Assembly Action Understanding from Fine-Grained Hand Motions, a Multi-camera and Deep Learning Approach

Enrique Coronado, Kosuke Fukuda, Ixchel G. Ramirez-Alpizar, Natsuki Yamanobe, Gentiane Venture and Kensuke Harada

Abstract— **This article presents a novel software architecture enabling the analysis of assembly actions from fine-grained hand motions. Unlike previous works that compel humans to wear ad-hoc devices or visual markers in the human body, our approach enables users to move without additional burdens. Modules developed are able to: (i) reconstruct the 3D motions of body and hands keypoints using multi-camera systems; (ii) recognize objects manipulated by humans, and (iii) analyze the relationship between the human motions and the manipulated objects. We implement different solutions based on OpenPose and Mediapipe for body and hand keypoint detection. Additionally, we discuss the suitability of these solutions for enabling real-time data processing. We also propose a novel method using Long Short-Term Memory (LSTM) deep neural networks to analyze the relationship between the detected human motions and manipulated objects. Experimental validations show the superiority of the proposed approach against previous works based on Hidden Markov Models (HMMs).**

I. Introduction

Flexibility and productivity demands towards Industry 4.0 have recently increased the interest in building cloud database systems able to store and share useful information enabling the easy generation of robotic applications [1]. A suitable approach for representing knowledge in these databases is the use of affordances, which in the manufacturing context can be understood as "the relationships that exist between objects available for manipulation and the associated actions that humans or robots can take with these objects" [2]. For building affordances it is necessary to analyze human actions. This task requires the segmentation and labeling of human motions as well as recognition of the objects involved in the manipulation task. However, these are challenging and complex processes often involving time-consuming and manual operations [2]. The objective of the project presented in this article is to create a modular and usable framework that enables the easy creation of large-scale databases using affordances as a knowledge source. Therefore, this article presents an approach enabling the analysis of the relationships between the human motions with the objects in the environment. These relationships are especially relevant in assembly monitoring [3] and human-robot collaboration (HRC) [4] applications. In this context, state-of-the-art frameworks enabling this capability are still rare and far from being usable (i.e, efficient, effective, and use-to-use) [5].

This article is organized as follows: in sections II and III we describe related work and summarize contributions. Section IV describes the proposed software architecture for data acquisition and processing. Section V describes a novel approach using Deep Learning to enable assembly action analysis. Section VII presents experimental settings and results. Conclusions follow.

II. Related work

Human motion understanding is a systematic process that performs the sensing, segmentation, recognition, and analysis of people's movements [6], [7]. Frameworks enabling this process are relevant for several disciplines, such as biomechanics [8], ergonomics [9] and Human-Robot Interaction (HRI) [10]. Popular machine learning approaches used in these works are Support Vector Machine (SVM) [11], Hidden Markov Models (HMM) [11], Gaussian Mixture Models (GMM) [10], and Deep Neural Networks (DNN) [12]. In many of these works, the target actions to analyze are simple gestures or everyday motions that do not involve fine-grained motions of the fingertips. An exception is [13], in which 3D positions of hands and fingers are obtained from a Leap Motion sensor for enabling classification of signs from the American Sign Language. Many efforts focused on the recognition of human actions and gestures compel humans to wear special ad-hoc devices such as inertia sensors [14] and gloves [15]. Because these devices are wearable, they can represent a burden on the user [14]. As described in [5], most of these ad-hoc devices can be cumbersome to use due to the amount of sensors and cables attached to the body. Moreover, communication between humans and robots in manufacturing scenarios should be performed in a natural way. Therefore, anything that can disturb co-workers should not be attached to their body [5]. Other works propose the use of daily-use wearable devices such as smartwatches to recognize human activities [16], control robot actions [10] and for Human-Robot Collaboration (HRC) tasks [17]. However, these devices can hardly be used to recognize fine-grained movements of hands. Image-based systems are also popular methods used for the analysis of human motions. These systems are basically classified as markers-based and marker-less. On the one hand, markers-based approaches

E. Coronado and G. Venture, are with the Department of Mechanical Systems Engineering, Tokyo University of Agriculture and Technology, 2-21-16 Nakacho, Koganei, Tokyo, Japan.

K. Fukuda and K. Harada are with the Department of Systems Innovation, Graduate School of Science and Engineering, Osaka University, 1-3 Machikaneyama-cho, Toyonaka, 560-8531, Japan.

I. G. Ramirez-Alpizar, N. Yamanobe and K. Harada are with the Automation Research Team, Industrial CPS Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan.

Corresponding author's email: enriquecoronadozu@gmail.com.

require that the user wears an special outfit and a set of markers carefully attached to the body. Therefore, this approach presents similar issues that those works using wearable devices. On the other hand, marker-less systems enable humans to move and act in their environment naturally. Therefore, they can be considered more suitable for unstructured and real-world scenarios [5]. Marker-less approaches can use single- multiple- and depth-based cameras. While the use of single and depth cameras enables an easy setup, they often provide poor robustness and performance in real settings [5]. Works developing gesture and action recognition systems using single and depth cameras are reviewed in [18].

There exists a broad variety of works in different disciplines using image-based systems for the analysis of actions or gestures from human motions. In [18] is reviewed some of the most relevant applications. In many of them, body, object, or hand recognition, as well as motion analysis, are processed offline. These approaches are in fact unsuitable for real-time interaction with robots. In the context of manufacturing, very few related works can be found in the literature. However, and to the best of our knowledge, none of the related works meet the requirements presented in this article. These requirements, which are focused on the context of manufacturing and robotics are: (i) real-time detection and 3D reconstruction of fine-grain human hands, (ii) real-time recognition of objects for assembly tasks, (iii) component-based, and distributed system architecture, and (iv) analysis of actions being executed by the humans.

Research articles performing the analysis of assembly actions from human motions and the manipulated objects are still rare. A relevant exception is [15]. However, the approach used in [15] requires users to wear gloves and markers when performing assembly tasks. Their approach was recently improved in [2], by substituting the hand motion acquisition system based on gloves by a four-camera system that uses OpenPose [19] for obtaining body and hand 2D positions. However, the work presented in [2] was constrained to perform offline. In order to reach the current video standard frame speed (29.97 fps) in a single-camera system using OpenPose for skeleton recognition, it is required the use of high-performance CPUs and GPUs [20]. In multi-camera systems where body and hand recognition algorithms must be processed in parallel for images of each camera, the frame rate can be drastically reduced. Therefore, one of the challenges of this article was to analyze alternatives to reach the real-time body and hand recognition in multi-camera systems. Other relevant and recent works that share some of the objectives of this article are [21] [3]. On the one hand, [21] proposes a multi-modal sensor fusion approach for action recognition of assembly tasks. However, this approach requires the use of wearable sensors. Moreover, this approach uses Kinect, which as explained by [3], presents poor performance and robustness in real settings as well as limited recognition range; therefore, making this device more suitable for the field of entertainment. On the other hand, [3] presents a single-camera approach for recognizing assembly actions. However, this approach is limited to repetitive and tool-dependent actions, such as wrenching and hammering. Due to the use of a single-camera system, their approach mostly relies on algorithms for object recognition rather than the analysis of the body and hand motions. Moreover, authors reported recognition issues when parts of the body or objects were occluded. On the other hand, a multi-camera system can be used to deal with occlusion issues.

III. Contribution

As described in section II most of the human understanding frameworks reported in the literature are limited to be used in some specific disciplines or scenarios, burden the human motion using some wearable device, or require expensive computational resources to be effective. Therefore, the main contribution of this article is the creation of a usable (efficient, effective, and easy-to-use) software architecture to enable the analysis of human action when performing assembly tasks. To enable efficient and effective recognition of the human body and hand motions we propose different solutions based on OpenPose and Mediapipe. We evaluated the performance and suitability of these solutions for enabling real-time processing by using data obtained from a real assembly task. Due to the very recent launching of Mediapipe, this type of comparative analysis represents a novel task. After obtaining 3D motion of human hands, effective human action understanding is performed by a set of long short-term memory (LSTM) recurrent neural networks. One of these neural networks is designed and trained to narrow down the pool of possible human actions. Then, a second LTSM network uses this information to enable effective recognition of assembly actions. The performance superiority of this approach is validated and compared with the method used in [2].

IV. Software architecture

In order to enable the easy re-use and integration of the proposed methods with different image acquisition systems and robots, we propose a modular and distributed software architecture. This architecture enables real-time recognition of assembly objects, human body and hands as well as data collection and visualization. Figure 1 shows the general overview of this architecture. Modules composing this architecture are connected using the publish-subscribe pattern using the ZeroMQ [22] back-end of the NEP framework [23]. This enables high-performance communication between the software modules [24] and the easy installation and re-use of these modules in many different versions of Windows, OSX, and Linux systems [23]. Integration with the Robot Operating System (ROS) version 1.0 and 2.0 is a trivial task that can be easily performed by changing the type of back-end communication method to use when defining a publisher in NEP. An example of how to change the back-end method in NEP is shown in [23]. Modules composing the proposed software architecture are defined below.

A. User Interface

We developed a user interface (figure 2) based on Node.js to improve the usability of the system. This interface has

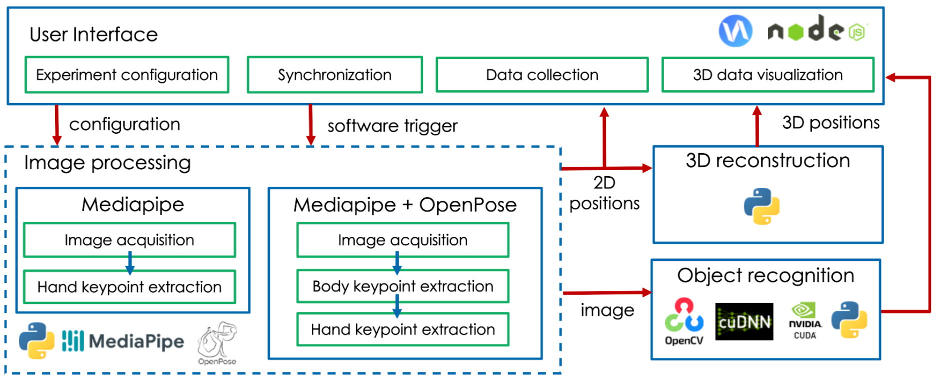
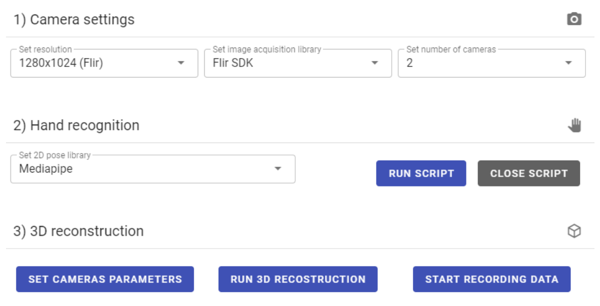


Fig. 1. General system architecture enabling data visualization, data collection, and online recognition of the human body, hands, and assembly objects. Modules are connected using the publish-subscribe communication pattern.

three sections: 1) Camera settings, 2) Hand recognition and 3) 3D reconstruction. In section 1) the user can select the type of camera to use and their resolution as well as the number of cameras to use in the experiments. In section 2) the user can select the type of algorithms used for hand recognition and tracking (i.e., Mediapipe or OpenPose) as well as start and stop the python scripts performing these actions. In section 3) a set of buttons enable the user to set the camera parameters, execute the python script performing 3D reconstruction of hand positions as well as save 2D and 3D information of detected human hands in text files. This interface also sends a software trigger, which is used to synchronize image acquisition from the cameras.


1) Camera settings

Set resolution: 1280x1024 (Flir)
Set image acquisition library: Flir SDK
Set number of cameras: 2

2) Hand recognition

Set 2D pose library: Mediapipe

3) 3D reconstruction

Fig. 2. User interface developed for enabling synchronization of images, data collection and 3D visualization of body and hands

B. Image acquisition and key-points extraction

To obtain the position of the human hands we use the python versions of OpenPose and Mediapipe [25]. While OpenPose has been widely used in different disciplines including robotics, Mediapipe is a novel library developed and very recently launched by Google [26]. Therefore, its use and suitability in robotic applications is an unexplored area. Due to its robustness, the official version of OpenPose available in [27] is probably the most used for body detection. However, this version requires top-end hardware to enable real-time recognition of the human body and is considered highly inefficient [28]. Moreover, lightweight implementations of OpenPose, such as [29], do not offer many of the tools available in the official version of OpenPose, such as hand recognition. On the other, Mediapipe was designed to enable fast machine learning inference and processing even on common hardware. In this project, the main focus is to obtain the 3D position of human fingers to be able to analyze the relationship between manipulated objects and fine-grained hand movements by using multi-camera systems. For this, we propose different approaches for overcoming the low frame rate issue that is associated with the official version of OpenPose. The first approach consists of the use of Mediapipe to directly obtain and extract key-points of human hands. The second approach combines Mediapipe and OpenPose. It is relevant to highlight that the OpenPose hand recognition algorithm requires the knowledge of the position of the wrist, elbow, and shoulder to work. Therefore, rather than using the results of OpenPose skeleton tracking as the input of the hand recognition methods in OpenPose, we use Mediapipe to find the wrist, elbow, and shoulder positions. These outputs from Mediapipe algorithms for body detection are then used as inputs for the OpenPose methods to recognize the hand key-points. This approach is referred on this article as “Mediapipe + OpenPose”. Performance evaluation of these approaches is presented in section VII. Images used as input to this module can be acquired online by a synchronized multi-camera system or be loaded from the computer.

C. Recognition of assembly objects

For enabling fast recognition of assembly objects, we developed an object recognition module that integrates the Deep Neural Networks (DNN) methods of OpenCV. For this, we compiled OpenCV with CUDA and cuDNN support. To validate this module we used a YOLO-based [30] deep learning model. We use several images of assembly parts of an airplane toy (figure 3) to train this model. However, the implemented module can be used to recognize other types of objects using different algorithms, such as MobileNetSSD [31] and Mask-RCNN [32]. Figure 4 show examples of hand recognition using only Mediapipe. Figure 5 shows examples of the outputs of the Mediapipe + OpenPose approach, which uses Mediapipe for fast skeleton tracking (not displayed), and

OpenPose for hand recognition. Figures 4 and 5 also show examples of the outputs of the object recognition modules using the trained model in YOLO.

Fig. 3. Airplane toy used as assembly task.

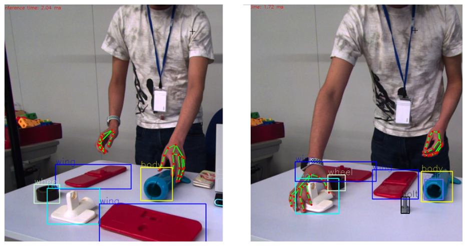


Fig. 4. Left and right images represent examples of hand recognition using Mediapipe and object recognition using the trained YOLO model

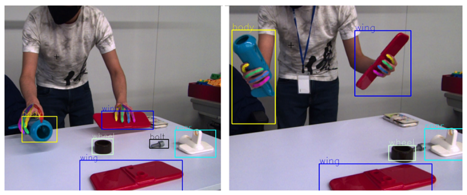


Fig. 5. Left and right images represent examples of hand recognition using Mediapipe + OpenPose module and object recognition using YOLO

D. 3D reconstruction of human body and hands

This module uses the Linear-Eigen method [33] for enabling the 3D reconstruction of the human body and hands from 2D key-points published by the image processing module. In this method the expression $\boldsymbol{x}_i = \boldsymbol{P}_i\boldsymbol{X}$ is used to represent the mapping between a 3D space point $\boldsymbol{X}$ and their 2D correspondence in the i-th image. This 2D correspondence is defined in homogeneous coordinates as $\boldsymbol{x}_i = w(v_i, v_i, 1)^T$, where v_i and u_i are the 2D position in the $i - th$ image and w a scale factor. This mapping is performed by the camera projection matrix $\boldsymbol{P}_i$ which is derived from the camera extrinsic and intrinsic parameters. To simplify calculation $\boldsymbol{P}_i$ is often represented as

$$\boldsymbol{P}_i = \begin{bmatrix} \boldsymbol{p}_i^{1T} \\ \boldsymbol{p}_i^{2T} \\ \boldsymbol{p}_i^{3T} \end{bmatrix} \left(\mathbb{R}^{3\times 4}\right). \tag{1}$$

Consequently the expression $\boldsymbol{x}_i = \boldsymbol{P}_i\boldsymbol{X}$ can be represented as

$$\begin{aligned} wu_i &= \boldsymbol{p}_i^{1T}\boldsymbol{X} \\ wv_i &= \boldsymbol{p}_i^{2T}\boldsymbol{X} \\ w &= \boldsymbol{p}_i^{3T}\boldsymbol{X} \end{aligned} \tag{2}$$

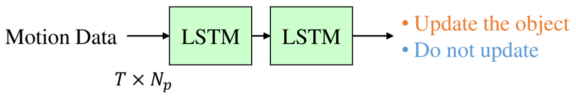


Fig. 6. Identification of in-hand object using LSTM.

Equation 2 is then reformulated to create a $\boldsymbol{AX} = 0$ system by eliminating the scaling factor w, the system becomes

$$\begin{aligned} u_i\boldsymbol{p}_i^{3T}\boldsymbol{X} - \boldsymbol{p}_i^{1T}\boldsymbol{X} &= 0 \\ v_i\boldsymbol{p}_i^{3T}\boldsymbol{X} - \boldsymbol{p}_i^{2T}\boldsymbol{X} &= 0 \end{aligned} \tag{3}$$

The objective in the Linear-Eigen method is to solve the homogeneous linear equations system that is given by the expression $\boldsymbol{AX} = 0$ subject to the constraint $\|\boldsymbol{X}\| = 1$. This can be done by performing the singular value decomposition on the matrix $\boldsymbol{A}$. More details are described in [33]. Outputs of this module are the 3D position of body and human hands for posterior data visualization and processing of some monitoring or robotics system.

V. Assembly Action Recognition Using LSTM

Previous work [2] used Hidden Markov Models (HMMs) to recognize assembly actions on segmented motion data, i.e. the assembly sequence recorded had to be segmented into single actions. As HMMs have few parameters, their ability to represent and thus recognize assembly actions with enough precision is limited. In contrast, Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) [34] have a higher ability to represent sequences of data.

The action recognition using LSTM is performed for each time frame, therefore there is no need of segmenting the assembly sequence into single actions, as it is the case when using HMMs. We use two modules of LSTM, one for identifying the in-hand manipulated object and the other for action recognition.

The pair of in-hand manipulated objects are identified similarly to [2]. In this work, instead of using HMMs, we use two layers of LSTM (as shown in figure 6) to determine whether the recognized object (as explained in section IV-C) should be updated or not, at each time frame for its input to the second LSTM module. The object will be updated to that whose recognized bounding box has one or more finger's inside it (as shown in figure 7), if the action is: free, pick or release; and the object will not be updated if the action is hold or manipulate.

The action recognition module is shown in figure 8. In this module, we also employ two layers of LSTM for action recognition. The number of layers was determined experimentally. The first LSTM layer has the following three inputs:

1) Sequence of time frames. From the time we want to recognize the action to the T width past frames of data (N_p dimension). In this work, we use $T = 10$ (about 1 second). Each time frame contains the following data:

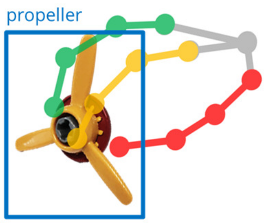


Fig. 7. Human's fingers position inside the recognized bounding box of an object.

- Joints angles of Thumb, Index, and Middle fingers
- Norm of the wrist velocity
- The average of the fingers' velocity norm
- Opening width of the fingers and its change

2) The pair of identified objects in-hand (output of the first LSTM module), with the addition of the label "nothing in-hand", express as a one-hot vector of size N_o.
3) The possible actions selected based on the Action Relationship, with the addition of the label "none", express as a vector of size N_a. The selected actions are represented with 1 and the rest are set to 0.

The output of the first layer is the input to the second layer of LSTM. The second layer's output is input to a full convolutional layer, where the Softmax function outputs an N_a sized vector with the likelihoods for each action. The action with the highest likelihood is the action recognition result.

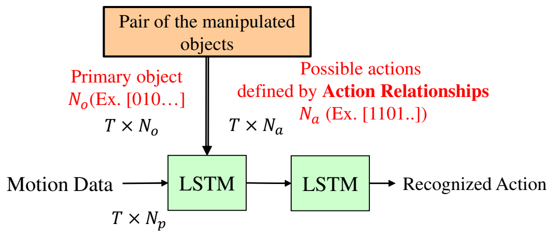


Fig. 8. Action recognition using LSTM.

VI. Validation of the Proposed Software Architecture

One of the main challenges presented in this article is to create a software architecture enabling real-time data processing of human movements from multi-camera systems. Therefore, in this section, we evaluate the performance of the hand recognition modules proposed in this work. Dimension used to evaluate the implemented modules are *inference time (latency)*, and *recognition rates*. Hand recognition approaches compared in this study are: (i) OpenPose in Python (implemented in this article and often used in previous works), (ii) Mediapipe in Python (proposed in this article), and (iii) Mediapipe + OpenPose in Python (proposed in this article). We tested two modules used in Mediapipe to obtain hand key-points from images. The first one is denoted as Mediapipe hands, which was launched in the first release of Mediapipe in July 2020 [26]. The second one is denoted as MediaPipe holistic [35], which performs simultaneous hand, face, and body recognition and was recently launched in December 2020. We used a laptop Alienware M15 with a 2.60 GHz Intel(R) Core(TM) i7-107050H CPU with 16 GB of RAM and an Nvidia GeForce RTX 2070 with 8 GB GDDR6 in Windows 10 for testing. Table I shows the results obtained for latency for the different methods integrated. We measured the time required to process images in a single-camera ($n = 1$) system, a stereo-camera ($n = 2$) system, and a multiple-camera system with $n = 3$, where n represents the number of images that must be processed simultaneously. Values of table I represent the mean value in seconds after simultaneously processing 100 images (from each camera) in the proposed systems. It is relevant to highlight that an "out of memory error" message appeared when using OpenPose for both body and hand recognition when the number of images was $n = 3$. This is due to the amount of GPU memory required for processing each image is approximately 3.5 GB. Therefore, to execute a multi-camera system of more than 3 cameras, it will be required a more powerful GPU with at least 12 GB of GPU memory.

Recognition rates for the implemented hand recognition approaches are shown in table II. This table shows the recognition rates (from 0 to 1) for the left and right hands. These results were obtained from a total of 610 images of a user performing the assembly of the airplane toy shown in figure 3. Additionally we measured the latency presented by the object recognition module using the same set of images. This module was tested with the YOLO model trained to recognize the assembly parts of the airplane toy. The mean latency value obtained was 0.326 seconds when using the CPU pre-compiled version of OpenCV, and a mean of 0.063 seconds when using an OpenCV version compiled from source to enable the use of GPUs.

Results of table I and II indicate that the implemented approaches using Mediapipe provide acceptable performance in comparison to those using OpenPose. While Mediapipe hands and Mediapipe holistic modules present in general better latency, OpenPose, and Mediapipe + OpenPose solutions provided better recognition rates. As shown in table II, recognition rates obtained with OpenPose were 0.91 and 1.0 seconds for the left and right hands respectively. These results outperform results obtained from Mediapipe hands (which obtained 0.66 and 0.78) and holistic (which obtained 0.75 and 0.99) methods. However, OpenPose requires very expensive hardware to enable real-time processing. As an example, we consider the case where one camera is used ($n = 1$). With this configuration and using OpenPose for body and hand recognition, we obtained a latency of 0.2131. This suggests that the used hardware could barely get 4.7 frames per second (fps). This frame rate can be increased using the Mediapipe + OpenPose option to approximately 7.3 fps, which latency results were 0.1347 seconds. These values are much lower than the current video standard frame speed of 29.97 fps. On the other hand, we obtained much lower

TABLE I

LATENCY COMPARISONS OF IMPLEMENTED APPROACHES FOR HAND RECOGNITION. OPENPOSE REPRESENTS THE STATE-OF-THE-ART SOLUTION.

Method	$n = 1$	$n = 2$	$n = 3$
OpenPose (body + hands)	0.2131	0.4125	error
Mediapipe (hands)	0.0250	0.0268	0.0283
Mediapipe holistic (hands)	0.0367	0.0381	0.0469
Mediapipe (body) + OpenPose (hands)	0.1347	0.2403	0.3476

TABLE II

RECOGNITION RATES OF IMPLEMENTED APPROACHES FOR HAND RECOGNITION WITH $n = 1$

Method	Left hand	Right hand
OpenPose hands	0.91	1.00
Mediapipe hands	0.66	0.78
Mediapipe holistic	0.75	0.99

TABLE III

COMPARISON OF ACTION RECOGNITION RESULTS.

	In-hand object identification	Accuracy	
		obj. identification	obj. Ground Truth
HMM	61.5%	39.2%	71.6%
LSTM	69.2%	54.0%	85.1%

latency results using the module that implements Mediapipe to directly perform the hand detection. Results of Mediapipe hands were 0.025 (40 fps) when using one camera and 0.0283 (35.3 fps) when using images from 3 cameras at the same time. Results of Mediapipe holistic were 0.0367 (27.2 fps) when using one camera and 0.0469 (21.32 fps) when using images from 3 cameras. These results suggest that Mediapipe implementations can be better suited for applications requiring real-time data processing and when the computational resources available are limited.

VII. VALIDATION OF THE PROPOSED LSTM NETWORK

A. Comparison between HMM and LTSM for action recognition

To demonstrate the validity of the proposed action recognition framework using LSTM, we also use HMMs for action recognition as in [2]. We recorded the data of two subjects (five times per subject) assembling the toy airplane (figure 3) for training both the HMMs and the LSTM. In the case of the LSTM, we used Gaussian Noise (average 0 and distribution $5[\mathrm{mm}]$) as data augmentation of the training data (added two sets of data). For validation, we recorded the assembly motions of five no pre-trained subjects (twice per subject) different from those recorded for training (total of 10 assembly sequences). We divided the recorded sequence into single motions, labeled them, and annotated the manipulated object of each hand to create the Ground Truth of each sequence to compute the accuracy (percentage of correctly recognized frames) of both methods. As mentioned before, the LSTM performs the action recognition per time frame, thus we do not need to perform a segmentation of the assembly sequence, which is needed for the HMM. Table III summarizes the obtained recognition results, where "obj. identification" denotes that the pair of manipulated objects is obtained through object recognition, and "obj. Ground Truth" means that we give the correct manipulated object as input, all of them when using OpenPose only.

As it can be seen, both the in-hand object identification and the action recognition accuracy when using LSTM (69.2% and 85.1%, respectively) is better than the accuracy obtained when using HMMs (61.5% and 71.6%, respectively). Even when we give the correct sequence segmentation of the actions to the HMM the accuracy is 80.5%, which is still below the accuracy when using LSTM. Therefore, for the following analysis we only use LSTM.

B. Comparison between OpenPose and Mediapipe for action recognition using LSTM

The proposed LSTM model was trained and validated offline using the 3D information of hand motions produced by OpenPose. However, as shown in section VII-B, the online processing of data using OpenPose for multi-camera systems requires very expensive hardware resources. In order to test the robustness of the proposed LSTM model, we use a set of images ($n = 4$) of one experimental subject performing an assembly task as inputs to the module developed using Mediapipe for hand detection. These results obtained using Mediapipe hands and the modules defined in section IV are used as inputs to the LSTM model. With this process, we try to understand how much can the use of data from other libraries for hand recognition (which are more efficient but less robust) affects the recognition rates of our LSTM model. It is relevant to highlight that the images used in this evaluation were not used for training the LSTM model. We compare the outputs of the LSTM using the same experimental data when human hands are obtained from OpenPose and Mediapipe. While the accuracy reached using data from OpenPose was 63.04%, the accuracy when using Mediapipe was 51.02%. These preliminary results were as expected, since OpenPose provides more robust recognition capabilities. Even with less accurate data provided by Mediapipe, as shown in table II, the difference in recognition was less than 15%.

VIII. CONCLUSION

We proposed a set of modules enabling data collection and analysis of assembly motions. Unlike most previous works our system only uses images. Since this does not represent an additional burden to human motions, our approach can be more suitable to enable natural interactions in HRI and HRC scenarios. We proposed different solutions to enable the real-time data processing of human motions. The suitability of these solutions will depend on the available computational resources as well as the required accuracy and frame rate. While a solution based on Mediapipe could offer acceptable accuracy with less computational resources, which can be suitable for real-time data processing, OpenPose would provide better recognition rates. However, due to their more expensive requirements, solutions using OpenPose could be in many cases better suited for offline data processing. We also demonstrated that using LSTMs yields better action recognition than HMMs, and as it does not need to segment the sequence of motions into simple actions the recognition process is faster and accurate. In the future, we would like to test using more training data for the LSTM, and also search for other possible combinations of motion data that could speed off the recognition without sacrificing much accuracy.

Acknowledgment

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] N. Yamanobe, W. Wan, I. G. Ramirez-Alpizar, D. Petit, T. Tsuji, S. Akizuki, M. Hashimoto, K. Nagata, and K. Harada, "A brief review of affordance in robotic manipulation research," *Advanced Robotics*, vol. 31, no. 19-20, pp. 1086–1101, 2017.

[2] K. Fukuda, N. Yamanobe, I. G. Ramirez-Alpizar, and K. Harada, "Assembly motion recognition framework using only images," in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 1242–1247.

[3] C. Chen, T. Wang, D. Li, and J. Hong, "Repetitive assembly action recognition based on object detection and pose estimation," *Journal of Manufacturing Systems*, vol. 55, pp. 325–333, 2020.

[4] K. Darvish, F. Wanderlingh, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, "Flexible human–robot cooperation models for assisted shop-floor tasks," *Mechatronics*, vol. 51, pp. 97–114, 2018.

[5] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: A review," *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, 2018.

[6] G. V. Kale and V. H. Patil, "A study of vision based human motion recognition and analysis," *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 7, no. 2, pp. 75–92, 2016.

[7] N. Noceti, A. Sciutti, and F. Rea, "Modelling human motion."

[8] M. E. Segura, E. Coronado, M. Maya, A. Cardenas, and D. Piovesan, "Analysis of recoverable falls via microsoft kinect: Identification of third-order ankle dynamics," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, 2016.

[9] P. Maurice, A. Malaisé, C. Amiot, N. Paris, G.-J. Richard, O. Rochel, and S. Ivaldi, "Human movement and ergonomics: An industry-oriented dataset for collaborative robotics," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1529–1537, 2019.

[10] E. Coronado, J. Villalobos, B. Bruno, and F. Mastrogiovanni, "Gesture-based robot control: Design challenges and evaluation with humans," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2761–2767.

[11] T. Mori, Y. Nejigane, M. Shimosaka, Y. Segawa, T. Harada, and T. Sato, "Online recognition and segmentation for time-series motion with hmm and conceptual relation of actions," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3864–3870.

[12] F. M. Noori, B. Wallace, M. Z. Uddin, and J. Torresen, "A robust human activity recognition approach using openpose, motion features, and deep recurrent neural network," in *Scandinavian conference on image analysis*. Springer, 2019, pp. 299–310.

[13] V. Hernandez, T. Suzuki, and G. Venture, "Convolutional and recurrent neural network for human activity recognition: Application on american sign language," *PloS one*, vol. 15, no. 2, p. e0228869, 2020.

[14] P. Neto, M. Simão, N. Mendes, and M. Safeea, "Gesture-based human-robot interaction for human assistance in manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 101, no. 1, pp. 119–135, 2019.

[15] K. Fukuda, I. G. Ramirez-Alpizar, N. Yamanobe, D. Petit, K. Nagata, and K. Harada, "Recognition of assembly tasks based on the actions associated to the manipulated objects," in *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2019, pp. 193–198.

[16] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R. Zaccaria, "Analysis of human behavior recognition algorithms based on acceleration data," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1602–1607.

[17] P. K. Murali, K. Darvish, and F. Mastrogiovanni, "Deployment and evaluation of a flexible human–robot collaboration model based on and/or graphs in a manufacturing environment," *Intelligent Service Robotics*, vol. 13, no. 4, pp. 439–457, 2020.

[18] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: a review of techniques," *journal of Imaging*, vol. 6, no. 8, p. 73, 2020.

[19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[20] J. Akira, T. Fujii, S. Sato, and H. Nakahara, "An fpga realization of openpose based on a sparse weight convolutional neural network," in *2018 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2018, pp. 310–313.

[21] M. Al-Amin, W. Tao, D. Doell, R. Lingard, Z. Yin, M. C. Leu, and R. Qin, "Action recognition in manufacturing assembly using multimodal sensor fusion," *Procedia Manufacturing*, vol. 39, pp. 158–167, 2019.

[22] J. Lauener, W. Sliwinski *et al.*, "How to design & implement a modern communication middleware based on zeromq," in *Proc of ICALEPCS*, vol. 17, 2017, pp. 45–51.

[23] E. Coronado and G. Venture, "Towards iot-aided human–robot interaction using nep and ros: A platform-independent, accessible and distributed approach," *Sensors*, vol. 20, no. 5, p. 1500, 2020.

[24] A. Dworak, F. Ehm, P. Charrue, and W. Sliwinski, "The new cern controls middleware," in *Journal of Physics: Conference Series*, vol. 396, no. 1. IOP Publishing, 2012, p. 012017.

[25] Google, "Mediapipe official page," https://mediapipe.dev/, [Online; accessed 01-01-2021].

[26] ——. Mediapipe releases. [Online; accessed 01-01-2021]. [Online]. Available: https://github.com/google/mediapipe/releases

[27] C. P. C. Lab. Openpose github. [Online; accessed 01-01-2021]. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose

[28] D. Groos, H. Ramampiaro, and E. A. Ihlen, "Efficientpose: Scalable single-person pose estimation," *Applied Intelligence*, pp. 1–16, 2020.

[29] D. Osokin, "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose," in *arXiv preprint arXiv:1811.12004*, 2018.

[30] J. Redmon, S. Divvala, R. Girshick, and 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.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[32] K. He, G. Gkioxari, and P. G. Dollár, "R., 2017. mask rcnn," in *2017 IEEE International Conference on Computer Vision*, 2017, pp. 1440–1448.

[33] J. Chen, D. Wu, P. Song, F. Deng, Y. He, and S. Pang, "Multi-view triangulation: Systematic comparison and an improved method," *IEEE Access*, vol. 8, pp. 21 017–21 027, 2020.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] Google, "Mediapipe holistic," https://ai.googleblog.com/2020/12/mediapipe-holistic-simultaneous-face.html, [Online; accessed 01-01-2021].