https://doi.org/10.7840/kics.2024.49.5.773

Human Imitation Manipulator System Based on 2D Image Recognition

Jin Su Park◆, Soo Young Shin°

ABSTRACT

This paper proposes a control system that uses deep learning to extract the positions of joints from shoulder to hand from 2D images, enabling a manipulator to mimic human movements. The proposed system utilizes a 2D camera to capture the appearance of a person as an image, and employs deep learning-based object recognition techniques to extract 3D coordinates of joints from the images. The extracted coordinates are then converted into vectors to obtain joint-specific rotation angles, which are subsequently used as input for controlling the manipulator. The simulation environment is implemented using ROS Gazebo and Moveit packages, while the actual robot control is conducted using Python and C++ for improved response speed. The functionality of the proposed system is validated through simulations and by employing a manipulator.

Key Words : Manipulators, Object Detection, Deep Learning

Ⅰ. Introduction

Recently advancements in robotics technology, the application areas have expanded, highlighting the remarkable utility of robots in replacing or assisting human tasks. Particularly, due to the challenges in securing labor force caused by COVID-19, the exceptional applicability of robots has been emphasized.

Industry 4.0 is one of the proposed ideas to address the challenges faced by the manufacturing industry due to COVID-19[1]. It encompasses foundational information technology (IT), operations technology (OT), and data infrastructure, along with solutions such as digital twins and logistics automation. These technologies are being reconfigured into a unified form known as the smart factory, leading to transformative changes in the manufacturing industry.

The smart factory is a system that incorporates various ICT technologies such as the Internet of Things (IoT), big data, cloud computing, and Cyber-Physical Systems (CPS) to revolutionize product manufacturing. Baotong and Jiafu propose a hierarchical structure for building smart factories, aiming to advance the manufacturing industry. They divide the key technologies for overcoming current technological challenges into different layers and propose corresponding solutions[2].

Manipulator is the technology which perform various roles in a smart factory. It replaces dangerous and difficult processes performed by humans in manufacturing industry to improve time and cost efficiency of production. To perform various role, it is required control technology which support this. Previous, programming and remote controller were used, but these are unintuitive and requiring trained worker. To overcome these limitation, control methods with different concepts from existing control

※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICAN(ICT Challenge and Advanced Network of HRD) program(IITP-2024-RS-2022-00156394) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation)
◆ First Author : Kumoh National Institute of Technology, jp@kumoh.ac.kr, 학생회원
° Corresponding Author : Kumoh National Institute of Technology, wdragon@kumoh.ac.kr, 종신회원
논문번호 : 202311-132-D-RN, Received November 6, 2023; Revised January 16, 2024; Accepted February 5, 2024

methods are being researched. Representative technologies include flexible sensors or vision-based robot control technology[3-6].

In recent years, Vision-Based manipulator control which is one of the way which provide easy and intuitive control is being widely studied. Because of these are relatively inexpensive and durable than flexible sensor based technologies. Gesture recognition control is the most representative vision-based method. However, Gesture recognition control have limitation which recognizing a few gestures to control robots. Because of this, there are limitations in utilizing all the various functions of the manipulator[5,6]. Accordingly, robots that imitate human behavior rather than recognize gestures are also being studied. But, this also has the limitation that only one of the hands and arms can be controlled[7,8].

In this paper, we proposes a 2D object recognitionbased robot control system as an improvement to existing robot control technology that imitates user movements to replace humans which manipulators in manufacturing. The proposed system captures the user's arms and hands using a camera and utilizes deep learning-based 2D object recognition technology to compute the positions of joints in 3D space. By forming vectors from the calculated 3D coordinates, the rotation angles and relative angles of each joint are computed, enabling the control of the robot.

The structure, communication method, algorithms, and necessary equations of the proposed system are introduced in Chapter 2. Chapter 3 describes the experimental environment for simulation and actual robot operation, along with specific experimental methods. Finally, Chapter 4 presents suggestions for future research and a concise conclusion.

Ⅱ. System

2.1 System Architecture

Fig 1. is show a picture of the proposed system model. The proposed system consists of a Robot-Control Module and a User-Detection Module. The User-Detection Module captures the user's

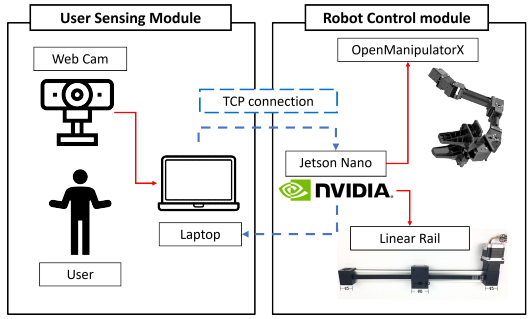

Fig. 1. Implemented system's structure

appearance using a 2D camera. Subsequently, the coordinates of each joint are extracted from the captured images. These joint coordinates are used to generate vectors, and vector operations are performed to calculate the rotation angles and relative angles for each joint.

The Robot-Control Module utilizes the radian values of each joint, which are calculated by the User-Detection Module, to control the robot. TCP protocol is employed for communication between modules situated in separate spaces. The Robot-Control Module saves the calculated radian values as files, which serve as the basis for controlling the robot. For the implement system, we are using Linear Rail which similar to real manufacture environment.

Algorithm 1 is present flow of the proposed system. At the system's initialization, it make a connection between User-Detection Module and Robot-Control Module. If the communication is successfully established, the camera is activated to detect the user. Once user is recognized, the 3D coordinates of the user's joints are extracted, and the individual joint angles (relative angles and rotation angles) are computed. These calculated values are transmitted to the Robot-Control Module through TCP communication, and the iterative process continues until the completion of robot control. While controlling the robot, user open or close the left hand, able to switch the mode "arm control" and "rail control". The arm control mode, robot mimic the user and The rail control mode, system check the location of left hand to control the rail.

Algorithm 1: Proposed System

Initialization:
Establish continuous TCP connection between User-Detection and Robot-Control Modules.
Turn on the Camera.
Input: 2D RGB images from the camera.
Output: Robot Operating values : $Opr = [[J_1, J_2, J_3, J_4, J_5, J_6], St_{grp}, St_{rl}]$
Main Control Loop:
while *User operating system* **do**
    User-Detection Module's task
    Capture 2D RGB Image
    if *Human detected in the image* **then**
        Extract Key-point coordinates
        Calculate Joint Angles
        if *Right Hand is Closed* **then**
            Change the value to close the gripper : $(St_{grp} = 0)$
        else if *Right Hand is Opened* **then**
            Change the value to open the gripper : $(St_{grp} = 1)$
        if *Left Hand is Closed* **then**
            Change the value to control the joint : Renew the values on $[J_1, J_2, J_3, J_4, J_5, J_6]$
        else if *Left Hand is Opened* **then**
            Check the left hand coordinates.
            if *Left hand is located to the left of the left shoulder* **then**
                Change the value to control the rail moves the robot to the left side : $(St_{rl} = -1)$
            else if *Left hand is located to the right of the left shoulder* **then**
                Change the value to control the rail moves the robot to the right side : $(St_{rl} = 1)$
        Send Robot operating values(Opr) through TCP
        Robot-Control Module's task
        Read TCP Messages(Opr) for new values
        Update Robot Control based on received values
Termination:
Turn off the Camera.
Close the TCP connection.

2.1.1 User-Detection Module

User-Detection Module have 2 Major Role which are Extracting Key-points coordinates of user and Calculate values to control the robot

To Extracting Key-points coordinates, the User-Detection Module uses Python to process images acquired from the camera. In this paper Using MediaPipe and OpenCV library to extract the keypoint coordinates of the user's joints from the image. MediaPipe is a deep learning library for human recognition developed by Google. MediaPipe provides two-step detectortracker ML pipeline model. First locates region-of-interest (ROI) of the target objects for each model and within the frame for recognizing human pose, face, and hand. And then, subsequently predicts the landmarks or segmentation mask in the ROI-cropped frame. MediaPipe's feature is that it recognizes human in two-dimensional images and outputs the coordinates of keypoints in three dimensions through pre-trained model. Media Pipe

provides deep learning models for faces, pose, and hand, making system implementation simple. It makes reduce the cost of system implementation and allows coordinates to be predicted based on key-points recognized through deep learning even if part of the user's body is occluded

In the paper, MediaPipe's three models, Pose, Face, and Hand, are used to extract the user's Key-Point coordinates. Figure 2 shows the detected key-points from each model. Extracted Key-Points have an independent coordinate system for each model.

To ensure recognition accuracy during robot control, the distance between the user and the camera is measured and guided so that the user can be positioned at a predefined distance. Distance is measured using the coordinates corresponding to Iris among the key-points extracted from the face model. The distance is calculated through the correlation of gap and number of pixels between two irises. In this paper define the gap between the two iris as 12cm and create an equation to calculate the distance by measuring the change in the number of pixels according to the distance between the user and the camera. This allows the user to maintain a certain distance from the camera and ensures recognition accuracy by ensuring that all key points are recognized.

Key-points extracted from the Hand model are used

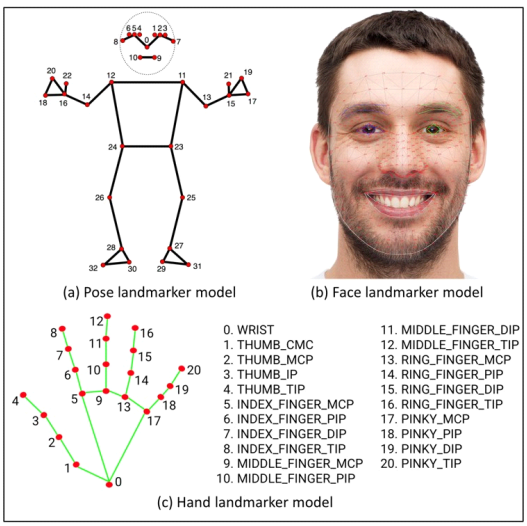

Fig. 2. Detected key-points from each model

to determine whether the user has closed his/her hand. In Figure 2-(c), the distance from the wrist to the first joint of the middle finger (landmarks 0 and 9) is defined as Distance A, and the distance from the wrist to the tip of the middle finger (landmarks 0 and 12) is defined as Distance B. If the length of Distance A is shorter than Distance B, it is recognized that the user has opened hand. Conversely, if the length of Distance A is longer than Distance B, it is recognized that the user has closed hand.

Algorithm 2 is the process of calculating the angle of the joint. To calculate the angle of each joint, create Links using key-points extracted from the Pose model. Convert the generated vector from the cartesian coordinate system to the spherical coordinate system. Θ and ϕ values calculated through coordinate system transformation correspond to the relative angle and rotation angle. The rotation angle and included angle are each used to control the robot with the input values of the corresponding motors.

Fig 3 is the example of the angle calculation at the Elbow joint. First, generate link vector using coordinates of Elbow and Wrist(Landmark 14 and 16). Set the landmark 14 as origin, transform the coordinate from Cartesian to Spherical coordinates.

Algorithm 2: Joints angle Calculation

Input: Recognized human keypoint coordinates
Output: Radian values to control manipulator
Define Key-point Coordinates
Joints = [(Joint_0x, Joint_0y, Joint_0z), ⋯ (Joint_Nx, Joint_Ny, Joint_Nz)]
Define Link Vectors
Link_Vectors = []
for $N \leftarrow 0$ ***to*** *Number_of_Joints - 1* **do**
    Link_N = (Joint_[N+1]x - Joint_Nx, Joint_[N+1]y - Joint_Ny, Joint_[N+1]z - Joint_Nz)
    Link_Vectors.append(Link_N)
Transform Link Vectors from Cartesian to Spherical Coordinates
for ***each*** *Link* ***in*** *Link_Vectors* **do**
    $\theta = \text{atan}(\sqrt{Link\_x^2 + Link\_y^2}/Link\_z)$
    $\phi = \text{atan}(Link\_y/Link\_x)$
    Store the θ and ϕ for each joints

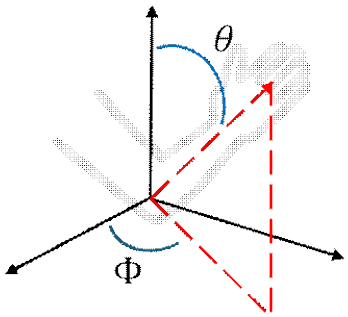

Fig. 3. Example of the angle calculation

Calculate θ is the input of relative angle at the Elbow and ϕ is the input of rotattion angle at the Elbow.

2.1.2 Robot-Control Module

Robot-Control Module have 2 Major Role which operating robot based on the value came from user-detection module and establish TCP server socket to receive the values.

Upon Robot-Control Module initialization, a TCP socket is created to establish communication and configure the necessary settings. When a connection request is received from the User-Detection Module, it is accepted, and messages are received. The received angle information is saved in a file, which is subsequently read to control the robot. The stored values enable the repetition of identical motions in the future, and the system allows the user to manually modify the angles at specific time points. If a blank message is received, the system terminates, and the TCP communication is closed.

In system implementation, the robot and gripper are connected to the module through USB, and the rail is connected to the motor driver and serial communication through module's GPIO pins. The Module controls the robot using Python with stored the values received through TCP in an array for robot control. The array contains 8 values: 6 values corresponding to the robot's joints (rotation and relative angles of the shoulder, elbow, and wrist), a value representing the open/closed state of the gripper, and a value for rail control.

2.2 Communication System

In this paper, We use a VPN server to establish the User-Detection Module and Robot-Control Module as if they were part of an internal network, ensuring both security and connectivity. Through the VPN, TCP communication was enabled between the User-Detection Module and Robot-Control Module, similar to an internal network, without requiring additional configurations for external networks including 5G. The host for TCP communication is the Robot-Control Module, while the User-Detection Module operates as a client and communicates with the Robot-Control Module from an external source.

Ⅲ. System Verification

3.1 System Verification Process

Table 1 shows the system resources utilized in this study for system validation. The system verification is performed in two stages. Firstly, a simulation environment is established using Gazebo on a laptop equipped with a camera to verify the proper functioning of the user recognition and joint angle calculation in the User-Detection Module. Subsequently, the Robot-Control Module is implemented on a physical robot to confirm its operation. To ensure secure and connected TCP communication, the User-Detection Module and The Robot-Control Module are configured like an internal network using a VPN server.

3.2 System Simulation

To verify the operation of the User-Detection Module before actual manipulator control, we created a simulation environment. For verification, two models were selected: UR5, which is widely used in industrial settings and has joint ranges similar to those of a human, and OpenManipulator-X, which was used for the actual system implementation[11,12]. In the simulation, we were not using linear rail to verify

Table 1. Experimental Environment

Name	Configuration
User-Detection Module	i5-1135G7 (Laptop)
Camera	ABKO APC890W
Robot-Control Module	Jetson Nano
OS	Ubuntu 18.04

manipulator control side. The simulation was created using ROS Gazebo and operate using MoveIt package.

MoveIt is an open-source software, it provides tools for robot kinematics, motion planning which can use in not only simulation but also real hard ware[10]. The operation of each module can be verified through simulation, and actual hardware can be controlled identically using the MoveIt package. In the implemented system, the robot operates using the Motion Planning Tool of the MoveIt package. At this time, it able to set the Planning Time, which takes for the robot to reach the input value. In the simulation, Planning Time is set to 50ms and the robot is synchronized with the angle input from the User-Detection Module every 50ms. The total latency when synchronizing every 50ms without considering network speed is about 25ms.

Table 2 provides information about the robots used for system verification. UR5 has six degrees of freedom, and all joints can rotate 360°. OpenManipulator-X has four degrees of freedom, with constraints on the range of motion for each axis. The ranges specified in rows 1 to 4 represent the shoulder rotation angle, inter-joint angle, elbow inter-joint angle, and wrist angle, respectively.

Figure 4 illustrates the simulation of UR5 and OpenManipulator-X. It shows that the extraction of joint coordinates and angle calculations in the

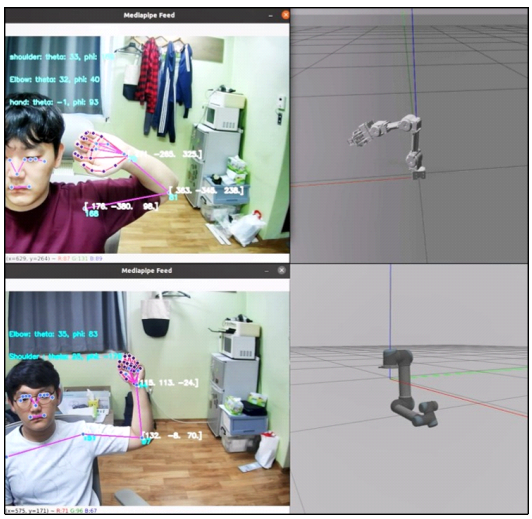

Fig. 4. System Simulation

Table 2. Information for the Manipulator

	UR5	OpenManipulator-X
DoF	6 Axis	4 Axis
Operating range	± 360°	① -130° ~ 0°
		② -90° ~ 90°
		③ -90° ~ 45°
		④ -30° ~ 30°

User-Detection Module are functioning correctly, as the manipulator in Gazebo is able to mimic the posture of the user's arm. However, unlike UR5, OpenManipulator-X has limita- tions in the number of drivable axes and range of mo- tion, which means it may not be able to reproduce certain movements perfectly.

3.3 Actual System Verification

After validating the operation of the User-Detection Module through simulation, The Robot-Control Module is implemented and connected to the OpenManipulator-X for real-world testing. The MCU (Microcontroller Unit) for The Robot-Control Module is Jetson Nano, and the software utilizes Ubuntu 18.04 with ROS Melodic, which is supported by Jetson Nano. The robot connects to Jetson Nano via USB, and the rail connects via GPIO pins. Robot control uses the MoveIt package in the same way as simulation. For hardware operation stability, unlike simulation, the Planning Time was increased to 100ms when controlling the actual robot. The total latency when synchronizing every 100ms without considering network speed is about 200ms.

Due to the high latency, the MoveIt package is used only for basic verification of actual robot movements. To reduce latency, control the robot directly using the Python API. Using the Python API, control values transmitted from the User-Detection Module can be synchronized with the robot without planning time.

Figure 5 present the appearance of the implemented The Robot-Control Module. To perform functions simi- lar to those in actual industrial settings, the manipulator is installed on conveyor belt. For intuitive robot control by the user, the robot is mounted on the side of a pillar, mimic a human arm. When controlling the robot using the Python API, the

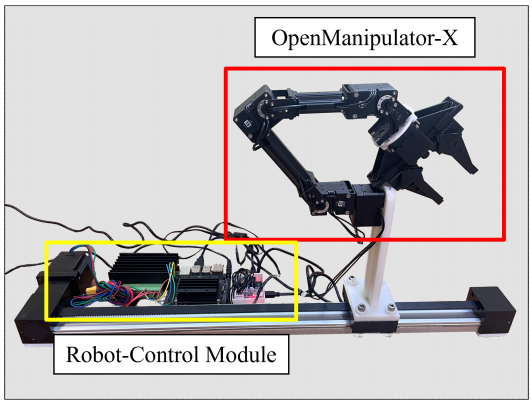

Fig. 5. Robot-Control Module Hardware

measured latency was 50 to 70 ms, which was not much different from the simulation.

Figure 6 present the components of the implemented Robot-Control Module. From left to right, Motor driver for linear rail, Jetson Nano, U2D2 which is TTL level communication module for each joint of manipulator. Jetson Nano send calculated value to U2D2 and motor driver to control each joint and linear rail.

Figure 7 presents the validation process of the imple- mented system. We verify the system through a Pick- and-Place process which is one of the common action with manipulator. The user follows the instructions pro- vided by the interface and waits until they are positioned at an appropriate distance. The manipulator is operated according to the user's actions. The manipulator is pick up a can, move along the rail towards the trash bin, and place the can into the bin. We repeat the process 30 times for each person and check the success rate of it.

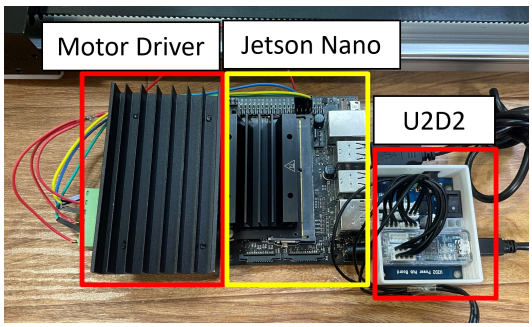

Fig. 6. Implemented system's structure

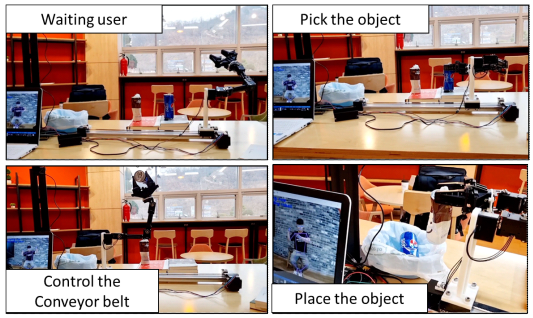

Fig. 7. Robot-Control Module Hardware

Each tester transfer two cans to the trash bin within a defined time frame. The system's performance is measured by counting the number of successful and failed attempts. The time limit for transferring the two cans is set to 2 minutes, and dropping a can or failing to place it inside the trash bin are considered a failure.

Table 3 presents the number of succeed and failed attempts, as well as the success rate, during the 30 trials for 3 people. All of failures were due to the cans falling over and the manipulator reaching its operational limit, resulting in the inability to grasp the cans again.

Table 3. Result of Iterative Execution

Number of Repetitions	Success	Fail
30	25	5
30	26	4
30	23	7
Success Rate(%)		82.2%

Ⅳ. Conclusion

This paper proposes a technique for controlling a robot based on the extraction of user joint coordinates using the MediaPipe API, a deep learning-based 2D object recognition API. The proposed system utilizes a camera to acquire information related to the user's movements. The API and Python code are then used to process this information into joint rotation angles for controlling the robot. The control values saved as a csv file could be modified as needed, allowing the operations for each process to be combined and performed sequentially, which could then be executed

repeatedly. It's expected to support fast robot teaching to help manufacture.

Since the coordinates are extracted from 2D images, proposed system have a potential limitation in accuracy if all the user's joints are not fully visible in the frontal view. To solve this issue, it may be necessary to use multiple cameras to improve recognition rates in blind spots. At this time, if the angle of the joint can be measured, other types of sensors other than cameras can be used.

In addition, there were difficulties in precisely driving the end-effector from a remote location due to latency and limitations in the operating range when controlling the robot. To overcome this, we are working on the research of Grasping planning algorithm with reinforcement learning. Which is attach the sensor on the end-effector of the robot and when the end-effector closer to the object, the size and shape of the object are recognized and the robot automatically pick the object.

References

[1] M. Agrawal, K. Eloot, M. Mancini, and A. Patel, *Industry 4.0: Reimagining manufacturing operations after COVID-19* (2020), Retrieved May 3, 2023. (https://mck.co/3gqSPSn)

[2] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505-6519, 2018. (https://doi.org/10.1109/ACCESS.2017.2783682)

[3] T. Fegade, Y. Kurle, S. Nikale and P. Kalpund, "Wireless gesture controlled Semi-Humanoid Robot," *2016 ICRAIE*, pp. 1-5, Jaipur, India, 2016. (https://doi.org/10.1109/ICRAIE.2016.7939511)

[4] P. Sihombing, R. B. Muhammad, H. Herriyance, and E. Elviwani, "Robotic arm controlling based on fingers and hand gesture," *2020 3rd Int. Conf. MECnIT*, pp. 40-45, Medan, Indonesia, 2020. (https://doi.org/10.1109/MECnIT48290.2020.9166592)

[5] M. K. Sudarshan, V. Vinamraa, M. Adarsh, V. H. Mer, and M. Deepak, "Wireless gesture controlled robotic arm," *2022 IEEE North Karnataka Subsection Flagship Int. Conf. (NKCon)*, pp. 1-6, Vijaypur, India, 2022. (https://doi.org/10.1109/NKCon56289.2022.10126678)

[6] H. Song, W. Feng, N. Guan, X. Huang, and Z. Luo, "Towards robust ego-centric hand gesture analysis for robot control," *2016 IEEE ICSIP*, pp. 661-666, Beijing, China, 2016. (https://doi.org/10.1109/SIPROCESS.2016.7888345)

[7] M. Altayeb, "Hand gestures replicating robot arm based on MediaPipe," *IJEEI*, vol. 11, no. 3, 2023. (https://doi.org/10.52549/ijeei.v11i3.4491)

[8] Q. Weiming, Z. Xiaomei, H. Jiwei, and L. Ping, "Real-time virtual UR5 robot imitation of human motion based on 3D camera," *2020 5th ICMCCE*, pp. 387-391, Harbin, China, 2020. (https://doi.org/10.1109/ICMCCE51767.2020.00092)

[9] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *CVPR Wkshp. Comput. Vis. for Augmented and Virtual Reality*, Seattle, WA, USA, 2020. (https://doi.org/10.48550/arXiv.2006.10214)

[10] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," *J. Softw. Eng. Robotics*, vol. 5, no. 1, pp. 3-16, May 2014. (https://doi.org/10.6092/JOSER_2014_05_01_p3)

[11] S. Edwards, et al., *Universal Robot*(2014), Retrieved May 3, 2023. (https://github.com/ros-industrial/universal_robot)

[12] ROBOTIS, *OpenMANIPULATOR-X* (2017), Retrieved May 3, 2023. (https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/)

Jin Su Park

Aug. 2016 : B.S. degree, School of Electronical Communication, Kumoh National Institute of Technology, Gumi, South Korea.

Sept. 2022~Current : M.Eng. student, Dept. of IT Convergence, Kumoh National Institute of Technology, Gumi, South Korea.

<Research Interest> Deep Learning, Image Processing, Manipulator

[ORCID:0009-0008-9806-4817]

Soo Young Shin

Feb. 1999 : B.Eng. degree, School of Electrical and Electronic Engineering, Seoul National University.

Feb. 2001 : M.Eng. degree, School of Electrical, Seoul National University.

Feb. 2006 : Ph.D. degree, School of Electrical Engineering and Computer Science, Seoul National University.

July 2006~June 2007 : Post Doc. Researcher, School of Electrical Engineering, University of Washington, Seattle, USA.

2007~2010 : Senior Researcher, WiMAX Design Laboratory, Samsung Electronics, Suwon, South Korea.

Sept. 2010~Current : Professor, School of Electronic Engineering, Kumoh National Institute of Technology.

<Research Interest> 5G/ 6G wireless communications and networks, signal processing, the Internet of Things, mixed reality, and drone applications.

[ORCID:0000-0002-2526-2395]