Proceedings of the ASME 2020 International Mechanical Engineering Congress and Exposition
IMECE 2020
November 13-19, 2020, Portland, USA

IMECE-21436

DESIGN OF A REAL-TIME HUMAN-ROBOT COLLABORATION SYSTEM USING DYNAMIC GESTURES

Haodong Chen*, Ming C. Leu, Wenjin Tao
Department of Mechanical and Aerospace Engineering
Missouri University of Science and Technology
Rolla, MO 65409, USA

Zhaozheng Yin
Department of Biomedical Informatics &
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA

ABSTRACT

With the development of industrial automation and artificial intelligence, robotic systems are developing into an essential part of factory production, and the human-robot collaboration (HRC) becomes a new trend in the industrial field. In our previous work, ten dynamic gestures have been designed for communication between a human worker and a robot in manufacturing scenarios, and a dynamic gesture recognition model based on Convolutional Neural Networks (CNN) has been developed. Based on the model, this study aims to design and develop a new real-time HRC system based on multi-threading method and the CNN. This system enables the real-time interaction between a human worker and a robotic arm based on dynamic gestures. Firstly, a multi-threading architecture is constructed for high-speed operation and fast response while schedule more than one task at the same time. Next, A real-time dynamic gesture recognition algorithm is developed, where a human worker's behavior and motion are continuously monitored and captured, and motion history images (MHIs) are generated in real-time. The generation of the MHIs and their identification using the classification model are synchronously accomplished. If a designated dynamic gesture is detected, it is immediately transmitted to the robotic arm to conduct a real-time response. A Graphic User Interface (GUI) for the integration of the proposed HRC system is developed for the visualization of the real-time motion history and classification results of the gesture identification. A series of actual collaboration experiments are carried out between a human worker and a six-degree-of-freedom (6 DOF) Comau industrial robot, and the experimental results show the feasibility and robustness of the proposed system.

Keywords: Human-robot collaboration, Real-time System, Motion History Image, Multiple Threads

1. INTRODUCTION

Human-robot collaboration (HRC) systems become one of the promising solutions towards flexible automation in the context of Industry 4.0. It has been shown that in the industry with a high degree of automation, the HRC system can increase production efficiency and also provide more flexibility in the work environment. Collaborative human-robot workspaces, which combine flexibilities of humans and productivities of robots, are increasingly gathering attention in both manufacturing industries and research communities [1–3]. Many theories, methodologies, and applications concerning collaborative processes between human workers and robots were designed and developed. In 2013, a finite state automaton system for structuring the collaborative behavior of industrial robots was designed for safety and productivity problems in the HRC system [4]. In 2019, a study focusing on fluency in HRC providing an analytical model for four objective metrics was published [5]. There were also some other related research

*Corresponding author, Email address: hc373@mst.edu

results for the HRC system, such as the hand-guidance [6], vocal command controlling [7], task teaching [8], collision avoidance [9], and task time optimization [10].

Ideally, an HRC system should work similarly to human-human collaboration on the factory floor. However, it is challenging to develop such HRC applications with desired productivity because of the space-separation and the time-discontinuity of workers and robots. In the limited communication channels between human workers and industrial robots, gesture communication has been effectively applied, thus robots need to understand and respond to human gestures correctly in order to collaborate with human workers seamlessly. In the upper limbs of a human being, the shoulder motion has three degree of freedoms (DOFs) (i.e., abduction/adduction, flexion/extension, and internal/external rotation), the elbow joint allows two DOFs (i.e., flexion/extension and supination/pronation), the wrist joint has two DOFs (i.e., flexion/extension and radial/ulnardeviation), and each hand has twenty-one DOFs (i.e., flexion/extension and abduction/adduction) [11]. All these together realize that the human upper limb can conduct an infinite number of possible paths even for a simple task, and this redundancy feature of the DOF can be taken as a beneficial feature because it provides more flexibility in the gesture-based HRC [12]. In 2011, a designed gesture recognition system was constructed by T. Ende, et al. [13]. It could gather their designed gestures for the HRC system and the recognition rate was over 80 %. Later, In 2018, a set of robust gestures was designed by Islam et al., which could be used for a diver to control an underwater robot in collaborative task execution [14]. Some studies attempted to exploit the motion capture, recognition, and prediction. In 2014, a novel gesture capture and recognition method was developed, which segmented the palm and finger pixels and obtained impressive performance in the recognition [15]. In 2018, a long motion (16 s) prediction technique was created by V. Unhelkar et al., and this method could realize motion recognition with a prediction time horizon up to six seconds [16].

To obtain a better extraction of features and a higher accuracy of recognition, many researchers focus on the applications of deep learning methods on identification tasks. For example, the convolutional neural network (CNN) is applied in diagnostics to make health management decisions [17]. In [18], the authors adopted CNN for rotating machinery condition monitoring. A discrete Fourier Transform of the two accelerometers was calculated as the input to the network. Similarly, another deep learning model was designed in 2017 for the bearing fault diagnosis, in which the wavelet packet energy images were generated as inputs [19].

This paper presents a real-time human-robot collaboration system with the ability to detect and recognize designed dynamic gestures of a human worker in real time and control a robotic arm based on gesture recognition results automatically.

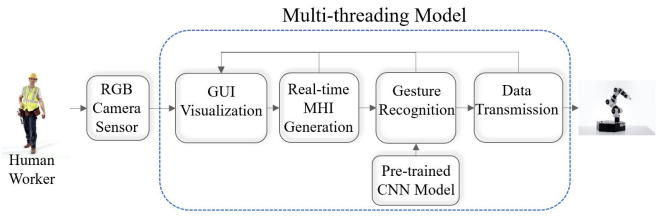


FIGURE 1: SYSTEM OVERVIEW.

An overview of the overall system is shown in Fig. 1. This system is based on the construction of multi-threading, which can realize the execution of multiple threads (the graphic user interface (GUI) running, the real-time Motion History Image (MHI) generation, the simultaneous gesture recognition, and the data transmission) concurrently. The first thread is constructed for a designed GUI, which realizes the straightforward control and operation of the other three threads and related systemic activities. For the system input, the second thread is designed to capture movements of a human worker by a camera and generate the extract features of the movements constantly. A real-time frame-based MHI method is developed for the dynamic feature extraction, which can update a new MHI when a new frame is input. Meanwhile, the third thread utilizes the MHIs as inputs of a deep learning model to carry out dynamic gesture recognition. Besides, the fourth thread is started at the same time to construct a real-time updating queue of recognition results and transform it into a robotic arm. The robotic arm can response with corresponding movements. The main contribution of this paper is design of the real-time MHI algorithm, construction of the multi-threading framework, and integration of the new HRC system.

This paper is organized as follows. In section 2, our previously designed work for this HRC system is presented, including ten designed dynamic HRC gestures and a deep learning model for gesture recognition. Section 3 provides an overview of the multi-threading construction and details of the real-time system strategy. Section 4 demonstrates a six-degrees-of-freedom (6 DoF) robotic arm of the system. Section 5 conducts the system integration and shows the system applications on the robotic arm. Finally, in section 6 provides the conclusions.

2. PREVIOUS WORK

In the previous work [20], a deep learning model of a set of designed dynamic HRC gestures was built based on CNN for gesture recognition. Ten arm-based dynamic gestures are designed based on Iconic and Deictic rules [21]. Fig. 2 gives the gesture $left$ as an example of these dynamic gesture. The yellow shows the gesture trajectory, which is swinging the left arm straight out and up to the side with the index finger

FIGURE 2: THE DYNAMIC GESTURE *LEFT*.

extended until the arm reaches the shoulder level. The recognition accuracy based on the pre-trained CNN model is higher than 99.8%.

To extract features of dynamic gestures and construct the dataset for the CNN model, The MHI approach is adopted to transform a gesture video clip into a static scalar-valued image where more recently moving pixels are brighter and vice-versa. A video-based MHI H_τ (x, y, t) can be obtained from an update function Ψ (x, y, t) using the following formula:

$$H_\tau(x,y,t) = \begin{cases} \tau & \text{if } \Psi(x,y,t) = 1 \\ \max(0, H_\tau(x,y,t-1) - \delta)) & \text{otherwise} \end{cases} \quad (1)$$

where x and y are the image pixel coordinates and t is time. Ψ (x, y, t) represents the movement of an object in the current video frame, the duration τ denotes the temporal extent of a movement, and δ is the decay parameter. Fig. 3 shows the process of a video-based MHI generation. It can be seen that an MHI is obtained from binary images of the sequential frames of the input video. The binary images are generated using the frame subtraction:

$$\Psi(x,y,t) = \begin{cases} 1 & \text{if } |I(x,y,t) - I(x,y,t-\triangle)| \geq \xi \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where the $\Psi(x,y,t)$ represents the binary image, and ξ is a threshold. The threshold ξ is used to eliminate the background noise in the MHIs. The $I(x,y,t)$ is the intensity value of pixel location with the coordinate (x, y) at the tth frame of the image sequence. $\triangle$ is the temporal difference between two pixels at the same location but at different times [22].

3. MULTI-THREADING CONSTRUCTION

For a real-time system, it should be designed with the abilities of high-speed operation and fast response while can schedule more than one task and manage limited resources at the same time. All the functions and algorithms in the real-time operation should be isolated and free of interference from

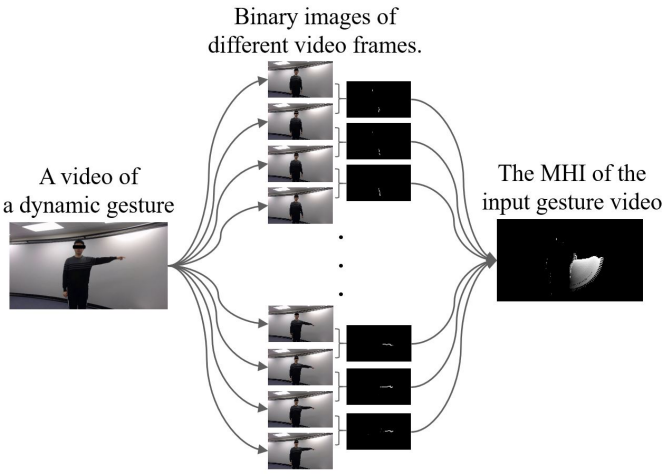


FIGURE 3: THE GENERATION PROGRESS OF A VIDEO-BASED MHI.

others [23]. The multi-threading is one solution for this issue, which is the process of executing multiple threads simultaneously with one central processing unit (CPU). A thread is the smallest execution sequence of programmed instructions that can be managed independently by a schedule [24]. In this section, the multi-threading model designed in building the real-time HRC system is illustrated. Fig. 4 shows the structure and function of the multi-threading model of this system. In this model, four threads are built for different parallel tasks, including GUI operation (GUI thread), real-time MHI generation (MHI thread), real-time gesture recognition (recognition thread), and data transmission (transmission thread). To satisfy the requirements of the real-time system operation, all these four threads can carry out their own designed tasks constantly and concurrently. Meanwhile, these threads can utilize outputs of other threads saved in a queue, which is a linear data structure that stores items in the First In First Out (FIFO) manner [25].

As shown in Fig. 4, when the system is started, the GUI thread always stays active and waits for the operation of users. For the recognition thread and the transmission thread, only when the MHI thread generates at least one image, they are activated. Otherwise, these two threads stay waiting with empty queues (the Queue1 and Queue2), which are constructed for constantly ordered inputs. Once the MHI thread is triggered by the start button of the GUI thread. The recognition thread is called up firstly, and the transmission thread would follow it closely, and then all four threads are in the operation mode at the same time. The MHI thread reads the input frames of the camera, and each new frame triggers a new MHI. The newest MHI is shown on the display window of the GUI with the current frame and also be saved in an image queue (Queue 1)

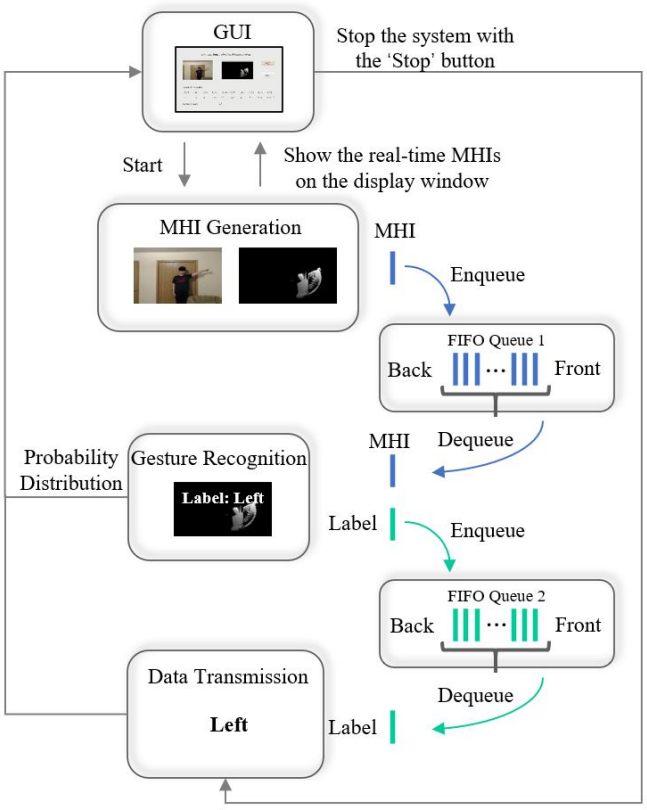


FIGURE 4: THE MULTI-THREADING MODEL OF THE REAL-TIME HRC SYSTEM.

for the recognition thread. At the same time, The recognition thread reads the data in the image queue (Queue 1) and call the pre-trained CNN model to recognize the gesture and output the recognition result, i.e., the probability distribution and the gesture label, on the GUI. A label queue (Queue 2) is used to save the labels constantly. Meanwhile, the transmission thread sends the results in the output queue (Queue 2) into the robotic arm with the same sequence, and the robotic arm could give corresponding designed responses.

3.1. GUI Operation

To realize visualization of the system operation, and assistance for human workers without programming skills in the system controlling, a GUI software is designed. As shown in Fig. 5, two buttons, two display windows, and a set of distribution labels are displayed in this GUI. The start button can be pushed to launch the whole system, including the four threads and related devices (camera, robotic arm, etc.). The stop button can be utilized to stop the system operation. The current camera and the newest MHI are updated on the two windows continually. The human worker can check the current behaviors

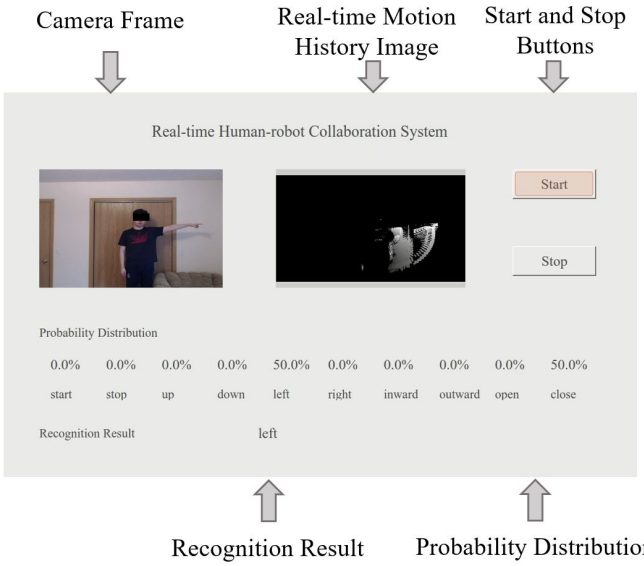
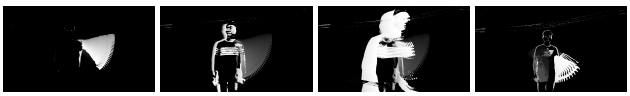


FIGURE 5: THE DESIGNED GUI.

and corresponding updating MHI on the GUI [26–28]. Then the probability distribution of the current gesture and the final recognition result are shown on the bottom of the GUI.

3.2. Real-time MHI Generation

In order to extract the dynamic features of the designed gestures, the MHI method is adopted and modified into a real-time version. An MHI is a kind of temporal template, which is the weighted sum of successive frames, and the weight decay as time is further away. In the previous works, we applied this method to obtain MHIs of the dynamic gesture videos and constructed a data set of the CNN model with these images. However, in a manufacturing stage where a human worker carries out gestures constantly, the performance of the video-based MHI method is limited. One reason is that a video-based MHI generator cannot detect the accurate the start and end time of the real-time movements of a human worker, which means almost all MHIs might be invalid. As shown in Fig. 6, Fig. 6 (a) missed the start of the gesture $left$. Fig. 6 (b) and (c) captured the late end time and recorded irrelevant movements. Fig. 6 (d) mixed two constant gestures together. With these invalid results, the system would suffer great challenges to continue operating properly, i.e., these invalid MHIs could decrease the fault tolerance of the whole system [29]. Another reason is that for a same human worker, the performance time of different gestures are not equal, and for a same gesture, the time cost of different workers might be different. Even for a same gesture, one worker might spend different time on it when it is performed many times. So the



(a) (b) (c) (d)

FIGURE 6: INVALID VIDEO-BASED MHI SAMPLES.

video-based MHI generator is not powerful enough for the real-time situations.

For the solutions to the above problems, the real-time MHI generation method is designed here. Based on Eq. 1, and 2 in section 2, there are mainly three parameters in the process of a video-based MHI (Fig. 3), i.e., the duration time τ, the decay parameter δ, and the threshold ξ. In the real-time MHI generation, the main idea is the transformation of video-based MHI into frame-based MHI while representing the duration time τ with the decay parameter δ during the MHI generation. As shown in Fig. 7, when the camera is started, the input intensity value images $I(x,y,t)$ of input frames are captured continually. After that, as shown in Eq. 3, the binary images $\Psi(x,y,t)$ in the MHI generation are calculated by subtraction of every two adjacent intensity value images. Note that the temporal difference $\triangle$ in Eq. 2 set as 1 frame in Eq. 3, which means every new input frame could be recorded and trigger a new binary image $\Psi(x,y,t)$ as follows:

$$\Psi(x,y,t) = \begin{cases} 1 & \text{if } \mid I(x,y,t) - I(x,y,t-1) \mid \geq \xi \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\Psi(x,y,t)$ represents the binary image, and ξ is the threshold. The threshold ξ is used to eliminate the background noises in the MHIs. $I(x,y,t)$ is the intensity value of pixel location with the coordinate (x, y) at the tth frame of the image sequence [22].

Later, the program collects the new binary image $\Psi(x,y,t)$ into the constant updating real-time MHI continuously. Meanwhile, the value of all previous pixels in the updating MHI are reduced by δ, until it is zero. In an 8-bit gray scale image, the value of the pixel is between 0 and 255. Typically zero is taken to be black, and 255 is taken to be white. [30]. This step generates the scalar-valued feature of the MHI, where more recently moving pixels are brighter and vice-versa. More importantly, the duration time τ of all input binary images in the real-time MHI can be denoted by the decay parameter δ. In the binary image, the value of movement pixels are set as 255 (white) in the first time input, and each new input frame updates the real-time MHI with adding a new binary image $\Psi(x,y,t)$ and reduce the value of previous pixels by δ. After $\lceil 255/\delta \rceil$ (the rounded up result of $255/\delta$) times decay, the white pixels

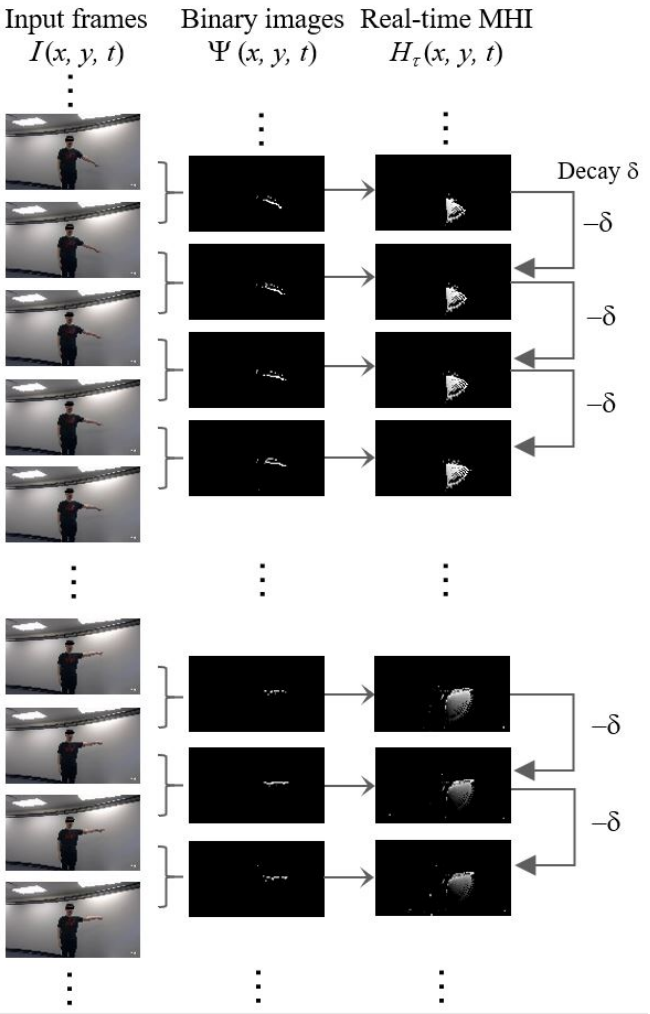


FIGURE 7: THE GENERATION PROCESS OF THE REAL-TIME MHI.

become black. So the duration time τ can be set as $255/\delta$ frames. Note that the frame rate of the camera in this system is set as 30 frames per second. Finally, the real-time MHI can be obtained as follows:

$$H_\tau(x,y,t) = \begin{cases} \tau \ (\tau = 255/\delta) & \text{if } \Psi(x,y,t) = 1 \\ \max(0, H_\tau(x,y,t-1) - \delta)) & \text{otherwise} \end{cases} \tag{4}$$

In the above procedure, the threshold ξ is set as 10 based on validation experiments in the previous work [20]. With regard to the duration time τ, the lengths of all the gesture videos in the data set are calculated and the obtained results are as shown in Table. 1. It can be seen that the biggest value of the length is equal to or less than 35 frames (frame rate 30 frames/second). For the purpose of recording all gestures completely, the duration

TABLE 1: DISTRIBUTION OF THE DURATION TIME IN THE DATA SET.

Threshold τ value (integer)	20∼25	25∼30	30∼35
Number of the videos	495	2410	1665

time τ is set as 35 frames in this system. Based on Eq. 5, the decay parameter δ can be calculated as 7.29 to satisfy the temporal extent of movements in the real-time MHI. After all parameters are set, and the real-time MHI in this system can be calculated as:

$$H_\tau(x,y,t) = \begin{cases} \tau = 35 frames & \text{if } \Psi(x,y,t) = 1 \\ \max(0, H_\tau(x,y,t-1)) - 7.29) & \text{otherwise} \end{cases} \quad (5)$$

In comparison with the video-based MHI creation method, the advantages of the real-time MHI generator designed above are obvious. First of all, it realizes the real-time and continuous MHI generation of current frames. Moreover, one new input frame can trigger a new MHI with a new binary image means that every input frame is recorded in the real-time MHI, i.e., every movement (more than one frame) of the human worker can update more than one similar MHI, and which is of great importance in supplying the higher fault tolerance and recognition accuracy than the video-based MHI for the later real-time gesture identification.

The pseudo-code of the real-time MHI generation is shown as Algorithm 1.

Algorithm 1 Real-time MHI Generation

Input: *P-img* /*Previous input frame image*/, *F-img* /*Newest input frame image*/, τ /*Duration time*/, ξ /*Threshold*/;
Output: *F-MHI* /*Frame-based real-time MHI*/;

```
function REAL-TIME MHI(P-img, F-MHI)
    F-MHI ← O /*Initialization of the F−MHI*/;
    δ ← ⌈255/δ⌉ /*Obtain of the decay parameter*/;
    for each F-img do
        diff-img ← |F-img − P-img|;
        F-MHI ← max(0, F-MHI − δ);
        F-MHI(diff-img > ξ) ← 255;
        P-img ← F-img;
    end for
end function
```

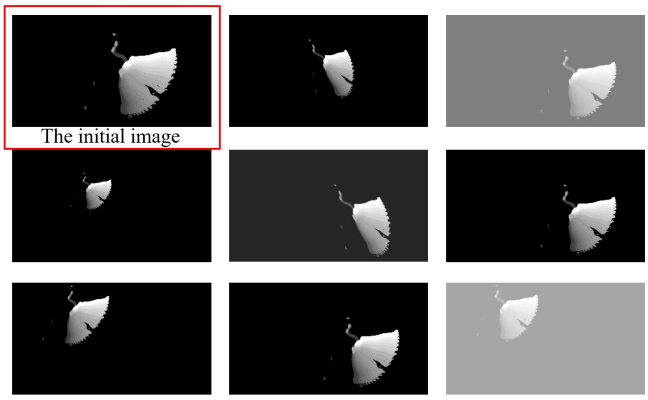


FIGURE 8: SAMPLES OF AUGMENTED MHIS.

3.3. Real-time Gesture Recognition

Since the MHIs can be created in real time, the recognition thread is developed for the gesture identification with the pre-trained deep learning model in the previous work. In order to increase the variations of images in the training data, two new data augmentation methods are adopted to artificially create new training data together with the brightness change and shift methods, including zooms and perspective transformation. Fig. 8 shows some samples of the augmented images. Note that the first image is the initial image before augmentation, and all the augmentation methods are combined together randomly.

With the CNN model, the recognition thread extracts the MHIs saved in the Queue 1 in Fig. 4 and output distribution of the identification probabilities on the GUI. The probability distribution describes the likelihood of obtaining the possible values that a random variable can assume in the sample space [31]. In the neural network model using the new data set (Fig. 9), It can be seen that the last 10-dimensional score vector layer is transformed by the softmax activation function to output the predicted probabilities. The softmax function is calculated as follows:

$$P(x_i) = \frac{e^{x_i - max(x_k)}}{\sum_{k=1}^{10} e^{x_k - man(x_k)}} \quad for\ i = 1, ..., 10 \quad (6)$$

where $P(x_i)$ is the predicted probability of being class i for sample x, x_k is the weighted inputs of the softmax layer, and 10 is the number of gestures. $\max(x_k)$ is the maximal number of the output layer before the normalization. The subtraction of the maximum in x_k can improve numerical stability. Because an exponential function grows very fast, the exponential result of even a moderate-magnitude positive x_k can be very huge, which makes the scaling sum huge, and dividing by a huge number can cause arithmetic computation problems. A solution to avoid this problem is the subtraction of the $\max(x_k)$ [32].

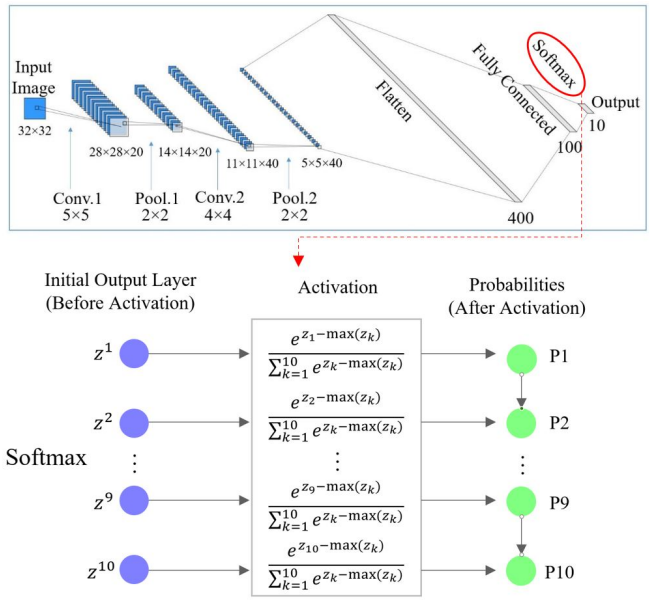


FIGURE 9: THE SOFTMAX LAYER OF THE CNN MODEL. (The 'Conv.' and 'Pool.' denote the operations of convolution and pooling, respectively).

The softmax function can conduct the normalization process through limiting the output of the function into the range 0∼1, and making all the outputs adding up to 1. An impossible event is denoted by 0 and a sure event is denoted by 1 [33].

In the probability distribution above, the uniform distribution leads to the most conservative estimate of uncertainty, i.e., it gives the largest standard deviation. The distribution with a single spike leads to the least conservative estimate of uncertainty, i.e., it gives the largest confidence. Note that with the subtraction of the $\max(x_k)$ in the softmax operation, the distribution of the probabilities might seem less distinct. However, the recognition label can be obtained by the output layer values before the softmax operation, which means the index of the maximum in the initial output layer is identified as the gesture label.

Generally, a movement that belongs to the ten dynamic gestures could be recognized correctly. For the other gestures, the distribution could have a uniform feature. However, it might share a similar distribution with a single spike. At this time, a threshold is set to eliminate the confusing movements to avoid the wrong inputs to the robotic arm. In the pre-trained CNN model, the maximum in the initial output layer are all larger than 10000, so an easy way is to set the threshold as 10000 to filter the invalid gestures before they are input to the label queue (Queue 2 in Fig. 4) of the system.

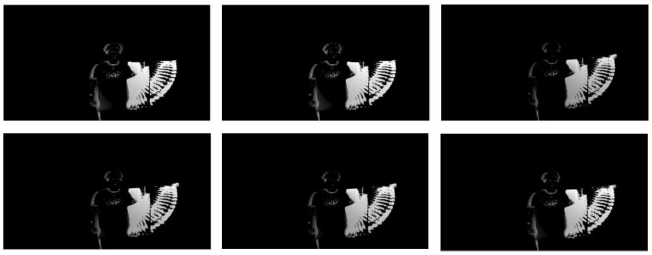


FIGURE 10: SIMILAR ADJACENT MHIS.

3.4. Transmission of Recognition Results

Since the input gestures are recognized and saved into the label queue, the data transmission thread extracts the gesture labels and transmit them into the robotic arm. Note that for one movement, the real-time MHI method might generate several similar adjacent images as shown in Fig. 10. However, the intent of the human worker performing this gesture is to enable the robotic arm to give a one-time response. Considering the duration time τ in the real-time MHI is set as 35 frames, which means that if all 35 constant MHIs are recognized as the same label, these images might be generated from one same input gesture. Concerning this issue, the manner of the data transmission thread can be designed as sending the same gesture label with a frequency of one time per 35 frames. In this way, the robotic arm can execute commands of the human worker accurately. If two same gestures are given constantly, the same response could be carried out twice without missing any control commands.

4. DESIGN OF ROBOTIC ARM RESPONSES

In order to interact with human workers, designated responding robot movements are matched with ten dynamic gestures. During the collaboration, the robotic arm gives corresponding responses for different input gestures. In the ten dynamic gestures designed before, there are mainly two types, i.e., calibration and operation gestures. The calibration gestures include some routine procedures of industrial robots (e.g., *start* - initialization, and *stop* - emergency stop). Specifically, the *start* gesture controls the robot to conduct the calibration, which is the initialization process of identifying certain parameters in the kinematic structure of a robot, such as the relative position of robot links. When the calibration command is given by *start* gesture of the human worker, the robotic arm executes with a specific start pose. The *stop* gesture can stop the movement of the robot and let it stay in the current pose. In terms of the operation gestures, they can instruct the robot to move the end-effector along the six directions (up/down, left/right, inward/outward) of the workspace, or control the gripper (a common end-effector of a robotic am) to open/close. As the response of an operation gesture input, the robotic arm would

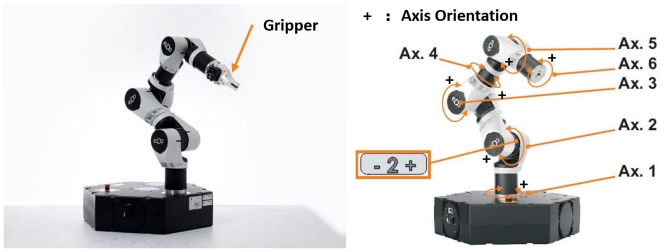


FIGURE 11: THE E.DO ROBOTIC ARM AND POSITIVE ORIENTATIONS OF SIX JOINTS.

conduct the kinematics calculation and the end-effector would move a designed distance along the related direction until the end-effector reaches the edge of the workspace, or any joint angle reaches the limitation.

5. SYSTEM INTEGRATION AND APPLICATIONS

5.1. Robotic Parameters

In this section, using a 6-DOF e.DO robotic arm, we have conducted the system integration and applications of our proposed real-time human-robot collaboration system. This e.DO robotic arm is equipped with a gripper as the end-effector, and the axis directions are shown in Fig. 11. The parameters of the robotic arm are shown in Table. 2, including the ranges of six joints.

TABLE 2: RANGE OF THE SIX JOINTS.

Axis Label	Stroke/(degree)	Maximum Speed/(deg/s)
Axis 1	+/- 180 deg	22.8 deg/s
Axis 2	+/- 99 deg	22.8 deg/s
Axis 3	+/- 99 deg	22.8 deg/s
Axis 4	+/- 180 deg	33.6 deg/s
Axis 5	+/- 104 deg	33.6 deg/s
Axis 6	+/- 180 deg	33.6 deg/s

5.2. System Application

The proposed real-time HRC system is tested to demonstrate its feasibility. The platform is Ubuntu 16.04 on a laptop with NVIDIA GTX 850 M and main memory 8G bytes. Logitech HD Webcam C920 is used to capture gestures. The

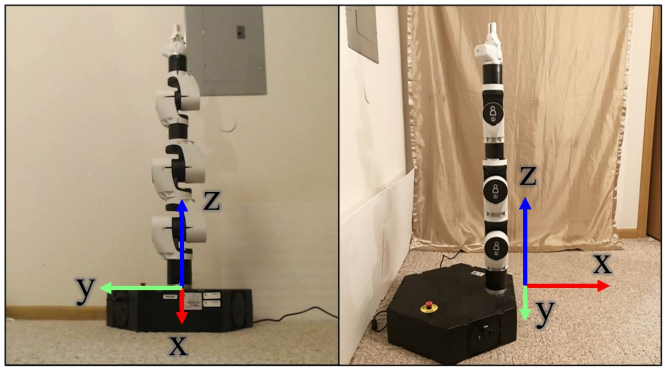
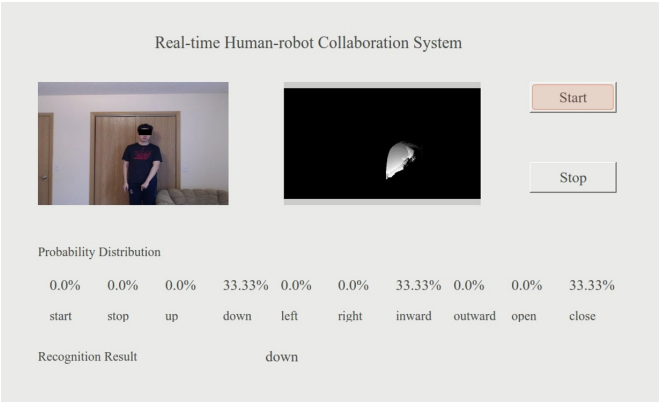
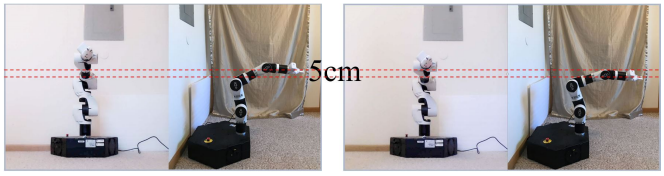


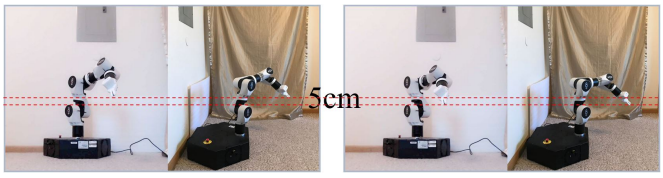
FIGURE 12: INITIALIZATION POSE FOR THE GESTURE *START*.



(a) recognition result:down

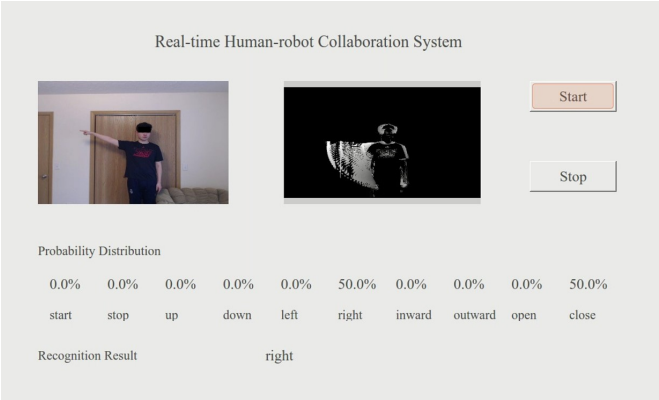


(b) the initial pose and the targeted pose

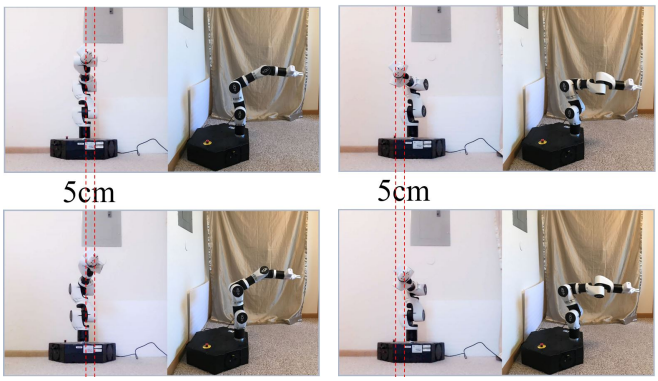


(c) the initial pose and the targeted pose

FIGURE 13: EXPERIMENTAL RESULTS OF THE GESTURE *DOWN*.

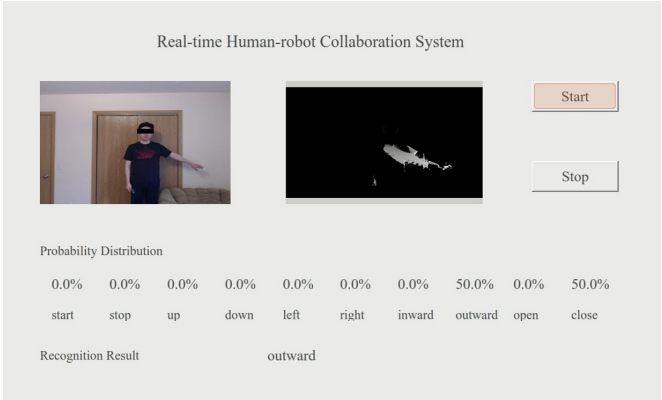


(a) recognition result:right

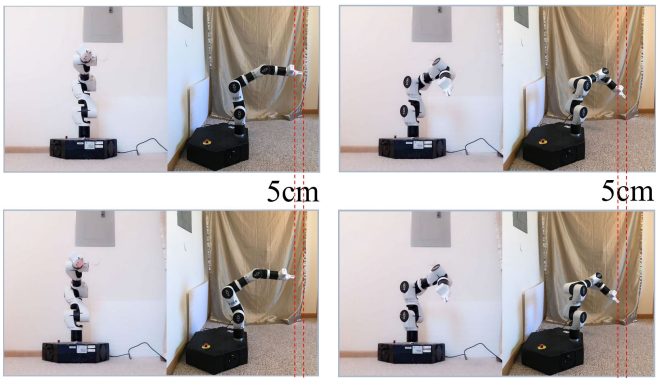


(b) initial pose and targeted pose (c) initial pose and targeted pose

FIGURE 14: EXPERIMENTAL RESULTS OF THE GESTURE *RIGHT*.



(a) recognition result:outward



(b) initial pose and targeted pose (c) initial pose and targeted pose

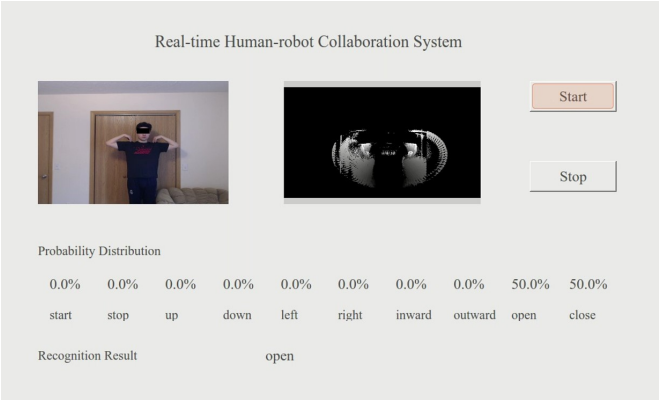
FIGURE 15: EXPERIMENTAL RESULTS OF THE GESTURE *OUTWARD*.

software development environment is Python 3.5 with image processing library OpenCV 4.1.2 installed. The ROS (Robot Operating System) is adopted for the communication between the laptop and the robotic arm. In order to satisfy the safety, the executing distance for each gesture is set as 5 centimeters at one time in these experiments.

When the calibration command is given by *start* gesture of the human worker, the robotic arm gives feedback with the pose in Fig. 12. In this pose, all the joint angles are zero and the robotic arm straights up. The coordinate system of the robotic arm is shown in the Fig. 12.

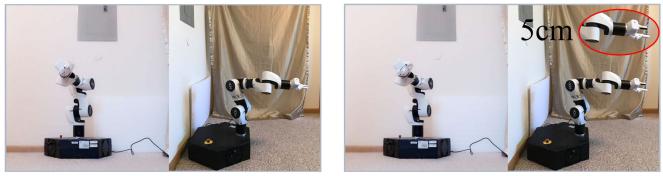
When the robotic arm receives operation gesture command from the system, the movement of the end-effector is conducted. Since the operation gestures are designed in pairs, and the responses of the robotic arm follow the same manner, only four gestures (*right*, *down*, *outward*, *open*) and related robotic movements are given in the experiments to show the performance of the whole system. To meet safety requirements, for every input operation gesture, the distance of movement along the specific direction is set 5 centimeters in these experiments. Fig. 13, 14, 15, and 16 show the recognition results of the real-time input gestures by the human worker, and the response movements of the robotic arm from different initial poses.

In Fig. 13 (a), it can be seen that the current camera frames and corresponding MHIs are updating on the two display windows, respectively. Note that the MHI generation in the right display window has a mild unstable delay relative to the current movements of the human worker due to the limited operation abilities of the experimental devices. The recognition probability distribution of the input gesture is shown on the GUI at the same time. It shows that the distribution of the probabilities seems less distinct because of the softmax layer processing, i.e., both *down* and *inward* are 33%. However, the recognition label can be obtained by the output layer values before the softmax operation, which means the index of the maximum in the initial output layer is identified as the gesture label. The final identification result, i.e., the gesture label *down*,



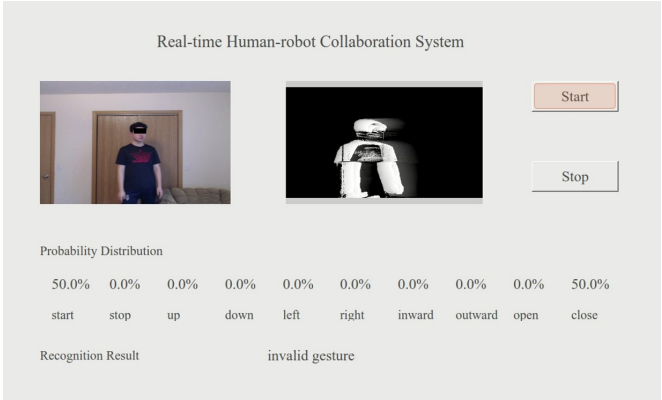
(a) recognition result:open

(b) targeted pose

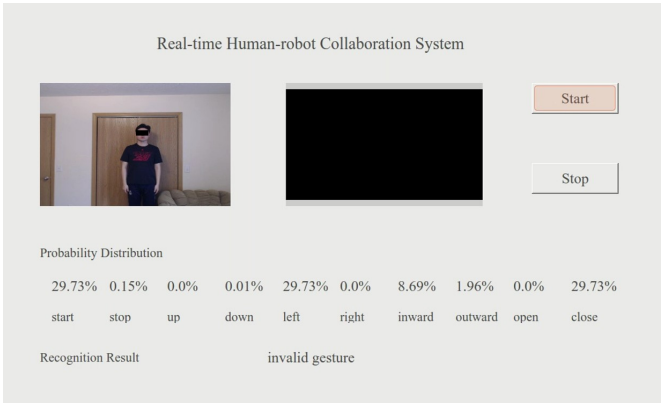


(c) targeted pose

FIGURE 16: EXPERIMENTAL RESULTS OF THE GESTURE *OPEN*.



(a)



(b)

FIGURE 17: INVALID MOVEMENT FILTRATION.

is output on the bottom of the interface, which is transmitted into the robotic arm. And when the robotic arm obtains a recognition label, the end-effector would move 5 centimeters along the corresponding direction or open the gripper. As shown in Fig. 13 (b) and (c), the same responses of the robotic arm are given for the corresponding gestures no matter the which pose (position and orientation) the robotic arm is on unless the end-effector or any joint reach the limitation of the dimension, which is same in Fig. 14, 15, and 16. Note that the gesture *outward* is defined with respect to the human worker, and the human worker is in a mirrored relationship with the robotic arm. So the the robotic arm in Fig. 15 moves back along the negative x axis as response of the gesture *outward*.

In the working mode, if the movements other than the designed gestures are performed by the human worker, the system would regard them as invalid gestures (as shown in Fig. 17). The robotic arm would not response to these invalid commands and stay in the current pose. There are mainly two cases of invalid gesture, including dynamic and static. In Fig. 17 (a), the probability distribution shows the dynamic invalid gesture has a higher probability to be similar to the gesture *start* and *close*. In Fig. 17 (b), when the human worker stops moving and keep static, the input invalid frame-based MHI generates a more uniform distribution.

6. CONCLUSION

In this paper, a real-time HRC system is developed for the gesture-based interaction between a human worker and a robot. In this system, the multi-threading technique is applied for operations of four parallel tasks, including user interface control, gesture capture and feature extraction, target recognition, and data transmission. A GUI is designed for system monitoring and operation. A real-time MHI method is designed for the feature extraction of the input dynamic gesture frames, and the CNN-based identification is carried out to obtain the label of the input gesture features. The data transmission between the host computer and a six DOFs robotic

arm is implemented simultaneously. In the experiments, A human worker can control the robotic arm by designed dynamic gestures in real-time pattern in this system. If the system detects movements other than expecting gestures, it would recognize these movements as invalid commands and the robotic arm would wait for the next command instead of giving responses. The robotic arm would stop movements when the end-effector touches the edge of the workspace or any joint is in the limiting position. It is shown that the behaviors of the human worker behaviors can be continuously captured and identified, and when designated gestures appear, the robotic arm can conduct responding operations accordingly to interact with the worker sequentially and seamlessly.

ACKNOWLEDGMENT

This research work is supported by the National Science Foundation via Cyber-Physical Sensing (CPS) Synergy project CMMI-1646162 and National Robotics Initiative (NRI) project CMMI-1954548, and also by the Intelligent Systems Center at Missouri University of Science and Technology. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Wenjin Tao, Ming C Leu, and Zhaozheng Yin. American sign language alphabet recognition using convolutional neural networks with multiview augmentation and inference fusion. *Engineering Applications of Artificial Intelligence*, 76:202–213, 2018.

[2] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[3] Wenjin Tao, Zehao Lai, Ming C Leu, and Zhaozheng Yin. Worker activity recognition in smart manufacturing using imu and semg signals with convolutional neural networks. *Procedia Manufacturing*, 26:1159–1166, 2018.

[4] Hao Ding, Jakob Heyn, Björn Matthias, and Harald Staab. Structured collaborative behavior of industrial robots in mixed human-robot environments. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1101–1106. IEEE, 2013.

[5] Guy Hoffman. Evaluating fluency in human–robot collaboration. *IEEE Transactions on Human-Machine Systems*, 49(3):209–218, 2019.

[6] Mohammad Safeea, Richard Bearee, and Pedro Neto. End-effector precise hand-guiding for collaborative robots. In *Iberian Robotics conference*, pages 595–605. Springer, 2017.

[7] Patrik Gustavsson, Anna Syberfeldt, Rodney Brewster, and Lihui Wang. Human-robot collaboration demonstrator combining speech recognition and haptic control. *Procedia CIRP*, 63:396–401, 2017.

[8] Mathias Haage, Grigoris Piperagkas, Christos Papadopoulos, Ioannis Mariolis, Jacek Malec, Yasemin Bekiroglu, Mikael Hedelind, and Dimitrios Tzovaras. Teaching assembly by demonstration using advanced human robot interaction and a knowledge integration framework. *Procedia Manufacturing*, 11:164–173, 2017.

[9] Claus Lenz, Markus Rickert, Giorgio Panin, and Alois Knoll. Constraint task-based control in industrial settings. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3058–3063. IEEE, 2009.

[10] Andrea Maria Zanchettin, Andrea Casalino, Luigi Piroddi, and Paolo Rocco. Prediction of human activity patterns for human–robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, 15(7):3934–3942, 2018.

[11] Sonia Duprey, Alexandre Naaim, Florent Moissenet, Mickaël Begon, and Laurence Cheze. Kinematic models of the upper limb joints for multibody kinematics optimisation: An overview. *Journal of biomechanics*, 62:87–94, 2017.

[12] Mark L Latash. *Fundamentals of motor control*. Academic Press, 2012.

[13] Tobias Ende, Sami Haddadin, Sven Parusel, Tilo Wüsthoff, Marc Hassenzahl, and Alin Albu-Schäffer. A human-centered approach to robot gesture based communication within collaborative working processes. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3367–3374. IEEE, 2011.

[14] Md Jahidul Islam, Marc Ho, and Junaed Sattar. Understanding human motion and gestures for underwater human–robot collaboration. *Journal of Field Robotics*, 36(5):851–873, 2019.

[15] Zhihua Chen, Jungtae Kim, Jianning Liang, Jing Zhang, and Yubo Yuan. Real-time hand gesture recognition using finger segmentation. *The Scientific World Journal*, 2014, 2014.

[16] Vaibhav V Unhelkar, Przemyslaw A Lasota, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A Shah. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters*, 3(3):2394–2401, 2018.

[17] Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. *Mechanical systems and signal processing*, 42(1-2):314–334, 2014.

[18] Olivier Janssens, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufier, Steven Verstockt, Rik Van de Walle, and Sofie Van Hoecke. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, 377:331–345, 2016.

[19] Xiaoxi Ding and Qingbo He. Energy-fluctuated multiscale feature learning with deep convnet for intelligent spindle bearing fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, 66(8):1926–1935, 2017.

[20] Haodong Chen, Wenjin Tao, Ming C. Leu, and Zhaozheng Yin. Dynamic gesture design and recognition for human-robot collaboration with convolutional neural networks. *2020 International Symposium on Flexible Automation (ISFA)*, 2020.

[21] David McNeill. *Gesture and thought*. University of Chicago press, 2008.

[22] Zhaozheng Yin and Robert Collins. Moving object localization in thermal imagery by forward-backward motion history images. In *Augmented Vision Perception in Infrared*, pages 271–291. Springer, 2009.

[23] Luis Miguel Bergasa, Jesus Nuevo, Miguel A Sotelo, Rafael Barea, and María Elena Lopez. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):63–77, 2006.

[24] Avinoam Kolodny, Uri Weiser, and Shahar Kvatinsky. Memristor based multithreading, December 31 2019. US Patent 10,521,237.

[25] Elshad Karimov. Queue. In *Data Structures and Algorithms in Swift*, pages 33–40. Springer, 2020.

[26] Haodong Chen, Zhiqiang Teng, Zheng Guo, and Ping Zhao. An integrated target acquisition approach and graphical user interface tool for parallel manipulator assembly. *Journal of Computing and Information Science in Engineering*, 20(2), 2020.

[27] Haodong Chen, Hongbo Zhu, Zhiqiang Teng, and Ping Zhao. Design of a robotic rehabilitation system for mild cognitive impairment based on computer vision. *Journal of Engineering and Science in Medical Diagnostics and Therapy*, 3(2), 2020.

[28] Haodong Chen, Yifan Wang, Zheng Guo, Wenxiu Chen, and Ping Zhao. A gui software for automatic assembly based on machine vision. In *2018 IEEE International Conference on Mechatronics, Robotics and Automation (ICMRA)*, pages 105–111. IEEE, 2018.

[29] Paul Shala Henry, Robert Bennett, Irwin Gerszberg, Farhad Barzegar, Donald J Barnickel, and Thomas M Willis III. Method and apparatus that provides fault tolerance in a communication network, April 4 2017. US Patent 9,615,269.

[30] Md Atiqur Rahman Ahad, Joo Kooi Tan, Hyoungseop Kim, and Seiji Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281, 2012.

[31] Kyongmin Yeo and Igor Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212–1231, 2019.

[32] James D McCaffrey. An empirical study of pairwise test set generation using a genetic algorithm. In *2010 Seventh International Conference on Information Technology: New Generations*, pages 992–997. IEEE, 2010.

[33] Binghui Chen, Weihong Deng, and Junping Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5372–5381, 2017.