# การออกแบบและพัฒนาระบบควบคุมแขนกลระยะไกล โดยใช้การจับคู่แบบจลนศาสตร์ต่างกันด้วยท่าทางของมือของมนุษย์

นางสาวธัญญ์นภัส จารุโชคภูริวัจน์



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเมคคาทรอนิกส์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2566

# DESIGN AND DEVELOPMENT OF HUMAN GRIPPER CONTROL FOR NON-HOMOGENOUS KINEMATICS ROBOT ARM WITH TELEOPERATION

TANNAPAT JARUCHOKPHURIWAJ



A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Mechatronics Engineering
Suranaree University of Technology
Academic Year 2023

**การออกแบบและพัฒนาระบบควบคุมแขนกลระยะไกล โดยใช้การจับคู่แบบจลนศาสตร์ต่างกันด้วยท่าทางของมือของมนุษย์**

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

.....

(ผศ. ดร.ชัยยุทธ สัมภาวะคุปต์)
ประธานกรรมการ

.....

(อ. ดร.จิตติมา วระกุล)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

.....

(ผศ. ดร.โศรฎา แข็งการ)
กรรมการ

.....

(รศ. ดร.ยุพาพร รักสกุลพิวัฒน์)
รองอธิการบดีฝ่ายวิชาการและประกันคุณภาพ

.....

(รศ. ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ธัญญ์นภัส จารุโชคภูริวัจน์ : การออกแบบและพัฒนาระบบควบคุมแขนกลระยะไกล โดยใช้การจับคู่แบบจลนศาสตร์ต่างกันด้วยท่าทางของมือของมนุษย์ (DESIGN AND DEVELOPMENT OF HUMAN GRIPPER CONTROL FOR NON-HOMOGENOUS KINEMATICS ROBOT ARM WITH TELEOPERATION)
อาจารย์ที่ปรึกษา : อาจารย์ ดร.จิตติมา วระกุล, 87 หน้า.

คำสำคัญ: แขนกล/ถุงมือควบคุม/การควบคุมระยะไกล/หุ่นยนต์/หุ่นยนต์แขนกล

งานวิจัยนี้ต้องการออกแบบและพัฒนาระบบควบคุมหุ่นยนต์ที่สามารถควบคุมแขนกลได้ในระยะไกล โดยใช้การจับคู่แบบจลนศาสตร์ต่างกัน เนื่องจากแขนกลถูกนำไปใช้งานหลากหลายมากขึ้น แต่ในบางงานที่มีความสลับซับซ้อนยังต้องการการควบคุมด้วยมนุษย์ ซึ่งระบบควบคุมหุ่นยนต์นี้เหมาะกับการนำมาใช้ร่วมกับงานดังกล่าว ด้วยการรับตำแหน่งและท่าทางจากฝ่ามือของมนุษย์ และใช้เซ็นเซอร์วัดความเคลื่อนไหวและระบบระบุตำแหน่ง โดยอุปกรณ์ควบคุมที่ถูกใช้สำหรับการวิจัยเป็นอุปกรณ์ที่สวมใส่บริเวณมือร่วมกับเซ็นเซอร์วัดการเคลื่อนไหวและระบบกล้อง 3 มิติ ในการหาตำแหน่งเพื่อใช้ควบคุมแขนกลแบบ 6 องศาอิสระ ระบบทั้งหมดจะประกอบไปด้วยส่วนรับข้อมูลเซนเซอร์ ส่วนของการผสานข้อมูลและส่วนที่ใช้ในการส่งต่อข้อมูลเพื่อสั่งการแขนกลผ่าน หุ่นยนต์ที่ใช้ในการวิจัยคือ MyCobot280 Pi โดยมีเซนเซอร์วัดความเฉื่อยเป็น Witmotion JY61P กล้องวัดความลึกที่ใช้คือ Intel Realsense D435 กล้องจะสามารถตรวจจับมือได้ด้วยโมเดลการเรียนรู้ของเครื่องแบบ MediaPipe หลังจากนั้น จะนำเอาข้อต่อที่สามารถตรวจจับได้มาสร้างสมการความสัมพันธ์หาจุดอ้างอิงของมือเพื่อวัดระยะและหาการหมุนของมือในแนวแกน yaw ส่วนค่าของ roll กับ pitch จะวัดได้จากการนำข้อมูลของ IMU ไปใช้ร่วมกับตัวกรองแมดกวิก (Madgwick's filter) หลังจากนั้นข้อมูลทั้งหมดจะถูกนำมาใช้ในการติดตามการเคลื่อนที่ของมือด้วยตัวกรองคาลมาน (Kalman filter) ข้อมูลที่ได้จากตัวกรองคาลมานจะถูกส่งไปควบคุมแขนกลระยะไกลผ่านโปรโตคอล MQTT โดยมีโบรกเกอร์ในการส่งข้อมูลเป็น HiveMQ ซึ่งเป็นโบรกเกอร์ที่ทำงานผ่านระบบคลาวด์ ผลการทดสอบแสดงให้เห็นว่าค่าความคลาดเคลื่อนในการตรวจวัดตำแหน่งของมืออยู่ในช่วงที่ยอมรับได้ โดยมีค่าเฉลี่ยในแกน X, Y และ Z อยู่ที่ 2.44 มม., 2.22 มม., และ 2.33 มม. ตามลำดับ สำหรับมุม yaw ค่าความคลาดเคลื่อนเฉลี่ยอยู่ที่ 0.772 องศา และค่าความคลาดเคลื่อนในการวัดมุม pitch และ roll ด้วย IMU อยู่ในช่วง -0.60 ถึง 0.16 องศาในมุมต่ำกว่า 75 องศา ระบบการติดตามการเคลื่อนไหวของมือด้วยตัวกรองคาลมานมีความเสถียรและแม่นยำสามารถช่วยประมาณการเคลื่อนที่ของมุมในกรณีที่กล้องหลุดการติดตามจากมือชั่วขณะได้ เมื่อนำไปสั่งการ MyCobot Pi280 จำเป็นจะต้องลดความถี่ในการส่งข้อมูลการสั่งการลง เป็นทุก ๆ 4 วินาที เพื่อให้แขนกลสามารถเคลื่อนที่

ได้ทันและสุดท้ายระบบสามารถนำข้อมูลการเคลื่อนที่ของมือไปสั่งการแขนกลจากระยะไกลได้อย่างถูกต้องโดยเป็นอิสระจากจลนศาสตร์ที่ต่างกันได้



สาขาวิชาวิศวกรรมเมคคาทรอนิกส์ ลายมือชื่อนักศึกษา .....

ปีการศึกษา 2566 ลายมือชื่ออาจารย์ที่ปรึกษา.....

TANNAPAT JARUCHOKPHURIWAJ : DESIGN AND DEVELOPMENT OF HUMAN GRIPPER CONTROL FOR NON-HOMOGENOUS KINEMATICS ROBOT ARM WITH TELEOPERATION.
THESIS ADVISOR : JITTIMA VARAGUL, Ph.D., 87 PP.

Keywords: Robot arm/eleoperation/Gripper/Robotics/Humanoid robot/Non-homogenous

This research aims to design and develop a robotic control system capable of remotely controlling a robotic arm using non-homogenous kinematic matching. As robotic arms are increasingly used in various applications, some complex tasks still require human control. This robotic control system is suitable for such tasks by receiving positions and postures from the human hand using motion sensors and a positioning system. The control device used for the research is a wearable device on the hand, combined with motion sensors and a 3D camera system to determine positions for controlling a robotic arm with 6 degrees of freedom. The entire system consists of a sensor data acquisition part, a data integration part, and a part for transmitting data to control the robotic arm. The robot used in the research is the MyCobot280 Pi, with an inertial sensor Witmotion JY61P and a depth camera Intel Realsense D435. The camera can detect the hand using a machine learning model called MediaPipe. The detected joints are then used to create equations to find the reference point of the hand for measuring distance and rotation in the yaw axis. The roll and pitch values are measured using IMU data combined with Madgwick's filter. All this data is then used to track hand movements with a Kalman filter. The data from the Kalman filter is transmitted to control the robotic arm remotely through the MQTT protocol, with HiveMQ as the cloud-based broker.Test results show that the position measurement errors for the hand are within acceptable ranges, with average errors in the X, Y, and Z axes at 2.44 mm, 2.22 mm, and 2.33 mm, respectively. For the yaw angle, the average error is 0.772 degrees. The pitch and roll angle errors measured with the IMU are within -0.60 to 0.16 degrees for angles below 75 degrees. The hand movement tracking system with the Kalman filter is stable and accurate, helping to

estimate movement angles when the camera momentarily loses track of the hand. When applying controlling to the MyCobot Pi280, it is necessary to reduce the command transmission frequency to every 4 seconds to ensure that the robotic arm can move in time. Finally, the system can correctly command the robotic arm's movements remotely, independent of the different kinematics.



School of Mechatronics Engineering
Academic Year 2023

Student's Signature.....
Advisor's Signature.....

# กิตติกรรมประกาศ

การทำวิทยานิพนธ์นี้สำเร็จลุล่วงได้ด้วยความช่วยเหลือในการให้คำปรึกษา คำแนะนำ รวมถึงสนับสนุนด้านวิชาการ ด้านการดำเนินงาน และด้านจิตใจ จนกระทั่งทำให้วิทยานิพนธ์นี้สามารถบรรลุตามวัตถุประสงค์ได้ จึงขอแสดงความขอบคุณต่อบุคคลดังต่อไปนี้

อาจารย์ ดร.จิตติมา วระกุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้ความรู้ คำปรึกษา คำแนะนำ ความคิดเห็นที่เป็นประโยชน์เกี่ยวกับแนวทางการทำวิจัย แนะแนวทางที่เหมาะสมและปรับปรุงขอบเขตของงานวิจัยให้สอดคล้องกับจุดประสงค์ของงานวิจัยเพื่อประโยชน์สูงสุดต่องานวิจัย ตัวผู้วิจัย และผู้ที่จะนำงานวิจัยไปต่อยอด งบประมาณในการสนับสนุนในการจัดทำวิทยานิพนธ์ รวมไปถึงช่วยตรวจสอบเอกสาร บทความวิจัย และวิทยานิพนธ์จนสำเร็จลุล่วงสมบูรณ์

รองศาสตราจารย์ ดร.จิระพล ศรีเสริฐผล หัวหน้าสาขาวิชาวิศวกรรมเมคคาทรอนิกส์ ที่ให้การสนับสนุนและแสดงความเป็นห่วงต่อผู้วิจัย ทั้งยังช่วยให้คำปรึกษา คำแนะนำ และแนะแนวทางในการทำวิจัยเพื่อให้งานวิจัยสามารถบรรลุวัตถุประสงค์ตามที่ผู้วิจัยต้องการ

อาจารย์ ดร.ณัฐวัฒน์ พิณรัตน์ อาจารย์สาขาวิชาวิศวกรรมอุตสาหการ ที่ได้ให้การสนับสนุนและอำนวยความสะดวกทางด้านสถานที่และอุปกรณ์ที่เกี่ยวข้องกับงานวิจัย

อาจารย์อภิลักษณ์ หล่อนกลาง อาจารย์สาขาวิศวกรรมเครื่องกล ที่ให้คำปรึกษา คำแนะนำ เป็นผู้รับฟังและให้การสนับสนุนในด้านต่าง ๆ จนกระทั่งสามารถจัดทำวิทยานิพนธ์จนสำเร็จ

นายภูวนาถ เผือกทอง เพื่อนบัณฑิตศึกษาที่ช่วยเหลือในการให้คำปรึกษา คำแนะนำ ทั้งด้านความรู้และกำลังใจในการทำวิจัยและจัดทำวิทยานิพนธ์ให้สำเร็จลุล่วง อำนวยความสะดวกในด้านต่าง ๆ รวมทั้งยังเป็นจุดเริ่มต้นและแรงบันดาลใจให้ผู้วิจัยมีความสนใจในการศึกษาต่อระดับปริญญาโท

นางสาวปาณิศา สิทธิกมล ที่ให้ความช่วยเหลือในการคำปรึกษา วางแผนการเรียน ให้กำลังใจ เป็นผู้รับฟังและให้การสนับสนุน รวมถึงช่วยอำนวยความสะดวกในด้านการศึกษาจนสามารถจัดทำวิทยานิพนธ์สำเร็จ

บิดา มารดา คุณยาย น้องชายและญาติพี่น้องที่คอยให้ความช่วยเหลือ ให้กำลังใจ สนับสนุนด้านต่าง ๆ เป็นทั้งที่ปรึกษาในชีวิตและแรงบันดาลใจในการทำให้สำเร็จ ทั้งยังช่วยอำนวยความสะดวกอย่างสุดความสามารถ เพื่อให้ผู้วิจัยสามารถศึกษาและทำวิจัยสำเร็จได้อย่างราบรื่น

สุดท้ายนี้ ขอขอบคุณอาจารย์และทุกท่านที่ไม่ได้กล่าวถึง ณ ที่นี้ ที่มีส่วนช่วยในการสนับสนุนและอำนวยความสะดวกในด้านต่าง ๆ จนกระทั่งสามารถทำให้วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงได้ด้วยดี

ธัญญ์นภัส จารุโชคภูริวัจน์

# สารบัญ

**หน้า**


บทคัดย่อ (ภาษาไทย) ..... ก
บทคัดย่อ (ภาษาอังกฤษ) ..... ค
กิตติกรรมประกาศ ..... จ
สารบัญ ..... ฉ
สารบัญตาราง ..... ฌ
สารบัญรูป ..... ญ

**บทที่**

**1 บทนำ** ..... 1
1.1 ความเป็นมาและความสำคัญของปัญหา ..... 1
1.2 วัตถุประสงค์ของการวิจัย ..... 2
1.3 ขอบเขตของการวิจัย ..... 2
1.4 ประโยชน์ที่คาดว่าจะได้รับ ..... 2
1.5 แผนการดำเนินงาน ..... 3
1.6 การจัดรูปเล่มวิทยานิพนธ์ ..... 4

**2 ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง** ..... 5
2.1 Inertial Measurement Unit (IMU) ..... 5
2.2 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System) กับการใช้งานร่วมกับแขนกล ..... 7
2.3 แขนกล 6 องศาอิสระ MyCobot Pi 280 ..... 8
2.4 Servo Motor ..... 9
2.5 จลนศาสตร์ไปข้างหน้า (Forward Kinematics) ..... 10
2.6 จลนศาสตร์ผันกลับ (Inverse Kinematics) ..... 13
2.7 ตัวควบคุม PID ..... 17
2.8 IMU Filter Madgwick ..... 18
2.9 ตัวกรองคาลมาน (Kalman filter) ..... 21

# สารบัญ (ต่อ)

**หน้า**

2.10 กล้องวัดความลึก ..... 21
2.11 Hive MQ Broker ..... 23
**3 วิธีการดำเนินการ** ..... 25
3.1 ขั้นตอนการดำเนินงานวิจัย ..... 25
3.2 การออกแบบระบบควบคุมสั่งการ ..... 26
3.3 การออกแบบและสร้างอุปกรณ์สวมใส่ที่แขน ..... 26
3.4 การออกแบบและสร้างแท่นกล้องสำหรับการตรวจจับมือ ..... 27
3.5 การออกแบบระบบหาตำแหน่งและท่าทางของมือด้วยกล้องความลึก (depth camera) และ IMU ..... 28
3.6 การออกแบบระบบติดตามการเคลื่อนที่ของมือ (hand tracking system) ..... 30
3.7 การออกแบบระบบส่งผ่านข้อมูล ..... 33
3.8 การทดสอบความแม่นยำของการตรวจจับด้วยกล้องวัดความลึก ..... 35
3.9 การทดสอบความแม่นยำของการตรวจจับมุมด้วย IMU ..... 36
3.10 การทดสอบการส่งผ่านข้อมูลและการควบคุม ..... 36
3.11 การทดสอบการควบคุมร่วมกับหุ่น MyCobot Pi280 ..... 36
**4 ผลการวิจัยและอภิปรายผล** ..... 38
4.1 กล่าวนำ ..... 38
4.2 ผลการทดสอบความแม่นยำของการตรวจจับด้วยกล้องวัดความลึก ..... 38
4.3 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ..... 46
4.4 ผลการทดสอบการติดตามการเคลื่อนที่ของมือ ..... 49
4.5 ผลการทดสอบการส่งข้อมูล ..... 51
4.6 ผลการทดสอบการทำงานร่วมกับหุ่นยนต์จริง ..... 52
**5 สรุปผลและข้อเสนอแนะ** ..... 57
5.1 สรุปผลงานวิจัย ..... 57
5.2 ข้อเสนอแนะ ..... 58
รายการอ้างอิง ..... 59

# สารบัญ (ต่อ)

**หน้า**

ภาคผนวก

ภาคผนวก ก. โปรแกรมภาษาไพทอนที่ใช้สำหรับผสานรวมข้อมูลเซนเซอร์เพื่อทำการกรองสำหรับการติดตามด้วยตัวกรองคาลมาน..... 63

ภาคผนวก ข. โปรแกรมภาษาไพทอนที่ใช้ในการดึงข้อมูลจาก depth camera สำหรับการระบุตำแหน่งและมุม yaw.....63

ภาคผนวก ค. โปรแกรมภาษาไพทอนที่ใช้ในการส่งข่อมูลจาก Kalman filter ไป HiveMQ ...74

ภาคผนวก ง. โปรแกรมภาษาไพทอนที่ใช้ในการรับข้อมูลจาก HiveMQ ไปควบคุมแขนกล....74

ภาคผนวก จ. บทความวิชาการที่ได้รับการตีพิมพ์เผยแผร่ในระหว่างศึกษา.....80

ประวัติผู้เขียน.....87

# สารบัญตาราง

**ตารางที่** **หน้า**


1.1 แผนการดำเนินงาน ..... 3
2.1 ข้อมูลของ JY61P Witmotion ..... 6
2.2 DH Parameter ของ Mycobot280 Pi ..... 13
4.1 ผลการทดสอบความแม่นยำในการตรวจวัดตำแหน่งของมือ ..... 38
4.2 สรุปผลการทดสอบการตรวจจับตำแหน่งด้วยกล้องวัดความลึก ..... 39
4.3 ผลการทดสอบความแม่นยำในการตรวจวัดมุม yaw ของมือ ..... 39
4.4 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน pitch ..... 46
4.5 สรุปผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน pitch ..... 47
4.6 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน roll ..... 48
4.7 สรุปผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน roll ..... 49
4.8 ผลการทดสอบการส่งต่อข้อมูลระหว่างระบบควบคุม ..... 51
4.9 ค่าข้อมูลตำแหน่งและท่าทางที่มีการนำไปสั่งการหุ่นยนต์ My Cobot Pi280 ..... 54

# สารบัญรูป

**รูปที่** **หน้า**


2.1 แกนตรวจจับข้อมูลของ Gyroscope ..... 5
2.2 ตัวอย่างการใช้งานชุดโปรแกรม MoveIt ..... 7
2.3 หน้าตาของ myCobot280 ..... 8
2.4 ระยะข้อต่อและพื้นที่การทำงาน ..... 9
2.5 ส่วนประกอบของเซอร์โวมอเตอร์ ..... 9
2.6 Closed-loop system ของเซอร์โวมอเตอร์และอุปกรณ์ป้อนกลับข้อมูล ..... 10
2.7 นิยามของตัวแปรในวิธีการ DH ..... 11
2.8 Kinematic Diagram ของหุ่นยนต์ KUKA ..... 13
2.9 ภาพมุมมองด้านบนของแกนที่ 5 ..... 14
2.10 ภาพฉายระนาบของข้อต่อที่ 6 ไปหาฐาน ..... 15
2.11 ตัวควบคุมพีไอดี ..... 17
2.12 block diagram โครงสร้างของ IMU Filter madgwick ..... 20
2.13 Baseline และ disparity ..... 23
3.1 ขั้นตอนการดำเนินงานวิจัย ..... 25
3.2 อุปกรณ์สวมใส่สำหรับวัดค่าด้วยเซนเซอร์ IMU ..... 27
3.3 อุปกรณ์สวมใส่สำหรับวัดค่าเมื่อสวมเข้ากับมือ ..... 27
3.4 รูป CAD ของแท่นอ้างอิงระนาบมือ ..... 28
3.5 จุดอ้างอิงของมือที่ได้จากโมเดล MediaPipe ..... 28
3.6 ตัวอย่างภาพความลึกจาก depth camera ..... 29
3.7 ตัวอย่างการหาตำแหน่ง XY และ Z ของมือด้วย depth camera ..... 29
3.8 ตัวอย่างการทดสอบหามุมของการหมุนของมือที่มุม 15 องศา ..... 30
3.9 แผนผังแสดงภาพรวมของการส่งข้อมูลของระบบ ..... 33
3.10 หน้าตาของ dashboard ของ HiveMQ ..... 34
3.11 ตัวอย่างหน้าจอ WebClient ที่ใช้ monitor การส่งข้อมูล ..... 34
3.12 ตัวอย่างหน้าจอ monitor การเคลื่อนที่ของแขนกล mycobot280 ..... 35

# สารบัญรูป (ต่อ)

**รูปที่** **หน้า**


3.13 จุดอ้างอิงสำหรับการตรวจวัดตำแหน่งด้วยกล้องความลึก ..... 35
3.14 ตัวอย่างการทดสอบวัดความแม่นยำการตรวจจับมุมของมือด้วยกล้องความลึก ..... 36
4.1 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 1 ..... 40
4.2 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 2 ..... 41
4.3 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 3 ..... 41
4.4 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 4 ..... 41
4.5 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 5 ..... 42
4.6 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 6 ..... 42
4.7 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 7 ..... 42
4.8 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 8 ..... 43
4.9 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 9 ..... 43
4.10 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 90 องศา ..... 44
4.11 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -90 องศา ..... 44
4.12 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -45 องศา ..... 44
4.13 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 15 องศา ..... 45
4.14 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 0 องศา ..... 45
4.15 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -30 องศา ..... 45
4.16 เส้นทางการเคลื่อนที่ของมือที่ได้จากตัวกรองคาลมาน แบบ 3 มิติ ..... 50
4.17 เส้นทางการเคลื่อนที่ของมือที่ได้จากตัวกรองคาลมาน แบบ 2 มิติ ..... 50
4.18 ความสัมพันธ์ของข้อมูลตำแหน่งเทียบกับเวลา ..... 53
4.19 ความสัมพันธ์ของข้อมูลท่าทางในมุมออยเลอร์เทียบกับเวลา ..... 53
4.20 ความสัมพันธ์ของข้อมูลตำแหน่งเทียบกับเวลาที่แขนกลรับไปปฏิบัติ ..... 54
4.21 ความสัมพันธ์ของข้อมูลท่าทางในมุมออยเลอร์เทียบกับเวลาที่แขนกลรับไปปฏิบัติ ..... 55
4.22 (ซ้าย) ภาพแขนกล (ขวา) มือที่ใช้ในการควบคุม ณ วินาทีที่ 48 ..... 55
4.23 (ซ้าย) ภาพแขนกล (ขวา) มือที่ใช้ในการควบคุม ณ วินาทีที่ 72 ..... 56

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

แขนกลเป็นหุ่นยนต์ประเภทหนึ่งที่ถูกออกแบบให้ทำงานแทนมนุษย์ เป็นเครื่องมือหนึ่งที่ช่วยเพิ่มประสิทธิภาพการทำงานเพื่อแก้ปัญหาการทำงานที่เป็นอันตรายต่อมนุษย์ งานที่ต้องทำซ้ำไปซ้ำมา งานที่ต้องทำต่อเนื่องเป็นเวลานาน รวมไปถึงงานที่เกินขีดกำจัดของมนุษย์ เช่น งานที่ต้องยกวัตถุที่หนัก เป็นต้น ซึ่งสามารถพบเห็นการใช้แขนกลอย่างกว้างขวางในภาคอุตสาหกรรมต่าง ๆ โดยเฉพาะในภาคอุตสาหกรรมการผลิต และในปัจจุบันที่การสื่อสารระยะไกลมีการพัฒนาและเติบโตขึ้นเรื่อย ๆ แขนกลจึงถูกนำมาพัฒนาร่วมกับการสื่อสารระยะไกลให้สามารถควบคุมแขนกลได้ในระยะไกลโดยการส่งข้อมูลผ่านเครือข่ายไร้สาย ไม่ว่าจะเป็นเครือข่ายอินเทอร์เน็ต สัญญาณโทรศัพท์ สัญญาณดาวเทียม เป็นต้น

ถึงแม้ว่าแขนกลจะสามารถตั้งค่าให้สามารถทำงานซ้ำไปซ้ำมาได้ (Automation) แต่ในบางครั้งงานที่มีความสลับซับซ้อนยังต้องการการควบคุมด้วยมนุษย์ (Human motion control) โดยอ่านท่าทางและการเคลื่อนไหวจากอุปกรณ์ที่มนุษย์สวมใส่ ซึ่งอุปกรณ์ที่นำมาติดไว้เพื่ออ่านท่าทางและการเคลื่อนไหวสามารถทำได้หลายวิธี เช่น การใช้โครงกระดูกภายนอก (Exoskeleton) สวมไว้ที่แขนของมนุษย์เพื่ออ่านค่าองศาการหมุนของข้อต่อจุดต่าง ๆ หรือการวิเคราะห์ท่าทางและการเคลื่อนไหวของมนุษย์ด้วยระบบกล้องจับความเคลื่อนไหว (Motion capture system) เป็นต้น ซึ่งวิธีการแต่ละวิธีมีข้อดีข้อเสียที่แตกต่างกัน เช่น วิธีการใช้ Exoskeleton นั้น จลนศาสตร์ของแขนกลและแขนคนต้องคล้ายกัน จึงยากที่จะนำไปประยุกต์ใช้ในแขนกลอุตสาหกรรมที่หลากหลายได้ ส่วน Motion capture system นั้นมีราคาที่สูงในการติดตั้ง การมองหาวิธีที่จะสามารถควบคุมแขนกลอุตสาหกรรมด้วยมนุษย์ที่มีราคาถูกและสามารถนำไปประยุกต์ได้หลากหลายแบบจึงมีคุณค่าในงานวิจัยอุตสาหกรรม

วิทยานิพนธ์นี้จึงต้องการพัฒนารูปแบบการควบคุมที่มีราคาถูก และประยุกต์ใช้กับหุ่นยนต์ที่มีจลนศาสตร์หลากหลายแบบได้และไม่จำเป็นต้องมีโมเดลของจลนศาสตร์ของหุ่นยนต์เหมือนแขนของมนุษย์ (Non-homogenous kinematic robot arm) โดยต้องการออกแบบและสร้างระบบถุงมือที่สามารถควบคุมแขนกลให้สามารถเคลื่อนที่ไปในตำแหน่ง (Position) เดียวกันและมีท่าทาง (Orientation) เดียวกันกับมือของมนุษย์ได้ ด้วยการคำนวณจลนศาสตร์ย้อนกลับ (Inverse

kinematic) และเซนเซอร์วัดการเคลื่อนไหว และระบบการหาตำแหน่งด้วยกล้องวัดความลึก ทำงานร่วมกันเพื่อทำการควบคุมหุ่นยนต์ได้ผ่านระบบไร้สายระยะไกล

## 1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อออกแบบ และสร้างระบบควบคุมแขนกลด้วยการรับตำแหน่งและท่าทางจากฝ่ามือของมนุษย์ โดยใช้เซ็นเซอร์วัดความเคลื่อนไหวและระบบระบุตำแหน่งด้วยกล้อง

1.2.2 เพื่อศึกษาการควบคุมแขนกลแบบ Non-homogenous kinematic

## 1.3 ขอบเขตของการวิจัย

1.3.1 อุปกรณ์ควบคุมเป็นอุปกรณ์สวมใส่ขนาดเล็กบริเวณมือ ร่วมกับเซนเซอร์วัดการเคลื่อนไหวและระบบกล้องวัดความลึก

1.3.2 หุ่นยนต์แขนกลเป็นแบบ 6 องศาอิสระ

1.3.3 ความแม่นยำของตำแหน่ง $\pm$ 3 เซนติเมตร

1.3.4 ความแม่นยำท่าทาง $\pm$ 5 องศา

1.3.5 สามารถควบคุมแขนกลได้จากระยะไกล

## 1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถใช้ถุงมือควบคุมแขนกลได้อย่างอิสระ

1.4.2 สามารถนำระบบถุงมือไปใช้ในการแขนควบคุมแขนกลได้หลายประเภท

## 1.5 แผนการดำเนินงาน

แผนการดำเนินงานของงานวิจัยนี้แสดงดังในตารางที่ 1.1

ตารางที่ 1.1 แผนการดำเนินงาน

| กิจกรรม/ขั้นตอนการดำเนินการ | เดือนที่ | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 ทบทวนวรรณกรรม บทความและศึกษาความรู้ที่เกี่ยวข้องกับงานวิจัย | | | | | | | | | | | | | | | | | | |
| 2 ออกแบบถุงมือควบคุมและแขนกลเพื่อใช้ในโปรแกรมจำลอง | | | | | | | | | | | | | | | | | | |
| 3 เขียนโปรแกรมคำนวณจลศาสตร์ย้อนกลับแบบ 6 องศาอิสระเพื่อหาตำแหน่งพร้อมทดสอบการทำงานเบื้องต้น | | | | | | | | | | | | | | | | | | |
| 4 เขียนโปรแกรมควบคุมท่าทางของแขนกลและทดสอบการทำงานเบื้องต้น | | | | | | | | | | | | | | | | | | |
| 5 ทดสอบการทำงานของแขนกลในโปรแกรมจำลองโดยรับข้อมูลจากท่าทางของมนุษย์ | | | | | | | | | | | | | | | | | | |
| 6 ผลิตถุงมือควบคุมและแขนกล และทดสอบการใช้งานเบื้องต้น | | | | | | | | | | | | | | | | | | |
| 7 ทดสอบและวิเคราะห์ผล | | | | | | | | | | | | | | | | | | |
| 8 เขียนวิทยานิพนธ์ | | | | | | | | | | | | | | | | | | |

## 1.6 การจัดรูปเล่มวิทยานิพนธ์

วิทยานิพนธ์ประกอบไปด้วย 5 บทซึ่งมีรายละเอียดดังนี้

บทที่ 1 เป็นบทนำซึ่งจะกล่าวถึงควาสำคัญและปัญหาเบื้องต้น และขอบเขตของงานวิจัย พร้อมทั้งประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง จะอธิบายถึงทฤษฎีที่เกี่ยวข้องกับการสร้างและการพัฒนาระบบถุงมือเพื่อใช้ในการควบคุมแขนกล การออกแบบแขนกล ครอบคลุมไปถึงบทความและงานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์นี้

บทที่ 3 วิธีดำเนินงานวิจัย จะอธิบายถึงการออกแบบและการสร้างระบบถุงมือและแขนกล

บทที่ 4 วิเคราะห์ผลการทดสอบการนำระบบถุงมือไปใช้ในการควบคุมแขนกล

บทที่ 5 สรุปผลและข้อเสนอแนะ

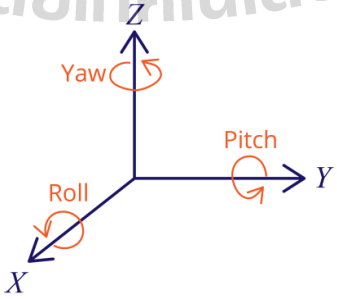
# บทที่ 2
# ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

## 2.1 Inertial Measurement Unit (IMU)

Inertial Measurement Unit หรือ IMU เป็นอุปกรณ์ที่ใช้ตรวจจับการเคลื่อนไหว (Movement) และทิศทาง (Orientation) มักถูกนำไปใช้กับระบบนำทางแบบเฉื่อย (Inertial Navigation System) เพื่อควบคุมอากาศยาน เรือ เรือดำน้ำ และอากาศยานไร้คนขับ (Unmanned Aerial Vehicles: UAVs) นอกจากนี้ IMU ยังถูกนำไปประยุกต์ใช้ในชีวิตประจำวันมากขึ้น เช่น ถูกนำมาติดตั้งคู่กับสมาร์ตโฟน smartwatch GPS และอื่น ๆ อีกมากมาย โดยอาศัยข้อมูลจากเซนเซอร์ 2 ตัวคือ ความเร่งในแนวแกน x, y, z ที่อ่านได้จาก Accelerometer และความเร็วเชิงมุมรอบแกน x, y, z ที่อ่านได้จาก Gyroscope

Accelerometer เป็นเซนเซอร์ที่ตรวจจับการเคลื่อนที่โดยตรวจจับวัตถุเมื่อมีการเคลื่อนที่จากจุดหนึ่งไปอีกจุดหนึ่งจากอัตราเร่งที่เกิดขึ้นขณะเคลื่อนที่ ซึ่งภายในเซนเซอจะใช้ผลึกขนาดเล็กในการวัดความเค้น (Stress) ที่เกิดจากการสั่นสะเทือนขณะเคลื่อนที่ หากมีการสั่นสะเทือนเกิดขึ้น ผลึกจะเข้าสู่สภาวะถูกเค้นและทำให้เกิดค่าแรงดันไฟฟ้า โดยค่าแรงดันไฟฟ้าที่เกิดขึ้นสามารถนำไปคำนวณเพื่อหาอัตราเร่งและวิเคราะห์อัตราการเคลื่อนที่ได้

Gyroscope เป็นเซนเซอร์ที่อาศัยแรงโน้มถ่วงของโลกในการหาทิศทาง โดยติดโรเตอร์ (Rotor) บนแกนหมุนตรงกลางวงล้อที่มีเสถียรภาพ เมื่อหมุนแกน โรเตอร์จะอยู่กับที่เพื่อระบุแรงดึงดูดจากศูนย์กลางซึ่งมีทิศทางที่ชี้ลงตามแรงดึงดูดของโลก โดย Gyroscope สามารถวัดอัตราเชิงมุมได้ 3 ประเภทขึ้นอยู่กับทิศทาง คือ ค่าที่วัดได้จากการหมุนรอบแกน X (Roll), ค่าที่วัดได้จากการหมุนรอบแกน Y (Pitch) และค่าที่วัดได้จากการหมุนรอบแกน Z (Yaw) ดังแสดงในรูปที่




รูปที่ 2.1 แกนตรวจจับข้อมูลของ Gyroscope

อุปกรณ์ IMU ที่ใช้ในงานวิจัยนี้คือ IMU WItMotion JY61P มีระบบวัดทิศทางการเคลื่อนไหวสามมิติที่มีประสิทธิภาพสูงโดยใช้เทคโนโลยี MEMS ซึ่งประกอบด้วยไจโรสโคปสามแกนและแอคเซลเลอโรมิเตอร์สามแกน นำเสนอความแม่นยำสูง มีการชดเชยทิศทางสามแกนแบบเรียลไทม์ มีการกรองข้อมูลด้วยตัวกรองคาลมานร่วมกับอัลกอริธึมการเฉพาะสามารถให้ข้อมูลด้วยอัตราการอัปเดตสูงถึง คุณสมบัติของ JY61P ดังกล่าวแสดงดังตารางต่อไปนี้

ตารางที่ 2.1 ข้อมูลของ JY61P Witmotion

| พารามิเตอร์ | เงื่อนไข | ค่าปกติ |
|---|---|---|
| Measuring range | | ±16g |
| Resolution ratio | ±16g | 0.0005(g/LSB) |
| RMS noise | Bandwidth=100Hz | 0.75~1mg-rms |
| Static zero drift | Place horizontally | ±20~40mg |
| Temperature drift | -40℃ ~ +85℃ | ±0.15mg/℃ |
| Bandwidth | | 5~256Hz |
| Measuring range | | ±2000°/s |
| Resolution ratio | ±2000°/s | 0.061(°/s)/(LSB) |
| RMS noise | Bandwidth=100Hz | 0.005(°/s) |
| Static zero drift | Place horizontally | ±0.5~1°/s |
| Temperature drift | -40℃ ~ +85℃ | ±0.005~0.015 (°/s)/℃ |
| Bandwidth | | 5~256Hz |
| Measuring range | | X:±180° |
| | | Y:±90° |
| Inclination accuracy | | 0.2° |
| Resolution ratio | Place horizontally | 0.0055° |
| Temperature drift | -40℃ ~ +85℃ | ±0.5~1° |
| Measuring range | | Z:±180° |
| Heading accuracy | 6-axis algorithm, static | 0.5° （Dynamic integral cumulative error exists） |
| | | |
| Resolution ratio | Place horizontally | 0.0055° |

## 2.2 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System) กับการใช้งานร่วมกับแขนกล

ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System : ROS) เป็นการรวมกันของชุดโปรแกรม เครื่องมือสำหรับการพัฒนาหุ่นยนต์ที่มีความนิยมและมีความสามารถที่สูงมากในปัจจุบัน เกิดขึ้นเมื่อปี 2550 ในบริษัท willow garage ประเทศสหรัฐอเมริกา แต่ต่อมาอยู่ถายใต้การดูแลของมูลนิธิหุ่นยนต์แบบเปิด (Open Source Robotic Foundation หรือ OSRF) จนถึงปัจจุบัน มีการพัฒนามาอย่างต่อเนื่องตั้งแต่ ROS1 และ ROS2 รวมทั้งยังมีชุดโปรแกรมฟรีเกิดขึ้นมากมาย ซึ่งหนึ่งในนั้นคือชุดโปรแกรมสำหรับการใช้ในหุ่นยนต์แขนกลอุตสาหกรรมโดยกลุ่ม ROS Industrial ได้แก่ชุดโปรแกรม MoveIt

ROS MoveIt เป็นชุดโปรแกรมที่ช่วยลดความยุ่งยากของการสั่งการแขนกลทุกรูปแบบโดยใช้ ROS เป็นเครื่องมือ สามารถใช้ในการสั่งการโดยใช้ Forward Kinematic, Inverse Kinematic หรือทำ Path Planning ได้โดยที่ผู้ใช้ไม่จำเป็นต้องไปเขียนโค้ดเพื่อแก้สมการต่าง ๆ เหล่านี้เลย ซึ่งเหมาะกับแขนกลอุตสาหกรรมทั่วไปอย่างมาก เพราะส่วนใหญ่แล้วจะมีจำนวนข้อต่อที่ซับซ้อนมากกว่า 5DOF เป็นต้นไป ทำให้เกิดความยากในการใช้งาน นอกจากนี้ ด้วยความที่ใช้ ROS เป็นเครื่องมือ ก็สามารถทำให้เราสามารถนำแขนกลไปต่อใช้งานร่วมกับเซนเซอร์อื่น ๆ ได้ง่าย เช่น กล้อง เป็นต้น ตัวอย่างหน้าการใช้งานชุดโปรแกรม MoveIt แสดงดังในรูปต่อไปนี้

รูปที่ 2.2 ตัวอย่างการใช้งานชุดโปรแกรม MoveIt

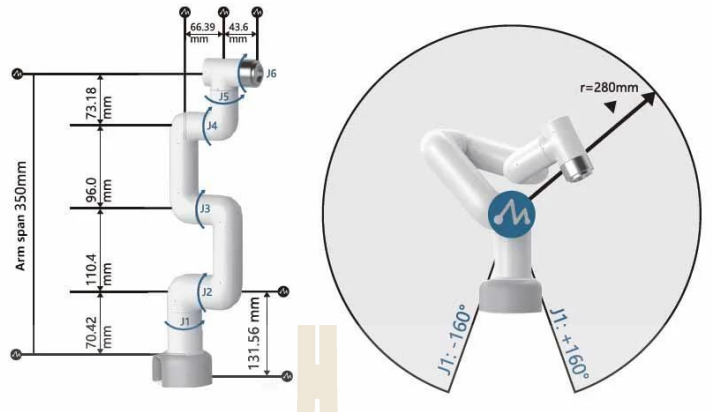
ROS MoveIt เป็นเครื่องมือที่นิยมเป็นอย่างมากในการเขียนโปรแกรมหุ่นยนต์ที่มีจลนศาสตร์ที่ซับซ้อน มีจำนวนองศาอิสระเยอะมากกว่า 6 และมีงานวิจัยที่นำไปทดสอบแล้วจริง สำหรับการทำ path planning , motion planning, obstacle avoidance, visual grasping ตัวอย่างเช่นในปี 2017 คุณ H. Sergio และคณะ (2017) ได้ทำการทดสอบออกแบบหุ่นยนต์เชื่อม ที่ใช้ stepper

motor ในการขับเคลื่อน โดยมีการเคลื่อนที่ 6 องศาอิสระ ซึ่งมีการใช้ ROS MoveIt เข้ามาช่วยในการเขียนโปรแกรมร่วมกับโปรแกรมจำลองและภาพจากกล้อง [21] และ M. Rajesh (2018) ได้นำ ROS Moveit เข้ามาควบคุมแขนกล 6 องศาอิสระด้วยเช่นกัน แต่จะเป็นการควบคุมโดยใช้แป้นพิมพ์ [8] หรือ D. Hao (2017) ได้ทำการออกแบบและสร้างหุ่นยนต์เคลื่อนที่ที่มีแขนกล 6 องศาอิสระติดอยู่กับหุ่นด้วยได้นำ MoveIt เข้ามาเป็นตัวทำ path planning ของแขนกลที่อยู่ด้านบน [8] ประสิทธิภาพของ ROS Moveit ไม่ได้อยู่เพียงงานวิจัยเท่านั้น ยังสามารถนำไปใช้ในภาพอุตสาหกรรมได้ด้วย ดังที่ H. Tianqi และคณะ (2021) และ L. Ren (2020) ที่นำ ROS Moveit ไปใช้กับหุ่นยนต์อุตสาหกรรมเช่น หุ่นยนต์ UR5 และ Autonomous Mobile Industrial Robot (AIMR) ตามลำดับ [25], [17]

## 2.3 แขนกล 6 องศาอิสระ MyCobot Pi 280

MyCobot Pi280 เป็นแขนกล 6 องศาอิสระในรูปแบบของ Cobot (Collaborative robot) ถูกพัฒนาโดย Elephant Robotics เพื่อใช้ในการศึกษาเรียนรู้การควบคุมแขนกลโดยมีระบบควบคุมเป็น raspberry pi ทำงานร่วมกับ Serial bus servo มี API ในการเชื่อมต่อ ROS และการเขียนโปรแกรมด้วยภาษา Python ทำให้เหมาะสมที่จะถูกนำมาใช้ในงานวิจัยนี้ โดยรูปร่างหน้าตาและขนาดต่าง ๆ ของ MyCobot 280Pi แสดงดังในรูปต่อไปนี้

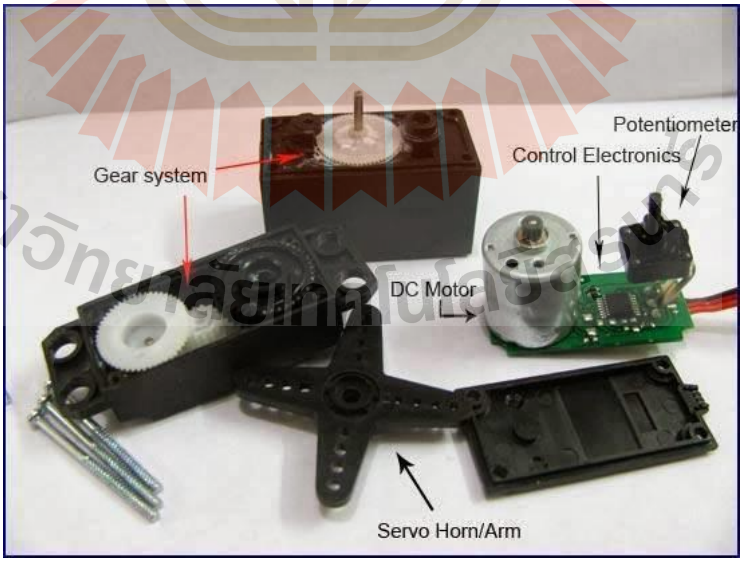

รูปที่ 2.3 หน้าตาของ myCobot280




รูปที่ 2.4 ระยะข้อต่อและพื้นที่การทำงาน

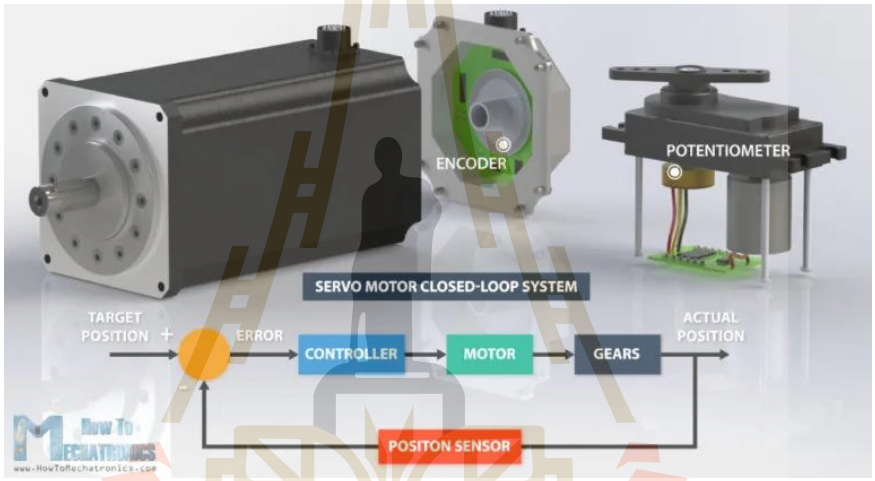
## 2.4 Servo Motor

เซอร์โวมอเตอร์ (Servo Motor) เป็นการรวมมอเตอร์ไฟฟ้ากระแส (DC Motor) ตรงเข้ากับวงจรควบคุม ซึ่งสามารถควบคุมให้เป็นไปตามที่ต้องการ เช่น ควบคุมความเร็วหรือเปลี่ยนแปลงองศา เซอร์โวขนาดเล็กมักถูกนำไปใช้ในควบคุม control surface ของเครื่องบินบังคับหรือเรือบังคับ, แขนกล หรือการทำผลิตภัณฑ์ต้นแบบ (Prototype) โดยเซอร์โวมอเตอร์ประกอบไปด้วยมอเตอร์กระแสตรง (DC Motor), ชุดเฟืองทดรอบ (Gear system), Potentiometer และวงจรควบคุม (Control electronics) ดังแสดงในรูปที่ 2.5




รูปที่ 2.5 ส่วนประกอบของเซอร์โวมอเตอร์

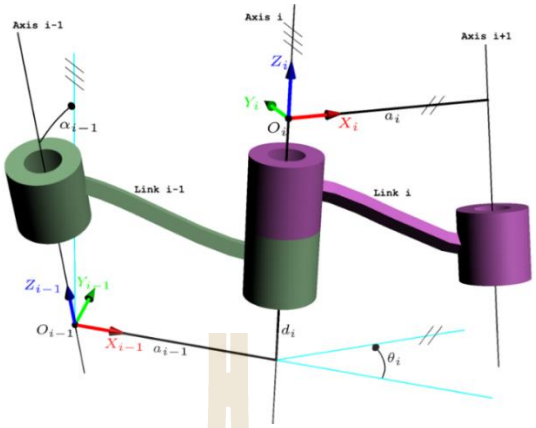
การทำงานของเซอร์โวมอเตอร์เป็นการใช้การควบคุมระบบปิด (closed-loop system) ด้วยการป้อนกลับข้อมูลในการควบคุมตำแหน่งและการเคลื่อนที่ เริ่มจากป้อนข้อมูล (Target position) ให้กับวงจรควบคุมเพื่อให้วงจรควบคุมการคำนวณทิศทางการหมุนไปที่องศาที่ต้องการ โดยมี Potentiometer ที่ติดอยู่กับชุดเฟืองทดรอบคอยตรวจสอบตำแหน่งของมอเตอร์ที่หมุนเพื่อเปรียบเทียบค่า ถ้าตรวจสอบแล้วพบว่าตำแหน่งเริ่มใกล้กับข้อมูลที่ป้อน วงจรควบคุมจะสั่งให้มอเตอร์หมุนช้าลงเพื่อให้ค่าใกล้เคียงกับข้อมูลที่ป้อนมากที่สุด เมื่อได้ตำแหน่งที่ต้องการก็จะทำการตรวจสอบตำแหน่งเป็นระยะ หากพบว่าตำแหน่งคลาดเคลื่อน วงจรควบคุมก็จะทำการสั่งให้มอเตอร์หมุนไปยังตำแหน่งที่ต้องการ โดยสามารถดูได้จากแผนภาพดังแสดงในรูปที่ 2.6




รูปที่ 2.6 Closed-loop system ของเซอร์โวมอเตอร์และอุปกรณ์ป้อนกลับข้อมูล

## 2.5 จลนศาสตร์ไปข้างหน้า (Forward Kinematics)

จลนศาสตร์ไปข้างหน้า (Forward Kinematic) สำหรับหุ่นยนต์แขนกล คือการแปลงการเคลื่อนที่ของมุมของข้อต่อต่าง ๆ ให้เป็นพิกัดที่จะเคลื่อนที่ไปของจุดที่สนใจ เช่น เมื่อมีการนำมุมของข้อต่อต่าง ๆ ไปคำนวณ จะได้ว่าปลายแขนของหุ่นยนต์จะไปอยู่ที่พิกัด x y z เท่าใด เป็นต้น ซึ่งวิธีการคำนวณสามารถใช้การคำนวณเชิงเรขาคณิต การคำนวณเชิงพีชคณิต หรือใช้วิธีเชิงตัวเลขก็ได้ แต่ที่นิยมสำหรับหุ่นยนต์ที่มีหลายแกน จะเป็นการคำนวณเชิงพีชคณิตร่วมกับวิธีการของ Denavit – Hartenburg (DH - method) โดยหลักการของวิธีการนี้จะสร้างตารางอธิบายการเชื่อมต่อของข้อต่อและ link ต่าง ๆ โดยใช้ตัวแปรเพียง 4 ตัว ซึ่งจะอธิบายการเลื่อนที่ (Translation) และการหมุน (Rotation) ระหว่าง joint ได้ โดยนิยามดังรูปที่ 2.7




รูปที่ 2.7 นิยามของตัวแปรในวิธีการ DH

Link length หรือ ความยาวของ link แทนด้วย $a$ จะเป็นระยะทางระหว่างแกนของกรอบอ้างอิงที่อยู่ติดกันตามแนวของแกน x ของเฟรมอ้างอิงก่อนหน้า

Link twist หรือ การบิด ของ link แทนด้วย $\alpha$ จะเป็นมุมที่แกน z ของกรอบอ้างอิงที่อยู่ติดกันกระทำต่อกัน โดยหมุนรอบแกน x ของเฟรมอ้างอิงก่อนหน้า

Link offset หรือ ระยะการแยกตัวของ link แทนด้วย $d$ เป็นระยะทางระหว่างจุดศูนย์กลางอ้างอิงของเฟรมทั้งสองเฟรมที่พิจารณาตามแนวแกนของเฟรมอ้างอิงที่สนใจปัจจุบัน

Joint angle หรือ มุมของข้อต่อ แทนด้วย $\theta$ เป็นมุมการหมุนของข้อต่อรอบแกน z ของเฟรมอ้างอิงที่สนใจ

เมื่อสามารถหาพารามิเตอร์เหล่านี้ได้จะทำให้เราสามารถเขียนเป็นตาราง DH parameters ได้ดังตารางที่ ซึ่งจะสามารถนำไปใส่ในเมทริกซ์การแปลง (Transformation Matrix) ความสัมพันธ์ระหว่างเฟรมได้ตามสมการดังต่อไปนี้ โดยเมทริกซ์ชุดแรกจะเป็นความสัมพันธ์ของการเลื่อนที่และการหมุนของ link หรือส่วนของตัวแปร $a$ และ $\alpha$

$$T_z(a_i)T_z(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

ส่วนเมทริกซ์ชุดที่สองจะเป็นความสัมพันธ์ของการเลื่อนที่และการหมุนของข้อต่อ (joint) ซึ่งได้เมทริกซ์ดังต่อไปนี้

$$T_z(d_i)T_z(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

ซึ่งเมื่อนำความสัมพันธ์ทั้งสองตัวมารวมกันโดยการคูณกันตามสมการดังต่อไปนี้

$$T_{i-1}^{i} = T_z(\theta_i)T_z(d_i)T_z(a_i)T_z(\alpha_i) \quad (2.3)$$

จะได้ผลลัพธ์เป็น

$$T_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

ซึ่งจะใช้เป็นเมทริกซ์ในการแสดงความสัมพันธ์ของเฟรมอ้างอิงที่อยู่ติดกัน ซึ่งในแขนกลนั้นจะมีหลายเฟรมอ้างอิง การจะหาความสัมพันธ์ของเฟรมที่ต่อ ๆ กันทั้งหมดสามารถทำได้โดยนำเมทริกซ์การแปลงของแต่ละเฟรมมารวมกันดังสมการต่อไปนี้

$$T_0^n = T_0^1 T_1^2 T_2^3 \dots T_{n-1}^n \quad (2.5)$$

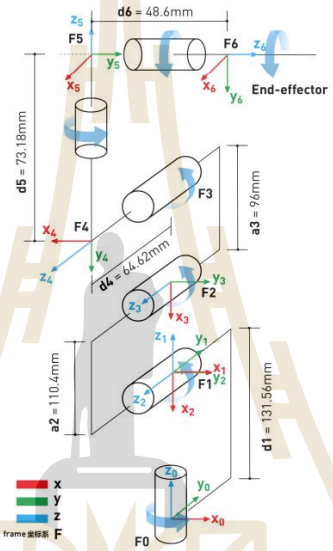
โดยผลลัพธ์สุดท้ายจะได้เป็น Transformation matrix ที่แสดงความสัมพันธ์ของการเลื่อนที่และการหมุนของเฟรมอ้างอิงเริ่มต้น ไปจนถึงเฟรมอ้างอิงสุดท้ายที่นำมาคูณกัน ดังในรูปแบบต่อไปนี้

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

โดยกรอบสีน้ำเงินจะแสดงในส่วนของการอธิบายการเลื่อนที่ และสีแดงจะเป็นส่วนของการอธิบายการหมุน เมื่อนำไปคำนวณก็จะสามารถหาผลลัพธ์ของการแปลงจลนศาสตร์ไปข้างหน้าของแขนกลนั้น ๆ ได้

## 2.6 จลนศาสตร์ผันกลับ (Inverse Kinematics)

จลนศาสตร์ผกผัน (Inverse Kinematic) เป็นวิธีการแปลงตำแหน่งของพิกัด X Y Z ที่ต้องการให้หุ่นเคลื่อนที่ไป เป็นค่าของการเคลื่อนที่ของข้อต่อต่าง ๆ สำหรับแขนกลประเภทหุ่นยนต์ ซึ่งประกอบด้วยหลายวิธีเช่น วิธีการเชิงตัวเลข วิธีการเชิงพีชคณิตเป็นต้น โดยสำหรับหุ่นยนต์แขนกล 6 องศาอิสระที่เลือกใช้ในงานวิจัยนี้อ้างอิงมาจากหุ่นยนต์แขนกลแบบ MyCobot280 Pi ซึ่งสามารถเขียนแผนผังสำหรับการคำนวณ Denavit -Hatenburg parameters (DH parameters) ได้ดังรูปที่ 2.8



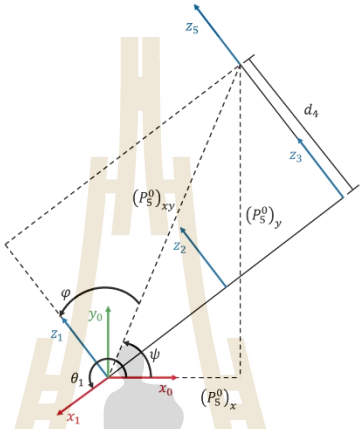

รูปที่ 2.8 Kinematic Diagram ของหุ่นยนต์ KUKA

โดยจากภาพข้างต้น ตารางแสดงค่าพารามิเตอร์ของ Denavit Hatenburg (DH - Parameter) แสดงได้ดังตารางที่

ตารางที่ 2.2 DH Parameter ของ Mycobot280 Pi

| $i$ | $\alpha_i - 1$ | $a_{i-1}$ | $d_i$ | $\theta_i$ | offset |
|---|---|---|---|---|---|
| 1 | $\pi/2$ | 0 | 131.56 | $\theta_1$ | 0 |
| 2 | 0 | -110.4 | 0 | $\theta_2$ | $-\pi/2$ |
| 3 | 0 | -96 | 0 | $\theta_3$ | 0 |
| 4 | $\pi/2$ | 0 | 64.62 | $\theta_4$ | $-\pi/2$ |
| 5 | $-\pi/2$ | 0 | 73.18 | $\theta_5$ | $\pi/2$ |
| 6 | 0 | 0 | 48.6 | $\theta_6$ | 0 |

หากใช้วิธีเชิงพีชคณิตจะสามารถหา Inverse Kinematic ของหุ่นยนต์แขนกลที่มี Kinematic Model แบบเดียวกับ My Cobot280 pi ได้ดังต่อไปนี้ โดยใช้สมการทางคณิตศาสตร์ในการหาข้อต่อที่มีการควบคุมตำแหน่งก่อนจากนั้นข้อต่อที่เหลือซึ่งกำหนดแนวการวางตัวของส่วนปลาย (Orientation) สามารถหมุนได้อย่างอิสระตามที่ต้องการ



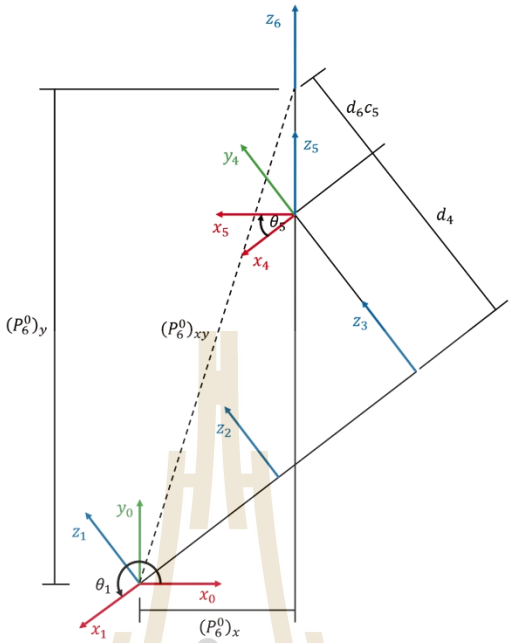

รูปที่ 2.9 ภาพมุมมองด้านบนของแกนที่ 5

$$\vec{P}_5^0 = T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.7}$$

$$\psi = \text{atan2}\left((P_5^0)_x, (P_5^0)_y\right) \tag{2.8}$$

$$\phi = \pm \arccos\left(\frac{d_4}{(P_5^0)_{xy}}\right) = \pm \arccos\left(\frac{d_4}{\sqrt{(P_5^0)_x{}^2 + (P_5^0)_y{}^2}}\right) \tag{2.9}$$

$$\theta_1 = \psi + \phi + \frac{\pi}{2} \tag{2.10}$$




รูปที่ 2.10 ภาพฉายระนาบของข้อต่อที่ 6 ไปหาฐาน

$$P_6^1 = ({}_6^0T)^{-1}P_6^0 \tag{2.11}$$

$$\theta_5 = \pm\cos^{-1}\left(\frac{(P_6^1)_z - d_4}{d_6}\right) \tag{2.12}$$

$$T_1^6 = ((T_1^0)^{-1}\,T_6^0)^{-1} \tag{2.13}$$

$$-\sin(\theta_6)\sin(\theta_5) = z_y \tag{2.14}$$

$$\cos(\theta_6)\sin(\theta_5) = z_x \tag{2.15}$$

$$\theta_6 = \operatorname{atan2}\left(\frac{-z_y}{\sin(\theta_5)}, \frac{z_x}{\sin(\theta_5)}\right) \tag{2.16}$$

$$T_4^1 = T_6^1 T_4^6 = T_6^1 \left(T_5^4 T_6^5\right)^{-1} \tag{2.17}$$

$$\vec{P}_3^1 = T_4^1 \begin{bmatrix} 0 \\ -d_4 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.18}$$

$$\theta_3 = \pm \arccos\left(\frac{\left\|\overrightarrow{P_3^1}\right\|^2 - a_2^2 - a_3^2}{2a_2 a_3}\right) \tag{2.19}$$

$$\theta_2 = -\operatorname{atan2}\left((P_3^1)_y, -(P_3^1)_x\right) + \arcsin\left(\frac{a_3 \sin(\theta_3)}{\|\vec{P}_3^1\|}\right) \tag{2.20}$$

$$T_4^3 = T_1^3 T_4^1 = (T_2^1 T_3^2)^{-1} T_4^1 \tag{2.21}$$

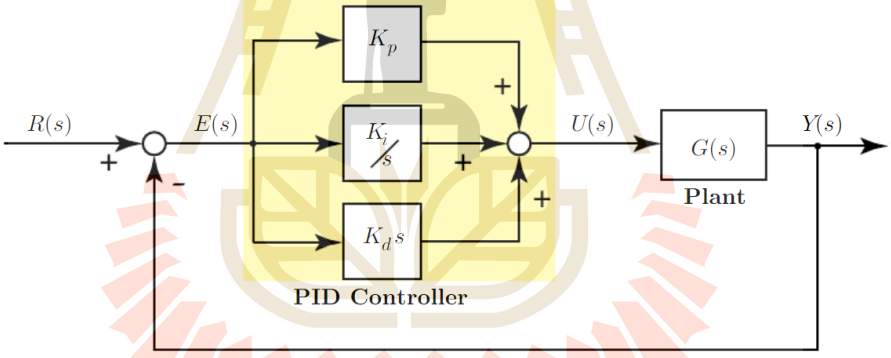
$$\theta_4 = \operatorname{atan2}\left(x_y, x_x\right) \tag{2.22}$$

เมื่อหาค่ามุมของข้อต่อ ที่ควบคุมตำแหน่ง X Y Z ได้แล้ว สามารถกำหนด orientation ของจุดปลายได้โดยตรง โดยกำหนดที่ข้อต่อที่ใช้ควบคุมแกนหมุนแต่ละแกนได้เลย ก็จะได้คำตอบของ inverse kinematic ออกมา จากนั้นก็สามารถนำไปควบคุมหุ่นยนต์ได้ นอกจากวิธีการคำนวณโดยวิธีพีชคณิตแล้วก็สามารถคำนวณจากการคิดย้อนกลับของ transformation matrix ที่ได้จากการคำนวณจลนศาสตร์ไปข้างหน้าดังที่มีการเสนอโดย D. Mahidzal (2011) ซึ่งทำการวิเคราะห์จลนศาสตร์ทั้งสองแบบของหุ่นยนต์ KR-16KS [10]

## 2.7 ตัวควบคุม PID

ตัวควบคุมพีไอดี (PID Controller) เป็นเทคนิคที่ใช้ในการควบคุมระบบการเคลื่อนที่ ระบบทางไฟฟ้า หรือระบบอื่น ๆ ซึ่งเป็นอัลกอริทึมที่เรียบง่ายและนิยมกันอย่างแพร่หลาย หน้าที่ของตัวควบคุมนั้นจะช่วยควบคุมให้ผลลัพธ์ที่ได้ออกมาจากระบบมีค่าเท่ากับหรือใกล้เคียงกับ input ที่ใส่เข้าไป เช่น การนำตัวควบคุมพีไอดีไปควบคุมมอเตอร์เพื่อให้ความเร็วการหมุนตรงกับความเร็วที่ต้องการ ตัวควบคุมพีไอดีมีส่วนของตัวควบคุมอยู่สามส่วนได้แก่ P หรือ Proportional control , I หรือ Integral control และ D หรือ Derivative control ซึ่งพารามิเตอร์ของแต่ละส่วนก็จะส่งผลต่อการควบคุมของระบบ

สำหรับงานวิจัยนี้ การเคลื่อนที่ขึ้นลงของมอเตอร์สเต็ปควรจะต้องมีตัวควบคุมตำแหน่งและความเร็วการขึ้นลงในแต่ละช่วง เพื่อลดเวลาในการเคลื่อนที่ลง เพราะหากใช้ความเร็วคงที่ทุกครั้งจะทำให้การวิ่งขึ้นลงของแกนดิ่งไม่เหมาะสมต่อการใช้งานจริงและเสียเวลาไปบางส่วน ซึ่งตัวควบคุมพีไอดีสามารถนำมาประยุกต์ได้




รูปที่ 2.11 ตัวควบคุมพีไอดี

การกำหนดปัญหาสำหรับตัวควบคุมพีไอดีแสดงดังในรูปด้านบน โดยระบบหรือแพลนท์ (plant) จะถูกแทนด้วยตัวแปร $G(s)$ ส่วนเอาท์พุทจากระบบแทนด้วย $Y(s)$ ส่วนที่เป็นอินพุทแทนได้ด้วย $\mathrm{R(s)}$ เมื่อระบบเกิดการทำงานจะมีความต่างระหว่างอินพุทกับเอาท์พุทเกิดขึ้น หรือ เรียกว่า error แทนด้วย $E(s)$ ซึ่งตัวค่าคลาดเคลื่อนนี้จะถูกนำเข้าไปในตัวควบคุม PID เพื่อสร้างสัญญาณ $U(s)$ ขึ้นมาแล้วส่งเข้าแพลนท์อีกทีเพื่อให้แพลนท์มีการปรับตัวสำหรับลดความต่างระหว่างอินพุทกับเอาท์พุทให้เหลือน้อยที่สุด โดยตัวควบคุมแต่ละเทอมประกอบด้วย เทอม P จะมีค่าตัวคูณ (gain) แทนด้วย $K_p$ ส่วนเทอม I มีตัวคูณ $K_i$ และเทอม d มีตัวคูณ $K_d$ ซึ่งตัวคูณแต่ละตัวจะถูกนำไปกระทำกับค่า error ที่เกิดขึ้น โดยในเทอมของ P ตัวคูณจะถูกนำไปคูณกับ error ที่เกิดขึ้นปัจจุบัน

(the present value of the error) ยิ่ง error เยอะ สัญญาณที่ส่งออกไปจากตัวควบคุมก็จะมีค่าสูง การปรับจูนเทอมของ P จะช่วยเพิ่มหรือลดความเร็วในการตอบสนองของระบบ แต่หากปรับสูงเกินไป อาจจะเกิดสัญญาณส่วนเกิน (Overshoot) ออกมา หรือหากน้อยเกินไป ระบบก็จะตอบสนองช้ากว่าที่ต้องการ แต่หากปรับความเร็วการตอบสนองได้ตามต้องการแล้วแต่เอาต์พุตของระบบยังมีค่าไม่เท่ากับค่าอินพุตแม้ว่าระบบจะเข้าสู่สภาวะคงที่ไปแล้ว (steady state) ความแตกต่างที่เกิดขึ้นจะถูกเรียกว่า ความแตกต่างในสภาวะคงที่ (steady state error) ซึ่งเกินความสามารถของตัวควบคุมพีอย่างเดียวไปแล้ว ต้องมีการใส่ตัวควบคุมไอเข้ามาเพิ่ม โดยในตัวควบคุม I นั้น ค่าของตัวคูณจะถูกนำไปคูณกับความแตกต่างสะสม (accumulated error) ในช่วงเวลาทั้งหมด ซึ่งสามารถหาได้จากการอินทิเกรตระหว่าง error ที่เกิดขึ้นกับเวลาที่มีการเปลี่ยนแปลงไป ดังสมการต่อไปนี้

$$I\ term\ signal = K_i \int_0^t e(\tau)d\tau \tag{2.23}$$

โดยสัญญาณที่ได้จากตัวควบคุมส่วนนี้จะถูกนำไปเพิ่มให้กับตัวควบคุมแบบสัดส่วนเพิ่มสัญญาณที่ยังไม่พอสำหรับทำให้ระบบเข้าสู่ setpoint ที่ต้องการได้ แต่การใส่เทอม I เข้าไปอาจมีผลทำให้ความเร็วในการตอบสนองช้าลง ดังนั้นอาจจะต้องตัวควบคุมในส่วนเทอมของ D เข้าไป โดยในส่วนของตัวควบคุม D ค่าของตัวคูณจะถูกนำไปคูณเข้ากับอนุพันธ์ของ error เทียบกับช่วงเวลา ซึ่งสัญญาณที่ได้จากเทอม D สามารถแสดงได้ดังสมการต่อไปนี้

$$D\ term\ signal = K_d \frac{de(t)}{dt} \tag{2.24}$$

จากสมการข้างต้นจะพบว่าเมื่อมีการเปลี่ยนแปลงของ error จะทำให้สัญญาณควบคุมจากเทอมของ D มีค่าที่เพิ่มขึ้น ซึ่งการเปลี่ยนแปลงของ error ก็จะมีทิศทางการเปลี่ยนแปลงอยู่ด้วย ยิ่ง error เปลี่ยนแปลงเร็วมาก สัญญาณที่ได้จากเทอมนี้จะยิ่งเยอะ เมื่อนำสัญญาณจากเทอมการควบคุมทั้งสามมารวมกันก็จะได้สัญญาณออกมาสำหรับไปสั่งการระบบให้เป็นไปตามต้องการ

## 2.8 IMU Filter Madgwick

ตัวกรอง IMU ของ Madgwick หรือ IMU Filter Madgwick ถูกเสนอโดย Sebastian O G Madgwick ในปี 2011 [20] เป็นตัวกรองที่ใช้สำหรับหาท่าทาง (orientation) ของเซนเซอร์ IMU จากข้อมูลที่ได้มาจาก accelerometers, gyroscopes หรือนำข้อมูลจาก magnetometers มารวมด้วยก็ได้เช่นกัน ตัวกรองนี้ถูกนำเสนอโดย Madgwick Sebastian จุดเด่นของอัลกอริทึมนี้ประกอบด้วย

1) การเปลี่ยนแปลงของพารามิเตอร์เดี่ยวจะถูกกำหนดโดยคุณลักษณะของระบบที่สามารถสำรวจได้
2) ทำงานได้ดีแม้มีอัตราการสุ่มตัวอย่างที่ต่ำ (Low sampling rates)
3) สามารถใช้ข้อมูลสนามแม่เหล็กในขณะนั้นมาใช้แล้วชดเชยการ distort ได้เลยทันทีทันใด
4) สามารถชดเชย bias ของ gyroscope ได้เลย

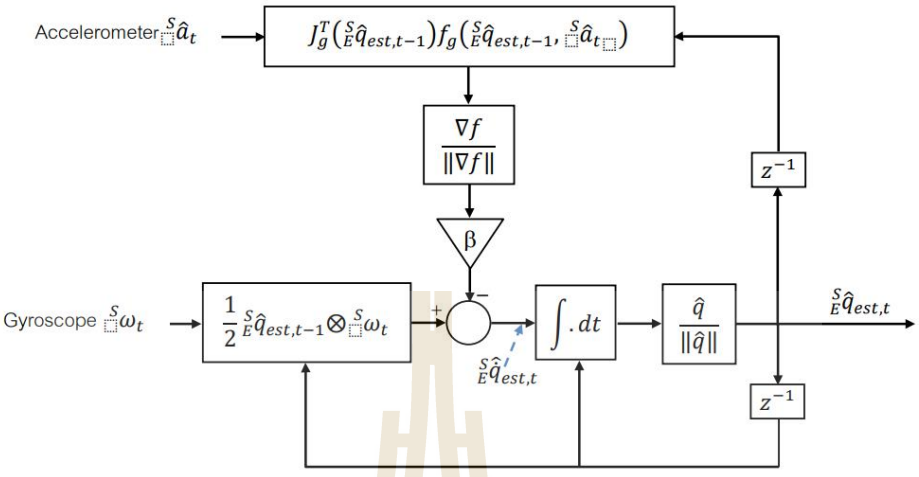
การหา orientation ระหว่างเฟรมอ้างอิงของโลกกับเซนเซอร์ด้วยอัตราความเร็วเชิงมุมจาก gyroscope สามารถทำได้โดยการอินทิเกรตอนุพันธ์ของควอร์เทอรเนียน (quaternion : เป็นระบบอ้างอิงแบบหนึ่งใช้แสดงการหมุนของวัตถุ) ดังสมการต่อไปนี้

$$q_{\omega,t} = q_{t-1} + \dot{q}_{\omega,t}\Delta t = q_{t-1} + \frac{1}{2}(q_{t-1}\omega_t^s)\Delta t \tag{2.25}$$

เมื่อ $\Delta t$ คือ เวลาที่เปลี่ยนไป และ $\omega_t^s$ คือเมทริกซ์ของความเร็วเชิงมุมในแต่ละแกน นอกจากการหา orientation ด้วยอัตราเชิงมุมแล้ว ยังมีการหา orientation ด้วยการใช้ algorithm gradient descent ด้วยโดยมีฟังก์ชันวัตถุประสงค์ (objective function) ดังต่อไปนี้

$$\begin{aligned} f(q, d^E, s^s) &= q^* d^E q - s^s \\ &= \begin{bmatrix} 2d_x\left(\frac{1}{2} - q_y^2 q_z^2\right) + 2d_y(q_w q_z + q_x q_y) + 2d_z(q_x q_z - q_w q_y) - s_x \\ 2d_x(q_x q_y - q_w q_z) + 2d_y\left(\frac{1}{2} - q_x^2 q_z^2\right) + 2d_z(q_w q_x + q_y q_z) - s_y \\ 2d_x(q_w q_y + q_x q_z) + 2d_y(q_y q_z - q_w q_x) + 2d_z\left(\frac{1}{2} - q_x^2 q_y^2\right) - s_z \end{bmatrix} \end{aligned} \tag{2.26}$$

เมื่อ $d^E$ คือ ท่าทางของเฟรมอ้างอิงของเซ็นเซอร์ที่ถูกกำหนดเอาไว้ก่อนเทียบกับเฟรมอ้างอิงของโลก และ $s^s$ คือ ทิศทางที่วัดได้ในเฟรมอ้างอิงของตัวเซนเซอร์เอง ส่วน $q^*$ คือ คอนจูเกตของควอเทอร์เนียน ซึ่งหลังจากการ optimization แล้วนั้น จะได้ค่าควอเทอร์เนียนออกมา ซึ่งมีความเป็นไปได้มากที่สุดว่าจะเป็นท่าทางที่ใกล้เคียงที่สุดของตัว IMU จริง ไดอะแกรมโครงสร้างของ IMU filter madgwick สามารถแสดงดังในรูปที่




รูปที่ 2.12 block diagram โครงสร้างของ IMU Filter madgwick

Madgwick filter ประสบความสำเร็จในการใช้ประมาณท่าทางการเคลื่อนที่ในงานต่าง ๆ เชิงพาณิชย์ในอุปกรณ์ตรวจจับความเคลื่อนไหวต่าง ๆ ที่ต้องใช้พลังงานต่ำ โดย S. Omid (2021) ได้นำไปเปรียบเทียบกับวิธีการที่ใช้ Kalman filter เป็นฐานพบว่าให้ความแม่นยำที่เพียงพอกันในขณะที่ใช้ทรัพยากรในการคำนวณน้อยกว่า จากนั้นคุณ S. Omid ได้ทำการปรับปรุงอัลกอริทึมโดยใส่การหาค่าที่เหมาะสมที่สุด (optmization) ด้วย Gradient Descent ในช่วงการหาความถูกต้องของการทำนายเข้าไปด้วย พบว่า สามารถปรับปรุง dynamic RMS error ได้ลงไป 37.2% [15] G. Mattia และคณะ (2021) และ L. Xin ได้นำ IMU filter madgwick ไปใช้สำหรับติดตามการเคลื่อนที่ของมนุษย์ เช่น การเดิน และขยับตัว [12], [26] นอกจากนี้ ในงานด้านหุ่นยนต์ที่มีข้อจำกัดในเรื่องของทรัพยากรการคำนวณก็สามารถนำ filter madgwick ไปหาท่าทางของหุ่นยนต์ได้เช่นกัน อย่างเช่น Z. Jasper (2021) ที่นำไปใช้กับหุ่นยนต์วงกลม *(spherical robot) (*IMU-based pose-estimation for spherical robots with limited resources*)* หรือ L. Simon (2018) ได้นำ madgwick filter ไปหาท่าทางของอากาศยานปีกหมุนสี่ใบพัดเทียบกับ Extended Kalman Filter(EKF) และ Mahony filter พบว่า madgwick filter ใช้เวลาในการคำนวณน้อยกว่า EKF แต่ใช้เวลามากกว่า Mahony filter แต่ได้ RMS error ใกล้เคียงค่าที่ได้จาก Extended Kalman Filter [24]

## 2.9 ตัวกรองคาลมาน (Kalman filter)

ตัวกรองคาลมาน (Kalman Filter) เป็นอัลกอริธึมทางคณิตศาสตร์ที่ใช้สำหรับการประมาณค่าของตัวแปรในระบบที่เปลี่ยนแปลงไปตามเวลา โดยใช้ข้อมูลที่ได้จากการวัดค่าที่มีความไม่แน่นอน ตัวกรองคาลมานถูกนำมาใช้ในหลาย ๆ ด้าน เช่น ระบบนำทาง, การติดตามวัตถุ, และการควบคุมหุ่นยนต์ เป็นต้น

หลักการทำงานของตัวกรองคาลมานประกอบด้วยสองขั้นตอนหลัก:

1) **ขั้นตอนการทำนาย (Prediction Step)**:
   1.1) ทำการทำนายค่าของสถานะ (state) ในช่วงเวลาถัดไป โดยใช้แบบจำลองของระบบ
   1.2) ทำนายค่าความไม่แน่นอนของสถานะ ซึ่งจะคำนวณจากค่าความไม่แน่นอนของสถานะในช่วงเวลาก่อนหน้าและค่าความไม่แน่นอนของกระบวนการ (process noise)
2) **ขั้นตอนการอัพเดต (Update Step)**:
   2.1) ทำการวัดค่า (measurement) ของสถานะในช่วงเวลาปัจจุบัน
   2.2) คำนวณค่าคาดเคลื่อน (Kalman gain) ซึ่งจะใช้ในการปรับปรุงค่าทำนายของสถานะ
   2.3) อัพเดต ค่าของสถานะและความไม่แน่นอนของสถานะโดยใช้ค่าคาดเคลื่อน

ด้วยการใช้ขั้นตอนการทำนายและการอัพเดตในทุกช่วงเวลา ตัวกรองคาลมานสามารถประมาณค่าของสถานะที่ดีที่สุดในระบบที่มีความไม่แน่นอนได้อย่างมีประสิทธิภาพ โดยรูปของสมการที่ใช้ในแต่ละขั้นตอนของตัวกรองคาลมานจะถูกกล่าวถึงในบทที่ 3 ในหัวข้อการออกแบบระบบติดตามการเคลื่อนที่ของมือ

## 2.10 กล้องวัดความลึก

กล้องวัดความลึก (Depth Camera) เป็นอุปกรณ์ที่สำคัญในการสร้างภาพสามมิติหรือแผนที่ความลึกของพื้นที่ โดยเทคโนโลยีที่ใช้ในการวัดระยะทางนั้นมีหลายประเภท เช่น เทคโนโลยี Time-of-Flight (ToF), Structured Light, Stereo Vision และ LiDAR แต่ละเทคโนโลยีมีหลักการทำงานที่แตกต่างกัน เพื่อวัดระยะทางจากกล้องไปยังวัตถุในภาพได้อย่างแม่นยำและมีประสิทธิภาพ เทคโนโลยี Time-of-Flight (ToF) ใช้การวัดเวลาที่แสงเดินทางจากกล้องไปยังวัตถุและกลับมา เพื่อคำนวณระยะทางโดยใช้ความเร็วของแสง ส่วนเทคโนโลยี Structured Light ใช้การปล่อยลำแสงรูปแบบเฉพาะไปยังวัตถุ และตรวจจับการบิดเบือนของลำแสงเพื่อคำนวณระยะทาง ขณะที่เทคโนโลยี

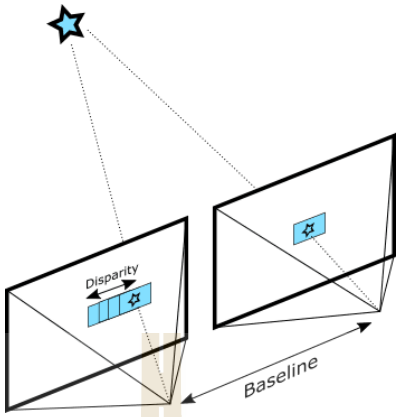
Stereo Vision ใช้กล้องสองตัวในการถ่ายภาพจากมุมมองที่แตกต่างกัน และคำนวณระยะทางโดยใช้หลักการพารัลแลกซ์ สุดท้าย คือ เทคโนโลยี LiDAR ที่ใช้การปล่อยแสงเลเซอร์และวัดเวลาที่แสงสะท้อนกลับมา เพื่อวัดระยะทางได้อย่างแม่นยำ

การนำข้อมูลความลึกที่ได้จากกล้องวัดความลึกไปใช้นั้นมีหลากหลายแอปพลิเคชัน เช่น การนำทางและการควบคุมการเคลื่อนที่ของหุ่นยนต์ การประมวลผลภาพเพื่อการจดจำและติดตามวัตถุ การวัดขนาดและรูปร่างของวัตถุ และการสร้างโมเดลสามมิติ ข้อมูลความลึกที่แม่นยำสามารถช่วยเพิ่มประสิทธิภาพในการทำงานและเพิ่มความสามารถในการตรวจจับและประมวลผลข้อมูลในหลากหลายด้าน ตัวอย่างเช่น ในการควบคุมการเคลื่อนที่ของหุ่นยนต์ ข้อมูลความลึกช่วยให้หุ่นยนต์สามารถหลีกเลี่ยงสิ่งกีดขวางและเคลื่อนที่ไปยังเป้าหมายได้อย่างปลอดภัยและมีประสิทธิภาพ นอกจากนี้ ข้อมูลความลึกยังมีประโยชน์ในการสร้างภาพสามมิติและแผนที่ความลึกของพื้นที่ ซึ่งสามารถนำไปใช้ในการสำรวจและวิเคราะห์พื้นที่ในงานวิจัยและอุตสาหกรรมต่าง ๆ

ด้วยความสามารถในการวัดระยะทางและสร้างภาพสามมิติที่แม่นยำ กล้องวัดความลึกได้กลายเป็นเครื่องมือสำคัญในหลายสาขา เช่น หุ่นยนต์, ยานยนต์อัตโนมัติ, การสร้างภาพสามมิติ และการทำแผนที่สามมิติ เทคโนโลยีที่ใช้ในกล้องวัดความลึกยังคงพัฒนาอย่างต่อเนื่อง เพื่อให้ได้ข้อมูลที่แม่นยำและประสิทธิภาพสูงขึ้น นอกจากนี้ ความก้าวหน้าทางเทคโนโลยียังทำให้กล้องวัดความลึกสามารถใช้งานได้ในหลากหลายสถานการณ์และสภาพแวดล้อม ทำให้สามารถนำไปประยุกต์ใช้ในงานที่ต้องการการวัดระยะทางและการสร้างภาพสามมิติได้มากขึ้น ทั้งในด้านอุตสาหกรรม การแพทย์ การสำรวจ และงานวิจัยต่าง ๆ โดยกล้องที่ใช้ในงานวิจัยนี้เป็นกล้องแบบ Stereoscopic vision ซึ่งการคำนวณความลึกจากการใช้กล้องสเตอริโอ (Depth from Stereo) เป็นอัลกอริธึมด้านการมองเห็นทางคอมพิวเตอร์แบบคลาสสิกที่ได้รับแรงบันดาลใจจากระบบการมองเห็นแบบสองตาของมนุษย์ อัลกอริธึมนี้พึ่งพาการมีพอร์ตมองเห็นสองพอร์ตที่ขนานกัน และคำนวณความลึกโดยการประมาณค่าความแตกต่างระหว่างจุดสำคัญที่ตรงกันในภาพจากกล้องซ้ายและกล้องขวา ระบบจะตรวจจับจุดที่ตรงกันในภาพจากกล้องทั้งสอง และใช้สมการทางเรขาคณิตในการคำนวณระยะทางไปยังวัตถุ การคำนวณระยะทางทำได้โดยการใช้สูตร:

$$Depth = \frac{B \times f}{d} \tag{2.27}$$

โดยที่ $B$ คือ ระยะห่างระหว่างเลนส์กล้องสองตัวบน depth camera, $f$ คือ ความยาวโฟกัสของเลนส์กล้อง, และ $d$ คือ ความแตกต่างของตำแหน่งจุดเดียวกันในภาพสองภาพ (disparity) การคำนวณนี้ช่วยให้ระบบสามารถสร้างแผนที่ความลึกที่แม่นยำของพื้นที่และวัตถุในภาพได้




รูปที่ 2.13 Baseline และ disparity

## 2.11 Hive MQ Broker

HiveMQ เป็นแพลตฟอร์มซอฟต์แวร์สำหรับการสื่อสารแบบ machine-to-machine (M2M) และ Internet of Things (IoT) โดยใช้โปรโตคอล MQTT (Message Queuing Telemetry Transport) ซึ่งเป็นโปรโตคอลมาตรฐานที่ออกแบบมาเพื่อการส่งข้อมูลที่มีน้ำหนักเบาและมีประสิทธิภาพสูง HiveMQ ถูกพัฒนาขึ้นโดยบริษัท dc-square GmbH และมีจุดเด่นในเรื่องการจัดการการเชื่อมต่อจำนวนมาก การส่งข้อมูลแบบ real-time และการรับรองความปลอดภัยของข้อมูล

**คุณสมบัติหลักของ HiveMQ:**

1) รองรับการเชื่อมต่อจำนวนมาก:

HiveMQ สามารถรองรับการเชื่อมต่อจากอุปกรณ์ IoT หลายล้านอุปกรณ์พร้อมกัน ด้วยการใช้โครงสร้างพื้นฐานที่มีประสิทธิภาพและยืดหยุ่น ทำให้สามารถจัดการกับการรับส่งข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ

2) การส่งข้อมูลแบบ Real-time:

ด้วยโปรโตคอล MQTT ซึ่งเป็นโปรโตคอลที่มีน้ำหนักเบา HiveMQ สามารถส่งข้อมูลแบบ real-time ด้วยความหน่วงต่ำ (low latency) ทำให้อุปกรณ์ IoT สามารถแลกเปลี่ยนข้อมูลกันได้อย่างรวดเร็วและมีประสิทธิภาพ

3) ความปลอดภัย:

HiveMQ มีฟีเจอร์ด้านความปลอดภัยที่ครอบคลุม เช่น การเข้ารหัสข้อมูลด้วย SSL/TLS การตรวจสอบสิทธิ์ของอุปกรณ์ (authentication) และการควบคุมการเข้าถึง (access control) ทำให้สามารถรับรองความปลอดภัยของข้อมูลที่ส่งผ่านระบบได้

4) การขยายตัวและการปรับขนาด:

HiveMQ ออกแบบมาให้สามารถขยายตัวได้ง่าย สามารถเพิ่มหรือลดจำนวนโหนด (nodes) ในคลัสเตอร์ตามความต้องการได้ ทำให้สามารถปรับขนาดการใช้งานได้ตามปริมาณงานและจำนวนอุปกรณ์ที่เชื่อมต่อ

5) การบูรณาการกับระบบอื่น:

HiveMQ รองรับการบูรณาการกับแพลตฟอร์มและระบบอื่น ๆ เช่น ระบบคลาวด์, บริการข้อมูล และแพลตฟอร์ม IoT อื่น ๆ ทำให้ง่ายต่อการนำไปใช้งานในระบบที่มีอยู่เดิม

HiveMQ เหมาะสำหรับการใช้งานในหลาย ๆ ด้านที่ต้องการการสื่อสารระหว่างอุปกรณ์ IoT เช่น การเกษตรอัจฉริยะ (smart agriculture), เมืองอัจฉริยะ (smart cities), ระบบบ้านอัจฉริยะ (smart homes), การควบคุมอุตสาหกรรม (industrial control), และการดูแลสุขภาพ (healthcare) ด้วยความสามารถในการส่งข้อมูลแบบ real-time และการจัดการการเชื่อมต่อจำนวนมาก HiveMQ จึงเป็นเครื่องมือที่มีประสิทธิภาพสำหรับการพัฒนาและจัดการระบบ IoT ที่ซับซ้อน

# บทที่ 3
# วิธีการดำเนินการ

## 3.1 ขั้นตอนการดำเนินงานวิจัย

สำหรับงานวิจัยนี้มีขั้นตอนหลักในการวิจัยและพัฒนาระบบให้สามารถใช้งานได้ดังต่อไปนี้


รูปที่ 3.1 ขั้นตอนการดำเนินงานวิจัย

## 3.2 การออกแบบระบบควบคุมสั่งการ

ระบบควบคุมสั่งการจะแบ่งออกเป็น 3 ส่วน ได้แก่ ส่วนของการรับข้อมูล ส่วนของการผสานข้อมูล และส่วนของการส่งต่อข้อมูล โดยหน้าที่และหลักการสำคัญของส่วนประกอบแต่ละส่วนมีดังต่อไปนี้

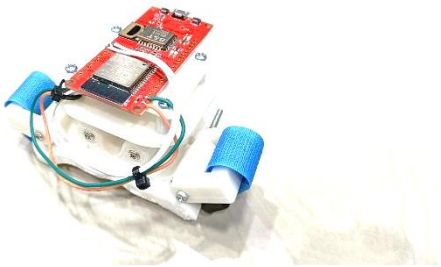
1) ส่วนของการรับข้อมูล จะเป็นส่วนที่ทำหน้าที่รับค่าจากเซนเซอร์สองตัว ได้แก่ IMU และกล้องสามมิติเข้ามา โดยกล้อง 3 มิติจะใช้อ่านค่าตำแหน่งในแนวแกน X Y และ Z พร้อมทั้งอ่านค่ามุม yaw ของการหมุนของฝ่ามือ ส่วน IMU จะใช้วัดค่าคววามเอียงของฝ่ามือในแนวแกน roll และ pitch แล้วนำข้อมูลมารวมกันเพื่อนำไปผ่านตัวกรอง (filter)

2) ส่วนของการผสานข้อมูล เป็นส่วนที่ทำการประมวลผลข้อมูลร่วมกันระหว่างข้อมูลจาก IMU และข้อมูลจาก depth camera โดยจะมีตัวประมวลผลตัวกรองคาลมาน (Kalman filter) กับ model machine learning ชื่อ MediaPipe อยู่ภายใน หลังจากประมวลผลข้อมูลได้แล้วจะส่งไปยังหน่วยถัดไป

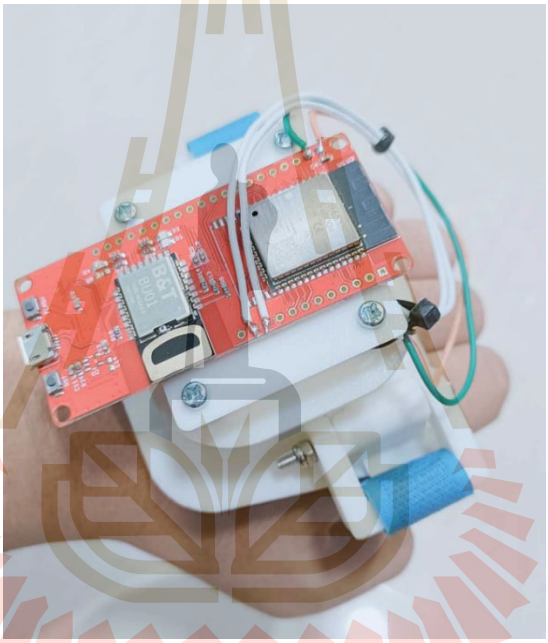
3) ส่วนของการส่งต่อข้อมูล ทำหน้าที่สองอย่าง ได้แก่ เป็นส่วนที่นำข้อมูลที่ประมวลผลได้ส่งต่อไปยังระบบ MQTT cloud broker โดย broker ในการส่งข้อมูลที่เลือกใช้คือ HiveMQ ส่วนหน้าที่ที่สองคือการรับข้อมูลจาก HiveMQ ไปส่งให้กับแขนกลเพื่อสั่งการการเคลื่อนที่ โดยในส่วนการควบคุมแขนกลนี้จะอยู่ในคอมพิวเตอร์ที่สามารถเชื่อมต่อกับแขนกล MyCobot 280 ได้

## 3.3 การออกแบบและสร้างอุปกรณ์สวมใส่ที่แขน

การออกแบบอุปกรณ์สวมใส่สำหรับใส่ที่มือของผู้ใช้งาน จะต้องคำนึงถึงความง่ายในการสวมใส่และต้องสามารถบรรจุอุปกรณ์ไมโครคอนโทรลเลอร์และเซนเซอร์ IMU ที่ใช้ได้ โดยการขึ้นรูปของชิ้นงานจะใช้เครื่องพิมพ์สามมิติเป็นหลัก โดยจะออกแบบให้เข้ากับรูปทรงของมือ และให้บังบริเวณด้านล่างของฝ่ามือน้อยที่สุด (ต้องใช้ในการตรวจจับด้วยกล้อง) โดยการออกแบบใช้โปรแกรม Solidworks ในการออกแบบ โดยในการออกแบบอุปกรณ์สวมใส่จะจัดให้ไมโครคอนโทรลเลอร์ ESP32 อยู่ด้านบน ส่วนตรงกลางระหว่างด้านหลังฝ่ามือและด้านบนจะเป็น IMU sensor ที่มีแนวแกน X ชี้ไปทางด้านเดียวกับนิ้ว และแกน Y ของ IMU ชี้ไปทางด้านซ้ายเมื่อเทียบกับแกน X โดยผลของการออกแบบและประกอบอุปกรณ์สวมใส่แสดงดังในรูปต่อไปนี้



รูปที่ 3.2 อุปกรณ์สวมใส่สำหรับวัดค่าด้วยเซนเซอร์ IMU



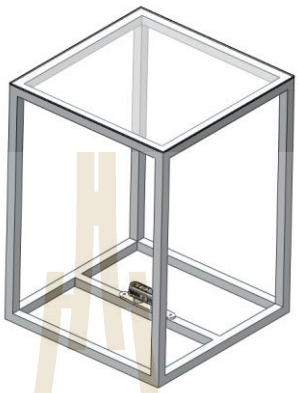
รูปที่ 3.3 อุปกรณ์สวมใส่สำหรับวัดค่าเมื่อสวมเข้ากับมือ



## 3.4 การออกแบบและสร้างแท่นกล้องสำหรับการตรวจจับมือ

ในการใช้งานกล้อง depth camera สำหรับการตรวจจับตำแหน่งของมือนั้น จะต้องคำนึงถึงระยะใกล้ที่สุดที่กล้องจะสามารถหาความลึกหรือตำแหน่งของวัตถุได้ (Stereoscopic Minimum depth distance) โดยกล้อง Intel Realsense D435 นั้น จะมีค่าดังกล่าวอยู่ที่ประมาณ 28 เซนติเมตร ดังนั้นระนาบอ้างอิงของมือจะต้องอยู่ที่ระยะมากกว่าค่าดังกล่าว จึงมีการออกแบบแท่นสำหริบติดกล้องและกำหนดระนาบของมือขึ้นมาดังรูปต่อไปนี้ โดยระยะของระนาบมือจากกล้องจะ

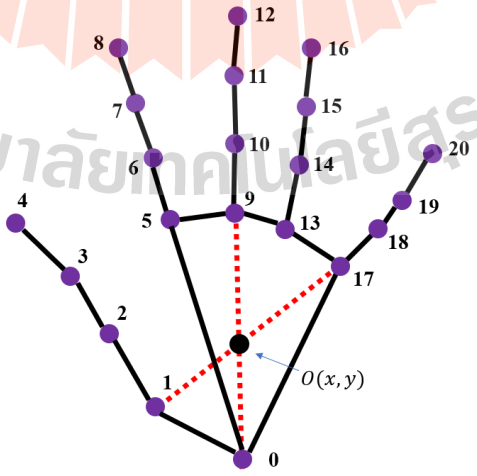
อยู่ที่ประมาณ 48 เซนติเมตร ส่วนความกว้างและยาวของระนายจะอยู่ที่ 40 x 40 เซนติเมตร ตามลำดับ



รูปที่ 3.4 รูป CAD ของแท่นอ้างอิงระนาบมือ

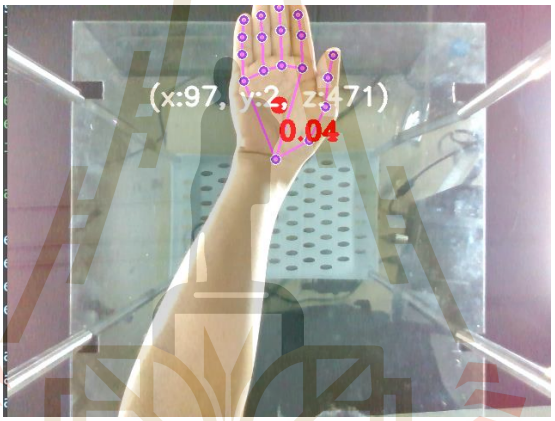
## 3.5 การออกแบบระบบหาตำแหน่งและท่าทางของมือด้วยกล้องความลึก (depth camera) และ IMU

ในการตรวจจับมือด้วย depth camera และโมเดล machine learning ของ MediaPipe จะสามารถสร้าง skeleton ของข้อต่อของมือได้ทั้งหมดจำนวน 20 จุดดังในรูปด้านล่าง โดยจุดอ้างอิงสำหรับการระบุตำแหน่งของมือ O(x,y) จะใช้จุดตัดของเส้นตรงที่ลากจากจุดที่ 1 ไป 17 และจุดที่ 0 ไป 9 ซึ่งจะเป็นพิกัดที่จะนำไปหาตำแหน่งในพิกัดบนโลกของความเป็นจริงร่วมกับ depth image




รูปที่ 3.5 จุดอ้างอิงของมือที่ได้จากโมเดล MediaPipe

รูปที่ 3.6 ตัวอย่างภาพความลึกจาก depth camera




รูปที่ 3.7 ตัวอย่างการหาตำแหน่ง XY และ $Z$ ของมือด้วย depth camera

ข้อมูลจาก depth image จะเป็นระยะห่างจากวัตถุถึงกล้อง หากต้องการพิกัด x y ของจุดอ้างอิง จะต้องนำพิกัดของจุดอ้างอิงในหน่วย pixel กับระยะห่างที่วัดได้ไปคำนวณกับ principal point ซึ่งเป็น intrinsic parameter ของกล้องที่เป็นตัวกำหนด optical center point ของกล้องดังสมการด้านล่า โดยปกติแล้ว principal point จะใกล้เคียง image center

$$x = D\left(O_x - P_{px}\right)/f_x \tag{3.1}$$

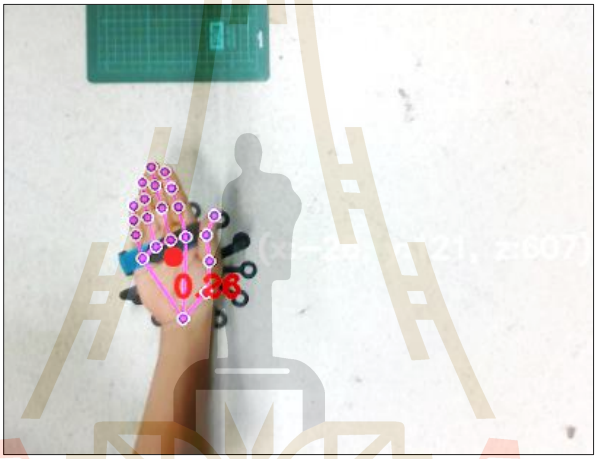
$$y = D\left(O_y - P_{py}\right)/f_y \tag{3.2}$$

เมื่อ D คือ ระยะจากกล้องถึงจุดอ้างอิง

$P_{px}$ และ $P_{py}$ คือ จุดกึ่งกลางในแนวแกน x และแกน y

$f_x$ และ $f_y$ คือ ความยาวโฟกัสของกล้อง

ในส่วนของการหาทิศทางการหมุนของมือด้วย depth camera จะมีการวาดเส้นอ้างอิงจากจุดที่ 9 ถึงจุดที่ 10 มุม 0 องศาถูกกำหนดเมื่อเส้นนี้ตั้งฉากกับระนาบแนวนอนของภาพ มุมจะเป็นบวกเมื่อหมุนทวนเข็มนาฬิกาจากตำแหน่งเริ่มต้น และเป็นลบเมื่อหมุนตามเข็มนาฬิกา ตัวอย่างของการตรวจวัดมุมการหมุนของมือด้วย depth camera แสดงดังในรูปต่อไปนี้



รูปที่ 3.8 ตัวอย่างการทดสอบหามุมของการหมุนของมือที่มุม 15 องศา

ส่วนในการตรวจจับท่าทางของมือด้วย IMU จะใช้ตัวกรอง Madgwick เพื่อหา orientation ของมือออกมา โดยจะพิจารณาค่าของ Roll และ pitch เป็นหลัก ส่วนค่า yaw จะใช้ค่าที่ได้จาก depth camera เพื่อหลีกเลี่ยงปัญหา gyro drift ของ IMU ในแนวแกน yaw และค่า yaw ที่สามารถอ่านได้จาก depth camera จะเป็น absolute data ทำให้ค่าที่อ่านได้มีความเสถียรและแม่นยำกว่า

## 3.6 การออกแบบระบบติดตามการเคลื่อนที่ของมือ (hand tracking system)

ในงานวิจัยนี้ตำแหน่งและท่าทางของมือจะถูกกรองและติดตามด้วย linear Kalman filter จากการที่มีข้อมูล pose ก่อนหน้าของมือ และข้อมูลการวัดปัจจุบัน จะทำให้ Kalman filter สามารถติดตามการเคลื่อนที่ของมือได้อย่างแม่นยำ โดยใช้ state equation และ measurement equation ในการอธิบายระบบ กำหนดให้ hand position coordinate สำหรับ Kalman filter คือ $[x\ y\ z]^T$ โดยเป็นพิกัดของตำแหน่งบนโลกแห่งความเป็นจริงที่วัดได้จาก depth camera โดยค่าของ z

coordinate จะมีค่าเท่ากับระยะห่างของวัตถุกับกล้องที่วัดได้ ส่วน orientation ของมือ คือ $[\phi \quad \theta \quad \psi]^T$ เป็นค่ามุมของ roll, pitch และ yaw ตามลำดับ ความเร็วเชิงเส้นและความเร็วเชิงมุมของการเคลื่อนที่ของมือคือ $\left[v_x\ v_y\ v_z\ \ \omega_x\ \ \omega_y\ \ \omega_z\right]^T$ โดย $v_x$ , $v_y$ และ $v_z$ แทนส่วนประกอบความเร็วการเคลื่อนที่ในแนวแกน 3 มิติตามลำดับ ส่วน $\omega_x, \omega_y$และ $\omega_z$ แทนส่วนประกอบความเร็วเชิงมุมของการเคลื่อนที่รอบแนวแกน 3 มิติ ตามลำดับ หากตั้งสมมติฐานให้การอัพเดทการ tracking ทำใน time interval สั้น ๆ จะสามารถ assume ได้ว่าการเปลี่ยนแปลงเป็นแบบ uniform และจะสามารถเขียน state matrix ได้ดังต่อไปนี้ โดยกำหนดให้เวลาปัจจุบัน คือ $k$

$$s_k = \left[x_k\ \ y_k\ \ z_k\ \ \phi_k\ \ \theta_k\ \ \psi_k\ \ v_{x_k}\ v_{y_k}\ v_{z_k}\ \omega_{x_k}\ \ \omega_{y_k}\ \ \omega_{z_k}\right]^T \tag{3.3}$$

$$S_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ \phi_{k-1} \\ \theta_{k-1} \\ \psi_{k-1} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \\ v_{z_{k-1}} \\ \omega_{x_{k-1}} \\ \omega_{y_{k-1}} \\ \omega_{z_{k-1}} \end{bmatrix} \tag{3.4}$$

เมื่อ depth camera และ IMU sensor สามารถตรวจวัดการเคลื่อนที่ของมือได้ measurement data จากเซนเซอร์ทั้งสองจะถูกนำไป update ค่าที่ประมาณได้จาก Kalman predictor โดย measurement matrix ของการวัด ($Z_k$) สำหรับป้อนให้ Kalman filter จะอยู่ในรูปของ

$$Z_k = HS_k + v_k \tag{3.5}$$

เมื่อ $H$ คือ state- to-measurement matrix which ถูกใช้เพื่อแปลง state space ไปหา measurement space และ $v_k$ คือ noise in measurement process โดย measurement vector at time k ของ depth camera คือ

$$Z_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} S_k + v_k \tag{3.6}$$

measurement vector at time k ของ IMU คือ

$$Z_k = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} S_k + v_k \quad (3.7)$$

The error covariance matrix in prediction state $(P_k^-)$ can be calculated by the equation as follow

$$P_k^- = AP_{k-1}A^T + Q \quad (3.8)$$

เมื่อ $Q$ *คือ* process noise covariance ของ Kalman filter ใน prediction step หลังจากผ่าน precition step ขั้นตอนต่อไปคือ update step ซึ่งจะมีการคำนวณ Kalman gain $(K_k)$โดยใช้ state-to-measurement matrix $(H)$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (3.9)$$

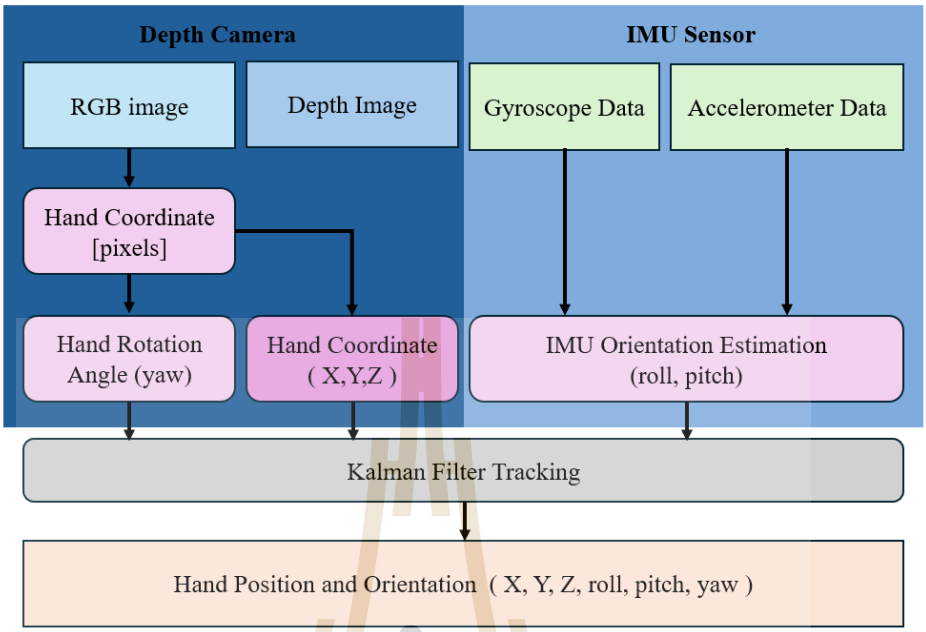
เมื่อ $R$ คือ measurement convariance matrix ค่าของ $R$ จะถูกสังเกตโดยใช้ข้อมูลของการทดลอง เมื่อได้ Kalman gain มาแล้ว ค่าของ Kalman gain จะถูกนำไป update state ที่ได้จากการ predict มา $(S_k^-)$ ตามสมการดังต่อไปนี้

$$S_k = S_k^- + K_k (Z_k - HS_k^-) \quad (3.10)$$

หลังจากนั้น error covariance matrix $P_k^+$ จะถูก update ค่า uncertainty อีกครั้งผ่านสมการ

$$P_k^+ = (I - K_k H) \cdot P_k^- \quad (3.11)$$

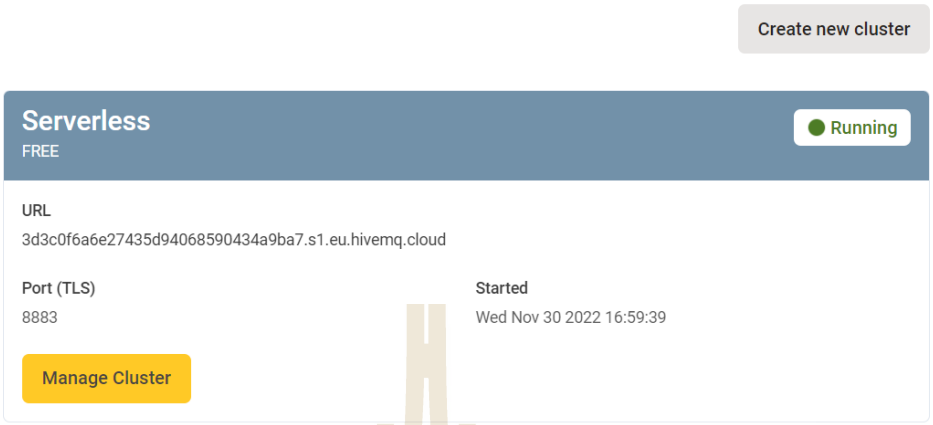
การทำงานของ Kalman filter จะทำอย่างต่อเนื่อง เพื่อประมาณสภาวะการเคลื่อนที่จริงของมือให้มีข้อมูลที่แม่นยำที่สุด โดยแผนผังแสดงการเชื่อมโยงของข้อมูลในระบบสำหรับงานวิจัยนี้แสดงดังต่อไปนี้



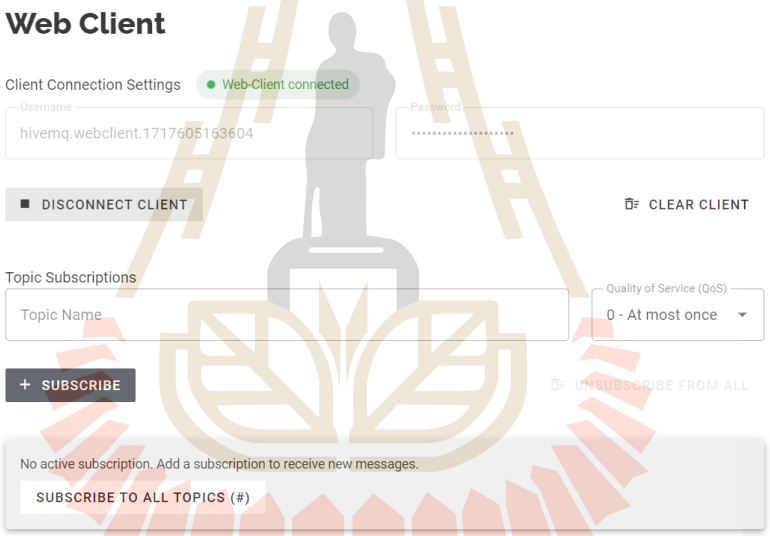

รูปที่ 3.9 แผนผังแสดงภาพรวมของการส่งข้อมูลของระบบ

## 3.7 การออกแบบระบบส่งผ่านข้อมูล

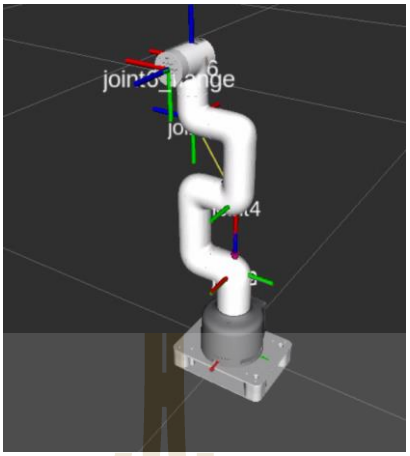
ระบบส่งผ่านข้อมูลจะใช้การเชื่อมต่อไปยัง Broker HiveMQ ซึ่งเป็น cloud broker สำหรับการเชื่อมต่อระบบ IoT ผ่านอินเทอร์เน็ตด้วยโปรโตคอล MQTT โดยในการส่งข้อมูลจากตัว tracking filter ไปยัง HiveMQ จะเขียนโปรแกรมด้วยภาษา Python โดยใช้ library Paho.MQTT จากนั้นในฝั่งของการรับข้อมูลจาก HiveMQ เพื่อส่งไปยัง MyCobot 280 จะใช้ Python ในการเขียนโปรแกรมเช่นเดียวกัน ซึ่งในฝั่งที่รับข้อมูลไปจะใช้ ROS และ MyCobot 280 API เป็นตัวสั่งการหลังจากรับค่าตำแหน่งและท่าทางของมุมไปแล้ว ตัวอย่างหน้าตาของ Hive MQ และ MyCobot280 ROS API แสดงดังในรูปต่อไปนี้ โดยในการ monitor การส่งข้อมูลผ่าน HiveMQ สามารถใช้ web client ที่สร้างขึ้นบน HiveMQ ลองทดสอบการส่งผ่านข้อมูลต่าง ๆ ได้




รูปที่ 3.10 หน้าตาของ dashboard ของ HiveMQ




รูปที่ 3.11 ตัวอย่างหน้าจอ WebClient ที่ใช้ monitor การส่งข้อมูล

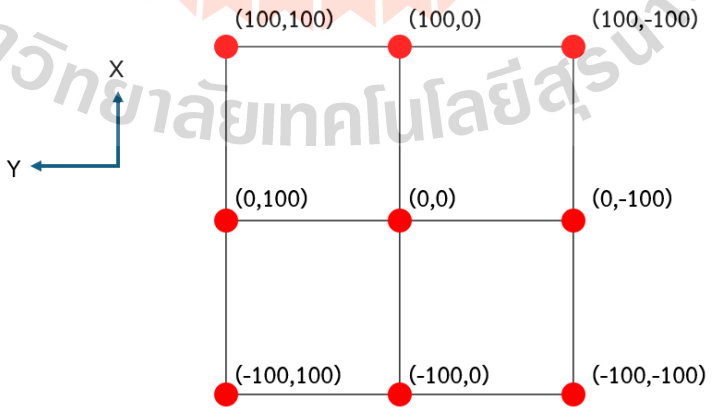


รูปที่ 3.12 ตัวอย่างหน้าจอ monitor การเคลื่อนที่ของแขนกล mycobot280

## 3.8 การทดสอบความแม่นยำของการตรวจจับด้วยกล้องวัดความลึก

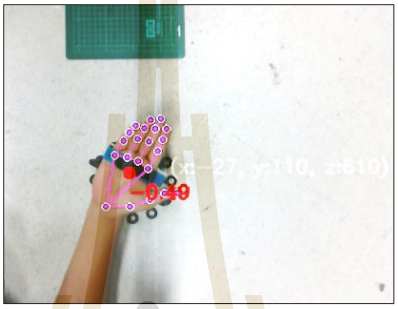
ในการทดสอบความแม่นยำของการตรวจจับด้วยกล้องวัดความลึกจะแบ่งการทดสอบเป็น 2 ส่วน โดยส่วนที่หนึ่งจะเป็นการทดสอบวัดความแม่นยำของการหาตำแหน่ง และส่วนที่สองจะเป็นการทดสอบเพื่อวัดความแม่นยำของการหามุม yaw ของมือ

ในการทดสอบหาความแม่นยำของการวัดตำแหน่งของมือ จะทำการทดสอบบนระนาบที่กำหนดเอาไว้โดยใช้ระนาบที่ห่างจากกล้องประมาณ 48 เซนติเมตรหรือ 480 มิลลิเมตร โดยมีกล้องอยู่ที่ตำแหน่งศูนย์กลาง แล้วกำหนดจุดวัดขึ้นมา 9 จุดที่รู้ระยะที่แน่นอนทั้งสามแกน จากนั้นทำการนำมือไปวางบนจุดที่กำหนด โดยให้ฝ่ามือเป็นจุดอ้างอิง จากนั้นทำการวัดผลและบันทึกผลเป็นตารางเพื่อนำมาสรุปผลข้อมูล โดยจุดที่กำหนดสำหรับการวัดจะใช้จุดดังรูปต่อไปนี้




รูปที่ 3.13 จุดอ้างอิงสำหรับการตรวจวัดตำแหน่งด้วยกล้องความลึก

ส่วนการทดสอบหาความแม่นยำของการวัดมุม yaw จากกล้อง จะใช้การสร้างแม่พิมพ์ด้วย 3D printer ขึ้นมาแล้วนำถึงมือไปวางไว้ในตำแหน่งที่กำหนดเอาไว้ เพื่อให้ได้มุมตามที่ต้องการ แล้วนำไปเปรียบเทียบกับค่าที่อ่านได้จากกล้อง โดยค่ามุมที่ทดสอบจะเป็นมุมตั้งแต่ 0 ถึง 90 องศา โดยเพิ่มทีละ 15 องศาทั้งด้านทิศตามเข็มและทวนเข็มนาฬิกา ตัวอย่างการทดสอบความแม่นยำของมุมแสดงดังในรูปต่อไปนี้



รูปที่ 3.14 ตัวอย่างการทดสอบวัดความแม่นยำการตรวจจับมุมของมือด้วยกล้องความลึก

## 3.9 การทดสอบความแม่นยำของการตรวจจับมุมด้วย IMU

การทดสอบความแม่นยำการวัดมุมของมือด้วย IMU จะทำโดยการนำถุงมือที่มี IMU ติดตั้งอยู่ไปวางไว้บนแท่นที่ถูกสร้างขึ้นเป็นมุมต่าง ๆ เอาไว้ โดยเริ่มจาก 0 ถึง 90 องศา ซึ่งทำจาก 3D printer เพื่อให้ได้ค่ามุมเทียบที่มีความแม่นยำมากที่สุดจากนั้นนำค่าที่ได้บันทึกไว้เป็นตารางทั้งในแนวแกน pitch และ roll

## 3.10 การทดสอบการส่งผ่านข้อมูลและการควบคุม

ในการทดสอบการส่งผ่านข้อมูลและการควบคุม จะใช้การ monitor การรับส่งข้อมูลผ่าน HiveMQ Web client ในกรยืนยันการรับส่งข้อมูลและความถูกต้องของข้อมูล โดยการส่งข้อมูลจะทำทุก ๆ 0.1 วินาที แล้วดูผลการตอบสนองของแขนกล หากหุ่นยนต์สามารถตอบสนองตามท่าทางของการเคลื่อนที่ข้อมือในข้อมูลชุดดังกล่าวได้ จะถือว่าผ่าน โดยจะทำการทดสอบทั้งหมด 20 ครั้งแล้วทำเป็นตารางสรุปผลข้อมูล

## 3.11 การทดสอบการควบคุมร่วมกับหุ่น MyCobot Pi280

ในการทดสอบการควบคุม MyCobot Pi280 จะนำท่าทางและตำแหน่งของมือส่งผ่าน MQTT ไปยังหุ่นที่อยู่ในเครือข่ายไร้สาย (Wireless LAN) เดียวกัน โดยจะดึงข้อมูลของตำแหน่งและ

ท่าทางของมือไปทุก ๆ 4 วินาที เพื่อให้หุ่นแขนกลสามารถเคลื่อนที่ไปจุดหมายได้ตามที่กำหนด เวลาที่ใช้ในการทดสอบอยู่ที่ 1 นาที (60 วินาที) จากนั้นจะทำการบันทึกข้อมูลของจุดที่ใช้ในการเคลื่อนที่ไปเทียบกับท่าทางของแขนกล

# บทที่ 4
# ผลการวิจัยและอภิปรายผล

## 4.1 กล่าวนำ

ในส่วนนี้ผลการทดสอบจะถูกนำเสนอในรูปแบบของตาราง รูปภาพ กราฟ หรือแผนผังการเปรียบเทียบอื่น ๆ เพื่อให้ข้อมูลมีความชัดเจนและเข้าใจง่าย จากนั้นจะเข้าสู่การวิเคราะห์ผลที่ได้จากการทดสอบ เพื่อสรุปข้อมูลอย่างเป็นระบบและสอดคล้องกับวัตถุประสงค์ของงานวิจัย

## 4.2 ผลการทดสอบความแม่นยำของการตรวจจับด้วยกล้องวัดความลึก

เมื่อทำการทดสอบการตรวจวัดตำแหน่งของมือด้วยกล้องวัดความลึก จะได้ตารางผลการทดสอบทั้งหมดดังตารางต่อไปนี้

ตารางที่ 4.1 ผลการทดสอบความแม่นยำในการตรวจวัดตำแหน่งของมือ

| X [mm] | | | | Y [mm] | | | | Z [mm] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Actual | Measured | Error | absolute error | Actual | Measured | Error | absolute error | Actual | Measured | Error | absolute error |
| 100 | 97 | -3 | 3 | 100 | 103 | 3 | 3 | 475 | 475 | 0 | 0 |
| 100 | 97 | -3 | 3 | 0 | 2 | 2 | 2 | 475 | 471 | -4 | 4 |
| 100 | 97 | -3 | 3 | -100 | -97 | 3 | 3 | 475 | 475 | 0 | 0 |
| 0 | 0 | 0 | 0 | 100 | 98 | -2 | 2 | 475 | 478 | 3 | 3 |
| 0 | 3 | 3 | 3 | 0 | 1 | 1 | 1 | 475 | 477 | 2 | 2 |
| 0 | -2 | -2 | 2 | -100 | -98 | 2 | 2 | 475 | 479 | 4 | 4 |
| -100 | -104 | -4 | 4 | 100 | 100 | 0 | 0 | 475 | 477 | 2 | 2 |
| -100 | -102 | -2 | 2 | 0 | -2 | -2 | 2 | 475 | 476 | 1 | 1 |
| -100 | -102 | -2 | 2 | -100 | -95 | 5 | 5 | 475 | 480 | 5 | 5 |

ตารางที่ 4.2 สรุปผลการทดสอบการตรวจจับตำแหน่งด้วยกล้องวัดความลึก

| | X [mm] | Y [mm] | Z [mm] |
|---|---|---|---|
| Average | 2.44 | 2.22 | 2.33 |
| Min | 0 | 0 | 0 |
| Max | 4 | 5 | 5 |

ตารางที่ 4.3 ผลการทดสอบความแม่นยำในการตรวจวัดมุม yaw ของมือ

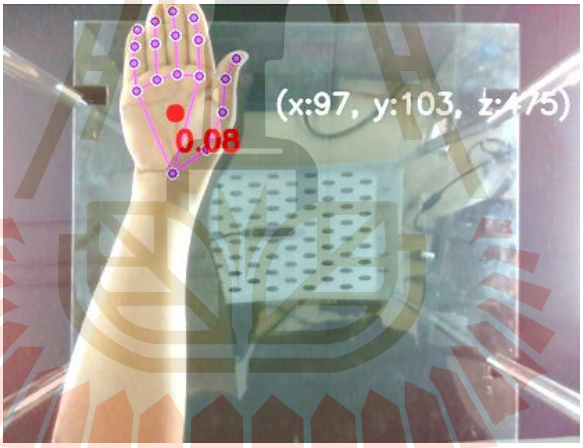
| Yaw Angle [rad] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| References | | Measured Angle [rads] | | | Error [rads] | | | |
| Degs | Rads | 1 | 2 | 3 | 1 | 2 | 3 | Absolute Average |
| 90 | 1.57 | 1.57 | 1.57 | 1.58 | 0.00 | 0.00 | -0.010 | 0.004 |
| 75 | 1.31 | 1.33 | 1.33 | 1.30 | -0.02 | 0.00 | 0.030 | 0.017 |
| 60 | 1.05 | 1.02 | 1.05 | 1.04 | 0.03 | -0.03 | 0.010 | 0.022 |
| 45 | 0.79 | 0.78 | 0.78 | 0.79 | 0.01 | 0.00 | -0.010 | 0.005 |
| 30 | 0.52 | 0.52 | 0.52 | 0.51 | 0.00 | 0.00 | 0.010 | 0.005 |
| 15 | 0.26 | 0.25 | 0.26 | 0.26 | 0.01 | -0.01 | 0.000 | 0.007 |
| 0 | 0.00 | 0.02 | 0.00 | 0.01 | -0.02 | 0.02 | -0.010 | 0.017 |
| -15 | -0.26 | -0.24 | -0.26 | -0.23 | -0.02 | 0.02 | -0.030 | 0.024 |
| -30 | -0.52 | -0.51 | -0.52 | -0.49 | -0.01 | 0.01 | -0.030 | 0.018 |
| -45 | -0.79 | -0.77 | -0.78 | -0.81 | -0.02 | 0.01 | 0.030 | 0.018 |
| -60 | -1.05 | -1.07 | -1.03 | -1.02 | 0.02 | -0.04 | -0.010 | 0.024 |
| -75 | -1.31 | -1.32 | -1.31 | -1.32 | 0.01 | -0.01 | 0.010 | 0.010 |
| -90 | -1.57 | -1.57 | -1.58 | -1.58 | 0.00 | 0.01 | 0.000 | 0.004 |
| Average [deg] | | | | | 0.772 | | | |
| Min [deg] | | | | | 0.206 | | | |
| Max [deg] | | | | | 1.390 | | | |

จากผลการทดสอบความแม่นยำในการตรวจวัดตำแหน่งของมือด้วยกล้องวัดความลึกในตารางที่ 4.1 พบว่า ค่าความคลาดเคลื่อนเฉลี่ยในแกน X, Y และ Z อยู่ที่ 2.44 มม., 2.22 มม., และ 2.33 มม. ตามลำดับ โดยมีค่าความคลาดเคลื่อนสูงสุดในแกน X อยู่ที่ 4 มม. และในแกน Y และ Z อยู่ที่ 5 มม. ค่าความคลาดเคลื่อนน้อยที่สุดในทุกแกนอยู่ที่ 0 มม. การวัดผลในตารางที่ 4.1 แสดงให้เห็นว่าในหลาย ๆ การวัดผล ค่าความคลาดเคลื่อนมีแนวโน้มที่จะแกว่งอยู่ในช่วง 2-3 มม. ซึ่งเป็น

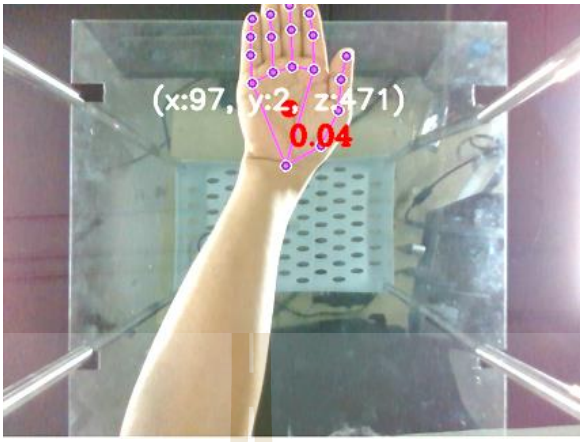
ระดับที่ยอมรับได้สำหรับการใช้งาน และเมื่อนำตัวเลขทั้งหมดมาหาค่าเฉลี่ย ค่าสูงสุดและค่าต่ำสุดของความคลาดเคลื่อนจะได้ผลลัพธ์ดังในตารางที่ 4.2

จากผลการทดสอบความแม่นยำในการตรวจวัดมุม yaw ของมือด้วยกล้องวัดความลึก พบว่าค่าความคลาดเคลื่อนเฉลี่ยอยู่ที่ 0.772 องศา ซึ่งเป็นค่าที่บ่งบอกถึงความแม่นยำในการวัดมุม yaw ของกล้อง ความคลาดเคลื่อนสูงสุดอยู่ที่ 1.390 องศา ในกรณีของมุม -60 องศา และความคลาดเคลื่อนต่ำสุดอยู่ที่ 0.206 องศา ในกรณีของมุม 15 องศา ความแตกต่างของค่าความคลาดเคลื่อนในแต่ละมุมแสดงให้เห็นว่ากล้องมีความแม่นยำที่คงที่ในระดับที่ยอมรับได้ ไม่ว่าจะเป็นการวัดมุมในเชิงบวกหรือลบ ค่าความคลาดเคลื่อนที่ตรวจพบอยู่ในระดับที่ดีและไม่ส่งผลกระทบต่อการใช้งานจริง การทดสอบนี้ยืนยันว่ากล้องวัดความลึกมีความเสถียรและความน่าเชื่อถือในการตรวจวัดมุม yaw ของมือ

โดยตัวอย่างชุดข้อมูลของการทดสอบการตรวจวัดตำแหน่งของมือด้วยกล้องวัดความลึกจะมาจากภาพดังต่อไปนี้



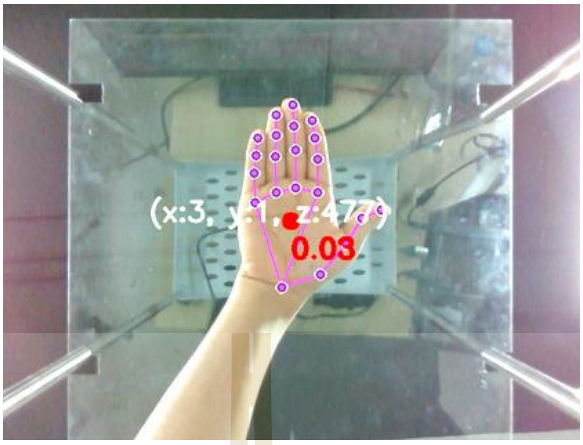

รูปที่ 4.1 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 1



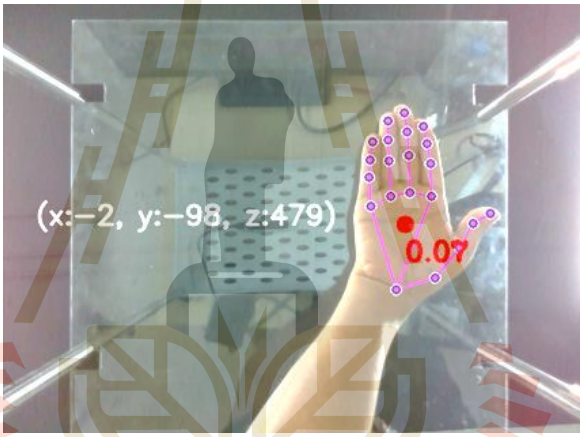

รูปที่ 4.2 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 2


รูปที่ 4.3 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 3


รูปที่ 4.4 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 4

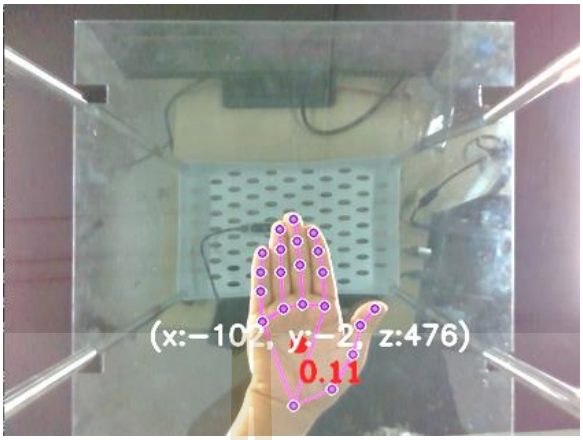



รูปที่ 4.5 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 5




รูปที่ 4.6 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 6

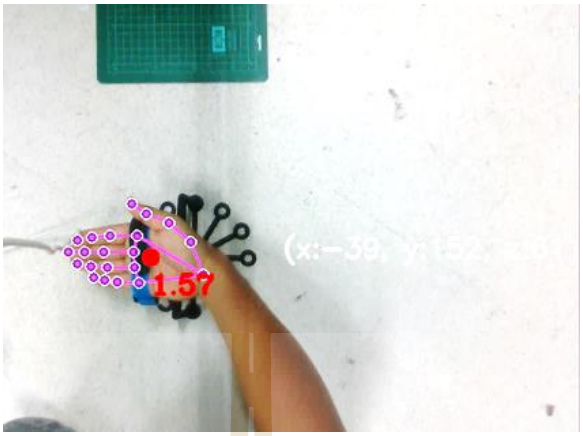

รูปที่ 4.7 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 7




รูปที่ 4.8 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 8


รูปที่ 4.9 การตรวจวัดตำแหน่งของมือ ณ จุดทดสอบที่ 9

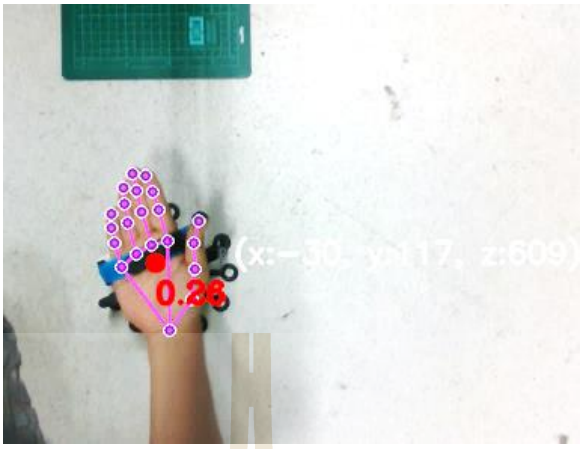
ตัวอย่างชุดข้อมูลของการทดสอบการตรวจวัดมุม yaw ของมือด้วยกล้องวัดความลึกแสดงดังในรูปต่อไปนี้



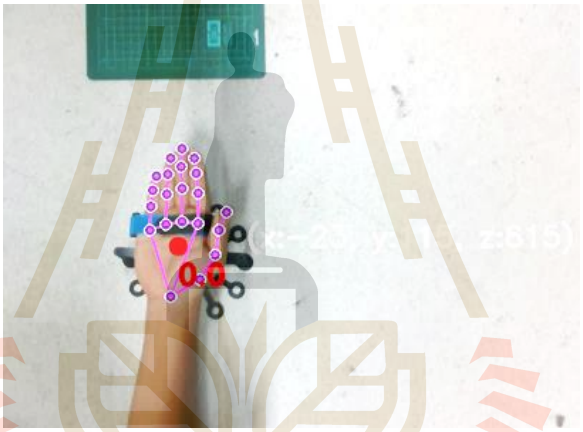

รูปที่ 4.10 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 90 องศา


รูปที่ 4.11 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -90 องศา


รูปที่ 4.12 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -45 องศา




รูปที่ 4.13 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 15 องศา



รูปที่ 4.14 การวัดมุม yaw ด้วยกล้องความลึกที่มุม 0 องศา


รูปที่ 4.15 การวัดมุม yaw ด้วยกล้องความลึกที่มุม -30 องศา

## 4.3 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU

เมื่อทำการทดสอบการตรวจวัดท่าทางของมือในแนวแกน pitch และ roll จะได้ตารางผลการทดสอบทั้งหมดดังตารางต่อไปนี้

ตารางที่ 4.4 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน pitch

| Pitch angle[deg] | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 15 | 30 | 45 | 60 | 75 | 90 |
| 1 | -0.94 | 15.51 | 30.94 | 45.35 | 61.05 | 75.50 | 86.06 |
| 2 | -0.92 | 15.45 | 30.37 | 45.32 | 61.05 | 75.42 | 86.12 |
| 3 | -0.92 | 15.38 | 30.39 | 45.29 | 60.95 | 75.30 | 86.08 |
| 4 | -0.91 | 15.38 | 30.51 | 45.25 | 58.47 | 75.26 | 86.09 |
| 5 | -0.91 | 15.41 | 30.47 | 45.23 | 58.77 | 75.21 | 86.09 |
| 6 | -0.90 | 15.46 | 30.42 | 45.21 | 60.38 | 75.16 | 86.08 |
| 7 | -0.90 | 15.50 | 30.36 | 45.19 | 60.56 | 75.07 | 86.11 |
| 8 | -0.88 | 15.44 | 30.44 | 45.16 | 60.63 | 74.93 | 86.08 |
| 9 | -0.87 | 15.44 | 30.47 | 45.15 | 60.69 | 74.61 | 86.06 |
| 10 | -0.87 | 15.44 | 30.43 | 45.14 | 60.73 | 74.69 | 86.04 |
| 11 | -0.87 | 15.39 | 30.34 | 45.14 | 60.77 | 74.75 | 86.04 |
| 12 | -0.85 | 15.38 | 30.17 | 45.13 | 60.79 | 74.71 | 86.11 |
| 13 | -0.85 | 15.42 | 30.10 | 45.17 | 60.81 | 74.43 | 86.09 |
| 14 | -0.83 | 15.45 | 30.08 | 45.14 | 60.84 | 73.86 | 86.05 |
| 15 | -0.83 | 15.39 | 30.22 | 45.14 | 60.86 | 75.01 | 86.06 |
| 16 | -0.83 | 15.39 | 30.22 | 45.15 | 60.90 | 75.12 | 86.06 |
| 17 | -0.83 | 15.29 | 30.16 | 45.08 | 60.92 | 75.16 | 86.04 |
| 18 | -0.83 | 15.28 | 30.18 | 45.11 | 60.94 | 75.15 | 86.07 |
| 19 | -0.83 | 15.32 | 30.19 | 45.05 | 60.96 | 75.18 | 86.08 |
| 20 | -0.83 | 15.33 | 30.18 | 45.05 | 60.97 | 75.23 | 86.12 |
| Ave | 0.03 | 15.40 | 30.33 | 45.17 | 60.60 | 74.99 | 86.08 |
| Error | -0.03 | -0.40 | -0.33 | -0.17 | -0.60 | 0.01 | 3.92 |
| Total Average | 0.34 | | | | | | |

ตารางที่ 4.5 สรุปผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน pitch

| Pitch Angle | 0 | 15 | 30 | 45 | 60 | 75 | 90 |
|---|---|---|---|---|---|---|---|
| Average Angle | -0.87 | 15.40 | 30.33 | 45.17 | 60.60 | 74.99 | 86.08 |
| Average Error | -0.03 | -0.40 | -0.33 | -0.17 | -0.60 | 0.01 | 3.92 |
| Total Average Error | 0.34 | | | | | | |

จากผลการทดสอบการตรวจวัดท่าทางของมือในแนวแกน pitch ตามตารางมรา 4-4 และ 4-5 โดยใช้ IMU ที่มุมตั้งแต่ 0 ถึง 90 องศา พบว่าความแม่นยำของการวัดมีความคลาดเคลื่อนที่ยอมรับได้ในช่วงมุมต่ำกว่า 75 องศา โดยมีค่าความผิดพลาดเฉลี่ยอยู่ในช่วง -0.60 ถึง 0.01 องศา อย่างไรก็ตาม เมื่อมุมวัดสูงขึ้นถึง 90 องศา ค่าความผิดพลาดเฉลี่ยเพิ่มขึ้นถึง 3.92 องศา แสดงให้เห็นถึงข้อจำกัดของ IMU ในการวัดมุมที่สูงขึ้น แม้ว่าค่าความผิดพลาดรวมเฉลี่ยทั้งระบบจะอยู่ที่ 0.34 องศา ซึ่งถือว่ายอมรับได้ในงานวิจัยบางประเภท ผลการทดสอบนี้จึงชี้ให้เห็นถึงขีดความสามารถและข้อจำกัดของ IMU ที่ควรนำไปปรับปรุงและพัฒนาในการออกแบบและใช้งานในอนาคต

ตารางที่ 4.6 ผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน roll

| Roll angle [deg] | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 15 | 30 | 45 | 60 | 75 | 90 |
| 1 | 0.41 | 15.30 | 30.90 | 45.37 | 60.13 | 74.83 | 86.19 |
| 2 | 0.28 | 15.18 | 30.98 | 45.32 | 60.12 | 74.83 | 86.11 |
| 3 | 0.26 | 15.12 | 30.89 | 45.35 | 60.06 | 74.82 | 86.11 |
| 4 | 0.24 | 15.14 | 30.67 | 45.30 | 59.98 | 74.77 | 86.11 |
| 5 | 0.24 | 15.11 | 30.70 | 45.29 | 59.95 | 74.78 | 86.11 |
| 6 | 0.25 | 15.11 | 30.77 | 45.23 | 59.80 | 74.79 | 86.12 |
| 7 | 0.23 | 15.03 | 30.82 | 45.30 | 59.36 | 74.83 | 86.07 |
| 8 | 0.23 | 14.96 | 30.88 | 45.55 | 59.40 | 74.83 | 86.04 |
| 9 | 0.22 | 14.91 | 30.86 | 45.59 | 59.36 | 74.83 | 86.07 |
| 10 | 0.23 | 15.03 | 30.74 | 45.60 | 59.56 | 74.85 | 86.03 |
| 11 | 0.21 | 14.98 | 30.62 | 45.62 | 58.94 | 74.84 | 86.03 |
| 12 | 0.18 | 15.16 | 30.60 | 45.70 | 59.06 | 74.84 | 86.03 |
| 13 | 0.18 | 15.18 | 30.61 | 45.73 | 58.92 | 74.83 | 86.03 |
| 14 | 0.14 | 15.25 | 30.59 | 45.75 | 59.01 | 74.82 | 86.02 |
| 15 | 0.16 | 15.03 | 30.51 | 45.73 | 59.64 | 74.86 | 86.02 |
| 16 | 0.19 | 15.01 | 30.03 | 45.67 | 59.84 | 74.87 | 86.02 |
| 17 | 0.18 | 15.12 | 30.03 | 45.73 | 59.57 | 74.89 | 86.04 |
| 18 | 0.17 | 15.14 | 30.04 | 45.30 | 59.48 | 74.89 | 86.06 |
| 19 | 0.17 | 15.27 | 30.03 | 45.30 | 59.43 | 74.90 | 86.04 |
| 20 | 0.18 | 15.26 | 30.10 | 45.32 | 59.41 | 74.89 | 86.03 |
| Ave | 0.22 | 15.11 | 30.57 | 45.49 | 59.55 | 74.84 | 86.06 |
| Error | -0.22 | -0.11 | -0.57 | -0.49 | 0.45 | 0.16 | 3.94 |
| Total Average | 0.45 | | | | | | |

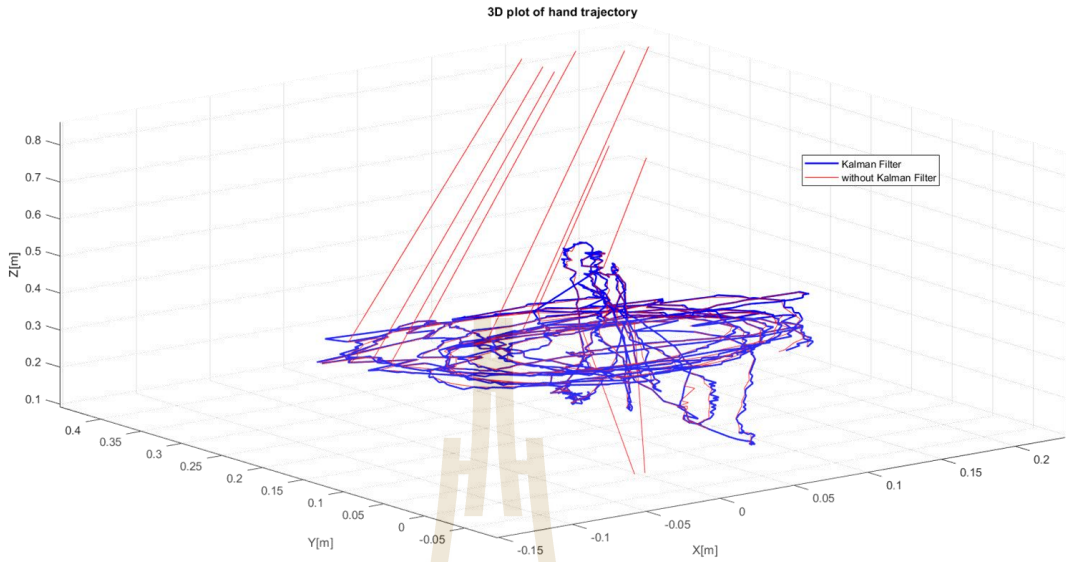
ตารางที่ 4.7 สรุปผลการทดสอบความแม่นยำของการวัดมุมด้วย IMU ในแนวแกน roll

| Roll Angle | 0 | 15 | 30 | 45 | 60 | 75 | 90 |
|---|---|---|---|---|---|---|---|
| Average Angle | 0.22 | 15.11 | 30.57 | 45.49 | 59.55 | 74.84 | 86.06 |
| Average Error | -0.22 | -0.11 | -0.57 | -0.49 | 0.45 | 0.16 | 3.94 |
| Total Average Error | 0.45 | | | | | | |

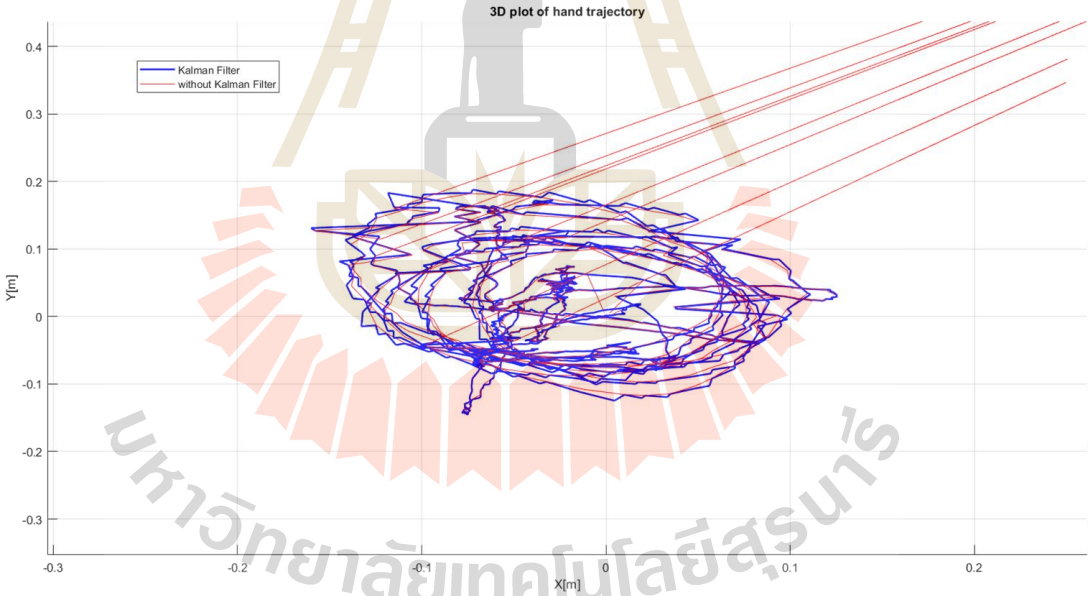
จากผลการทดสอบการตรวจวัดท่าทางของมือในแนวแกน roll โดยใช้ IMU ที่มุมตั้งแต่ 0 ถึง 90 องศาในตารางที่ 4.6 และ 4.7 พบว่า ความแม่นยำของการวัดมีความคลาดเคลื่อนในระดับที่ยอมรับได้ในมุมที่ต่ำกว่า 75 องศา โดยมีค่าความผิดพลาดเฉลี่ยอยู่ในช่วง -0.57 ถึง 0.16 องศา อย่างไรก็ตาม เมื่อมุมวัดสูงขึ้นถึง 90 องศา ค่าความผิดพลาดเฉลี่ยจะเพิ่มขึ้นถึง 3.94 องศา ซึ่งแสดงถึงข้อจำกัดของ IMU ในการวัดมุมที่สูงขึ้น แม้ว่าค่าความผิดพลาดรวมเฉลี่ยทั้งระบบจะอยู่ที่ 0.45 องศา ซึ่งถือว่าค่อนข้างน้อยและยอมรับได้ในงานวิจัยบางประเภท การทดสอบนี้ชี้ให้เห็นว่าการวัดมุมในแนวแกน roll มีความแม่นยำที่ดีในมุมต่ำ แต่เมื่อมุมสูงขึ้นการวัดจะมีความผิดพลาดเพิ่มขึ้น ผลการทดสอบนี้มีประโยชน์ในการปรับปรุงและพัฒนาการใช้งาน IMU ในการวัดท่าทางของมือในอนาคต เพื่อให้มีความแม่นยำสูงขึ้น

## 4.4 ผลการทดสอบการติดตามการเคลื่อนที่ของมือ

เมื่อทำการทดสอบการติดตามการเคลื่อนที่ของมือด้วยตัวกรองคาลมาน (Kalman filter) แล้วนำมาวาดกราฟเส้นทางการเคลื่อนที่ของมือได้ผลลัพธ์ดังต่อไปนี้




รูปที่ 4.16 เส้นทางการเคลื่อนที่ของมือที่ได้จากตัวกรองคาลมาน แบบ 3 มิติ




รูปที่ 4.17 เส้นทางการเคลื่อนที่ของมือที่ได้จากตัวกรองคาลมาน แบบ 2 มิติ

หลังจากการรวมเซ็นเซอร์ทั้งหมดและใช้ตัวกรองคาลมานเข้ามาในการติดตามการเคลื่อนที่ของมือ ความเสถียรของการติดตามการเคลื่อนที่ของมือมือได้รับการปรับปรุงอย่างมีนัยสำคัญ จากการทดสอบการเคลื่อนไหวของมือในลักษณะวงกลมในรูปที่ 4.16 และ รูปที่ 4.17 ซึ่งเปรียบเทียบ

เส้นทางการเคลื่อนไหวของมือทั้งที่มีและไม่มีการใช้ตัวกรองคาลมาน สามารถสังเกตได้ว่าในกรณีที่ข้อมูลไม่ได้ถูกประมวลผลด้วยตัวกรองคาลมาน จะมีบางช่วงที่เส้นสีแดงเบี่ยงเบนไปจากแนวโน้มที่ประเมินไว้ โดยเหตุการณ์นี้เกิดขึ้นเมื่อข้อมูลตำแหน่งที่ได้รับมาจากจากกล้องมีการสูญเสียการติดตามหรือการตรวจจับไป ทำให้ระยะที่อ่านได้ถูกปรับให้ไปตำแหน่งของระยะสูงสุด โดย เมื่อเปรียบเทียบกับข้อมูลที่ผ่านการประมวลผลด้วยตัวกรองคาลมาน จะเห็นได้ว่าตัวกรองคาลมานสามารถรักษาการติดตามได้ดี สามารถประมาณช่วงการเคลื่อนที่ที่หายไปได้อย่างต่อเนื่องดังที่แสดงโดยเส้นสีน้ำเงิน

## 4.5 ผลการทดสอบการส่งข้อมูล

ผลการทดสอบการส่งต่อข้อมูลระหว่างระบบควบคุม ในการทดสอบการส่งข้อมูลระหว่างหน่วยประมวลผลข้อมูลไปแขนกลจะมีการตรวจสอบความถูกต้องของการส่งข้อมูลในแต่ละขั้น และสามารถบันทึกผลได้เป็นตารางต่อไปนี้ โดยผลการทดสอบแต่ละขั้นจะให้ผลได้สองแบบคือผ่านและไม่ผ่าน

ตารางที่ 4.8 ผลการทดสอบการส่งข้อมูลระหว่างหน่วย

| ครั้งที่ | จากหน่วยประมวลผลไป HiveMQ | จาก HiveMQ มาที่แขนกล | แขนกลสามารถเคลื่อนที่ได้ | การเคลื่อนที่สอดคล้องกับมือ |
|---|---|---|---|---|
| 1 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 2 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 3 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 4 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 5 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 6 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 7 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 8 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 9 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 10 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 11 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 12 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 13 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 14 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |

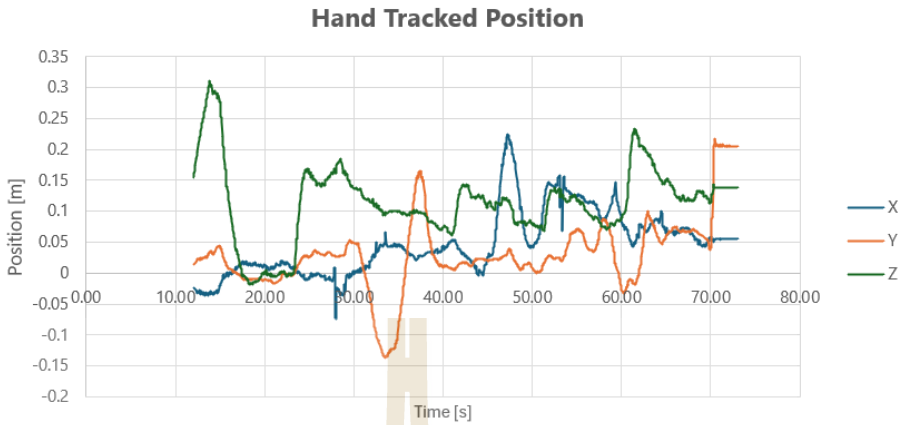
ตารางที่ 4.8 ผลการทดสอบการส่งข้อมูลระหว่างหน่วย (ต่อ)

| ครั้งที่ | จากหน่วยประมวลผลไป HiveMQ | จาก HiveMQ มาที่แขนกล | แขนกลสามารถเคลื่อนที่ได้ | การเคลื่อนที่สอดคล้องกับมือ |
|---|---|---|---|---|
| 15 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 16 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 17 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 18 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 19 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |
| 20 | ผ่าน | ผ่าน | ผ่าน | ผ่าน |

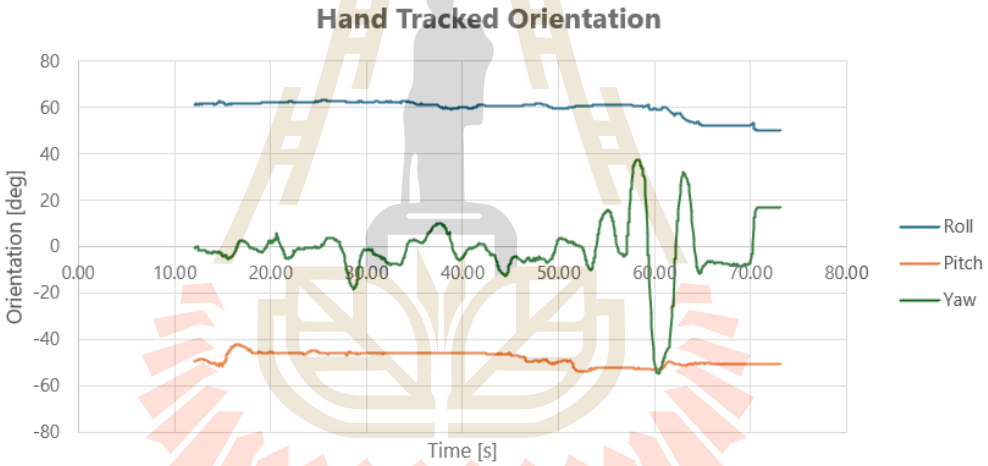
จากผลการทดสอบการส่งข้อมูลระหว่างหน่วยประมวลผลข้อมูลไปยังแขนกล พบว่าทุกขั้นตอนของการส่งข้อมูลตั้งแต่หน่วยประมวลผลไปยัง HiveMQ การส่งจาก HiveMQ ไปยังแขนกล การเคลื่อนที่ของแขนกล และการเคลื่อนที่ที่สอดคล้องกับมือ ได้ผลการทดสอบเป็น "ผ่าน" ทั้งหมดในทุกครั้ง รวม 20 ครั้ง ซึ่งแสดงถึงความถูกต้องและเสถียรของการเชื่อมต่อและการส่งข้อมูลระหว่างระบบ

## 4.6 ผลการทดสอบการทำงานร่วมกับหุ่นยนต์จริง

เมื่อทำการทดสอบการส่งข้อมูลของการติดตามการเคลื่อนที่ของมือเข้ากับหุ่นยนต์ Mycobot Pi280 ได้ผลการทดสอบการติดตามตำแหน่งและท่าทางของมือและสามารถนำมาวาดกราฟแสดงตำแหน่งและท่าทางได้ได้ดังรูปที่ 4.18 และรูปที่ 4.19 โดยเป็นค่าข้อมูลเทียบกับเวลาประมาณ 60 วินาทีที่ได้มีการบันทึกข้อมูลและทำการทดสอบ




รูปที่ 4.18 ความสัมพันธ์ของข้อมูลตำแหน่งเทียบกับเวลา

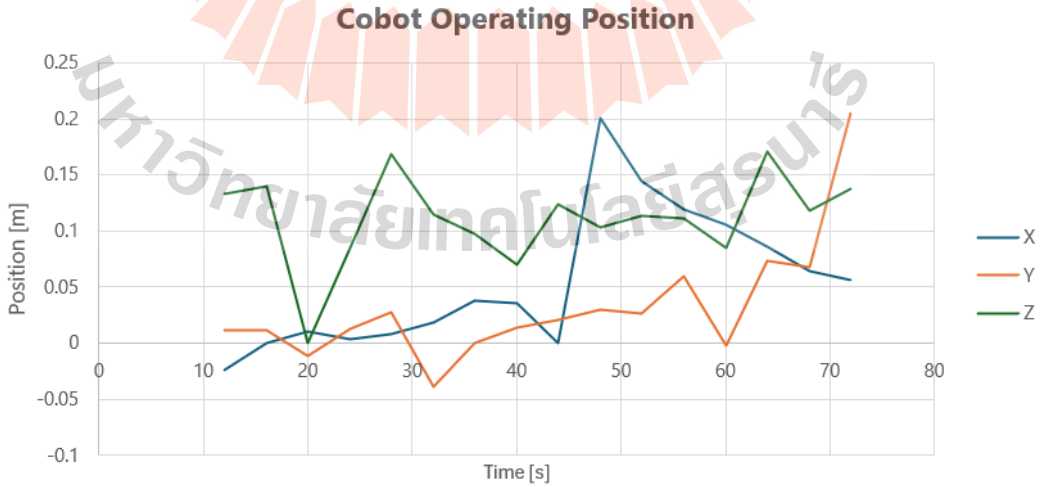



รูปที่ 4.19 ความสัมพันธ์ของข้อมูลท่าทางในมุมออยเลอร์เทียบกับเวลา

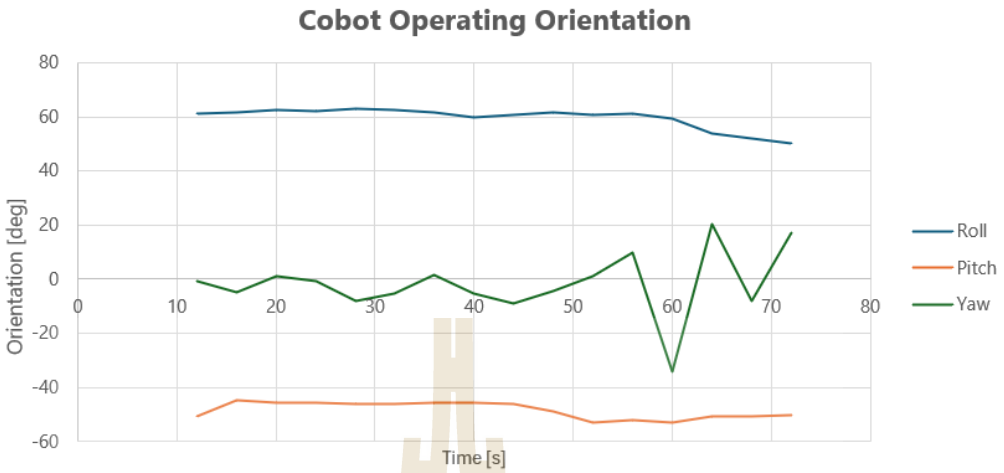
แต่เนื่องจากหุ่นยนต์จะต้องใช้เวลาแปลง Inverse Kinematics และเวลาในการเคลื่อนที่ไปยังจุดต่าง ๆ จึงไม่สามารถเคลื่อนที่ไปตามตำแหน่งและท่าทางของมือได้อย่างต่อเนื่องทันทีทันใดจึงมีการรับข้อมูลไปที่แขนกลทุก ๆ 4 วินาทีโดยตำแหน่งและท่าทางของมือที่แขนกลรับไปคำนวณการเคลื่อนที่และปฏิบัติการเคลื่อนที่ไปยังจุดที่กำหนดแสดงดังในตารางที่ 4.9 และสามารถวาดเป็นกราฟได้ดังรูปที่ 4.20 และ รูปที่ 4.21

ตารางที่ 4.9 ค่าข้อมูลตำแหน่งและท่าทางที่มีการนำไปสั่งการหุ่นยนต์ My Cobot Pi280

| เวลา | X | Y | Z | Roll | Pitch | Yaw |
|---|---|---|---|---|---|---|
| 12 | -0.024 | 0.012 | 0.133 | 61.168 | -50.611 | -0.854 |
| 16 | 0.000 | 0.012 | 0.140 | 61.68 | -44.812 | -4.824 |
| 20 | 0.010 | -0.011 | 0.000 | 62.399 | -45.548 | 1.098 |
| 24 | 0.003 | 0.013 | 0.085 | 62.241 | -45.669 | -0.526 |
| 28 | 0.008 | 0.028 | 0.169 | 62.818 | -45.876 | -7.908 |
| 32 | 0.018 | -0.039 | 0.115 | 62.350 | -45.891 | -5.286 |
| 36 | 0.038 | 0.000 | 0.097 | 61.414 | -45.817 | 1.752 |
| 40 | 0.036 | 0.014 | 0.07 | 59.994 | -45.653 | -5.378 |
| 44 | 0.000 | 0.021 | 0.124 | 60.761 | -45.988 | -9.063 |
| 48 | 0.201 | 0.030 | 0.103 | 61.690 | -48.941 | -4.576 |
| 52 | 0.144 | 0.026 | 0.114 | 60.493 | -52.965 | 1.057 |
| 56 | 0.119 | 0.060 | 0.111 | 61.224 | -52.102 | 9.619 |
| 60 | 0.106 | -0.002 | 0.085 | 59.262 | -52.859 | -34.077 |
| 64 | 0.086 | 0.073 | 0.171 | 53.926 | -50.519 | 20.186 |
| 68 | 0.064 | 0.068 | 0.118 | 52.057 | -50.480 | -8.017 |
| 72 | 0.056 | 0.205 | 0.138 | 50.040 | -50.285 | 17.189 |

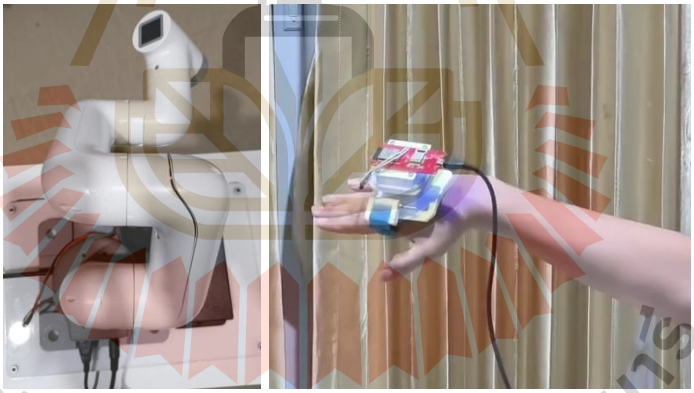



รูปที่ 4.20 ความสัมพันธ์ของข้อมูลตำแหน่งเทียบกับเวลาที่แขนกลรับไปปฏิบัติ

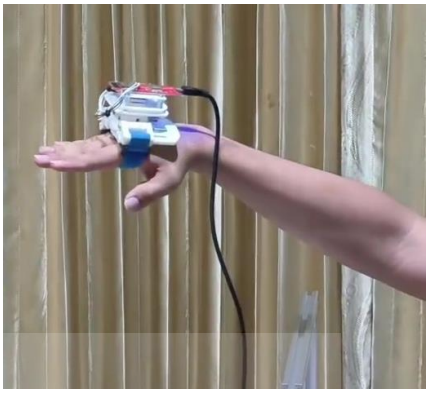



รูปที่ 4.21 ความสัมพันธ์ของข้อมูลท่าทางในมุมออยเลอร์เทียบกับเวลาที่แขนกลรับไปปฏิบัติ

ตัวอย่างของภาพการเคลื่อนที่ของแขนกลจริงแสดงดังในรูปที่ และรูปที่ ซึ่งเป็นการเคลื่อนที่ในเวลา 48และ 72 วินาทีตามลำดับ



รูปที่ 4.22 (ซ้าย) ภาพแขนกล (ขวา) มือที่ใช้ในการควบคุม ณ วินาทีที่ 48

รูปที่ 4.23 (ซ้าย) ภาพแขนกล (ขวา) มือที่ใช้ในการควบคุม ณ วินาทีที่ 72

# บทที่ 5
# สรุปผลและข้อเสนอแนะ

## 5.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อออกแบบและสร้างระบบควบคุมแขนกลด้วยการรับตำแหน่งและท่าทางจากฝ่ามือของมนุษย์โดยใช้เซ็นเซอร์วัดความเคลื่อนไหวและระบบระบุตำแหน่งด้วยกล้องสำหรับพัฒนาระบบควบคุมแขนกลระยะไกล โดยใช้การจับคู่แบบจลนศาสตร์ต่างกัน ด้วยท่าทางของมือของมนุษย์ ผลการทดสอบความแม่นยำในการตรวจวัดตำแหน่งของมือด้วยกล้องวัดความลึกพบว่าค่าความคลาดเคลื่อนเฉลี่ยในแกน X, Y และ Z อยู่ที่ 2.44 มม., 2.22 มม., และ 2.33 มม. ตามลำดับ โดยมีค่าความคลาดเคลื่อนสูงสุดในแกน X อยู่ที่ 4 มม. และในแกน Y และ Z อยู่ที่ 5 มม. การวัดผลแสดงให้เห็นว่าค่าความคลาดเคลื่อนมีแนวโน้มที่จะแกว่งอยู่ในช่วง 2-3 มม. ซึ่งถือว่าเป็นระดับที่ยอมรับได้ในข้อกำหนด 3 เซนติเมตรตามขอบเขตที่ตั้งเอาไว้

นอกจากนี้ ผลการทดสอบความแม่นยำในการตรวจวัดมุม yaw ของมือด้วยกล้องวัดความลึกแสดงให้เห็นว่าค่าความคลาดเคลื่อนเฉลี่ยอยู่ที่ 0.772 องศา ค่าความคลาดเคลื่อนสูงสุดอยู่ที่ 1.390 องศา และค่าความคลาดเคลื่อนต่ำสุดอยู่ที่ 0.206 องศา ความแม่นยำที่คงที่ในระดับที่ยอมรับได้ทั้งในมุมบวกและลบที่ 5 องศา ชี้ให้เห็นว่ากล้องวัดความลึกมีความเสถียรและความน่าเชื่อถือในการตรวจวัดมุม yaw ของมือ นอกจากนี้ผลการทดสอบความแม่นยำในการวัดมุม pitch และ roll ของมือด้วย IMU พบว่าค่าความคลาดเคลื่อนอยู่ในช่วงที่ยอมรับ 5 องศาเช่นกัน โดยในช่วงมุมต่ำกว่า 75 องศา โดยมีค่าความผิดพลาดเฉลี่ยอยู่ในช่วง -0.60 ถึง 0.16 องศา แต่เมื่อมุมวัดสูงขึ้นถึง 90 องศา ค่าความผิดพลาดเฉลี่ยเพิ่มขึ้นถึง 3.94 องศา แสดงถึงข้อจำกัดของ IMU ในการวัดมุมที่สูงขึ้น ผลการทดสอบนี้แสดงให้เห็นถึงความสามารถและข้อจำกัดของ IMU ที่ควรนำไปปรับปรุงและพัฒนาในการออกแบบและใช้งานในอนาคต

เพื่อศึกษาการควบคุมแขนกลแบบ Non-homogenous kinematic ในส่วนของการทดสอบการติดตามการเคลื่อนที่ของมือด้วยตัวกรองคาลมาน (Kalman filter) พบว่า ความเสถียรของข้อมูลการติดตามการเคลื่อนที่ของมือได้รับการปรับปรุงอย่างมีนัยสำคัญเมื่อเราใส่ตัวกรองคาลมานช้าไป จากการทดสอบการเคลื่อนไหวของมือในลักษณะวงกลม พบว่าเมื่อข้อมูลไม่ได้ถูกประมวลผลด้วยตัวกรองคาลมานจะมีบางช่วงที่เส้นทางการเคลื่อนไหวเบี่ยงเบนไปจากแนวโน้มที่ควรจะเป็น เนื่องจากข้อมูลตำแหน่งที่ได้รับจากกล้องมีการสูญเสียการติดตามหรือกล้องไม่สามารถตรวจจับมือได้ แต่เมื่อใช้ตัวกรองคาลมาน สามารถรักษาการติดตาม (tracking) ได้อย่างต่อเนื่องและแม่นยำ

ผลการทดสอบการส่งข้อมูลระหว่างหน่วยประมวลผลข้อมูลไปยังแขนกล แสดงให้เห็นว่าทุกขั้นตอนของการส่งข้อมูลตั้งแต่หน่วยประมวลผลไปยัง HiveMQ การส่งจาก HiveMQ ไปยังแขนกล การเคลื่อนที่ของแขนกล และการเคลื่อนที่ที่สอดคล้องกับมือ ได้ผลการทดสอบเป็น "ผ่าน" ทั้งหมดในทุกครั้ง รวม 20 ครั้ง ซึ่งแสดงถึงความถูกต้องและเสถียรของการเชื่อมต่อและการส่งข้อมูลระหว่างระบบ ผลการทดสอบนี้ยืนยันว่าระบบทั้งหมดตั้งแต่การส่งข้อมูล การรับข้อมูล การประมวลผล และการตอบสนองของแขนกล ทำงานได้อย่างมีประสิทธิภาพและน่าเชื่อถือ โดยแขนกล 6 องศาอิสระรุ่น MyCobot Pi 280 สามารถเคลื่อนที่ได้ตามท่าทางที่กำหนดโดยไม่สนใจความคล้ายกันของ kinematics ของมือ แต่ว่าต้องใช้เวลาในการเคลื่อนที่และทำ path planning อย่างน้อย 2-3 วินาที ทำให้ต้องลดความถี่ในการดึงข้อมูลการเคลื่อนที่ของมือลงเป็นทุก ๆ 4 วินาทีเพื่อให้ระบบสามารถทำงานได้อย่างถูกต้อง นอกจากนี้การใช้ระบบ MQTT cloud broker ก็สามารถทำให้สามารถควบคุมหุ่นยนต์แขนกลด้วยการเคลื่อนที่ของมือจากระยะไกลได้

## 5.2 ข้อเสนอแนะ

5.2.1 ผลการทดสอบแสดงให้เห็นว่าค่าความผิดพลาดเฉลี่ยของ IMU เพิ่มขึ้นเมื่อวัดมุมที่สูงกว่า 75 องศา ควรพิจารณาใช้ IMU ที่มีความแม่นยำสูงขึ้นหรือปรับปรุงอัลกอริทึมการประมวลผลเพื่อลดความผิดพลาดในมุมสูง

5.2.2 การใช้เซ็นเซอร์หลายชนิดร่วมกัน เช่น กล้องวัดความลึกและ IMU สามารถเพิ่มความแม่นยำในการติดตามตำแหน่งและท่าทางของมือได้ การผสานข้อมูลจากหลายแหล่งอาจช่วยลดค่าความคลาดเคลื่อนและเพิ่มความน่าเชื่อถือของระบบ หากสามารถนำเซนเซอร์อื่นมาวัดค่าเพิ่มได้จะช่วยให้ความแม่นยำสูงขึ้น

5.2.3 แม้ว่าตัวกรองคาลมานจะช่วยปรับปรุงความเสถียรของการติดตามการเคลื่อนไหวได้ แต่สามารถปรับแต่งพารามิเตอร์ของตัวกรองให้เหมาะสมกับสภาพแวดล้อมการใช้งานจริงมากยิ่งขึ้นเพื่อเพิ่มความแม่นยำในการประมวลผลข้อมูล หรืออาจจะใช้ตัวกรองอื่นเข้ามาเพิ่มเติมได้

5.2.4 ควรทดสอบระบบควบคุมแขนกลในสภาวะแวดล้อมที่หลากหลาย เช่น แสงที่แตกต่างกัน พื้นผิวที่ต่างกัน และการเคลื่อนไหวของมือที่ซับซ้อนยิ่งขึ้น เพื่อประเมินความสามารถของระบบในการทำงานในสถานการณ์ที่แตกต่างกันและปรับปรุงตามข้อจำกัดที่พบ

5.2.5 ควรทดสอบกับแขนกลหลายแบบที่มีการเคลื่อนที่แตกต่างกัน และมีความรวดเร็วในการเคลื่อนที่ที่สูงขึ้นเพื่อให้สามารถตอบสนองได้ทันต่อการเคลื่อนที่ของมือ

# รายการอ้างอิง


Artronshop. (2564). *Artronshop*. เข้าถึงได้จาก ทุกเรื่องที่ควรรู้เกี่ยวกับเซอร์โวมอเตอร์และการใช้งาน: https://www.artronshop.co.th/article/92/ทุกเรื่องที่ควรรู้เกี่ยวกับเซอร์โวมอเตอร์และการใช้งาน

Balaji Gunasekaran. (11 September 2021). *Embedde Inventor*. เรียกใช้เมื่อ 22 June 2022 จาก IMU Sensors: Everything You Need To Know!: https://embeddedinventor.com/what-is-an-imu-sensor-a-complete-guide-for-beginners/

Barak Or. (31 July 2021). *Towards Data Science*. เรียกใช้เมื่อ 22 June 2022 จาก What is IMU?: https://towardsdatascience.com/what-is-imu-9565e55b44c

Charles Pao. (15 November 2018). *CEVA's Experts Blog*. เรียกใช้เมื่อ 22 June 2022 จาก What is an IMU Sensor?: https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/

Dejan. (ม.ป.ป.). *How To Mechatronics*. เรียกใช้เมื่อ 22 June 2022 จาก How to Control Servo Motors with Arduino – Complete Guide: https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

*ESP32 with MPU-6050 Accelerometer, Gyroscope and Temperature Sensor (Arduino)*. (ม.ป.ป.). เข้าถึงได้จาก Random Nerd Tutorials: https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/

Hao and Xiong, Jing and Xia, Zeyang Deng. (2017). Mobile manipulation task simulation using ROS with MoveIt. *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, (หน้า 612-616). doi:10.1109/RCAR.2017.8311930

Josh Schertz. (13 July 2017). *Josh Schertz*. เรียกใช้เมื่อ 22 June 2022 จาก Robotics Nanodegree Project 2 - Pick and Place: https://joshschertz.com/2017/07/ 13/Robotics-Nanodegree-Project-2/

Mahidzal and Tan, Jian-Ding Dahari. (2011). Forward and Inverse Kinematics Model for Robotic Welding Process Using KR-16KS KUKA Robot. *2011 Fourth International Conference on Modeling, Simulation and Applied Optimization*, (หน้า 1-6). doi:10.1109/ICMSAO.2011.5775598

Mattia and Budau Petrea, Razvan Andrei and Roberto, Oboe and Reggiani, Monica and Menegatti, Emanuele and Tagliapietra, Luca Guidolin. (2021). On the Accuracy of IMUs for Human Motion Tracking: a Comparative Evaluation. *2021 IEEE International Conference on Mechatronics (ICM)*, (หน้า 1-6). doi:10.1109/ICM46511.2021.9385684

*Moving robots into the future*. (ม.ป.ป.). เข้าถึงได้จาก MoveIt: https://moveit.ros.org/

Nitish Puri. (28 September 2017). *nitishpuri.github.io*. เรียกใช้เมื่อ 22 June 2022 จาก Inverse Kinematics on Kuka Arm using ROS and Python: https://nitishpuri.github.io/posts/robotics/inverse-kinematics-on-kuka-arm-using-ros-and-python/

Omid Sarbishei. (2016). On the Accuracy Improvement of Low-Power Orientation Filters Using IMU and MARG Sensor Arrays. *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, (หน้า 1542-1545). doi:10.1109/ISCAS.2016.7538856

Random Nerd Tutorials. (ม.ป.ป.). *Random Nerd Tutorials*. เรียกใช้เมื่อ 22 June 2022 จาก ESP32 with MPU-6050 Accelerometer, Gyroscope and Temperature Sensor (Arduino): https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/

Ren C. and Lee, Shang Lun and Wen, Yu Cheng and Hsu, Chin Hao Luo. (2020). Modular ROS Based Autonomous Mobile Industrial Robot System for Automated Intelligent Manufacturing Applications. *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, (หน้า 1673-1678). doi:10.1109/AIM43001.2020.9158800

Robert Paz. (January 2001). *ResearchGate*. เรียกใช้เมื่อ 22 June 2022 จาก The Design of the PID Controller: https://www.researchgate.net/publication/ 237528809 _The_Design_of_the_PID_Controller

Ryan Goodrich. (31 May 2018). *LiveScience*. เรียกใช้เมื่อ 22 June 2022 จาก Accelerometer vs. Gyroscope: What's the Difference?: https://www.livescience.com/40103-accelerometer-vs-gyroscope.html

Sebastian O. H. and Harrison, Andrew J. L. and Vaidyanathan, Ravi Madgwick. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. *2011 IEEE International Conference on Rehabilitation Robotics*, (หน้า 1-7). doi: 10.1109/ICORR.2011.5975346

Sergio and Maldonado-Mendez, Carolina and Marin-Hernandez, Antonio and Rios-Figueroa, Homero Vladimir and Vazquez-Leal, Hector and Palacios-Hernandez, Elvia R. Hernandez-Mendez. (2017). Design and implementation of a robotic arm using ROS and MoveIt! *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, (หน้า 1-6). doi:10.1109/ROPEC.2017.8261666

Shuaikang and Li, Zhitian and Liu, Yunfei and Zhang, Haifeng and Zou, Xudong Zheng. (2022). An Optimization-Based UWB-IMU Fusion Framework for UGV. *IEEE Sensors Journal, 22*, 4369-4377. doi:10.1109/JSEN.2022.3144660

Simone A. and Burnham, Kaleb D. Ludwig. (2018). Comparison of Euler Estimate using Extended Kalman Filter, Madgwick and Mahony on Quadcopter Flight Data. *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, (หน้า 1236-1241). doi:10.1109/ICUAS.2018.8453465

SUTHINEE KERDKAEW. (24 ธันวาคม 2021). *CNX SOFTWARE*. เรียกใช้เมื่อ 22 มิถุนายน 2022 จาก บอร์ด ESP32 UWB มีโมดูล DW1000 สำหรับระบุตำแหน่งในอาคารที่แม่นยำ: https://th.cnx-software.com/2021/12/24/บอร์ด-esp32-uwb-dw1000-indoor-positioning-แม่นยำ/

Tanmay and Kedia, Sanchit and Panchmatia, Raj and Parmar, Devaj and Sawant, Dattatray Haldankar. (2022). Design of Robotic Manipulator for Welding using ROS and Gazebo. *2022 IEEE Delhi Section Conference (DELCON)*, (หน้า 1-6). doi:10.1109/DELCON54057.2022.9753305

Tianqi and Xu, He and Liu, Ding Huang. (2021). ROS Based Obstacle Avoidance Motion Planning of UR5 Manipulator. *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, *1*, หน้า 1-6. doi:10.1109/ICNSC52481.2021.9702223

Yang Wang Xin Li. (2019). Evaluation of AHRS algorithms for Foot-Mounted Inertial-based Indoor Navigation Systems. *Open Geosciences*, 48-63. doi:10.1515/geo-2019-0005

เคนชิน (นามแฝง). (1 กรกฎาคม 2564). *ThaiWare*. เรียกใช้เมื่อ 22 มิถุนายน 2022 จาก Gyroscope กับ Accelerometer เซนเซอร์คืออะไร ? และแตกต่างกันอย่างไร ?: https://tips.thaiware. com/427.html

ฉัตรา อัศวาวุฒิ, และ โอภาส จุฑาเทพ. (2560). การรวมสัญญาณเซนเซอร์วัดแรงเฉื่อยและแสดงค่าแบบกราฟฟิ ก3 มิติ. *การประชุมนำเสนอผลงานวิจัยระดับบัณฑิตศึกษา*, 1416-1422. เรียกใช้เมื่อ 22 มิถุนายน 2022 จาก https://rsujournals.rsu.ac.th/index.php/rgrc/article/download/1181/927/

# ภาคผนวก ก

## โปรแกรมภาษาไพทอนที่ใช้สำหรับผสานรวมข้อมูลเซนเซอร์เพื่อทำการกรองสำหรับการติดตามด้วยตัวกรองคาลมาน

```
#!/usr/bin/env python3

import rospy
from nav_msgs.msg import Odometry
from sensor_msgs.msg import Imu
from geometry_msgs.msg import Pose, Quaternion
import numpy as np
import tf
import math

class KalmanNode:
    def __init__(self):
        rospy.init_node('kalman_filter_node')

        self.odom_sub = rospy.Subscriber('odom_raw', Odometry,
self.odom_callback)
        self.imu_sub = rospy.Subscriber('imu/data', Imu,
self.imu_callback)

        self.filtered_pose_pub =
rospy.Publisher('filtered_pose', Pose, queue_size=10)

        self.tf_broadcaster = tf.TransformBroadcaster()
        rospy.loginfo('Kalman Filter Node Running')

        self.dt = 1.0 / 100  # Time step
        self.std_acc = 1.0   # Standard deviation of
acceleration
        self.std_meas = 0.1  # Measurement noise standard
deviation

        self.init_kalman_filter()

    def init_kalman_filter(self):
        # State vector [x, y, z, roll, pitch, yaw, vx, vy, vz,
vroll, vpitch, vyaw]
        self.x = np.zeros((12, 1))

        # State transition matrix
        self.A = np.eye(12)
        for i in range(6):
            self.A[i, i+6] = self.dt

        # Control matrix
        self.B = np.zeros((12, 6))
        for i in range(6):
            self.B[i, i] = 0.5 * self.dt**2
            self.B[i+6, i] = self.dt

        # Measurement matrix
        self.H = np.zeros((6, 12))
        for i in range(6):
            self.H[i, i] = 1
```

```
        # Process noise covariance
        self.Q = np.eye(12) * self.std_acc**2

        # Measurement noise covariance
        self.R = np.eye(6) * self.std_meas**2

        # Error covariance matrix
        self.P = np.eye(12)

    def predict(self):
        self.x = np.dot(self.A, self.x)
        self.P = np.dot(np.dot(self.A, self.P), self.A.T) +
self.Q

    def update(self, z):
        S = np.dot(self.H, np.dot(self.P, self.H.T)) + self.R
        K = np.dot(np.dot(self.P, self.H.T), np.linalg.inv(S))
        self.x = self.x + np.dot(K, (z - np.dot(self.H,
self.x)))
        I = np.eye(12)
        self.P = (I - np.dot(K, self.H)) * self.P

    def odom_callback(self, msg):
        z = np.zeros((6, 1))
        z[0, 0] = msg.pose.pose.position.x
        z[1, 0] = msg.pose.pose.position.y
        z[2, 0] = msg.pose.pose.position.z

        orientation = msg.pose.pose.orientation
        (_, _, yaw) =
tf.transformations.euler_from_quaternion([orientation.x,
orientation.y, orientation.z, orientation.w])
        z[5, 0] = yaw

        self.predict()
        self.update(z)

        self.publish_filtered_pose()

    def imu_callback(self, msg):
        z = np.zeros((6, 1))
        orientation = msg.orientation
        (roll, pitch, _) =
tf.transformations.euler_from_quaternion([orientation.x,
orientation.y, orientation.z, orientation.w])
        z[3, 0] = roll
        z[4, 0] = pitch

        self.predict()
        self.update(z)

        self.publish_filtered_pose()
```

```
    def publish_filtered_pose(self):
        pose = Pose()
        pose.position.x = self.x[0, 0]
        pose.position.y = self.x[1, 0]
        pose.position.z = self.x[2, 0]

        q = tf.transformations.quaternion_from_euler(self.x[3,
0], self.x[4, 0], self.x[5, 0])
        pose.orientation = Quaternion(*q)

        self.filtered_pose_pub.publish(pose)

        self.tf_broadcaster.sendTransform(
            (pose.position.x, pose.position.y,
pose.position.z),
            q,
            rospy.Time.now(),
            "filtered_pose",
            "odom"
        )

def main():
    kn = KalmanNode()
    rospy.spin()

if __name__ == "__main__":
    main()
```

# ภาคผนวก ข

## โปรแกรมภาษาไพทอนที่ใช้ในการดึงข้อมูลจาก depth camera สำหรับการระบุตำแหน่งและมุม yaw

```
#!/usr/bin/env python3
import cv2
import numpy as np
import pyrealsense2 as rs
from timeit import default_timer as timer
import mediapipe as mp
import os
import math
from decimal import Decimal, ROUND_HALF_UP


import rospy
from std_msgs.msg import String
from nav_msgs.msg import Odometry
from geometry_msgs.msg import Twist,Pose2D,TransformStamped
import tf2_ros
import tf


# Initialize the RealSense pipeline
pipeline = rs.pipeline()
config = rs.config()

config = rs.config()
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

pipeline = rs.pipeline()
profile = pipeline.start(config)

align_to = rs.stream.color
align = rs.align(align_to)

intr =
profile.get_stream(rs.stream.color).as_video_stream_profile().get_int
rinsics()

accum_time = 0
curr_fps = 0
fps = "FPS: ??"
prev_time = timer()
# ...
# (Previous code remains the same)

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

hand_center  = [0.0 , 0.0]



tf_broadcaster = tf2_ros.TransformBroadcaster()
def talker():
    rospy.init_node('talker', anonymous=True)
    angle_yaw = 0.0
    Xtarget = 0.0
    Ytarget = 0.0
    Ztarget = 0.0
```

```
    pub = rospy.Publisher('chatter', String, queue_size=10)
    odom_pub = rospy.Publisher('odom_raw',Odometry, queue_size=20)

    #rate = rospy.Rate(50) # 10hz

    with mp_hands.Hands(min_detection_confidence=0.8,
min_tracking_confidence=0.5) as hands:
        while True:
            # Wait for a coherent pair of frames: depth and color
            frames = pipeline.wait_for_frames()
            aligned_frames = align.process(frames)
            color_frame = aligned_frames.get_color_frame()
            depth_frame = aligned_frames.get_depth_frame()


            if not depth_frame or not color_frame:
                continue
            depth_image = np.asanyarray(depth_frame.get_data())
            color_image = np.asanyarray(color_frame.get_data())

            frame = color_image
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            cv2.imshow("Depth Image", depth_image)

            # Flip on horizontal
            image = cv2.flip(image, 1)

            # Set flag
            image.flags.writeable = False

            # Detections
            results = hands.process(image)

            # Set flag to true
            image.flags.writeable = True

            # RGB 2 BGR
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
            lml = []
            xl = []
            yl = []

            # Rendering results
            if results.multi_hand_landmarks:

                for num, hand in
enumerate(results.multi_hand_landmarks):
                    mp_drawing.draw_landmarks(image, hand,
mp_hands.HAND_CONNECTIONS,

mp_drawing.DrawingSpec(color=(121, 22, 76), thickness=2,
circle_radius=4),

mp_drawing.DrawingSpec(color=(250, 44, 250), thickness=2,
circle_radius=2),
                                              )
                for id, lm in
enumerate(results.multi_hand_landmarks[0].landmark):
```

```
                    h, w, _ = image.shape
                    xc, yc = int(lm.x * w), int(lm.y * h)
                    lml.append([id, xc, yc])
                    xl.append(xc)
                    yl.append(yc)


                if len(lml) > 0:

                    x1, y1 = lml[9][1], lml[9][2]
                    x2, y2 = lml[12][1], lml[12][2]

                    #cv2.circle(image, (x1, y1), 10, (255, 0, 128),
cv2.FILLED)
                    #cv2.circle(image, (x2, y2), 10, (255, 0, 128),
cv2.FILLED)
                    #cv2.line(image, (x1, y1), (x2, y2), (255, 0,
128), 3)
                    distance = math.hypot(x2 - x1, y2 - y1)
                    #cv2.putText(image, str(int(distance)), (200,
200), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 128), 3)


                    #Center Finding

                    #Line1
                    line1_x1, line1_y1 = lml[0][1], lml[0][2]
                    line1_x2, line1_y2 = lml[9][1], lml[9][2]
                    m1 = 0
                    b1 = 0

                    if (line1_x2-line1_x1) != 0:
                        m1 = (line1_y2 - line1_y1)/(line1_x2-
line1_x1)
                        b1  = line1_y2 - m1 * line1_x2

                    line2_x1, line2_y1 = lml[2][1], lml[2][2]
                    line2_x2, line2_y2 = lml[17][1], lml[17][2]
                    m2 = 0
                    b2 = 0

                    if (line2_x2-line2_x1) != 0:
                        m2 = (line2_y2 - line2_y1)/(line2_x2-
line2_x1)
                        b2  = line2_y2 - m2 * line2_x2

                    x_center = 0
                    y_center = 0
                    if (m2-m1) != 0:
                        x_center = int((b2-b1) / (m1 - m2))
                        y_center = int(m1 * x_center + b1)

                        cv2.circle(image, (x_center, y_center), 10,
(0, 0, 255), cv2.FILLED)
                        #cv2.putText(image, "H", (x_center + 10,
y_center + 10), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 3)
                    print("({} , {})".format(x_center,y_center))
```

```
                    #Calculate hand Angle
                    #Ref line
                    rl_x1_p, rl_y1_p = lml[9][1], lml[9][2]
                    rl_x2_p, rl_y2_p = lml[12][1], lml[12][2]
                    rl_m = 0
                    rl_b = 0

                    rl_x1 = rl_x1_p * np.cos(1.5707) - np.sin(1.5707)
* rl_y1_p
                    rl_y1 = rl_x1_p * np.sin(1.5707) + np.cos(1.5707)
* rl_y1_p

                    rl_x2 = rl_x2_p * np.cos(1.5707) - np.sin(1.5707)
* rl_y2_p
                    rl_y2 = rl_x2_p * np.sin(1.5707) + np.cos(1.5707)
* rl_y2_p


                    if (rl_x2 - rl_x1) != 0:
                        rl_m = (rl_y2 - rl_y1)/(rl_x2 - rl_x1)
                        rl_b  = rl_y2 - rl_b * rl_x2

                        angle_raw = -1*(np.arctan2( (rl_y2 - rl_y1),
(rl_x2 - rl_x1)))

                        cv2.putText(image, str(round(angle_raw, 2)),
(x_center, y_center + 40), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255),
3)

                    hand_center[0] = max(min(abs(640 -
x_center),639), 1)    #Converting to RS frame
                    hand_center[1] = max(min(abs(y_center), 479), 1)
                    if len(hand_center) > 0:
                        print(hand_center)
                        print()

                        dist =
depth_frame.get_distance(hand_center[0], hand_center[1])*1000
#convert to mm

                        #dist =
depth_frame.get_distance(int(hand_center[0]),
int(hand_center[1]))*1000 #convert to mm

                        print(dist)
                        #depth_intrin =
depth_frame.profile.as_video_stream_profile().intrinsics
                        #depth_pixel = [hand_center[0],
hand_center[1]]
                        #dist_res =
rs.rs2_deproject_pixel_to_point(depth_intrin, depth_pixel,
depth_frame.get_distance(*depth_pixel))

                        #calculate real world coordinates

                        theta = 0
                        Xtemp = dist*(hand_center[0] -
intr.ppx)/intr.fx
```

```
                        Ytemp = dist*(hand_center[1] -
intr.ppy)/intr.fy
                        Ztemp = dist

                        Xtarget = Xtemp #- 32.5 #35 is RGB camera
module offset from the center of the realsense
                        Ytarget = -(Ztemp*math.sin(theta) +
Ytemp*math.cos(theta))
                        Ztarget = Ztemp*math.cos(theta) +
Ytemp*math.sin(theta)

                        coordinates_text = "(" +
"x:"+str(Decimal(str(Ytarget)).quantize(Decimal('0'),
rounding=ROUND_HALF_UP)) + \
                                           ", "+"y:"+
str(Decimal(str(Xtarget)).quantize(Decimal('0'),
rounding=ROUND_HALF_UP)) + \
                                           ", "+"z:"+
str(Decimal(str(Ztarget)).quantize(Decimal('0'),
rounding=ROUND_HALF_UP)) + ")"

                        print(coordinates_text)

                        font = cv2.FONT_HERSHEY_SIMPLEX
                        cv2.putText(image, text=coordinates_text,
org=(int(hand_center[0])-160, int(hand_center[1])),
                                    fontFace=font, fontScale = 1,
color=(255,255,255), thickness = 2, lineType=cv2.LINE_AA)

                        #print("x, y : " +
str(int(center_coordinates_array[i][0])))
                    angle_yaw = round(angle_raw, 2)

            quat =
tf.transformations.quaternion_from_euler(0,0,angle_yaw)

            t = TransformStamped()
            t.header.frame_id = "odom"
            t.child_frame_id= "base_link"
            t.header.stamp = rospy.Time.now()
            t.transform.translation.x = float(Ytarget/1000)
            t.transform.translation.y = float(Xtarget/1000)
            t.transform.translation.z = float(Ztarget/1000)
            t.transform.rotation.x = quat[0]
            t.transform.rotation.y = quat[1]
            t.transform.rotation.z = quat[2]
            t.transform.rotation.w = quat[3]
            #tf_broadcaster.sendTransform(t)

            #publish odometry
            odom = Odometry()
            odom.header.stamp = rospy.Time.now()
            odom.header.frame_id = "odom"
            odom.child_frame_id = "base_link"
            #Add pose to odom
            odom.pose.pose.position.x = float(Ytarget/1000)
            odom.pose.pose.position.y = float(Xtarget/1000)
            odom.pose.pose.position.z = float(Ztarget/1000)
```

```
                odom.pose.pose.orientation.x = quat[0]
                odom.pose.pose.orientation.y = quat[1]
                odom.pose.pose.orientation.z = quat[2]
                odom.pose.pose.orientation.w = quat[3]

                odom.pose.covariance[0] = 0.0001
                odom.pose.covariance[7] = 0.0001
                odom.pose.covariance[35] = 0.01
                odom.twist.covariance[0] = 0.00001
                odom.twist.covariance[7] = 0.00001
                odom.twist.covariance[35] = 0.01

                odom_pub.publish(odom)


                cv2.imshow("Coordinate visualizer", image)

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
cv2.destroyAllWindows()
```

# ภาคผนวก ค

## โปรแกรมภาษาไพทอนที่ใช้ในการส่งข่อมูลจาก Kalman filter ไป HiveMQ

```
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Pose
import paho.mqtt.client as mqtt
import json

class MqttPublisherNode:
    def __init__(self):
        rospy.init_node('mqtt_publisher_node')

        # Subscribe to the filtered_pose topic
        self.pose_sub = rospy.Subscriber('filtered_pose',
Pose, self.pose_callback)

        # MQTT setup
        self.mqtt_client = mqtt.Client()
        self.mqtt_client.username_pw_set("tnp_iot",
"tnp_iot01")
        self.mqtt_client.connect("3d3c0f6a6e27435d94068590434a
9ba7.s1.eu.hivemq.cloud", 1883, 60)  # Connect to HiveMQ
broker

        rospy.loginfo("MQTT Publisher Node Running")

    def pose_callback(self, msg):
        pose_data = {
            "position": {
                "x": msg.position.x,
                "y": msg.position.y,
                "z": msg.position.z
            },
            "orientation": {
                "x": msg.orientation.x,
                "y": msg.orientation.y,
                "z": msg.orientation.z,
                "w": msg.orientation.w
            }
        }

        # Convert the pose data to JSON
        pose_json = json.dumps(pose_data)

        # Publish the pose data to the MQTT broker
        self.mqtt_client.publish("cobot/filtered_pose",
pose_json)
        rospy.loginfo("Published filtered pose to MQTT: %s",
pose_json)

    def spin(self):
        rospy.spin()
        self.mqtt_client.loop_start()  # Start the MQTT loop
```

```
def main():
    node = MqttPublisherNode()
    node.spin()

if __name__ == "__main__":
    main()
```

# ภาคผนวก ง

## โปรแกรมภาษาไพทอนที่ใช้ในการรับข้อมูลจาก HiveMQ ไปควบคุมแขนกล

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Pose
from mycobot_ros.srv import SetJointAngles,
SetJointAnglesRequest
import paho.mqtt.client as mqtt
import json
import math

class MyPiCobotController:
    def __init__(self):
        rospy.init_node('mypicobot_controller_node')

        # MQTT setup
        self.mqtt_client = mqtt.Client()
        self.mqtt_client.username_pw_set("tnp_iot",
"tnp_iot01")
        self.mqtt_client.on_connect = self.on_connect
        self.mqtt_client.on_message = self.on_message

self.mqtt_client.connect("3d3c0f6a6e27435d94068590434a9ba7.s
1.eu.hivemq.cloud", 1883, 60)

        self.pose = None
        self.control_rate = rospy.Rate(10)  # 10 Hz control
loop

        rospy.wait_for_service('/mycobot/set_joint_angles')
        self.set_joint_angles_service =
rospy.ServiceProxy('/mycobot/set_joint_angles',
SetJointAngles)

        rospy.loginfo("MyPiCobot Controller Node Running")
        self.mqtt_client.loop_start()

    def on_connect(self, client, userdata, flags, rc):
        rospy.loginfo("Connected to MQTT broker with result
code " + str(rc))
        client.subscribe("cobot/filtered_pose")

    def on_message(self, client, userdata, msg):
        try:
            pose_data = json.loads(msg.payload.decode())
            pose = Pose()
            pose.position.x = pose_data["position"]["x"]
            pose.position.y = pose_data["position"]["y"]
            pose.position.z = pose_data["position"]["z"]
            pose.orientation.x =
pose_data["orientation"]["x"]
            pose.orientation.y =
pose_data["orientation"]["y"]
            pose.orientation.z =
pose_data["orientation"]["z"]
```

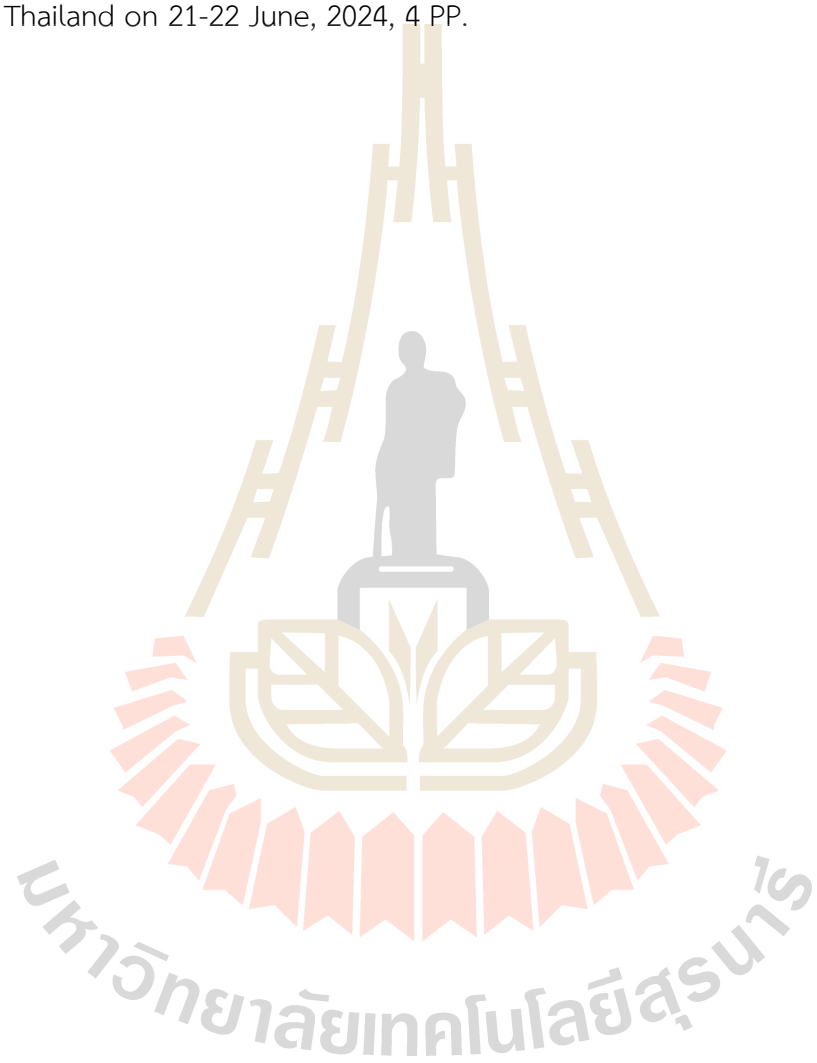
```
            pose.orientation.w =
pose_data["orientation"]["w"]
            self.pose = pose
        except Exception as e:
            rospy.logerr("Error processing message: %s",
str(e))

    def pose_to_joint_angles(self, pose):
              joint_angles = [0] * 6
        joint_angles[0] = math.atan2(pose.position.y,
pose.position.x)
        joint_angles[1] = math.atan2(pose.position.z,
math.sqrt(pose.position.x**2 + pose.position.y**2))
        joint_angles[2] = 0
        joint_angles[3] = 0
        joint_angles[4] = 0
        joint_angles[5] = 0
        return joint_angles

    def control_loop(self):
        while not rospy.is_shutdown():
            if self.pose:
                try:
                    joint_angles =
self.pose_to_joint_angles(self.pose)
                    request = SetJointAnglesRequest()
                    request.angles = joint_angles
                    self.set_joint_angles_service(request)
                    rospy.loginfo("Sent joint angles to
MyPiCobot: %s", joint_angles)
                except rospy.ServiceException as e:
                    rospy.logerr("Service call failed: %s",
str(e))
            self.control_rate.sleep()

def main():
    controller = MyPiCobotController()
    controller.control_loop()

if __name__ == "__main__":
    main()
```

# ภาคผนวก จ

## บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างการศึกษา

## รายชื่อบทความวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในขณะศึกษา

Tannaapat jurachokphuriwaj, Jittima varakul (2024) ACCURATE HAND POSE ESTIMATION USING DEPTH CAMERA AND IMU, International Conference on Robotics, Machine Learning and Artificial Intelligence (ICRMLAI) in Bangkok Thailand on 21-22 June, 2024, 4 PP.



SA-MLAI-BNKK-200624-4912

SARC
South Asian Research Center

# SOUTH ASIAN RESEARCH CENTER

## International Conference on

Robotics, Machine Learning and Artificial Intelligence

# CERTIFICATE

*of*

## PRESENTATION

*Presented to*

Tannapat Jaruchokphuriwaj

*for presenting a paper entitled "Accurate Hand Pose Estimation Using Depth Camera and IMU" at the International Conference on Robotics, Machine Learning and Artificial Intelligence(ICRMLAI) held in Bangkok, Thailand on 21st - 22nd June, 2024.*

Conference Coordinator
South Asian Research Center

Chairman
South Asian Research Center

www.sarc.net.in | sarc.net.in@gmail.com

# ACCURATE HAND POSE ESTIMATION USING DEPTH CAMERA AND IMU

[1]TANNAPAT JARUCHOKPHURIWAJ, [2]JITTIMA VARAGUL

[1]School of Mechatronics Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand
[2]School of Manufacturing Automation and Robotics Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand

Email: [1]tannapat.tnp@gmail.com, [2]jittima@sut.ac.th

**Abstract:**. This paper presents a novel hand pose estimation system that integrates depth camera and inertial measurement unit (IMU) data to achieve high accuracy in detecting and tracking hand movements. The system leverages the Intel RealSense D435 depth camera and the Witmotion JY61P IMU sensor for precise hand position and orientation estimation. Hand detection is performed using Google's MediaPipe model, which identifies reference points on the hand from RGB and depth images, while the IMU provides additional pitch and roll angle measurements. Experimental results demonstrate the system's accuracy, with average positional errors of 2.33 mm and yaw angle errors of 0.057 degrees from the depth camera, and roll and pitch angle errors of 0.45 degrees and 0.34 degrees respectively from the IMU.

***Index terms:*** Hand pose estimation, Depth Camera, Inertial Measurement Unit.

## I. INTRODUCTION

Hand pose estimation is a critical component in the development of advanced human-computer interaction (HCI) systems. It serves as a foundational technology for numerous applications, including virtual reality, augmented reality, robotics, and assistive technologies. Accurate and real-time hand pose estimation enables intuitive and natural interaction between humans and machines, thereby enhancing user experience and expanding the capabilities of interactive systems.

In recent years, several methods have been proposed to tackle the challenges of hand pose estimation. One study demonstrated the feasibility of implementing hand pose recognition systems on embedded computers using artificial intelligence, highlighting the importance of efficient algorithms for real-time applications [1]. Similarly, robust hand tracking has been achieved by employing the Kinect sensor in conjunction with the Kalman filter, emphasizing the integration of sensor data for improved accuracy [2].

Hand tracking based on stereo cameras has also been explored extensively. Utilization of stereo camera setups for motion control in robotic arms has demonstrated the potential for precise hand movement capture in dynamic environments [3]. Another study explored human hand tracking using MATLAB to control Arduino-based robotic arms, showcasing the versatility of different programming environments in developing hand tracking systems [4].

The advancements in hand gesture recognition have paved the way for innovative applications such as learning-aid tools and HCI teaching systems. Development of a learning-aid tool using hand gesture-based interaction highlights the educational potential of hand tracking technologies [5]. Additionally, a human-computer interaction teaching system based on hand tracking has been designed, demonstrating the application of these technologies in educational contexts [6].

Recent developments have also seen the integration of deep learning techniques and marker-based enhancements for hand tracking. A marker-enhanced hand tracking method using deep learning significantly improves tracking accuracy and robustness [7]. Moreover, real-time hand tracking approaches using optical flow and CAMshift algorithms have been proposed, illustrating the effectiveness of combining traditional computer vision techniques with modern machine learning methods [8].

The combination of depth cameras and inertial measurement units (IMUs) offers a promising approach for hand pose estimation. Depth cameras provide rich spatial information, while IMUs contribute temporal motion data. In this paper, we propose a novel hand pose estimation system that leverages the capabilities of depth cameras and IMUs. By integrating these technologies, we aim to achieve high accuracy data. Our approach combines the sensor technology with state-of-the-art machine learning algorithms to provide a robust and efficient solution for hand pose estimation.

## II. SYSTEM OVERVIEW

The proposed system integrates input data from a depth camera and an inertial measurement unit (IMU). These data sources are fused using an Extended Kalman Filter to estimate the hand's position and orientation accurately. The system comprises a wearable device and an external device, both capable of communication via computer interfaces. Specifically, the depth camera utilized is the Intel RealSense D435, while the IMU sensor employed is the Witmotion JY61P, which incorporates the MPU6050 component for measuring angular velocity and linear acceleration.

The system's operation commences with the capture of RGB images from the depth camera to detect and locate reference points on the hand using Google's MediaPipe machine learning model. This model identifies the reference points of each joint in the hand. The results from this initial step are then used to calculate the real-world positions by correlating them with the depth map generated by the depth camera. Additionally, the RGB camera images aid in determining the hand's rotational angle within the 2D plane of the image, thereby providing the yaw angle. Subsequently, data from the gyroscope and accelerometer sensors are filtered to ascertain the orientation of the IMU. The IMU is attached to the wearable device on the hand as in Figure. 1, enabling the system to capture comprehensive orientation data of the hand. This data is then fused.

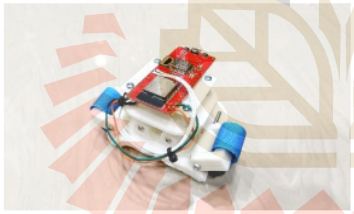


**Figure 1: Wearable device for hand.**

## III. DEPTH CAMERA BASED HAND DETECTION AND POSE ESTIATION

Using a machine learning model, MediaPipe can detect hands and create a skeleton of 20 hand joint points, as shown in Figure. 2. The reference point for hand positioning, $O(x, y)$, is determined by the intersection of lines drawn from point 1 to point 17 and from point 0 to point 9. This coordinate is used to find the real-world position in conjunction with the depth image. The depth image provides the distance from the object to the camera.

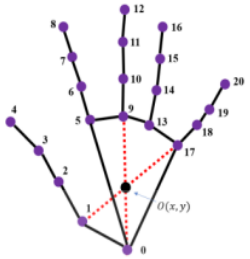



**Figure2: Hand Reference Joint in MediaPipe.**

To obtain the x and y coordinates of the reference point, the pixel coordinates of the reference point and the measured distance need to be calculated using the principal point, an intrinsic parameter of the camera that determines the optical center point of the camera, as shown in Equation 1 and Equation 2. Typically, the principal point is close to the image center.

$$x = D(O_x - P_{p_x})/f_x \quad (1)$$

$$y = D(O_y - P_{p_y})/f_y \quad (2)$$

Where $D$ is the distance from camera to hand reference point. $P_{p_x}$ and $P_{p_y}$ are the principal point in x and y axes, respectively. $f_x$ and $f_y$ are the focal length of the camera. To determine the orientation of the hand, a reference line is drawn between points 9 and 10. The angle is considered 0 degrees when this line is perpendicular to the horizontal plane of the image. Angles increase positively when the line is rotated counterclockwise from this initial position and decrease negatively when rotated clockwise.

## IV. IMU BASED HAND ORIENTATION ESTIMATION

Beyond merely estimating the yaw angle of the hand via depth camera imagery, it is imperative to also ascertain the pitch and roll angles. To achieve this, we advocate the employment of an Inertial Measurement Unit (IMU). This IMU leverages data from both a gyroscope and an accelerometer, subsequently processed through a Kalman filter to precisely deduce the hand's orientation. The IMU is ingeniously integrated into a glove, functioning as a wearable device, as illustrated in Figure. 1. The uppermost layer of the wearable device incorporates an ESP32 microcontroller board, which interfaces with the IMU and facilitates data transmission to a computer. The IMU is strategically positioned between the back of the hand and the ESP32, with its X-axis aligned with the fingers' direction and its Z-axis oriented perpendicular to the palm. The wearable device is fastened to the hand using straps, meticulously designed to minimize any obstruction of the palm's surface area.

## V. EXPERIMENTAL RESULTS

To test the accuracy of detecting the position and orientation of the hand using a depth camera and IMU, we conduct detection tests and compare the results with actual positions. In the depth camera tests, we measure position error and yaw angle error. The position error is measured by comparing the hand's position on the same plane within a test area not exceeding 40x40 cm. The angle error is measured against a 3D printed part with fixed, clearly defined angles. When the hand is placed on the designated jig, the desired hand angle is achieved and then compared with the read values.

For testing the orientation obtained from the IMU, a jig created from a 3D printer is used, incrementally set at angles from 0 to 90 degrees in 15-degree increments. The IMU is placed at the designated angles, and the read values are compared with the sensor data to calculate the resulting error.

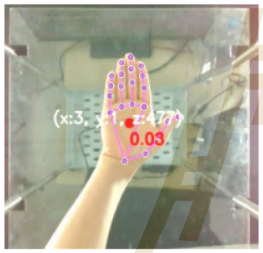



**Figure 3: Example of hand position test.**

The computational setup for running the model comprises an Intel Core i5 CPU paired with a GeForce GTX 1650 GPU. Hand detection within both RGB and depth images is performed at a resolution of 640x480 pixels. Illustrative examples of hand detection in testing for position and orientation evaluation are presented in Figure.3 and Figure.4, respectively.


**Figure 4: Example of hand yaw angle test.**

The data presented in Table 1 shows the error in position estimation using the depth camera. The average positional errors are relatively low, with 2.44 mm in the X-axis, 2.22 mm in the Y-axis, and 2.33 mm in the Z-axis. The yaw angle error averages to 0.057 degrees. The minimal positional error recorded is 0.00 mm across all axes, indicating instances of highly accurate detection. However, the maximum positional errors recorded are 4.00 mm, 5.00 mm, and 5.00 mm for the X, Y, and Z axes respectively, with a yaw angle error peaking at 0.110 degrees. These variations, particularly the maximum values, suggest potential challenges in the depth camera's resolution or calibration, especially at extreme values.

Table 1
ERROR OF DETECTION FROM DEPTH CAMERA

| Error of estimation from depth camera | | | | |
|---|---|---|---|---|
| Value | X [mm] | Y[mm] | Z[mm] | Yaw Angle [deg] |
| Average | 2.44 | 2.22 | 2.33 | 0.057 |
| Min | 0.00 | 0.00 | 0.00 | 0.030 |
| Max | 4.00 | 5.00 | 5.00 | 0.110 |

Table 2 illustrates the error in roll and pitch angle estimations using the IMU sensor. The average errors for roll and pitch are 0.45 degrees and 0.34 degrees, respectively, reflecting a high degree of accuracy in the IMU's measurements. The minimum errors recorded are 0.11 degrees for roll and a remarkably low 0.01 degrees for pitch. Conversely, the maximum errors observed are significantly higher, at 3.94 degrees for roll and 3.92 degrees for pitch. These maximum error values, particularly at higher angles, highlight potential issues with the IMU's sensor resolution or calibration effectiveness, suggesting that the sensor's performance may degrade at extreme angles.

Table 2
ERROR OF DETECTION FROM IMU SENSOR

| Error of estimation from IMU sensor | | |
|---|---|---|
| Value | Roll[deg] | Pitch[deg] |
| Average | 0.45 | 0.34 |
| Min | 0.11 | 0.01 |
| Max | 3.94 | 3.92 |

## CONCLUSION

The proposed hand pose estimation system effectively combines the capabilities of depth cameras and IMUs, demonstrating high accuracy in detecting hand positions and orientations. The integration of these sensors, processed through an Extended Kalman Filter, results in robust performance suitable for real-time applications. The experimental results indicate minimal average errors, with the depth camera achieving positional errors of approximately 2.33 mm and yaw angle errors of 0.057 degrees, and the IMU achieving roll and pitch errors of 0.45 degrees and 0.34 degrees, respectively. Nevertheless, the system's performance at extreme angles reveals areas for improvement, particularly in sensor resolution and calibration. Future work will focus on addressing these limitations to further enhance the accuracy and reliability of hand pose estimation in dynamic environments.

## REFERENCES

[1] D. Fernández, "Development of a hand pose recognition system on an embedded computer using Artificial Intelligence," *in Proc. IEEE XXVI International Conference On Electronics, Electrical Engineering And Computing (INTERCON)*, 2019, pp. 1-4.

[2] L. Yang, M. Wang, and T. Li, "Hand Tracking Based on the Kinect and Kalman Filter," in *Proc. IEEE 3rd International Conference On Information Systems And Computer Aided Education (ICISCAE)*, 2020, pp. 708-714.

[3] B. Sugandi, H. Toar, and A. Alfianto, "Hand tracking-based motion control for robot arm using stereo camera," in *Proc. International Conference On Applied Engineering (ICAE)*, 2018, pp. 1-5.

[4] P. Lengare and M. Rane, "Human hand tracking using MATLAB to control Arduino based robotic arm," in *Proc. International Conference On Pervasive Computing (ICPC)*, 2015, pp. 1-4.

[5] B. Boruah, A. Talukdar, and K. Sarma, "Development of a learning-aid tool using hand gesture based human computer interaction system," in *Proc. Advanced Communication Technologies And Signal Processing (ACTS)*, 2021, pp. 1-5.

[6] K. Naganandhini, B. Sowmya, S. Pavish, J. Nitesh, R. Karthika, and E. Prabhu, "Hand Tracking Based Human-Computer Interaction Teaching System," in *Proc. 3rd International Conference On Issues And Challenges In Intelligent Computing Techniques (ICICT)*, 2022, pp. 1-5.

[7] D. Yang, X. Cao, Y. Liu, and S. Xu, "Marker-enhanced hand tracking with deep learning," in *Proc. International Conference On Virtual Reality, Human-Computer Interaction And Artificial Intelligence (VRHCIAI)*, 2022, pp. 37-42.

[8] U. Soni, A. Trivedi, and N. Roberts, "Real-time hand tracking using integrated optical flow and CAMshift algorithm," in *Proc. Second International Conference On Research In Computational Intelligence And Communication Networks (ICRCICN)*, 2016, pp. 135-140.

★★★

# ประวัติผู้เขียน

นางสาวธัญญ์นภัส จารุโชคภูริวัจน์ เกิดเมื่อวันที่ 17 มกราคม พ.ศ.2541 ที่อำเภอเมือง จังหวัดนครราชสีมา เริ่มศึกษาชั้นประถมศึกษาที่โรงเรียนอนุบาลชัยภูมิ อำเภอเมือง จังหวัดชัยภูมิ ระดับมัธยมศึกษาตอนต้นและตอนปลายที่โรงเรียนสตรีชัยภูมิ อำเภอเมือง จังหวัดชัยภูมิ และสำเร็จการศึกษาวิศวกรรมศาตรบัณฑิต สาขาวิศวกรรมอากาศยาน สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา เมื่อปี พ.ศ.2562 และเข้าศึกษาต่อในระดับวิศวกรรมศาสตรมหาบัณฑิต หลักสูตรวิศวกรรมเมคคาทรอนิกส์ ณ สถาบันการศึกษาเดิม