Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length article

MuViH: Multi-View Hand gesture dataset and recognition pipeline for human–robot interaction in a collaborative robotic finishing platform

Corentin Hubert [a,1], Nathan Odic [a,1], Marie Noel [a], Sidney Gharib [a], Seyedhossein H.H. Zargarbashi [b], Lama Séoud [a,*]

[a] *Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Montréal, Québec, Canada*
[b] *Aerospace Research Center, National Research Council (NRC), Montréal, Québec, Canada*

ARTICLE INFO

Dataset link: https://doi.org/10.5683/SP3/JZJTGG

Keywords:
Vision-based human–robot interaction
Gesture recognition
Hand detection
Occlusion
Multi-view
Dataset

ABSTRACT

The proliferation of tedious and repetitive tasks on production lines has accelerated the deployment of automated robots. This has also led to a demand for more flexible robots, known as cobots, that can work in collaboration with operators to perform a variety of tasks in different contexts. This paper explores the potential of computer vision-based hand gesture recognition as a means of human–robot interaction within cobotic platforms. Our research focuses on the challenges of gesture recognition in the face of visual occlusions and different camera viewpoints, typical of part finishing tasks in a real-world industrial setting. We introduce a new dataset, MuViH (Multi-View Hand gesture), which features a high variability in camera viewpoints, human operator characteristics, and occlusions, and is fully annotated for hand detection and gesture recognition. We then present a comprehensive hand gesture recognition pipeline that leverages this dataset. Our pipeline incorporates a multi-view aggregation step that significantly enhances gesture recognition accuracy, particularly in the case of visual occlusions. Thanks to extensive experiments and cross-validation on the MuViH dataset and another public dataset, HANDS, our approach demonstrates state-of-the-art performance in gesture recognition. This breakthrough underlines the potential of integrating robust vision-based interaction techniques into cobotic systems, improving flexibility and speed on the production line.

1. Introduction

Advanced manufacturing has brought new automation technologies, including robots, which offer the considerable advantage of executing tasks that are tedious and repetitive, and that pose a high risk of injury for personnel on the production line. While industries generally use pre-programmed robots with rigid codes that are not designed to cooperate with humans, some manufacturing processes such as polishing and deburring of metallic parts, used for finishing, require proper parameter settings, such as the geometry of the part to be machined, the tool required and the tool trajectory. Interactive cyber–physical systems offer an elegant solution, giving the human operator essentially more elevated roles of inspection and supervision. The concept of human–robot collaboration does not focus on replacing humans with robots but rather on collaboration between them in a common workspace: the cobotic platform.

The collaboration is directed and coordinated by the human operator providing commands to be executed by the cobot. Possible communication channels are varied, it can be either traditional, verbal, gestural, or through brain signal analysis [1,2]. Traditional approaches include mouse, keyboard and tactile screens. Despite its robustness, it requires a fixed operating space. Verbal communication can be difficult to implement on production lines where noise is highly prevalent. Gesture-based communication offers strong intuition and high flexibility, it is a natural way to convey information. Hand gesture recognition has been explored in previous work for human–robot interaction in the context of manufacturing [3]. Its accuracy, however, is closely related to the sensing technology and algorithm used [4].

Sensing technologies can be divided into wearable and vision-based devices. Having workers wear additional equipment dedicated solely for communication with the cobot, such as gloves equipped with inertial measurement units or a connected watch, is generally to be ruled out, as such equipment creates a constraint for the worker [5]. Vision-based solutions, by contrast, extract the spatial information required to recognize an operator's action, such as a pointing gesture [6,7].

Vision-based hand gesture recognition essentially relies on acquiring images of the operator, detecting the hands and recognizing the

* Corresponding author.
E-mail address: lama.seoud@polymtl.ca (L. Séoud).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.rcim.2025.102957
Received 18 June 2024; Received in revised form 23 December 2024; Accepted 2 January 2025
Available online 22 January 2025

gesture. The latter two steps are nowadays tackled using deep learning, requiring large annotated datasets to train models for each task.

Among the remaining challenges of vision-based hand gesture recognition is the sensitivity to visual occlusions and viewpoint of the camera in relation to the operator's position. In the particular context of a cobotic platform, occlusions are frequent since both the cobot and the operator are assumed to move. Moreover, as we do not want to restrict the operating space, we aim to offer flexibility for the operator to communicate with the cobot from anywhere around the platform.

The general objective of this paper is therefore to develop a static hand gesture recognition pipeline that is robust to visual occlusions and to different viewpoints. Our contributions are:

- A new hand gesture recognition dataset, named MuViH (Multi-View Hand), with high variability in terms of camera viewpoints, human characteristics and occlusions. It contains more than 85,000 images captured by 6 RGB-D cameras positioned at different locations around the cobotic platform. It comprises 20 participants (13 male, 7 female) performing 10 gestures with their hands. The background is highly cluttered. The placement of the cobot is modified on several occasions during data collection, creating various uncontrolled visual occlusions. The dataset is fully annotated for hand detection and static gesture recognition. It is made available for academic purpose upon request (https://doi.org/10.5683/SP3/JZJTGG).
- A complete hand gesture recognition pipeline, evaluated by cross-validation on two datasets (the proposed MuViH and the publicly available HANDS [8]), showing state-of-the-art performance in single-view mode for hand detection and gesture recognition. The pipeline comprises a fine-tuned YOLOv8 hand detection model, and a gesture classification model based on ResNet, to recognize gestures among the 10 present in the dataset.
- A multi-view aggregation step that improves by 14% the recognition accuracy in comparison to single-view, more precisely in cases of visual occlusions.

2. Related works

2.1. Hand gesture recognition dataset

The state of the art of datasets for hand gesture recognition covers a wide range of scenarios, in both controlled and more complex environments. There are several datasets for static gesture recognition created for sign language translation [9,10]. In the field of manufacturing, the HANDS dataset [8] has been used for gesture-based human–robot interaction [6]. More recently, a richer dataset for static gesture recognition, named HaGRID, has been published [11]. Nevertheless, in all these RGB-D datasets, the human is ideally placed facing the camera resulting in limited variability in terms of viewpoint. The EgoGesture [12] dataset, captured using a head-mounted Intel Realsense SR300, provides a wide variety of scenes and gestures, but the camera-to-hand distance is very short (≤ 1 m) and visual occlusions are absent. In the HGM-4 dataset [13], four different angles of view are provided for hand gesture recognition, but images have undergone background removal. This absence of background makes it difficult for a hand detection or gesture recognition model trained on this data to generalize to real scenarios in cluttered environments.

There is thus a need for a new public dataset for static hand gesture recognition with sufficient images, variability in terms of viewpoints, human characteristics and camera-to-hand distances, visual occlusions and realistic scenarios.

2.2. Hand detection

Numerous neural network architectures have been developed for object recognition and tested on various datasets [14]. Early deep learning methods used CNNs, such as R-CNN [15], which first generates region proposals and then uses a CNN to extract features from these regions and classify them. Fast R-CNN [16] then improved on this proposal, applying the CNN to the entire image and not just to the proposed region. Faster R-CNN [17] improved the speed of this method by replacing Fast R-CNN's region proposal generation method with a Region Proposal Network (RPN). The disadvantage of these two-step methods is their inference time. This is why single-shot methods have been developed. YOLO [18] was the first of these methods, processing the image in a single pass through the network. The image is divided into a grid, and each grid cell is responsible for detecting objects in that region. SSD [19] improved this method by using feature maps of different resolutions to detect objects at different scales. Over the years, new YOLO versions [20] have been released, improving the efficiency of CNNs in particular. In contrast to the aforementioned CNN-based methods, DETR [21] reformulates object detection as an ensemble matching problem, using transformers to model the spatial relationships between objects in the image.

2.3. Hand gesture recognition

Predicting dynamic hand gestures, where motion carries semantic information, requires the use of neural networks that take into account the temporal scale. For this reason, the networks used in the literature are often recurrent neural networks (RNN) [22] or long short-term memory networks (LSTM) [23]. However, these networks do not receive images as input but rather hand pose estimation outputs [24].

Recognition of static gestures is a simpler problem than that of dynamic gestures, as the temporal component no longer needs to be taken into account upstream. It can, however, be exploited downstream to correct a model's output according to its predictions on neighboring images. This is why convolution neural networks (CNNs) [25] can be sufficient to solve the problem. Methods like MEGURU [6], using an R-FCN network [26], predict gestures solely on the basis of hand detection, showing that hand pose estimation is no longer a prerequisite.

3. Material

3.1. Cobotic platform

A collaborative robotic platform has been purposefully designed to serve as a dedicated testbed for cobotic industrial applications within the Aerospace Manufacturing Technologies Centre (AMTC) at the National Research Council of Canada (NRC) [27]. A schematic top view of the platform is shown in Fig. 2. It is composed of a UR10/CB3 cobot mounted upside down on a linear range extender, which is supported by a fixed gantry structure (hashed area in Fig. 2) and an ergonomically designed downdraft table facilitating part handling and clamping. The hashed area is inaccessible to the operator, as it contains, among other things, the cobot's power supply system and is not easy to navigate.

3.2. Optical sensors

A disadvantage of optical systems is the possibility for the visually monitored area to be obscured partially during the operation either because of the robot's movement or by the operator's activity. This creates visual occlusions resulting in a part of the workspace volume not being monitored. Hence, hand gestures might not always be visible by the camera. Minimizing these occlusions in the collaborative workspace can be achieved by increasing the number of cameras and by carefully choosing their arrangement [28]. Many affordable RGB-D sensors are

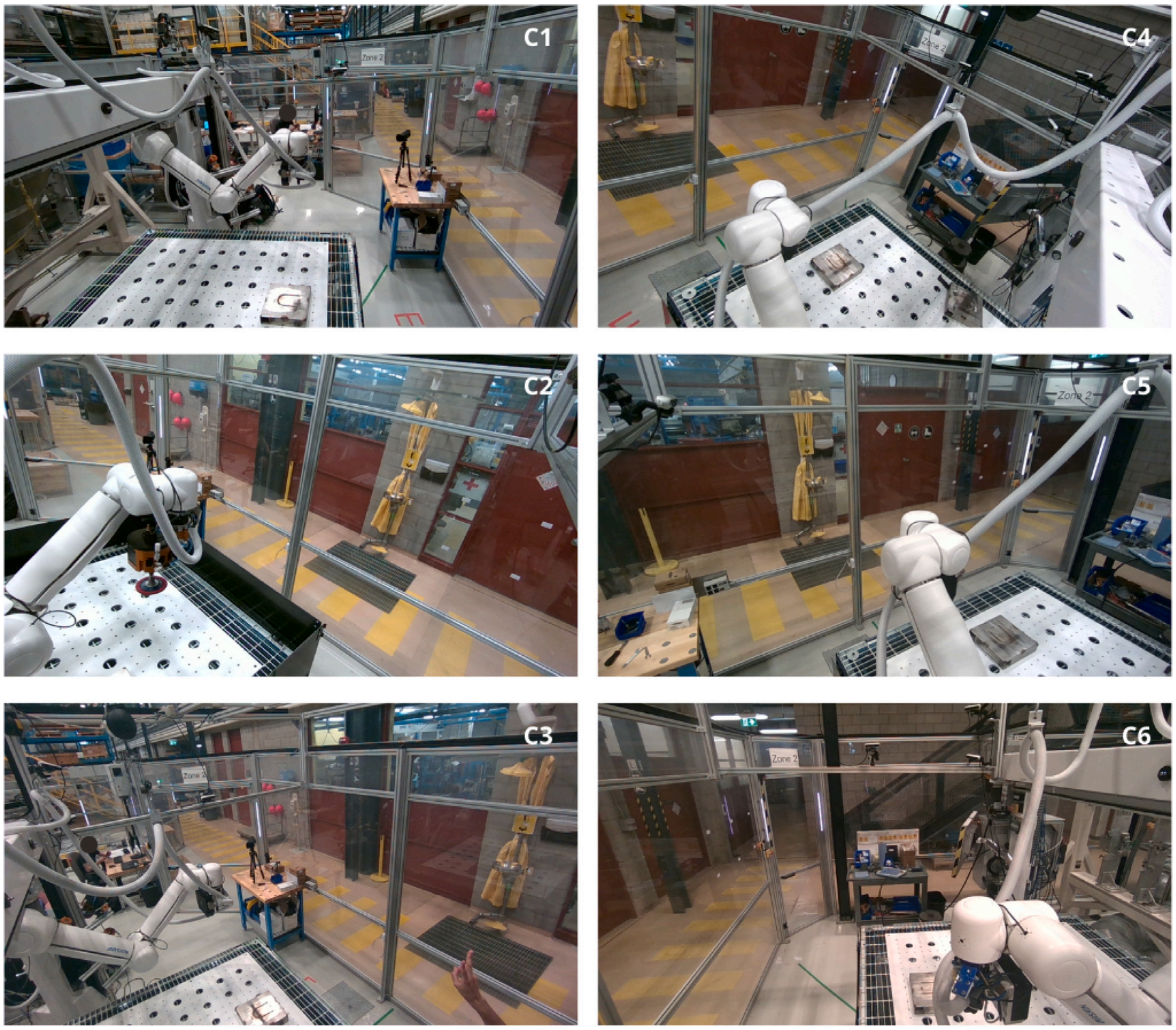


Fig. 1. Viewing angle of each of the six RGB-D cameras around the cobotic platform.

available on the market, like Microsoft's Kinect Azure, Orbec's Astra and Intel Realsense's D400 series. Each of these uses a different range sensing technique (respectively time-of-flight, structured-light and active stereo-vision) [29]. Active stereo offers one important advantage over the other two techniques: the projected light patterns do not need to be structured. Consequently, having multiple active stereo-based sensors running simultaneously only adds more resolution to the light pattern, resulting in better depthmaps. For this reason, we selected Intel RealSense D455 cameras for our platform. Camera placement was determined by the authors based on the following:

1. Available guidelines for human motion capture [30] recommend installing cameras so as to circumscribe the capture volume at a high elevation and maximize camera coverage and capture volume size,
2. Considering that in our cobotic platform (illustrated in Fig. 2), one side of the downdraft table is inaccessible to the operator, there was no need to place cameras on the opposite side,
3. Cameras could not be placed inside the robot workspace volume.

As illustrated in Fig. 2, the sensors are positioned at two different heights (2 m and 2.3 m from the ground for levels 1 and 2 respectively). Fig. 1 shows the viewpoints from each camera.

Four areas are defined around the table. Each area is covered by at least two cameras, two of which are at different heights. This increases the number of viewpoints for each area. Thus, areas 1, 2, 3 and 4 are covered by cameras (c4, c6), (c2, c4, c5), (c2, c3, c5) and (c1, c3) respectively.

3.3. Visual occlusions

During real-life operations of the cobotics platform, the cobot moves above the downdraft table to perform the manufacturing tasks and the operator can move freely around the table to supervise the operations. There are therefore spatial configurations where the operator can be partly hidden by the cobot, potentially hiding the hand(s) performing the gesture. To reflect this in the dataset, the cobot configuration was changed randomly at several points in time during the creation of the dataset, resulting in a variety of occlusions that we did not control for in a deterministic manner.

4. Method

4.1. Dataset creation pipeline

To ensure good generalization performance of hand detection and gesture recognition models, it is important to use a training set with high variability. The latter can be expressed in terms of camera configuration (viewing angle and distance from the operator), human characteristics (morphology, gender, skin color, manual preference), and environment (operator location in the platform, presence of occlusions, presence of more than one human).

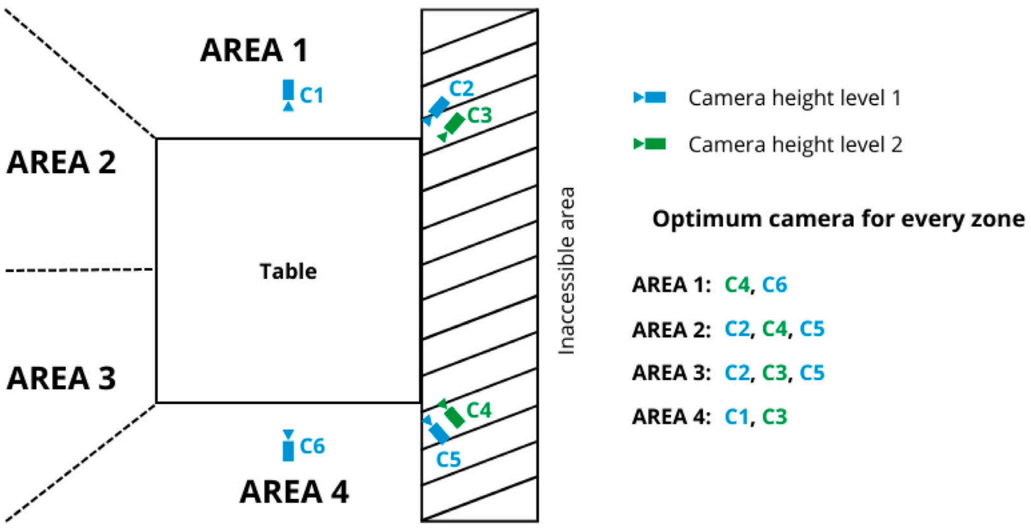


Fig. 2. Schematic top view of the cobotic platform with optical sensors layout.

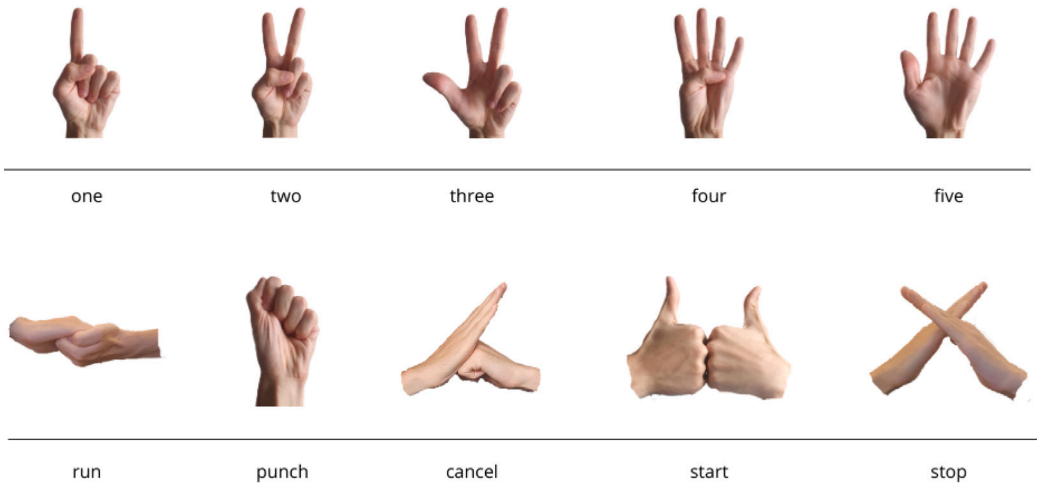


Fig. 3. Proposed gestures dictionary to communicate with the cobot.

The variability of viewpoints is made possible by the use of 6 RGB-D sensors simultaneously during data collection. In terms of human characteristics, we established that a total of 20 participants would be sufficient for this work, while, in comparison, similar public databases [8,13] involved only 5 participants each.

To communicate with the cobot, a dictionary containing 10 gestures was created (Fig. 3). It is desirable that the dictionary be easily adapted to different industries. The use of new tools within the cobotic platform must not compromise the established set of gestures. Otherwise, the addition of a new gesture would require re-training a recognition model with an additional class. This is why the gestures chosen for tools and parameters selection are simply numbers (`one` through `five`), and their meaning is generic, not specific to a particular industrial context. The remaining gestures (except for `punch`) all require both hands. The reason for using two-handed gestures is that there is little chance that an operator passing through the cobotic cell will make these gestures inadvertently. These gestures are therefore reserved for the most critical commands, such as starting communication (`start`), canceling previous gestures (`cancel`), starting a task (`run`) or stopping the robot while it is working (`stop`).

Each participant is asked to perform the 10 gestures illustrated in Fig. 3, some of which are performed twice: the one-handed gestures are performed both with the right and left hands. For the two-handed gestures, the symmetrical `start` is executed once. For the asymmetrical gestures, `cancel` is performed in both of the right/left hand configurations, whereas the participants perform `run` and `stop` without paying attention to hand sides. In all, there are 17 recording sequences (one per gesture), each lasting one minute. For each sequence, the participant is asked to move around in a single predetermined zone (Fig. 2), ensuring that the optimum camera set is fixed. All participants make at least four gestures per area, and a given gesture is performed in different areas by different participants. In this way, gestures (resp. participants) are equally performed (resp. present) in each area.

Natural language concepts and the requirement to maximize gesture contrast and recognition should guide the design and selection of 3D-modeled hand gestures for human–computer interaction [31]. Moreover, it is desirable that the dictionary be easily adapted to different industries. The addition of new tools or commands within the cobotic platform must not compromise the established set of gestures. Otherwise, the addition of a new gesture would require re-training a recognition model with an additional class. For these reasons, a dictionary containing 10 gestures was created (see Fig. 3). The gestures chosen for tools and parameters selection are simply a `punch` and numbers `one` through `five`; their meaning is generic, not specific to a particular industrial context. They can easily be combined to create a

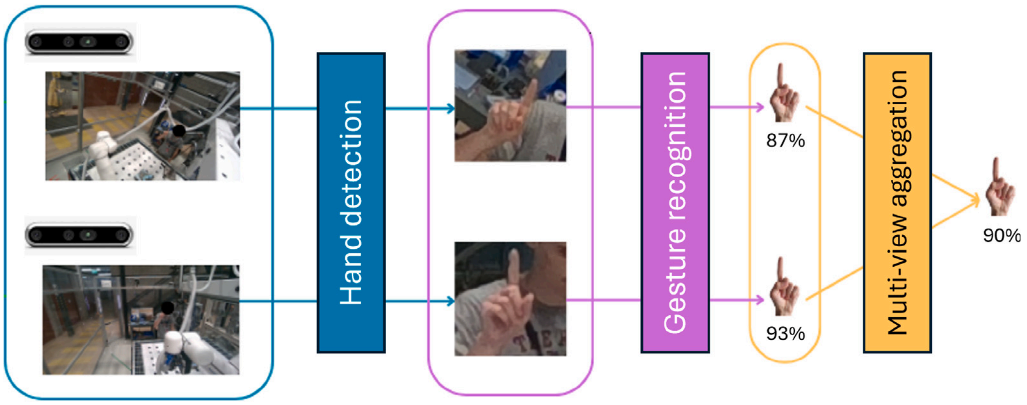


Fig. 4. Overall hand gesture recognition pipeline.

complete vocabulary for a multitude of actions. The remaining gestures all require both hands. The reason for using two-handed gestures is that there is little chance that an operator passing through the cobotic platform will make these gestures inadvertently. These gestures are therefore reserved for the most critical commands, such as starting communication (`start`), canceling previous gestures (`cancel`), starting a task (`run`) or stopping the robot while it is working (`stop`). The numbers (`one` through `five`) and the `punch` and `run` gestures are based on previous work [6] for an industrial collaborative workstation. They are included in the HANDS dataset [8], thus allowing for a straightforward comparison with the latter.

The one-minute duration of each sequence allows the participant to move around in the designated area, increasing the diversity of viewpoints and occlusions. The cobot spatial configuration was changed several times during the whole data collection to further increase variability in terms of occlusions.

For data collection, a script was written in C++. It records RGB, aligned depth images and automatically associated gestures and areas to the images. The frame rate was set to 30 FPS, but in practice this is never reached. The image resolution is 1280 × 720 pixels.

The position of the hands making the gestures, defined by bounding boxes, was annotated manually by a single operator for all captured images in which the hand making the gesture is actually visible (not occluded).

Since Intel RealSense cameras capture the scene asynchronously, image timestamps are recorded and used a posteriori to match images between cameras (fixing the maximum allowed duration $\Delta t = 100$ ms between timestamps). Once the images are matched, they form a group of synchronized images. We retain 100 groups of synchronized images per gesture, and each group has at least one image in which a hand is annotated.

The creation of this dataset has been approved by the Ethics Review Boards of Polytechnique Montréal and the NRC.

4.2. Hand gesture recognition pipeline

The pipeline described in the present work comprises three distinct parts, as shown in Fig. 4:

1. Hand detection in RGB-D images.
2. Gesture recognition based on the hands detected by each camera.
3. Combining gesture predictions from different cameras.

In the remainder of this paper, we assume that we only have images from well-positioned cameras (as shown in Fig. 2), thus the operator is always in the field of view.

4.3. Hand detection with RGB image

For the hand detection step, we considered several algorithms. We ruled out two-stage algorithms [15–17], as they are too slow and deliver too little performance compared with more recent architectures. We also chose not to use algorithms using transformers such as DETR [21] and its derivatives, because they are relatively slow and because their main advantage, bipartite matching, is not useful in our case, given that only a single gesture is detected in the image. The algorithms of the YOLO [18] family of models were therefore the best solution, for their speed and high performance in a variety of contexts. We used the latest version available, YOLOv8 [20], as it was the most powerful of the series. This architecture directly returns the coordinates of the bounding boxes surrounding the hand in each image. We therefore fine-tune the model on our data so that it only detects hands that are actually performing a gesture.

4.4. Gesture recognition with RGB-d image

Once a hand is detected in an image, the latter is cropped to the bounding box and fed into a gesture classification network. We propose a 10 class ResNet-inspired architecture [32], comprising a sequence of residual blocks with four convolutions each. This type of architecture has proven its robustness over time. In addition, it is relatively fast compared with more recent architectures, such as those based on transformers. One can easily modify this architecture by choosing the number of bottleneck blocks, to strike the desired balance between classification performance and inference time. In this work, we considered six residual blocks, each followed by a 2 × 2 pooling stage. For an input image of size 100 × 100, these blocks provide a feature map of size 1 × 1. This feature map is then fed into three fully connected layers to obtain a class probability vector. The final architecture is illustrated in the supplemental materials.

Since the network input size is fixed at 100 × 100 pixels, an image preprocessing step consisting of bilinear interpolation is necessary. However, to prevent image distortion, we fix the aspect ratio of the bounding box and use zero padding when necessary.

4.5. Multi-view aggregation

At this stage, we have as many probability vectors as there are well-positioned cameras at each instant. We now need to combine these different predictions to determine which gesture is actually performed. To this end, we propose to simply average the different probability vectors. The gesture with the highest probability after averaging is the one estimated to be performed. One of the advantages of this method is that it can be easily generalized: whatever the position of the cameras or their number, averaging the probabilities should always have the same impact on the results obtained.

5. Experimental setup

5.1. Implementation details

For both detection and recognition models, the graphics card used for training is a GeForce RTX 3070 with 8 GB of VRAM. To fine-tune the hand detection model, Adam optimizer is used with a learning rate equal to 10^{-3}, during 10 epochs and the batch size is 16 for an image size of 640×640 pixels. To train the gesture classification model from scratch, Adam optimizer is used with a learning rate equal to 10^{-4}, during 12 epochs with a batch size of 10.

5.2. Model evaluation and metrics

To quantitatively assess the performance of the proposed pipeline, a 5-fold cross-validation is conducted: first, on a public hand gesture dataset (HANDS [8]) for direct comparison with previous work [6], and second on the proposed MuViH dataset. For each fold, images of 80% of the participants are used for training, while the remaining 20% of participants are used for testing.

To evaluate the hand detection model, 4 metrics are used: mean average precision at a 50% IoU threshold (mAP_{50}), averaged precision for IoU thresholds between 50% and 95% ($mAP_{50:95}$), precision and recall at a 50% IoU threshold. Additionally, a false positive analysis is performed on the subset of images in which the hand making the gesture is occluded and thus not annotated.

To evaluate the gesture classification model separately, without taking hand detection into account, we use the bounding boxes from the reference annotations available in both the HANDS and MuViH datasets. From these annotations, we extract the hand ROI in the image, pre-process it, and predict the gesture with the trained model. The accuracy is calculated as the percentage of gestures that have been correctly predicted. A confusion matrix is also computed to report per-class accuracy.

To evaluate the complete hand gesture recognition pipeline (detection and classification), we use the bounding boxes predicted by the hand detection model as input to the gesture classification model and report the classification accuracy. Note that in this case, classification accuracy is computed in two ways: overall accuracy and accuracy on hands that are correctly detected only.

6. Results and discussion

6.1. Muvih dataset

The dataset contains a total of 85,000 images from 20 participants performing 17 gestures captured simultaneously by 6 cameras. Participant demographics are provided in Fig. 8 to show the variability in terms of operator morphology (sex and body mass index (BMI)) in the MuViH dataset. This variability allows for better generalization of our models.

For each gesture made by each participant, there are 100 groups of synchronous images from 2 or 3 cameras, i.e. 200 or 300 images depending on the area in which the gesture was made. As a result, there are 4,200 or 4,300 images per participant. Some sample images from the dataset are shown in Fig. 5. These examples show the complexity of the recognition task: cluttered background, visual occlusions and non-ideal viewpoints.

Fig. 6 provides an obvious example of visual occlusion, where no annotation of the hand is possible. Such cases account for exactly 6,236 images, which corresponds to less than 8% of the whole MuViH dataset.

Fig. 7 shows a few images from a recording sequence of a participant performing a gesture (`run`) while moving within a zone of the cobotic cell.

Table 1
Cross validation on HANDS dataset (metrics in %).

HANDS	Test subject					Average
	1	2	3	4	5	
			Hand Detection			
mAP_{50}	99.5	99.2	99.0	99.5	99.5	99.3
$mAP_{50:95}$	81.4	78.9	75.3	82.5	83.4	80.3
Precision	99.5	97.4	98.5	99.4	99.9	98.9
Recall	99.7	97.2	96.4	100	100	98.7
			Gesture Classification			
Accuracy	93.5	97.5	86.9	92.7	97.8	93.7
			Detection and Classification			
Accuracy	85.7	93.8	77.3	92.3	95.7	88.9

6.2. Pipeline evaluation on the HANDS dataset

The aim of this section is to present and discuss the results of the various pipeline elements, as well as the complete pipeline, on the HANDS dataset. A summary of the results is available in Table 1.

With regard to hand detection, the results are highly satisfactory, since almost all hands are detected by the algorithm if we refer to the mAP_{50}, which returns the percentage of hands correctly detected with an IoU threshold set at 50%. The mAP remains acceptable as the IoU threshold increases, demonstrating the accuracy of hand detection. Recall and accuracy are also high, demonstrating respectively the low number of undetected hands and wrongly detected hands, further demonstrating the robustness of our model.

For gesture classification only (with manually annotated bounding boxes as input), the results are also quite high, with an average accuracy of 93.7%. However, it should be noted that accuracy decreases when subject 3 is taken as the test subject (86.9%).

On the overall pipeline (hand detection followed by gesture classification), we observe a decrease of 4.8% in accuracy in comparison to gesture recognition with manual bounding boxes. This can be explained by the fact that when a hand is detected, it may only partly lie inside the predicted bounding box, thus making it harder for the classification model to recognize the gesture. Still, these results remain satisfactory when compared with those reported in [6] using the same dataset. It is important to bear in mind that the comparison is not straightforward. Indeed, in [6], the authors had access to images of a sixth subject which were not made public in the HANDS dataset [8]. In addition, the authors did not perform a cross-validation, they rather used images from subjects 1, 2, 3 and 6 for their training and subjects 4 and 5 for testing. Therefore, we compare their test results on subjects 4 and 5 with the average of our results obtained with the two models tested on these same subjects. In all cases, both their model and ours are trained on the same number of subjects (4) and tested on the same subjects. The comparison is therefore relevant. On average, on subjects 4 and 5, the total accuracy we obtain with our pipeline is 93.9% (92.3% for subject 4 and 95.7% for subject 5), while the accuracy on the same test set is reported to be 90.0% in [6].

Furthermore, it can be seen that the accuracy for subject 3 is lower than for the other subjects. This can be explained first, by the fact that subject 3 is further away from the camera than the other subjects, which reduces the resolution of the image patch fed to the classification network; and second, by the fact that subject 3, as opposed to subjects 1, 2, 4 and 5, moved around during the recording as illustrated in Fig. 9). This increases the variability of the data, making the task more difficult. When we train a model with images from subjects 1, 2, 4 and 5, the model does not learn this variability. This highlights the overall lack of variability in the HANDS dataset.

This experiment on a public dataset demonstrates that our proposed pipeline and its elements taken separately perform as well if not better than the pipeline presented in [6]. We also provide a more rigorous performance evaluation using cross-validation on this dataset, rather than a single train/test split.

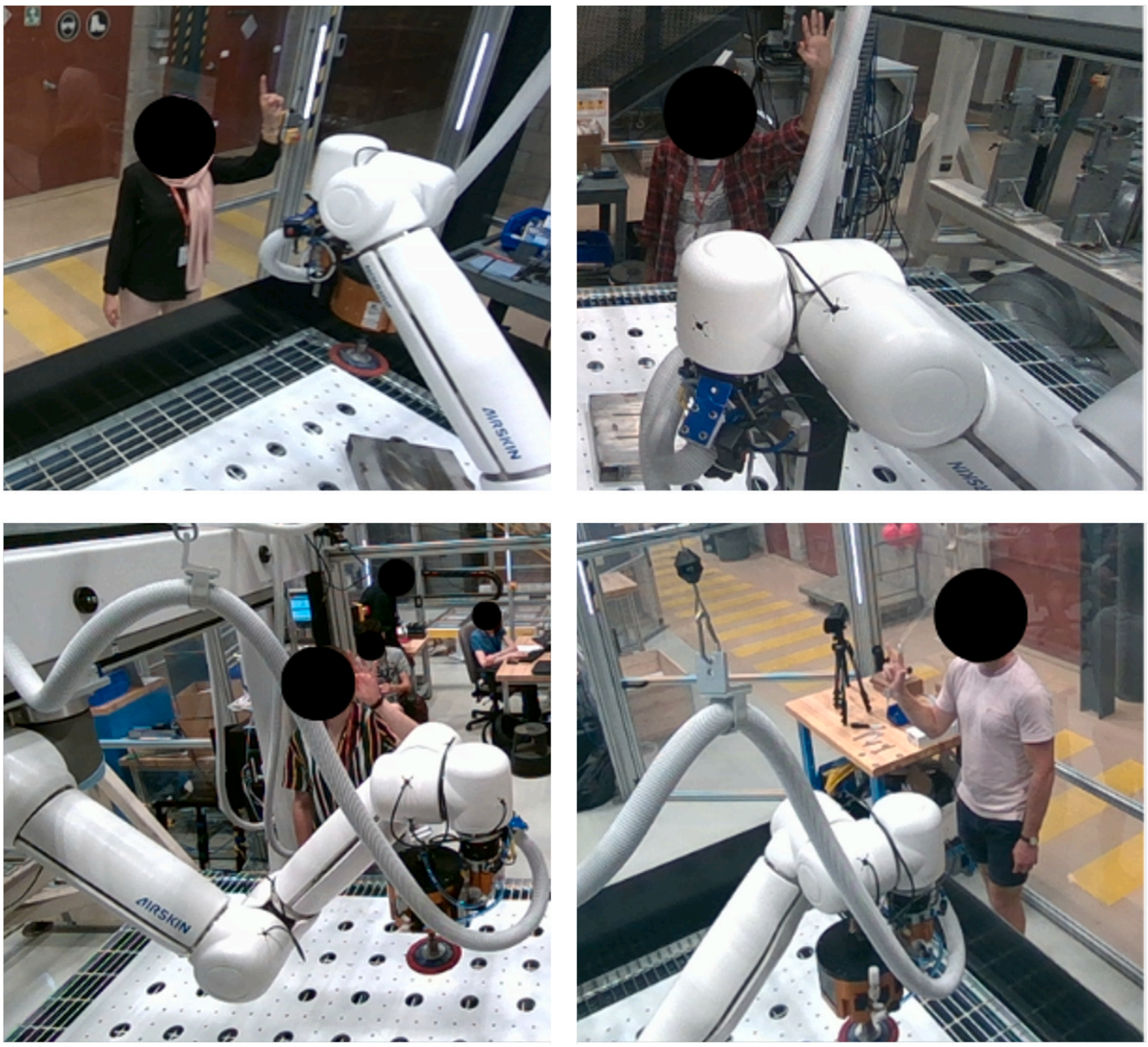


Fig. 5. Sample images from the dataset. The images are cropped here to better show the operator in each case.

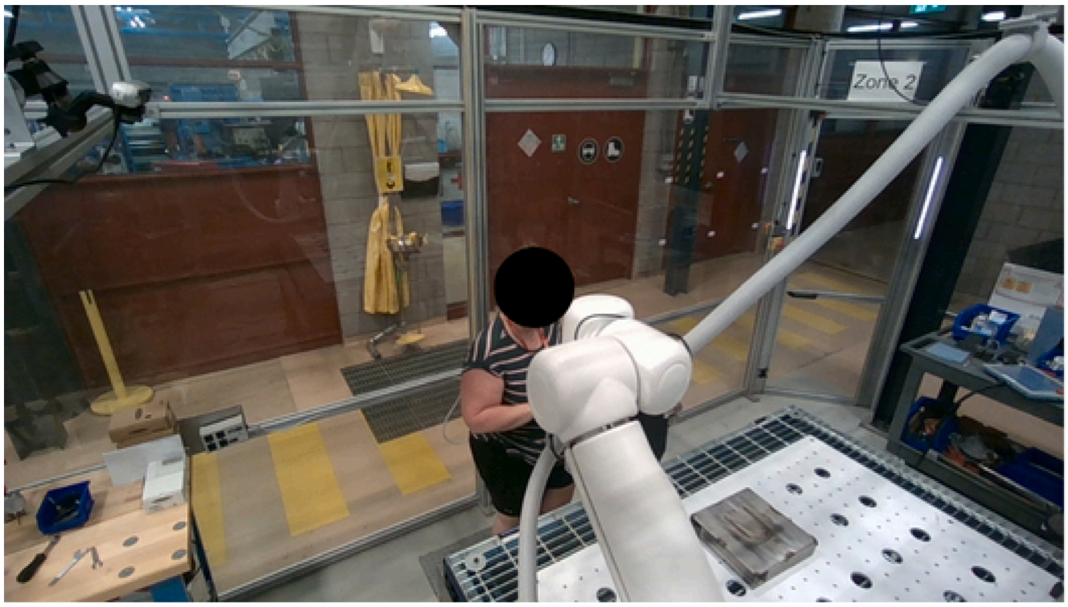


Fig. 6. Example of a non-annotated image in the dataset (hands hidden by cobot).

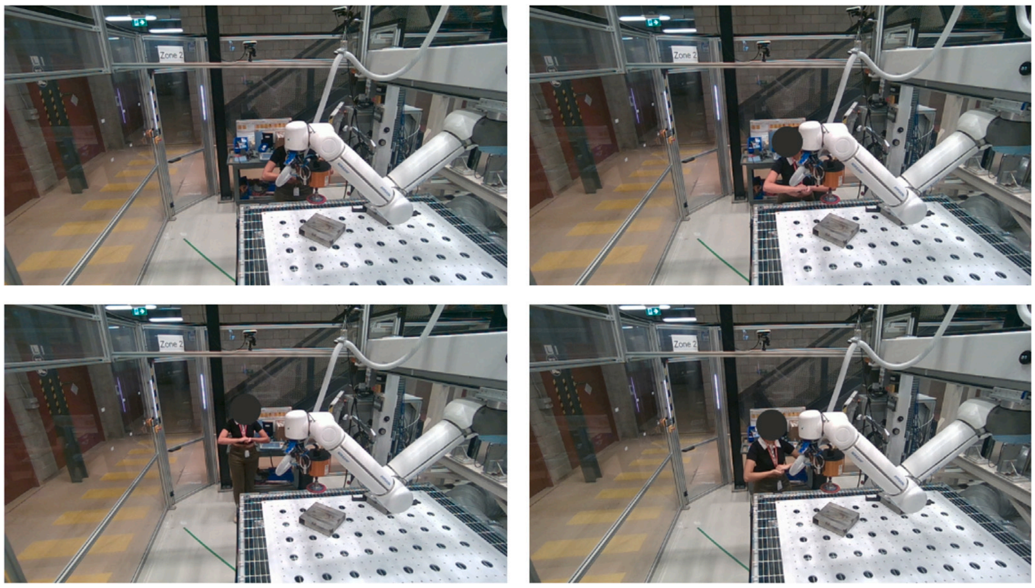
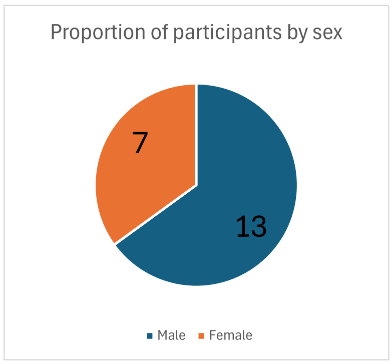


Fig. 7. Example of a participant moving within an area.



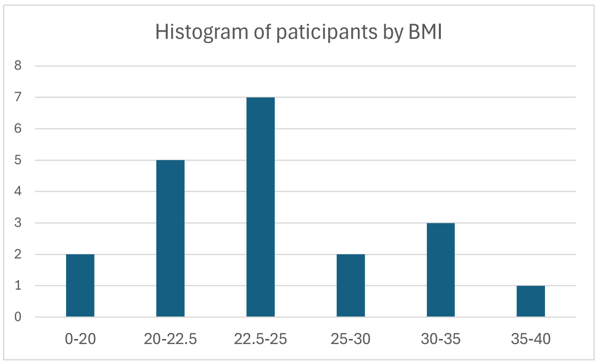


Fig. 8. Participants demographics.

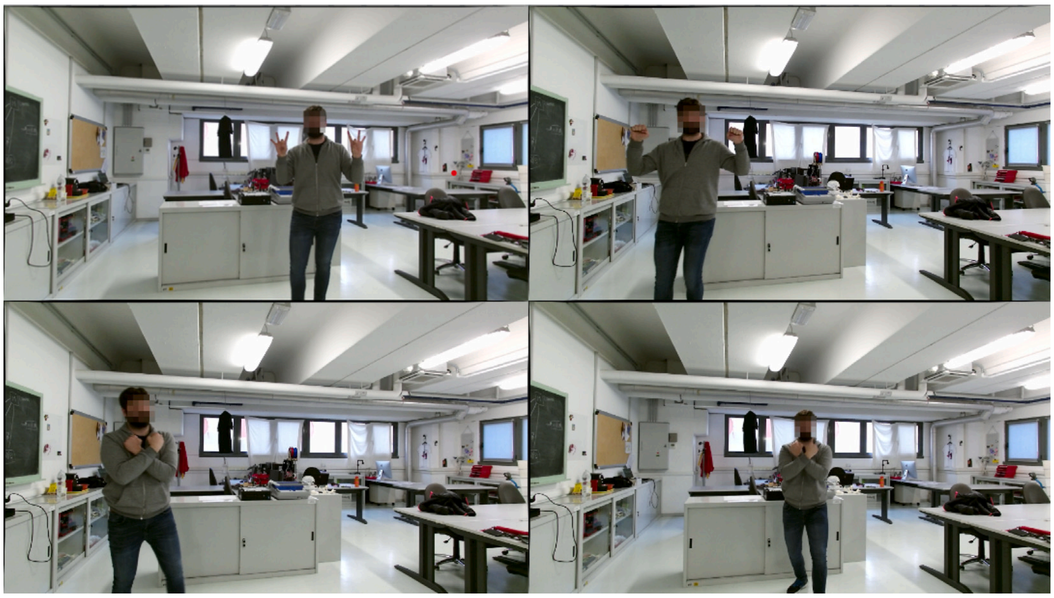


Fig. 9. Four images illustrating the movements of subject 3 from the HANDS dataset [8].

Table 2
Cross-validation on MuViH dataset (metrics in %).

MuViH	Test fold					Average
	1	2	3	4	5	
Hand Detection						
mAP_{50}	96.3	97.9	87.0	98.3	97.8	95.5
$mAP_{50:95}$	50.0	53.0	47.5	53.3	52.1	51.2
Precision	96.0	95.7	93.2	97.3	95.4	95.5
Recall	90.2	95.0	78.7	96.4	94.7	91.0
Gesture Classification						
Accuracy	89.2	88.8	84.7	91.4	92.1	89.2
Detection and Classification						
Accuracy	77.5	80.0	65.2	84.0	83.3	78.0
Accuracy (detected hands only)	85.3	85.2	85.6	88.7	91.4	87.3

6.3. Pipeline evaluation on muvih dataset

The cross-validation results of the proposed hand gesture recognition pipeline on the new MuViH dataset are provided in Table 2. Unlike Table 1, here each fold includes images of several participants: fold 1 comprises participants 1 to 4, fold 2 comprises participants 5 to 8, and so on. Moreover, we introduce an additional metric to evaluate the complete pipeline. While "Accuracy" measures the percentage of correctly recognized gestures among all annotated images, "Accuracy for detected hands only" measures the percentage of correctly recognized gestures among all images where a hand is detected. (This metric was not relevant in the case of HANDS, as almost all hands were detected).

The hand detection results for the MuViH dataset are lower than those obtained with HANDS (95.5% mAP_{50} compared to 99.3% previously). This can be attributed to the higher complexity of the MuViH dataset. While in HANDS, the hands are always visible to the camera, in MuViH several elements (cobot and equipment) can occlude the line of sight of the sensor, as illustrated by the example in Fig. 11. The results decrease even more sharply when the confidence threshold is increased ($mAP_{50:95}$), but once again the precision and recall are greater than 90%, which indicates sufficient robustness for the 50% threshold. Note that fold 3 has much lower results than the others, as discussed below.

Fig. 10 illustrates the number of false positive hand detections obtained in the 6,236 images (<8% of the dataset) where no hand annotation was possible due to occlusions, as a function of the confidence threshold on the detection model. Considering that only the hand performing the gesture (for single hand gestures) is annotated in the dataset and used to train the detection model, it is expected that, since the model will often detect the second hand, these will be considered as false positives. Nevertheless, with a 50% threshold for which the average mAP is 95.5% (Table 2), we only have 1,013 false hand detections out of the 6,236 images considered. Note that multiple false positives can be detected in a single image.

Similarly, the results for hand gesture recognition and for the complete pipeline are lower than for the HANDS dataset (89.2% accuracy compared to 93.7%, and 78.0% compared to 88.9% previously), explained again by the complexity of the environment in MuViH. Nevertheless, if we take into account only images where a hand is detected by the model, the accuracy remains quite close to that obtained on the other dataset (87.3%).

Tables 3 and 4 provide respectively the cumulative confusion matrix (sum of the confusion matrices of the 5 classification models in the cross-validation) and the per-class accuracies. The latter are computed as the accuracy of each class (the positive class) against all other classes (taken collectively as the negative class). These results show that the model seems to be slightly more accurate for the two-handed gestures (around 99%), which correspond to the most critical commands, as well as the `punch` gesture (99.0%), compared to the `one` to `five` one-handed gestures (between 96% and 98%).

According to the confusion matrix in Table 3, three pairs of gestures show higher confusion rate: `two` vs. `three`, `four` vs. `five`, and `stop` vs. `cancel`. Gestures `three` and `two` and gestures `five` and `four` only vary by the thumb open or closed, respectively. The thumb, located at the end of the hand, is sometimes difficult to distinguish in the image, either because it is confused with the background, or because it is occluded. Additionally, gestures `stop` and `cancel` also show some level of confusion; this can be explained by the similarity in diagonal alignment of the hands between those two gestures. Nevertheless, for all gestures, the per-class accuracy is greater than 96.3%.

All the results in Table 2 are noticeably lower for fold 3. This result can be explained by the fact that the eleventh participant (part of fold 3) is the only one with dark skin. Table 5 highlights the performance specifically on this subject. Quantitatively, the accuracy for this participant alone is much lower than the average over the three other participants in the same fold. The same goes for the accuracy for detected hands only, illustrating a poorer performance of the detection model on this participant's images. The average results on the three other participants are however in the same range as the other folds. The model has thus great difficulty generalizing to skin colors different than the ones seen in training.

6.4. Datasets crossover

This experiment consists of training the model on one dataset (either HANDS or MuViH) and evaluating it on the other one. The aim of this experiment is to evaluate the generalization capability of our pipeline and to study the impact of dataset variability on hand gesture recognition.

We cross-referenced training and testing data between the HANDS and MuViH datasets for detection (Table 6), recognition (Table 7) and the entire pipeline (Table 8). The recognition model is trained only on the gestures that are common to both datasets (13 in total). The results in these tables are obtained by averaging those of the 5 cross-validation models. For example, for the column "Training on HANDS"and the row "Test on MuViH", the 5 models trained on the 5 folds of the HANDS dataset are tested on the entire MuViH dataset.

Common to these three tables is the poor performance of the models trained on HANDS when tested on the MuViH dataset. As a result, for the complete pipeline, the model trained on HANDS performs barely better than random gesture prediction with 13 classes. This is mainly due to the low performance of the hand detection model (mAP_{50} = 6.3%). In addition, this metric does not take into account the number of images for which no hands were detected, which nevertheless has an impact on the result of the entire pipeline.

These results give rise to two interesting observations:

1. Training on the HANDS dataset does not allow good generalization to the MuViH dataset, which is more representative of practical reality in industrial conditions, with an operator who can move freely around the platform.
2. Conversely, the model trained on MuViH offers better generalization to other datasets. Our results show that when the classification model is trained on MuViH and tested on HANDS, it achieves a higher accuracy (96.6%) than when trained and tested on HANDS (93.7%).

6.5. Comparison between multi and single-view results

We now want to evaluate the effect of aggregating the overall pipeline's predictions from multiple synchronized views. For this experiment, we only use the MuViH dataset since it was acquired in multi-camera mode. Table 9 presents the results for the single-view and multi-view modes in terms of overall detection and classification pipeline's accuracy.

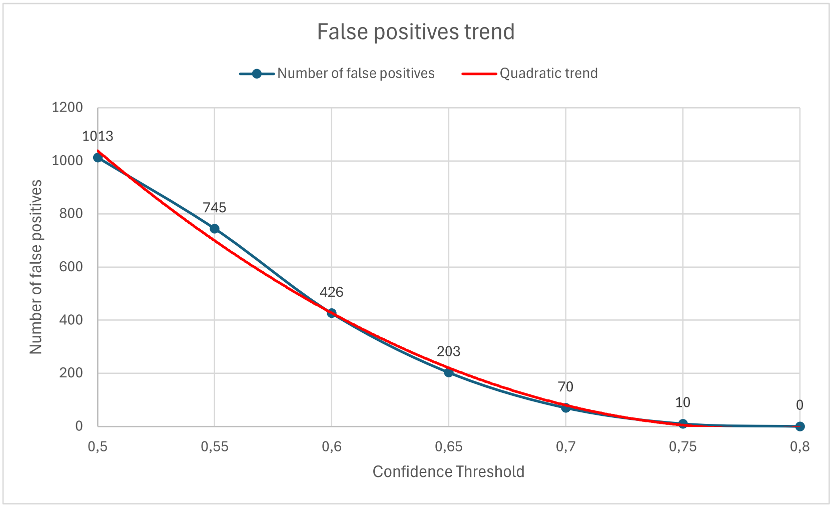


Fig. 10. Hand detection's false positive analysis on the 6236 images of the MuViH dataset where no hand annotation was made possible due to occlusions.

Table 3
Confusion matrix for gesture classification on MuViH. Each entry corresponds to a number of images.

GT	Predicted label										Total
	One	Two	Three	Four	Five	Punch	Run	Start	Stop	Cancel	
one	**8133**	567	161	38	28	221	10	23	7	32	9220
two	366	**7883**	733	234	30	14	11	9	14	39	9333
three	133	578	**8100**	190	208	27	25	27	29	32	9349
four	38	177	293	**7770**	877	54	14	36	29	62	9350
five	14	24	219	584	**8324**	15	16	3	14	104	9317
punch	44	7	34	23	16	**9069**	33	8	18	102	9354
run	3	6	56	46	10	89	**4069**	28	19	67	4393
start	57	25	28	26	51	8	34	**4283**	18	72	4602
stop	7	3	24	12	11	33	82	7	**4079**	367	4625
cancel	18	39	32	72	144	41	111	52	238	**8474**	9221
Total	8813	9309	9680	8995	9699	9571	4405	4476	4465	9351	78 764

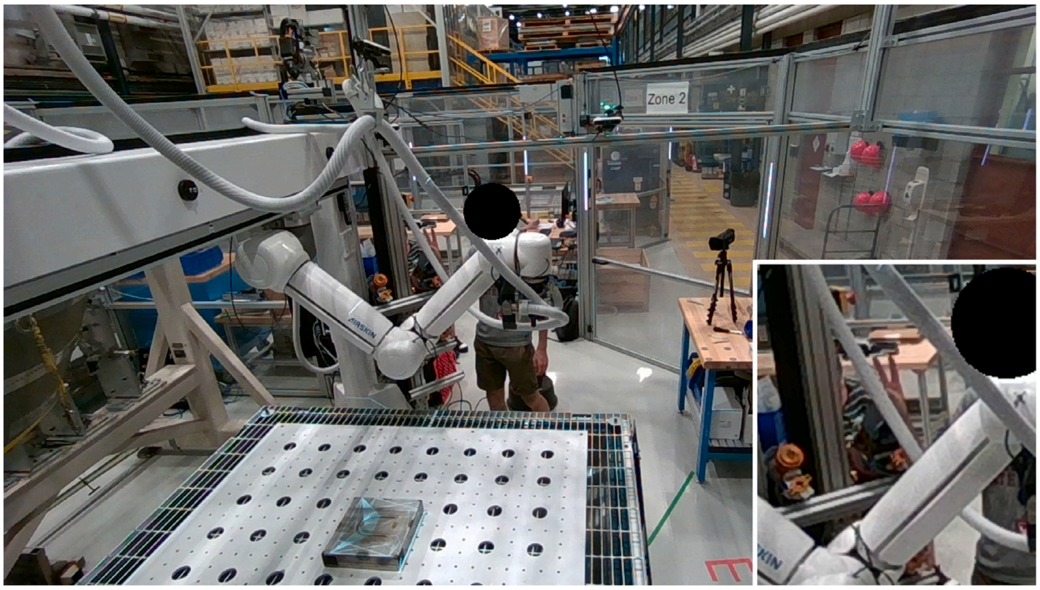


Fig. 11. Example of a case of occlusion, the gesture performed is a 4.

Note that the accuracy in single-view mode takes into account all images in the dataset, including those in which the hand is not visible. This explains why the results for the single-view in Table 9 are lower than those presented in Table 2, in which non-annotated images are not taken into account. When the hand is occluded, the multi-view mode can still make a correct gesture recognition thanks to the other images in the group, as the hands are likely to be visible in at least one of them.

Table 9 shows a large improvement (14%) when multiple views are aggregated. If we compare these results to those in Table 2, we see that the multi-view performance is comparable to the complete

Table 4
Per-class accuracies for gesture classification on MuViH.

	Per-class accuracy
one	97.8%
two	96.3%
three	96.4%
four	96.4%
five	97.0%
punch	99.0%
run	99.2%
start	99.3%
stop	98.8%
cancel	97.9%

Table 5
Comparison of complete pipeline results (in %) for participant 11 versus the other participants in fold 3.

MuViH	Participant 11	Other participants
Accuracy	12.9	82.0
Accuracy for detected hands only	51.9	88.6

Table 6
Average hand detection results (in %).

mAP_{50}		Trained on	
		HANDS	MuViH
Tested on	HANDS	99.3	84.7
	MuViH	6.3	95.5

Table 7
Average gesture classification results (in %).

Accuracy		Trained on	
		HANDS	MuViH
Tested on	HANDS	93.7	96.6
	MuViH	35.0	89.2

Table 8
Average results of complete pipeline (in %).

Accuracy		Trained on	
		HANDS	MuViH
Tested on	HANDS	88.9	78.6
	MuViH	8.4	87.3

Table 9
Results (in %) in single- and multi-view settings for the MuViH dataset.

MuViH	Test fold					Average
	1	2	3	4	5	
Single-view	72.9	74.5	60.0	76.6	79.8	72.8
Multi-view	88.1	87.6	75.2	91.9	91.5	86.8

pipeline for detected hands only (86.8% vs 87.3% accuracy). In a case of occlusion of the hand, the multi-camera system can then use another camera to obtain a correct prediction of the gesture performed. It also allows to correct any prediction error on one camera by comparing with predictions made on the other ones. An example of a group of images for which a multi-view approach can correct errors is shown in Fig. 12.

6.6. Inference time and size

The complete pipeline execution time for an image is 37.5 ms, including 14.3 ms for detection and 23.2 ms for classification, using an nVidia RTX 3060 GPU. This equates to around 27 frames per second. The pipeline could still be improved in terms of speed, but the acquisition frequency of the Intel RealSense D455 camera is limited to 30 frames per second.

Moreover, the inference model weighs less than 1 Gb on the GPU, making it deployable on multiple cameras with a single GPU while maintaining acceptable execution times.

Industrial-grade GPUs, such as NVIDIA A100 or A40, typically offer higher computational power, memory capacity, and parallelism compared to an RTX 3060 (the one used for our experiments). This could further reduce execution times, potentially enabling the system to process more frames per second, even beyond the RealSense D455's 30 frames per second limit. The inference model's small size (less than 1 GB) is a significant advantage. On GPUs with large memory capacities, it becomes feasible to deploy the pipeline across multiple cameras, handling several streams simultaneously. Edge computing is also a viable option for decentralized setups, especially in scenarios requiring low latency and reduced network dependency. Devices like NVIDIA Jetson series or Google Coral offer lower computational power compared to an RTX 3060, but the lightweight nature of the pipeline (model size under 1 GB and sub-40 ms execution time) aligns well with their capabilities.

6.7. General discussion & limitations

So far, the proposed solution only predicts gestures from a group of two to three synchronized images from different viewpoints taken at a single timestamp. The aggregation of different viewpoints has proven to enhance the hand gesture recognition, more specifically in situations of visual occlusion. In future work, we envisage a temporal aggregation of images, taking into account several successive frames to determine a posteriori whether or not a gesture is being made. A similar solution has been exploited in previous work [6].

The camera placement that was considered for the MuViH data collection was based on guidelines and constrained by the geometric configuration of the platform. While camera placement in MuViH has not been necessarily optimized for the platform under consideration, we can make the following recommendation for adaptability to different platforms: each possible operator's position around the platform should be covered by at least one camera. This minimum number could be increased to two or three in scenarios where a robot is moving around, possibly occluding the cameras field of view. Algorithms to determine the camera placement have recently been proposed, such as the one in [28]. Such a tool could be used to maximize the coverage of the workspace.

Furthermore, our current methodology requires the detection and classification algorithms to be run on all the cameras. This is necessary to ensure that the gesture recognition comes from a viewing angle free of occlusion. In the interest of efficiency, we are also investigating the automatic selection of the camera with the clearest view of the operator's hands. This will allow the detection and recognition algorithms to be run only on the optimal camera.

Unlike other public datasets, all images in MuViH were recorded in the same cobotic environment. Although it offers a great variability in movements and viewpoints, the background of the cobotic cell remains unchanged. Nevertheless, we were able to show that the models trained on MuViH are able to adapt to different backgrounds, such as the one in HANDS.

The limitations of the detection and recognition models presented lie mainly in their declining performance when it comes to dark-skinned individuals, compromising their reliability in real-life applications. This shortcoming can largely be attributed to their under-representation in the MuViH database used to train these models. This observation underlines the importance of more diverse data collection to represent all populations more fairly. To mitigate under-representation and enhance models generalization, additional data augmentation such as random modification of hue and saturation channels, histogram equalization could be considered. Moreover, balanced sampling strategies could ensure that participants with diverse skin tones are equitably represented in training batches.

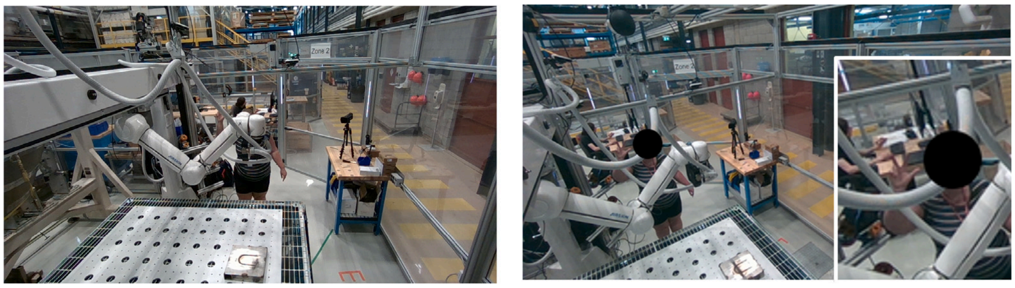


Fig. 12. Illustration of how the multi-camera configuration can correct occlusion problems. In the image on the left we are able to locate the hand but not recognize the gesture, whereas in the image on the right, obtained from another point of view, we can recognize gesture three.

7. Conclusion

We developed a complete hand gesture recognition pipeline that takes into account images from several cameras. We also created a new dataset, named MuViH, with a large amount of variability and occlusions; this allowed us to build a robust pipeline that can generalize to other datasets. In order to demonstrate the versatility of the new dataset, we tested our model on an existing dataset (HANDS) and showed that it performed as well as the model from the HANDS authors. Meanwhile, when our model was trained on their data, it did not generalize well. We also showed that a multi-camera system can greatly limit the impact of occlusions in the cobotic cell, as well as providing a larger field of view and allowing the operator to move more freely around the cell. The MuViH dataset is entirely hand-annotated and allows full training for a hand detection model as well as a gesture recognition model. In future work, we intend to annotate a subset of the images for segmentation of the human body in the cobotic cell. This will make it possible to train segmentation networks that are robust to occlusions.

CRediT authorship contribution statement

Corentin Hubert: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Nathan Odic:** Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Marie Noel:** Software. **Sidney Gharib:** Validation, Software. **Seyedhossein H.H. Zargarbashi:** Writing – review & editing, Resources, Project administration, Funding acquisition. **Lama Séoud:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was conducted as part of a project supported by the National Research Council Canada (NRC)'s METALTec industrial research group, the National Program Office, and the Metal Transformation Research and Innovation Consortium (CRITM). The authors thank Julien-Mathieu Audet, NRC researcher, as well as the technical team at NRC's Aerospace Manufacturing Technologies Center (Christian Corbeil, François Ferland, Henri Elame and Diego Garcia) for their help in preparing the cobotic platform for acquiring images. The authors also thank Philippe Debanné for his valuable help with preparing the research ethics approval, logistics and manuscript editing. Finally, we thank all the data collection's participants.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.rcim.2025.102957.

Data availability

The MuViH dataset is made available for academic purpose upon request (https://doi.org/10.5683/SP3/JZJTGG).

References

[1] S. Mahmud, X. Lin, J.-H. Kim, Interface for human machine interaction for assistant devices: A review, in: 2020 10th Annual Computing and Communication Workshop and Conference, CCWC, IEEE, 2020, pp. 0768–0773.
[2] V. Suma, Computer vision for human-machine interaction - review, J. Trends Comput. Sci. Smart Technol. (TCSST) 1 (02) (2019) 131–139.
[3] D. Gorecky, M. Schmitt, M. Loskyll, D. Zühlke, Human-machine-interaction in the industry 4.0 era, in: 2014 12th IEEE International Conference on Industrial Informatics, INDIN, Ieee, 2014, pp. 289–294.
[4] L. Guo, Z. Lu, L. Yao, Human-machine interaction sensing technology based on hand gesture recognition: A review, IEEE Trans. Hum.-Mach. Syst. 51 (4) (2021) 300–309.
[5] E. Nazarova, O. Sautenkov, M.A. Cabrera, J. Tirado, V. Serpiva, V. Rakhmatulin, D. Tsetserukou, Cobotar: interaction with robots using omnidirectionally projected image and DNN-based gesture recognition, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2021, pp. 2590–2595.
[6] C. Nuzzi, S. Pasinetti, R. Pagani, S. Ghidini, M. Beschi, G. Coffetti, G. Sansoni, MEGURU: a gesture-based robot program builder for Meta-Collaborative workstations, Robot. Comput.-Integr. Manuf. 68 (2021) 102085.
[7] Malima, Ozgur, Cetin, A fast algorithm for vision-based hand gesture recognition for robot control, in: 2006 IEEE 14th Signal Processing and Communications Applications, Ieee, 2006, pp. 1–4.
[8] C. Nuzzi, S. Pasinetti, R. Pagani, G. Coffetti, G. Sansoni, HANDS: an RGB-D dataset of static hand-gestures for human-robot interaction, Data Brief 35 (2021) 106791.
[9] A. Mavi, A new dataset and proposed convolutional neural network architecture for classification of American sign language digits, 2020, arXiv preprint arXiv:2011.08927.
[10] A.H. Alrubayi, M.A. Ahmed, A. Zaidan, A.S. Albahri, B. Zaidan, O.S. Albahri, A.H. Alamoodi, M. Alazab, A pattern recognition model for static gestures in malaysian sign language based on machine learning techniques, Comput. Electr. Eng. 95 (2021) 107383.
[11] A. Kapitanov, K. Kvanchiani, A. Nagaev, R. Kraynov, A. Makhliarchuk, HaGRID–HAnd Gesture Recognition Image Dataset, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 4572–4581.
[12] Y. Zhang, C. Cao, J. Cheng, H. Lu, EgoGesture: A new dataset and benchmark for egocentric hand gesture recognition, IEEE Trans. Multimed. 20 (5) (2018) 1038–1050.
[13] V.T. Hoang, HGM-4: A new multi-cameras dataset for hand gesture recognition, Data Brief 30 (2020) 105676.
[14] Z. Zou, K. Chen, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey, Proc. IEEE 111 (3) (2023) 257–276.
[15] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[16] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[17] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. 28 (2015).

[18] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37.

[20] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics YOLO, 2023, URL https://github.com/ultralytics/ultralytics.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.

[22] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.

[23] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[24] Q. Gao, Y. Chen, Z. Ju, Y. Liang, Dynamic hand gesture recognition based on 3D hand pose estimation for human–robot interaction, IEEE Sens. J. 22 (18) (2021) 17421–17430.

[25] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[26] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, Adv. Neural Inf. Process. Syst. 29 (2016).

[27] S. Zargarbashi, G. Côté, J. Boisvert, R. Meziane, C. Xu, C. Hubert, S. Jocelyn, C. Gosselin, Collaborative robotic finishing platform for metal part processing towards industry 5.0, in: Proceedings of the 15th International Conference on Mechanical and Aerospace Engineering, 2024, in press.

[28] P. Oščádal, T. Kot, T. Spurnỳ, J. Suder, M. Vocetka, L. Dobeš, Z. Bobovskỳ, Camera arrangement optimization for workspace monitoring in human–robot collaboration, Sensors 23 (1) (2022) 295.

[29] M.-A. Drouin, L. Seoud, Consumer-grade RGB-D cameras, 3D Imaging, Anal. Appl. (2020) 215–264.

[30] OptiTrack, Camera placement, 2023, https://docs.optitrack.com/hardware/camera-placement. (Online; accessed 31-October-2024).

[31] D. Rempel, M.J. Camilleri, D.L. Lee, The design of hand gestures for human–computer interaction: Lessons from sign language interpreters, Int. J. Hum.-Comput. Stud. 72 (10–11) (2014) 728–735.

[32] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.