การประเมินตำแหน่งด้วยกล้อง KINECT และการประยุกต์ใช้ในหุ่นยนต์

DISTANCE MEASUREMENT WITH KINECT CAMERA AND ITS APPLICATIONS IN MOBILE ROBOTS

ณรงค์ชัย จันทร์สมมิตร
เดชาธร โรจนอนันต์
ธนทัต วุฒิคุณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

การประเมินตำแหน่งด้วยกล้อง KINECT และการประยุกต์ใช้ในหุ่นยนต์

DISTANCE MEASUREMENT WITH KINECT CAMERA AND ITS APPLICATIONS IN MOBILE ROBOTS

ณรงค์ชัย จันทร์สมมิตร

เดชาธร โรจนอนันต์

ธนทัต วุฒิคุณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

DISTANCE MEASUREMENT WITH KINECT CAMERA AND ITS APPLICATIONS IN MOBILE ROBOTS

NARONGCHAI JUNSOMMIT
DEACHATORN ROJJANAANAN
TANATAT WUTIKUN

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2014

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประเมินตำแหน่งด้วยกล้อง KINECT และการประยุกต์ใช้ในหุ่นยนต์
DISTANCE MEASUREMENT WITH KINECT CAMERA AND ITS APPLICATIONS IN MOBILE ROBOT

ผู้จัดทำ นายณรงค์ชัย จันทร์สมมิตร 54010370
นายเดชาธร โรจนอนันต์ 54010478
นายธนทัต วุฒิคุณ 54010537

.....อาจารย์ที่ปรึกษา
(ดร.รัชนี กุลยานนท์)

การประเมินค่าตำแหน่งด้วยกล้อง KINECT และการประยุกต์ใช้ในหุ่นยนต์

โดย

นายณรงค์ชัย	จันทร์สมมิตร	54010370
นายเดชาธร	โรจนอนันต์	54010478
นายธนทัต	วุฒิคุณ	54010537

อาจารย์ที่ปรึกษา

ดร.รัชนี กุลยานนท์

ปีการศึกษา 2557

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการประยุกต์ใช้ของกล้อง Kinect เพื่อให้หุ่นยนต์ช่วยเหลือผู้สูงอายุ โดยโครงสร้างของระบบนั้นประกอบด้วยโปรแกรมควบคุมแขนกล กับโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่ จุดมุ่งหมายของการทำโครงงานนี้คือ การสร้างและออกแบบโปรแกรมให้สามารถควบคุมหุ่นยนต์เพื่อช่วยเหลือผู้สูงอายุในการติดตามและหยิบจับวัตถุสิ่งของได้ สำหรับขั้นตอนการดำเนินการนั้นเริ่มจากการศึกษาจลนศาสตร์ของแขนกล ช่วยในการคำนวณสมการของแขนกล หลังจากนั้นจึงศึกษาการทำงานของกล้อง Kinect เพื่อตรวจร่างกายของมนุษย์

ผลการทดลองสามารถสรุปได้ว่า โปรแกรมควบคุมแขนกลในโหมด V มีความคลาดเคลื่อนในการเคลื่อนที่ไปยังตำแหน่งของแกน X และแกน Z โดยเฉลี่ย เท่ากับ 0.509 cm และ 0.622 cm และในโหมด H มีความคลาดเคลื่อนในการเคลื่อนที่ไปยังตำแหน่งของแกน X และแกน Z โดยเฉลี่ย เท่ากับ 0.305 cm และ 0.040 cm ส่วนของโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่มีความคลาดเคลื่อนของการวัดระยะเท่ากับ 3.615 cm และโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่ และหุ่นยนต์สามารถติดตามบุคคล 1 ท่าน ได้อย่างแม่นยำ

DISTANCE MEASUREMENT WITH KINECT CAMERA AND ITS APPLICATIONS IN MOBILE ROBOTS

By

Mr.Narongchai Junsommit 54010370

Mr.Deachatorn Rojjanaanan 54010478

Mr.Tanatat Wutikun 54010537

Advisor

Dr.Rutchanee Gullayanon

Academic 2014

ABSTRACT

This thesis The application of the Kinect camera to robotic helping the elderly. The structure of the system consists of a driver arm. With image processing and motion control applications. The aim of this project is. Create and design To control robots to assist the elderly in tracking objects and handling things. For the process of doing it began to study the kinematics of the robot arm. Help to calculate the equation of the mechanical arm. After studying the work of the Kinect camera to detect the human body.

The results conclude that. Driver Robot Mode V has a deviation of mobility to the position of the X-axis and Z-axis average of 0.509 cm and 0.622 cm, and the H mode is a deviation of mobility to the position of the X and Z axis by. average of 0.305 cm and 0.040 cm of the application of image processing and motion control tolerances of measurement equal to 3.615 cm and application of image processing and motion control. And the robot can follow one person accurately.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา ดร.รัชนี กุลยานนท์ เป็นอย่างสูงที่คอยให้คำปรึกษาแนะนำวิธีการแก้ปัญหาต่างๆ ในโครงงานทั้งทางทฤษฎี และทางปฏิบัติแก่คณะผู้จัดทำมาโดยตลอด ทำให้ผู้จัดทำเข้าใจถึงที่มาของปัญหาและสามารถแก้ไขปัญหาต่างๆ ได้อย่างถูกวิธี รวมทั้งยังเอื้อเฟื้ออุปกรณ์ที่จำเป็น และความช่วยเหลืออื่นๆ ที่เป็นประโยชน์ต่อโครงงาน

ขอขอบพระคุณ รุ่นพี่และรุ่นน้องภาควิชาวิศวกรรมการวัดและควบคุม ที่ช่วยให้คำปรึกษาและช่วยแก้ไขในส่วนที่คณะผู้จัดทำยังไม่มีความเข้าใจเป็นอย่างดี

ขอขอบพระคุณ อาจารย์สองเมือง นันทขว้าง และทีม ที่ได้เอื้อเฟื้อหุ่นยนต์ ให้แก่คณะผู้จัดทำ

ขอขอบคุณเพื่อนๆ ทุกคน ที่คอยให้กำลังใจสนับสนุนอุปกรณ์ที่ขาดเหลือกระตุ้นเตือนรวมทั้งคอยถามไถ่ความคืบหน้าของโครงงานอยู่เสมอ

สุดท้ายนี้คณะผู้จัดทำขอขอบพระคุณบิดา มารดา และครอบครัว ที่เป็นกำลังใจที่ดีตลอดมา รวมถึงการสนับสนุนในเรื่องของงบประมาณที่ขาดเหลือ ตลอดจนเป็นแรงบันดาลใจที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ได้

ผู้จัดทำ
ณรงค์ชัย จันทร์สมมิตร
เดชาธร โรจนอนันต์
ธนทัต วุฒิคุณ

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันนี้มีผู้สูงอายุจำนวนมากขึ้น ซึ่งจำเป็นต้องมีผู้ดูแลอยู่ใกล้ชิด แต่เนื่องจากหลากหลายปัจจัยที่ในบางครั้งผู้สูงอายุต้องอยู่เพียงลำพัง อาจทำให้ผู้สูงอายุเหล่านี้ที่ไม่ได้มีคนดูแลได้อย่างทั่วถึงหรือตลอดเวลา ในบางครั้งอาจทำให้เกิดอันตรายกับบุคคลเหล่านี้ จากการทำกิจวัตรประจำวันต่างๆ ทางคณะผู้จัดทำโครงงานนี้ ได้ตระหนักถึงปัญหาเหล่านี้ จึงได้มีแนวคิดที่จะพัฒนาหุ่นยนต์ซึ่งจะสามารถช่วยอำนวยความสะดวกให้แก่คนชรา ในยามที่ไม่มีบุคคลดูแล ทำให้สามารถดำเนินชีวิตได้อย่างสะดวกสบายยิ่งขึ้น

โครงงานนี้จึงได้นำหลักการของการตรวจจับตำแหน่งที่ตั้งบุคคลของกล้อง Kinect เพื่อทำเป็นโปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์ และค่าตำแหน่งที่ตั้งที่ได้จากกล้อง Kinect จะสามารถนำไปเข้าสู่โปรแกรมควบคุมแขนกล โดยใช้หลักการของจลนศาสตร์แขนกล เพื่อสั่งการให้แขนกลสามารถเคลื่อนที่ไปหยิบจับวัตถุจากมือผู้ใช้ได้อีกด้วย

1.2 วัตถุประสงค์

1. เพื่อศึกษาการใช้งานกล้อง Kinect ในการตรวจจับระยะห่างระหว่างมนุษย์กับกล้อง
2. เพื่อศึกษาวิธีการตรวจจับมนุษย์ของกล้อง Kinect
3. เพื่อศึกษาการคำนวณจลนศาสตร์ของแขนกล
4. เพื่อออกแบบระบบควบคุมแขนกลและขับเคลื่อนหุ่นยนต์ไปยังพิกัดที่ต้องการ

1.3 ขอบเขตการศึกษา

1. ใช้วิธีการตรวจจับมนุษย์ด้วยการตรวจจับกระดูก
2. ใช้บุคคล 1 ท่าน ในการทดสอบการวัดระยะห่างของมนุษย์กับกล้อง Kinect
3. ใช้ระยะห่างที่ได้จากกล้อง Kinect มาขับเคลื่อนหุ่นยนต์
4. ใช้แขนกลและระบบขับเคลื่อนของหุ่นยนต์ Cira ในการทดสอบโปรแกรม

1.4 ประโยชน์ที่ได้รับ

1. ได้ระบบตรวจจับมนุษย์ตามเวลาจริง
2. พัฒนาระบบแขนกลและขับเคลื่อนของหุ่นยนต์ให้ไปยังเป้าหมาย
3. หุ่นยนต์สามารถติดตามผู้ใช้ 1 ท่าน
4. พัฒนาให้หุ่นยนต์สามารถช่วยเหลือผู้สูงอายุในซูเปอร์มาร์เก็ตต่อไป

1.5 ขั้นตอนการดำเนินงานวิจัย

1. ศึกษาทฤษฎีที่เกี่ยวข้องกับจลนศาสตร์ของแขนกล
2. ศึกษาการใช้งานกล้อง Kinect ในการตรวจจับโครงร่างมนุษย์
3. เสนอวิธีตรวจจับโครงร่างของมนุษย์โดยใช้กล้อง Kinect
4. พัฒนาโปรแกรมตรวจจับตำแหน่งกลางอกของมนุษย์โดยกล้อง Kinect
5. พัฒนาโปรแกรมคำนวณสมการจลนศาสตร์ของแขนกล
6. พัฒนาโปรแกรมควบคุมแขนกล
7. พัฒนาโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่ของหุ่นยนต์
8. แสดงผลการทดสอบความแม่นยำของระยะที่ตรวจจับได้จากกล้อง Kinect
9. สรุปผล

1.6 รายละเอียดของปริญญานิพนธ์

ในปริญญานิพนธ์ฉบับนี้ประกอบไปด้วย 5 บท โดยมีรายละเอียดของแต่ละบทดังต่อไปนี้

บทที่ 1 จะกล่าวนำบอกถึงวัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน ประโยชน์ที่ได้รับจากการทำโครงงานนี้

บทที่ 2 จะอธิบายเกี่ยวกับทฤษฎีจลนศาสตร์ของแขนกล หลักการทำงานของกล้อง Kinect และการเชื่อมต่อกับกล้อง Kinect

บทที่ 3 จะอธิบายถึงโครงสร้างและส่วนประกอบของแขนกล การคำนวณสมการจลนศาสตร์ของแขนกล โปรแกรมควบคุมแขนกล และโปรแกรมประมวลผลภาพด้วยกล้อง Kinect

บทที่ 4 จะเป็นส่วนผลที่ได้จากการทดลองและวิเคราะห์ผล ซึ่งประกอบด้วยพารามิเตอร์ต่างๆ ของแขนกลและกล้อง Kinect ผลทดลองจากหุ่นยนต์

บทที่ 5 เป็นบทสุดท้าย จะสรุปรวมและวิเคราะห์ปัญหาที่พบ ตลอดจนแนวทางการแก้ไขต่อไป

บทที่ 2

ทฤษฎี และความรู้ที่เกี่ยวข้อง

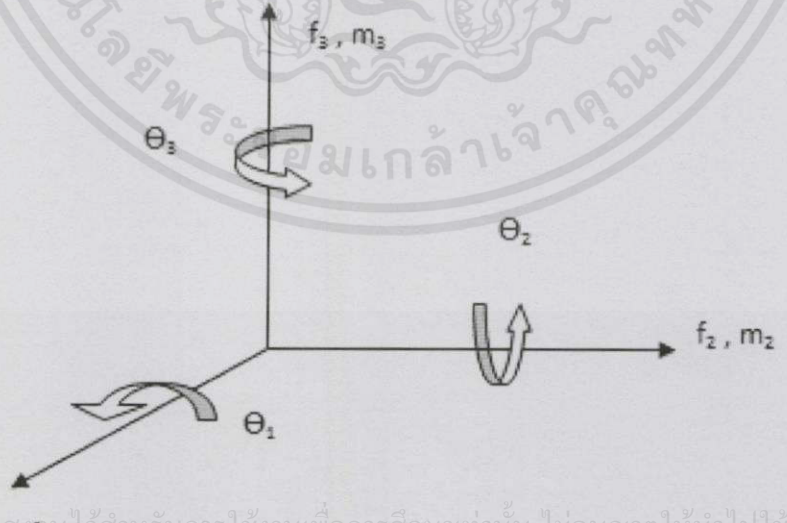
จากที่ได้กล่าวในบทที่ 1 แล้วว่าก่อนที่จะมีการออกแบบและสร้างระบบควบคุมแขนกลและขับเคลื่อนหุ่นยนต์นั้น จำเป็นต้องศึกษาสมการจลนศาสตร์ของแขนกล และหลักในการทำงานของกล้อง Kinect เพื่อให้สามารถออกแบบระบบให้ได้มีประสิทธิภาพสูงสุด ซึ่งข้อมูลและทฤษฎีที่เกี่ยวข้องกับปริญญานิพนธ์นี้มีดังนี้

2.1 จลนศาสตร์ข้างหน้าของหุ่นยนต์ (Forward Kinematics Analysis)

เครื่องมือที่ใช้บรรยายตำแหน่งและทิศทางการหมุนในระนาบ 3 มิติ จะถูกนำมาประยุกต์ใช้กับหุ่นยนต์ โดยการติดตั้งเฟรมบนข้อต่อของหุ่นยนต์ ตามหลักการของเดนาวิทและฮาร์เทนเบิร์ก (Denavit-Hatenberg Convention) ตำแหน่งและการหมุนเหล่านี้จะมีความสัมพันธ์กับพารามิเตอร์ของข้อต่อ (Joint Parameter) ด้วยการแปลงสัมพัทธ์เทียบกับเฟรมของข้อต่อก่อนหน้าไปเรื่อยๆ จะทำให้ทราบตำแหน่งและทิศทางการหมุนของเฟรมที่ติดกับส่วนปลายแขน รวมถึงท่าทาง (Posture) ของหุ่นยนต์ในฟังก์ชันพารามิเตอร์ของข้อต่อ การวิเคราะห์นี้เรียกว่า จลนศาสตร์ข้างหน้าของหุ่นยนต์ (Forward Kinematics Analysis)

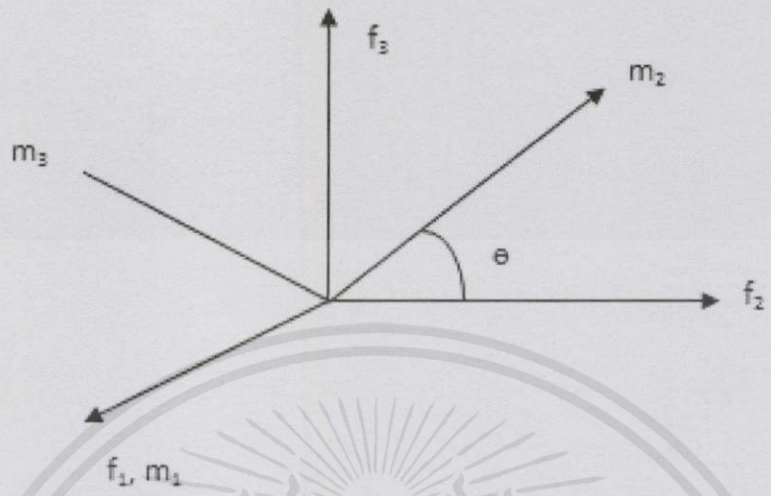
2.1.1 สมการการหมุน

ถ้าเฟรมของหุ่นยนต์ m คือ m_1, m_2, m_3 และเฟรมของฐาน (Base) f คือ f_1, f_2, f_3 เป็นเฟรมที่ทับกันโดยมีทิศทางการหมุนเป็น θ แสดงดังรูปที่ 2.1



รูปที่ 2.1 พื้นฐานการหมุนในระนาบ 3 มิติ

เมื่อหมุนเฟรมแกน m แกนใดแกนหนึ่ง ผลของการหมุนแกนใดๆ ของเฟรม m จะได้สมการการหมุน (Rotation Matrix) ในรูปที่ 2.2 แสดงการหมุนแกน m_1



รูปที่ 2.2 การหมุนของแกน m_1

ผลของการหมุนแกน m_1 ได้ดังสมการที่ (2.1)

$$R_x(\theta) = \begin{bmatrix} f_1.m_1 & f_1.m_2 & f_1.m_3 \\ f_2.m_1 & f_2 m_2 & f_2.m_3 \\ f_3.m_1 & f_3.m_2 & f_3.m_3 \end{bmatrix} \tag{2.1}$$

เมื่อหมุน m_1 และแทนค่าในสมการการหมุน เวกเตอร์ $(f_1,\ f_2,\ f_3)$ และ$(m_1,\ m_2,\ m_3)$ ที่ตั้งฉากกันเมื่อฉายแกนในระนาบ 3 มิติ (Dot Product) จะมีค่าเป็น 0 จะได้

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & f_2.m_2 & f_2.m_3 \\ 0 & f_3.m_2 & f_3.m_3 \end{bmatrix} \tag{2.2}$$

เมื่อทำการฉายแกนในระนาบ 3 มิติ (Dot Product) สามารถเขียนได้เป็น

กำหนดให้ $S_\theta = \sin(\theta)$ และ $C_\theta = \cos(\theta)$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\theta & -S_\theta \\ 0 & S_\theta & C_\theta \end{bmatrix} \tag{2.3}$$

เมทริกซ์ $R_X(\theta)$ หมายถึงเมทริกซ์การหมุนรอบพิกัดแกน X ด้วยมุม θ และในทำนองเดียวกันการหมุนรอบพิกัดแกน Y แสดงดังสมการที่ (2.4) และแกน Z แสดงดังสมการที่ (2.5)

$$R_Y(\theta) = \begin{bmatrix} C_\theta & 0 & S_\theta \\ 0 & 1 & 0 \\ -S_\theta & 0 & C_\theta \end{bmatrix} \quad (2.4)$$

$$R_Z(\theta) = \begin{bmatrix} C_\theta & -S_\theta & 0 \\ S_\theta & C_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

เมทริกซ์ทั้งสามนี้คือ เมทริกซ์การหมุนมูลฐาน (Basic Rotation Matrix) ซึ่งแสดงการหมุนรอบแกนพิกัดอ้างอิง เมทริกซ์การหมุนมูลฐานนี้มีประโยชน์ในการสร้างเมทริกซ์การหมุนทั่วไปที่ซับซ้อนกว่าต่อไป

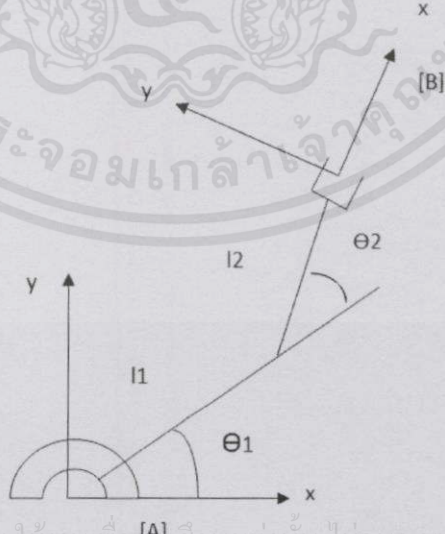
2.1.2 ปัญหาจลนศาสตร์ไปข้างหน้า

หุ่นยนต์ประกอบขึ้นจากการต่อกันของก้านต่อ (Links) ด้วยข้อต่อ ไล่เรียงลำดับไปจากส่วนฐานถึงส่วนปลายแขน ในลักษณะโครงสร้างแบบโซ่เปิด โดยภาพรวมหุ่นยนต์สามารถเคลื่อนที่ได้ด้วยการขับเคลื่อนของตัวขับ ซึ่งมักติดตั้งอยู่ที่ข้อต่อ ดังนั้นท่าทางของหุ่นยนต์ถูกกำหนดด้วยค่าพารามิเตอร์ของข้อต่อ หากต้องการทราบว่าหุ่นยนต์เคลื่อนที่อย่างไร ก็จะต้องทราบความสัมพันธ์หรือฟังก์ชันของตำแหน่งและทิศทางการหมุนของหุ่นยนต์

จากรูปที่ 2.3 เนื่องจากตำแหน่งและทิศทางการหมุนของวัตถุอธิบายได้ด้วยเมทริกซ์การแปลงเอกพันธ์ ดังนั้นเฟรม [A] และ [B] จึงถูกติดตั้งบนส่วนปลายแขนและฐานของหุ่นยนต์ ทำให้การแก้ปัญหาจลนศาสตร์ไปข้างหน้ารูปแบบหนึ่ง จึงเป็นการคำนวณเมทริกซ์การแปลงเอกพันธ์

$$T_B^A \ (q_1, \ q_2, \dots q_3)$$

ในพจน์ของตัวแปรข้อต่อ $q_1, q_2, \dots q_n$ จำนวน n ข้อต่อ



รูปที่ 2.3 แผนภาพของหุ่นยนต์ที่เคลื่อนที่ในระนาบ มีจำนวนองศาเสรีเท่ากับหมุนรอบพิกัดแกน Z ด้วยมุม θ_1 และ θ_2

เมื่อมาพิจารณารูปที่ 2.3 ซึ่งแสดงแผนภาพของหุ่นยนต์ที่เคลื่อนที่ในระนาบ มีจำนวนองศาเสรีเท่ากับ 2 ความยาวของก้านต่อ (Links) ทั้งสองคือ l_1 และ l_2 ข้อต่อของมุมทั้งสองถูกควบคุมด้วยมุม θ_1 และ θ_2 เฟรม $[A]$ และ $[B]$ ถูกติดตั้งดังในรูป จะวิเคราะห์จลนศาสตร์ไปข้างหน้าของหุ่นยนต์ จากรูปที่ 2.3 พิกัดของจุดกำเนิดของ $[B]$ เทียบกับ $[A]$ สามารถเขียนได้เป็น

กำหนดให้ $C_1 = \cos(\theta_1)$ $C_{12} = \cos(\theta_1 + \theta_2)$ ส่วน $S_1 = \sin(\theta_1)$ และ $S_{12} = \sin(\theta_1 + \theta_2)$

$$O_B^A = \begin{bmatrix} l_1C_1 + l_2C_{12} \\ l_1S_1 + l_2C_{12} \\ 0 \end{bmatrix} \tag{2.6}$$

นี่คือสมการการเลื่อน (Translations) ส่วนการหมุนของ $[B]$ เทียบกับ $[A]$ ซึ่งทราบว่าเป็นการหมุนรอบพิกัดแกน Z ด้วยมุม θ_1 และ θ_2 ดังนั้นจากเมทริกซ์การหมุนมูลฐาน ซึ่งแสดงการหมุนรอบแกนพิกัดอ้างอิง Z คือ

$$R_B^A(\theta) = \begin{bmatrix} C_{12} & -S_{12} & 0 \\ S_{12} & C_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

ดังนั้น ตำแหน่งและทิศทางการหมุนของส่วนปลายแขน จึงสามารถเขียนเป็นฟังก์ชันของตัวแปรข้อต่อ θ_1 และ θ_2ได้ด้วยเมทริกซ์การแปลงเอกพันธ์

$$T_B^A(\theta_1, \theta_2) = \begin{bmatrix} C_{12} & -S_{12} & 0 & l_1C_1 + l_2C_{12} \\ S_{12} & C_{12} & 0 & l_1S_1 + l_2C_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

2.1.3 สัญญานิยมของเดนาวิทและฮาเทนเบิร์ก (The Danavit – Hartenberg Representation)

เมื่อทำการเขียนเมทริกซ์การแปลงเอกพันธ์ T_{i-1}^i ในฟังก์ชันของตัวแปลข้อต่อ q_i แล้ว ก็จะสามารถคำนวณหาทิศทางของหุ่นยนต์ได้ แต่ในกรณีที่โครงสร้างหุ่นยนต์มีความสลับซับซ้อน การวิเคราะห์จลนศาสตร์ไปข้างหน้าโดยพิจารณาจากเรขาคณิตโดยตรงอาจทำได้ลำบาก และเกิดความผิดพลาดได้ง่าย ดังนั้นเพื่อความเป็นระบบในการวิเคราะห์ การสร้างและติดตั้งเฟรมจะกระทำได้โดยวิธี D-H Representation

ขั้นตอนการติดตั้งเฟรมตามวิธี D-H Representation จะสมมติว่าข้อต่อของหุ่นยนต์เป็นข้อต่ออย่างง่ายชนิดข้อต่อเชิงมุมหรือข้อต้อเชิงเส้นเท่านั้น ในกรณีเป็นข้อต่อผสมให้ทำการแยกข้อต่อเป็นการต่อกันแบบอนุกรมของข้อต่ออย่างง่ายเสียก่อน โดยมีวิธีดังนี้

1. กำหนดหมายเลขของก้านต่อ (Links) จำนวน $n+1$ ก้านต่อ โดยเริ่มจากก้านต่อที่ 0 สำหรับฐาน (Base) ไล่เรียงไปเรื่อยๆ จนถึงก้านต่อที่ n ซึ่งมีปลายแขนติดอยู่

2. กำหนดหมายเลขสำหรับข้อต่อ จำนวน n ข้อต่อ โดยเริ่มจากข้อต่อที่ 1 ซึ่งเชื่อมก้านต่อที่ 1 เข้ากับฐาน ไล่เรียงไปเรื่อยๆ ไปสู่ข้อต่อของส่วนปลายแขน ข้อต่อที่ i จะทำการเชื่อมก้านต่อที่ $i-1$

3. ติดตั้งเฟรมฐาน $[A]$ จากรูปที่ 2.3 ที่ฐาน เพื่อเป็นเฟรมอ้างอิง โดยการติดตั้งสามารถทำได้อย่างเสรี ควรเลือกติดตั้งเฟรมในลักษณะที่ทำให้เมทริกซ์การแปลงเอกพันธ์ T_0^A มีความซับซ้อนน้อยที่สุด

4. ติดตั้งเฟรมของส่วนปลายแขน $[B]$ ควรเลือกติดตั้งเฟรมในลักษณะที่ทำให้เมทริกซ์การแปลงเอกพันธ์ T_B^n มีความซับซ้อนน้อยที่สุดเท่าที่จะทำได้

5. ติดตั้งเฟรม $[i]$ ซึ่งเคลื่อนไปด้วยกันกับก้านต่อที่ i โดยการสร้างพิกัด z_i ให้อยู่ในแนวเดียวกันกับแกนหมุนหรือแกนเลื่อน และทิศทางบวกชี้ไปยังทิศทางที่ก่อให้เกิดการหมุนหรือการเลื่อนขนานที่เป็นบวก

6. สร้างพิกัด x_i ให้อยู่ในแนวเดียวกับเส้นตั้งฉากร่วม (Common Normal Line) ระหว่างพิกัด z_i และ z_{i+1} ทิศทางชี้ไปยังพิกัด z_i ไปยัง z_{i+1}

7. ในกรณีที่แกนพิกัด z_i และ z_{i+1} ตัดกัน แกนพิกัด x_i จะตั้งฉากกับระนาบที่สร้างขึ้นโดยพิกัด z_i และ z_{i+1} ทิศทางบวกชี้ในทิศทางไปสู่ข้อต่อของส่วนปลายแขน

8. ในกรณีที่แกนพิกัด z_i และ z_{i+1} ขนานกัน แกนพิกัด x_i ควรเลือกให้ตัดกับแกนพิกัด x_{i+1} เพื่อทำให้เมทริกซ์การแปลงเอกพันธ์ T_{i-1}^i มีความซับซ้อนน้อยลง

9. สำหรับแกนพิกัด x_n นั้น สามารถสร้างได้ตามสะดวก แต่หากเป็นไปได้ ควรเลือกเพื่อทำให้เมทริกซ์ T_{n-1}^n และ T_B^n ซับซ้อนน้อยลง

10. จุดกำเนิดของ $[i]$ คือจุดตัดของแกนพิกัด x_i และ z_i

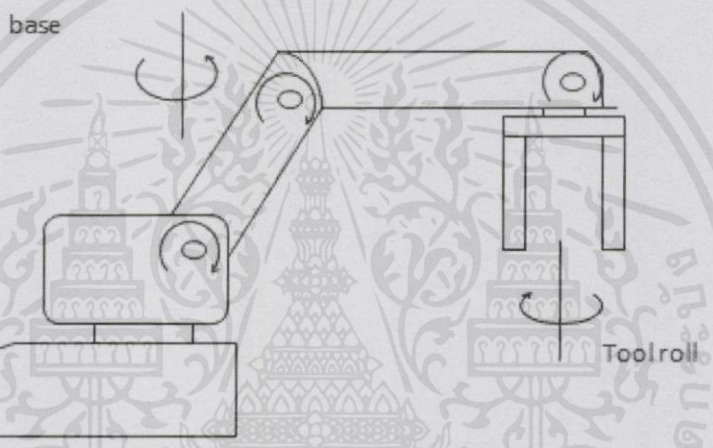
11. ติดตั้งเฟรม $[0]$ ซึ่งอยู่กับฐาน โดยควรเลือกให้ทับกันกับเฟรม $[1]$ ณ ตำแหน่งเริ่มต้น (Home Position) ของหุ่นยนต์

หลังจากที่ได้ทำการติดตั้งเฟรมก้านต่อของหุ่นยนต์แล้ว ก็จะสามารถกำหนดค่าพารามิเตอร์ 4 ตัว เรียกว่า พารามิเตอร์ของเดนาวิทและฮาร์เทนเบิร์ก (Denavit-Hartenberg Parameters) ประกอบไปด้วย

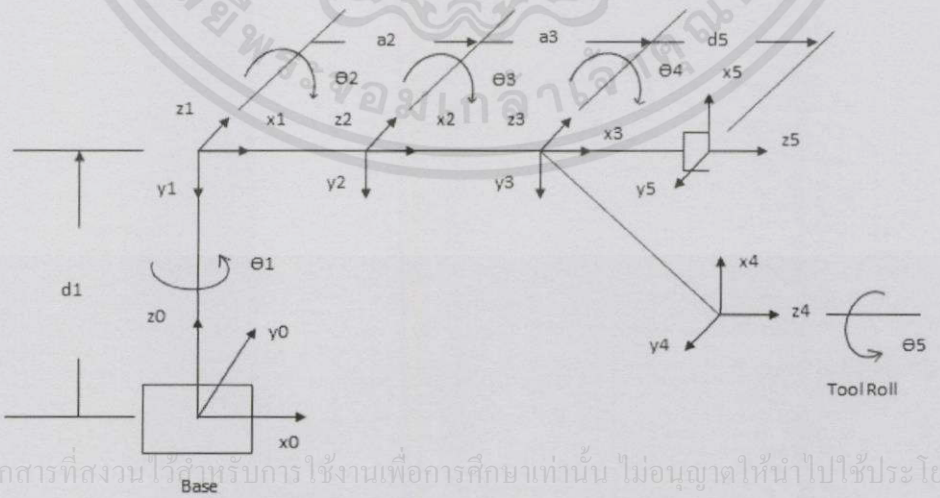
- ความยาวของก้านต่อ (Link Length) แทนด้วยสัญลักษณ์ a_{i-1} คือ ระยะจากแกนพิกัด z_{i-1} ไปยัง z_i ตามแนวพิกัดแกน x

- มุมบิดของก้านต่อ (Link Twist) เป็นมุมบิดระหว่างแกนก้านต่อที่ $i-1$ แทนด้วยสัญลักษณ์ a_{i-1} คือ มุมที่วัดจากพิกัดแกน z_{i-1} ไปยัง z_i รอบแกนพิกัด x_{i-1}
- ระยะเลื่อนของข้อต่อ (Joint Displacement) แทนด้วยสัญลักษณ์ d_i คือ ระยะทางจากแกนพิกัด x_{i-1} ไปยัง x_i ตามแนวแกนพิกัด z
- มุมของข้อต่อ (Joint Angle) แทนด้วยสัญลักษณ์ θ_i คือ มุมที่วัดจากแกนพิกัด x_{i-1} ไปยัง x_i รอบแกนพิกัด z

ในที่นี้จะแสดงตัวอย่างแขนกลแบบ Alpha II แสดงในรูปที่ 2.4 โดยการวิเคราะห์แขนกลแบบ Alpha II มี Link Coordinate ดังแสดงในรูปที่ 2.5 และมีพารามิเตอร์ของเดนาวิทและฮาร์เทนเบิร์ก ดังในตารางที่ 2.1 ตามลำดับ



รูปที่ 2.4 แขนกลแบบ Alpha II



รูปที่ 2.5 Link Coordinates ของแขนกลแบบ Alpha II

ตารางที่ 2.1 D-H Parameters ของแขนกลแบบ Alpha II

Axis	θ	d	a	α
1	0	d_1	0	-90°
2	0	0	a_2	0
3	0	0	a_3	0
4	-90°	0	0	-90°
5	0	d_5	0	0

2.2 จลนศาสตร์แบบย้อนกลับ (Inverse Kinematic)

เป็นการวิเคราะห์หาฟังก์ชันของตัวแปรข้อต่อ ในพจน์ของตำแหน่งและทิศทางการหมุนของหุ่นยนต์ที่ถูกกำหนดมาให้สำหรับหุ่นยนต์แล้ว ปัญหาจลนศาสตร์แบบย้อนกลับจะมีความสลับซับซ้อนกว่าปัญหาจลนศาสตร์แบบไปข้างหน้า เนื่องด้วยการเคลื่อนที่ในสามมิติ ต้องอธิบายด้วยเรขาคณิตและการส่งจากปริภูมิของข้อต่อไปยังปริภูมิของการทำงานนั้นเกี่ยวข้องกับฟังก์ชันตรีโกณมิติ (Trigonometric Function) ซึ่งเป็นฟังก์ชันไม่เชิงเส้น (Nonlinear Function) และไม่ใช่ฟังก์ชันหนึ่งต่อหนึ่งทั่วถึง (Bijective Function) ดังนั้นการวิเคราะห์จลนศาสตร์ย้อนกลับจึงไม่มีขั้นตอนที่ตายตัวดังเช่นกรณีจลนศาสตร์แบบไปข้างหน้า

2.2.1 ลักษณะของผลเฉลยในปัญหาจลนศาสตร์แบบย้อนกลับ

จากการวิเคราะห์ปัญหาจลนศาสตร์แบบไปข้างหน้า กล่าวคือ เมื่อกำหนดเวกเตอร์ของข้อหุ่นยนต์มาให้ จะสามารถคำนวณตำแหน่งและทิศทางการหมุนของหุ่นยนต์ได้ หากใช้เมทริกซ์แปลงเอกพันธ์จะเขียนอธิบายได้เป็น

$$T = T(\bar{q}) \tag{2.9}$$

2.2.2 การมีอยู่ของคำตอบ

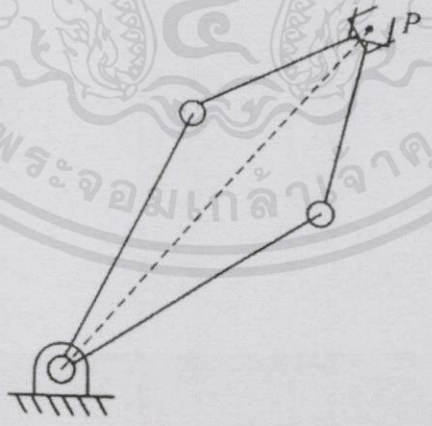
ในการแก้ปัญหา $\bar{q}$ คือ คำตอบของ $\bar{q}$ ที่ทำให้หุ่นยนต์มีตำแหน่งและทิศทางการหมุนคงที่กำหนดนั้น มีอยู่จริงหรือไม่ เรียกว่าการมีอยู่ของคำตอบ (Existence of Solution) เงื่อนไขที่จำเป็นและเพียงพอของการมีอยู่ของคำตอบคือ ตำแหน่งและทิศทางของส่วนปลายแขนที่กำหนดมาให้จะต้องอยู่ในปริภูมิทำงานของหุ่นยนต์ ในกรณีทั่วไปจำนวนองศาเสรีสำหรับการกำหนดตำแหน่งและทิศทางการหมุนใดๆ ในปริภูมิสามมิติมีค่าเท่ากับ 6 ดังนั้นหากต้องการให้มีคำตอบสำหรับ $\bar{q}$ เสมอ ไม่

ว่าจะกำหนดตำแหน่งและทิศทางการหมุนใดๆ ก็ตาม หุ่นยนต์จะต้องมีข้อต่อที่เป็นอิสระต่อกันทางจลนศาสตร์ (Kinematically Independent) อย่างน้อย 6 ข้อต่อในปริภูมิของการเคลื่อนที่ที่ต้องการ กรณีที่หุ่นยนต์มีจำนวนข้อต่อน้อยกว่า 6 ข้อต่อ จะส่งผลให้ไม่สามารถหาคำตอบของ $\bar{q}$ ได้ในบางตำแหน่งและทิศทางการหมุนใดๆ เนื่องจากปริภูมิทำงานเป็นปริภูมิย่อย (Subspace) ของปริภูมิสามมิติ อย่างไรก็ดี ในบางกรณีปริภูมิของการทำงานอาจถูกกำหนดให้เป็นปริภูมิย่อยของปริภูมิสามมิติ ในกรณีเหล่านี้ถึงแม้ว่าหุ่นยนต์จะมีจำนวนข้อต่อน้อยกว่า 6 ข้อต่อ แต่ก็อาจยังสามารถเคลื่อนที่ได้ตามที่ต้องการ อันอนุมานถึงการมีอยู่ของคำตอบของปัญหาจลนศาสตร์แบบย้อนกลับ

การมีอยู่ของคำตอบยังขึ้นอยู่กับขอบเขตของข้อต่อ (Joint Limit) และสิ่งกีดขวาง (Obstacle) อีกด้วย ค่าของตัวแปรข้อต่อที่เป็นคำตอบของฟังก์ชันของ $T(\bar{q})$ อาจไม่สามารถเกิดขึ้นได้เนื่องจากอยู่นอกขอบเขตของข้อต่อ หรือทำให้เกิดการชนกันระหว่างก้านต่อของหุ่นยนต์กับสิ่งกีดขวาง เพราะฉะนั้นในทางปฏิบัติจำเป็นต้องมีการตรวจสอบค่าของตัวแปรข้อต่อที่แก้มาได้ก่อนนำไปใช้เสมอ

2.2.3 การมีคำตอบหลายคำตอบ

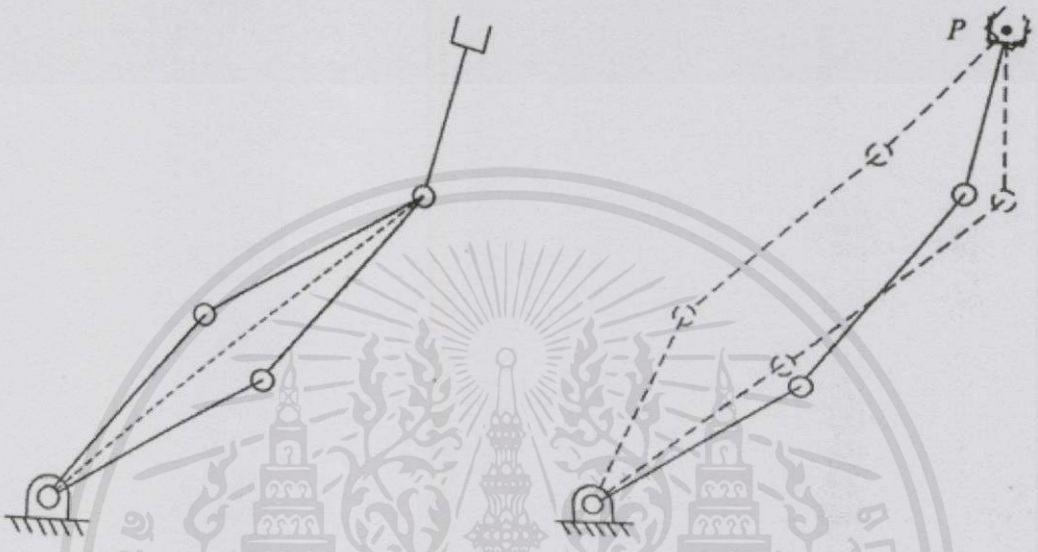
หากผลเฉลยของเวกเตอร์ข้อต่อ $\bar{q}$ สำหรับตำแหน่งและทิศทางการหมุนหนึ่งๆ มีอยู่จริงแล้ว มีความเป็นไปได้ที่ผลเฉลยนั้นจะมีหลายคำตอบ เนื่องด้วยฟังก์ชันตรีโกณมิติไม่ใช่ฟังก์ชันหนึ่งต่อหนึ่ง (Injective Function) ยกตัวอย่างเช่น หุ่นยนต์ที่เคลื่อนที่ในระนาบและมีจำนวนองศาเสรีเท่ากับ 2 ดังแสดงในรูปที่ 2.6 ปริภูมิทำงานของหุ่นยนต์คือเซตของตำแหน่งของส่วนปลายแขนที่อยู่ในขอบเขตการเข้าถึงได้ของมัน จากแผนภาพของหุ่นยนต์พบว่า เมื่อกำหนดตำแหน่งใดๆ ให้หุ่นยนต์จะสามารถอยู่ที่ตำแหน่งนั้นได้ใน 2 ลักษณะ สอดคล้องกับคำตอบ 2 คำตอบที่แก้ด้วยวิธีการทางคณิตศาสตร์



รูปที่ 2.6 ส่วนของปลายแขนของหุ่นยนต์สามารถอยู่ ณ จุด P ได้ 2 ลักษณะ

เมื่อพิจารณาหุ่นยนต์ที่เคลื่อนที่ในระนาบแต่มีจำนวนองศาเสรีเท่ากับ 3 ดังแสดงในรูปที่ 2.7 ปริภูมิทำงานของหุ่นยนต์คือ เซตของตำแหน่งและการหมุนในระนาบของส่วนปลายแขน เพราะฉะนั้น

เมื่อกำหนดตำแหน่งและการหมุนใดๆ ให้หุ่นยนต์จะสามารถอยู่ ณ ตำแหน่งและการหมุนได้ใน 2 ลักษณะ ทว่าหากการหมุนของส่วนปลายแขนไม่มีผลต่อการทำงานของหุ่นยนต์ เสมือนหนึ่งว่าปริภูมิทำงานคือเซตของตำแหน่งเท่านั้นแล้ว เมื่อกำหนดตำแหน่งใดๆ ให้หุ่นยนต์จะสามารถอยู่ ณ ตำแหน่งนั้นได้หลากหลายรูปแบบ สอดคล้องกับการหมุนใดๆ ของส่วนปลายแขนจากข้อต่อที่ 3 ซึ่งเมื่อวิเคราะห์ด้วยวิธีการทางคณิตศาสตร์จะพบว่าจำนวนของคำตอบจะเป็นอนันต์



รูปที่ 2.7 คำตอบของหุ่นยนต์ที่มีจำนวนองศาเสรีเท่ากับ 3 เมื่อกำหนดตำแหน่ง และทิศทางการหมุน (ซ้าย) และเมื่อกำหนดเฉพาะตำแหน่ง (ขวา) ของส่วนปลายแขน

จำนวนของคำตอบสำหรับตำแหน่งและทิศทางการหมุนหนึ่งๆ ขึ้นอยู่กับจำนวนข้อต่อของหุ่นยนต์ หากหุ่นยนต์มีจำนวนข้อต่อที่ถูกควบคุมมากกว่าจำนวนองศาเสรีของปริภูมิทำงานแล้ว จำนวนของคำตอบจะเป็นอนันต์ ในขณะที่เมื่อจำนวนข้อต่อน้อยกว่าจำนวนองศาเสรีแล้ว อาจไม่สามารถหาผลเฉลยของเวกเตอร์ข้อต่อสำหรับบางตำแหน่งและทิศทางการหมุนได้ นอกจากนี้แล้ว จำนวนคำตอบนี้ยังขึ้นอยู่กับลักษณะโครงสร้างทางจลนศาสตร์ของหุ่นยนต์ ซึ่งอาจอธิบายได้ด้วยพารามิเตอร์ของเดนาวิทและฮาเทนเบิร์ก โดยทั่วไปแล้วยิ่งจำนวนพารามิเตอร์ของความยาวของก้านต่อ a_i หรือระยะเลื่อนขนานของข้อต่อ d_i ที่มีค่าไม่เท่ากับ 0 นั้นมีมากเท่าใด จะยิ่งทำให้จำนวนคำตอบมีมากขึ้น ในทำนองเดียวกัน ยิ่งจำนวนพารามิเตอร์ของมุมบิดของก้านต่อ α_i หรือมุมของข้อต่อ θ_i ที่มีค่าไม่เท่ากับ 0° หรือ 90° นั้นมีมากเท่าใด ก็จะยิ่งทำให้จำนวนคำตอบมีมากขึ้น

ในกรณีที่ผลเฉลยมีหลายคำตอบแต่จำนวนของคำตอบมีค่าจำกัด (Finite Number of Solutions) จะต้องมีการตัดสินใจว่าจะใช้คำตอบใด คำตอบหลายคำตอบเหล่านั้นสามารถแบ่งแยกออกเป็นสาขาของคำตอบ (Branch of Solutions) ได้ตามเงื่อนไขของตัวแปร เช่น หุ่นยนต์ในรูปที่ 2.6 มีสาขาของคำตอบอยู่ 2 สาขา ตามเงื่อนไข $\theta_2 > 0$ หรือ $\theta_2 < 0$ โดยปกติแล้วผลเฉลยของ

เวกเตอร์ของข้อต่อ $\bar{q}$ ที่ตรงกับตำแหน่งและทิศทางการหมุนของจุดใดๆ ในแนววิถี (Trajectory) การเคลื่อนที่หนึ่งๆ มักจะถูกเลือกให้เป็นคำตอบที่อยู่ในสาขาเดียวกัน เพราะว่าการข้ามสาขาของคำตอบจะต้องผ่านสภาวะเอกฐานของหุ่นยนต์ (Robot Singularity) อันทำให้ค่าที่ได้จากการคำนวณหาความเร็วของข้อต่อนั้นมีขนาด (Magnitude) ใหญ่มาก สอดคล้องกับการเคลื่อนที่แบบกระโดดจากตำแหน่งและทิศทางการหมุนในสาขาของคำตอบหนึ่งๆ ไปยังตำแหน่งและทิศทางการหมุนในอีกสาขาของคำตอบ ส่งผลให้วงควบคุม (Control Loop) เกิดความไม่เสถียร (Instability)

2.2.4 วิธีการในการวิเคราะห์จลนศาสตร์แบบย้อนกลับ

โดยปกติแล้วไม่มีวิธีการทั่วไปที่สามารถใช้แก้สมการพีชคณิตแบบไม่เชิงเส้นใดๆ ก็ได้ เว้นเสียแต่วิธีทางเชิงเลข (Numerical Method) อย่างไรก็ตามวิธีทางเชิงเลขมีข้อเสียที่ว่า โดยปกติแล้วขั้นตอนวิธีจะให้คำตอบเพียงคำตอบเดียวซึ่งอาจมิใช่คำตอบที่ต้องการหรือไม่ได้อยู่ในสาขาเดียวกันกับคำตอบปัจจุบัน ยิ่งไปกว่านั้นหลักการของขั้นตอนวิธีมักเป็นวิธีทำซ้ำ (Iterative Method) ซึ่งต้องใช้เวลาที่ไม่มีแน่นอนและนานกว่า ซ้ำยังอาจประสบปัญหาคำตอบไม่ลู่เข้า (Divergence) ได้อีกด้วย ดังนั้นวิธีทางเชิงเลขสำหรับการหาผลเฉลยจลนศาสตร์แบบย้อนกลับมักไม่ถูกนำมาใช้ในการควบคุมแบบทันที (Real-Time Control) แต่มักใช้ในการจำลองระบบที่ซับซ้อนและไม่สามารถหาผลเฉลยรูปแบบปิด (Closed Form Solution) ได้

ในทางปฏิบัติหุ่นยนต์ที่ใช้งานจริงมักถูกออกแบบให้มีโครงสร้างที่ง่าย และสามารถหาผลเฉลยรูปแบบปิดได้ เพื่อประโยชน์หลักในการคำนวณให้ทันภายในอัตราการชักตัวอย่าง (Sampling Rate) ของระบบควบคุมแบบทันที ผลเฉลยของเวกเตอร์ของข้อต่อจะถูกจัดว่าเป็นผลเฉลยรูปแบบปิด หากเวกเตอร์ของข้อต่อ $\bar{q}$ สามารถเขียนเป็นฟังก์ชันชัดแจ้งของพารามิเตอร์ที่ใช้อธิบายตำแหน่งและทิศทางการหมุนที่กำหนดมาให้ แม้ว่าในความเป็นจริงแล้วบ่อยครั้งอาจต้องเขียนนิพจน์ในรูปแบบฟังก์ชันประกอบ (Composite Function) ย่อยๆ หรือฟังก์ชันโดยปริยาย (Implicit Function) ของพหุนาม (Polynomial) อันดับไม่เกิน 4 เพื่อลดความซับซ้อนวิธีการที่ใช้หาผลเฉลยรูปแบบปิดสามารถแบ่งได้เป็น สองวิธีการหลักในการแก้ไขปัญหา ได้แก่ วิธีพีชคณิต (Algebraic Approach) และวิธีการเรขาคณิต (Geometric Approach)

2.2.5 วิธีการพีชคณิต

วิธีการนี้จะแก้หาผลเฉลยของ $\bar{q}$ โดยการเปรียบเทียบตำแหน่งและทิศทางการหมุนของหุ่นยนต์ที่กำหนดมาให้กับนิพจน์สัญลักษณ์ (Symbolic Expression) ของตำแหน่งและทิศทางการหมุนที่ได้จากการวิเคราะห์จลนศาสตร์แบบไปข้างหน้า หากใช้เมทริกซ์การแปลงเอกพันธ์เพื่ออธิบายตำแหน่งและทิศทางการหมุนแล้ว จะสามารถสร้างสมการจากการเปรียบเทียบได้ เมื่อได้ทำการสร้างระบบสมการไม่เชิงเส้นเรียบร้อยแล้ว ก็จะทำการแก้ระบบสมการเพื่อหาคำตอบของ $\bar{q}$ ต่อไป เพื่อการนี้เทคนิคและวิธีการต่างๆ ในการแก้ระบบสมการสามารถนำมาใช้ได้โดยเฉพาะอย่างยิ่งเอกลักษณ์ทางตรีโกณมิติ โดยหลักการแล้วเทคนิคต่างๆ จะถูกนำมาใช้เพื่อกำจัดตัวแปรสเกลาร์หลายตัวในสมการ

หนึ่งให้เหลือเพียงตัวแปรเดียว แล้วจึงแก้หาตัวแปรนั้นโดยเขียนเป็นฟังก์ชันชัดแจ้ง ประเด็นเรื่องการมีอยู่ของคำตอบและการมีคำตอบหลายคำตอบต้องนำมาพิจารณาอันจะนำไปสู่เงื่อนไขต่างๆ ของพารามิเตอร์

2.2.6 วิธีการเรขาคณิต

วิธีการเรขาคณิตจะทำการสร้างสมการจากความสัมพันธ์ทางเรขาคณิตในโครงสร้างของหุ่นยนต์ โดยทั่วไปแล้วความสัมพันธ์ทางเรขาคณิตมักมีที่มาจากหลักการรวมกันของเวกเตอร์ที่ประกอบเข้าด้วยกันเป็นด้านของรูปหลายเหลี่ยมเพื่อให้ได้เวกเตอร์ศูนย์ อนึ่ง รูปหลายเหลี่ยมที่ใช้นี้ไม่จำเป็นต้องอยู่ในระนาบสำหรับหุ่นยนต์ที่มีค่าของพารามิเตอร์ของเดนาวิทและฮาเทนเบิร์ก $a_i = 0$, $d_i = 0$ และ $\alpha_i = 0^\circ$ หรือ $\pm\frac{\pi}{2}$ อยู่หลายตัวแล้ว อาจสามารถหาความสัมพันธ์ทางเรขาคณิตเหล่านั้นได้โดยการฉายภาพ (Project) ของรูปหลายเหลี่ยมลงบนระนาบเสียก่อน กล่าวคือ หากต้องการแก้หาตัวแปรข้อต่อ θ_i จะทำการฉายภาพของก้านต่อหุ่นยนต์ลงบนระนาบ $x_i - y_i$ แล้วจึงทำการวิเคราะห์เรขาคณิตในระนาบซึ่งมีความง่ายกว่า หากเป็นไปได้ความสัมพันธ์ทางเรขาคณิตหนึ่งๆ ควรที่จะมีความเกี่ยวข้องกับตัวแปรของข้อต่อที่ยังไม่ทราบค่าเพียงตัวเดียว

2.3 Kinect Camera

Kinect คือ อุปกรณ์ตรวจจับการเคลื่อนไหวที่ถูกพัฒนาขึ้นโดยบริษัท Microsoft ซึ่งสามารถทำให้ผู้ใช้สามารถควบคุมและมีปฏิสัมพันธ์กับเครื่องเล่นเกมหรือคอมพิวเตอร์ได้โดยใช้การออกท่าทางและคำสั่งเสียงแทนการใช้อุปกรณ์ควบคุมอื่นๆ

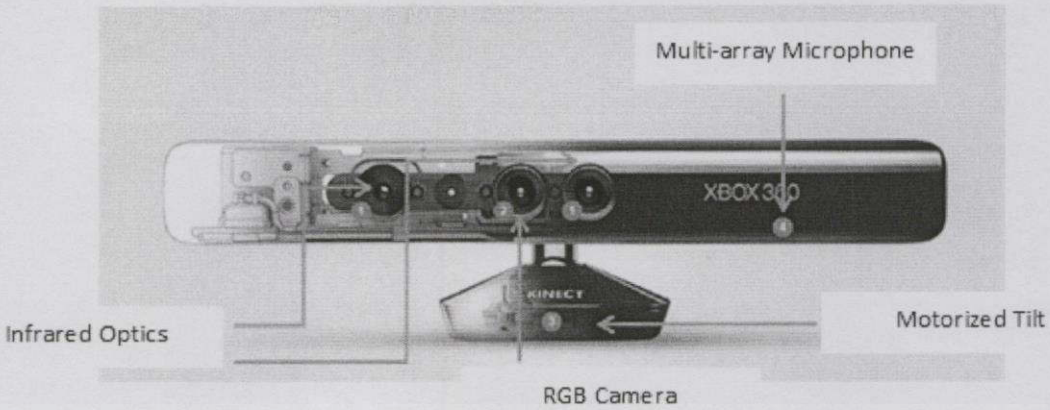
ส่วนประกอบที่สำคัญของ Kinect Camera ดังแสดงในรูปที่ 2.8 ประกอบด้วย

1. Infrared Optics (Depth Sensor) ซึ่งประกอบไปด้วย Infrared Projector ทำหน้าที่ฉายแสงอินฟราเรดซึ่งมองด้วยตาเปล่าไม่เห็นออกมาเป็นแพทเทิร์นจุด และมี Infrared Camera ที่ทำหน้าที่รับแสงอินฟราเรดที่ถูกฉายออกไปเพื่อใช้ในการสร้างภาพที่บอกความลึกตื้นของวัตถุ (Depth Map)

2. RGB Camera เป็นกล้องถ่ายภาพเคลื่อนไหวใช้รับข้อมูลสีและช่วยในระบบจดจำใบหน้า (Face Recognition)

3. Motorized Tilt ทำหน้าที่ในการปรับการมองเห็นของกล้อง Kinect ให้เงยขึ้นหรือก้มลงได้ 27°

4. Multi-array Microphone มีหน้าที่ใช้รับเสียงและมีส่วนช่วยในการระบุตำแหน่งของผู้ใช้ว่าอยู่ที่ไหน มีไมโครโฟนทั้งหมดจำนวน 4 ตัว สามารถใช้แยกแยะเสียงของผู้เล่นและเสียงรบกวนภายนอกออกจากกันได้

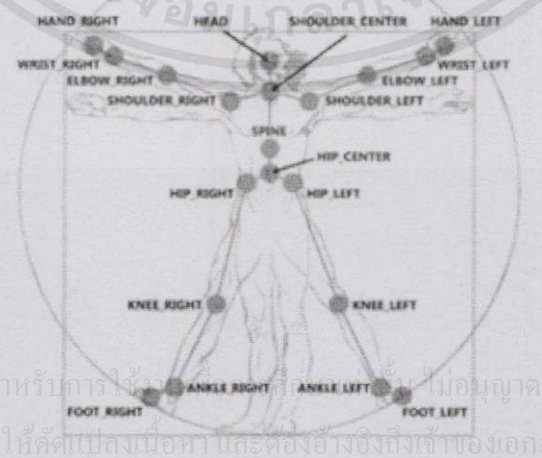


รูปที่ 2.8 ส่วนประกอบของ Kinect Camera

2.3.1 หลักการทำงานของ Kinect Camera

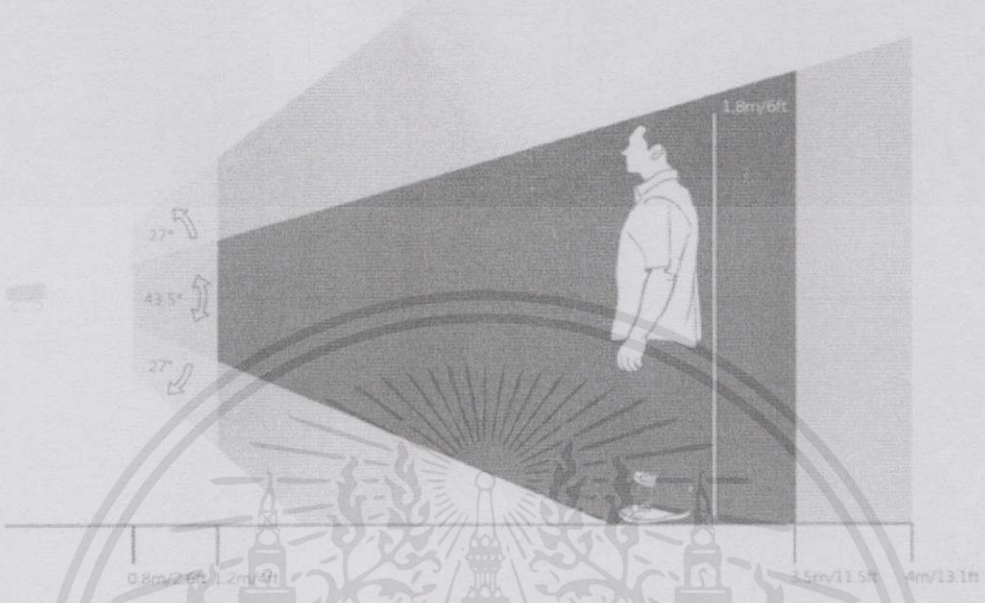
Microsoft Kinect Sensor จะตรวจจับการเคลื่อนไหวของผู้ใช้โดยเริ่มต้นจากการให้ Infrared Projector ฉายแสงอินฟราเรดในลักษณะเป็นแพทเทิร์นจุด จากนั้น Infrared Camera จะรับระดับความสว่างของแสงที่ตกกระทบบนวัตถุ โดยวัตถุที่อยู่ใกล้แสงจะมีความสว่างมากและวัตถุที่อยู่ไกลจะมีแสงสว่างน้อยกว่า จากนั้นจะนำข้อมูลที่ได้ไปประมวลผลด้วยซอฟต์แวร์จะได้ระดับความลึกตื้นออกมาดังภาพด้านล่าง โดยวัตถุที่อยู่ใกล้กับตัวเซนเซอร์จะมีสีอ่อนและวัตถุที่อยู่ไกลจะมีสีเข้มไล่ระดับกันไป

เมื่อ Kinect รู้ระดับความตื้นลึกแล้วก็จะสามารถแยกผู้เล่นออกจากสภาพแวดล้อมภายในห้องได้ นอกจากนั้น Kinect Camera ยังมีระบบ Skeletal Tracking ที่ใช้ติดตามโครงกระดูกของผู้ใช้งาน ซึ่งสามารถติดตามได้มากที่สุด 2 คน แต่จะมองเห็นทั้งหมด 6 คน ซึ่งรูปที่ 2.9 แสดงภาพของโครงกระดูกมนุษย์ที่แทนด้วยข้อต่อ 20 จุดสำคัญตามร่างกาย



รูปที่ 2.9 จุดสำคัญตามร่างกาย

ระยะการตรวจจับการเคลื่อนไหวของ Kinect นั้นจะตรวจจับได้ใกล้ที่สุดในระยะที่ห่างออกไป 0.8 เมตร จนไปถึง 4 เมตร แต่ระยะที่ทำการตรวจจับได้ดีที่สุดคือ ห่างจาก Kinect ตั้งแต่ 1.2 เมตร จนไปถึง 3.5 เมตร มุมมองการเห็นของกล้องแสดงดังรูปที่ 2.10



รูปที่ 2.10 มุมมองการมองเห็นของ Kinect Sensor

2.3.2 การเชื่อมต่อกับคอมพิวเตอร์และการพัฒนา

สำหรับการพัฒนา Application กับ Kinect นั้นมี Software Library ที่น่าเลือกใช้อยู่ 2 ค่ายคือ Kinect SDK และ Open Kinect ซึ่งแต่ละ Library ก็มีข้อดีและข้อเสียที่แตกต่างกันไป อย่างเช่น Kinect SDK ซึ่งเป็นของ Microsoft เองก็มีข้อดี คือ Hardware เป็นของตัวเอง เพราะฉะนั้นชุดคำสั่งต่างๆ จะอัพเดตออกมาเสมอ แต่มีข้อเสียคือเนื่องจากเป็นของ Microsoft ดังนั้น SDK นี้จึงใช้ได้กับระบบปฏิบัติการ Windows เท่านั้น จากจุดด้อยตรงนี้จึงมาเป็นจุดเด่นของทาง Open Kinect ซึ่งสามารถใช้บนระบบปฏิบัติการไหนก็ได้ เพราะตัว Library เป็น Open Source โดยผู้พัฒนาสามารถนำ Source Code ของ Library มา Compile บนระบบปฏิบัติการที่ต้องการได้ และอีกจุดเด่นของ Open Kinect คือมีภาษาที่ใช้พัฒนาได้หลายภาษา เช่น C, C++, Python, Java, Java Script, C#, Action Script เป็นต้น

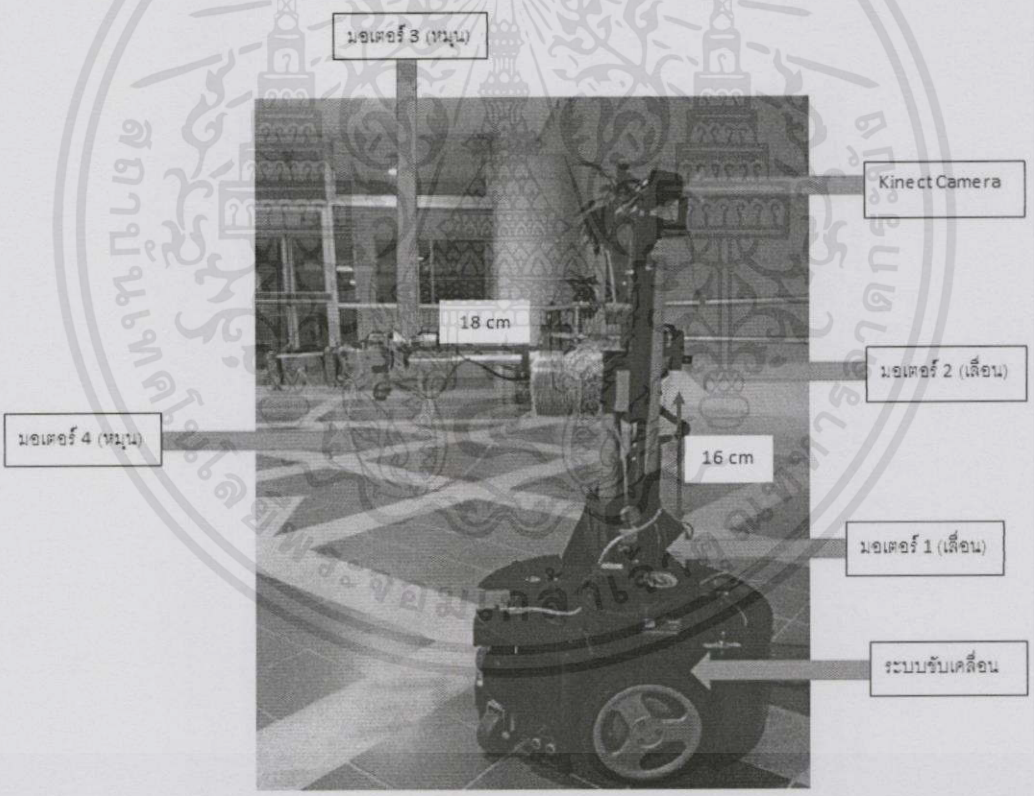
บทที่ 3

หุ่นยนต์และโปรแกรมที่ใช้ในการทดลอง

ในบทนี้จะกล่าวถึง หุ่นยนต์ที่ใช้ในการทดลอง การวิเคราะห์จลนศาสตร์ของแขนกล โปรแกรมที่ใช้ในการทดลอง และการทำงานของโปรแกรม

3.1 หุ่นยนต์ที่ใช้ในการทดลอง

ในการทำโครงงานนี้ใช้หุ่นยนต์ Cira ในการทดลองโดยได้รับการออกแบบและสร้างจากอาจารย์สองเมือง นันทขว้าง อาจารย์ประจำภาควิชาวิศวกรรมการวัดและควบคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และทีม หุ่นยนต์ Cira มีส่วนประกอบของแขนกล กล้อง Kinect และระบบขับเคลื่อน ดังแสดงรูปที่ 3.1



รูปที่ 3.1 หุ่นยนต์ Cira

3.1.1 แขนกล

แขนกลประกอบมอเตอร์ 4 ตัว โดยมีการทำงานคือ มอเตอร์ (1) เป็นแบบเลื่อนขึ้น-ลง ในแนวดิ่ง ระยะการทำงาน 0 – 16 cm. อัตราการหมุน 10 mm/s มอเตอร์ (2) เป็นแบบเลื่อนเข้า – ออก ในแนวระนาบ มีระยะทำงาน 0 – 18 cm. อัตราการหมุน 10 mm/s มอเตอร์ (3) เป็นแบบ

หมุน มีระยะทำงาน 0° - 360° รอบแนวระนาบ อัตราการหมุน = 5 rpm และมอเตอร์ (4) เป็นแบบหมุน มีระยะทำงาน 0° - 360° รอบแนวดิ่ง อัตราการหมุน = 5 rpm

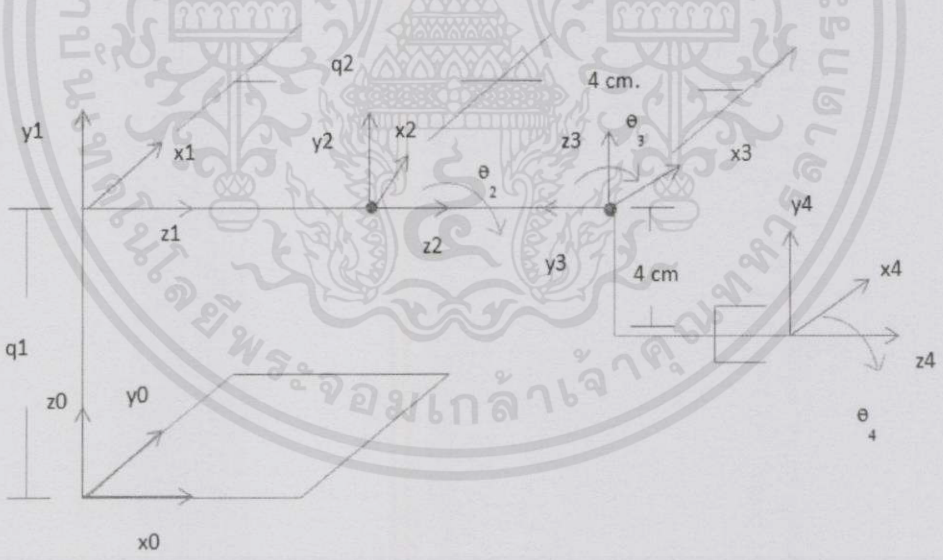
มอเตอร์แขนกลทั้ง 4 ตัวของหุ่นยนต์มีการเชื่อมต่อกับบอร์ด Arduino Atmega1280 และเชื่อมต่อบอร์ด Arduino กับคอมพิวเตอร์ผ่านสาย USB สำหรับการรับคำสั่งขับเคลื่อนมอเตอร์ ซึ่งจะสั่งผ่านคอมพิวเตอร์ โดยคำสั่งจะรับเป็นตัวอักษรอักขระในการเริ่มทำงานและรับตัวอักษรอักขระสำหรับคำสั่งหยุดทำงาน

3.1.2 Kinect Camera

กล้อง Kinect ของหุ่นยนต์ Cira ติดตั้งสูงจากพื้น เท่ากับ 98 cm. มุมเงยของกล้องปรับให้เป็น 0° ซึ่งกล้อง Kinect เชื่อมต่อกับคอมพิวเตอร์ การใช้งานเป็นการดึงไลบรารีของ Window SDK ที่พัฒนาโดย Microsoft มาใช้เขียนโปรแกรมควบคุมการเคลื่อนที่ เพื่อใช้ในการตรวจจับตำแหน่งร่างกายของมนุษย์

3.2 การวิเคราะห์จลนศาสตร์แขนกล

จากทฤษฎีที่ได้กล่าวในบทที่ 2 สามารถวิเคราะห์จลนศาสตร์แขนกลของหุ่นยนต์ Cira ได้ โดยมี Link Coordinate ของแขนกลหุ่นยนต์ Cira แสดงในรูปที่ 3.2 และสามารถหา D-H Parameters ได้ดังตารางที่ 3.1



รูปที่ 3.2 Link Coordinate ของแขนกลหุ่นยนต์ Cira

ตารางที่ 3.1 D-H Parameters ของแขนกลหุ่นยนต์ Cira

Link	z		x		Home
	$\theta°$	D (cm)	a (cm)	$\alpha°$	
1	90	q_1	0	90	48 cm
2	0	q_2	0	0	23 cm
3	q_3	3	0	0	90°
4	q_4	11.5	-3	0	0°

เมื่อหา D-H Parameters ได้แล้ว สามารถนำค่าต่างๆ มาเขียนเป็นสมการ Forward Kinematic และนำไปใช้คำนวณหาสมการ Inverse Kinematic ต่อไป ได้ดังนี้

$$T_{base}^{tool} = \begin{bmatrix} cos_4 & -sin_4 & 0 & Pcos_4 + q_2 + 3 \\ cos_3 sin_4 & cos_{34} & -sin_3 & Pcos_3 sin_4 + 3sin_3 \\ sin_{34} & sin_3 cos_4 & cos_3 & Psin_{34} - 3cos_4 + q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

สมการ Forward Kinematic

$$\begin{aligned} x &= Pcos_4 + q_2 + 3 \\ y &= Pcos_3 sin_4 + 3sin_3 \\ z &= Psin_{34} - 3cos_4 + q_1 \end{aligned} \quad (3.2)$$

สมการ Inverse Kinematic

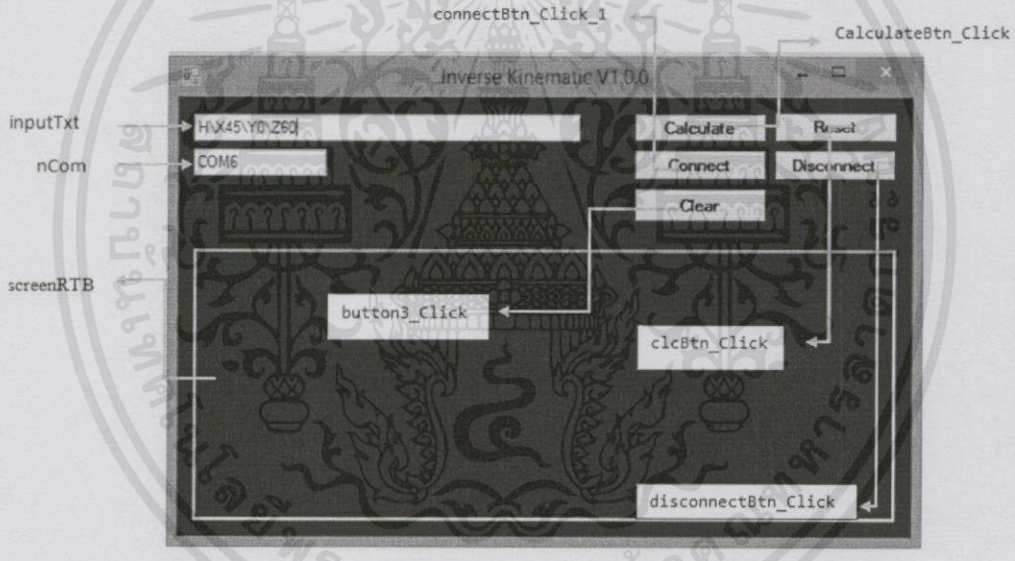
$$\begin{aligned} q_1 &= z - Psin_{34} + 3sin_3 \\ q_2 &= x - Pcos_4 - 3 \\ q_3 &= tan^{-1} \frac{(z-q_1)Psin_4 + 3Y}{PYsin_4 + 3q_1 - Z} \\ q_4 &= cos^{-1} \frac{X - q_2 - 3}{P} \end{aligned} \quad (3.3)$$

3.3 โปรแกรมที่ใช้ในการทดลอง

โปรแกรมที่ใช้ในการทดลองหุ่นยนต์ Cira ประกอบด้วยโปรแกรม 2 ส่วน คือ โปรแกรมควบคุมแขนกล และโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

3.3.1 โปรแกรมควบคุมแขนกล

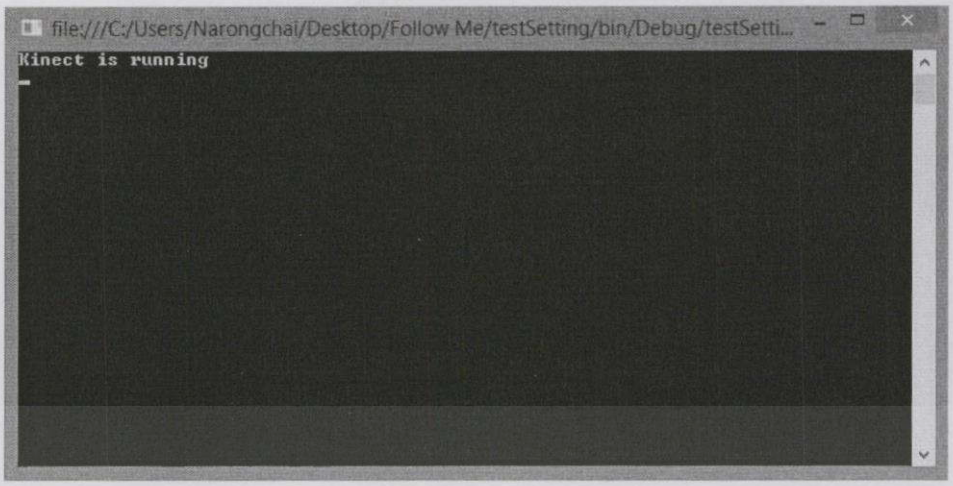
From ในการป้อนข้อมูลในพิกัด 3 มิติ ดังรูปที่ 3.3 หลักการใช้งานคือ ป้อนพิกัด X, Y, Z ตาม Range ที่แขนกลสามารถหยิบได้ในหน้าต่าง inputTxt ซึ่งแบ่งเป็น 2 กรณี คือ โหมด V (Vertical) และโหมด H (Horizontal) เมื่อป้อนค่าแล้วจึงกดปุ่ม Connect เพื่อเชื่อมต่อ และสามารถกดปุ่ม Calculate เพื่อคำนวณค่าที่ได้จะแสดงในหน้าต่าง ScreenRTB แล้วส่งค่าที่ได้ไปยังบอร์ด Arduino เพื่อบังคับแขนกล ส่งไปยัง Com ต่างๆ ซึ่งระบุได้ในหน้าต่าง nCom ส่วนปุ่ม Reset จะเป็นการสั่ง Reset แขนกลให้กลับมาอยู่ในตำแหน่งเดิมหลังการหยิบวัตถุ ปุ่ม Clear เป็นการ Clear ค่าที่แสดงใน ScreenRTB เมื่อเสร็จสิ้นการทำงานแล้วจึงสามารถกดปุ่ม Disconnect เพื่อตัดการเชื่อมต่อ



รูปที่ 3.3 โปรแกรมควบคุมแขนกล

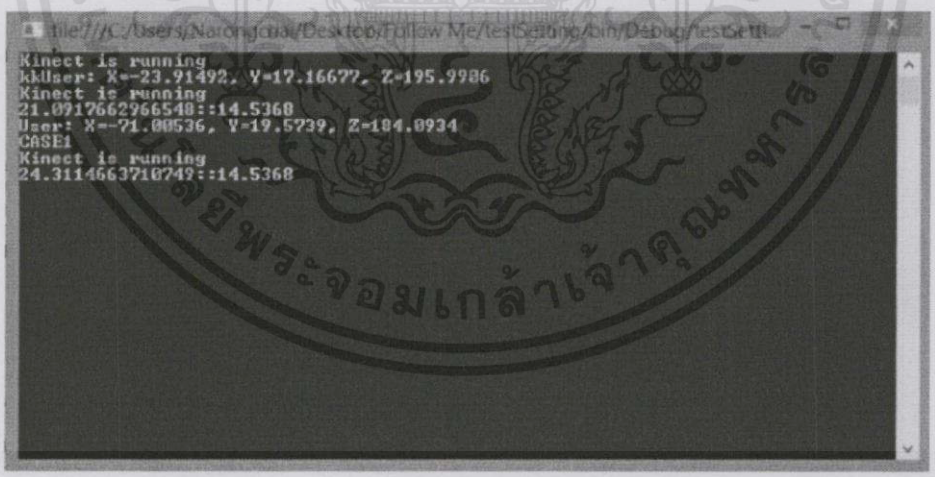
3.3.2 โปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

หลักการทำงานของโปรแกรมเป็นการกำหนดจุดของตำแหน่งโครงกระดูกบนร่างกาย เพื่อให้กล้อง Kinect ตรวจจับ ในโปรแกรมที่ได้ออกแบบคือ ส่วน Center Shoulder เมื่อทำการ Run โปรแกรม แล้วกล้อง Kinect สามารถตรวจจับได้แล้วจะปรากฏข้อความว่า “Kinect is Running” เพื่อแสดงว่าตอนนี้กล้อง Kinect กำลังตรวจจับตำแหน่งของร่างกายอยู่ ดังรูปที่ 3.4



รูปที่ 3.4 การแสดงผลเมื่อกล้อง Kinect ตรวจจับตำแหน่งของร่างกาย

เมื่อกล้อง Kinect ตรวจจับตำแหน่งของร่างกายได้แล้ว ค่าที่ได้จากการตรวจจับก็คือ พิกัดตำแหน่ง X, Y, Z ของตำแหน่งที่ตรวจจับ เมื่อได้ค่าของพิกัดมาแล้วก็จะนำค่าทั้ง 3 มาคำนวณผ่าน Algorithm ของโปรแกรมที่ได้ออกแบบขึ้น ทำให้สามารถรู้ระยะห่างระหว่างบุคคลที่หุ่น Cira ตรวจจับ แล้วจึงส่งคำสั่งการเคลื่อนที่ไปยังบอร์ด Arduino ของระบบขับเคลื่อนเพื่อสั่งให้หุ่นยนต์ Cira เคลื่อนที่ติดตามบุคคล รูปที่ 3.5 เป็นการแสดงค่าพิกัด X, Y, Z ที่กล้อง Kinect ตรวจจับ

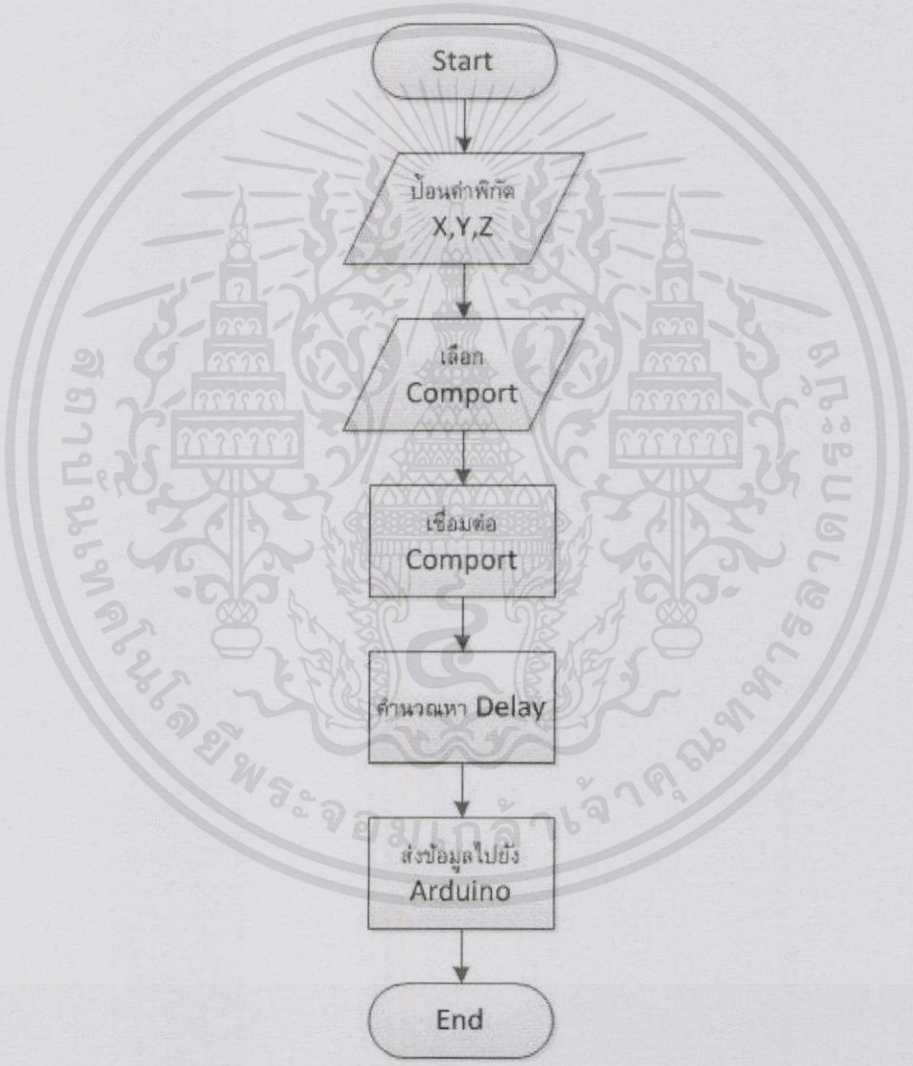


รูปที่ 3.5 การแสดงผลค่าพิกัด X, Y, Z ที่กล้อง Kinect ตรวจจับได้

3.4 แผนผังลำดับงานของโปรแกรม

3.4.1 โปรแกรมควบคุมแขนกล

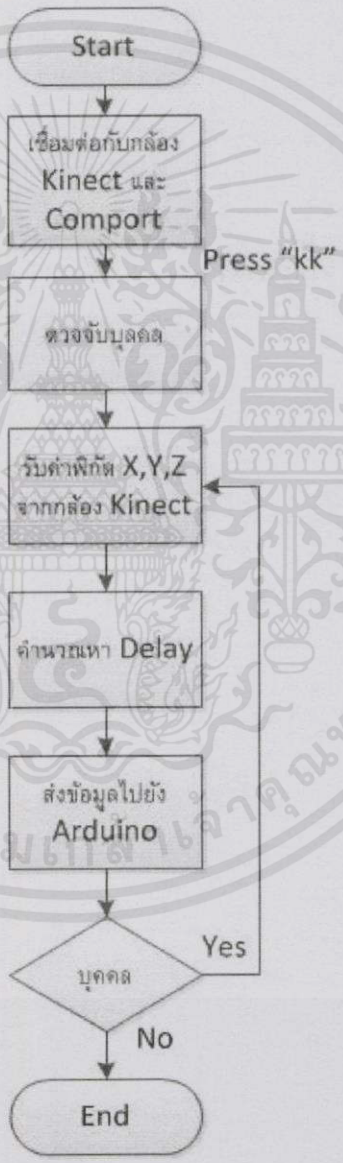
โปรแกรมมีการทำงาน โดยเริ่มจากการป้อนพิกัดที่ต้องการที่จะให้แขนกลเคลื่อนที่ไป แล้วเลือกการเชื่อมต่อ Comport ของ Arduino กับคอมพิวเตอร์ให้ตรงกัน เมื่อเชื่อมต่อเรียบร้อย โปรแกรมจะคำนวณได้ Delay เป็นตัวเลขแล้วเก็บข้อมูลไว้ แล้วจะมีการส่งคำสั่งเคลื่อนที่มอเตอร์ของแขนกลที่เป็นตัวอักษรอักขระไปยัง Arduino และใช้การหน่วงเวลาที่คำนวณ เป็นตัวหน่วงเวลาก่อนที่จะส่งคำสั่งหยุดการทำงานมอเตอร์ของแขนกลที่เป็นตัวอักษรอักขระให้หุ่นยนต์อีกครั้งหนึ่ง แสดงดังรูปที่ 3.6



รูปที่ 3.6 แผนผังการทำงานของโปรแกรมควบคุมแขนกล

3.4.2 โปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

การทำงานของโปรแกรมมีการทำงานเริ่มจากเชื่อมต่อกับกล้อง Kinect และเชื่อมต่อ Arduino กับคอมพิวเตอร์ ในการทำงานครั้งแรกต้องกดปุ่ม k บนแป้นพิมพ์ 2 ครั้ง โปรแกรมจะตรวจจับและรับพิกัดของบุคคลจากกล้อง Kinect จึงคำนวณได้ Delay เป็นตัวเลขแล้วเก็บข้อมูลไว้ แล้วจะมีการส่งคำสั่งขับเคลื่อนที่เป็นตัวอักษรอักขระไปยัง Arduino และใช้การหน่วงเวลาที่คำนวณเป็นตัวหน่วงเวลาก่อนที่จะส่งคำสั่งหยุดขับเคลื่อนที่เป็นตัวอักษรอักขระให้หุ่นยนต์อีกครั้งหนึ่ง และในการทำงานครั้งต่อๆ ไปกล้องจะตรวจสอบหาบุคคล ถ้ามีบุคคลโปรแกรมจะเริ่มต้นทำงานตามเดิม ถ้าไม่มีบุคคลโปรแกรมก็จะไม่มีการทำงาน แสดงดังรูปที่ 3.7



รูปที่ 3.7 แผนผังการทำงานของโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

บทที่ 4

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองและผลการทดลองในการเขียนโปรแกรมควบคุมแขนกลและโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่ โดยโปรแกรมควบคุมแขนกลมีการทำงานใน 2 โหมด คือโหมด V และโหมด H ส่วนโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่จะใช้พิกัดจากกล้อง Kinect เพื่อขับเคลื่อนหุ่นยนต์ให้เคลื่อนที่ไปยังเป้าหมายได้ รายละเอียดการทดลองมีดังนี้

1. การทดลองโปรแกรมควบคุมแขนกลในโหมด V
2. การทดลองโปรแกรมควบคุมแขนกลในโหมด H
3. การทดลองการรับพิกัดของโปรแกรมประมวลผลภาพ
4. การทดลองสมรรถภาพของระบบขับเคลื่อน

4.1 การทดลองโปรแกรมควบคุมแขนกลในโหมด V

การทดลองนี้เป็นการทดลองให้แขนกลไปหยิบวัตถุในแนวตั้ง แสดงดังรูปที่ 4.1 โดยให้พิกัดสมมติในแกน X และแกน Z จำนวนทั้งหมด 10 จุด โดยมีระยะการทำงานในแกน X คือ 38.0 – 54.8 cm และในแกน Z เท่ากับ 70.4 – 84.0 cm ทดลองจำนวนจุดละ 10 ครั้ง เพื่อวัดความแม่นยำของแขนกลในการเคลื่อนที่ไปยังพิกัดเป้าหมาย ในตารางที่ 4.1 แสดงค่าพารามิเตอร์ต่างๆ ของแขนกลและตารางที่ 4.2 แสดงผลการทดลองเฉลี่ยความคลาดเคลื่อนในแต่ละพิกัด และความคลาดเคลื่อนเฉลี่ย

รูปที่ 4.1 วัตถุวางในแนวตั้ง

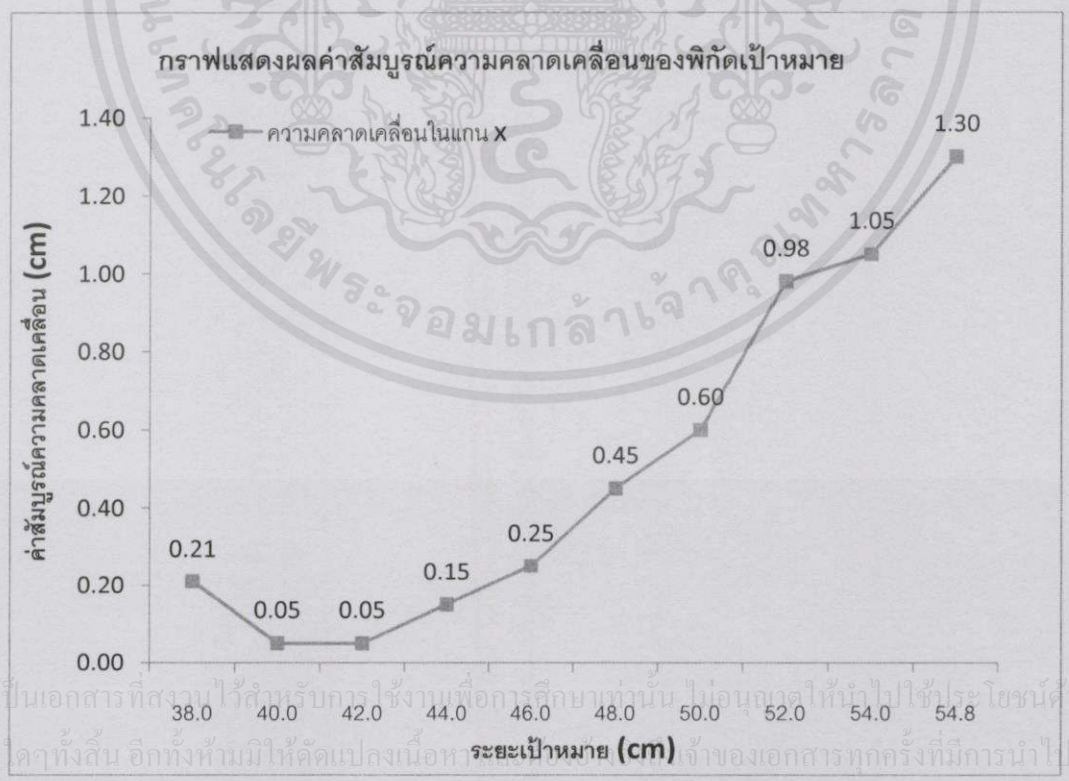
ตารางที่ 4.1 พารามิเตอร์ของแขนกลในโหมด V

พิกัดเป้าหมาย (x, y, z)	$q_1(cm)$	$q_2(cm)$	$q_3(cm)$	$q_4(cm)$
(38.0, 0, 70.4)	1.4	0.9	0°	0°
(40.0, 0, 72.0)	3.0	2.9	0°	0°
(42.0, 0, 74.0)	5.0	4.9	0°	0°
(44.0, 0, 76.0)	7.0	6.9	0°	0°
(46.0, 0, 78.0)	9.0	8.9	0°	0°
(48.0, 0, 80.0)	11.0	10.9	0°	0°
(50.0, 0, 82.0)	13.0	12.9	0°	0°
(52.0, 0, 82.0)	13.0	14.9	0°	0°
(54.0, 0, 84.0)	15.0	16.9	0°	0°
(54.8, 0, 84.0)	15.0	17.7	0°	0°

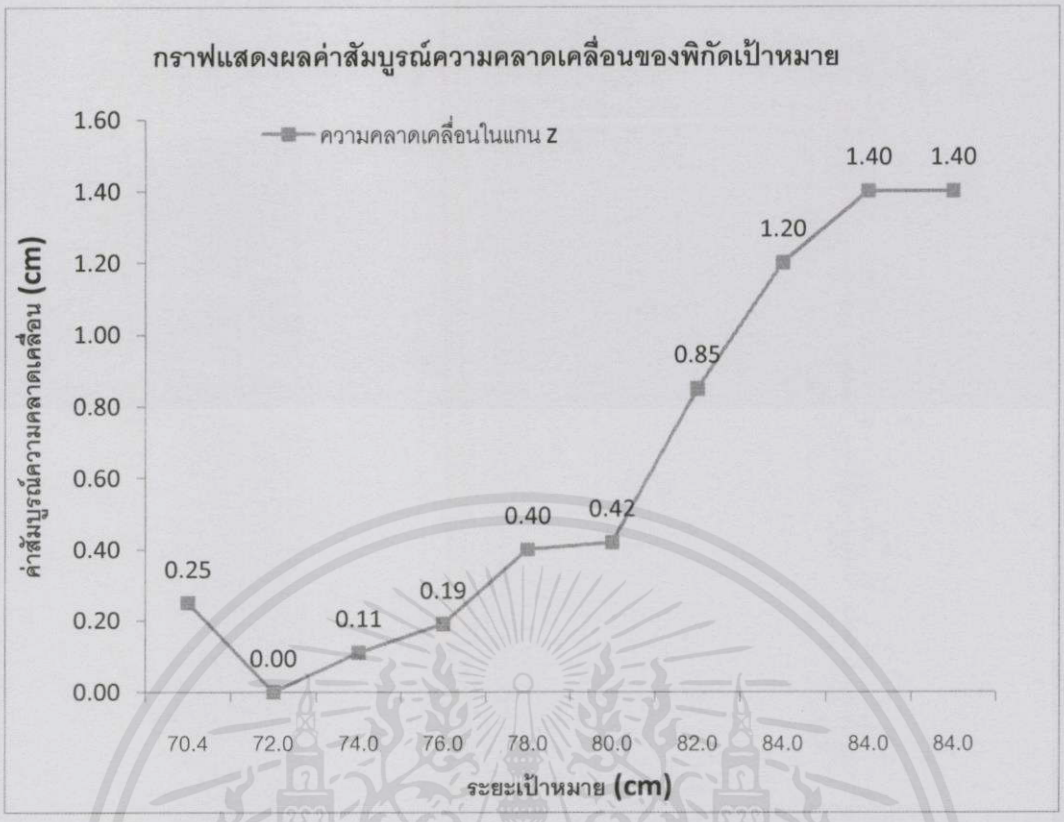
ตารางที่ 4.2 ผลการทดลองโปรแกรมควบคุมแขนกลในโหมด V

จุด	พิกัดเป้าหมาย		ผลการทดลองเฉลี่ย		ค่าสัมบูรณ์ความคลาดเคลื่อน	
	x(cm)	z(cm)	x(cm)	z(cm)	x(cm)	z(cm)
1	38.00	70.40	38.21	70.15	0.21	0.25
2	40.00	72.00	39.95	72.00	0.05	0.00
3	42.00	74.00	42.05	73.89	0.05	0.11
4	44.00	76.00	43.85	75.81	0.15	0.19
5	46.00	78.00	45.75	77.60	0.25	0.40
6	48.00	80.00	47.55	79.58	0.45	0.42
7	50.00	82.00	49.40	81.15	0.60	0.85
8	52.00	84.00	51.02	82.80	0.98	1.20
9	54.00	84.00	52.95	82.60	1.05	1.40
10	54.80	84.00	53.50	82.60	1.30	1.40
ความคลาดเคลื่อนเฉลี่ย					**0.509**	**0.622**

ผลจากการทดลองทั้งหมดมีความคลาดเคลื่อนเฉลี่ย ในแกน X เท่ากับ 0.509 cm และในแกน Z เท่ากับ 0.622 cm ในรูปที่ 4.2 แสดงกราฟความคลาดเคลื่อนในแกน X และรูปที่ 4.3 แสดงกราฟความคลาดเคลื่อนในแกน Z



รูปที่ 4.2 ค่าสัมบูรณ์ความคลาดเคลื่อนในแกน X ของโหมด V



รูปที่ 4.3 ค่าสัมบูรณ์ความคลาดเคลื่อนในแกน Z ของโหมด V

4.2 การทดลองโปรแกรมควบคุมแขนกลในโหมด H

การทดลองนี้เป็นการทดลองให้แขนกลไปหยิบวัตถุในแนวนอน แสดงดังรูปที่ 4.4 โดยให้พิกัดสมมติในแกน X และแกน Z จำนวนทั้งหมด 10 จุด โดยมีระยะการทำงานในแกน X คือ 25.0 – 41.0 cm และในแกน Z เท่ากับ 62.5 – 69.0 cm ทดลองจำนวนจุดละ 10 ครั้ง เพื่อวัดความแม่นยำของแขนกลในการเคลื่อนที่ไปยังพิกัดเป้าหมาย ในตารางที่ 4.3 แสดงค่าพารามิเตอร์ต่างๆ ของแขนกล และตารางที่ 4.4 แสดงผลการทดลองเฉลี่ยความคลาดเคลื่อนในแต่ละพิกัด และความคลาดเคลื่อนเฉลี่ย

รูปที่ 4.4 วัตถุวางในแนวนอน

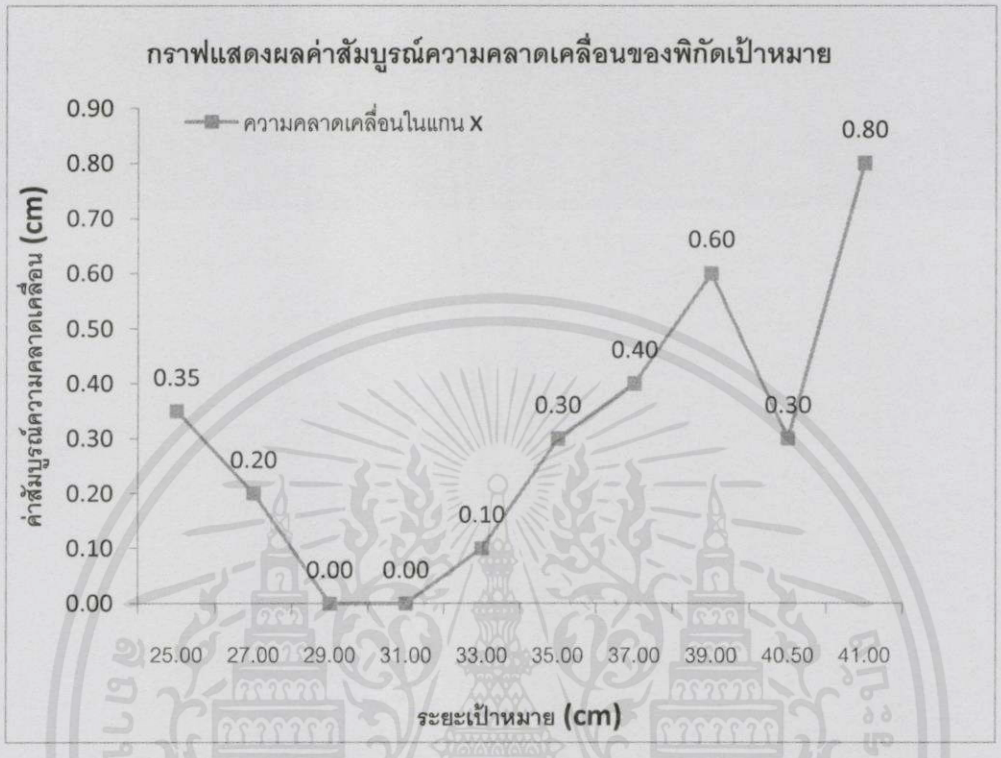
ตารางที่ 4.3 พารามิเตอร์ของแขนกลในโหมด H

พิกัดเป้าหมาย (x, y, z)	$q_1(cm)$	$q_2(cm)$	$q_3(cm)$	$q_4(cm)$
(25.0, 0, 62.5)	1.5	0.8	-90°	90°
(27.0, 0, 64.0)	3.0	2.8	0°	0°
(29.0, 0, 65.5)	4.5	4.8	0°	0°
(31.0, 0, 67.0)	6.0	6.8	0°	0°
(33.0, 0, 68.5)	7.5	8.8	0°	0°
(35.0, 0, 69.0)	8.0	10.8	0°	0°
(37.0, 0, 69.0)	8.0	12.8	0°	0°
(39.0, 0, 69.0)	8.0	14.8	0°	0°
(40.5, 0, 69.0)	8.0	16.3	0°	0°
(41.0, 0, 69.0)	8.0	16.8	0°	0°

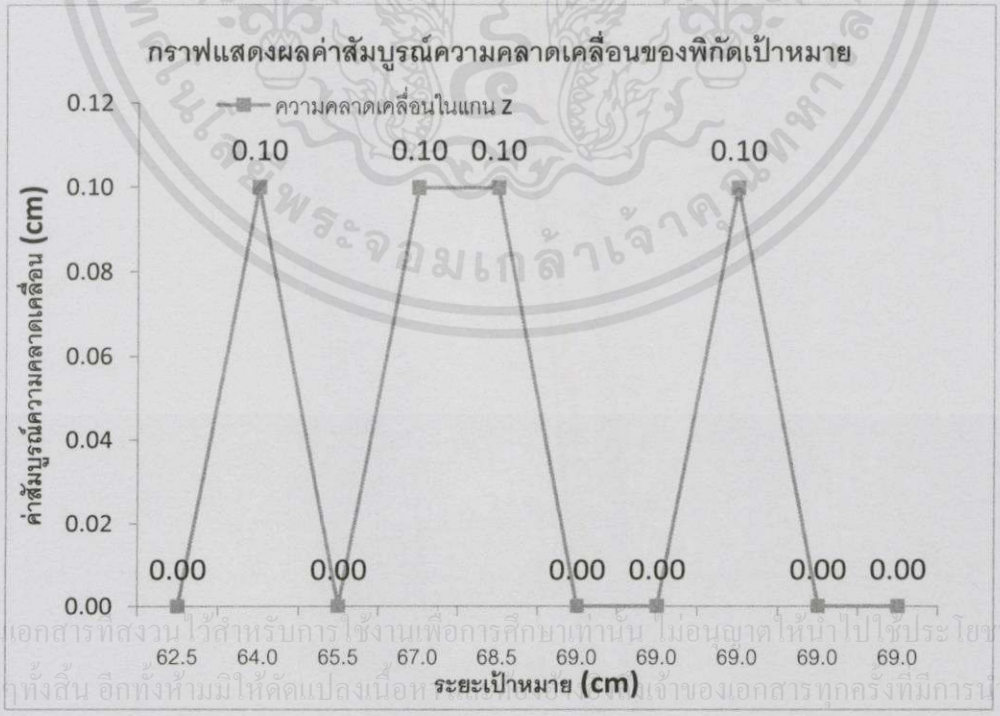
ตารางที่ 4.4 ผลการทดลองโปรแกรมควบคุมแขนกลในโหมด H

ครั้งที่	พิกัดเป้าหมาย		ผลการทดลองเฉลี่ย		ค่าสัมบูรณ์ความคลาดเคลื่อน	
	x(cm)	z(cm)	x(cm)	z(cm)	x(cm)	z(cm)
1	25.00	62.50	25.35	62.50	0.35	0.00
2	27.00	64.00	27.20	64.10	0.20	0.10
3	29.00	65.50	29.00	65.50	0.00	0.00
4	31.00	67.00	31.00	67.10	0.00	0.10
5	33.00	68.50	32.90	68.40	0.10	0.10
6	35.00	69.00	34.70	69.00	0.30	0.00
7	37.00	69.00	36.60	69.00	0.40	0.00
8	39.00	69.00	38.40	68.90	0.60	0.10
9	40.50	69.00	40.20	69.00	0.30	0.00
10	41.00	69.00	40.20	69.00	0.80	0.00
ความคลาดเคลื่อนเฉลี่ย					**0.305**	**0.040**

ผลจากการทดลองทั้งหมดมีความคลาดเคลื่อนเฉลี่ย ในแกน X เท่ากับ 0.305 cm และในแกน Z เท่ากับ 0.040 cm ในรูปที่ 4.5 แสดงกราฟความคลาดเคลื่อนในแกน X และรูปที่ 4.6 แสดงกราฟความคลาดเคลื่อนในแกน Z ของแต่ละพิกัด



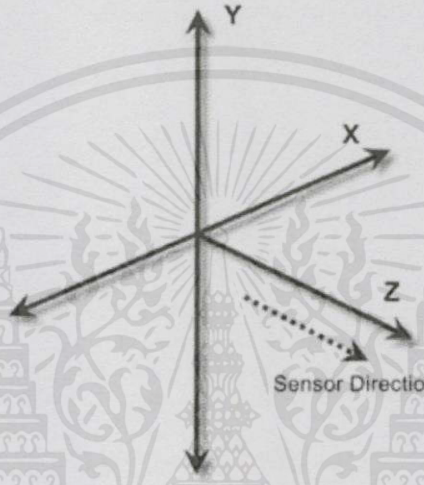
รูปที่ 4.5 ค่าสัมบูรณ์ความคลาดเคลื่อนในแกน X ของโหมด H



รูปที่ 4.6 ค่าสัมบูรณ์ความคลาดเคลื่อนในแกน Z ของโหมด H

4.3 การทดลองการรับพิกัดของโปรแกรมประมวลผลภาพ

การทดลองนี้เป็นการทดลองการจับภาพของกล้อง Kinect โดยให้บุคคลยืนตรงกลางของกล้อง Kinect และยืนห่างเป็นระยะต่างๆ โดยอยู่ในระยะการทำงานของกล้อง Kinect ซึ่งพิกัดในแกน Z เป็นระยะห่างระหว่างบุคคลและกล้อง Kinect และพิกัดในแกน X เป็นระยะทางซ้ายมือของกล้อง Kinect แสดงพิกัดการมองเห็นของกล้อง Kinect ในรูปที่ 4.7 ทดลองจุดละ 10 ครั้ง เพื่อให้ทราบถึงความแม่นยำของการตรวจจับบุคคลโดยใช้กล้อง Kinect ซึ่งผลการทดลองจะแสดงความคลาดเคลื่อนเฉลี่ยในแกน Z และแกน X แสดงในตารางที่ 4.5

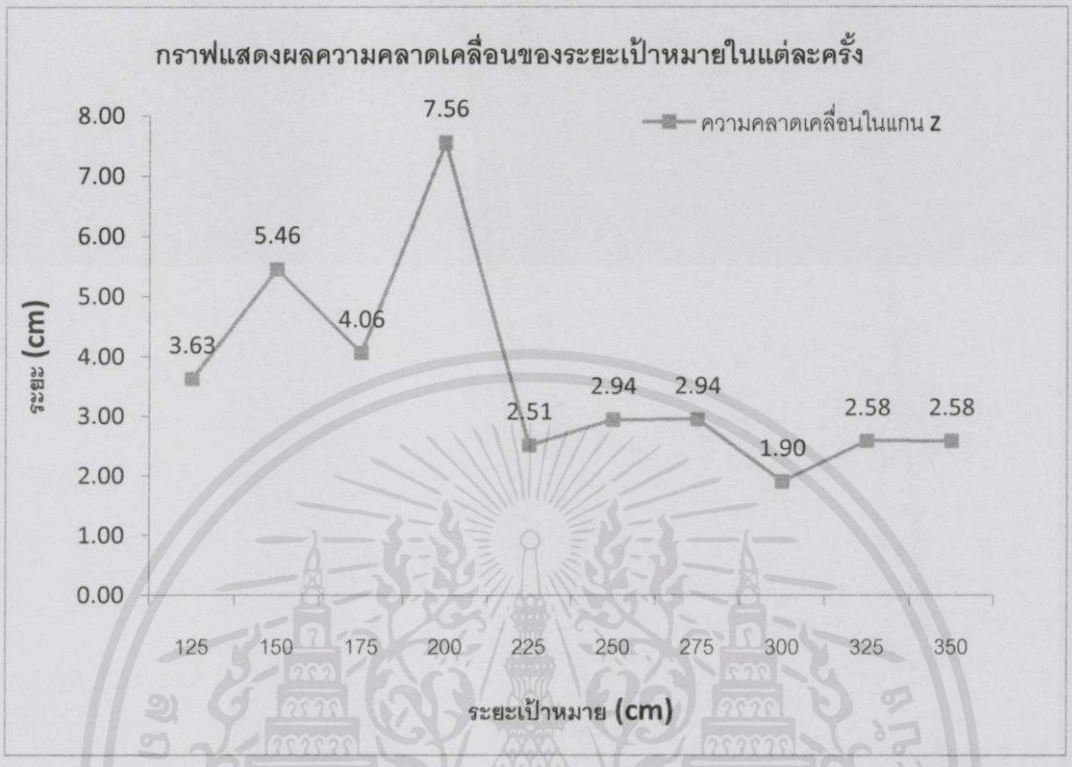


รูปที่ 4.7 พิกัดการมองเห็นของกล้อง Kinect

ตารางที่ 4.5 ผลการทดลองการรับพิกัดของโปรแกรมประมวลผลภาพ

จุด	พิกัดเป้าหมาย		ผลการทดลองเฉลี่ย		ค่าสัมบูรณ์ความคลาดเคลื่อน	
	x(cm)	z(cm)	x(cm)	z(cm)	x(cm)	z(cm)
1	0.00	125.00	0.38	128.63	0.38	3.63
2	0.00	150.00	1.62	155.46	1.62	5.46
3	0.00	175.00	0.47	179.06	0.47	4.06
4	0.00	200.00	1.73	207.56	1.73	7.56
5	0.00	225.00	2.41	227.51	2.41	2.51
6	0.00	250.00	0.40	252.94	0.40	2.94
7	0.00	275.00	0.70	277.94	0.70	2.94
8	0.00	300.00	0.84	301.90	0.84	1.90
9	0.00	325.00	1.09	327.58	1.09	2.58
10	0.00	350.00	0.86	352.58	0.86	2.58
ความคลาดเคลื่อนเฉลี่ย					**1.049**	**3.615**

ผลจากการทดลองมีความคลาดเคลื่อนเฉลี่ยในแกน X เท่ากับ 1.049 และในแกน Z เท่ากับ 3.615 รูปที่ 4.8 แสดงค่าความคลาดเคลื่อนในแกน Z ของแต่ละพิกัด



รูปที่ 4.8 ค่าสัมบูรณ์ความคลาดเคลื่อนในแกน Z ของโปรแกรมประมวลผลภาพ

4.4 การทดลองสมรรถภาพของระบบขับเคลื่อน

การทดลองนี้เป็นการทดลองระบบขับเคลื่อนของหุ่นยนต์ โดยให้หุ่นยนต์เดินหน้าตรง หมุนซ้าย หมุนขวา และถอยหลังตรง บริเวณลานเอนกประสงค์บนตึก ECC คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เพื่อวัดหาค่าหน่วงเวลาของระบบขับเคลื่อน ซึ่งนำไปใช้ในการสั่งการให้ระบบขับเคลื่อนทำงาน และหยุดการทำงาน

4.4.1 การทดลองเดินหน้าตรง

ให้หุ่นยนต์ขับเคลื่อนเดินหน้าตรง ด้วยความเร็ว 50% เป็นระยะทาง 90 cm เป็นจำนวน 10 ครั้ง โดยมีผลการทดลองแสดงดังตารางที่ 4.6 เพื่อนำไปคำนวณหาเวลาที่ใช้ในการเคลื่อนที่เดินหน้าระยะทาง 1 cm ของระบบขับเคลื่อนหุ่นยนต์ และนำไปใช้ในการคำนวณค่าหน่วงเวลาการเคลื่อนที่ในกรณีหุ่นยนต์เดินหน้าตรงต่อไป

ตารางที่ 4.6 ผลการทดลองเดินหน้าตรง

ครั้งที่	เวลา(s)
1	5.39
2	5.45
3	5.46
4	5.48
5	5.41
6	5.56
7	5.43
8	5.39
9	5.46
10	5.45
เวลาเฉลี่ย(s)	**5.448**

ผลการทดลองได้ระยะเวลาของระบบขับเคลื่อนในการเคลื่อนที่เดินหน้าเป็นระยะทาง 90 cm เท่ากับ 5.448 วินาที

4.4.2 การทดลองหมุนซ้าย

ให้หุ่นยนต์ขับเคลื่อนหมุนซ้ายด้วยความเร็ว 50 เปอร์เซ็นต์ เป็นระยะ 360° เป็นจำนวน 10 ครั้ง โดยมีผลการทดลองแสดงดังตารางที่ 4.7 เพื่อนำไปคำนวณหาเวลาในการเคลื่อนที่หมุนซ้าย 1° ของระบบขับเคลื่อนหุ่นยนต์ และนำไปใช้ในการคำนวณค่าหน่วงเวลาการเคลื่อนที่ในกรณีหุ่นยนต์หันซ้ายต่อไป

ตารางที่ 4.7 ผลการทดลองหมุนซ้าย

ครั้งที่	เวลา(s)
1	9.33
2	9.34
3	9.36
4	9.38
5	9.2
6	9.25
7	9.23
8	9.35
9	9.29
10	9.33
เวลาเฉลี่ย(s)	**9.306**

ผลการทดลองได้ระยะเวลาของระบบขับเคลื่อนในการเคลื่อนที่หมุนซ้ายเท่า 360° เท่ากับ 9.306 วินาที

4.4.3 การทดลองหมุนขวา

ให้หุ่นยนต์ขับเคลื่อนหมุนขวาด้วยความเร็ว 50 เปอร์เซ็นต์ เป็นระยะ 360° เป็นจำนวน 10 ครั้ง โดยมีผลการทดลองแสดงดังตารางที่ 4.8 เพื่อนำไปคำนวณหาเวลาในการเคลื่อนที่หมุนขวา 1° ของระบบขับเคลื่อนหุ่นยนต์ และนำไปใช้ในการคำนวณค่าหน่วงเวลาการเคลื่อนที่ในกรณีหุ่นยนต์หันขวาต่อไป

ตารางที่ 4.8 ผลการทดลองหมุนขวา

ครั้งที่	เวลา(s)
1	8.35
2	8.48
3	8.28
4	8.38
5	8.33
6	8.31
7	8.29
8	8.29
9	8.41
10	8.3
เวลาเฉลี่ย(s)	**8.342**

ผลการทดลองได้ระยะเวลาของระบบขับเคลื่อนในการเคลื่อนที่หมุนขวาเท่า 360 ° เท่ากับ 8.342 วินาที

4.4.4 การทดลองถอยหลังตรง

ให้หุ่นยนต์ขับเคลื่อนถอยหลังตรง เป็นระยะทาง 90 cm เป็นจำนวน 10 ครั้ง โดยมีผลการทดลองแสดงดังตารางที่ 4.9 เพื่อนำไปคำนวณหาเวลาที่ใช้ในการเคลื่อนที่ถอยหลังระยะทาง 1 cm ของระบบขับเคลื่อนหุ่นยนต์ และนำไปใช้ในการคำนวณค่าหน่วงเวลาการเคลื่อนที่ในกรณีหุ่นยนต์ถอยหลังต่อไป

ตารางที่ 4.9 ผลการทดลองถอยหลังตรง

ครั้งที่	เวลา(s)
1	3.35
2	3.31
3	3.36
4	3.36
5	3.35
6	3.35
7	3.38
8	3.38
9	3.43
10	3.34
เวลาเฉลี่ย(s)	**3.361**

ผลการทดลองได้ระยะเวลาของระบบขับเคลื่อนในการเคลื่อนที่ถอยหลังเป็นระยะทาง 90 cm เท่ากับ 3.361 วินาที

บทที่ 5
บทสรุปและวิจารณ์

5.1 บทสรุปและวิจารณ์

โครงงานนี้เป็นการสร้างระบบเพื่อควบคุมแขนกลและการเคลื่อนที่หุ่นยนต์ให้เดินตามผู้ใช้ที่ได้รับข้อมูลมาจาก Microsoft Kinect Sensor โดยใช้ภาษา C# ในการพัฒนา ซึ่งในโครงงานนี้โปรแกรมจะแบ่งออกเป็น 2 โปรแกรม ได้แก่ โปรแกรมควบคุมแขนกล และโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่ ซึ่งโปรแกรมควบคุมแขนกลจะรับพิกัดจากผู้ใช้ แล้วจะทำการคำนวณสมการจลนศาสตร์ของแขนกลจึงส่งตัวอักษรอักขระไปยัง Microcontroller แล้วสั่งงานให้ Motor ขับเคลื่อนทำงาน และโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่จะวัดระยะของผู้ใช้และส่งผ่านข้อมูลในรูปแบบตัวอักษรอักขระไปยัง Microcontroller แล้วสั่งงานให้ Motor ขับเคลื่อนทำงาน โดยโปรแกรมที่พัฒนาขึ้นนี้มีข้อจำกัดต่างๆ ดังนี้

1. เครื่องคอมพิวเตอร์ที่จะสามารถใช้งานโปรแกรมนี้ได้ต้องติดตั้ง .NET framework 4.5 ไว้ในเครื่องซึ่งสามารถดาวน์โหลดได้ทางเว็บไซต์ของ Microsoft

2. เครื่องคอมพิวเตอร์ที่จะสามารถใช้งานโปรแกรมนี้ได้ต้องติดตั้ง Driver ของ Kinect for Windows SDK v1.8 ไว้ในเครื่องก่อน

5.2 ปัญหาที่พบและแนวทางแก้ไข

จากการศึกษาและทำโครงงานนี้เกิดปัญหาคือ โปรแกรม Visual Studio มีการส่งข้อมูลออกไปยัง Arduino ที่ผิดพลาด ทำให้ไม่สามารถควบคุมหุ่นยนต์ได้

ในส่วนปัญหาที่พบได้ทำการแก้ไขโดยพยายามลองกระบวนการของโปรแกรม หลายๆ รูปแบบ

5.3 ข้อเสนอแนะและแนวทางในการค้นคว้าพัฒนา

เนื่องจากในปัจจุบันเทคโนโลยีหุ่นยนต์เข้ามามีบทบาทกับชีวิตประจำวันมากขึ้น ควรศึกษาเพิ่มเติมพัฒนาโปรแกรมให้สามารถใช้ท่าทางในการควบคุมการเคลื่อนที่ของหุ่นยนต์ และให้โปรแกรมประมวลผลภาพสามารถตรวจจับวัตถุและแยกแยะวัตถุแต่ละชนิดได้ เพื่อเป็นแนวทางในการพัฒนาโปรแกรมควบคุมแขนกลเพื่อหยิบจับวัตถุได้ อีกทั้งยังสามารถนำมาเป็นต้นแบบหุ่นยนต์ในประเทศไทยได้อีกด้วย

เอกสารอ้างอิง

[1] พงศ์แสน พิทักษ์วัชระ. **พื้นฐานของหุ่นยนต์ : กลศาสตร์ของหุ่นยนต์แบบอนุกรม**. กรุงเทพมหานคร : บริษัท วี.พริ้นท์ (1991) จำกัด, 2557.

[2] **เจาะลึกโครงสร้างของ Kinect สำหรับ Windows (ตอนที่ 2)**. (Online). แหล่งที่มา : https://kinectasia.wordpress.com/ 1 กุมภาพันธ์ 2558

[3] **Kinect for Windows Architectureh**. (Online). แหล่งที่มา : ttps://msdn.microsoft.com/en-us/library/jj131023.aspx 1 กุมภาพันธ์ 2558

[4] **Kinect Software Review**. (Online). แหล่งที่มา : http://piak.appstack.cc/2013/04/kinect-software-review.html 1 กุมภาพันธ์ 2558

[5] **Microsoft Kinect Sensor**. (Online). แหล่งที่มา : http://mcu56.learninginventions.org/?page_id=244 1 กุมภาพันธ์ 2558

ภาคผนวก

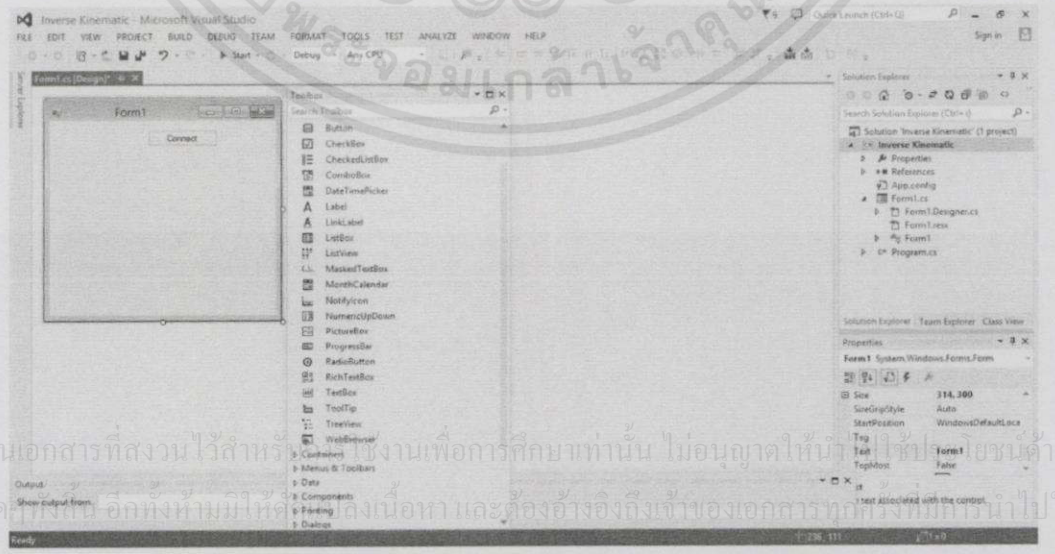
ภาคผนวก ก

การใช้งานโปรแกรม Visual Studio 2013

โปรแกรมควบคุมแขนกล

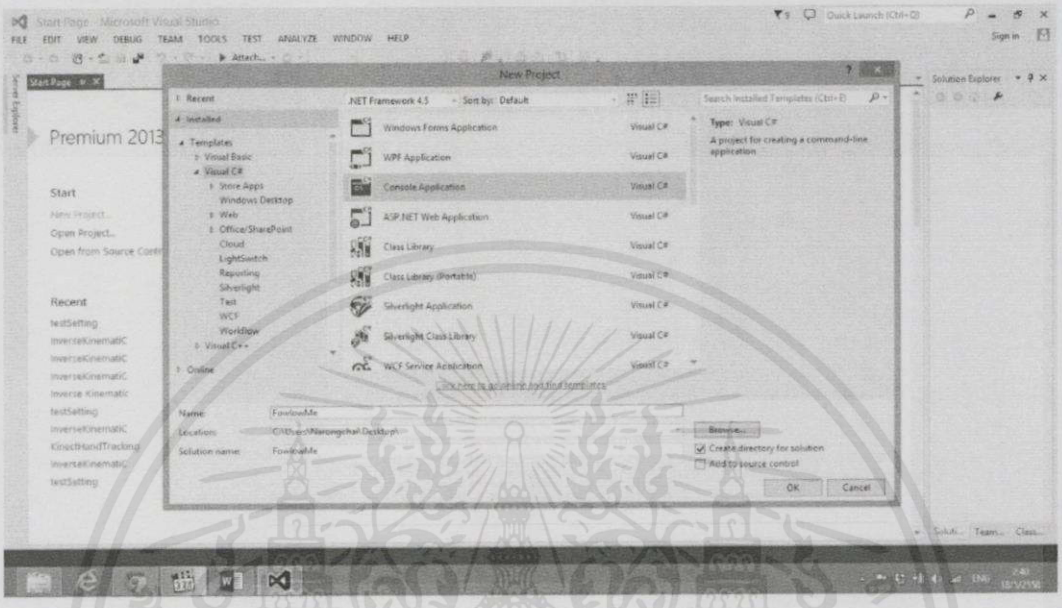
ใช้โปรแกรม Visual Studio 2013 ในการรับคำสั่ง และส่งคำสั่งออกไปยัง Arduino เพื่อบังคับแขนกล ตามค่าที่ป้อนผ่านโปรแกรม สำหรับโปรแกรมที่ใช้ออกแบบเป็น Visual C# แบบ Windows Forms Application

เมื่อเข้าสู่โปรแกรม Visual Studio 2013 เลือก New Project เลือก Windows Forms Application เมื่อโปรแกรมแสดงหน้าต่างขึ้นมา สามารถเลือกใช้ได้จากแถบ View => Toolbox ซึ่งสามารถเลือกเครื่องมือที่ต้องการใช้แล้วลากไปวางยัง Form ดังในรูปที่ 3.3 นั้นเป็นการนำเข้าเครื่องมือ button ไปยัง Form

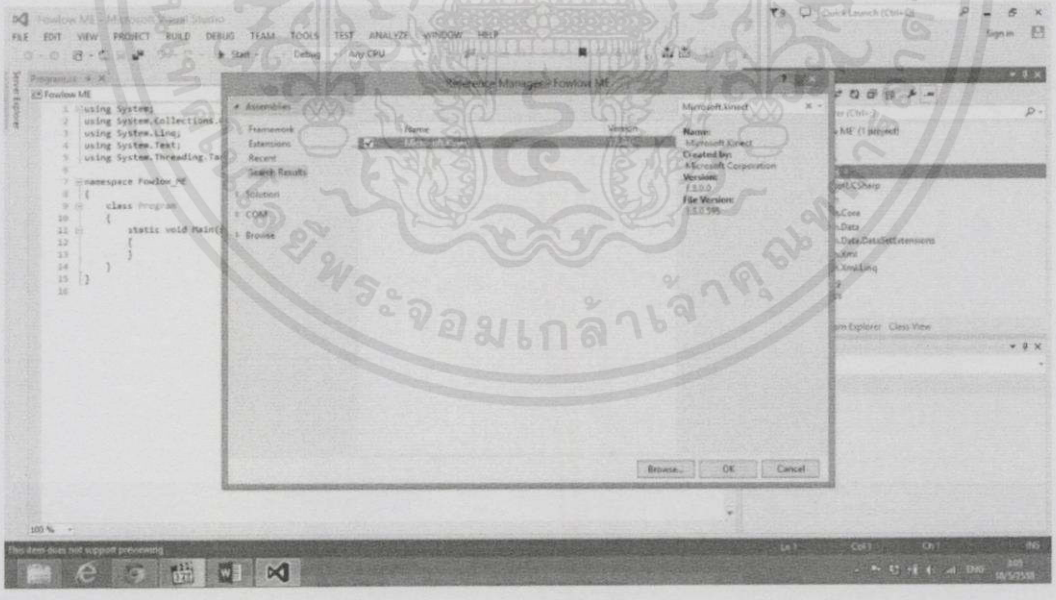


โปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

ใช้โปรแกรม Visual Studio 2013 ในการรับคำสั่ง และส่งคำสั่งออกไปยัง Arduino เพื่อขับเคลื่อนหุ่นยนต์ ตามค่าพิกัดที่รับได้จากล้อง Kinect สำหรับโปรแกรมที่ใช้ออกแบบเป็น Visual C# แบบ Console Application



การ Add References Microsoft.Kinect ในการเรียกใช้งาน Library ของ Windows SDK เพื่อพัฒนาโปรแกรมประมวลผลภาพ



ภาคผนวก ข

โค้ดโปรแกรมควบคุมแขนกล

โปรแกรมควบคุมแขนกล ใช้สำหรับสั่งงานแขนกลให้ไปยังเป้าหมายที่ต้องการ

InverseKinematiC.C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;


namespace InverseKinematic_V1._0
{
   public partial class inverseKiFrm : Form
   {
        public string sendQ1;
        public string sendQ2;
        public string sendQ3;
        public string sendQ4;
        public inverseKiFrm()
        {
           InitializeComponent();
        }

        private void calculateBtn_Click(object sender, EventArgs e)
            {
               string input = inputTxt.Text;
               string[] inputSP = input.Split(new string[] { "\\" }StringSplitOptions.None);
               string axis = inputSP[0];
               double x = Convert.ToDouble(inputSP[1].Replace("X", null));
               double y = Convert.ToDouble(inputSP[2].Replace("Y", null));
               double z = Convert.ToDouble(inputSP[3].Replace("Z", null));
               int DelayQ1 = 740;
               int DelayQ2 = 1500;

               {
               double Q1, Q2, Q3, Q4, QO11, QO12, QO13, QO14, QO21, QO22, QO23, QO24, XTO1,
                      YTO1, ZTO1, XTO2, YTO2, ZTO2, TQ1, TQ2;
               float toolAdjX = 11.5F;
               screenRTB.SelectionColor = Color.White;
               screenRTB.AppendText("\n X = " + Convert.ToString(x) + "\t" + "Y = " +
                       Convert.ToString(y) + "\t" + "Z = " + Convert.ToString(z) + "\n");
                  switch (axis)
                       {
                            case "V":
                            int VoffsetZ = 72;
                            float VoffsetX = 25.6f;
                            if (((x >= 36) && (x <= 54.8)) && ((y >= 0) && (y <= 11.5)) &&
                                                    ((z >= 68) && (z <= 84)))
                             {
                                QO13 = 0 * (Math.PI / 180);
                                QO14 = Math.Asin((y - (3 * Math.Sin(QO13))) / (toolAdjX *
                                Math.Cos(QO13)));
                                QO11 = z - (toolAdjX * Math.Sin(QO13) * Math.Sin(QO14)) + (3
*                               Math.Cos(QO13)) - VoffsetZ;
                                QO12 = x - (toolAdjX * Math.Cos(QO14)) - VoffsetX;
                                XTO1 = QO12 + (toolAdjX * Math.Cos(QO14)) + VoffsetX;
```

```
                    YTO1 = (toolAdjX * Math.Cos(QO13) * Math.Sin(QO14)) + (3 *
                    Math.Sin(QO13));
                    ZTO1 = (toolAdjX * Math.Sin(QO13) * Math.Sin(QO14)) - (3 *
                    Math.Cos(QO13)) + QO11 + VoffsetZ;
                    double errX = (Math.Abs(x - XTO1)) * (100 / x);
                    double errY;
                if (y == 0)
                   {
                    errY = 0;

                   }
                else
                   {
                    errY = (Math.Abs(y - YTO1)) * (100 / y);
                   }
double errZ = (Math.Abs(z - ZTO1)) * (100 / z);
double err = (errX + errY + errZ) / 3;
Q1 = QO11 * 1000;
Q2 = QO12 * 1000;
Q3 = QO13 * (180 / Math.PI);
Q4 = QO14 * (180 / Math.PI);
TQ1 = Math.Ceiling(Q1);
TQ2 = Math.Ceiling(Q2);
screenRTB.SelectionColor = Color.PaleGreen;
screenRTB.AppendText("TQ1 : " + Q1 + "ms" + "\t" + "TQ2 : " + Q2 + "ms" + "\t" + "TQ3
                : " + Q3 + "ms" + "\t" + "TQ4 : " + Q4 + "ms" + "\n");
screenRTB.SelectionColor = Color.PaleVioletRed;
screenRTB.AppendText("================================================================
                =======" + "\n");
          if (((TQ1 + DelayQ1) > 10000) && ((TQ2 + DelayQ2) > 10000))
             {
                String data2 = "CV" + Convert.ToString(TQ1 + DelayQ1) +
                Convert.ToString(TQ2 + DelayQ2) + "0000" + Convert.ToString(Q3) +
                "0000" + Convert.ToString(Q4);
                sport.Write(data2);
             }
          else if (((TQ1 + DelayQ1) > 10000) && ((TQ2 + DelayQ2) < 10000))
             {
                String data3 = "CV" + Convert.ToString(TQ1 + DelayQ1) + "0" +
                Convert.ToString(TQ2 + DelayQ2) + "0000" + Convert.ToString(Q3) +
                "0000" + Convert.ToString(Q4);
                sport.Write(data3);
             }
          else if (((TQ1 + DelayQ1) < 10000) && ((TQ2 + DelayQ2) > 10000))
             {
                String data3 = "CV0" + Convert.ToString(TQ1 + DelayQ1) +
                Convert.ToString(TQ2 + DelayQ2) + "0000" + Convert.ToString(Q3) +
                "0000" + Convert.ToString(Q4);
                sport.Write(data3);
             }
          else
             {
                String data3 = "CV0" + Convert.ToString(TQ1 + DelayQ1) + "0" +
                Convert.ToString(TQ2 + DelayQ2) + "0000" + Convert.ToString(Q3) +
                "0000" + Convert.ToString(Q4);
                sport.Write(data3);
             }
           }
                else
                  {
                     screenRTB.SelectionColor = Color.Red;
                     screenRTB.AppendText("Out of range !!!");
                  }
                break;


                case "H":
                float HoffsetZ = 72.5f;
                float HoffsetX = 24.2f;
                if (((x >= 23) && (x <= 42)) && ((y >= 0) && (y <= 11.5)) && ((z
                     >=     61) && (z <= 69)))
                  {
                    QO13 = (-90) * (Math.PI / 180);
```

```
Q014 = (90) * (Math.PI / 180);
Q011 = z - (toolAdjX * Math.Sin(Q013) * Math.Sin(Q014)) + (3 *
Math.Cos(Q013)) - HoffsetZ;
Q012 = x - (toolAdjX * Math.Cos(Q014)) - HoffsetX;

Q023 = (90) * (Math.PI / 180);
Q024 = (-90) * (Math.PI / 180);
Q021 = z - (toolAdjX * Math.Sin(Q023) * Math.Sin(Q024)) + (3 *
Math.Cos(Q023)) - HoffsetZ;
Q022 = x - (toolAdjX * Math.Cos(Q024)) - HoffsetX;

XTO1 = Q012 + (toolAdjX * Math.Cos(Q014)) + HoffsetX;
YTO1 = (toolAdjX * Math.Cos(Q013) * Math.Sin(Q014)) + (3 *
Math.Sin(Q013));
ZTO1 = (toolAdjX * Math.Sin(Q013) * Math.Sin(Q014)) - (3 *
Math.Cos(Q013)) + Q011 + HoffsetZ;
double errO1X = (Math.Abs(x - XTO1)) * (100 / x);
double errO1Y;
   if (y == 0)
     {
       errO1Y = 0;

     }
   else
     {
       errO1Y = (Math.Abs(y - YTO1)) * (100 / y);
     }

double errO1Z = (Math.Abs(z - ZTO1)) * (100 / z);
double errO1 = (errO1X + errO1Y + errO1Z) / 3;
XTO2 = Q022 + (toolAdjX * Math.Cos(Q024)) + HoffsetX;
YTO2 = (toolAdjX * Math.Cos(Q023) * Math.Sin(Q024)) + (3 *
Math.Sin(Q023));
ZTO2 = (toolAdjX * Math.Sin(Q023) * Math.Sin(Q024)) - (3 *
Math.Cos(Q023)) + Q021 + HoffsetZ;
double errO2X = (Math.Abs(x - XTO2)) * (100 / x);
double errO2Y;
   if (y == 0)
     {
       errO2Y = 0;

     }
   else
     {
       errO2Y = (Math.Abs(y - YTO2)) * (100 / y);
     }

double errO2Z = (Math.Abs(z - ZTO2)) * (100 / z);
double errO2 = (errO2X + errO2Y + errO2Z) / 3;

   if (errO1 > errO2)
     {
       Q1 = Q021 * 1000;
       Q2 = Q022 * 1000;
       Q3 = (Q023 * (180 / Math.PI) * (100 / 3));
       Q4 = (Q024 * (180 / Math.PI) * (100 / 3));
     }
   else if (errO1 == errO2)
     {
       Q1 = Q011 * 1000;
       Q2 = Q012 * 1000;
       Q3 = Q013 * (180 / Math.PI) * (100 / 3);
       Q4 = Q014 * (180 / Math.PI) * (100 / 3);
     }
   else
     {
       Q1 = Q011 * 1000;
       Q2 = Q012 * 1000F;
       Q3 = Q013;
       Q4 = Q014;
     }
screenRTB.SelectionColor = Color.PaleGreen;
```

```
screenRTB.AppendText("TQ1 : " + Q1 + "ms" + "\t" + "TQ2 : " + Q2 + "ms" + "\t" + "TQ3
                    : " + Q3 + "ms" + "\t" + "TQ4 : " + Q4 + "ms" + "\n");
screenRTB.SelectionColor = Color.OrangeRed;
screenRTB.AppendText("<Option 1> Error Average : " + "\t" + Convert.ToString(errO1) +
                    "\n");
screenRTB.SelectionColor = Color.OrangeRed;
screenRTB.AppendText("<Option 2> Error Average : " + "\t" + Convert.ToString(errO2) +
                    "\n");
screenRTB.SelectionColor = Color.PaleVioletRed;
screenRTB.AppendText("================================================================
                    =======" + "\n");

double checkQ1 = Q1 + DelayQ1;
double checkQ2 = Q2 + DelayQ2;
int SetQ3Q4 = 1190;
   if ((checkQ1 >= 10000) && (checkQ2 >= 10000) && (Q3 >= 10000) && (Q4 >= 10000))
      {
       if (Q3 < 0)
        {
           if (Q4 < 0)
            {
              String data4 = "MT" + (Convert.ToString(Q3 - SetQ3Q4)).Replace('-',
              'i')   +(Convert.ToString(Q4   -   SetQ3Q4)).Replace('-',   'i')   +
              Convert.ToString(checkQ1 + 8000) + Convert.ToString(checkQ2) + 08000;
              sport.Write(data4);
            }
           else
            {
              String data4 = "MT" + (Convert.ToString(Q3 - SetQ3Q4)).Replace('-',
              'i')    +    'p'    +    (Convert.ToString(Q4    +    SetQ3Q4))    +
              Convert.ToString(checkQ1 + 8000) + Convert.ToString(checkQ2) + 08000;
             sport.Write(data4);
            }
        }
       else
        {
           if (Q4 < 0)
            {
              String data4 = "MT" + 'p' + (Convert.ToString(Q3 + SetQ3Q4)) +
              (Convert.ToString(Q4    -    SetQ3Q4)).Replace('-',    'i')    +
              Convert.ToString(checkQ1 + 8000) + Convert.ToString(checkQ2) +
              08000;
              sport.Write(data4);
            }
           else
            {
              String data4 = "MT" + 'p' + (Convert.ToString(Q3 + SetQ3Q4)) + 'p' +
              (Convert.ToString(Q4 + SetQ3Q4)) + Convert.ToString(checkQ1 + 8000)
              + Convert.ToString(checkQ2) + 08000;
             sport.Write(data4);
            }
        }
      }
   else
      {
         double addZeroQ1;
         double addZeroQ2;
         double addZeroQ3;
         double addZeroQ4;


          if (Q3 < 0)
           {
             if (Q4 < 0)
                {
                  addZeroQ1 = Math.Ceiling(checkQ1);
                  addZeroQ2 = Math.Ceiling(checkQ2);
                  addZeroQ3 = Math.Ceiling(Q3) - SetQ3Q4;
                  addZeroQ4 = Math.Ceiling(Q4) - SetQ3Q4;
                }
             else
                {
                  addZeroQ1 = Math.Ceiling(checkQ1);
                  addZeroQ2 = Math.Ceiling(checkQ2);
```

```
                    addZeroQ3 = Math.Ceiling(Q3) - SetQ3Q4;
                    addZeroQ4 = Math.Ceiling(Q4) + SetQ3Q4;
                  }
            }
               else
                {
                  if (Q4 < 0)
                     {
                       addZeroQ1 = Math.Ceiling(checkQ1);
                       addZeroQ2 = Math.Ceiling(checkQ2);
                       addZeroQ3 = Math.Ceiling(Q3) + SetQ3Q4;
                       addZeroQ4 = Math.Ceiling(Q4) - SetQ3Q4;
                     }
                  else
                     {
                       addZeroQ1 = Math.Ceiling(checkQ1);
                       addZeroQ2 = Math.Ceiling(checkQ2);
                       addZeroQ3 = Math.Ceiling(Q3) + SetQ3Q4;
                       addZeroQ4 = Math.Ceiling(Q4) + SetQ3Q4;
                     }
                }

         if (addZeroQ1 < 10000)
            {
               sendQ1 = "0" + Convert.ToString(addZeroQ1);
            }
         else
            {
               sendQ1 = Convert.ToString(addZeroQ1);
            }
         if (addZeroQ2 < 10000)
            {
               sendQ2 = "0" + Convert.ToString(addZeroQ2);
            }
         else
            {
               sendQ2 = Convert.ToString(addZeroQ2);
            }
         if ((addZeroQ3 < 10000) || (addZeroQ3 > -10000))
            {
              if (addZeroQ3 < 0)
                {
                  sendQ3 = "i0" + Convert.ToString(addZeroQ3).TrimStart('-');
                }
         else
            {
               {
                  sendQ3 = "p0" + Convert.ToString(addZeroQ3);
               }
            }
         }
         if ((addZeroQ4 < 10000) || (addZeroQ4 > -10000))
            {
                 if (addZeroQ4 < 0)
                  {
                    sendQ4 = "i0" + Convert.ToString(addZeroQ4).TrimStart('-');
                  }
         else
            {
               {
                  sendQ4 = "p0" + Convert.ToString(addZeroQ4);
               }
            }
            }
      string sendData = "MT" + sendQ3 + sendQ4 + Convert.ToString(((Convert.ToInt16(sendQ1)
                        + 8000))) + sendQ2 + "08000";
      sport.Write(sendData);
      sendQ1 = null;
      sendQ2 = null;
      sendQ3 = null;
      sendQ4 = null;
   }
}
   else
```

```
            {
              screenRTB.SelectionColor = Color.Red;
              screenRTB.AppendText("Out of range !!!");
            }
          break;
        }
      }
}

private void clcBtn_Click(object sender, EventArgs e)
{
   string input = inputTxt.Text;
   string[] inputSP = input.Split(new string[] { "\\" }, StringSplitOptions.None);
   string axis = inputSP[0];
   double x = Convert.ToDouble(inputSP[1].Replace("X", null));
   double y = Convert.ToDouble(inputSP[2].Replace("Y", null));
   double z = Convert.ToDouble(inputSP[3].Replace("Z", null));
   int delayQ1 = 740;
   int delayQ2 = 400;
   int setQ3Q4 = 1190;
    {
       double Q1, Q2, Q3, Q4, QO11, QO12, QO13, QO14, QO21, QO22, QO23, QO24, XTO1, YTO1,
       ZTO1, XTO2, YTO2, ZTO2 ,TQ1 , TQ2;
       float toolAdjX = 11.5F;
   switch (axis)
      {
       case "V":
       if (((x >= 36) && (x <= 54.8)) && ((y >= 0) && (y <= 11.5)) && ((z >= 68) && (z <=
84)))
           {
             QO13 = 0 * (Math.PI / 180);
             QO14 = Math.Asin((y - (3 * Math.Sin(QO13))) / (toolAdjX * Math.Cos(QO13)));
             QO11 = z - (toolAdjX * Math.Sin(QO13) * Math.Sin(QO14)) + (3 * Math.Cos(QO13))
                    - 71;
             QO12 = x - (toolAdjX * Math.Cos(QO14)) - 24.5;
             XTO1 = QO12 + (toolAdjX * Math.Cos(QO14)) + 24.5;
             YTO1 = (toolAdjX * Math.Cos(QO13) * Math.Sin(QO14)) + (3 * Math.Sin(QO13));
             ZTO1 = (toolAdjX * Math.Sin(QO13) * Math.Sin(QO14)) - (3 * Math.Cos(QO13)) +
                    QO11 + 71;
             double errX = (Math.Abs(x - XTO1)) * (100 / x);
             double errY;

             if (y == 0)
              {
                 errY = 0;

              }
             else
              {
                 errY = (Math.Abs(y - YTO1)) * (100 / y);
              }
             double errZ = (Math.Abs(z - ZTO1)) * (100 / z);
             double err = (errX + errY + errZ) / 3;
             Q1 = QO11 * 1000;
             Q2 = QO12 * 1000;
             Q3 = QO13 * (180 / Math.PI);
             Q4 = QO14 * (180 / Math.PI);
             TQ1 = Math.Ceiling(Q1) - 1000;
             TQ2 = Math.Ceiling(Q2);
             screenRTB.SelectionColor = Color.PaleGreen;
             screenRTB.AppendText("Reset" + "\n");
             screenRTB.AppendText("=============================================================
                                  ==============" + "\n");

             if (((TQ1 + delayQ1) > 10000) && ((TQ2 + delayQ2) > 10000))
              {
                 String data2 = "RV" + Convert.ToString(TQ1 + delayQ1) +
                                Convert.ToString(TQ2 + delayQ2) + "0000" +
                                Convert.ToString(Q3) + "0000" + Convert.ToString(Q4) ;
                 sport.Write(data2);
              }
             else if (((TQ1 + delayQ1) > 10000) && ((TQ2 + delayQ2) < 10000))
              {
```

```
                String data3 = "RV" + Convert.ToString(TQ1 + delayQ1) + "0" +
                               Convert.ToString(TQ2+delayQ2) + "0000" +
                               Convert.ToString(Q3) + "0000" + Convert.ToString(Q4);
                sport.Write(data3);
              }
            else if (((TQ1 + delayQ1) < 10000) && ((TQ2 + delayQ2) > 10000))
              {
                String data3 = "RV0" + Convert.ToString(TQ1 + delayQ1) +
                               Convert.ToString(TQ2 + delayQ2) + "0000" +
                               Convert.ToString(Q3) + "0000" + Convert.ToString(Q4);
                sport.Write(data3);
              }
            else
              {
                String data3 = "RV0" + Convert.ToString(TQ1 + delayQ1) + "0" +
                               Convert.ToString(TQ2 + delayQ2) + "0000" +
                               Convert.ToString(Q3) + "0000" + Convert.ToString(Q4);
                sport.Write(data3);
              }
          }
        else
          {
            screenRTB.SelectionColor = Color.Red;
            screenRTB.AppendText("Out of range !!!");
          }
        break;
        }
    }
}
private void inputTxt_TextChanged(object sender, EventArgs e)
{

}
private void screenRTB_TextChanged(object sender, EventArgs e)
{

}
#region Serial config

public System.IO.Ports.SerialPort sport;
public void serialport_connect(String port, int baudrate, Parity parity, int databits,
            StopBits stopbits)
{
   DateTime dt = DateTime.Now;
   String dtn = dt.ToShortTimeString();

   sport = new System.IO.Ports.SerialPort(
   port, baudrate, parity, databits, stopbits);
   try
   {
     sport.Open();
     screenRTB.SelectionColor = Color.DarkSalmon;
     screenRTB.AppendText("[" + dtn + "] " + "Connected\n");
   }
   catch (Exception ex) { MessageBox.Show(ex.ToString(), "Error"); }
}
private void connectBtn_Click_1(object sender, EventArgs e)
{
   String port = nCom.Text;
   int baudrate = Convert.ToInt32(9600);
   Parity parity = (Parity)Enum.Parse(typeof(Parity), "None");
   int databits = Convert.ToInt32(8);
   StopBits stopbits = StopBits.One;
   serialport_connect(port, baudrate, parity, databits, stopbits);
}

private void disconnectBtn_Click_1(object sender, EventArgs e)
{
   DateTime dt = DateTime.Now;
   String dtn = dt.ToShortTimeString();

    if (sport.IsOpen)
     {
       sport.Close();
```

```
            screenRTB.SelectionColor = Color.DarkSalmon;
            screenRTB.AppendText("[" + dtn + "] " + "Disconnected\n");
        }
    }
    #endregion

    private void inverseKiFrm_Load(object sender, EventArgs e)
    {

    }
    private void button3_Click(object sender, EventArgs e)
    {
        screenRTB.Clear();
    }
    private void nCom_TextChanged(object sender, EventArgs e)
    {

    }
    private void button4_Click(object sender, EventArgs e)
    {
        string input = inputTxt.Text;
        string[] inputSP = input.Split(new string[] { "\\" }, StringSplitOptions.None);
        string axis = inputSP[0];
        double x = Convert.ToDouble(inputSP[1].Replace("X", null));
        double y = Convert.ToDouble(inputSP[2].Replace("Y", null));
        double z = Convert.ToDouble(inputSP[3].Replace("Z", null));
        int DelayQ1 = 740;
        int DelayQ2 = 1500;
        {
            double Q1, Q2, Q3, Q4, QO11, QO12, QO13, QO14, QO21, QO22, QO23, QO24, XTO1, YTO1,
                   ZTO1, XTO2, YTO2, ZTO2, TQ1, TQ2;
            float toolAdjX = 11.5F;
            screenRTB.SelectionColor = Color.White;
            screenRTB.AppendText("\n  X  =  "  +  Convert.ToString(x)  +  "\t"  +  "Y  =  "  +
                                 Convert.ToString(y) + "\t" + "Z = " + Convert.ToString(z) + "\n");
            switch (axis)
            {
                case "H":
                float HoffsetZ = 72.5f;
                float HoffsetX = 24.2f;
                if (((x >= 23) && (x <= 42)) && ((y >= 0) && (y <= 11.5)) && ((z >= 61) && (z
                    <= 69)))
                {
                    QO13 = (-90) * (Math.PI / 180);
                    QO14 = (90) * (Math.PI / 180);
                    QO11  =  z  -  (toolAdjX  *  Math.Sin(QO13)  *  Math.Sin(QO14))  +  (3  *
                             Math.Cos(QO13)) - HoffsetZ;
                    QO12 = x - (toolAdjX * Math.Cos(QO14)) - HoffsetX;

                    QO23 = (90) * (Math.PI / 180);
                    QO24 = (-90) * (Math.PI / 180);
                    QO21  =  z  -  (toolAdjX  *  Math.Sin(QO23)  *  Math.Sin(QO24))  +  (3  *
                             Math.Cos(QO23)) - HoffsetZ;
                    QO22 = x - (toolAdjX * Math.Cos(QO24)) - HoffsetX;

                    XTO1 = QO12 + (toolAdjX * Math.Cos(QO14)) + HoffsetX;
                    YTO1 = (toolAdjX * Math.Cos(QO13) * Math.Sin(QO14)) + (3 * Math.Sin(QO13));
                    ZTO1 = (toolAdjX * Math.Sin(QO13) * Math.Sin(QO14)) - (3 * Math.Cos(QO13))
                           + QO11 + HoffsetZ;
                    double errO1X = (Math.Abs(x - XTO1)) * (100 / x);
                    double errO1Y;

                     if (y == 0)
                      {
                          errO1Y = 0;
                      }
                else
                    {
                        errO1Y = (Math.Abs(y - YTO1)) * (100 / y);
                    }
                double errO1Z = (Math.Abs(z - ZTO1)) * (100 / z);
                double errO1 = (errO1X + errO1Y + errO1Z) / 3;
                XTO2 = QO22 + (toolAdjX * Math.Cos(QO24)) + HoffsetX;
                YTO2 = (toolAdjX * Math.Cos(QO23) * Math.Sin(QO24)) + (3 * Math.Sin(QO23));
```

```
                ZTO2 = (toolAdjX * Math.Sin(QO23) * Math.Sin(QO24)) - (3 * Math.Cos(QO23)) +
                       QO21 + HoffsetZ;
                double errO2X = (Math.Abs(x - XTO2)) * (100 / x);
                double errO2Y;

                    if (y == 0)
                       {
                          errO2Y = 0;
                       }
                    else
                       {
                          errO2Y = (Math.Abs(y - YTO2)) * (100 / y);
                }
                double errO2Z = (Math.Abs(z - ZTO2)) * (100 / z);
                double errO2 = (errO2X + errO2Y + errO2Z) / 3;
                if (errO1 > errO2)
                 {
                    Q1 = QO21 * 1000;
                    Q2 = QO22 * 1000;
                    Q3 = (QO23 * (180 / Math.PI) * (100 / 3));
                    Q4 = (QO24 * (180 / Math.PI) * (100 / 3));
                 }
                else if (errO1 == errO2)
                 {
                    Q1 = QO11 * 1000;
                    Q2 = QO12 * 1000;
                    Q3 = QO13 * (180 / Math.PI) * (100 / 3);
                    Q4 = QO14 * (180 / Math.PI) * (100 / 3);
                 }
                else
                 {
                    Q1 = QO11 * 1000;
                    Q2 = QO12 * 1000F;
                    Q3 = QO13;
                    Q4 = QO14;
                 }
            screenRTB.SelectionColor = Color.PaleGreen;
            screenRTB.AppendText("TQ1 : " + Q1 + "ms" + "\t" + "TQ2 : " + Q2 + "ms" + "\t" + "TQ3 :
                    " + Q3 + "ms" + "\t" + "TQ4 : " + Q4 + "ms" + "\n");
            screenRTB.SelectionColor = Color.OrangeRed;
            screenRTB.AppendText("<Option 1> Error Average : " + "\t" + Convert.ToString(errO1) +
                            "\n");
            screenRTB.SelectionColor = Color.OrangeRed;
            screenRTB.AppendText("<Option 2> Error Average : " + "\t" + Convert.ToString(errO2) +
                            "\n");
            screenRTB.SelectionColor = Color.PaleVioletRed;
            screenRTB.AppendText("=============================================================
                            ======" + "\n");
            double checkQ1 = Q1 + DelayQ1;
            double checkQ2 = Q2 + DelayQ2;
            int SetQ3Q4 = 1190;
                 if ((checkQ1 >= 10000) && (checkQ2 >= 10000) && (Q3 >= 10000) && (Q4 >= 10000))
                    {
                       if (Q3 < 0)
                       {
                         if (Q4 < 0)
                           {
String   data4   =   "RH"   +   (Convert.ToString(Q3   -   SetQ3Q4)).Replace('-',   'i')   +
                       (Convert.ToString(Q4 - SetQ3Q4)).Replace('-', 'i') + Convert.ToString(checkQ1
                       + 8000) + Convert.ToString(checkQ2) + 08000;
sport.Write(data4);
                         }
                        else
                         {
String  data4  =  "RH"  +  (Convert.ToString(Q3  -  SetQ3Q4)).Replace('-',  'i')  +  'p'  +
                         (Convert.ToString(Q4 + SetQ3Q4)) + Convert.ToString(checkQ1 + 8000) +
                         Convert.ToString(checkQ2) + 08000;
sport.Write(data4);
                         }
                       }
                      else
                         {
                           if (Q4 < 0)
                             {
```

```
String data4 = "RH" + 'p' + (Convert.ToString(Q3 + SetQ3Q4)) + (Convert.ToString(Q4 -
                SetQ3Q4)).Replace('-', 'i') + Convert.ToString(checkQ1 + 8000) +
                Convert.ToString(checkQ2) + 08000;
sport.Write(data4);
                    }
                else
                  {
String data4 = "RH" + 'p' + (Convert.ToString(Q3 + SetQ3Q4)) + 'p' + (Convert.ToString(Q4 +
                SetQ3Q4)) + Convert.ToString(checkQ1 + 8000) + Convert.ToString(checkQ2) +
                08000;
sport.Write(data4);
                  }
                }
            }
          else
             {
                double addZeroQ1;
                double addZeroQ2;
                double addZeroQ3;
                double addZeroQ4;
                if (Q3 < 0)
                   {
                      if (Q4 < 0)
                        {
                          addZeroQ1 = Math.Ceiling(checkQ1);
                          addZeroQ2 = Math.Ceiling(checkQ2);
                          addZeroQ3 = Math.Ceiling(Q3) - SetQ3Q4;
                          addZeroQ4 = Math.Ceiling(Q4) - SetQ3Q4;
                        }
                      else
                        {
                          addZeroQ1 = Math.Ceiling(checkQ1);
                          addZeroQ2 = Math.Ceiling(checkQ2);
                          addZeroQ3 = Math.Ceiling(Q3) - SetQ3Q4;
                          addZeroQ4 = Math.Ceiling(Q4) + SetQ3Q4;
                        }
                   }
          else
             {
                if (Q4 < 0)
                   {
                      addZeroQ1 = Math.Ceiling(checkQ1);
                      addZeroQ2 = Math.Ceiling(checkQ2);
                      addZeroQ3 = Math.Ceiling(Q3) + SetQ3Q4;
                      addZeroQ4 = Math.Ceiling(Q4) - SetQ3Q4;
                   }
                else
                   {
                      addZeroQ1 = Math.Ceiling(checkQ1);
                      addZeroQ2 = Math.Ceiling(checkQ2);
                      addZeroQ3 = Math.Ceiling(Q3) + SetQ3Q4;
                      addZeroQ4 = Math.Ceiling(Q4) + SetQ3Q4;
                      }
             }
        if (addZeroQ1 < 10000)
          {
            sendQ1 = "0" + Convert.ToString(addZeroQ1);
          }
        else
          {
            sendQ1 = Convert.ToString(addZeroQ1);
          }
        if (addZeroQ2 < 10000)
          {
            sendQ2 = "0" + Convert.ToString(addZeroQ2);
          }
        else
          {
            sendQ2 = Convert.ToString(addZeroQ2);
          }
        if ((addZeroQ3 < 10000) || (addZeroQ3 > -10000))
          {
            if (addZeroQ3 < 0)
          {
```

```
                sendQ3 = "i0" + Convert.ToString(addZeroQ3).TrimStart('-');
                }
                else
                 {
                   {
                     sendQ3 = "p0" + Convert.ToString(addZeroQ3);
                   }
                 }
              }
            if ((addZeroQ4 < 10000) || (addZeroQ4 > -10000))
              {
                if (addZeroQ4 < 0)
                 {
                   sendQ4 = "i0" + Convert.ToString(addZeroQ4).TrimStart('-');
                 }
                else
                 {
                    {
                       sendQ4 = "p0" + Convert.ToString(addZeroQ4);
                    }
                 }
              }
              string      sendData      =      "RH"      +      sendQ3      +      sendQ4      +
                                Convert.ToString(((Convert.ToInt16(sendQ1) + 8000))) + sendQ2 +
                                "08000";
              sport.Write(sendData);
              sendQ1 = null;
              sendQ2 = null;
              sendQ3 = null;
              sendQ4 = null;
            }
}
                  else
                  {
                    screenRTB.SelectionColor = Color.Red;
                    screenRTB.AppendText("Out of range !!!");
                  }
                  break;
                }
              }
            }
        }
}
```

ภาคผนวก ค

โค้ดโปรแกรมประมวลผลภาพและควบคุมการเคลื่อนที่

โปรแกรมประมวลผลภาพและติดตามบุคคล ใช้สำหรับการควบคุมให้หุ่นยนต์ติดตามบุคคล 1 ท่าน

testSetting.C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Kinect;
using System.IO;
using System.IO.Ports;
using System.Threading;
using System.Speech.Synthesis;
using System.Configuration.Internal;

namespace SkeletonExample
    {
      class Program
        {

          static void Main(string[] args)
              {
                SetFile settingData = new SetFile();
                settingData.isKeyDown = false;
                settingData.Save();
                KinectSensor sensor = KinectSensor.KinectSensors.Where(s => s.Status ==
                                      KinectStatus.Connected).FirstOrDefault();
                if (sensor == null)
                   {
                    Console.WriteLine("No Kinect sensor found!");
                    return;
                   }
                Tracker tracker = new Tracker(sensor);
                sensor.Start();
                while (Char.ToLowerInvariant(Console.ReadKey().KeyChar) != 'q') { }
                sensor.Stop();
              }
        }

      public class Tracker
        {
          SpeechSynthesizer _speechSynthe = new SpeechSynthesizer();
          SetFile datasetting = new SetFile();
          public bool serialIsOpen=false;
          public double trigAngle;
          public bool isStop;
          public void calTrigAngle()
               {
                 double percentageTrigAngle = 60;
                 trigAngle = (percentageTrigAngle * 24.228)/100;
                 double percentageTrigDis = 100;
                 trigDis = (percentageTrigDis * 150)/100;
               }
          public System.IO.Ports.SerialPort sport;
          public void connect()
               {
                 String port = "COM6";
                 int baudrate = Convert.ToInt32(9600);
                 Parity parity = (Parity)Enum.Parse(typeof(Parity), "None");
                 int databits = Convert.ToInt32(8);
                 StopBits stopbits = StopBits.One;
```

```
            if (serialIsOpen==false)
               {
                serialport_connect(port, baudrate, parity, databits, stopbits);
               }
            calTrigAngle();
          }
    public  void  serialport_connect(string  port,  int  baudrate,  Parity  parity,  int
                databits, StopBits stopbits)
          {
             serialIsOpen = true;
             DateTime dt = DateTime.Now;
             String dtn = dt.ToShortTimeString();

             sport  =  new  System.IO.Ports.SerialPort(port,  baudrate,  parity,  databits,
                       stopbits);

             try
               {
                  sport.Open();
               }
    catch (Exception ex) { Console.WriteLine(ex.ToString() + "Error"); }
    sport.Write("ATCMF");
    Thread.Sleep(2000);
    sport.Write("ATCMO");
  }
public void invertControl(float xValue,float yValue,float zValue)
  {

    if ((zValue > 150) && (xValue < 0))
          {
             int setR = 707;
             double roundR = 29.681;
             int stopF = 150;
             float distenceX = xValue;
             float Moom = (float)Math.Atan((-xValue) / zValue);
             float Moom2 = (float)(roundR * (Moom * (180 / Math.PI)) + setR );
             int MoomX = (int)Math.Ceiling(Moom2);
             Console.WriteLine(Moom);
             Console.WriteLine(Moom2);
             Console.WriteLine(MoomX);
             sport.WriteLine("MVU00R30");
             isStop = false;
             Thread.Sleep(MoomX);
             sport.WriteLine("MV0000.");
             isStop = true;
             Thread.Sleep(2000);
             float NewZ = (float)(Math.Sqrt(Math.Pow(zValue, 2) - Math.Pow(yValue, 2)));
             float  distence  =  (float)((Math.Sqrt(Math.Pow(xValue,  2)  +  Math.Pow(NewZ,
                                2))));
             float distence1 = ((float)(50.3 * (distence-stopF)));
             int distence2 = (Int32)Math.Ceiling(distence1);
             if (distence > 0)
               {
                 sport.WriteLine("MVU50.");
                 isStop = false ;
                 Thread.Sleep(distence2);
                 sport.WriteLine("MV0000.");
               }
          }
    else if ((zValue > 150) && (xValue > 0))
          {
             int setL = 0;
             string sentTurnLeft = "";
             setL = 711;
             if(isStop==true)
               {
                 sentTurnLeft = "MVU00L50";
               }
             else
               {
                 setL = 521;
                 sentTurnLeft = "MVU00L30";
               }
           double  roundL = 30.964;
```

```
            int stopF = 150;
            float distenceX = xValue;
            float Moom = (float)Math.Atan(xValue / zValue);
            float Moom2 = (float)(roundL * (Moom * (180 / Math.PI)) + setL);
            int MoomX = (int)Math.Ceiling(Moom2);
            Console.WriteLine(Moom);
            Console.WriteLine(Moom2);
            Console.WriteLine(MoomX);
            sport.WriteLine(sentTurnLeft);
            isStop = false;
            Thread.Sleep(MoomX);
            sport.WriteLine("MV0000.");
            isStop = true;
            Thread.Sleep(2000);
            float NewZ = (float)(Math.Sqrt(Math.Pow(zValue, 2) - Math.Pow(yValue, 2)));
            float distence = (float)((Math.Sqrt(Math.Pow(xValue, 2) + Math.Pow(NewZ,
                              2))));
            float distence1 = (((float)50.3 * (distence-stopF)));
            int distence2 = (Int32)Math.Ceiling(distence1);
            if (distence > 0)
              {
                sport.WriteLine("MVU50.");
                isStop = false;
                Thread.Sleep(distence2);
                sport.WriteLine("MV0000.");
              }
        }
            else if ((zValue < 150) && (xValue < 0) )
              {
                int setR = 707;
                double roundR = 29.681;
                float distenceX = xValue;
                float Moom = (float)Math.Atan((-xValue) / zValue);
                float Moom2 = (float)(roundR * (Moom * (180 / Math.PI)) + setR);
                int MoomX = (int)Math.Ceiling(Moom2);
                float NewZ = (float)(Math.Sqrt(Math.Pow(zValue, 2) - Math.Pow(yValue,
                              2)));
                float distence = (float)((Math.Sqrt(Math.Pow(xValue, 2) + Math.Pow(NewZ,
                              2))));
                Console.WriteLine(Moom);
                Console.WriteLine(Moom2);
                Console.WriteLine(MoomX);
                sport.WriteLine("MVD50.");
                isStop = false;
                Thread.Sleep(1081);
                sport.WriteLine("MV0000.");
                isStop = true;
                Thread.Sleep(2000);
                sport.WriteLine("MVU00R50");
                Thread.Sleep(MoomX);
                sport.WriteLine("MV0000.");

              }
            else if ((zValue < 150) && (xValue > 0))
              {
                int setL = 711;
                double roundL = 30.964;
                float distenceX = xValue;
                float Moom = (float)Math.Atan(xValue / zValue);
                float Moom2 = (float)(roundL * (Moom * (180 / Math.PI)) + setL);
                int MoomX = (int)Math.Ceiling(Moom2);
                float NewZ = (float)(Math.Sqrt(Math.Pow(zValue, 2) - Math.Pow(yValue,
                              2)));
                float distence = (float)((Math.Sqrt(Math.Pow(xValue, 2) + Math.Pow(NewZ,
                              2))));
                Console.WriteLine(Moom);
                Console.WriteLine(Moom2);
                Console.WriteLine(MoomX);
                sport.WriteLine("MVD50.");
                Thread.Sleep(1081);
                sport.WriteLine("MV0000.");
                Thread.Sleep(2000);
                sport.WriteLine("MVU00L50");
                Thread.Sleep(MoomX);
```

```
                    sport.WriteLine("MV0000.");


                }

        Console.WriteLine("User:  " + "X=" + Convert.ToString(xValue) + ",  " + "Y=" +
                          Convert.ToString(yValue)     +     ",     "     +     "Z="     +
                          Convert.ToString(zValue));
    }


private Skeleton[] skeletons = null;


public Tracker(KinectSensor sensor)
 {
  sensor.SkeletonFrameReady += SensorSkeletonFrameReady;
  sensor.SkeletonStream.Enable();
 }

private void SensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
 {
  connect();
  using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
   {
    if (skeletonFrame != null)
     {
      if (this.skeletons == null)
       {
        this.skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];
       }

      skeletonFrame.CopySkeletonDataTo(this.skeletons);

      Skeleton skeleton = this.skeletons.Where(s => s.TrackingState ==
                          SkeletonTrackingState.Tracked).FirstOrDefault();

      if (skeleton != null)
       {
        Joint j = skeleton.Joints[JointType.ShoulderCenter];

        if (j.TrackingState == JointTrackingState.Tracked)
         {

          float x=j.Position.X*100;
          float y = j.Position.Y * 100;
          float z = j.Position.Z * 100;
          Console.WriteLine("Kinect is running");

          if (datasetting.isKeyDown == true)
           {
            double checkAngle = (180 / Math.PI) *
                                Math.Atan(Convert.ToDouble(Math.Abs(x)) /
                                Convert.ToDouble(z));
            double checkDis = Math.Abs(z);
            Console.WriteLine(checkAngle + "::" + trigAngle);
            Console.WriteLine(checkDis + "::" + trigDis);

            if((z < 150) && (isStop == false))
             {
              sport.WriteLine("MVU0000.");
              isStop = true;
              Console.WriteLine("CASE0");
             }
            else if ((checkAngle >= trigAngle) && (z >= 150))
             {
              invertControl(x, y, z);
              Console.WriteLine("CASE1");

             }
            else if ((checkAngle < trigAngle) && (z >= 150))
             {
              sport.WriteLine("MVU50.");
              Thread.Sleep(2000);
```

```
                Console.WriteLine("CASE2");
                isStop = false;
              }
            else if ((checkAngle >= trigAngle) && (z <= 150) && (isStop == true))
              {
                invertControl(x, y, z);
                Console.WriteLine("CASE3");
              }
            else if ((checkAngle >= trigAngle) && (z <= 150) && (isStop == true))
              {
                sport.WriteLine("MVD50.");
                Thread.Sleep(1081);
                sport.WriteLine("MVU0000.");
                Console.WriteLine("CASE4");
                isStop = true;
              }
        }
      else
        {
          ConsoleKeyInfo keyPress = Console.ReadKey();

          if (keyPress.KeyChar == 'k')
            {
              invertControl(x, y, z);
              datasetting.isKeyDown = true;
              datasetting.Save();
            }
        }
    }
  }
}
```