Real-Time Kinect Fingers Tracking and Recognition for Servo Gripper Controller

Rosdiyana Samad, Lim Siong Hee, Mahfuzah Mustafa, Dwi Pebrianti, Nor Rul Hasma Abdullah and Nurul Hazlina Noordin

Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia.

rosdiyana@ump.edu.my

***Abstract*—The conventional method to control something like machines such as remote controllers or wearable sensors have its limitation and cannot cater the high demands in some scenarios. To overcome this situation, a real-time Kinect finger tracking and recognition is developed to control a servo gripper via a single board microcontroller, Arduino. This paper presents hand and finger tracking method that can determine the number of fingers and also the angle of the finger's position. The hand gesture is captured by the Kinect and go through a series of image processing and finally, the information will send to the Arduino. The image processing method includes a detection of hand using depth sensor on the Kinect, then finds the finger and calculates the angle of the finger's position. The convex hull algorithm is used to represent the region of the hand. The fingertips are recognized by calculating the angle of the fingertip and each angle is compared with threshold angle. The orientation and position of the fingers are obtained by finding the middle line of the finger and compares to the vertical line of the middle finger, then calculates the angle. The result shows that the number of fingers appeared in the display can be recognized and the orientation and position of each finger can be determined. A gripper also able to react simultaneously (open and close) based on the detected finger.**

***Index Terms*—Finger Tracking; Gesture Recognition; Kinect; Real-Time; Servo Gripper Control.**

I. INTRODUCTION

Nowadays, the development of the technology is growing rapidly and many advanced electronic devices have been developed to make human life become easier and comfortable. A human-computer interface (HCI) has always played an important role in people's lives, gives a positive impact and motivates many researchers in this area. There are many types of a control device such as keyboard, mouse, gamepad, remote control and many more. All of these are capable to make our life more comfortable and easy [1]. However, these types of control devices can be enhanced by implementing hand gesture recognition to control device, machine, computer or application without relying on conventional devices.

Recently, many wearable devices, egocentric cameras and virtual reality glasses have emerged in research fields such as computing, computer vision, robotics, rehabilitation and biomedical engineering. These devices may be worn under, over or in clothing. In earlier generations, these devices were too bulky to be worn but recent technological advances have made these devices lightweight and more suitable for daily use [2]. Although these devices are available in the market, however, some of them are pricey. There are some limitations, hardware complexities, some devices are wired and also need to be properly calibrated to make exact gesture recognition [3].

This paper proposes a prototype of the human-computer interface system that implements hand gestures in controlling a servo gripper. This system tracks the hand and finger, then recognizes the finger instruction to control the opening and closing of servo gripper. The Arduino UNO acts as microcontroller receives information from the finger gesture and transfer this information to the servo gripper. The implementation of hand gestures increases the freedom of users, where they do not have to wear the device that may cause uncomfortable to the users. In this method, Kinect sensor is used to provide the RGB and depth images. Kinect sensor is developed by Microsoft as Xbox 360 peripheral. In recent years, there is an increasing number of Kinect's applications in some areas, such as game entertainment, health and fitness [4]. This is because Kinect provides visual information and also depth (distance) information that makes the application more stable and accurate. In addition, it is relatively low in price compares to other devices available in the market.

II. RELATED WORKS

There is various hand and finger gesture recognition methods have been proposed recently. Most of the hand gestures can be recognized very well by using wearable devices such as glove, accelerometer and gyroscope. In addition, the position of the hand and orientations can be measured too. Although most of the previously proposed methods obtained more precise and accurate results, those methods limited the freedom of the user in conducting gesture and applications.

To make an application becomes more user-friendly, low cost and get a better result, many researchers choose Kinect sensor as the input device for their application or system. R. Afthoni et. al [5] utilized a Kinect sensor in their control system for manipulator robot based on the proportional-derivative control algorithm. The data from Kinect was processed by using inverse kinematics for mapping the position of user's joints to the manipulator robot. M. M. Ali et. al [6] developed a system that used the H20 mobile robots equipped with Kinects as visual sensors to recognize the required target and identify its pose related to the robot body. The H20 robot had dual arms where each arm consists of 6 revolute joints with 6-DOF. The detection and localization methods were performed for single colored and textured targets related to the handle to be grasped and to the placing holder.

In 2013, M. V. Liarokapis et. al [7] developed a teleoperation and telemanipulation with a robot arm and robot hand was performed by using a human to the robot motion mapping scheme. Two position trackers were used to capture position with the orientation of human end-effector (wrist) and human elbow in 3D space. A data glove was able to capture human hand kinematics. The human hand kinematics were captured with the CyberGlove II (Cyberglove Systems) data glove, which had 22 sensors measuring all joint values of all fingers. The efficacy of the proposed methods was experimentally validated. 3 years later, Chunxu Li et. al [4] implemented Kinect sensor in teleoperation of Nao robot, which was a humanoid robot with 5° of freedom (DOF) for each arm. This system was an online system that tracked user's head and arm to interact with Nao robot. The operator can move their head and arm in a natural way to manipulate the position of the Kinect sensor. The result showed that the Nao robot imitated the human movement more accurately. Other researchers had proposed a hand gesture with Kinect for different application such as a new character input system based on hand tapping gestures for Japanese hiragana and English characters that can be used to facilitate human-computer interaction [8]. The hand tapping gestures are motions for tapping keys on aerial virtual keypads by hands, which can be effectively used as a hand alphabet by anyone, including hearing impaired individuals. The users can effectively interact with computers by using a non-touch input system where only the Kinect sensor was used without any keyboard, mouse or body-worn device. The fingertip detection accuracies for the right and left hands were 97.0% and 96.3% respectively. However, a little problem occurred occasionally where the fingers were overlapped, which results in miscounting the number of stretched fingers. Z. Lai et. al [9] implemented Discrete Curve Evolution (DCE) method in their fingertips detection and hand recognition with Kinect sensor, which can eliminate contour noise and contour distortions. The fingertips were directly detected on the original contour curve and improved the precision of detection.

III. METHODOLOGY

The proposed methodology is divided into 4 stages, which are hand detection and tracking, finger recognition, Kinect and Arduino communications and controlling a gripper. This project is developed by using Microsoft 2012, Arduino software, OpenNI and OpenCV libraries.

A. *Hand Detection & Tracking*

The proposed system starts a hand detection and hand tracking with the use of NITE library [10]. The gesture detection begins by initializing 2 gestures, which are waving and pushing hands. The waving hand detector interprets hand movement as left to right waving carried out at least four times in a row, while pushing hand detector recognizes hand movement as pushes towards and away from the sensor.

After the initialization, the NITE gesture detection algorithm is able to track the hand position and return the information of the position of the middle part of the palm. The NITE gestures are derived from a stream of hand points and track a hand move through space over time. Each hand point is in the real-world 3D coordinate of the center of the palm.

To get the hand point of interest, a threshold is applied to the depth image, where the coordinate of the hand depth is used as a reference point. The range value of the threshold is +2 and -2 from the z-axis coordinate of the hand point and it is used to remove another unwanted region. The process of the threshold is shown in Figure 1(a) and 1(b). Figure 1(a) shows a threshold image with -2 from the reference depth point while Figure 1(b) shows threshold with +2 from the reference depth point. Then the hand image is transformed into a white image and then the hand contour is detected in the image. Figure 1(c) shows threshold hand's image. Figure 1(d) shows the image after the contour is applied.

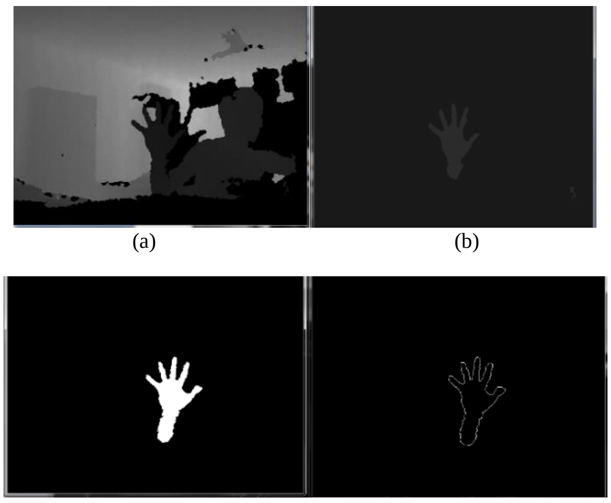


Figure 1: Image processing (a) Threshold with -2 from reference point (b) Threshold with +2 from the reference point (c) Threshold hand's image (d) Hand's contour is detected.

B. *Fingers Recognition*

To recognize gestures of the fingers, the convex hull and convex defect algorithm are used to detect the fingers. Convex hull and the convex defect will return several contour points (peaks and valley points) from the image that are recognized as the fingers and is used in gesture recognition.

In OpenCV, the starting point is calculated at 0° from the center of gravity (COG). To find these contour points it can be searched clockwise or anti-clockwise. The Figure 2 shows an example of COG of a hand and the orientation is started from the x-axis. Figure 3 shows a hand contour with convexity defects. Convexity defect is a cavity in an object (blob, contour) segmented out from an image. The area that does not belong to the object, but located inside of its outer boundary (convex hull) [11], [12]. The areas between fingers (all marked with arrows) of a hand contour are the convexity defects as shown in Figure 3.

Figure 4 shows the image of a hand, where each of the starting points, the deepest points and end points are joined together with blue lines, where the yellow circle is the deepest point and the purple circle is the starting point and the ending point.

To verify whether the point belongs to a finger, a few conditions are set before this point is set at a point of the fingertip. In general, the shape of a finger is thin and long, therefore, to verify it as a finger the deepest point should be bigger than a certain value. The angle Θ of the finger is calculated using Equation 1, where x and y are vectors and the angle must be smaller than 55°. This angle value is used to confirm the start point is calculated at a fingertip. Figure 5 shows a sample of finger detection.

$$\theta = \cos^{-1}\left[\frac{\bar{x}.\bar{y}}{|x||y|}\right] \quad (1)$$

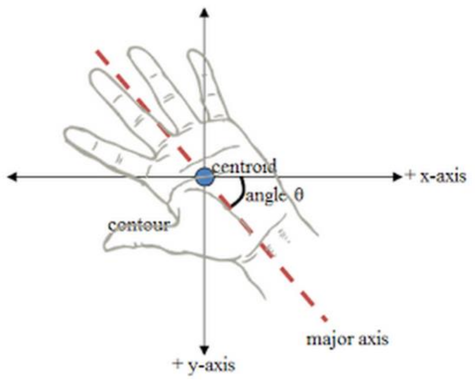


Figure 2: COG of a hand

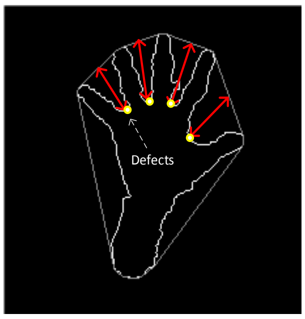


Figure 3: Convex defect.

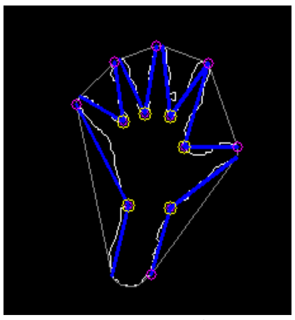


Figure 4: Convex defect on hand image.

Next, to find the finger pointing angle, a middle point from the two deepest points is calculated and then a new line is drawn from the start point to the middle point as shown in Fig. 6. Then, the new line is compared to the vertical axis and then calculates the angle using Equation 1. Figure 6 shows how the angle of the finger is calculated.

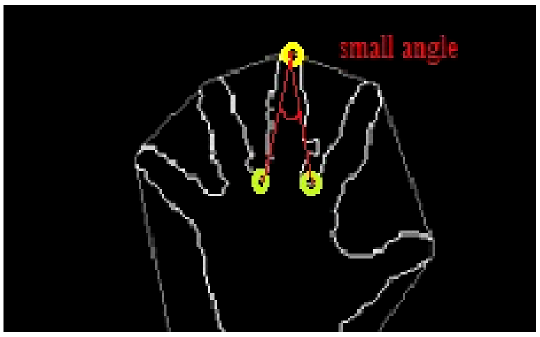


Figure 5: Sample of finger detection

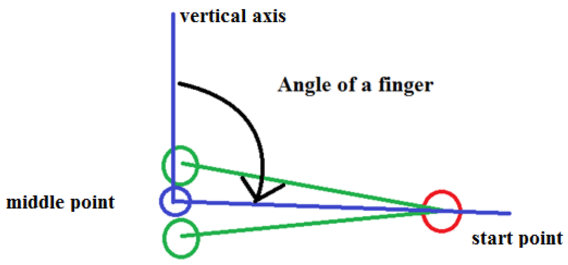


Figure 6: Finding of the angle of the finger

C. *Communication Between Kinect and Arduino*

A bridge called serial.cpp is created in the Microsoft Visual Studio to enable the data is transferred to Arduino. The serial.cpp contains the serial port number and baud rate values. These values must same with the setting of Arduino UNO so that the data can be transferred from Microsoft Visual Studio software to the Arduino UNO [12]. This bridge is able to get the feedback from the Arduino too but is not implemented in this system. The Microsoft Visual Studio will send the reading of the hand gesture (the number of fingers) to the Arduino UNO uses this bridge to control the gripper. This data transfer will show that the Kinect is able to control a controller Arduino UNO.

D. *Controlling a Gripper*

To control the gripper, the Arduino UNO is programmed so that it will move the gripper base on the input. The baud rate is set at 115200 Hz, which is same as in the Microsoft Visual Studio software. In the Arduino UNO, the port 9 is set as PWM which is used to control the servo motor. The servo motor is attached to the gripper and it is able to open about 90° widths. Therefore, the maximum value for the gripper to open is 90° and 0° to completely close. If 0 finger is detected, the gripper will close. If 5 fingers are detected, the gripper will open. If the fingers are detected between 1 and 4, the gripper is set to be close, the same condition with 0 finger detection.

IV. Results & Discussions

In this system, the hand is recognized using convex hull to find the outermost point and used it as a reference. However, the human fingertip is not perfectly sharp, but a bit curvy. This caused multiple points are detected at the fingertip as shown in Figure 7. There also many unwanted points are formed at the bottom of the wrist. These unwanted points have been removed to avoid false detection. After all the unwanted points are removed, the problem of multiple points in the same area is dissolved. Figure 8(a) shows an image with all the unwanted point are removed and Figure 8(b) shows a distant hand that is working fine.

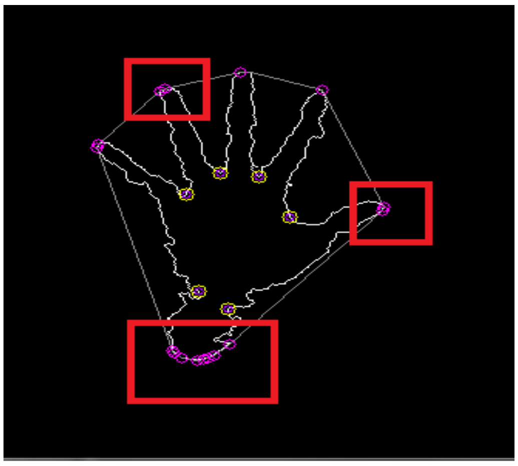


Figure 7: Multiple points on the hand image

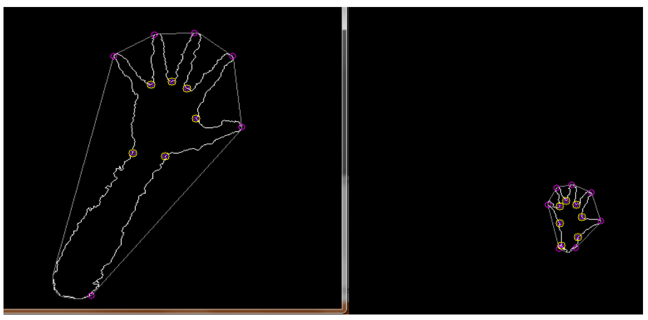

(a) (b)

Figure 8: Hand image (a) After the unwanted points are removed (b) A long distance hand image after removed unwanted points.

From the finger points that have been obtained, any detected point that meets the characteristic of a finger, which is the angle (between 2 vector lines) is smaller than 55° will be labelled as the number of fingers. The label starts from the right, parallel with an x-axis and each number of fingers are calculated in an anti-clockwise direction. Figure 9 shows a hand image with labels on the tips of the finger. All the fingertip points are connected in blue line to make it easier to see.

The label number on the fingertip can help to determine the number of fingers, but it cannot determine the finger name such as the thumb, index, middle, ring and little finger. This is due to how the convex hull is found the point. If the hand moves over the x-axis as shown in Figure 10, the numbering will still start with 1 at the index finger, followed by middle (2), ring (3) and little (4) fingers. The thumb will be the last with label number 5.

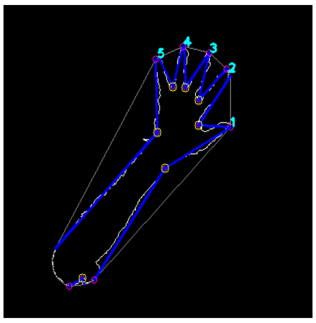


Figure 9: Hand with a label on each finger.

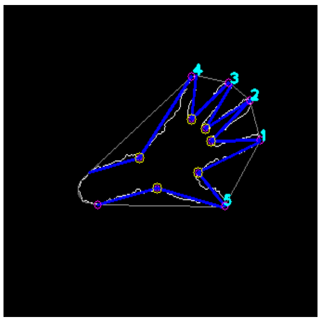


Figure 10: Labelling does not follow the fingers arrangement but it starts to count from the x-axis

After labelling the number of fingers, the angle of each finger is calculated. For an example in Figure 11 shows the white small circle that acts as the starting point and the purple small circle as the middle point of the two grey small circles (defects points). Both points form a straight line in the middle and it is used for the angle calculation. Then, the angle is compared with the y-axis and the value of the angle is displayed on each fingertip. From the Figure 11, it can be seen that the middle point is based on the defects points, but the defects point may not exist at the bottom part of the thumb and little finger (near the wrist). This may cause the angle calculation for these two fingers is different from the actual angle. Finally, all labels (angle and finger number) are combined together in a single image. Figure 12, 13 and 14 show different gestures of fingers with the label of finger number and angle.

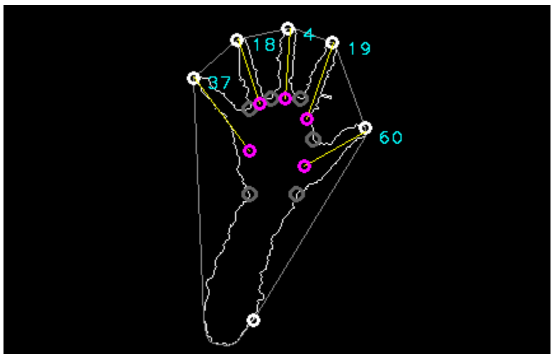


Figure 11: Angel of each of the finger pointing

The image stream of Kinect sensor is 30 frames/sec. The Arduino UNO had to control a servo motor that took time to move to a certain position, but the information from the Kinect is too much for the Arduino to handle and it caused delays to both Arduino and the Microsoft Visual Studio software. Sometimes, the Microsoft Visual Studio also stopped functioning due to Arduino unable to receive the data. To overcome this problem, the sum of data (finger number and angle) taken from 10 sequences of image frames is averaged before send it to the Arduino. This not only solved the problem but also give a better and accurate value.

The command prompt window that is shown in Figure 15 shows the connection between the Arduino and the Microsoft Visual Studio is established and the serial port number also display for the confirmation (if more than 1 Arduino board is used). Then, the number of tracked fingers is displayed in this window as shown in Figure 16. This message will not appear if the connection is not established and the system would not be able to run. In this command prompt window, the number of fingers that had been calculated were sent to the Arduino. These values were displayed so that the user can confirm that the calculated values are similar and correct with the data sent.

The gripper is opened and closed based on the detected finger. The gripper is following the finger gesture correctly when functioning. When there is no finger is detected, the gripper is closed and when five fingers are detected, the gripper is opened to the maximum. Figure 17(a) shows a close gripper when no finger is found, and Figure 17(b) shows an open gripper when five fingers are detected. This proves that this system is able to perform opening and closing gripper in real time based on the Kinect input.

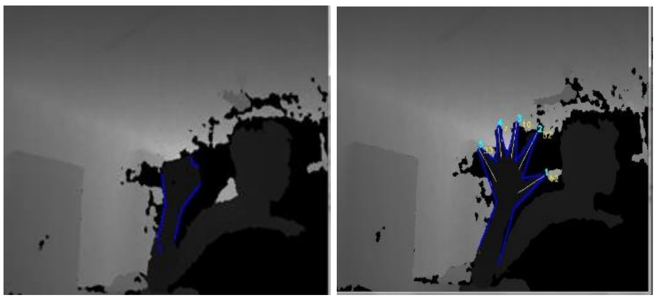
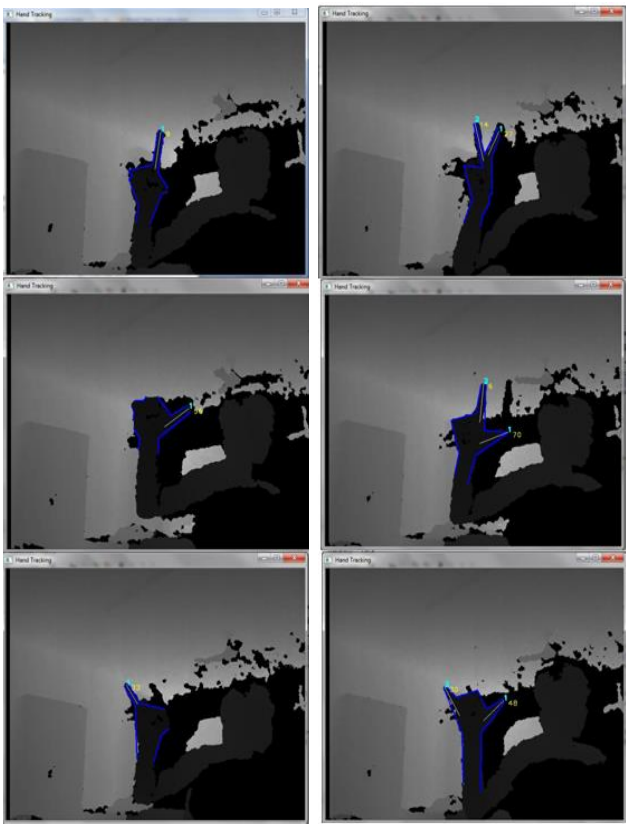
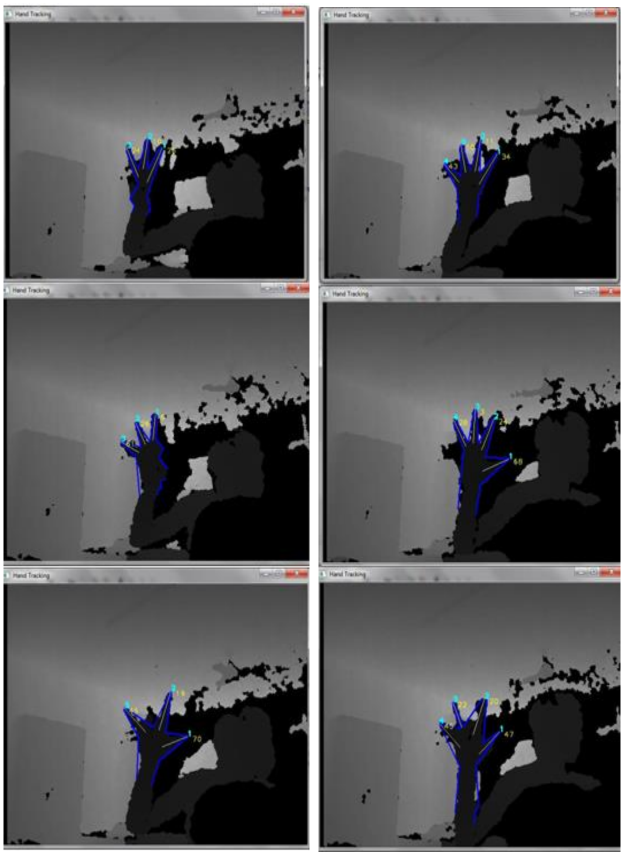

(a) (b)

Figure 12: Hand gesture with (a) no finger (b) 5 fingers.



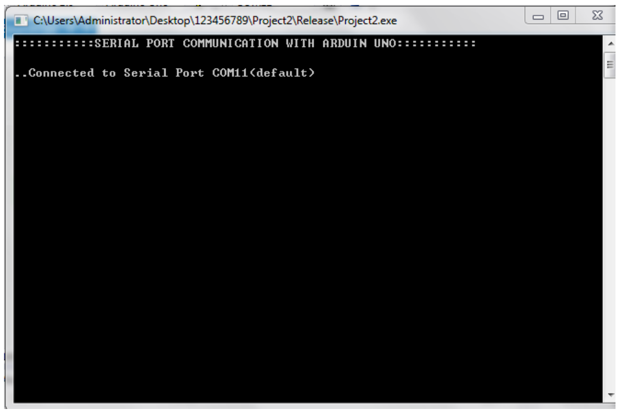
(a) (b)

Figure 13: Hand gesture with (a) different gesture of 1 finger (b) different gesture of 2 fingers.



(a) (b)

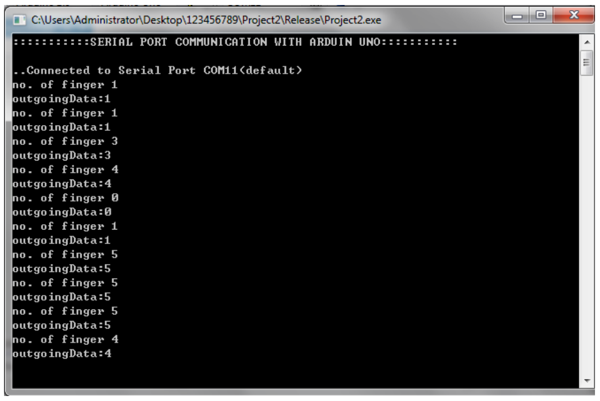
Figure 14: (a) Different gesture of 3 fingers (b) Different gesture of 4 fingers.



```
C:\Users\Administrator\Desktop\123456789\Project2\Release\Project2.exe
::::::::::SERIAL PORT COMMUNICATION WITH ARDUIN UNO::::::::::

..Connected to Serial Port COM11(default)
```

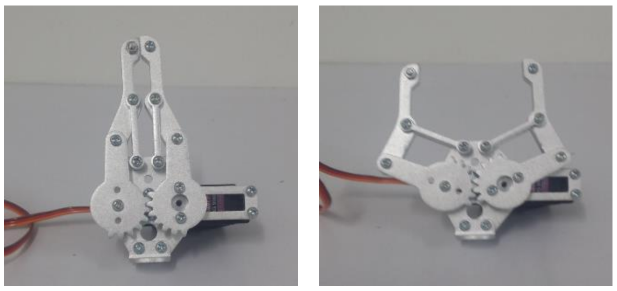
Figure 15: Show that the connection between Arduino and visual is established.



```
C:\Users\Administrator\Desktop\123456789\Project2\Release\Project2.exe
::::::::::SERIAL PORT COMMUNICATION WITH ARDUIN UNO::::::::::

..Connected to Serial Port COM11(default)
no. of finger 1
outgoingData:1
no. of finger 1
outgoingData:1
no. of finger 3
outgoingData:3
no. of finger 4
outgoingData:4
no. of finger 0
outgoingData:0
no. of finger 1
outgoingData:1
no. of finger 5
outgoingData:5
no. of finger 5
outgoingData:5
no. of finger 5
outgoingData:5
no. of finger 4
outgoingData:4
```

Figure 16: Number of fingers count and send the value to the Arduino.



(a) (b)

Figure 17: Gripper (a) A close gripper (b) An open gripper.

V. CONCLUSION

For a conclusion, a hand gesture recognition is successfully developed using Kinect sensor and the finger gesture was able to control a gripper. The use of depth information was an easy way of recognizing and tracking a hand gesture and the noise in-depth image in relatively lesser compared to other methods. The convex hull and convex defect were able to discriminate the fingertip from the center of the hand and then calculated the number of the fingers correctly. A gripper was also able to react simultaneously based on the input of the Kinect sensor.

Using the NITE library for tracking the hand gesture was able to make the user moves around the range of Kinect without the loss of the hand detection. This provides a great freedom to the user where they can control the gripper from range. This system was also able to apply to a robot where a place unreachable by a human yet the user remains safe, for example exploring the deep sea or in hazardous environments.

ACKNOWLEDGMENT

This research is supported by a Fundamental Research Grant Scheme (FRGS) (RDU160143) and Faculty of Electrical and Electronic Engineering, Universiti Malaysia Pahang (www.ump.edu.my).

REFERENCES

[1] C. A. Burande, R. M. Tugnayat, and N. K. Choudhary, "Advanced recognition techniques for human computer interaction," *in Proc. of The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010, pp. 480-483.

[2] Bulling and K. Kunze, "Eyewear computers for human-computer interaction," *in Interactions*, pp. 70-73, 2016. DOI=http://dx.doi.org/10.1145/2912886.

[3] M. M. F. M. Fareed, Q. I. Akram, S. B. A. Anees and A. H. Fakih, "Gesture based wireless single-armed robot in cartesian 3D space using Kinect," *in Proc. of The Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp.1210-1215

[4] C. Li, C.Yang, P. Liang, A. Cangelosi and J. Wan, "Development of Kinect based teleoperation of Nao robot," *in Proc. of International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2016, pp.133-138.

[5] R. Afthoni, A. Rizae and E. Susanto, "Proportional derivative control based robot arm system using Microsoft Kinect," *in Proc. of International Conference on Robotics, Biomimetics, Intelligent Computational Systems (ROBIONETICS),* Yogyakarta, Indonesia, 2013, pp. 24-29.

[6] M. M. Ali, H. Liu, N. Stoll, K. Thuro, "Intelligent arm manipulation system in Life Science Labs using H20 mobile robot and Kinect sensor," *in Proc. of IEEE 8th International Conference on Intelligent Systems,*2016, pp.382-387.

[7] M. V. Liarokapis, P. K. Artemiadis, and K. J. Kyriakopoulos, "Mapping human to robot motion with functional anthropomorphism for teleoperation and telemanipulation with robot arm hand systems," *in Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS),* 2013, pp. 2075-2075.

[8] J. Shin and C. M. Kim, "Non-touch character input system based on hand tapping gestures using Kinect sensor," *in IEEE Access Open Access Journal*, 2017. Doi: 10.1109/ACCESS.2017.2703783.

[9] Z. Lai, Z. Yao, C. Wang, H. Liang, H. Chen and W. Xia, "Fingertips detection and hand gesture recognition based on Discrete Curve Evolution with a Kinect sensor," *in Proc. of The International Conference on Visual Communications and Image Processing (VCIP),* 2016.

[10] PrimeSense, "OpenNI NiTE2.2.0.11," 2012. [Online]. Available: http://openni.ru/files/nite

[11] W. Yan, H. Chuanyan, Y. Guanghui, and W. Changbo, "A robust method of detecting hand gestures using depth sensors," *in Proc. of IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE),* 2012, pp. 72-77.

[12] R. Hartanto, A. Susanto, and P. I. Santosa, "Real time hand gesture movements tracking and recognizing system," *in Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS),* 2014, pp. 137-141

[13] Arduino, "Arduino Uno Rev3,". 2015. [Online]. Available: http://www.arduino.cc/en/Main/ArduinoBoardUno