Balkan Journal of Electrical and Computer Engineering (BAJECE)
Volume 1 Number 1, September 2023

Development of a Human-Robot Interaction System for Industrial Applications

Mustafa Can Bingol[1] and Omur Aydogmus[2]

1 Department of Electrical-Electronic Engineering Burdur Mehmet Akif Ersoy University, Burdur, Turkey,(e-mail: mcbingol@mehmetakif.edu.tr).

2 Department of Mechatronic Engineering Firat University, Elazig, Turkey, (e-mail: oaydogmus@firat.edu.tr).

Manuscript received Jan 12, 2023; accepted August 16, 2023.

Abstract— **The use of robots is on the rise, and this study focuses on developing manufacturing-assistant robot software for small production plants involved in non-mass production. The primary objective is to address the challenges of hiring expert robot operators by creating user-friendly software, thus enabling non-experts to operate robots effectively. The software comprises three main modules: the convolutional neural network (CNN), process selection-trajectory generation, and trajectory regulation. To initiate operations within these modules, operators record the desired process and its trajectory through hand gestures and index finger movements, captured in a video. The recorded video is then separated into images. These images undergo classification by the CNN module, which also calculates the positions of landmarks, such as joints and index finger's fingernail. Out of eight different pre-trained CNN structures tested, the Xception structure yielded the best result, with a test loss of 0.0051. Using the CNN's output data, the desired process is determined, and its trajectory is generated. The trajectory regulation module identifies the connection points between the generated trajectory and the object, subsequently eliminating unnecessary trajectory segments. The regulated trajectory, along with desired tasks like welding or sealing, is simulated using an industrial robot within a simulation environment. In conclusion, the developed software empowers non-expert operators to program industrial robots, particularly beneficial for companies with non-standardized production lines, where hiring expert robot operators might be challenging.**

Index Terms—**Classification and localization, Fingertip detection, Human-robot interaction, Welding process, Sealing process.**

I. Introduction

Robots can be classified according to the location (mobile and fixed), power systems (pneumatic, hydraulic, and electric), locomotion methods (stable, wheeled, legged, and others), or application areas (industrial and non-industrial) [1]. An industrial robot, as defined by the Robotic Institute of America [2], is a programmable mechanical device used to perform dangerous or repetitive tasks with high accuracy, replacing human labour. Collaborative robots accounted for 5.37%, 6.59%, and 7.54 % of the installed industrial robots between 2019 and 2021, respectively [3]. These installed industrial robots are used in various manufacturing processes such as machine tending, welding, and assembling, as shown in Figure 1. The aim of the current study is to transform a traditional industrial robot into a modern collaborative robot capable of performing welding and sealing processes.

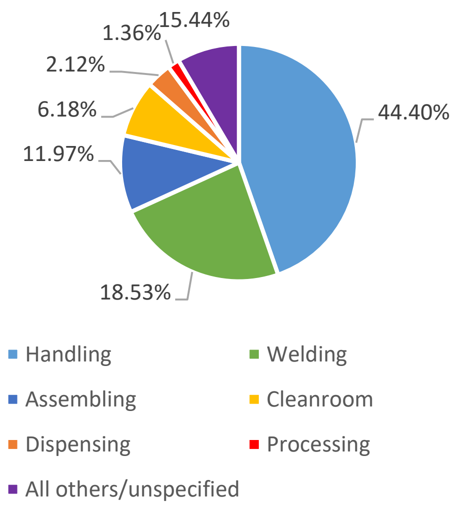


Fig. 1. Task distribution of industrial robots installed in 2022 [3]

Human-robot interaction (HRI) plays a crucial role in enabling human-robot collaboration (HRC). At least one communication channel, such as vision or speech should be used in order to occur HRI. Vision involves the interpretation of images captured by sensors such as cameras. Numerous studies in the literature have utilized vision to establish HRI in literature [4]–[6]. For instance, Hamabe et al. trained a lightweight robot for the assembly process using vision communication channels [7]. In another work, Ding et al. developed robot software to ensure safe manufacturing using vision [8]. In the current study, vision-based HRI software was developed.

Hand gestures recognition and fingertip position determination are commonly used methods for human-computer interaction and HRI [9]–[12]. Raheja et al. calculated to fingertip position by using skin colour of hand [9]. In another study, sequential mathematical operations were employed to obtain the fingertip position after subtracting the color-based hand image from the main image [13]. Other studies have also explored similar approaches using colour-based method [11], [12]. Another method employed for in hand gesture identification and fingertip position detection involves obtaining information from depth images captured by RGB-D sensors [14], [15]. Shin and Kim achieved an air writing process utilizing an RGB-D sensor and fingertip detection [16]. Similarly, another study successfully detected with successful by using RGB-D sensor [17]. Huang et al. has achieved fingertip detection by using a cascaded convolutional neural network (CNN) and RGB images, employing a different approach than the aforementioned studies [18]. In another study, air writing has been performed using color separation and faster R-CNN structures together [10]. In the robotic field, one notable example involves determining the orientation of the robot using an image from a sensor worn on the operator's wrist, along with hand gesture recognition and fingertip detection [19]. Many other studies have also utilized specialized sensors for detecting hand gestures or fingertips [20], [21]. In the current study, hand gestures and fingertip position were determined by processing the RGB images obtained from the environment using a single CNN.

The designed CNN structure incorporates a pre-trained CNN, and a transfer learning method was employed to train this designed CNN. Transfer learning is a skill that people often unwittingly use to apply an acquired ability to another similar task. It has been widely used in various applications, ranging from fault detection [22] to time series forecasting [23]. Li et al. used transfer learning to classify text data [24]. In another study, transfer learning has been utilized in order to process hyperspectral image [25]. Moreover, a robot has been successfully developed to detect damaged ropes on bridges using transfer learning [26]. Similarly, in another project, a robot employed transfer learning to distinguish objects from underground images [27]. In the current study, eight different pre-trained CNN architecture was trained for pre-defined task by using transfer learning and CNN structure that was obtained best result was chosen.

In the current study, a robot software was developed to assist in welding and sealing processes based on HRI. The developed software intended for use in small scale plants with non-mass production lines. Firstly, the desired task and a trajectory were defined according to the operator's hand gestures and positions of the fingertip. Next, the relationship between the defined trajectory and the metal object was searched. Finally, we implemented the obtained trajectory onto the robot in the simulation environment. As a result, the user could command the robot to perform the desired task without the need for manual programming. This study offers a user-friendly approach, allowing the robot to perform similar tasks without requiring any specific knowledge of robotics.

II. Material and Methods

A. Structure of Developed Software

Developed software consists of CNN, process selection-trajectory generation, and trajectory regulation modules. The modules of developed software and data traffic between modules are given in Figure 2.

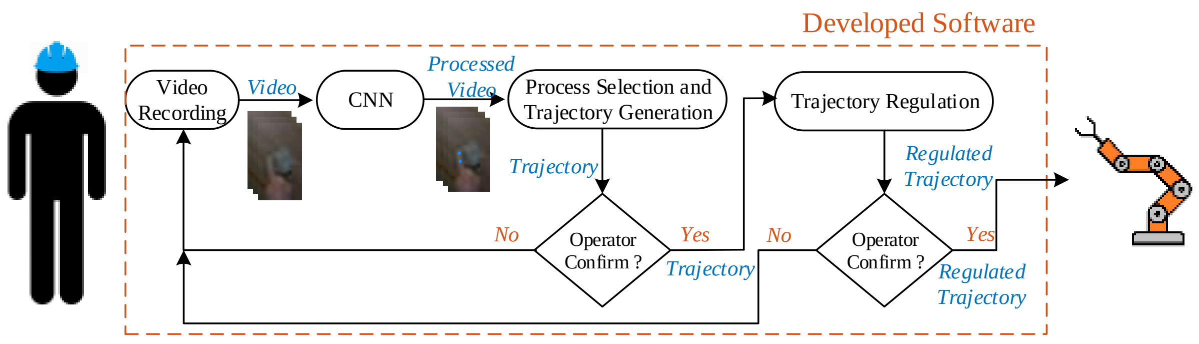


Fig. 2. Block diagram of developed software

A camera was used to be the robot aware of its environment and perceive the desired process of the robot. The camera's task was to record the hand movement of the operator in the robot's environment. The operator starts the video recording process before the operation, and the video recording is stopped by the operator when the defined process demonstration is finished. After the recorded video is separated into images, the images are sent to the CNN structure. CNN classifies the images according to hand gestures. If classifying result is calculated as index finger, middle joint, and fingernail positions of the index finger is produced by CNN. In the process selection and trajectory generation module, the type and start-end times of the operation such as welding or sealing are determined from hand motions. The index finger class between the start and end times determines the trajectory of the process. Then, the obtained process and trajectory are submitted for operator approval. If the operator does not confirm the process or trajectory, the program returns to the video recording stage. If the operator confirms the action and the trajectory, the trajectory regulation is applied to the trajectory. After the regulated trajectory is confirmed by the operator again, the determined process is carried out by the KUKA KR Agilus KR6 R900 sixx robot located in the simulation environment along the arranged trajectory. After the process is complete, the program returns to the first step.

1) CNN Module

Training of CNN, which is a part of developed software, consist of three steps as dataset forming, train dataset augmenting, and model training.

Fifty-three videos with a total size of 1.54GB were recorded in the experimental environment in order to create a dataset. Forty-five of these videos were used for the training and validation dataset. Rest of these videos were used for the test dataset. Total 8000 images, that were 180×320px size, were obtained from training and validation videos. 512 images, that were according to homogeneous each class, were randomly separated from this dataset for the validation dataset. Rest of these images was used as training dataset. Total 128 images, that were 180×320px size, were obtained from test videos to generate test dataset. Formed datasets were contained of four class as *Zero*, *One*, *Two*, and *Three*. Each class label was typified finger count of hand gestures. The data contained in *One* labelled class has 4 location information (x_J, y_J and x_T, y_T), as well as class labels. *J* and *T* letters were symbolized joint and fingernail of index finger, respectively. Class label and position data of datasets was given in Figure 3.

Data augmentation is an operation that artificial images, which are generated from training dataset images, are incorporated into the training dataset to increase the performance of CNN. The images were rotated 180° and the brightness of the rotated images were modified ratio of ±25% in order to increase the variety of images in the training data set. After these processes, obtained artificial images was added in the training dataset.

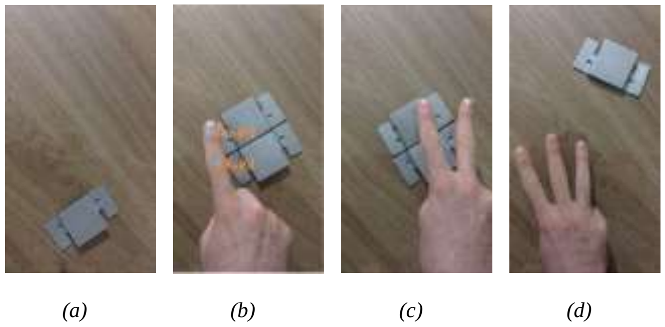


Fig. 3. Classes in the datasets; (a) Zero, (b) One, (c) Two, (d) Three

After the data sets were created, CNN structure was trained using the block diagram in Figure 4.

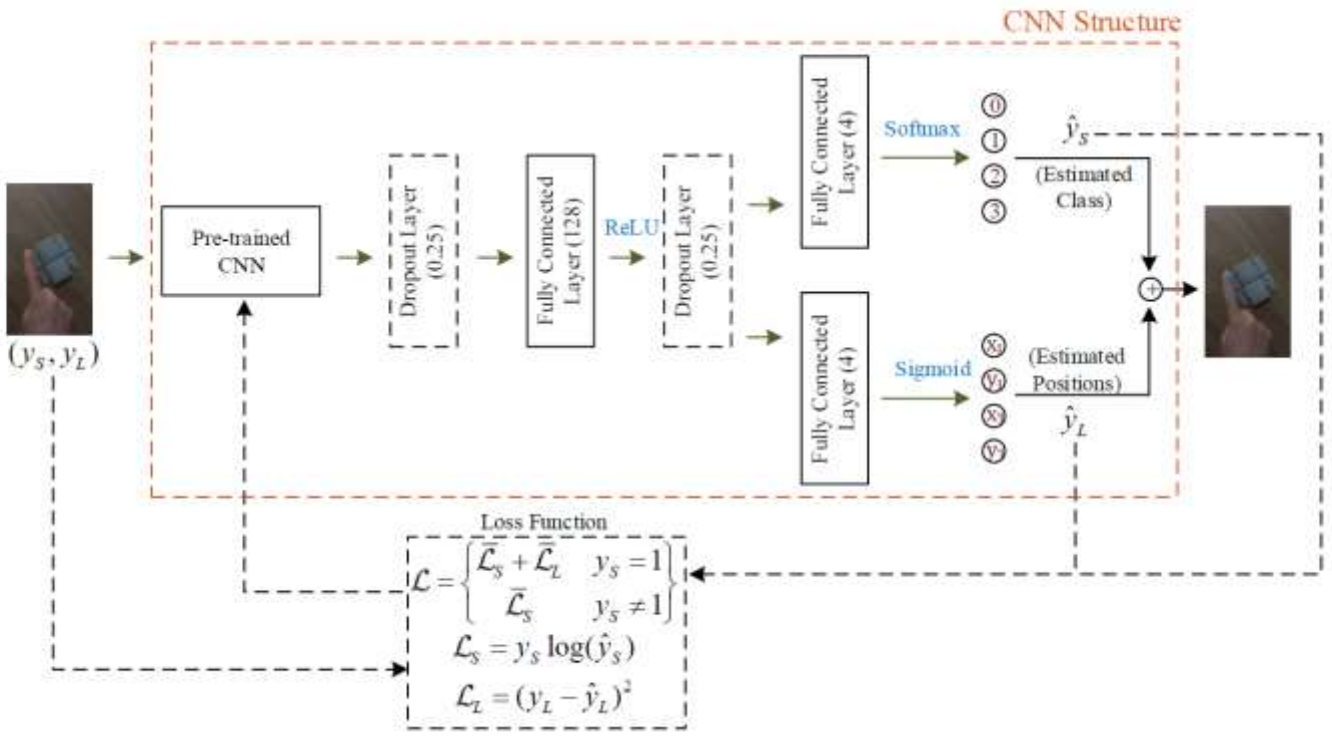

CNN Structure
(y_S, y_L)
Pre-trained CNN
Dropout Layer (0.25)
Fully Connected Layer (128)
ReLU
Dropout Layer (0.25)
Fully Connected Layer (4)
Softmax
0
1
2
3
$\hat{y}_S$
(Estimated Class)
Fully Connected Layer (4)
Sigmoid
x_J
y_J
x_T
y_T
(Estimated Positions)
$\hat{y}_L$
Loss Function
$\mathcal{L} = \begin{cases} \mathcal{L}_S + \mathcal{L}_L & y_S = 1 \\ \mathcal{L}_S & y_S \neq 1 \end{cases}$
$\mathcal{L}_S = y_S \log(\hat{y}_S)$
$\mathcal{L}_L = (y_L - \hat{y}_L)^2$

Fig. 4. Block diagram of CNN structure

y_S and y_L were shown class label and positions data of image in Figure 4, respectively. Eight different CNN model was tried as pre-trained CNN. $\hat{y}_S$ and $\hat{y}_L$ were symbolized predicted class and position data of CNN structure, respectively. Total, classification, and localization loss functions were presented as L, L_S, and L_L symbols, respectively. Cross-Entropy loss function was used as classification loss function and squared error was utilized as localization loss function. If the class of the input image is One, the total loss function is calculated by using a formula that sums the mean of L_S and L_L. Otherwise, the total loss function equals the mean of the classification loss function. The black dashed lines in Figure 4 show the blocks used only during the training of the CNN architecture. The dropout layer is also a structure consisting of dashed lines. The dropout layer ratio was chosen as 0.25. The black solid-lined blocks show the blocks used in both training and testing phases. In the fully connected layer, one of the black solid-lined blocks, there are 128, 4, and 4 neurons, respectively. ReLU, Softmax, and Sigmoid are activation functions found in network outputs. Detailed information about the functions of the blocks was mentioned in [28]–[30]. Also, the CNN structure was trained during 10 epochs by using the Adam optimization algorithm. The learning rate was chosen as 10^{-3} in the first 5 epochs and the learning rate was adjusted as 10^{-4} for the rest of the training process. Mini-batch size was chosen as 32. Weights of the pre-trained CNNs were set up as ImageNet and update of pre-trained CNNs weights were continued during the training process. The training process was shortened by using transfer learning.

2) Process Selection and Trajectory Generation Module

Process selection and trajectory generation were realized in this part of developed software by depending on class and position data obtained from CNN structure. Process selection operation was performed before trajectory generation. Firstly, noised class labels need to remove from obtained class labels to choose the process. The noised labels are formed by blurred images that occur when the operator's hand enters or quits on video. It is inspired by the exponential weight averages formula presented in Equation 1 to eliminate noised labels.

$$V_k = \beta V_{k-1} + (1-\beta)Q_k, k = 2,3,\ldots,N \quad (1)$$

In Equation 1, V, β, Q, N, and k refer to mean value, mean coefficient, current measure, total sample size, and discrete time index, respectively. How many samples will be averaged with β is determined by using as flows:

$$T_s = \frac{1}{1-\beta} \quad (2)$$

Total sample size is shown as T_s in Equation 2. Equation 1 was not used because the class output of CNN wasn't numerical value. Also, bias coefficient was not used owing to the same reason. Equation 3 that formed by inspiring Equation 1 was used to filter class labels.

$$fL_k = \begin{cases} Q_k & k \leq T_s \\ mod(fL_{k-T_s}, fL_{k-T_s+1}, \ldots, fL_{k-1}, Q_k) & k > T_s \end{cases} \quad (3)$$

fL is typified filtered label data in Equation 3. Raw and filtered label data that belong to the sealing process were given in Figure 5.

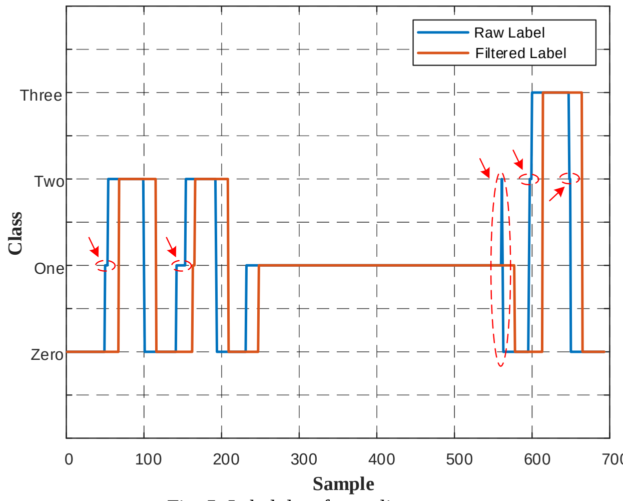


Fig. 5. Label data for sealing process

The red arrows in Figure 5 were shown noised label data. The detected noised labels were eliminated by using Equation 3. In the current study, β coefficient was chosen as 0.97. β coefficient is range of between 0 and 1 as can be understood from Equation 2. Also, while the noise increases as the β approaches 0, the inertia of the system increases as it approaches 1.

After filtering the label data, the process is determined by using the flowchart presented in Figure 6.

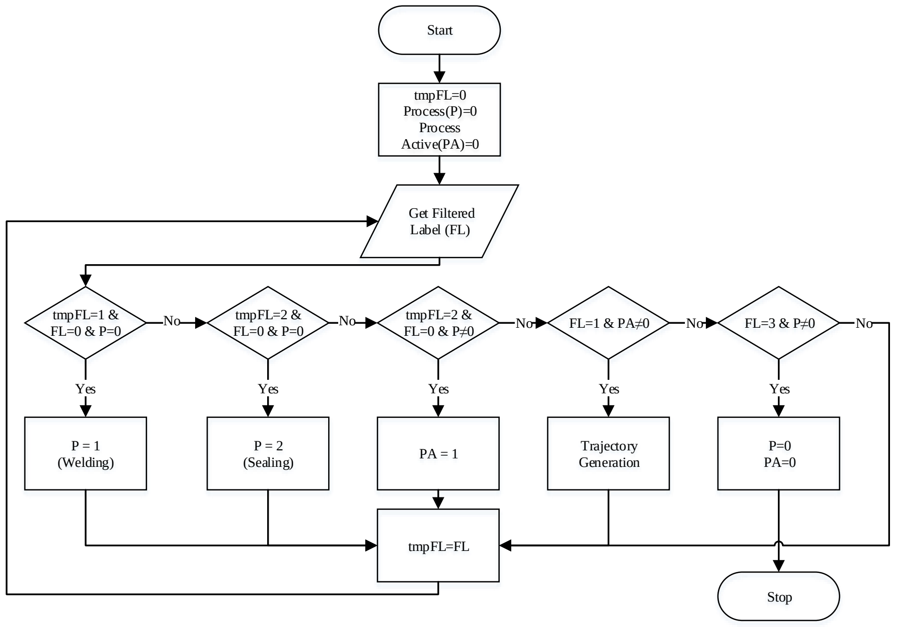


Fig. 6. Process determination flowchart

Firstly, the process is determined in Figure 6. After the defined process is activated, the trajectory generation is started when the filtered label is *One*. The trajectory generation process is continued until the filtered label is *Three*. The position and orientation of the index finger are calculated by using Equation 4-7 and the joint and fingernail position of the index finger. The reason for calculating the positions of the joint and fingernail of the index finger is that the operator can also determine the orientation of the manipulator in orientation-dependent operations such as welding.

$$\alpha = tan^{-1}(\frac{y_T - y_J}{x_T - x_J}) \tag{4}$$

$$d = \sqrt{(x_T - x_J)^2 + (y_T - y_J)^2} \tag{5}$$

$$x_F = x_T + \frac{d}{4}\sin(\alpha) \tag{6}$$

$$y_F = y_T + \frac{d}{4}\cos(\alpha) \tag{7}$$

α refers to the orientation angle of the index finger in Equation 4. d represents the distance between the joint and fingernail of the index finger in Equation 5. (x_F, y_F) are typified fingertip positions of the index finger on X and Y axes, respectively. The hand quickly moves towards the metal object during the trajectory generation process. The distance between two points was calculated using Equation 5 with finger position information obtained from two sequential images in order not to create a trajectory during this orientation process. If the calculated distance is lower than 7px, obtained (x_F, y_F) was included trajectory. Otherwise, (x_F, y_F) was not incorporated trajectory.

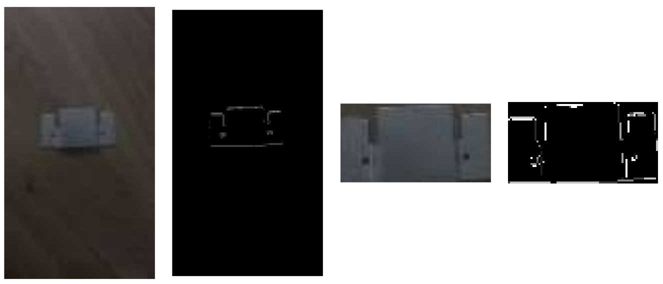
$$tx_k = \delta x_{F_k} + \emptyset\overline{(x_{F_{k-1}}, x_{F_{k-2}}, \dots, x_{F_{k-N}})} \tag{8}$$

$$ty_k = \delta y_{F_k} + \emptyset\overline{(y_{F_{k-1}}, y_{F_{k-2}}, \dots, y_{F_{k-N}})} \tag{9}$$

Trajectory of $trj_{2\times k} = [tx, ty]$ was obtained utilizing Equation 8 and 9. tx and ty refer to trajectory position on X and Y axes, respectively. $\delta, \emptyset$, and N represent last measure coefficient, mean coefficient, and count of elements to be averaged, respectively. These value of the coefficient were chosen as 0.5, 0.5, and 5, respectively. Noises on the trajectory were partially cleared by using Equations 8 and 9.

3) Trajectory Regulation Module

Trajectory regulation module was formed to establish relationship between generated trajectory and metal object and decrease noise on trajectory. An image that is not consist operator's hand was taken from the video to regulate trajectory. Initially, an image without operator's hand was taken from the video to regulate trajectory. The image was converted greyscale image and Sobel filter was applied to the greyscale image. Edges of object that is in the image was roughly calculated with the method as can be seen in Figure 7b. After this step, section of object in image was cropped by help of object edges. The cropped image was converted to grayscale image and blurred, respectively. Lastly, Sobel filter was applied on the blurred image and Figure 7d, that shows more clearly edges of object, was obtained. Blur filter was not used in first step because undesired edges were occurred in image.



(a) *(b)* *(c)* *(d)*
Fig. 7. Obtaining edge images process; (a) Original image, (b) Rough edge image, (c) Cropped image, (d) Edge image

Distance between each of the edge points in the obtained edge image and each of the trj points was measured by using Equation 5. trj points that were 15px away from edge points were removed from the trajectory after the measured distances. $mtrj$ trajectory that was not contained in the 15px away points was formed. Finally, the trajectory regulating process was carried out by applying a 20 × 1 dimensional median filter to the positions of the X and Y axis in $mtrj$. Figures 11 and 12 can be examined for a better understanding of the trajectory and regulated trajectory difference.

B. Simulation of Developed Software

In this study, the KUKA KR Agilus KR6 R900 sixx robot with 6 axes and an Euler wrist was used in our laboratory. The maximum payload and reach of this robot are 6kg and 901mm respectively. Also, the position repeatability of this robot is 0.03mm. Developed software was simulated on CoppeliaSim program that is a simulation environment. Before the simulation scene was not formed, a 3D solid model of the robot was drawn on the SolidWorks program. The formed 3D solid model was converted to URDF (Unified Robotic Description Format) with URFD exporter [31]. Then the URDF file was added to the designed scene as presented in Figure 8.

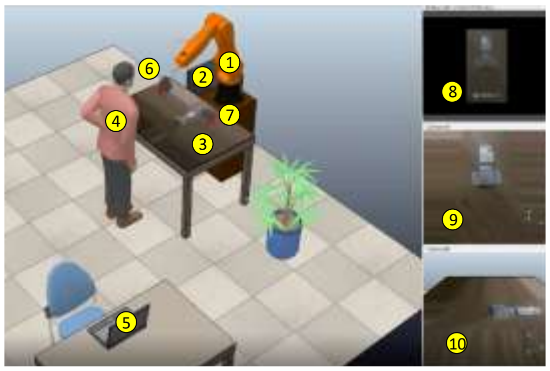


Fig. 8. Simulation environment

Manipulator (1) and robot PC (2) are components of the robot in Figure 9. Work plane, operator, and the PC that the software will run represents as (3), (4), and (5) respectively. Cameras (6-7) were added to the simulation environment to watch to the work plane from different angles. Video streams from the cameras were shown (9-10) windows, respectively. It is assumed that a camera is placed at the endpoint of the manipulator that records the hand movements of the operator and the video stream of this camera is presented on the screen (8). ManyCam program [32] was used to input video from outside to the simulation environment while creating the screen. The manipulator in the simulation environment was moved using MATLAB package program and the inverse kinematic solution of the simulation program. The simulation program was run using the Newton dynamic engine with 50ms step size.

III. Results

A part of the developed software is the CNN structure. The most important building block of this CNN architecture is pre-trained CNN structures. In the current study, CNN structure that using 8 different pre-trained CNN was trained. After the training process, training, validation, and test performance were presented in Table 1.

TABLE I
LOSS FUNCTION VALUES

Algorithms	PS[33] (MB)	TT (s)	L_{train}	L_{valid}	L_{test}
ResNet50[34]	98	1030	0.712	0.429	2.396
VGG16[35]	528	2460	1.539	1.515	1.412
DenseNET121[36]	33	2120	0.961	0.886	0.709
InceptionResNetV2 [37]	215	3820	0.016	0.003	0.533
EfficientNetB0[38]	29	2570	0.031	0.092	0.407
MobileNetV2[39]	14	1930	0.019	**0.002**	0.155
InceptionV3[40]	92	1720	0.017	0.230	0.080
Xception[41]	88	2200	**0.011**	0.003	**0.005**

Bold numbers indicate the best results.

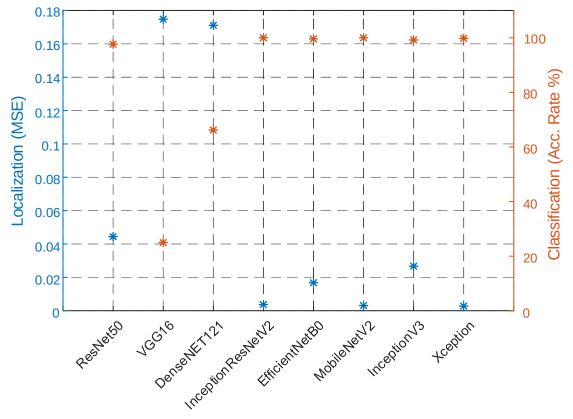
Parameter size and training time were shown as PS and TT in Table 1, respectively. The CNN training process was performed on the Google Colab platform. PC components used on this platform; GPU: Nvidia P100-16GB, CPU: Intel Xeon-2.30GHz, RAM: 25.51GB, Disk memory: 68.40GB. Since the best test result was obtained from the Xception algorithm, the loss values of Xception and other algorithms were compared by using multivariate Tukey comparison test and the Tukey test results are presented in Table 2.

TABLE II
XCEPTION AND OTHER METHODS COMPARISON

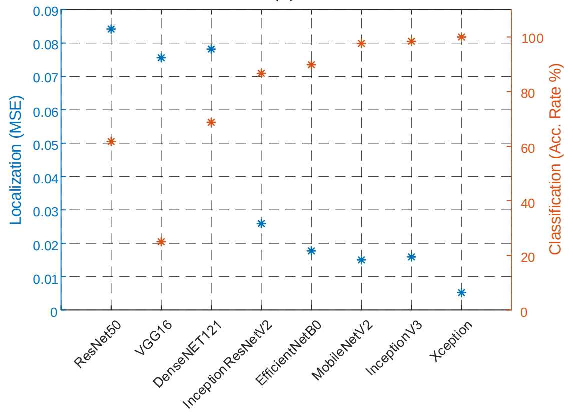
Algorithms	$\bar{L} \pm SD$	p (According to Xception)
ResNet50	1.1795±0.8683	**<0.05***
VGG16	1.4889±0.0552	**<0.05***
DenseNET121	0.8522±0.1057	0.238
InceptionResNetV2	0.1846±0.2468	0.999
EfficientNetB0	0.1771±0.1649	0.999
MobileNetV2	0.0592±0.0685	1.000
InceptionV3	0.1096±0.0894	1.000
Xception	0.0067±0.0033	-

**** Statistically significant difference. $\bar{L} \pm SD$ represents the mean of loss and standard deviation.***

The Xception algorithm was found to have statistical differences with the ResNET50 and VGG16 algorithms as can be seen in Table 2. There is no statistical difference between other algorithms and the Xception algorithm. In addition, after the training process was completed, all data in the validation and test datasets were classified by the CNN architecture and the relevant classes were located. These classification and localization results are shown in Figure 9.



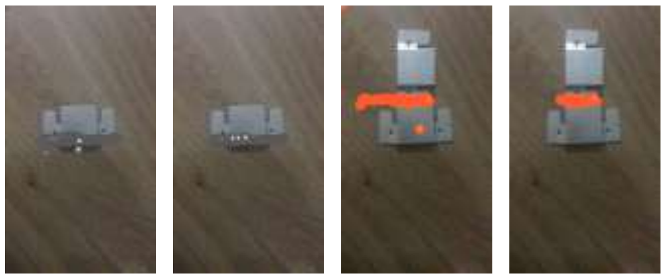
(a)



(b)

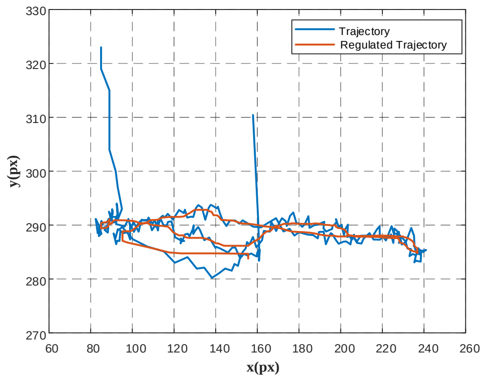
Fig. 9. Classification and localization results; (a) Results of the validation dataset, (b) Results of the test dataset

The Xception algorithm was used in the application as the best performance was obtained using the algorithm. Firstly, the operator recorded various sealing and welding process videos. These recorded videos were processed by CNN as seen in Appendix 1. Sealing process, welding process, and joint-fingernail of index finger were shown with black, red, and blue colour, respectively in Appendix 1. Then, generated trajectories were regulated and regulated trajectories were given in Figure 10 and 11.

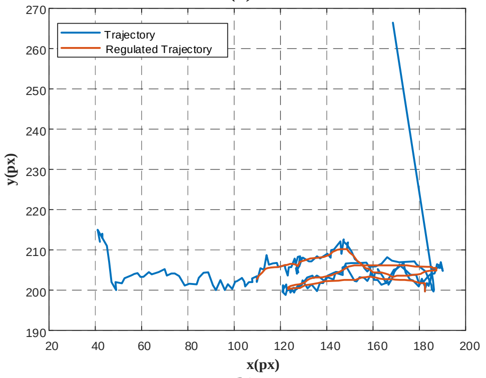


(a) *(b)* *(c)* *(d)*

Fig. 10. Visual results of trajectory regulation; (a) Sealing process trajectory, (b) Regulated sealing process trajectory, (c) Welding process trajectory, (d) Regulated welding process trajectory



(a)

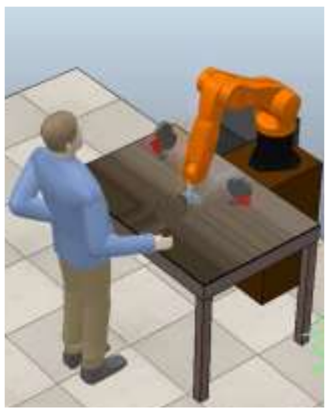


(b)

Fig. 11. Graphical results of trajectory regulation; (a) Sealing process trajectory, (b) Welding process trajectory

The regulated trajectories were sent to the robot that in the simulation environment. The desired tasks were simulated as seen in Figure 12.

Axis angles, axis moments, tool centre point (TCP) position and trajectory tracking error occurring during the sealing and welding process are presented in Figures 13 and 14.


(a)

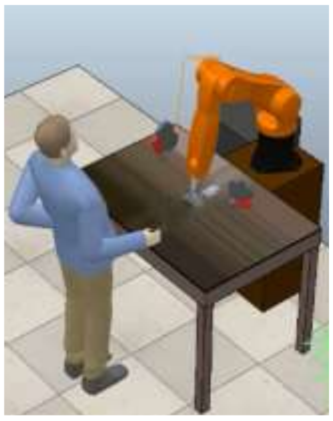
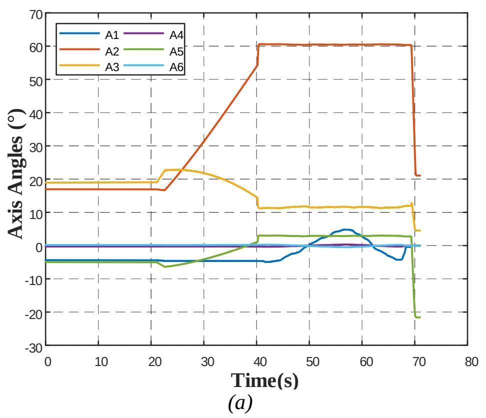
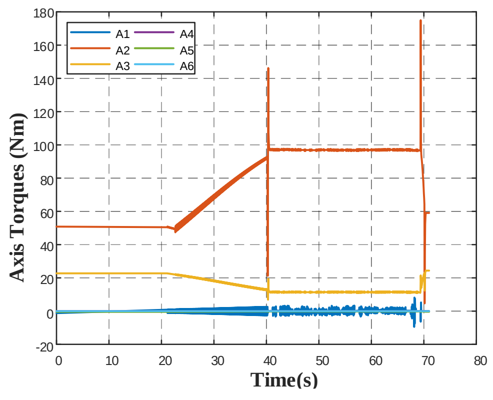

(b)

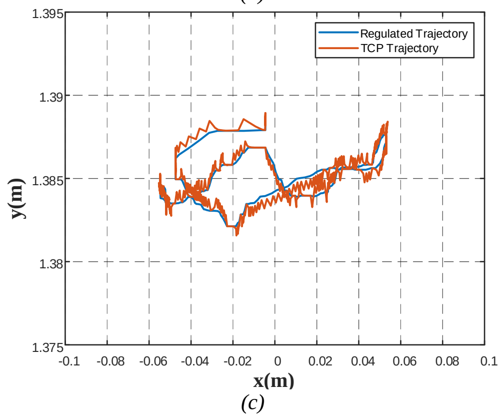
Fig. 12. Simulated desired tasks; (a) Sealing process, (b) Welding process



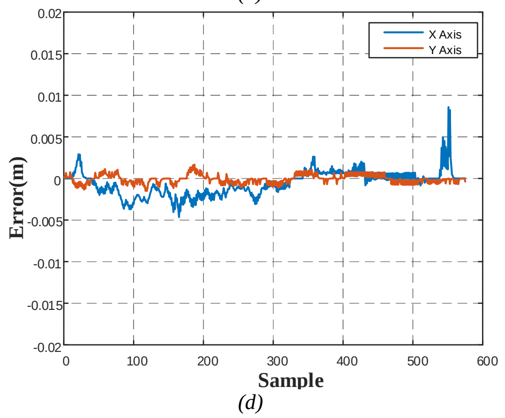
(a)



(b)



(c)



(d)

Fig. 13. Values occurring during the sealing process; (a) Axis angles, (b) Axis Moments, (c) TCP trajectory, (d) Error values during trajectory tracking

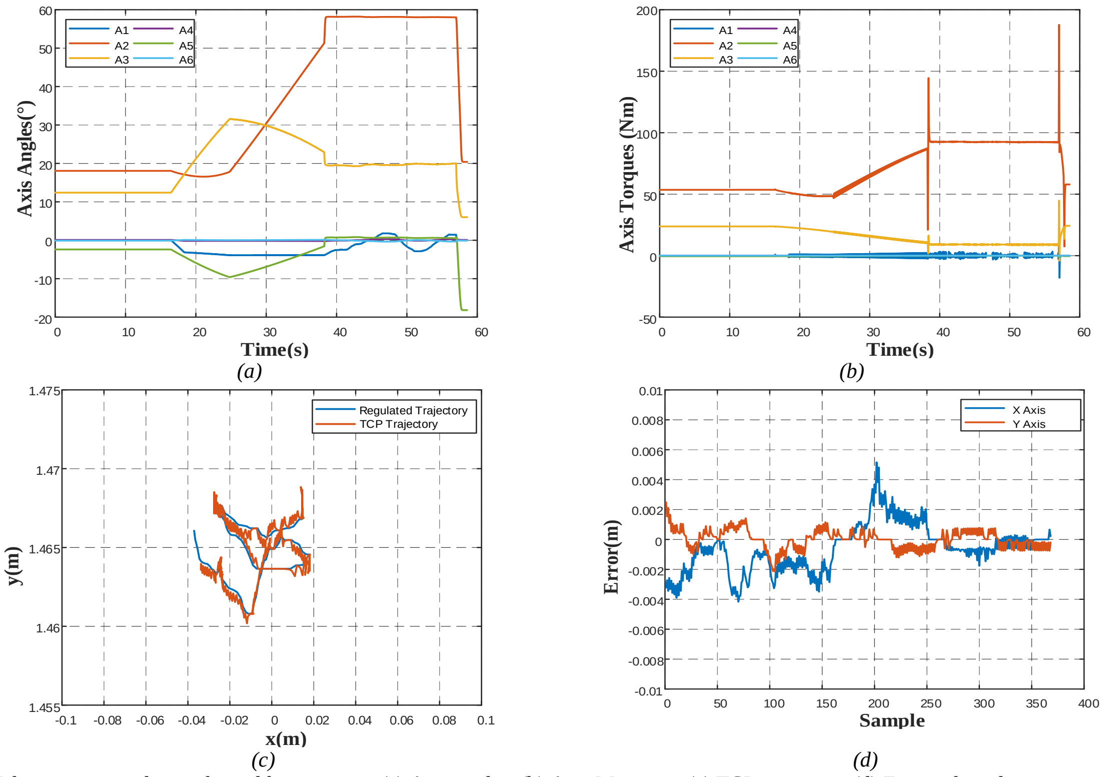
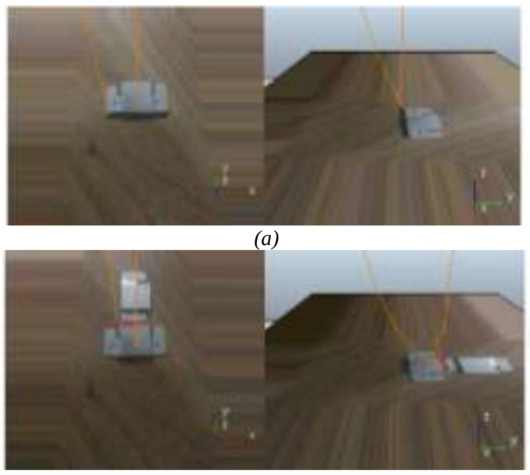


Fig. 14. Values occurring during the welding process; (a) Axis angles, (b) Axis Moments, (c) TCP trajectory, (d) Error values during trajectory tracking

After sealing and welding processes, metal object in work plane was given in Figure 15.



(b)

Fig. 15. Processed metal object; (a) Sealing process, (b) Welding process

The desired task by the operator was performed by the robot as it can be seen in Figure 15. The sealing process, welding process, and robot movements were given as videos in the [42] and [43], respectively.

IV. Discussion

Fingertip location was calculated by using skin colour in some studies when studies in the literature that fingertips detection were examined [11]–[13]. Depth images that were obtained from RGB-D sensors were used to detect fingertip position in other fingertip detection studies [14]–[16]. Skin colour and depth images were not used in this study. Also, when fingertip detection studies that were based on CNN structures were investigated, using cascade CNN structure was seen [18]. A single CNN was used as different from the study. In addition, when fingertip detection studies that were in the robotic field were researched, special sensors were developed to perceive hand gestures [19], [20]. In the current study, a standard camera was used to sense hand gestures.

In the current study, solving of classification and localization problem was implemented to hand gestures recognition and fingertip position detection. In this way, two different problems were solved with a single structure. This study has some limitations. In this study, the most important restriction of the current study is that the thicknesses of the parts to be machined were predefined and a standard depth was worked on. Another limitation is the CNN architectures used. Pre-trained CNN architectures were used to increase the accuracy performance by reducing the training time with the transfer learning method.

V. Conclusion

In this study, a robot software capable of performing processes such as sealing and welding was developed for small-scale plants without mass production capabilities. Operators without any prior robot education/knowledge can program the robot using finger movements through the developed robot software. This programmability capability was achieved through the integration of the CNN, process selection-trajectory generation, and trajectory regulation modules. The CNN structure consisted of a pre-trained CNN, fully connected layers, and activation functions connected in series. Eight pre-trained CNNs were trained on formed datasets and subsequently tested, with the Xception algorithm yielding the best result (L_{test}=0.0051). The CNN structure was used to classify image data and determine the positions of the robot's joints and the index finger's fingernail. With the classification data, the process selection and trajectory generation module detected the desired task, and the same module created the trajectory based on the positions data. Furthermore, a special algorithm was developed within the process selection and trajectory generation module to reduce any noise that may occur during video processing. The generated trajectory was then regulated by the trajectory regulation module to ensure proper alignment with the objects. Following this step, the robot performed the desired process within the simulation environment. In future work, an additional module will be developed to predict trajectories based on the objects and will be incorporated into the software. Subsequently, the software will undergo testing on a real robot. In addition, the developed software will become more improved by using other deep learning architectures such as LSTM.

References

[1] A. Dobra, “General classification of robots. Size criteria,” in *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, IEEE, 2014, pp. 1–6.

[2] “Defining The Industrial Robot Industry and All It Entails.” https://www.robotics.org/robotics/industrial-robot-industry-and-all-it-entails (accessed Sep. 28, 2020).

[3] IFR, *World Robotics 2022*. 2022. [Online]. Available: https://ifr.org/downloads/press2018/2022_WR_extended_version.pdf

[4] S. M. M. Rahman, Z. Liao, L. Jiang, and Y. Wang, “A regret-based autonomy allocation scheme for human-robot shared vision systems in collaborative assembly in manufacturing,” in *IEEE International Conference on Automation Science and Engineering*, 2016, pp. 897–902. doi: 10.1109/COASE.2016.7743497.

[5] H. Ding, M. Schipper, and B. Matthias, “Collaborative behavior design of industrial robots for multiple human-robot collaboration,” in *2013 44th International Symposium on Robotics, ISR 2013*, 2013. doi: 10.1109/ISR.2013.6695707.

[6] S. M. M. Rahman, Y. Wang, I. D. Walker, L. Mears, R. Pak, and S. Remy, “Trust-based compliant robot-human handovers of payloads in collaborative assembly in flexible manufacturing,” in *IEEE International Conference on Automation Science and Engineering*, 2016, pp. 355–360. doi: 10.1109/COASE.2016.7743428.

[7] T. Hamabe, H. Goto, and J. Miura, “A programming by demonstration system for human-robot collaborative assembly tasks,” in *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, 2015, pp. 1195–1201. doi: 10.1109/ROBIO.2015.7418934.

[8] H. Ding, J. Heyn, B. Matthias, and H. Staab, “Structured collaborative behavior of industrial robots in mixed human-robot environments,” in *IEEE International Conference on Automation Science and Engineering*, 2013, pp. 1101–1106. doi: 10.1109/CoASE.2013.6653962.

[9] J. L. Raheja, K. Das, and A. Chaudhary, “Fingertip Detection: A Fast Method with Natural Hand,” *Int. J. Embed. Syst. Comput. Eng. Local Copy*, vol. 3, no. 2, pp. 85–88, 2012, [Online]. Available: http://arxiv.org/abs/1212.0134

[10] S. Mukherjee, S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, “Fingertip detection and tracking for recognition of air-writing in videos,” *Expert Syst. Appl.*, vol. 136, pp. 217–229, 2019, doi: 10.1016/j.eswa.2019.06.034.

[11] S. K. Kang, M. Y. Nam, and P. K. Rhee, “Color based hand and finger detection technology for user interaction,” in *2008 International Conference on Convergence and Hybrid Information Technology, ICHIT 2008*, 2008, pp. 229–236. doi: 10.1109/ICHIT.2008.292.

[12] G. Wu and W. Kang, “Vision-Based Fingertip Tracking Utilizing Curvature Points Clustering and Hash Model Representation,” *IEEE Trans. Multimed.*, vol. 19, no. 8, pp. 1730–1741, 2017, doi: 10.1109/TMM.2017.2691538.

[13] G. Wu and W. Kang, “Robust Fingertip Detection in a Complex Environment,” *IEEE Trans. Multimed.*, vol. 18, no. 6, pp. 978–987, 2016, doi: 10.1109/TMM.2016.2545401.

[14] J. Yang, X. Ma, Y. Sun, and X. Lin, “LPPM-Net: Local-aware point processing module based 3D hand pose estimation for point cloud,” *Signal Processing: Image Communication*, vol. 90. p. 116036, 2021. doi: 10.1016/j.image.2020.116036.

[15] C. Wang, Z. Liu, M. Zhu, J. Zhao, and S. C. Chan, “A hand gesture recognition system based on canonical superpixel-graph,” *Signal Processing: Image Communication*, vol. 58. pp. 87–98, 2017. doi: 10.1016/j.image.2017.06.015.

[16] J. Shin and C. M. Kim, “Non-Touch Character Input System Based on Hand Tapping Gestures Using Kinect Sensor,” *IEEE Access*, vol. 5, pp. 10496–10505, 2017, doi: 10.1109/ACCESS.2017.2703783.

[17] J. L. Raheja, A. Chaudhary, and K. Singal, “Tracking of fingertips and centers of palm using KINECT,” in *Proceedings - CIMSim 2011: 3rd International Conference on Computational Intelligence, Modelling and Simulation*, 2011, pp. 248–252. doi: 10.1109/CIMSim.2011.51.

[18] Y. Huang, X. Liu, L. Jin, and X. Zhang, “DeepFinger: A Cascade Convolutional Neuron Network Approach to Finger Key Point Detection in Egocentric Vision with Mobile Camera,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, 2016, pp. 2944–2949. doi: 10.1109/SMC.2015.512.

[19] F. Chen *et al.*, “WristCam: A Wearable Sensor for Hand Trajectory Gesture Recognition and Intelligent Human-Robot Interaction,” *IEEE Sens. J.*, vol. 19, no. 19, pp. 8441–8451, 2019, doi: 10.1109/JSEN.2018.2877978.

[20] G. Shi, C. S. Chan, W. J. Li, K. S. Leung, Y. Zou, and Y. Jin, “Mobile human airbag system for fall protection using mems sensors and embedded SVM classifier,” *IEEE Sens. J.*, vol. 9, no. 5, pp. 495–503, 2009, doi: 10.1109/JSEN.2008.2012212.

[21] L. Peternel, N. Tsagarakis, and A. Ajoudani, “A human-robot co-manipulation approach based on human sensorimotor information,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 7, pp. 811–822, 2017, doi: 10.1109/TNSRE.2017.2694553.

[22] C. Li, S. Zhang, Y. Qin, and E. Estupinan, “A systematic review of deep transfer learning for machinery fault diagnosis,” *Neurocomputing*, vol. 407, pp. 121–135, 2020, doi: 10.1016/j.neucom.2020.04.045.

[23] R. Ye and Q. Dai, “Implementing transfer learning across different datasets for time series forecasting,” *Pattern Recognit.*, vol. 109, 2021, doi: 10.1016/j.patcog.2020.107617.

[24] Z. Li, B. Liu, and Y. Xiao, “Cluster and dynamic-TrAdaBoost-based transfer learning for text classification,” in *ICNC-FSKD 2017 - 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 2018, pp. 2291–2295. doi: 10.1109/FSKD.2017.8393128.

[25] S. Mei, X. Liu, G. Zhang, and Q. Du, “Sensor-specific Transfer Learning for Hyperspectral Image Processing,” in *2019 10th International Workshop on the Analysis of Multitemporal Remote Sensing Images, MultiTemp 2019*, 2019. doi: 10.1109/Multi-Temp.2019.8866896.

[26] S. Hou, B. Dong, H. Wang, and G. Wu, “Inspection of surface defects on stay cables using a robot and transfer learning,” *Autom. Constr.*, vol. 119, 2020, doi: 10.1016/j.autcon.2020.103382.

[27] G. A. Atkinson, W. Zhang, M. F. Hansen, M. L. Holloway, and A. A. Napier, “Image segmentation of underfloor scenes using a mask regions convolutional neural network with two-stage transfer learning,” *Autom. Constr.*, vol. 113, 2020, doi: 10.1016/j.autcon.2020.103118.

[28] M. C. Bingol and O. Aydogmus, “Practical application of a safe human-robot interaction software,” *Ind. Rob.*, vol. 47, no. 3, pp. 359–368, 2020, doi: 10.1108/IR-09-2019-0180.

[29] M. C. Bingol and O. Aydogmus, “Performing predefined tasks using the human–robot interaction on speech recognition for an industrial robot,” *Eng. Appl. Artif. Intell.*, vol. 95, 2020, doi: 10.1016/j.engappai.2020.103903.

[30] M. C. Bingol and Ö. Aydoğmuş, “İnsan-Robot Etkileşiminde İnsan Güvenliği için Çok Kanallı İletişim Kullanarak Evrişimli Sinir Ağı Tabanlı Bir Yazılımının Geliştirilmesi ve Uygulaması,” *Fırat Üniversitesi Müh. Bil. Derg.*, vol. 31, no. 2, pp. 489–495, 2019, doi: 10.35234/fumbd.557590.

[31] OSRF, “SolidWorks to URDF Exporter,” 2020. http://wiki.ros.org/sw_urdf_exporter (accessed Oct. 17, 2020).

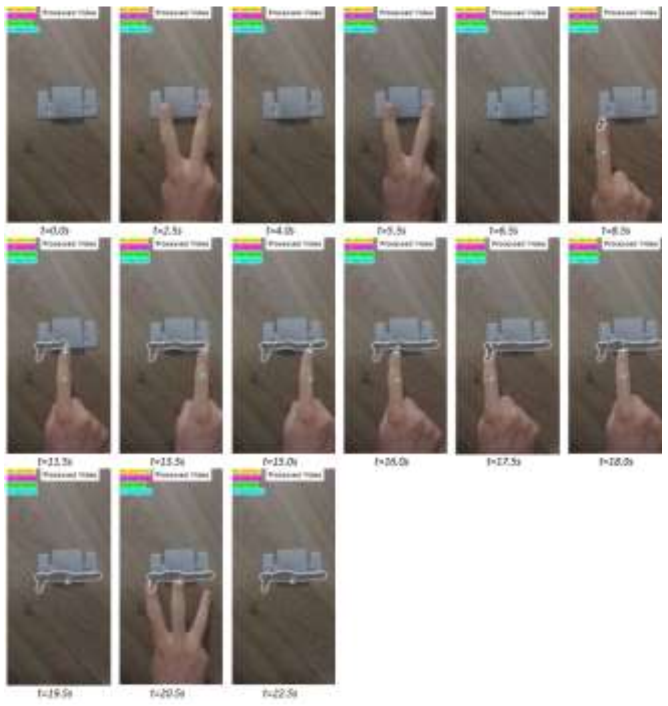
[32] “ManyCam Main Page.” https://manycam.com/ (accessed Oct. 17,

Balkan Journal of Electrical and Computer Engineering (BAJECE)
Volume 1 Number 1, September 2023

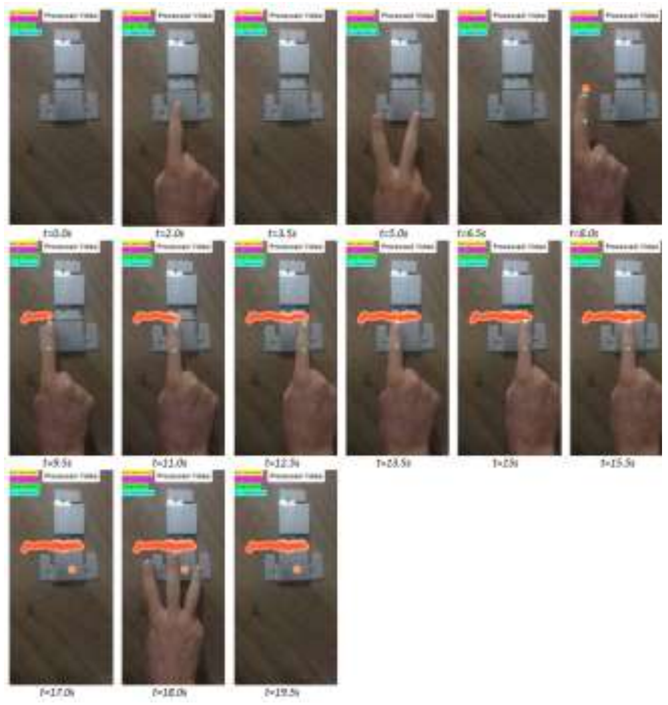
2020).
[33] TensorFlow, “Keras Applications,” 2020. https://keras.io/api/applications/%0A (accessed Oct. 15, 2020).
[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2015.
[35] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2014.
[36] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” 2016.
[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” 2016.
[38] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” 2019.
[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” 2018.
[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2015.
[41] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” 2016.
[42] M. C. Bingol and O. Aydogmus, “Sealing Process,” 2020. https://drive.google.com/file/d/1BbK_wv2Q76ItVLSbViC1Vb7YpxlN-tEa/view?usp=sharing (accessed Nov. 16, 2020).
[43] M. C. Bingol and O. Aydogmus, “Welding Process,” 2020. https://drive.google.com/file/d/1Y5jE8uPoiJMKLEHeG-wj1OqVRs1bO48-/view?usp=sharing (accessed Nov. 16, 2020).

APPENDICES

Appendix 1. Processing video images, (a) Sealing process, (b) Welding process



(a)



(b)