to prevent accidents due to unexpected contact between the operator and the robot, leading to an emergency stop. In this situation, the robot stays still until the contact disappear and then resumes its motion to the second waypoints.

The fourth graph displays the distance to the closest body part. The values are the raw ones provided by the Kinect V2 and are unitless. As mentioned previously, this distance is used to adapt the velocity limitation so that the robot can move quickly when nobody is around but slows down when an operator is approaching. The velocity limit is at a minimum of $0.02m/s$ at a distance of 300 and at a maximum of $0.3m/s$ at a distance of 600. The effect of this limitation can be observed multiple times, including after the beginning of the *Replay* phase where the distance suddenly drops below 300, enforcing a very slow motion of the robot.

The last graph shows the evolution of the scaling factor computed by OpenPHRI. A value equals to one means that no velocity reduction has to be performed to comply with the constraints (velocity and acceleration limits, separation distance monitoring and emergency or soft stop). When at least one constraint would not be respected considering the current inputs, the scaling

[6] `http://humanbenchmark.com/tests/reactiontime`

factor decreases below one to make sure that all constraints are satisfied. When the value reaches zero, the robot is at a complete stop. Using this technique allows to easily slow down the robot only when it is necessary.

7. Conclusion

In the perspective of smart factories – also known as factories of the future – we have introduced a real-time human-robot interaction framework for robot teaching using hand gestures. The framework relies on our novel rotation and background invariant robust hand gesture detector. This detector adapts a pre-trained state-of-the-art convolutional neural network, namely Inception V3, to the classification of 10 hand gestures. The CNN is trained on an image dataset of 10 hand gestures, recorded with the help of 10 volunteers. The dataset OpenSign, is open and available to the computer vision community for benchmarking.

On each image, OpenPose and Kinect V2 are integrated to extract 3D data of the human skeleton. From these data, we can localize the image regions containing the hands, and crop them from the rest of the image. This integration is also essential to ensure the safety of the human coworker in human-robot interaction. We perform background substitution and image processing operations (e.g., histogram equalization, introduction of salt and pepper noise etc.) on the cropped hand images to increase variance in the overall data before training the CNN. This allows the network to learn robust hand features, so that no time-consuming rigid image processing methods are required during the recognition phase. The accuracy of the trained CNN is validated with a set of test images and is found to be 98.9%. To reaffirm the quality of the hand gesture detector and to validate it on a mock-up example industrial scenario, we perform a robotic experiment. Safety and effectiveness of the experiment are guaranteed by our physical human-robot interaction library, OpenPHRI. Besides, real-time operation is established by asynchronous integration of the different modules present in our framework. The experiment proves the efficiency of the proposed

framework, that ensures an intuitive means for robot programming. The robot is also aware of its distance from the human worker thanks to the integration of Kinect V2 and OpenPose. To guarantee the safety of the human coworker when s/he is in the close vicinity, the robot slows down using the velocity scaling feature of OpenPHRI.

Despite the quantified accuracy and experimental results, the capabilities of our system are limited by the depth range of the vision sensor. Moreover, the system is trained and tested in indoor settings and may fail in bright light due to the resulting contrast in RGB images. Backgrounds with intense texture may also compromise detection. To handle this, distinct background images should be substituted in the hand images to train the proposed network. Nevertheless, we believe that the preliminary results presented in this paper are a very promising step towards the development of vision-based intuitive robot programming. We encourage researchers interested in these topics to profit from our open image dataset for benchmarking their algorithms, and to enrich the dataset with more images.

8. Acknowledgements

Osama Mazhar's PhD is sponsored by a French public scholarship from the Doctoral School of Université de Montpellier.

References

[1] D. Gorecky, M. Schmitt, M. Loskyll, D. Zhlke, Human-machine-interaction in the industry 4.0 era, in: IEEE Int. Conf. on Industrial Informatics, 2014, pp. 289–294.

[2] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, J. Alcazar, Gestures for Industry: Intuitive Human-Robot Communication from Human Observation, in: Proceedings of the 8th ACM/IEEE Int. Conf. on Human-robot Interaction, HRI '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 349–356.

[3] I. Maurtua, A. Ibarguren, J. Kildal, L. Susperregi, B. Sierra, Human-robot collaboration in industrial applications: Safety, interaction and trust, International Journal of Advanced Robotic Systems vol. 14 (4).

[4] S. S. Rautaray, A. Agrawal, Vision based hand gesture recognition for human computer interaction: A survey, Artificial Intelligence Review 43 (1) (2015) pp. 1–54.

[5] P. Neto, J. N. Pires, A. P. Moreira, High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition, Industrial Robot 37 (2010) pp 137–147.

[6] P. Neto, D. Pereira, J. N. Pires, A. P. Moreira, Gesture Recognition for Human-Robot Collaboration: A Review, in: Proceedings of the 7th Swedish Production Symposium, 2016, pp. 1–12.

[7] J. Ruiz-del-Solar, P. Loncomilla, N. Soto, A Survey on Deep Learning Methods for Robot Vision, CoRR abs/1803.10862 (2018).

[8] B. Navarro, A. Fonte, P. Fraisse, G. Poisson, C. Andrea, In pursuit of safety: An Open-Source library for Physical Human-Robot Interaction, IEEE Robotics Automation Magazine, 2018.

[9] X. V. Wang, Z. Kemény, J. Váncza, L. Wang, Human-robot collaborative assembly in cyber-physical production: Classification framework and implementation, CIRP Annals, Manufacturing Technology vol. 66 (1) (2017) pp. 5–8.

[10] P. Neto, D. Pereira, J. N. Pires, A. P. Moreira, Real-time and continuous hand gesture spotting: An approach based on artificial neural networks, in: IEEE Int. Conf. on Robotics and Automation, 2013, pp. 178–183.

[11] S. Makris, P. Tsarouchi, D. Surdilovic, J. Krüger, Intuitive dual arm robot programming for assembly operations, CIRP Annals, Manufacturing Technology vol. 63 (1) (2014) pp. 13–16.

[12] Z. Zhang, Microsoft Kinect Sensor and Its Effect, IEEE MultiMedia vol. 19 (2) (2012) pp. 4–10.

[13] Y. Yang, H. Yan, M. Dehghan, M. H. Ang, Real-time human-robot interaction in complex environment using kinect v2 image recognition, in: IEEE International Conference on Cybernetics and Intelligent Systems and IEEE Conference on Robotics, Automation and Mechatronics, 2015, pp. 112–117.

[14] M. G. Carmichael, D. Liu, A. Tran, R. Khonasty, S. Aldini, Robot Coworker for Abrasive Blasting: Lessons Learnt in Worker Posture Estimation, in: IEEE Int. Conf. on Robotics and Automation, 2018.

[15] J. Sell, P. O'Connor, The Xbox One System on a Chip and Kinect Sensor, IEEE Micro vol. 34 (2) (2014) pp. 44–53.

[16] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, A. Bhowmik, Intel(R) RealSense(TM) Stereoscopic Depth Cameras, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 1267–1276.

[17] G. Du, M. Chen, C. Liu, B. Zhang, P. Zhang, Online Robot Teaching with Natural Human-robot Interaction, IEEE Transactions on Industrial Electronics, 2018.

[18] E. Matsas, G.-C. Vosniakos, Design of a virtual reality training system for human–robot collaboration in manufacturing tasks, International Journal on Interactive Design and Manufacturing vol. 11 (2) (2017) pp. 139–153.

[19] M. Simo, P. Neto, O. Gibaru, Natural control of an industrial robot using hand gesture recognition with neural networks, in: 42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, pp. 5322–5327.

[20] G. Canal, S. Escalera, C. Angulo, A real-time human-robot interaction system based on gestures for assistive scenarios, Computer Vision and Image Understanding 149 (2016) pp. 65–77.

[21] J. L. Raheja, M. Chandra, A. Chaudhary, 3D Gesture based Real-time Object Selection and Recognition, Pattern Recognition Letters, 2017.

[22] Y. LeCun, F. J. Huang, L. Bottou, Learning methods for generic object recognition with invariance to pose and lighting, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, 2004, pp. 97–104.

[23] K. Ehlers, K. Brama, A human-robot interaction interface for mobile and stationary robots based on real-time 3D human body and hand-finger pose estimation, in: IEEE Int. Conf. on Emerging Technologies and Factory Automation, 2016, pp. 1–6.

[24] O. Mazhar, S. Ramdani, B. Navarro, R. Passama, A. Cherubini, Towards Real-time Physical Human-Robot Interaction using Skeleton Information and Hand Gestures, in: (Submitted to) IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018.

[25] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE Vol. 86 (11) (1998) pp. 2278–2324.

[26] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[27] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional Pose Machines, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[28] S. Das, M. Koperski, F. Bremond, G. Francesca, Action recognition based on a mixture of RGB and depth based skeleton, in: 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2017, pp. 1–6.

[29] C. Zimmermann, T. Welschehold, C. Dornhege, T. Brox, W. Burgard, 3D Human Pose Estimation in RGBD Images for Robotic Task Learning, in: IEEE Int. Conf. on Robotics and Automation, 2018.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common Objects in Context, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 740–755.

[31] R. C. Luo, Y. C. Wu, Hand Gesture Recognition for Human-Robot Interaction for Service Robot, in: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2012, pp. 318–323.

[32] P. Xu, A Real-time Hand Gesture Recognition and Human-Computer Interaction System, CoRR abs/1704.07296 (2017).

[33] M. V. den Bergh, D. Carton, R. D. Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. V. Gool, M. Buss, Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans, in: RO-MAN, 2011, pp. 357–362.

[34] G. Cicirelli, C. Attolico, C. Guaragnella, T. D'Orazio, A Kinect-based Gesture Recognition Approach for a Natural Human Robot Interface, International Journal of Advanced Robotic Systems Vol. 12 (3) (2015) pp. 22.

[35] M. Munaro, F. Basso, E. Menegatti, Tracking people within groups with RGB-D data, IEEE/RSJ International Conference on Intelligent Robots and Systems (2012) 2101–2107.

[36] P. Panteleris, I. Oikonomidis, A. A. Argyros, Using a Single RGB Frame for Real Time 3D Hand Pose Estimation in the Wild, in: IEEE Winter Conference on Applications of Computer Vision, IEEE, 2018, pp. 436–445.

[37] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV) Vol. 115 (3) (2015) pp. 211–252.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[40] F. Chollet, et al., Keras, `https://keras.io` (2015).

[41] R. Battison, Lexical Borrowing in American Sign Language.