# การออกแบบและพัฒนาโปรแกรมจำลองการทำงาน หุ่นยนต์อุตสาหกรรม

# DESIGN AND DEVELOP PROGRAM ROBOT SIMULATION

นายอาทิตย์ บุระซัน

MR. ARTHIT BURASUN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

**คณะวิศวกรรมศาสตร์**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ใบรับรองปริญญานิพนธ์**

**หัวข้อปริญญานิพนธ์** การออกแบบและพัฒนาโปรแกรมจำลองการทำงานหุ่นยนต์อุตสาหกรรม
DESIGH AND DEVELOPMENT PROGRAMMING ROBOT SIMULATION

**ชื่อนักศึกษา** นายอาทิตย์ บุระซัน รหัสประจำตัว 40010997

**หลักสูตร** วิศวกรรมศาสตรบัณฑิต

**สาขาวิชา** วิศวกรรมอุตสาหการ

**อาจารย์ผู้ควบคุมปริญญานิพนธ์**

(อาจารย์พลชัย โชติปรายนกุล)

(อาจารย์อุดม จันทร์จรัสสุข)

**หัวข้อปริญญานิพนธ์** การออกแบบและพัฒนาโปรแกรมจำลองการทำงานหุ่นยนต์อุตสาหกรรม
**นักศึกษา** นายอาทิตย์ บุระซัน รหัสประจำตัว 40010997
**หลักสูตร** วิศวกรรมศาสตรบัณฑิต
**สาขาวิชา** วิศวกรรมอุตสาหการ
**ปีการศึกษา** 2543
**อาจารย์ผู้ควบคุมปริญญานิพนธ์**
อาจารย์พลชัย โชติปรายนกุล
อาจารย์อุดม จันทร์จรัสสุข

## บทคัดย่อ

ในปัจจุบัน โรงงานอุตสาหกรรมได้ถูกออกแบบให้สามารถทำการผลิตผลิตภัณฑ์ในระดับอุตสาหกรรมขนาดใหญ่ (Mass Production) หรือสามารถทำการผลิตผลิตภัณฑ์ได้หลากหลาย บนสายการผลิตเดียวที่เรียกว่าระบบการผลิตแบบยืดหยุ่น (Flexible Manufacturing System; FMS) สำหรับการนำหุ่นยนต์อุตสาหกรรมมาช่วยในการผลิต สิ่งที่ต้องพิจารณาและตัดสินใจเลือกประเภทของหุ่นยนต์ที่นำเข้ามาใช้ในโรงงาน ผู้ใช้งานจะต้องใช้เวลาในการศึกษา เนื่องด้วยหุ่นยนต์แต่ละแบบจะมีความแตกต่างกันทั้งในด้านโครงสร้าง รูปแบบการทำงาน และพื้นที่ทำงานของหุ่นยนต์ โปรแกรมออกแบบและจำลองการทำงานของหุ่นยนต์อุตสาหกรรมนี้ ในโครงงานสามารถเป็นเครื่องมือตัวหนึ่งที่ช่วยให้ผู้ใช้งานสามารถตัดสินใจได้อย่างรวดเร็วและลดต้นทุน เวลาในการศึกษา โดยโครงงานได้สร้างโปรแกรมจำลองการทำงานหุ่นยนต์อุตสาหกรรมจากการเขียนด้วยโปรแกรมภาษาเดลไฟ โดยโปรแกรมจำลองการทำงานสามารถติดตั้งได้บนเครื่องคอมพิวเตอร์ส่วนบุคคล รูปแบบของโครงงานประกอบด้วยรูปแบบของหุ่นยนต์ 4 แบบ ที่ใช้กันทั่วไปในอุตสาหกรรม คือ หุ่นยนต์แบบ Cartesian, หุ่นยนต์แบบ Cylindrical, หุ่นยนต์แบบ Spherical, และหุ่นยนต์แบบ Articulated โดยนำพื้นฐานและทฤษฎี Robotics คือ Forward kinematics, Inverse kinematics, และ Trajectory Interpolators มาใช้ในการเขียนโปรแกรม ซึ่งจะทำการทดสอบโปรแกรมโดยจำลองการทำงานของหุ่นยนต์ทั้ง 4 แบบ โดยที่จะให้ส่วนปลายของแขนกล (End Effector) เข้าหาชิ้นงานได้อย่างถูกต้อง และราบเรียบที่สุด

| | | | |
|---|---|---|---|
| **Thesis Title** | DESIGN AND DEVELOPMENT PROGRAMMING ROBOT SIMULATION | | |
| **Student** | Mr. Arthit | Burasun | ID.Student 40010997 |
| **Degree** | Bachelor of Engineering | | |
| **Program** | Industrial Engineering | | |
| **Year** | 2000 | | |
| **Thesis Advisor** | Mr. Pholchai | Chotiprayanakul | |
| | Mr. Udom | Janjarussuk | |


## ABSTRACT

In present day. There are many Industrial Factories designed for flexible manufacturing system in mass product scale. Industrial Robots were considerated in factory. It use long time to consider. Robot simulation program useful a tool to help them for learning about type, structure, and workspace of robot. So this thesis was simulation program written by visual program with Delphi language. It's can install in personal computers. Program consist of 4 type of robot to use in industrial: Cartesian Robot, Cylindrical Robot, Spherical Robot, and Articulated Robot. Program used theory of robotics (Forward Kinematics, Inverse Kinematics, and Trajectory Interpolators) in structure. It can simulated 4 type of robot to move to end point (Target) by End Effector. We can see it move correctly and smooth.

# กิตติกรรมประกาศ

คำขอบคุณจากใจจริงสำหรับท่านเหล่านี้

| | |
|---|---|
| อ.พลชัย โชติปรายนกุล | - ในความกรุณาของพี่ที่อุตสาห์คอยช่วยเหลือ สั่งสอน และคอยจ้ำจี้จ้ำไชเวลาที่ข้าพเจ้าขี้เกียจทำงาน |
| ดร.สรรพสิทธิ์ ลิ่มนรรัตน์ | - ในความหวังดีและคอยห่วงใยดูแลเอาใจใส่ในตัวข้าพเจ้าเสมอๆ ขอบคุณมากๆครับพี่ |
| อ.อุดม จันทร์จรัสสุข | - ที่คอยแนะนำบางส่วนของโปรแกรมที่ข้าพเจ้าไม่รู้ และถ้าไม่มีพี่ตอนนี้ข้าพเจ้าก็คงจะยังไม่รู้อะไรเกี่ยวกับคอมพิวเตอร์แน่ๆ |
| อาจารย์ทุกท่านในภาค | - ที่เป็นทั้งอาจารย์และพี่ชายของพวกเรา |
| พ่อ - แม่ | - ที่ทำให้ผมมีวันนี้ พระคุณของพ่อกับแม่ บุญคุณของท่านชาตินี้ข้าพเจ้าคงไม่มีวันใช้หมด |
| เพื่อนๆทุกคน | - ที่เป็นกำลังและคอยห่วงใยข้าพเจ้าเสมอมา |

# สารบัญ

# สารบัญตาราง

# สารบัญภาพ

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันนี้ในโรงงานอุตสาหกรรมได้ถูกออกแบบให้สามารถทำการผลิตในระดับของอุตสาหกรรมขนาดใหญ่ หรือเป็นการผลิตที่สามารถผลิตผลิตภัณฑ์ได้หลายๆรูปแบบ(Flexible Manufacturing System) ในปริมาณของวัตถุที่ทำการผลิตแต่ละครั้ง การเปลี่ยนแปลงของวิธีการผลิตนี้จะเปลี่ยนแปลงได้ยากและต้องใช้ต้นทุนที่สูง ในอนาคตคอมพิวเตอร์ที่ใช้อยู่ในโรงงานอุตสาหกรรมจะถูกดัดแปลงให้มีความพร้อมรับความผันแปรของความต้องการของตลาด และความเปลี่ยนแปลงของการออกแบบ, วัตถุดิบ และเครื่องมือ กฎเกณฑ์ใหม่ๆจะถูกพัฒนาขึ้นสำหรับหุ่นยนต์และคอมพิวเตอร์ และตัวบุคคลผู้ใช้งาน

ในการตัดสินใจเลือกใช้หุ่นยนต์เพื่อนำมาใช้งานในโรงงานอุตสาหกรรม หรือเพื่อนำมาทำการศึกษาออกแบบจำลองการทำงานของหุ่นยนต์ในด้านต่างๆ ผู้ลงทุนหรือเจ้าของกิจการจะต้องพิจารณาถึงหุ่นยนต์ที่จะนำมาใช้ ซึ่งในการพิจารณาจะต้องใช้เวลาในการศึกษามาก เพราะต้องศึกษาทั้งโครงสร้างของหุ่นยนต์ รูปแบบการทำงาน พื้นที่ที่ใช้ในการทำงานของหุ่นยนต์ และราคาของหุ่นยนต์ที่จะซื้อเข้ามา งานวิจัยนี้จึงเป็นเครื่องมือช่วยในการตัดสินใจเลือกแก่ผู้ลงทุนหรือเจ้าของกิจการ ทำให้ประหยัดทั้งเวลาและต้นทุน ซึ่งงานวิจัยนี้ได้ทำการจำลองการทำงานของหุ่นยนต์อุตสาหกรรม โดยงานวิจัยนี้ได้สร้างด้วยการเขียนโปรแกรมแบบวิชวลด้วยภาษาเดลไฟ และสามารถนำมาติดตั้งเพื่อใช้งานในเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไปได้ รูปแบบของงานวิจัยนี้จะประกอบด้วยรูปแบบของหุ่นยนต์ 4 แบบที่ใช้กันอยู่ในโรงงานอุตสาหกรรม คือหุ่นยนต์แบบคาร์ทีเซียน, หุ่นยนต์แบบไซลินดริคอล, หุ่นยนต์แบบสเฟอริคอล, และหุ่นยนต์แบบอาร์ติคิวเลท ซึ่งจะนำมาทดสอบโปรแกรมโดยการจำลองการทำงานของหุ่นยนต์ทั้ง 4 แบบ โดยที่จะให้ส่วนปลายของแขนกลของหุ่นยนต์สามารถเข้าหาชิ้นงานได้อย่างถูกต้อง ราบเรียบ และเป็นไปได้

## 1.2 วัตถุประสงค์ของการศึกษา

เพื่อสามารถนำงานวิจัยที่นำมาเป็นตัวช่วยตัดสินใจในการเลือกใช้หุ่นยนต์อุตสาหกรรม หรือใช้ในการออกแบบและจำลองการทำงานของหุ่นยนต์อุตสาหกรรมได้ โดยที่ระบบสนับสนุนดังกล่าวจะต้องมีดังนี้

1. สามารถช่วยให้ผู้ใช้งานซื้อหุ่นยนต์อุตสาหกรรม สามารถตัดสินใจเลือกใช้ได้อย่างถูกต้องและรวดเร็ว
2. เป็นเครื่องมือช่วยในการตัดสินใจที่มีลักษณะระบบคงที่ในการใช้งาน
3. เพื่อเป็นการลดต้นทุนในการพัฒนาและตรวจสอบการทำงานของหุ่นยนต์อุตสาหกรรม
4. เป็นพื้นฐานการออกแบบและพัฒนาหุ่นยนต์อุตสาหกรรมต่อไป

## 1.3 ขอบเขตของการศึกษา

ทำการออกแบบโปรแกรมเพื่อการพัฒนาและจำลองการทำงานของหุ่นยนต์อุตสาหกรรม โดยใช้โปรแกรมแบบวิชวลด้วยภาษาเดลไฟ ทำการออกแบบการทำงานของหุ่นยนต์อุตสาหกรรมทั้ง 4 แบบ ได้แก่ หุ่นยนต์แบบคาร์ทีเซียน, หุ่นยนต์แบบไซลินดริคอล, หุ่นยนต์แบบสเฟอริคอล และหุ่นยนต์แบบอาร์ติคิวเลท ซึ่งจะนำมาทดสอบโปรแกรมโดยการจำลองการทำงานของหุ่นยนต์ทั้ง 4 แบบ โดยที่จะให้ส่วนปลายของแขนกลของหุ่นยนต์สามารถเข้าหาชิ้นงานได้อย่างถูกต้อง ราบเรียบ และเป็นไปได้ ด้วยกระบวนการทาง Forward Kinematics

## 1.4 วิธีการวิจัย

1. กำหนดปัญหาและวัตถุประสงค์ของงานวิจัย ด้วยการดำเนินการเขียนกรอบการทำงาน และแนวทางแก้ไขปัญหา
2. การศึกษาและวิเคราะห์ถึงทฤษฎีของหุ่นยนต์
3. การศึกษาการเขียนและการใช้งานโปรแกรมเดลไฟ 5.0
4. การเขียนโปรแกรมของหุ่นยนต์อุตสาหกรรม
5. การทำการทดสอบโปรแกรมที่เขียนขึ้นมา
6. สรุปผลจากการวิจัย และข้อเสนอแนะ

## 1.5 เครื่องมือที่ใช้ในการวิจัย

1. เครื่องคอมพิวเตอร์ (PC)
2. โปรแกรมเดลไฟ 5.0
3. หนังสือ ROBOTICS FOR ENGINEERING ของ Yoram Koren

## 1.6 ประโยชน์ของผลการวิจัย

1. สามารถช่วยให้ผู้ลงทุนซื้อหุ่นยนต์อุตสาหกรรม สามารถตัดสินใจได้อย่างถูกต้องและรวดเร็ว
2. สามารถช่วยลดต้นทุนการการศึกษาและการจัดซื้อหุ่นยนต์อุตสาหกรรม
3. สามารถพัฒนา และตรวจสอบหุ่นยนต์อุตสาหกรรมที่มีขนาดใหญ่ ในห้องปฏิบัติการขนาดเล็กได้
4. สามารถจำลองการทำงานของหุ่นยนต์อุตสาหกรรมได้ โดยไม่ต้องลงทุนซื้อตัวหุ่นยนต์มาจริงๆ

# บทที่ 2
# ทฤษฎี

## 2.1 แนวคิดพื้นฐานของระบบหุ่นยนต์ (Basic concept of robotics) [3; หน้า 1]

หุ่นยนต์ในอุตสาหกรรมมีลักษณะที่ดูแตกต่างจากมนุษย์ แต่มันสามารถทำงานแทนมนุษย์ได้ ซึ่งใช้กันอย่างกว้างขวางในงานอุตสาหกรรม เช่น งานเชื่อม งานพ่นสี งานตรวจสอบ งานประกอบ ในปัจจุบัน การพัฒนาหุ่นยนต์มุ่งไปที่การทำให้หุ่นยนต์สามารถมองเห็น ได้ยินและสามารถรับรู้การสัมผัสได้

หุ่นยนต์ใช้กันในงานอุตสาหกรรม โดยเฉพาะระบบอัตโนมัติ ซึ่งมีคอมพิวเตอร์เป็นหัวใจหลัก เทคโนโลยีทางด้านคอมพิวเตอร์ถูกพัฒนาอย่างรวดเร็วในเวลาเพียงไม่กี่ปี และมีการประยุกต์ใช้งานอย่างกว้างขวางในงานอุตสาหกรรม ทำให้ระบบการผลิตมีความยืดหยุ่นมากขึ้น และประหยัดเวลาลง

### 2.1.1 ประโยชน์และการใช้งานของหุ่นยนต์ [3; หน้า 2]

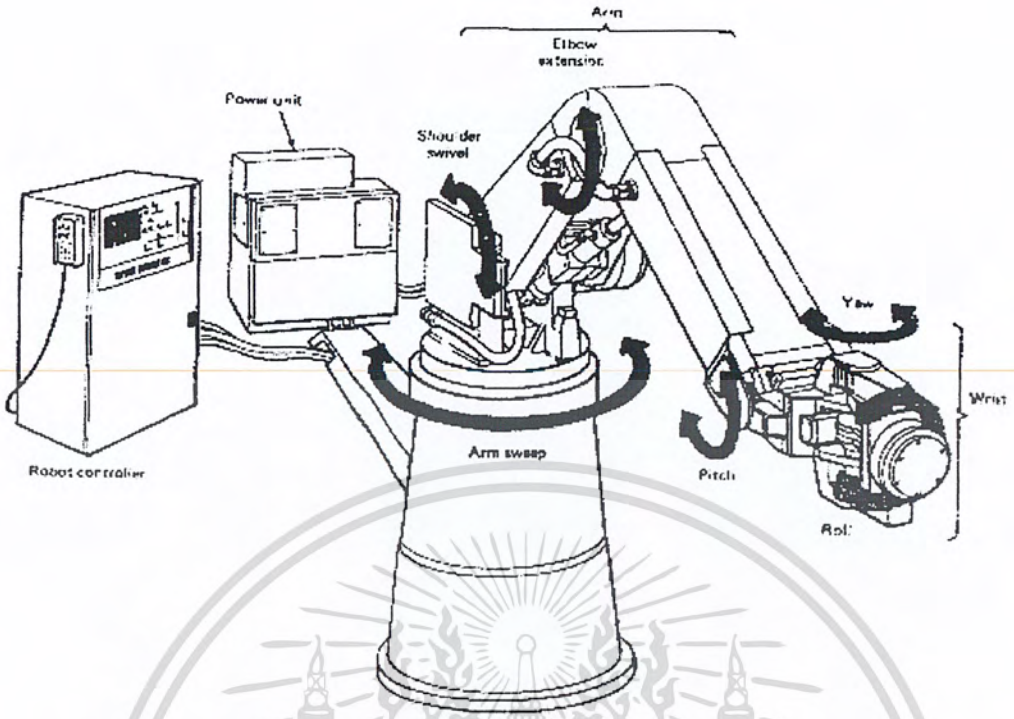
หุ่นยนต์อุตสาหกรรมตัวแรกถูกประดิษฐ์ขึ้นในปี ค.ศ. 1961 ซึ่งเป็นแขนกลที่ทำหน้าที่แทนแรงงานมนุษย์ และต่อมา หุ่นยนต์นับพันๆตัวก็ถูกใช้ในงานอุตสาหกรรมในประเทศสหรัฐอเมริกา ญี่ปุ่นและยุโรป

ในปัจจุบัน หุ่นยนต์อุตสาหกรรมเป็นเครื่องจักรกลไกที่ทำงานภายใต้การควบคุมของคอมพิวเตอร์ หุ่นยนต์ที่ใช้กันมากแสดงไว้ดังรูปที่ 2.1 ซึ่งมีลักษณะคล้ายแขนของมนุษย์ โดยมีจุดเชื่อมต่อที่ทำหน้าที่คล้ายหัวไหล่ ข้อศอกและข้อมือ ซึ่งอาจขับเคลื่อนด้วยระบบไฟฟ้า, ระบบนิวเมติก หรือระบบไฮดรอลิค

โดยคอมพิวเตอร์ จะเป็นหัวใจหลักของระบบหุ่นยนต์ในปัจจุบันถูกบรรจุด้วยโปรแกรมควบคุมและโปรแกรมทำงาน โปรแกรมควบคุมจะถูกจัดเตรียมมาโดยผู้ผลิตหุ่นยนต์ เพื่อควบคุมการทำงานในแต่ละจุดเชื่อมต่อ(joint) ส่วนโปรแกรมการทำงานจัดเขียนขึ้นโดยผู้ใช้หุ่นยนต์ เพื่อบังคับการเคลื่อนไหวของหุ่นยนต์ให้เป็นไปตามลักษณะงานที่ทำ โดยส่งข้อมูลของผู้ใช้ไปให้โปรแกรมควบคุมประมวลผลและปฏิบัติ

หุ่นยนต์อุตสาหกรรมสามารถทำงานได้อย่างมีประสิทธิภาพ ในความสามารถที่ทำงานโดยไม่มีการหยุดพักหรือเหนื่อยล้ากว่าตลอดทั้งวันสม่ำเสมอ, ไม่หลงลืมข้ามงานบางอย่างไป, ทำให้คุณภาพที่ได้สม่ำเสมอ และการควบคุมระดับคุณภาพสามารถกระทำได้โดยง่าย

นอกจากนั้น หุ่นยนต์ยังถูกใช้งานแทนในจุดที่เป็นอันตรายต่อมนุษย์ เช่น หุ่นยนต์ที่ใช้ในการอวกาศซึ่งถูกควบคุมด้วยระยะไกล, ในโรงงานไฟฟ้านิวเคลียร์, ในโรงงานเคมีที่เป็นพิษ, ในงานพ่นสีที่มีละอองสีที่เป็นอันตรายต่อสุขภาพ งานที่ต้องทำซ้ำๆหรืองานที่น่าเบื่อ และงานอีกมากมายที่มนุษย์ไม่พึงที่จะทำ




**รูปที่ 2.1** ตัวอย่างหุ่นยนต์อุตสาหกรรม [3; หน้า 4]

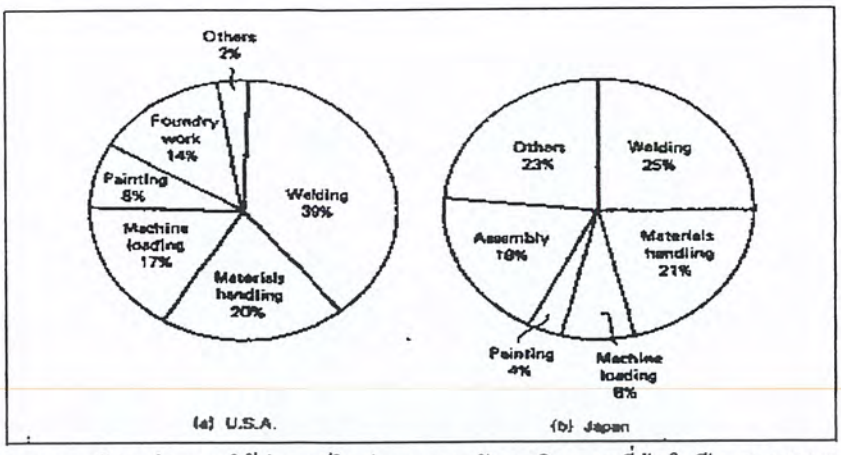
ข้อดีของหุ่นยนต์อุตสาหกรรมสามารถสรุปได้ดังนี้

- ความยืดหยุ่น
- เพิ่มความสามารถในการผลิต
- คุณภาพของผลิตภัณฑ์ที่ดี
- คุณภาพชีวิตของมนุษย์ที่ดีขึ้น

โรงงานอุตสาหกรรมในอนาคตจะมีลักษณะที่มีความยืดหยุ่นสูง และสามารถทำงานได้อย่างรวดเร็ว ซึ่งการผลิตจะเป็นระบบอัตโนมัติ และหุ่นยนต์จะถูกใช้ในการออกแบบ การประกอบ การตรวจสอบและการควบคุมขบวนการผลิตสามารปรับเปลี่ยนได้อย่างง่ายโดยการเปลี่ยนโปรแกรมคอมพิวเตอร์และขั้นตอนการทำงาน

การใช้งานหุ่นยนต์อุตสาหกรรมในปัจจุบันมีดังต่อไปนี้

- การเชื่อมจุด (Spot welding)
- การพ่นสี (Spray painting)
- การประกอบ (Assembly)
- การเชื่อมอาร์ค (Arc welding)
- การตรวจสอบ (Inspection)
- งานหล่อแม่พิมพ์และงานหลอมโลหะ (Die-Casting and forging operations)
- งานเจาะและตัดขึ้นรูปชิ้นส่วนโลหะ (Drilling and deburring of metal parts)




รูปที่ 2.2 แสดงอัตราส่วนการใช้หุ่นยนต์ในประเทศสหรัฐอเมริกาและญี่ปุ่นในปี ค.ศ. 1983 [3; หน้า 7]

### 2.1.2 การใช้งานหุ่นยนต์ในลักษณะอื่นๆ [3; หน้า 7]

นอกจากงานในอุตสาหกรรมแล้ว หุ่นยนต์ยังถูกใช้งานอื่นๆ เช่น ที่ประเทศออสเตรเลีย มีการวิจัยใช้หุ่นยนต์ในการตัดขนแกะ หุ่นยนต์ที่ใช้ในการหุ้มท่อ หุ่นยนต์ที่ใช้ในอวกาศสำหรับปล่อยดาวเทียมและนำดาวเทียมที่เลิกใช้แล้วกลับมา หุ่นยนต์สำหรับช่วยคนพิการในโรงพยาบาล ฯลฯ หุ่นยนต์อีกประเภทหนึ่งที่เป็นความฝันใฝ่ของมนุษย์คือ หุ่นยนต์ที่สามารถทำงานบ้านแทนมนุษย์ได้ เช่น ทำความสะอาดบ้าน ซักผ้า ทำอาหาร ฯลฯ ซึ่งเป็นงานที่ต้องใช้ทักษะและความคิด เช่น ตำแหน่งของเก้าอี้อาจไม่ได้อยู่ประจำที่เดิมทุกวัน กาแฟอาจหกในที่ๆต่างกัน เราอาจชอบรสชาติของอาหารที่ต่างกันในแต่ละวัน เป็นต้น ซึ่งงานเหล่านี้ล้วนยากต่อการตัดสินมใจและต้องใช้ไหวพริบทั้งสิ้น แต่ในปัจจุบันก็เริ่มมีหุ่นยนต์ประเภทนี้ออกมาให้เห็นบ้าง แต่ยังต้องพึ่งการควบคุมจากมนุษย์ เช่น หุ่นยนต์ทำความสะอาดกระจกบนอาคารสูง

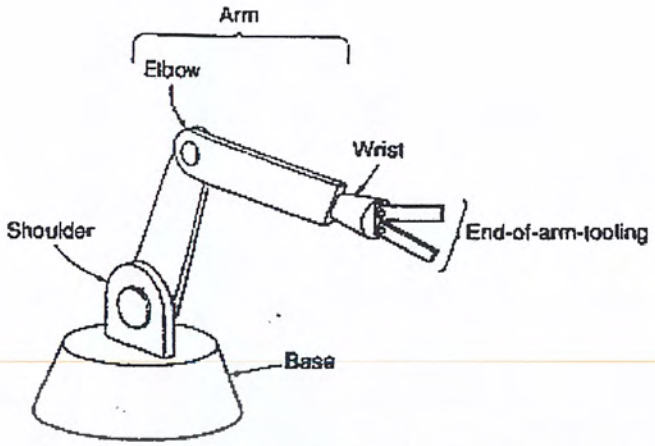
### 2.1.3 โครงสร้างพื้นฐานของหุ่นยนต์ [3; หน้า 11]

คำนิยามที่นิยมใช้สำหรับ "หุ่นยนต์" คือ "เครื่องจักรกลที่สามารถทำงานได้อย่างอัตโนมัติโดยเลียนแบบการทำงานของมนุษย์" หุ่นยนต์สมัยใหม่จะประกอบด้วย 3 ส่วนหลักๆคือ

1. โครงสร้างเชิงกล
2. ระบบขับเคลื่อน
3. คอมพิวเตอร์หรือระบบควบคุม

โครงสร้างเชิงกล ซึ่งประกอบด้วยโครงสร้างภายนอกที่มีลักษณะคล้ายแขน เรียกว่า แมนิพิวเลเตอร์ (Manipulator) และอุปกรณ์ที่บริเวณส่วนปลาย เรียกว่า เอนด์ เอฟเฟกเตอร์ (End Effector)

- แมนิพิวเลเตอร์ จะเป็นส่วนที่เคลื่อนไหวเพื่อให้เอนด์ เอฟเฟกเตอร์สามารถเข้าถึงจุดที่ต้องการได้ ซึ่งในมานิพิวเลเตอร์ยังแบ่งได้เป็น ไหล่ (Shoulder) ข้อศอก (Elbow) และข้อมือ (Wrist) ดังรูปที่ 2.3 บริเวณจุดต่อที่เลื่อนหรือหมุนเรียกว่า ข้อต่อ (joint) ส่วนข้อแต่ละข้อเรียกว่า ลิงค์ (link) หุ่นยนต์ที่มี n มิติจะหมายถึงหุ่นยนต์ที่มี n joint หรือสามารถเคลื่อนที่ได้ n แกน อย่างหุ่นยนต์ 6 แกน จะทำให้ส่วนปลายสามารถเข้าถึงทุกจุดในบริเวณ 3 มิติได้คลอบคลุม แต่ในการใช้งานบางอย่าง จะมีความต้องการของจำนวนแกนอยู่ที่ประมาณ 4-5 แกนเท่านั้น

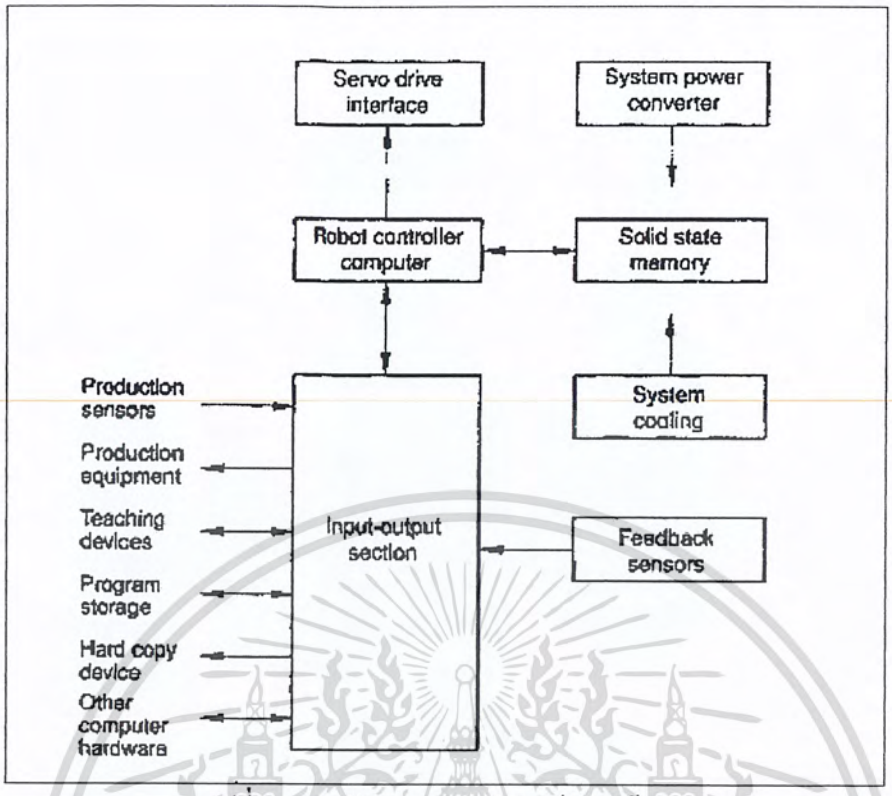



**รูปที่ 2.3** แสดงส่วนประกอบของ Manipulator [4; หน้า 2]

- เอนด์ เอฟเฟกเตอร์ เป็นอุปกรณ์ที่ติดตั้งอยู่บริเวณปลายข้อมือ เพื่อให้ทำงานตามที่กำหนดไว้ ซึ่งเอนด์ เอฟเฟกเตอร์ อาจจะเป็นหัวเชื่อม ปืนพ่นสี มือจับ เป็นต้น โดยทั่วไป ในการกำหนดตำแหน่งของหุ่นยนต์มักจะใช้บริเวณจุดปลายของ End Effector เป็นจุดกำหนด ซึ่งเรียกว่า จุดศูนย์กลางของเครื่องมือ (Tool center point; TCP)

ระบบขับเคลื่อนหรือระบบส่งกำลัง โดยทั่วไปจะแบ่งออกเป็น 3 ชนิดตามลำดับความนิยมการใช้งาน คือ ระบบมอเตอร์ไฟฟ้า, มอเตอร์, นิวเมติก และไฮดรอลิค ซึ่งเรียกรวมว่า Actuators ในหุ่นยนต์บางตัวอาจใช้ระบบขับเคลื่อนผสมกัน การใช้ระบบนิวเมติก, ไฮดรอลิค ส่วนใหญ่จะพบในหุ่นยนต์ชนิดนอนเซอร์โว ซึ่งเป็นการควบคุมแบบเปิด (Open loop system) มีการบอกตำแหน่งต่างๆเพียงไม่กี่ตำแหน่งโดยใช้ลิมิตสวิตช์ เป็นการทำงานแบบเป็นลำดับขั้น ส่วนหุ่นยนต์ชนิดเซอร์โว จะเป็นการควบคุมแบบระบบปิด (Close loop system) โดยมีการป้อนกลับของสัญญาณเพื่อบอกตำแหน่งต่างๆอย่างละเอียด ทำให้สามารถควบคุมตำแหน่งและความเร็ว ความเร่งได้อย่างแม่นยำ โดยทั่วไปจะใช้ระบบขับเคลื่อนด้วยเซอร์โวมอเตอร์

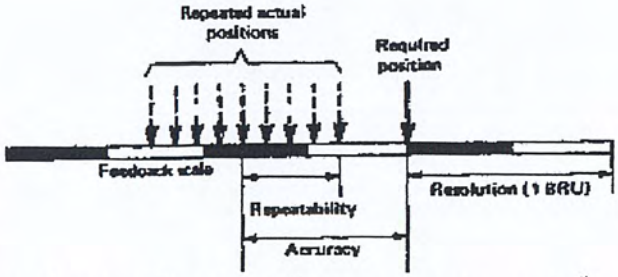
คอมพิวเตอร์ หรือชุดควบคุม เป็นส่วนที่ประมวลผลและทำหน้าที่เชื่อมต่อกับอุปกรณ์อินพุท/เอาท์พุท เพื่อควบคุมการเคลื่อนไหวของมานิพิวเลเตอร์ และลำดับการทำงานของหุ่นยนต์ ส่วนประกอบของชุดควบคุม สามารถแสดงดังรูปที่ 2.4



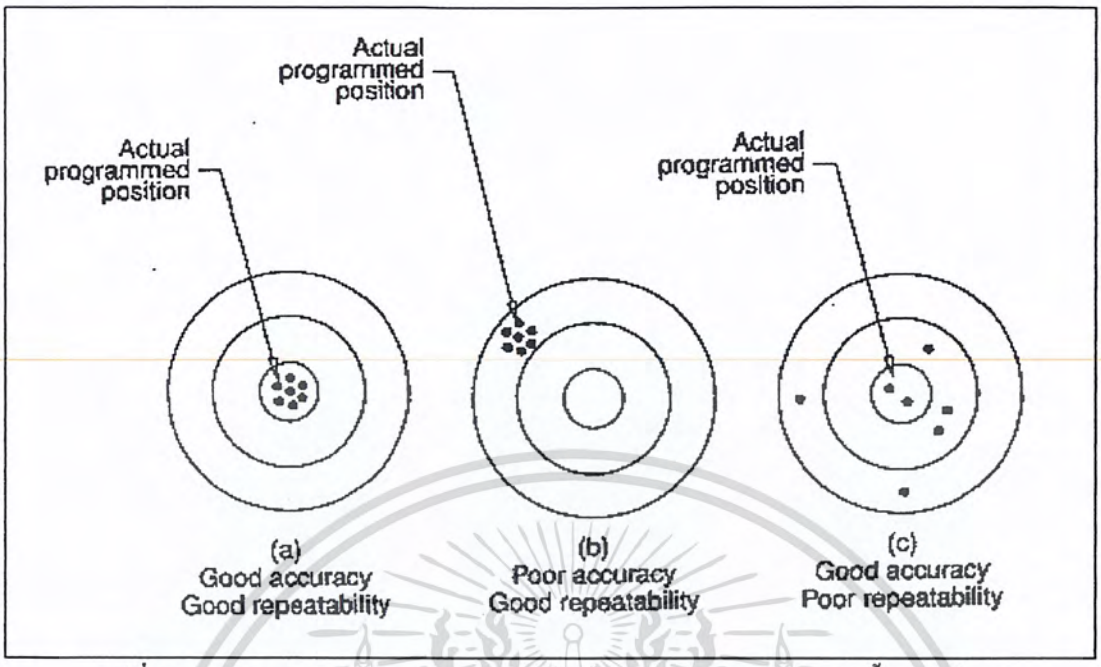

**รูปที่ 2.4** แผนผังการควบคุมหุ่นยนต์ [4; หน้า 3]

ส่วนความถูกต้องแม่นยำของหุ่นยนต์จะขึ้นอยู่กับระบบเชิงกล, ความละเอียดของหุ่นยนต์และโปรแกรมการควบคุม ข้อผิดพลาดเชิงกลได้แก่การต่อเชื่อมไม่สนิท (Backlash) ที่เกิดขึ้นในระบบเกียร์, ลีดสกรู และในกระบอกไฮดรอลิค นอกจากนี้ ระบบเชิงกลที่ไม่ดียังมีผลในกรณีที่หุ่นยนต์มีภาระที่แตกต่างกัน ส่วนความผิดพลาดที่เกิดจากโปรแกรมนั้นอาจเนื่องมาจากการปัดเศษ เป็นต้น เพื่อแก้ไขข้อผิดพลาดดังกล่าว ผู้ออกแบบหุ่นยนต์มักจะคำนึงถึงตำแหน่งในโปรแกรมอยู่ที่ค่าต่ำกว่า 1 BLU เช่น 1/2 BLU

ส่วนความสามารถในการทำซ้ำของหุ่นยนต์จะมีส่วนสัมพันธ์กับความถูกต้องของหุ่นยนต์ ถ้าหากเราให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งเดิมหลายๆครั้งภายใต้สภาวะแวดล้อมเดียวกัน เราจะพบว่า ตำแหน่งที่ได้อาจไม่เป็นจุดเดิมเสมอไป ซึ่งความสามารถในการทำซ้ำของหุ่นยนต์จะหาได้จากค่าเฉลี่ยของระยะผิดพลาด เช่น หุ่นยนต์ที่มีความสามารถในการทำซ้ำ $\pm 0.2$ มิลลิเมตร หมายถึง ตำแหน่งที่ได้จริงจะอยู่ภายในรัศมี 0.2 มิลลิเมตร วัดจากจุดที่สั่ง ในทางปฏิบัติแล้ว ความสามารถในการทำซ้ำของหุ่นยนต์จะมีความสำคัญมากกว่าความถูกต้องของหุ่นยนต์




**รูปที่ 2.5** แสดงค่าความละเอียด ความถูกต้องและความสามารถการทำซ้ำ [4; หน้า 3]




รูปที่ 2.6 แสดงความแตกต่างระหว่างความแม่นยำ และความสามารถในการทำซ้ำ [4; หน้า 4]

### 2.1.4 การแสดงตำแหน่งของหุ่นยนต์ [3; หน้า 19]

การแสดงตำแหน่งของหุ่นยนต์มักจะแสดงเป็นเลขจำนวนเต็ม ซึ่งเลขดังกล่าวจะเป็นตำแหน่งของหุ่นยนต์ในแต่ละแกน โดยทั่วไปจะใช้ค่าความละเอียดที่ได้จากสัญญาณป้อนกลับมาเป็นค่ากำหนดความละเอียดในหนึ่งหน่วย BLU เช่นให้สัญญาณที่ได้จาก Encoder 1 Pulse = 1 BLU

ส่วนตำแหน่งที่เก็บนั้นจะถูกเก็บเป็นเลขฐาน 2 ในระบบคอมพิวเตอร์ ซึ่งอาจจะมีจำนวนบิต ไม่เท่ากันขึ้นอยู่กับความละเอียดที่ต้องการและระยะสูงสุดที่ต้องการเคลื่อนที่ เช่น ต้องการความละเอียดของตำแหน่ง 0.01 มิลลิเมตร ระยะสูงสุดที่ต้องการเป็น 40 เซนติเมตร ดังนั้น จะมีตำแหน่งทั้งหมดเท่ากับ40,000 จุด ซึ่งต้องใช้จำนวนบิตเท่ากับ 16 บิต (ค่าสูงสุด = 65536 ตำแหน่ง) เป็นต้น ซึ่งในหุ่นยนต์อุตสาหกรรมแล้ว จำนวนบิตที่ใช้จะมีขนาด 32 บิตหรือ 64 บิต

### 2.1.5 การจัดประเภทของหุ่นยนต์อุตสาหกรรม [4; หน้า 1]

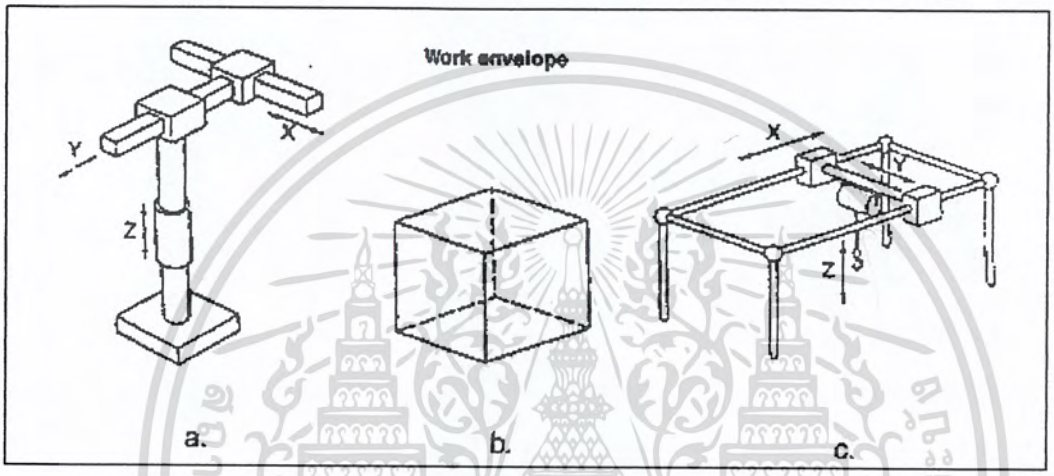
สามารถแยกตามลักษณะการจัดดังนี้

- จัดตามพิกัดของแขนแมนิพิวเลเตอร์ ซึ่งแบ่งออกเป็น 4 แบบ คือ หุ่นยนต์แบบคาร์ทีเชียน (Cartesian Robot), หุ่นยนต์แบบไซลินดริคอล (Cylindrical Robot), หุ่นยนต์แบบสเฟอริคอล (Spherical Robot), และหุ่นยนต์แบบอาร์ติคิวเลท (Articulated Robot)
- จัดตามมิติ
- จัดตามชนิดของแหล่งพลังงาน
- จัดตามลักษณะการเคลื่อนที่
- จัดตามลักษณะการควบคุมเส้นทาง
- จัดตามระดับความฉลาด

การจัดตามพิกัดของแมนิพิวเลเตอร์ จะพิจารณาจากการเคลื่อนไหวของแกนในการทำงาน ซึ่งในระบบ 3 มิติ หุ่นยนต์จะต้องสามารถเข้าถึงจุดต่างๆ ได้โดยการเคลื่อนที่ไปหน้า-ถอยหลัง ซ้าย-ขวา ขึ้น-ลง

**2.1.5.1 หุ่นยนต์แบบคาร์ทีเชียน (Cartesian Robot) [4; หน้า 1]**

หุ่นยนต์แบบคาร์ทีเชียน จะมีลักษณะคือ มี 3 แกนที่สามารถเคลื่อนที่เป็นเส้นตรงได้อย่างอิสระ โดยแกนที่เคลื่อนที่ซ้าย-ขวาจะเป็นแกน X แกนที่เคลื่อนที่ไปหน้า-หลังจะเป็นแกน Y ส่วนแกนที่เคลื่อนที่ขึ้น-ลงจะเป็นแกน Z แสดงดังรูปที่2.7




รูปที่ 2.7 [4; หน้า 6]

a) รูปแสดงการเคลื่อนที่ของหุ่นยนต์แบบคาร์ทีเชียน

b) ปริมาตรงานของหุ่นยนต์แบบคาร์ทีเชียน

c) การเคลื่อนที่ของเครน ที่มีลักษณะเหมือนการเคลื่อนที่ของหุ่นยนต์แบบคาร์ทีเชียน

ปริมาตรงาน (Work volume) ของหุ่นยนต์แบบคาร์ทีเชียน จะมีลักษณะเป็นสี่เหลี่ยมลูกบาศก์ ระบบขับเคลื่อนในหุ่นยนต์แบบนี้มักจะเป็นแบบเชิงเส้นหรือใช้บอล-สกรู

ข้อดีของหุ่นยนต์แบบคาร์ทีเชียน มีดังนี้

- การขยายปริมาตรงานสามารถทำได้ง่าย เนื่องจากมีการเคลื่อนที่แบบเชิงเส้นและแต่ละแกนแยกอิสระจากกัน
- การควบคุมทำได้ง่าย
- โครงสร้างค่อนข้างแข็งแรงและความถูกต้องแม่นยำสูง
- สามารถรับภาระหนักได้ เนื่องจากมีความสามารถในการรับภาระที่ตำแหน่งต่างๆ ไม่แตกต่างกันมาก

ข้อเสียของหุ่นยนต์แบบคาร์ทีเชียน มีดังนี้

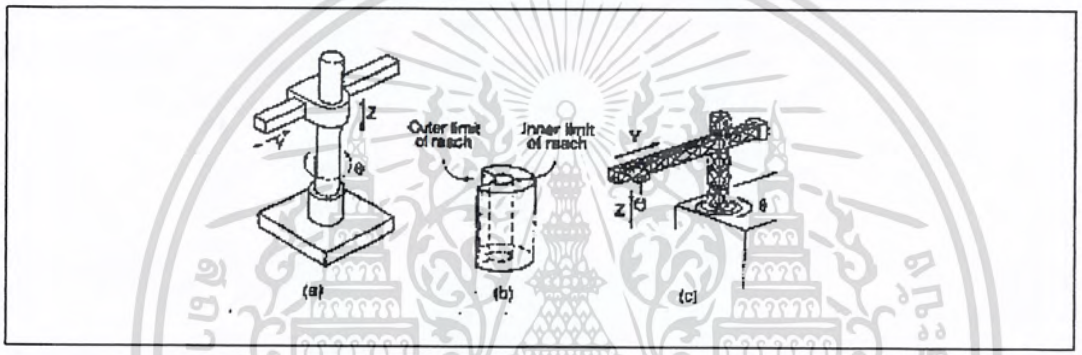
- การบำรุงรักษาทำได้ลำบากในบางแบบ
- การเคลื่อนที่สามารถได้ในทิศทางเดียว ณ เวลาหนึ่ง
- ความยืดหยุ่นเชิงกลต่ำ

การประยุกต์ใช้งาน

- งานหยิบวางวัสดุ
- งานประกอบชิ้นส่วน
- เครื่องซีเอ็นซี (CNC) ที่มีระบบโหลดอัตโนมัติ
- งานตรวจสอบ
- งานเชื่อม
- อื่นๆ

**2.1.5.2 หุ่นยนต์แบบไซลินดริคอล (Cylindrical Robot) [4; หน้า 3]**

ประกอบด้วยแกนที่เคลื่อนที่เป็นเส้นตรง 2 แกน และแกนหมุน 1 แกน แกนที่หมุนจะเป็นแกนของฐาน ส่วนแกนที่เคลื่อนที่เป็นเส้นตรงจะเป็นแกนที่เลื่อนเข้าออกและขึ้นลง ซึ่งแสดงดังรูปที่ 2.8




รูปที่ 2.8 [4; หน้า 7]

a) รูปแสดงการเคลื่อนที่ของหุ่นยนต์แบบไซลินดริคอล

b) ปริมาตรงานของหุ่นยนต์แบบไซลินดริคอล

c) การเคลื่อนที่ของเครนที่มีลักษณะเหมือนการเคลื่อนที่ของหุ่นยนต์แบบไซลินดริคอล

ความละเอียดของหุ่นยนต์แบบไซลินดริคอล แรงบิดที่ต้องการจากมอเตอร์และค่าโมเมนต์ความเฉื่อย จะมีค่าไม่เท่ากันในแกนหมุน ซึ่งขึ้นอยู่กับรัศมีหรือระยะห่างระหว่างมือจับถึงจุดหมุน

ข้อดีของหุ่นยนต์แบบไซลินดริคอล มีดังนี้

- มีปริมาตรงานมากกว่าหุ่นยนต์แบบคาร์ทีเซียน ทำให้เข้าถึงตำแหน่งในแนวนอนได้ไกลกว่า
- มีโครงสร้างที่สามารถเข้าถึงได้เป็นชั้นๆ
- มีความสามารถในการรับภาระสูง

ข้อเสียของหุ่นยนต์แบบไซลินดริคอล มีดังนี้

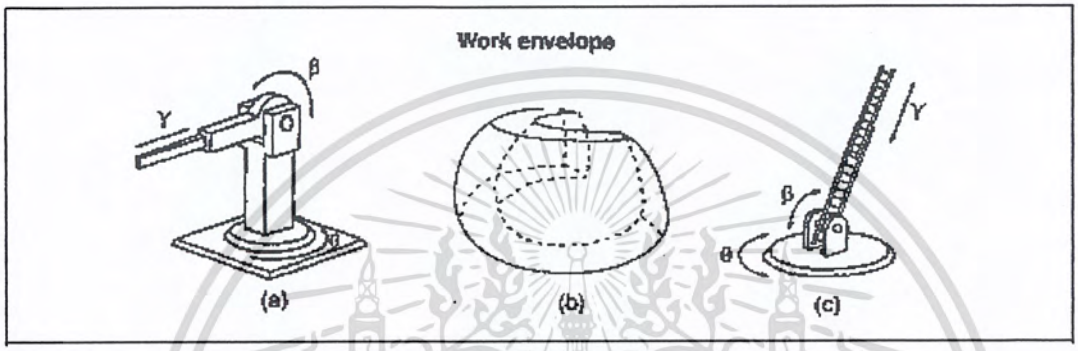
- โครงสร้างเชิงกลมีความแข็งแรงต่ำกว่าหุ่นยนต์แบบคาร์ทีเซียน
- ความถูกต้องต่ำในแกนหมุน
- การควบคุมทำได้ยากกว่าหุ่นยนต์แบบคาร์ทีเซียน

การประยุกต์ใช้งาน

- งานประกอบ
- งานตรวจสอบ
- อื่นๆ

**2.1.5.3 หุ่นยนต์แบบสเฟอริคอล (Spherical Robot) [4; หน้า 4]**

ประกอบด้วย 1 แกนที่เคลื่อนที่แบบเชิงเส้น และแกนหมุน 2 แกน ปริมาตรงานจะมีลักษณะคล้ายส่วนหนึ่งของทรงกลม ดังในรูปที่ 2.9




**รูปที่ 2.9** [4; หน้า 8]

a) รูปแสดงการเคลื่อนที่ของหุ่นยนต์แบบสเฟอริคอล

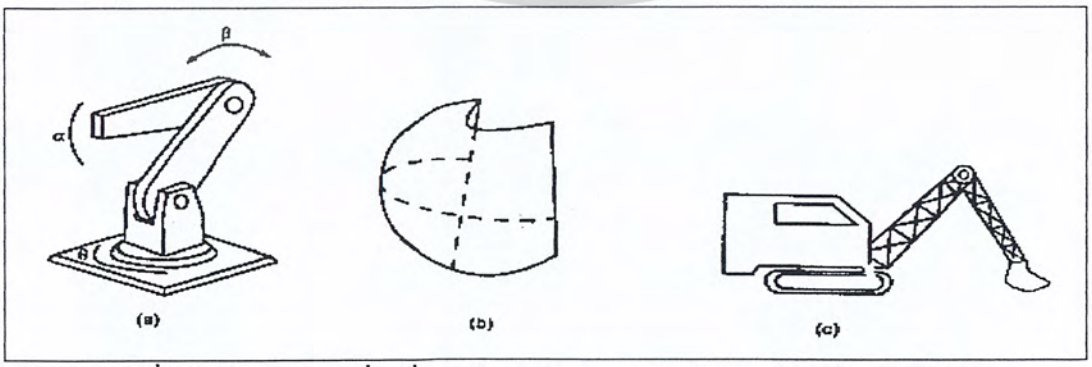
b) ปริมาตรงานของหุ่นยนต์แบบสเฟอริคอล

c) การเคลื่อนที่ของบันไดที่มีลักษณะเหมือนการเคลื่อนที่ของหุ่นยนต์แบบสเฟอริคอล

ลักษณะการเคลื่อนที่จะคล้ายหุ่นยนต์แบบไซลินดริคอลเพียงแต่การเคลื่อนที่ในแนวดิ่งจะถูกจำกัด

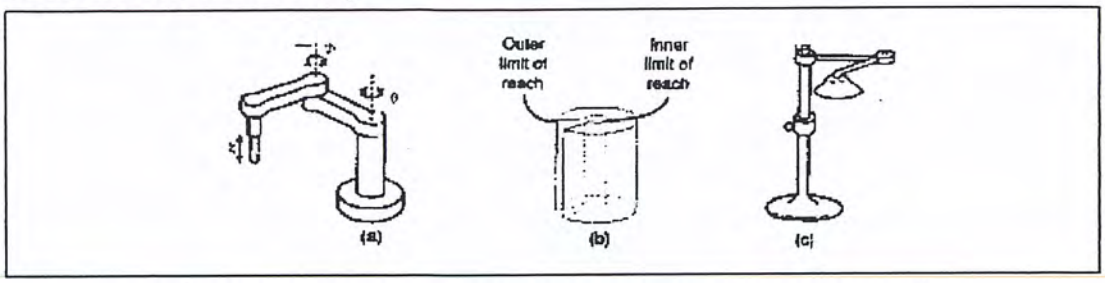
**2.1.5.4 หุ่นยนต์แบบอาร์ติคิวเลท (Articulated Robot) [4; หน้า 4]**

ทุกแกนจะเป็นแกนหมุนหมด ซึ่งมีลักษณะคล้ายกับแขนของคน สามารถแบ่งได้ 2 แบบ คือ จุดหมุนอยู่ในแนวดิ่ง และจุดหมุนอยู่ในแนวนอน




**รูปที่ 2.10** ลักษณะการเคลื่อนที่และปริมาตรงานของหุ่นยนต์แบบอาร์ติคิวเลท [4; หน้า 8]

ลักษณะของหุ่นยนต์แบบอาร์ติคิวเลท แบบแนวนอนหรือที่เรียกว่า หุ่นยนต์แบบสกาล่าจะมีลักษณะพิเศษคือ ประกอบด้วยแกนที่เคลื่อนที่เป็นเชิงเส้นในแนวดิ่ง 1 แกน (แกน Z) หุ่นยนต์แบบสกาล่ามักถูกใช้ในงานประกอบที่ต้องนำชิ้นส่วนเข้าไปในรู




**รูปที่ 2.11** ลักษณะการเคลื่อนที่และปริมาตรงานของหุ่นยนต์แบบสกาล่า [4; หน้า 9]

ข้อดีของหุ่นยนต์แบบอาร์ติคิวเลท โดยรวมแล้ว จะมีปริมาตรงานที่กว้างกว่าแบบอื่น ใช้เนื้อที่ในการติดตั้งน้อย และสามารถเข้าถึงตำแหน่งต่างๆได้หลายลักษณะ ส่วนข้อเสียเปรียบคือ การควบคุมทำได้ยาก อันเนื่องมาจากสภาวะของโหลด และโมเมนต์ของความเฉื่อยไม่คงที่

**2.1.5.5 จัดตามลักษณะการเคลื่อนที่ [4; หน้า 5]**

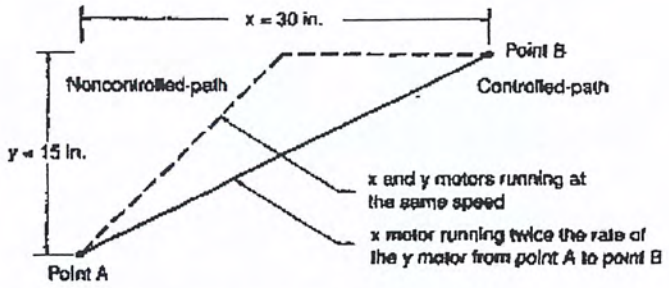
การเคลื่อนที่จากจุดหนึ่งไปยังอีกจุดหนึ่งของมานิพิวเลเตอร์สามารถแบ่งได้เป็น 4 แบบ คือ

- Slew motion แต่ละแกนจะเคลื่อนที่ด้วยความเร็วคงที่ตามที่คำนวณได้จากตำแหน่งเริ่มต้นไปยังตำแหน่งปลายทาง (กำหนดเวลาที่ต้องการ หาความเร็ว)
- Joint-interpolated motion กำหนดความเร็วในการเคลื่อนที่แต่ละแกนคงที่ หาเวลาที่ต้องการใช้มากที่สุด (แต่ละแกนใช้เวลาไม่เท่ากัน) จากนั้นจึงนำไปปรับความเร็วของแกนที่ใช้เวลาน้อยกว่าให้มีความเร็วลดลง
- Straight-line interpolation motion การเคลื่อนที่ของเอนด์ เอฟเฟกเตอร์จากจุดต้นไปยังจุดปลายเป็นแนวเส้นตรงในระบบพิกัดฉาก จากนั้นจึงแบ่งการเคลื่อนที่ไปตามเส้นตรงออกเป็นส่วนย่อยๆ
- Circular interpolation motion การเคลื่อนที่จากตำแหน่งต้นทางไปยังปลายทางจะมีลักษณะโค้ง จากนั้นจึงแบ่งส่วนโค้งออกเป็นเส้นตรงเล็กๆที่เชื่อมต่อกัน

**2.1.5.6 จัดตามลักษณะการควบคุมเส้นทาง [4; หน้า 5]**

ลักษณะการควบคุมเส้นทางแบ่งออกเป็น 4 แบบ คือ

- Limited-sequence ไม่ถูกใช้ในหุ่นยนต์ที่ควบคุมแบบเซอร์โว การทำงานจะมีลักษณะปืนไปตามลำดับโดยใช้ลิมิสวิตช์ ลักษณะงานที่ใช้จะค่อนข้างตายตัว
- Point-to-point ใช้แบ่งเส้นทางการเดินออกเป็นเส้นตรงหลายๆจุดต่อกัน แต่ไม่จำเป็นต้องเคลื่อนที่ทีละแกน (เดินแล้วหยุด)
- Controlled-path เส้นทางการเคลื่อนที่จะถูกควบคุมโดยตัวควบคุม ความเร็วการเคลื่อนที่ในแต่ละแกนจะสัมพันธ์กัน ความแตกต่างระหว่าง Point-to-point และ Controlled-path แสดงในรูปที่12 เส้นทางการเคลื่อนที่อาจจะเป็นเส้นตรง วงกลมหรือส่วนโค้งก็ได้




**รูปที่ 2.12** ความแตกต่างระหว่าง Point-to-point และ Controlled-path [4; หน้า 10]

4. Continuous-path เส้นทางการเคลื่อนที่จะมีลักษณะอิสระโดยการแบ่งออกเป็นจุดต่างๆเป็นจำนวนมาก ทำให้การเคลื่อนที่มีลักษณะที่ดูต่อเนื่อง

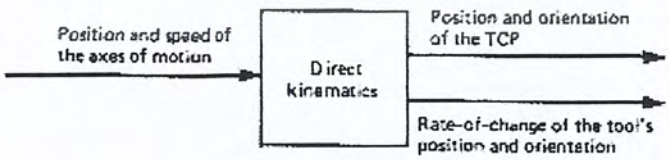
## 2.2 การวิเคราะห์คิเนเมติกส์และการเปลี่ยนตำแหน่งโคออร์ดิเนต (Kinemaitic Analysis and Coordinate Transformation) [3; หน้า 83]

ในระบบหุ่นยนต์ การแก้ปัญหาของไดเร็ค คิเนเมติกส์ (Direct Kinemarics) จะเกี่ยวข้องกับการกำหนดตำแหน่งและทิศทางและอัตราการเปลี่ยนแปลงของเอนด์ เอฟเฟกเตอร์ เหมือนกับฟังก์ชันของตำแหน่งและความเร็วของแกนที่เคลื่อนที่ แสดงในรูปที่ 2.13 ตำแหน่งของเอนด์ เอฟเฟกเตอร์ หรือเครื่องมือจะถูกกำหนดที่ TCP ตัวอย่างคือ ขอบของปืนเชื่อม หรือ ศูนย์กลางของมือจับ

การวิเคราะห์ไดเร็ค คิเนเมติกส์ จะใช้การแก้ปัญหาโดยวิธีทางเรขาคณิตโดยตรง ซึ่งจะอภิปรายในหัวข้อ 2.3.2 วิธีนี้เหมาะสำหรับ โครงสร้างคิเนเมติกส์อย่างง่าย แต่เป็นการยากในการประยุกต์เมื่อเกี่ยวข้องกับโครงสร้างที่มีหลายตัวแปรอิสระ

สำหรับการแก้ปัญหาโดยวิธีพื้นฐานของวิธีการแปลงเมตริกซ์ เปรียบเทียบกับวิธีการในตอนแรก วิธีนี้จะเหมาะสำหรับวิเคราะห์โครงสร้างที่มีหลายตัวแปรอิสระ และการใช้ประโยชน์ของเทคนิคอินเวอร์ส คิเนเมติกส์ (Inverse Kinematics) และการวิเคราะห์ทางไดนามิกส์ของแมนนิพิวเลเตอร์

การใช้เมตริกซ์ของการหมุน จะแสดงในหัวข้อ 2.2.6 ในการประยุกต์วิธีเพื่อแก้ปัญหาของอินเวอร์ส คิเนเมติกส์ ซึ่งสามารถใช้ได้หลายกรณี




**รูปที่ 2.13** การแปลงโดยใช้ Direct Kinematics [3; หน้า 84]

### 2.2.1 ปัญหาของไดเร็ค คิเนเมติกส์ ในหุ่นยนต์อุตสาหกรรม [3; หน้า 83]

การวิเคราะห์คิเนเมติกส์ ของระบบทางกลศาสตร์หมายถึง การกำหนดตำแหน่ง ความเร็ว และความเร่งขององค์ประกอบกลไกหลักที่พิจารณาการเปลี่ยนแปลงทางกลศาสตร์ การรวมกันของตำแหน่ง และความเร็วขององค์ประกอบหลักที่เวลาที่แน่นอนจะเกี่ยวข้องถึงสถานะขององค์ประกอบหลักนี้ สิ่งที่มีผลเกี่ยวข้องคือ แรงและทอร์กซึ่งจะมีผลกับมวลและภายในขององค์ประกอบหลัก ซึ่งจะพิจารณาในทางไดนามิกส์มากกว่าคิเนเมติกส์ และไม่อภิปรายกันในหัวข้อนี้ การคำนวณสถานะทางคิเนเมติกขององค์ประกอบทางกลศาสตร์ (เช่น ตำแหน่ง ทิศทาง และอัตราการเปลี่ยนแปลง) บนพื้นฐานของระบบแกนที่เคลื่อนที่จะเกี่ยวข้องกับปัญหาไดเร็ค คิเนเมติกส์ ปัญหาอินเวอร์ส คิเนเมติกส์ จะเกี่ยวกับการคำนวณของแกนที่เคลื่อนที่ซึ่งจะแปลงเป็น แขนที่เคลื่อนที่

ในกรณีของแมนิพิวเลเตอร์ นี้ การวิเคราะห์ไดเร็ค คิเนเมติกส์ จะกำหนดสถานะของเอนด์ เอฟเฟกเตอร์ เหมือนกับฟังก์ชันที่ทราบสถานะของการเปลี่ยนแปลงของข้อต่อ รวมไปถึงสถานะของเอนด์ เอฟเฟกเตอร์ การวิเคราะห์ Direct Kinematics จะรวมไปถึงการกำหนดตำแหน่งและทิศทางของลิงค์ และเวลาที่ใช้ ดังนั้นข้อมูลจะเป็นสิ่งจำเป็นเพื่อใช้ต่อในการวิเคราะห์ทางไดนามิกส์ หรือเมื่อตำแหน่งและทิศทางของเซนเซอร์ ที่ประกอบอยู่บนลิงค์เกี่ยวข้องกับการดำเนินการทางข้อมูล

ในหัวข้อนี้ หลายๆเทคนิคที่เกี่ยวกับการวิเคราะห์สำหรับการแก้ปัญหาของไดเร็ค คิเนเมติกส์ จะถูกแสดง เทคนิคเหล่านี้สามารถใช้ประโยชน์ไม่เฉพาะสำหรับการวิเคราะห์การเคลื่อนที่ของแมนิพิวเลเตอร์ แต่สามารถใช้ได้กับการเคลื่อนอย่างอื่นที่สัมพันธ์กับ ปัญหาระบบหุ่นยนต์ เช่น การเปลี่ยนตำแหน่งพิกัด สำหรับตัวอย่าง ของระบบพิกัด ที่ติดบนเซนเซอร์สามารถแปลงให้อยู่ในรูป world coordinate system (WCS) ซึ่งจะกำหนดตำแหน่งของฐานของหุ่นยนต์ ตัวอย่างอื่นๆคือ tool coordinate system (TCS) ซึ่งติดอยู่กับเอนด์ เอฟเฟกเตอร์ สามารถแปลงเป็นระบบพิกัดอื่นได้ ข้อมูลที่ถูกแปลงนี้สามารถนำมาเป็นเวกเตอร์ของความเร็วที่ต้องการที่กำหนดใน TCS หรือเป็นเวกเตอร์ของตำแหน่ง ในพิกัดของ sensor coordinate (SCS)

การวิเคราะห์ไดเร็ค คิเนเมติกส์ เป็นพื้นฐานของการแก้ปัญหาอินเวอร์ส คิเนเมติกส์ การแก้ปัญหาของอินเวอร์ส คิเนเมติกส์ จะกลายเป็นพื้นฐานของ Interpolator Algorithm ซึ่งเป็นแนวทางให้เอนด์ เอฟเฟกเตอร์เคลื่อนไปตามพื้นที่ที่ต้องการ Interpolator Supply เป็นสัญลักษณ์ของ control loop ซึ่งจะส่งสัญญาณกลับเพื่อให้เอนด์ เอฟเฟกเตอร์ ไปตามทางโคจรที่ต้องการ

ไดเร็ค คิเนเมติกส์ และการแก้ปัญหาอินเวอร์ส คิเนเมติกส์ เป็นเครื่องมือใน path-planning algorithms ของ CP robots โดยพื้นฐานไดเร็ค คิเนเมติกส์ ถูกใช้แปลง Trajectory Point ให้ใน joint coordinate system (JCS) หรือ TCS เป็น WCS และอินเวอร์ส คิเนเมติกส์ ถูกใช้แปลง Trajectory Point ในแกนการเคลื่อนที่

หัวข้อนี้จะเกี่ยวข้องกับข้อต่อ (Joint) อย่างง่าย (ซึ่งเป็นลิงค์ที่มีตัวแปรอิสระเพียง 1 ตัวแปรจากข้อต่อ) แต่ไม่เกี่ยวกับข้อต่อที่มีความซับซ้อน

### 2.2.2 การวิเคราะห์ไดเร็ค คิเนเมติกส์ โดยใช้พื้นฐานทางเรขาคณิต [3; หน้า 85]

สองตัวแปรอิสระในกลไกที่มีระนาบแบบอาร์ติคิวเลท แสดงในรูปที่ 2.14 *a* ลิงค์ที่ 2 หมุนรอบจอยท์ที่ 2 เป็นมุม $\theta_2$ วัดเทียบกับระบบพิกัดอ้างอิง $(X_1, Y_1)$ ซึ่งติดอยู่กับลิงค์ที่ 1 นอกจากนั้นลิงค์ที่ 3 หมุนรอบจอยท์ที่ 3 เป็นมุม $\theta_3$ เทียบกับลิงค์ที่ 2 TCPจะอยู่ที่ปลายของลิงค์ที่ 3

ปัญหา Direct Kinematics ของการกำหนดตำแหน่งของ TCP $(X_t, Y_t)$ จะเหมือนกับฟังก์ชันของค่ามุมของจอยท์สามารถแก้ได้โดยใช้เรขาคณิตโดยตรง

$$X_t = a_2 \cos\theta_2 + a_3 \cos\psi$$
$$Y_t = a_2 \sin\theta_2 + a_3 \sin\psi \quad (1)$$

ซึ่ง $\psi = \theta_2 + \theta_3$ นำเวลาไปหารสมการ (1) ได้ความเร็วของ TCP ในระบบพิกัดอ้างอิง $(X_t, Y_t)$:

$$\dot{X}_t = -[a_2 \sin\theta_2 + a_3 \sin\psi]\dot{\theta}_2 - [a_3 \sin\psi]\dot{\theta}_3$$
$$\dot{Y}_t = [a_2 \cos\theta_2 + a_3 \cos\psi]\dot{\theta}_2 + [a_3 \cos\psi]\dot{\theta}_3 \quad (2)$$

สมการ (1) และ (2) แสดงการแก้ปัญหาแบบไดเร็ค คิเนเมติกส์ สำหรับกลไกที่แสดงในรูปที่ 2.14 *a*

การแก้ปัญหาที่เหมือนกันสามารถเจอได้ในโครงสร้างแบบสเฟอริคอลที่มี 3 ตัวแปรอิสระ ดังแสดงในรูปที่ 14 *b* ในกรณีนี้ ระบบพิกัดอ้างอิง $(X_0, Y_0, Z_0)$ จะติดอยู่กับฐาน ซึ่งจะอ้างอิงกับลิงค์ O ด้วย ตำแหน่งของ TCP แสดงในระบบพิกัดที่ฐาน คือ

$$X_t = d_3 \cos\theta_1 \cos\theta_2$$
$$Y_t = d_3 \sin\theta_1 \cos\theta_2 \quad (3)$$
$$Z_t = d_3 \sin\theta_2$$

และความเร็วของ TCP ได้มาจากการนำเวลาหารสมการ (3)

$$\dot{X}_t = (\cos\theta_1 \cos\theta_2)\dot{d}_3 - (d_3 \sin\theta_1 \cos\theta_2)\dot{\theta}_1 - (d_3 \cos\theta_1 \sin\theta_2)\dot{\theta}_2$$
$$\dot{Y}_t = (\sin\theta_1 \cos\theta_2)\dot{d}_3 - (d_3 \cos\theta_1 \cos\theta_2)\dot{\theta}_1 - (d_3 \sin\theta_1 \sin\theta_2)\dot{\theta}_2 \quad (4)$$
$$\dot{Z}_t = (\sin\theta_2)\dot{d}_3 + (d_3 \cos\theta_2)\dot{\theta}_2$$

สมการ (3) และ (4) เป็นการแก้ปัญหา Direct Kinematics สำหรับกลไกในรูปที่ 2.14 *b*

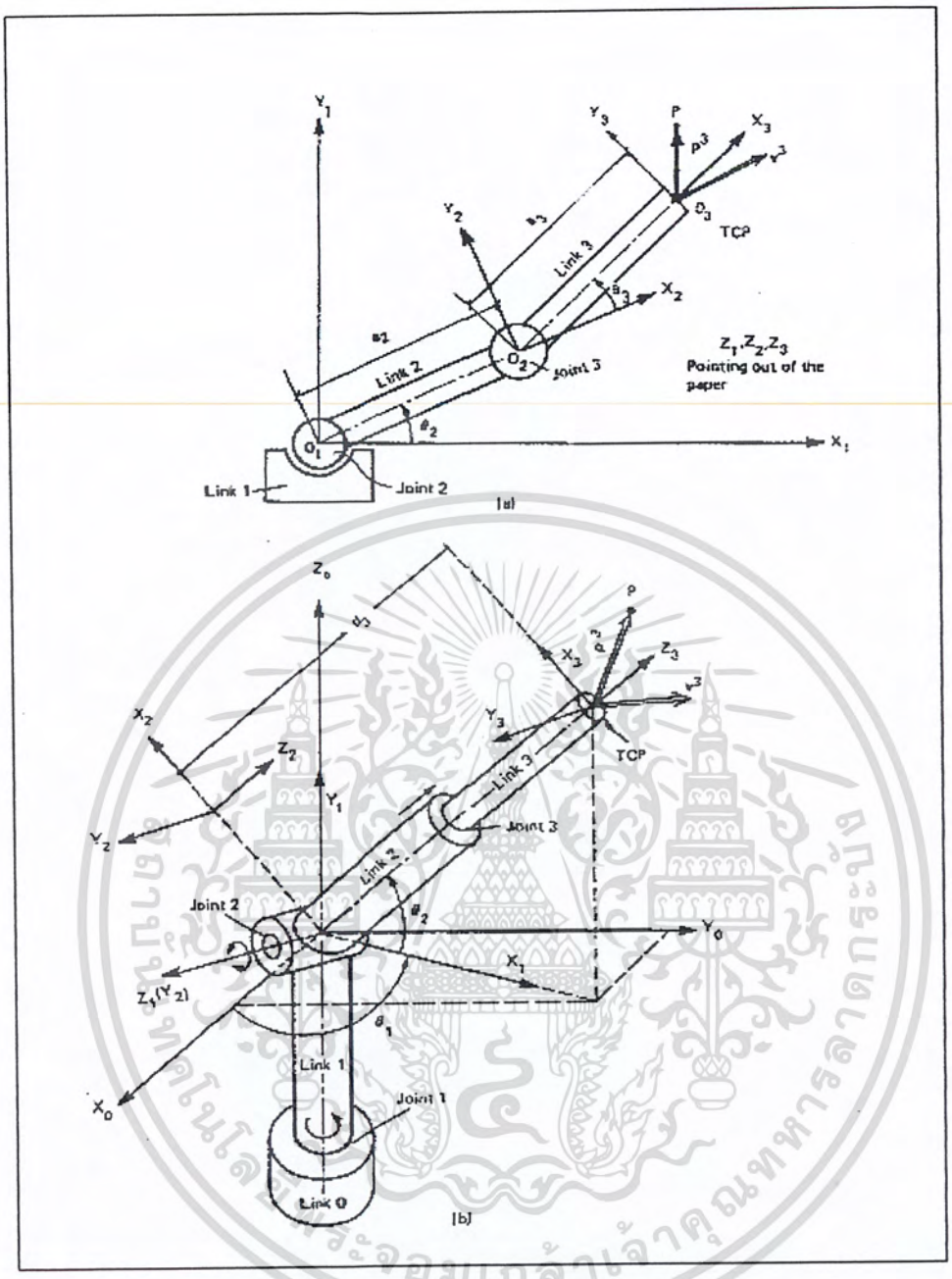
ตัวอย่างอื่นๆของโครงสร้างแบบคิเนเมติกแบบเปิดของอาร์ติคิวเลทที่มี 3 ตัวแปรอิสระ (three-degree-of-freedom articulated open kinematic chain mechanism) สามารถสร้างกลไกได้โดยการย้ายลิงค์ที่ 2 และ 3 ในรูปที่ 2.14 *b* โดยลิงค์ที่ 2 และ 3 จากรูปที่ 2.14 *a* (Prismatic Joint 3 ถูกย้ายโดย Joint ที่มีการหมุน) ตำแหน่งของ TCP ในกรณีนี้คือ

$$X_t = a_2 \cos\theta_2 \cos\theta_1 + a_3 \cos\psi \cos\theta_1$$
$$Y_t = a_2 \cos\theta_2 \sin\theta_1 + a_3 \cos\psi \sin\theta_1 \quad (5)$$
$$Z_t = a_2 \sin\theta_2 + a_3 \sin\psi$$

และความเร็วของ TCP คือ

$$\begin{aligned}\dot{X}_t = &-(a_2 \cos\theta_2 \sin\theta_1 + a_3 \cos\psi \sin\theta_1)\dot{\theta}_1 \\ &-(a_2 \sin\theta_2 \cos\theta_1 + a_3 \sin\psi \cos\theta_1)\dot{\theta}_2 \\ &-(a_3 \sin\psi \cos\theta_1)\dot{\theta}_3 \\ \dot{Y}_t = &-(a_2 \cos\theta_2 \cos\theta_1 + a_3 \cos\psi \cos\theta_1)\dot{\theta}_1 \\ &-(a_2 \sin\theta_2 \sin\theta_1 + a_3 \sin\psi \sin\theta_1)\dot{\theta}_2 \\ &-(a_3 \sin\psi \sin\theta_1)\dot{\theta}_3 \\ \dot{Z}_t = &(a_2 \cos\theta_2 + a_3 \cos\psi)\dot{\theta}_2 + (a_3 \cos\psi)\dot{\theta}_3\end{aligned} \quad (6)$$

ซึ่ง $\psi = \theta_2 + \theta_3$




**รูปที่ 2.14** [3; หน้า 86]

Robot manipulators; the assignment of the links' coordinate system according to the DH method

จากการสังเกตตัวอย่างทั้งสาม สามารถเห็นสิ่งหนึ่งได้ว่ามันไม่ใช่วิธีการที่เป็นระบบ และมีความซับซ้อนทางคณิตศาสตร์ที่ขึ้นกับโครงสร้าง ยิ่งกว่านั้นมันจะยุ่งยากในการแก้มากขึ้นเมื่อจำนวนของตัวแปรอิสระในกลไกเพิ่มขึ้น สำหรับกรณีนี้วิธีการที่ใช้อธิบายในหัวข้อถัดไปจะเป็นที่นิยม โดยมันจะไม่ขึ้นกับความซับซ้อนของกลไกและจำนวนของตัวแปรอิสระ

### 2.2.3 การแปลงพิกัดและเวกเตอร์โดยการใช้เมตริกซ์ [3; หน้า 88]

เทคนิคของการแปลงมีดังนี้

1. Complex numbers เป็นเทคนิคที่ใช้ประโยชน์ในการแก้ปัญหาในกรณีที่เป็น 2 มิติ แต่จะยุ่งยากในกรณีที่เป็น 3 มิติ

2. เมตริกซ์ของการหมุน (Rotation matrices) เป็นเทคนิคที่ใช้สำหรับการแปลงเวกเตอร์ (หัวข้อ 2.2.3.1)

3. The quaternion และเวกเตอร์ของการหมุน (rotation vectors) (หัวข้อ 2.2.6)

4. The homogeneous transformation matrices technique ซึ่งจะอภิปรายในหัวข้อนี้

วิธีการใช้ homogeneous transformation matrices ตั้งโดย เดนาวิท (Denavit) และ ฮาร์เทนเบิร์ก (Hartenberg) (1955) เพื่อใช้กับคิเนเมติกเชน (kinematic chain) วิธีการนี้จะอภิปรายในหัวข้อถัดไป

ขั้นแรกของเทคนิคการแปลงเมตริกซ์เป็นคิเนเมติกเชน จะเป็นการกำหนดระบบพิกัดในการย้ายลิงค์ของ Manupulator ทางขวามือของระบบพิกัด $O_i$ ใช้เป็นเครื่องหมายแทนเฟรม $O_i$ โดยติดอยู่ที่ลิงค์ $i$ กับตำแหน่ง origin ของข้อต่อที่ $i+1$ ตัวเลขของข้อต่อและลิงค์จะเริ่มนับจากที่ฐาน ดังแสดงในรูปที่ 2.14

การแปลงเมตริกซ์ เป็นการกำหนดความสัมพันธ์ทางเรขาคณิตระหว่างสองระบบพิกัด $O_i$ และ $O_{i-1}$ เป็นฟังก์ชันของตัวแปรข้อต่อ $Q_i$ ซึ่ง $Q_i = \theta_i$ สำหรับการหมุนข้อต่อ และ $Q_i = d_i$ สำหรับ Prismatic Joint แต่ละการแปลงเมตริกซ์จะประกอบด้วย เมตริกซ์ที่กำหนดทิศทาง และเวกเตอร์ของการเปลี่ยนตำแหน่ง ซึ่งจะอธิบายต่อไป

#### 2.2.3.1 เมตริกซ์ของทิศทางและเวกเตอร์ของการเปลี่ยนตำแหน่ง [3; หน้า 88]

รูปที่ 2.15 แสดงระนาบในกรณีที่เฟรม $O_i$ หมุนและเคลื่อนที่สัมพันธ์กับ $O_{i-1}$ เวกเตอร์ประกอบ $v^{i-1}$ สามารถแสดงได้โดย เวกเตอร์ประกอบ $v^i$

$$v_x^{i-1} = v_x^i \cos\theta_i - v_y^i \sin\theta_i \tag{7}$$

$$v_y^{i-1} = v_x^i \sin\theta_i + v_y^i \cos\theta_i \tag{8}$$

สมการ (4-7) และ (4-8) สามารถเขียนให้อยู่ในรูปเมตริกซ์ คือ

$$\begin{bmatrix} v_x^{i-1} \\ v_y^{i-1} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} v_x^i \\ v_y^i \end{bmatrix} \tag{9}$$

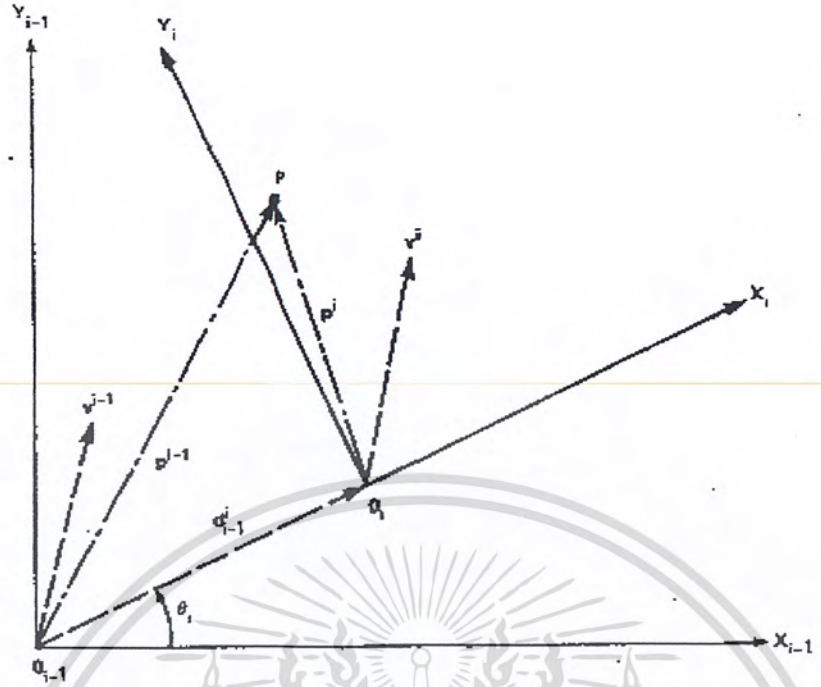
สมการ (4-9) สามารถเขียนให้อยู่ในรูปอื่นได้เป็น

$$v^{i-1} = C_{i-1}^i v^i \tag{10}$$

ซึ่ง

$$C_{i-1}^i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{11}$$

เมตริกซ์ในสมการ (4-11) จะเรียกว่าเมตริกซ์ของทิศทาง และใช้ในการแปลงเวกเตอร์ระหว่างสองลิงค์ที่ต่อกัน ตามสมการ (4-10) เมตริกซ์นี้จะใช้อธิบายทิศทางของเฟรม $O_i$ เทียบกับ $O_{i-1}$ และใช้ในการกำหนดทิศทางของลิงค์มันเรียกได้อีกอย่างว่าเมตริกซ์ของการหมุน หรือ direction cosine matrix (DCM)




**รูปที่ 2.15** การแปลงเวกเตอร์และจุดที่อยู่ระหว่างสองระบบพิกัดที่อยู่ในระนาบ [3; หน้า 89]

อินเวอร์สของการแปลง ของสมการ (10) จะใช้ในการแปลงส่วนประกอบของเวกเตอร์ จากเฟรม $O_{i-1}$ ไป $O_i$ ดังนี้

$$v^i = C_i^{i-1} v^{i-1} \tag{12}$$

ซึ่ง $C_i^{i-1}$ คือ เมตริกซ์ของทิศทางของเฟรม $O_{i-1}$ เทียบกับเฟรม $O_i$ โดย

$$C_i^{i-1} = \left[C_{i-1}^{i}\right]^{-1} = \left[C_{i-1}^{i}\right]^{T} \tag{13}$$

ซึ่ง $^{-1}$ เป็นสัญลักษณ์ของอินเวอร์ส และ $^{T}$ เป็นทรานสโพสของเมตริกซ์ อินเวอร์สของเมตริกซ์ของทิศทางจะเท่ากับทรานสโพส โดยที่เมตริกซ์ของทิศทางต้องมีจำนวนแถวเท่ากับจำนวนหลัก อินเวอร์สของการแปลงในสมการ (9) ตามสมการ (13) คือ

$$\begin{bmatrix} v_x^i \\ v_y^i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} v_x^{i-1} \\ v_y^{i-1} \end{bmatrix} \tag{14}$$

ข้อสังเกต ถ้า Joint $i$ เป็น Prismatic Joint เมตริกซ์ของทิศทางระหว่าง link $i$ และ $i-1$ จะเป็นเมตริกซ์เอกลักษณ์ ซึ่งเครื่องหมายจะไม่เปลี่ยนในการกำหนดทิศทาง

**Chaining Orientation Matrices** เวกเตอร์ $v^3$ ในเฟรม $O_3$ ในรูปที่ 2.14 *a* สามารถนำมาแสดงใน $O_2$ โดยใช้สมการ (9) สำหรับ $i = 3$ หรือในรูปของสมการ (10)

$$v^2 = C_2^3 v^3 \tag{15}$$

เช่นเดียวกัน $v^2$ สามารถแปลงให้อยู่ในเฟรม $O_1$ ดังนี้

$$v^1 = C_1^2 v^2 \tag{16}$$

ที่ซึ่งเมตริกซ์ C ได้มาจากสมการ (11) สมการ (15) และ (16) นำมารวมกันได้เป็น

$$v^1 = C_1^2 C_2^3 v^3 \tag{17}$$

หรือ

$$v^1 = C_1^3 v^3 \tag{18}$$

เมตริกซ์ $C_1^3$ เป็นการแปลงเมตริกซ์ของการหมุนจากเฟรม $O_3$ ไปยัง เฟรม $O_1$ การพิสูจน์จะอยู่บนพื้นฐานของการคูณเมตริกซ์ ดังนี้

$$C_1^3 = C_1^2 C_2^3$$

$$= \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 \\ \sin\theta_3 & \cos\theta_3 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_2\cos_3 - \sin\theta_2\sin\theta_3 & -\sin\theta_2\cos\theta_3 - \cos\theta_2\sin\theta_3 \\ \sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3 & \cos\theta_2\cos_3 - \sin\theta_2\sin\theta_3 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \tag{19}$$

ที่ซึ่ง $\psi = \theta_2 + \theta_3$ โดยดูได้จากรูปที่ 2.14 $a$ เมตริกซ์ของทิศทาง $C_1^3$ ในสมการ (19) ใช้อธิบายทิศทางของเฟรม $O_3$ เทียบกับเฟรม $O_1$

การดำเนินการที่อธิบายไปในสมการ (17) และ (19) ถูกเรียกว่า การดำเนินการลูกโซ่ (chaining operation) ผลลัพธ์จะเป็นเมตริกซ์ทิศทางระหว่าง 2 ลิงค์ ที่ไม่ได้ใช้ joint ที่เหมือนกัน เมตริกซ์ของทิศทางสามารถระบุสองระบบพิกัดใดๆ $O_i$ และ $O_j$ $\forall i > j$ โดยใช้วิธี chaining operation

$$C_j^i = C_j^{j+1} C_{j+1}^{j+2} \cdots C_{i-2}^{i-1} C_{i-1}^i \tag{20}$$

ผลลัพธ์เมตริกซ์ $C_j^i$ จะใช้อธิบายทิศทางของเฟรม $O_i$ (หรือ ลิงค์ $i$) เทียบกับ $O_j$ (หรือ ลิงค์ $j$) โดยทั่วไป การแปลงสามารถเขียนให้อยู่ในรูปของสมการ (10)

$$v^i = C_j^i v^i \tag{21}$$

ซึ่งเป็นการแปลเวกเตอร์จาก $O_i$ ไป $O_j$ วิธีการเดียวกันนี้สามารถนำมาประยุกต์ใช้ในกรณีที่เป็น 3 มิติได้ ซึ่ง $C_j^i$ จะเป็นเมตริกซ์ $3 \times 3$

**เวกเตอร์ของการเปลี่ยนตำแหน่งและเวกเตอร์ของตำแหน่ง** เวกเตอร์ของการเปลี่ยนตำแหน่ง $d_{i-1}^i$ (ดูรูปที่ 2.15) สามารถใช้อธิบายตำแหน่งของ origin ของเฟรม $O_i$ ใน $O_{i-1}$ และได้มาจาก

$$d_{i-1}^i = \begin{bmatrix} a_i \cos\theta_i \\ a_i \sin\theta_i \end{bmatrix} \tag{22}$$

ซึ่ง $a_i$ เป็นการกระจัดระหว่าง $O_i$ และ $O_{i-1}$

จุด P ซึ่งถูกระบุใน $O_i$ โดย เวกเตอร์ตำแหน่ง $p^i$ สามารถแสดงใน $O_{i-1}$ โดยเวกเตอร์

$$p^{i-1} = d_{i-1}^i + C_{i-1}^i p^i \tag{23}$$

ข้อสังเกต $p^i$ ถูกแสดงใน $O_{i-1}$ โดยใช้สมการ (10) และการบวกเวกเตอร์ซึ่งกระทำบนเฟรม $O_{i-1}$

**2.2.3.2 Homogeneous Transformation Matrices [3; หน้า 91]**

Homogeneous จะแสดงอยู่ในรูปของเวกเตอร์ 2 มิติ ซึ่งถูกรวมเป็นเมตริกซ์ที่มี 3 ส่วนประกอบ

$$v = \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} \tag{24}$$

สัญลักษณ์ของเวกเตอร์จากนี้ถึงหัวข้อ 2.2.5 จะสัมพันธ์กับการแสดงเป็น homogeneous การแสดง Homogeneous ของการแปลงเมตริกซ์ที่มี 2 มิติจะอยู่ในรูปของเมตริกซ์ $3\times3$ ชนิดเมตริกซ์ที่ใช้มี 3 ชนิดดังนี้

1. $T_{Ri-1}^{\ i}$ = Homogeneous rotation matrix ซึ่งจะแสดงการหมุนของเฟรม $O_i$ เทียบกับ $O_{i-1}$ และใช้แปลงเวกเตอร์ v จาก $O_i$ ไป $O_{i-1}$

2. $T_{Ti-1}^{\ i}$ = Homogeneous translation matrix ซึ่งจะแสดงการเคลื่อนตำแหน่ง (แต่ไม่หมุน) ของเฟรม $O_i$ เทียบกับ $O_{i-1}$ และใช้แปลงเวกเตอร์ของตำแหน่ง p จาก $O_i$ ไปยัง $O_{i-1}$

3. $T_{i-1}^{\ i}$ = Homogeneous displacement matrix ซึ่งจะแสดงการเคลื่อนตำแหน่งและการหมุนของเฟรม $O_i$ เทียบกับ $O_{i-1}$ และใช้แปลงเวกเตอร์ของตำแหน่ง p จาก $O_i$ ไปยัง $O_{i-1}$ เมตริกซ์ T เป็น เมตริกซ์เดนาวิท – ฮาร์เทนเบิร์ก (Denavit-Hartenberg; DH)

การดำเนินการของเมตริกซ์สุดท้ายจะเหมือนกันกับการรวมของการดำเนินการของสองเมตริกซ์แรก ซึ่งหมายถึง เวกเตอร์ของตำแหน่งสามารถจะหมุนก่อนโดยใช้ $T_R$ และต่อมาจึงเปลี่ยนตำแหน่งโดยใช้ $T_T$ หรือสามารถแปลงได้โดยตรงโดยใช้ เมตริกซ์ของการเคลื่อนที่ T

**The Homogeneous Rotation Matrix** สองมิติของ homogeneous rotation matrix ระหว่างเฟรม $O_i$ (ซึ่งหมุนเทียบกับ $O_{i-1}$) และเฟรม $O_{i-1}$ คือ

$$T_{Ri-1}^{\ i} = \left[\begin{array}{cc:c} & & 0 \\ & C_{i-1}^{i} & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{25}$$

ซึ่ง $C_{i-1}^{i}$ เป็น $2\times2$ ได้มาจากสมการ (11) เวกเตอร์ $v^i$ (ดูรูปที่ 2.15) จะถูกแปลงเป็น $v^{i-1}$ ด้วย homogeneous rotation matrix ดังนี้

$$\begin{bmatrix} v_x^{i-1} \\ v_y^{i-1} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{i} \\ v_y^{i} \\ 1 \end{bmatrix} \tag{26}$$

หรือเหมือนกันกับสมการ (10)

$$v^{i-1} = T_{Ri-1}^{\ i} v^i \tag{27}$$

**Homogeneous Translation Matrix** เป็น $3\times3$ two-dimensional homogeneous translation matrix ระหว่างเฟรม $O_i$ (ซึ่งเคลื่อนตำแหน่งขนานกับ $O_{i-1}$) และเฟรม $O_{i-1}$ คือ

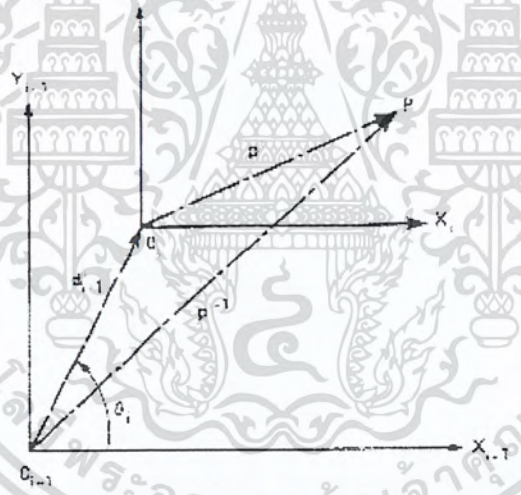
$$T_{T\,i-1}^{\ i} = \left[\begin{array}{cc|c} 1 & 0 & d_{i-1}^{i} \\ 0 & 1 & \\ \hline 0 & 0 & 1 \end{array}\right] \tag{28}$$

ซึ่ง $d_{i-1}^{i}$ คือ เวกเตอร์ของการเปลี่ยนตำแหน่งได้มาจากสมการ (22)

ตัวอย่าง สมมติเฟรม $O_i$ เคลื่อนตำแหน่งขนานกับเฟรม $O_{i-1}$ มีขนาดเท่ากับ $a_i$ ดังในรูปที่ 2.16 origin ของ $O_i$ ถูกอธิบายในเฟรม $O_{i-1}$ โดย $d_{i-1}^{i}$ ที่ได้มาจากสมการ (22) จุด P ถูกอธิบายในเฟรม $O_i$ และ $O_{i-1}$ โดยเวกเตอร์ของตำแหน่ง $\mathbf{p}^{i}$ และ $\mathbf{p}^{i-1}$ ตามลำดับ เวกเตอร์ $\mathbf{p}^{i}$ ถูกแปลงเป็น $\mathbf{p}^{i-1}$ด้วย homogeneous translation matrix ในสมการ (28) ดังนี้

$$\begin{bmatrix} p_x^{i-1} \\ p_y^{i-1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_i\cos\theta_i \\ 0 & 1 & a_i\sin\theta_i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{i} \\ p_y^{i} \\ 1 \end{bmatrix} = \begin{bmatrix} p_x^{i} + a_i\cos\theta_i \\ p_y^{i} + a_i\sin\theta_i \\ 1 \end{bmatrix} \tag{29}$$

ข้อสังเกต การแปลงนี้เป็นการรวมเวกเตอร์ $d_{i-1}^{i}$ และ $p^{i}$ ความสามารถของการรวมเวกเตอร์ในรูปเมตริกซ์นี้เป็นผลมาจากส่วนประกอบ 1 ซึ่งถูกรวมอยู่ในการแสดง homogeneous ซึ่งเป็นลักษณะของ homogeneous transformations ที่แตกต่างจากวิธีการอื่น



**รูปที่ 2.16** การแปลงพิกัดระหว่างระนาบของสองระบบพิกัด ซึ่งเคลื่อนที่สัมพันธ์กับอีกระนาบหนึ่ง [3; หน้า 93]

**The Homogeneous Displacement Matrix** $T_{i-1}^{i}$ ใช้แปลงจุดตำแหน่งพิกัดที่อยู่ใน $O_i$ ไป $O_{i-1}$ อ้างอิงกับ รูปที่ 2.16 เวกเตอร์ของตำแหน่ง $\mathbf{p}^{i-1}$ สามารถแสดงในเทอม $\mathbf{p}^{i}$ โดยการหมุน $\mathbf{p}^{i}$ ก่อน ดังนั้นจะอยู่ในระบบพิกัดที่ขนานกับเฟรม $O_{i-1}$ และเคลื่อนตำแหน่ง $\mathbf{p}^{i}$ที่ถูกหมุนโดยรวมตัวมันกับ $d_{i-1}^{i}$ ใช้ homogeneous transformation matrices กระบวนการนี้จะดำเนินการตามนี้

$$p^{i-1} = T_{T\,i-1}^{\ i}\, T_{R\,i-1}^{\ i}\, p^{i} \tag{30}$$

หรือ

$$p^{i-1} = T_{i-1}^{i}\, p^{i} \tag{31}$$

$T_{i-1}^{i}$ เป็น homogeneous displacement matrix ระหว่าง $O_i$ และ $O_{i-1}$ ได้มาจาก

$$T_{i-1}^{i} = T_{T\,i-1}^{\ i}\, T_{R\,i-1}^{\ i} \tag{32}$$

ข้อสังเกต สมการ (30) มีวิธีการแปลงที่เหมือนกันกับสมการ (4-23) ซึ่งเกี่ยวกันเฉพาะ การดำเนินการเมตริกซ์ การคูณเมตริกซ์ในสมการ (32) สำหรับ $T_T$ และ $T_R$ ได้มาจากสมการ (28) และ (25) ตามลำดับ ซึ่งได้ผลดังนี้

$$T_{i-1}^{i} = \left[\begin{array}{cc:c} & & \\ & C_{i-1}^{i} & d_{i-1}^{i} \\ & & \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{33}$$

ซึ่งเป็นรูปมาตรฐานของเมตริกซ์ของการเปลี่ยนตำแหน่ง เมตริกซ์ $C_{i-1}^{i}$ และเวกเตอร์ $d_{i-1}^{i}$ ใช้บอกทิศทางและตำแหน่งของ $O_i$ เทียบกับ $O_{i-1}$ ตามลำดับ

สำหรับสองระบบพิกัด ในรูปที่ 2.15 เมตริกซ์ $C_{i-1}^{i}$ ในสมการ (11) และเวกเตอร์ $d_{i-1}^{i}$ ในสมการ (22) นำมาแทนค่าลงในสมการ (33) ได้ผลคือ

$$T_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i & a_i\sin\theta_i \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

สำหรับสองระบบพิกัด $O_i$ และ $O_j$ ใดๆ เมตริกซ์ของการเปลี่ยนตำแหน่งคือ

$$T_{j}^{i} = \left[\begin{array}{cc:c} & & \\ & C_{j}^{i} & d_{j}^{i} \\ & & \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{35}$$

ซึ่ง $C_j^i$ ใช้บอกทิศทางของ $O_i$ ในเฟรม $O_j$ และ $d_j^i$ เป็น เวกเตอร์ของตำแหน่งของ origin ของ $O_i$ ใน $O_j$

การแปลงของ เวกเตอร์ของตำแหน่ง $\mathbf{p}^i$ ใน $O_i$ เป็น $\mathbf{p}^j$ ในเฟรม $O_j$ มีวิธีการดังนี้

$$p^j = T_j^i p^i \tag{36}$$

และ

$$T_j^i = T_{T\,j}^{\ i}\, T_{R\,j}^{\ i} \tag{37}$$

**Inverse Homogeneous Transformation** อินเวอร์สการแปลงของสมการ (4-36) มีวิธีการตามนี้

$$p^i = T_i^j p^j \tag{38}$$

เมื่อ

$$T_i^j = [T_j^i]^{-1} \tag{39}$$

อินเวอร์สของเมตริกซ์ของการกระจัดก็จะเหมือนกัน การทำอินเวอร์สของ $T_j^i$ ให้ง่าย สามารถทำได้จากการประยุกต์ การเคลื่อนที่และการหมุนเมตริกซ์

คุณสมบัติของอินเวอร์สเมตริกซ์

$$[\mathbf{A}\ \mathbf{B}]^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \tag{40}$$

จากสมการ (37) ได้ผลดังนี้

$$[T_j^i]^{-1} = [T_{Rj}^{\ i}]^{-1}[T_{Tj}^{\ i}]^{-1} \tag{41}$$

มันสามารถแสดงเป็นอินเวอร์สของ homogeneous rotation และ translation matrices คือ

$$[T_{Rj}^{\ i}]^{-1} = \left[\begin{array}{cc:c} & & 0 \\ & [C_j^i]^T & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{42}$$

และ

$$[T_{Rj}^{\ i}]^{-1} = \left[\begin{array}{cc:c} 1 & 0 & -d_j^i \\ 0 & 1 & \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{43}$$

จากรูปที่ 2.15 อินเวอร์สของการแปลงที่แสดงในสมการ (42) และ (43) จะได้มาจาก (สำหรับ $j = i-1$)

$$[T_{Ri-1}^{\ i}]^{-1} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{44}$$

$$[T_{Ti-1}^{\ i}]^{-1} = \begin{bmatrix} 1 & 0 & -a_i\cos\theta_i \\ 0 & 1 & -a_i\sin\theta_i \\ 0 & 0 & 1 \end{bmatrix} \tag{45}$$

แทนสมการ (44) และ (45) ลงในสมการ (41) สำหรับ $j = i-1$ ได้เป็นดังนี้

$$[T_{i-1}^{i}]^{-1} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & -a_i\cos\theta_i \\ -\sin\theta_i & \cos\theta_i & -a_i\sin\theta_i \\ 0 & 0 & 1 \end{bmatrix} \tag{46}$$

จากสมการ (35)

$$T_i^{i-1} = \left[\begin{array}{cc:c} & & d_i^{i-1} \\ & C_i^{i-1} & \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{47}$$

สมการ (46) และ (47) จะเหมือนกัน เมื่อ

$$d_i^{i-1} = \begin{bmatrix} -a_i \\ 0 \end{bmatrix} \qquad \text{และ} \qquad C_i^{i-1} = [C_{i-1}^i]^T \tag{48}$$

สำหรับ $d_i^{i-1}$ ใช้บอกตำแหน่ง origin ของ $O_{i-1}$ ในเฟรม $O_i$ และ $C_i^{i-1}$ ใช้บอกทิศทางของ $O_{i-1}$ ในเฟรม $O_i$ [ดูสมการ (13)]

**Chaining Displacement Matrices** การดำเนินการลูกโซ่ของ homogeneous displacement matrices จะเหมือนกันกับการดำเนินการลูกโซ่ของเมตริกซ์ของทิศทางสำหรับสองการแปลง:

$$p^j = T_j^i p^i \tag{49}$$

และ

$$p^i = T_i^k p^k \tag{50}$$

การดำเนินการลูกโซ่เป็นวิธีการ โดยการแทน $p^i$ จากสมการ (50) ลงในสมการ (49) ซึ่งผลที่ได้ คือ

$$p^j = T_j^i T_i^k p^k \tag{51}$$

ซึ่ง

$$T_j^k = T_j^i T_i^k \tag{52}$$

หรือในรูปทั่วไป เมื่อ $i > j$

$$T_j^i = T_j^{j+1} T_{j+1}^{j+2} \cdots T_{i-2}^{i-1} T_{i-1}^i \tag{53}$$

เมื่อกลับไปดูสามระบบพิกัดฉากรูปที่ 2.16a สองเมตริกซ์ของการกระจัด $T_1^2$ และ $T_2^3$ อยู่ในรูปของสมการ (34) เมื่อ $i = 2$ และ $i = 3$ ตามลำดับ การแปลงเวกเตอร์ของตำแหน่งของจุด $P$ จาก $O_3$ ไป $O_1$ ใช้การดำเนินการลูกโซ่ตามสมการ (51) ได้ผลเป็น

$$p^1 = T_1^3 p^3 \tag{54}$$

ซึ่ง

$$T_1^3 = T_1^2 T_2^3 = \left[\begin{array}{cc:c} \cos\psi & -\sin\psi & a_3\cos\psi + a_2\cos\theta_2 \\ \sin\psi & \cos\psi & a_3\sin\psi + a_2\sin\theta_2 \\ \hdashline 0 & 0 & 1 \end{array}\right] \tag{55}$$

และ $\psi = \theta_2 + \theta_3$ [ดูสมการ (19)]

TCP จะอยู่ที่ origin ของเฟรม $O_3$ และมีตำแหน่งสัมพันธ์กับ $O_1$ (ระบบพิกัดฉากอ้างอิง) แสดงโดยเวกเตอร์ของการเคลื่อนตำแหน่ง

$$\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = d_1^3 = \begin{bmatrix} a_2\cos\theta_2 + a_3\cos\psi \\ a_2\sin\theta_2 + a_3\sin\psi \end{bmatrix} \tag{56}$$

ซึ่งได้มาจาก $T_1^3$ ในสมการ (55) สมการ (56) จะเหมือนกันกับสมการ (1) ซึ่งได้มาจากการพิจารณาทางเรขาคณิตอย่างง่ายๆเมตริกซ์ของทิศทาง $C_1^3$ เหมือนกันกับสมการ (19) และใช้บอกทิศทางของ link 3 เทียบกับเฟรม $O_1$

#### 2.2.3.3 Three-Dimensional Homogeneous Transformations [3; หน้า 97]

การแสดง Homogeneous ของเวกเตอร์ 3 มิติจะมี 4 component

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix} \tag{57}$$

การแสดง Homogeneous ของการแปลงเมตริกซ์ 3 มิติประกอบด้วย homogeneous rotation, translation และเมตริกของการกระจัดซึ่งอยู่ในรูปเมตริกซ์ $4\times4$

**เมตริกซ์ของการหมุน (Rotation Matrix)** เป็นการแปลงเมตริกซ์ของการหมุนที่อยู่ระหว่างสองระบบพิกัดฉาก $O_i$ และ $O_{i-1}$ คือ

$$T_{Ri-1}^{\ i}=\left[\begin{array}{ccc|c} & C_{i-1}^{i} & & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{58}$$

ซึ่ง $O_i$ หมุนสัมพันธ์กับเฟรม $O_{i-1}$

นั่นคือสามเมตริกซ์ของการหมุนสำหรับอธิบายการหมุนของ $O_i$ ตามแกน $X_{i-1}, Y_{i-1}$ และ $Z_{i-1}$ สำหรับการหมุน $O_i$ เป็นมุม $\alpha_i$ รอบแกน $X_{i-1}$ เมตริกซ์ของการแปลงที่ใช้คือ

$$[T_{Ri-1}^{\ i}]_x=\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{59}$$

ในทางเดียวกัน หมุน $O_i$ เป็นมุม $\beta_i$ รอบแกน $Y_{i-1}$

$$[T_{Ri-1}^{\ i}]_y=\left[\begin{array}{ccc|c} \cos\beta_i & 0 & \sin\beta_i & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta_i & 0 & \cos\beta_i & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{60}$$

และ หมุน $O_i$ เป็นมุม $\theta_i$ รอบแกน $Z_{i-1}$

$$[T_{Ri-1}^{\ i}]_z=\left[\begin{array}{ccc|c} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{61}$$

**เมตริกซ์ของการเปลี่ยนตำแหน่ง (Translation Matrix)** homogeneous translation matrix สำหรับกรณีที่เป็น 3 มิติคือ

$$T_{Ti-1}^{\ i}=\left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & 1 & 0 & d_{i-1}^{i} \\ 0 & 0 & 1 & \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{62}$$

ซึ่ง $d_{i-1}^{i}$ ใช้บอก origin ของ $O_i$ ในเฟรม $O_{i-1}$ สมการ (62) จะเท่ากับสมการ (28) ในกรณีที่เป็นระนาบ

**เมตริกซ์ของการกระจัด (Displacement Matrix)** homogeneous displacement matrixได้มาโดย การหมุนและการเคลื่อนตำแหน่งของ $O_i$ เทียบกับเฟรม $O_{i-1}$ :

$$T_{i-1}^{i} = T_{T\,i-1}^{\ \ i}\, T_{R\,i-1}^{\ \ i} \tag{63}$$

แทนสมการ (58) และ (62) ลงใน (63) ผลที่ได้คือ

$$T_{i-1}^{i} = \left[\begin{array}{ccc:c} & C_{i-1}^{i} & & d_{i-1}^{i} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{64}$$

สำหรับสองระบบพิกัดฉาก $O_i$ และ $O_j$ ใดๆ เมตริกซ์ของการกระจัดคือ $T_j^i$ ซึ่ง $j = i-1$ ในสมการ (64)

**การแปลงเวกเตอร์ (Vector Transformation)** การแปลงเวกเตอร์ระหว่างสองระบบพิกัดฉาก $O_i$ และ $O_{i-1}$ จะเหมือนกันกับสมการ (27) ในกรณีที่เป็นระนาบ

$$v^{i-1} = T_{R\,i-1}^{\ \ i}\, v^{i} \tag{65}$$

ซึ่ง $v^i$ (แสดงอยู่ใน $O_i$) จะถูกแปลงเป็น $v^{i-1}$ (แสดงอยู่ใน $O_{i-1}$) และ $T_{R\,i-1}^{\ \ i}$ ได้มาจากสมการ (58)

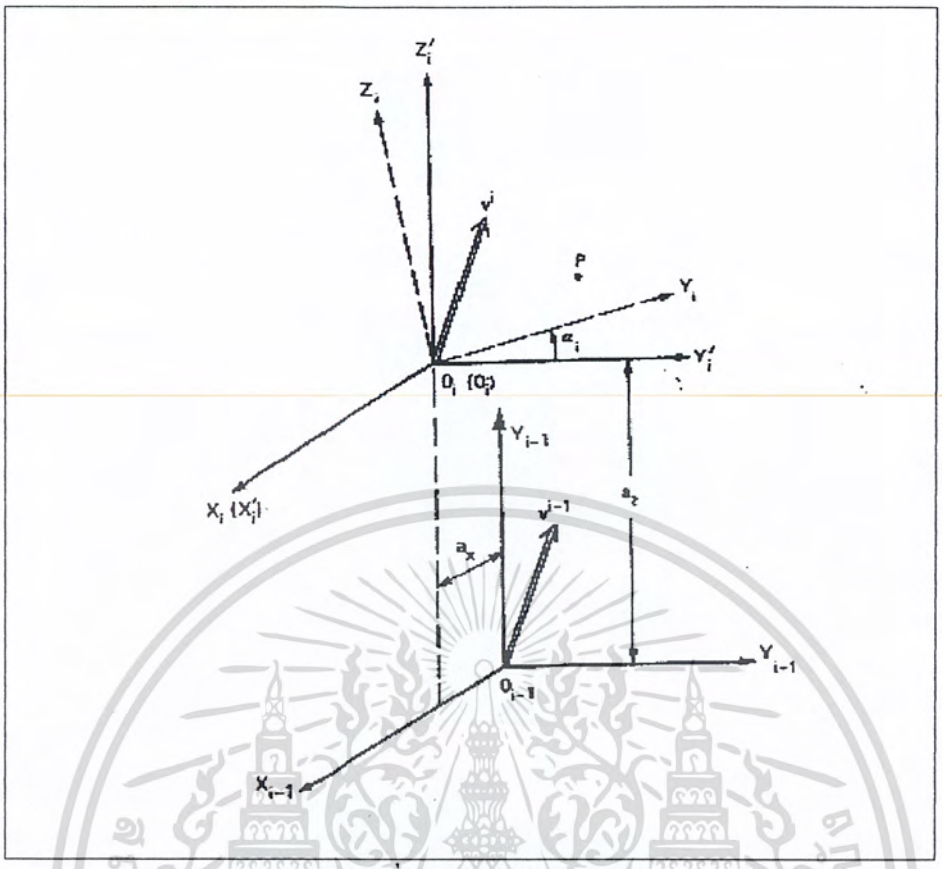
ตัวอย่าง พิจารณาสามระบบพิกัดฉาก $O_{i-1}, O_i'$ และ $O_i$ ใช้อธิบายในรูปที่ 2.17 เฟรม $O_i'$ เคลื่อนที่สัมพันธ์และขนานกับ $O_{i-1}$ เฟรม $O_i$ มาจากการหมุนรอบแกน $X_i'$ เป็นมุม $\alpha_i$ เวกเตอร์ v สามารถอธิบายได้โดย $v^i$ หรือ $v^{i-1}$ การแปลงเมตริกซ์ของการหมุนระหว่าง $O_i$ และ $O_{i-1}$ ได้มาจากสมการ (59) และใช้แปลง $v^i$ เป็น $v^{i-1}$ ตามสมการ (65)

$$\begin{bmatrix} v_x^{i-1} \\ v_y^{i-1} \\ v_z^{i-1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{i} \\ v_y^{i} \\ v_z^{i} \\ 1 \end{bmatrix} \tag{66}$$

เมื่อ $\alpha_i = \pi/2$ สมการ (66) จะกลายเป็น

$$\begin{bmatrix} v_x^{i-1} \\ v_y^{i-1} \\ v_z^{i-1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x^{i} \\ v_y^{i} \\ v_z^{i} \\ 1 \end{bmatrix} = \begin{bmatrix} v_x^{i} \\ -v_z^{i} \\ v_y^{i} \\ 1 \end{bmatrix} \tag{67}$$

ซึ่งจะหมายถึงกรณีที่โปรเจคชั่นของ **v** บน $Y_i$ (ได้แก่ $v_y^i$) มองเห็นในเฟรม $O_{i-1}$ บนแกน $Z_{i-1}$ และ $v_z^i$ มองเห็นใน $O_{i-1}$ บนแกน $-Y_{i-1}$




**รูปที่ 2.17** [3; หน้า99]

Vector and Coordinate transformations between two links' coordinate systems that are translated relative to each other

**การเปลี่ยนตำแหน่งของพิกัด (Point Coordinates Transformation)** จุดที่ถูกบ่งบอกโดยเวกเตอร์บอกตำแหน่ง p และถูกแปลงจาก $O_i$ ไป $O_{i-1}$ โดยใช้การแปลงเมตริกซ์ของการกระจัด

$$p^{i-1} = T_{i-1}^{i} p^{i} \tag{68}$$

ซึ่ง $T_{i-1}^{i}$ มาจากสมการ (64) สมการ (68) จะเท่ากันกับสมการ (4-31) เมื่อเป็นกรณีที่เป็นระนาบ การแปลงที่มาจากสมการ (68) จะเรียกว่า การแปลงพิกัดฉาก (coordinate transformation)

สำหรับตัวอย่าง สมมติ จุด $P$ในรูปที่ 2.17 มาจากเฟรม $O_i$ และจะแสดงให้อยู่ในเฟรม $O_{i-1}$ การแปลงเป็นวิธีการที่มี 2ขั้นตอนตามสมการ (63) ขั้นแรก $p^i$ จะหมุนไปยังเฟรม $O_i'$ โดยใช้เมตริกซของการหมุนในสมการ (59) และจะเปลี่ยนเป็น $p^{i'}$ โดยการเคลื่อนที่ ซึ่งใช้เมตริกซ์ของการเคลื่อนที่ การแปลงเมตริกซ์ที่ใช้ในการคำนวณเป็นดังนี้

$$T_{i-1}^{i} = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{69}$$

และผลลัพธ์ในสมการ (68) สำหรับตัวอย่างนี้ คือ

$$p^{i-1} = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p^i \qquad (70)$$

inverse homogeneous transformations และ การดำเนินการลูกโซ่ที่แสดงสำหรับกรณีของระนาบสามารถใช้ได้กับโครงสร้าง 3 มิติ การดำเนินการลูกโซ่เป็นวิธีการโดยการคูณเมตริกซ์ช่วยตามสมการ (35) ได้ผลเป็น

$$T_j^i = \left[\begin{array}{ccc:c} & C_j^i & & d_j^i \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \qquad (71)$$

**สรุป** The homogeneous transformation ระหว่างสองระบบพิกัดฉากที่ต่อเนื่องกัน ($O_i$ และ $O_{i-1}$) แสดงในรูปของเมตริกซ์ $T_{i-1}^i$ ซึ่งเรียกว่า homogeneous displacement matrix หรือเป็น displacement matrix ในการกำหนดเมตริกซ์ของการกระจัด สำหรับสองสองระบบพิกัดฉากที่ต่อเนื่องกัน ($O_{i-1}$ และ $O_i$) สามารถหาได้จากการพิจารณาทางเรขาคณิต ดังแสดงในรูปที่ 2.15

เมตริกซ์ $T_{i-1}^i$ ประกอบด้วยเวกเตอร์ของการเคลื่อนที่ $d_{i-1}^i$ และเมตริกซ์ของทิศทาง $C_{i-1}^i$ ทั้งสองอย่างนี้ใช้ในการแปลความหมายทางเรขาคณิต ส่วนประกอบของเวกเตอร์ของการเคลื่อนที่ $d_{i-1}^i$ ใช้แสดงโปรเจ็คชั่นของ origin ของเฟรม $O_i$ บน $O_{i-1}$ เมตริกซ์ของทิศทาง(บางครั้งเรียกว่าเมตริกซ์ของการหมุน) $C_{i-1}^i$ จะใช้อธิบายทิศทางของ $O_i$ เทียบกับเฟรม $O_{i-1}$ และสามารถใช้แปลงเวกเตอร์ระหว่างสองระบบพิกัดฉาก เวกเตอร์ **v** สามารถนำมาแสดงโดยใช้ส่วนของเส้นตรง และนำมาใช้กับ $v^{i-1}$ ในเฟรม $O_{i-1}$ หรือ $v^i$ ในเฟรม $O_i$ ทั้งสองเวกเตอร์นี้จะมีขนาดที่เท่ากันและมีทิศทางสัมบูรณ์ที่เหมือนกันเมตริกซ์ของทิศทางจะใช้แปลงเวกเตอร์ $v^i$ เป็น $v^{i-1}$ เมตริกซ์ของการกระจัด $T_{i-1}^i$ จะใช้สำหรับสิ่งดังต่อไปนี้

- แปลงจุดโคออร์ดิเนตจากเฟรม $O_i$ เป็น $O_{i-1}$ โดยใช้สมการ (36) หรือ (68) หรือแปลงจากเฟรม $O_j$ เป็น $O_i$ โดยใช้สมการ (38) ตัวอย่างได้แก่ เมื่อตำแหน่งของวัตถุถูกตรวจโดยเซนเซอร์ที่ติดอยู่ใน end effector และมันจำเป็นที่จะต้องแสดงตำแหน่งของมันใน WCS
- การแสดงเวกเตอร์ **v** ที่มาจากเฟรม $O_i$ ในเฟรม $O_j$ โดยใช้การแปลงของการหมุนในสมการ (27) หรือสมการ (65) ตัวอย่าง เมื่อความเร็วที่ต้องการของ TCP กำหนดใน TCS และถูกแปลงเป็น WCS เหตุการณ์นี้เกิดขึ้นเมื่อเซนเซอร์ ทำการวัดอยู่ใน TCS หรือเมื่อสอนหุ่นยนต์ใน TCS
- การแสดงตำแหน่งของ origin ของ $O_i$ ใน $O_j$ ซึ่งแสดงโดย $d_j^i$

กำหนดทิศทางของลิงค์ ซึ่ง เฟรม $O_i$ ถูกติดสัมพันธ์กับเฟรม $O_j$ การกำหนดได้มาโดยเมตริกซ์ $C_j^i$

สองวิธีการที่แตกต่างที่ใช้กับคิเนเมติกส์ของแขน มีดังนี้ เทคนิคทางเรขาคณิตทางตรง เป็นเทคนิคอย่างง่ายสำหรับกลไกที่มีจำนวนของแกนน้อย โดยเฉพาะตำแหน่งของ TCP ที่เกี่ยวข้อง สมการคณิตศาสตร์สำหรับเทคนิคนี้ใช้ได้กับแต่ละกลไก เพื่อให้เห็นโดยชัดเจน homogenous transformation เป็นวิธีการทั่วไปซึ่งซับซ้อนเกินไปสำหรับการแก้ปัญหาทำไม่ขึ้นกับจำนวนของแกน ขั้นตอนหลักๆประกอบด้วยการคูณเมตริกซ์ ดังนั้นจึง

เป็นวิธีการง่ายๆ ในการแก้ปัญหาจะมี การเพิ่มตำแหน่งของ TCP และการกำหนดทิศทางของเครื่องมือและตำแหน่งของจุดใดๆบนลิงค์ ลักษณะนี้เป็นลักษณะสำคัญเฉพาะเมื่อแขนมีไดนามิกส์เข้ามาเกี่ยวข้อง ดังนั้นวิธีการนี้บางครั้งจะมีประโยชน์สำหรับระบบกลไกที่มีจำนวนแกนน้อย

### 2.2.4 ข้อกำหนดของเดนาวิท-ฮาร์เทนเบิร์ก (DENAVIT-HARTENBERG COVENTION) [3; หน้า 101]

The homogeneous displacement matrix $T_{i-1}^{i}$ ที่อยู่ในหัวข้อ 2.2.3 จะได้มาจากแต่กรณีของสมการ (59) ถึง (62)

#### 2.2.4.1 เครื่องมือของข้อกำหนด DH (Implementing the DH Convention) [3; หน้า 102]

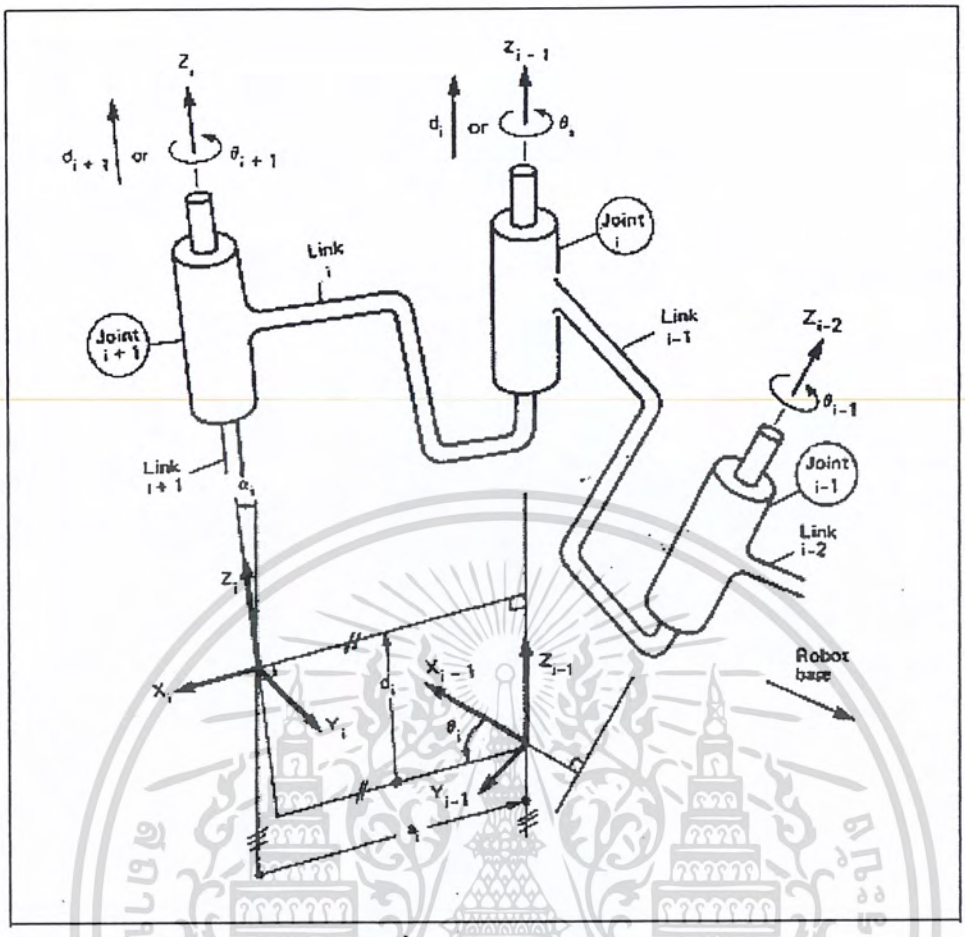
ข้อกำหนด DH เป็นเครื่องมือหลักในแขนหุ่นยนต์ (robot manipulators) ซึ่งประกอบด้วย open kinematic chain ที่มีอยู่ในแต่ละข้อต่อหนึ่งตัวแปรอิสระ และข้อต่อแต่ละอันเป็น revolute หรือ prismatic

Revolute และ prismatic joints จะพิจารณาเป็น lower pairs ได้แก่ ข้อต่อที่มีสองพื้นผิวที่เลื่อนบนอีกอันหนึ่งได้ ซึ่งจะอยู่ติดสัมผัสกัน ชนิด lower-pair จะอยู่ 6 ชนิดที่เป็นไปได้คือ revolute, prismatic, cylindrical, spherical (ball และ socket), screw, และ planar pair ซึ่งจะมีเฉพาะ revolute และ prismatic จะเป็นใช้กันทั่วไปใน robot manipulators

ข้อกำหนด DH เป็นเครื่องมือที่มีขั้นตอนดังนี้

1. Number the links and joint เริ่มต้นที่ฐาน ฐานของหุ่นยนต์จะแทนเครื่องหมายเป็นเหมือนกับ link 0 และ end effector จะเป็น link n ดังแสดงในรูปที่ 2.14*b* link $i$ เคลื่อนที่โดยเทียบกับ link $i-1$ รอบ (สำหรับ revolute) หรือตาม (สำหรับ prismatic) ข้อต่อ $i$ (ดูรูปที่ 2.18)
2. Establish links' coordinate systems สำหรับแต่ละข้อต่อที่เป็นไปตามกฎต่อไปนี้ (ดูรูปที่ 2.14 และ 2.18)
   - *a.* แกน $Z_{i-1}$ ที่เลือกจากแกนของการเคลื่อนที่ของ joint $i$ สำหรับ revolute joint, link $i$ จะหมุนเทียบกับ link $i-1$ รอบแกน $+Z_{i-1}$ เท่ากับ $+\theta_i$ สำหรับ prismatic joint, link $i$ จะเคลื่อนตำแหน่งสัมพันธ์กับ link $i-1$ ตามแนวแกน $+Z_{i-1}$ ด้วยขนาด $+d_i$
   - *b.* แกน $X_i$ ที่เลือกจะตั้งฉากกับแกน $Z_{i-1}$ (ได้แก่ มันจะตั้งฉากทั้ง $Z_{i-1}$ และ $Z_i$) ถ้า $Z_i$ และ $Z_{i-1}$ ไม่ตัดกัน แกน $X_i$ จะเป็นแกนสามัญของ $Z_i$ และ $Z_{i-1}$ และทิศทางของมันจะกำหนดได้จากแกน $Z_{i-1}$ ไปยังแกน $Z_i$ ถ้าแกน $Z_{i-1}$ และ $Z_i$ ตัดกันทิศทางของแกน $X_i$ จะกำหนดไม่ได้ และมันสามารถเลือกได้จากสองทิศทางที่เป็นไปได้ และถ้า $Z_{i-1}$ และ $Z_i$ เป็นเส้นเดียวกัน แกน $X_i$ จะสามารถเลือกได้จากตำแหน่งใดๆในระนาบที่ตั้งฉากกับมัน
   - *c.* แกน $Y_i$ จะเลือกได้จากการใช้กฎมือขวา

ข้อสังเกต การเปลี่ยนระบบพิกัดฉากจะไม่มีลักษณะเฉพาะ




**รูปที่ 2.18** [3; หน้า 103]

Links' Coordinate systems and joint parameters according to Denavit and Hartenberg method

3. Define the joint parameters ซึ่งเป็นค่าทางเรขาคณิต 4 ตัว $\theta_i, d_i, a_i \alpha_i$ (ดูรูปที่ 2.18)

   $\theta_i$ เป็นมุมระหว่างแกน $X_{i-1}$ และ $X_i$ ซึ่งมาจากการหมุน $X_{i-1}$ เป็น $X_i$ รอบแกน $Z_{i-1}$ สำหรับ revolute joint $\theta_i$ เป็นตัวแปร และสำหรับ prismatic joint $\theta_i$ เป็นค่าพารามิเตอร์ที่คงที่

   $d_i$ เป็นโคออร์ดิเนตของ origin ของเฟรม $O_i$ บนแกน $Z_{i-1}$ ได้แก่ ระยะทางระหว่าง origin ของเฟรม $O_{i-1}$ ที่ไปตัดกับแกน $Z_{i-1}$ กับ $X_i$ สำหรับ prismatic joint $d_i$ เป็นตัวแปร และสำหรับ revolute joint $d_i$ จะเป็นค่าคงที่

   $a_i$ เป็นระยะทางระหว่างแกน $Z_{i-1}$ และ $Z_i$ วัดตามแนวทางด้านลบของ $X_i$ จาก origin ของมันไปยังจุดที่ตัดกับแกน $Z_{i-1}$ (เป็นค่าคงที่)

   $\alpha_i$ เป็นมุมระหว่างแกน $Z_{i-1}$ และ $Z_i$ มาจากการหมุน $Z_{i-1}$ ไป $Z_i$ รอบแกน $X_i$ (เป็นค่าคงที่)

4. สำหรับ homogeneous displacement matrix สำหรับข้อต่อ $i$ โดยการประยุกต์สมการ (64)

$$T_{i-1}^{i} = \left[\begin{array}{ccc|c} & C_{i-1}^{i} & & d_{i-1}^{i} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (72)$$

$C_{i-1}^{i}$ เป็นเมตริกซ์ของทิศทางของ link $i$ เทียบกับ link $i-1$ และหาได้จาก

$$C_{i-1}^{i}=\begin{bmatrix}\cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i\\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i\\ 0 & \sin\alpha_i & \cos\alpha_i\end{bmatrix} \tag{73}$$

เมตริกซ์ของทิศทางจะมีเครื่องหมายเหมือนกับ DCM ของ link $i$ เทียบกับ link $i-1$ ตามหลักแล้วสามค่าพารามิเตอร์อิสระจะเกี่ยวข้องใช้การใช้กำหนดทิศทางระหว่างสองระบบพิกัดฉากใดๆ อย่างไรก็ตาม เมื่อ $C_{i-1}^{i}$ ในสมการ (73) มีเพียงสองค่าพารามิเตอร์อิสระ( $\theta_i$ และ $\alpha_i$ ) มันสามารถใช้ได้เฉพาะระบบพิกัดฉากที่มีทิศทางของแต่ละตัว โดยมีสองการหมุนที่ต่อเนื่องกัน หมุนเป็นมุม $\theta_i$ ก่อนและตามด้วย $\alpha_i$ ข้อสังเกต การหมุนของ lower-pair joints เพียงสองครั้งก็เพียงพอ ถ้าระบบพิกัดฉากของลิงค์ถูกกำหนดตามข้อตกลง DH มันจะได้มาจากเมตริกซ์ของทิศทางตามสมการ (4-73)

สามตัวประกอบด้านบนของคอลัมน์ทางขวาของ $T_{i-1}^{i}$ ในสมการ(72) เป็นส่วนประกอบของเวกเตอร์ของการเคลื่อนที่ $d_{i-1}^{i}$

$$d_{i-1}^{i}=\begin{bmatrix}a_i\cos\theta_i\\ a_i\sin\theta_i\\ d_i\end{bmatrix} \tag{74}$$

เวกเตอร์ $d_{i-1}^{i}$ ใช้บอกตำแหน่งของ origin ของเฟรม $O_i$ ในเฟรม $O_{i-1}$

**2.3.4.2 Obtaining the DH Displacement Matrices [3; หน้า 105]**

เมตริกซ์ของการกระจัดในสมการ (72) ถึง (74) ได้มาจากความต่อเนื่องของการหมุนและการเคลื่อนที่จากเฟรม $O_{i-1}$ ไปเฟรม $O_i$ ซึ่งจะมีลำดับการดำเนินการดังนี้

1. หมุน $O_{i-1}$ รอบแกน $Z_{i-1}$ เป็นมุม $\theta_i$
2. เคลื่อนเฟรม $O_{i-1}$ โดย $a_i$ และ $d_i$ เป็นค่าบนแกน $X_{i-1}$ และ $Z_{i-1}$ ตามลำดับ
3. หมุนเฟรม $O_{i-1}$ (จากในขั้นที่ 2) เป็นมุม $\alpha_i$ รอบแกน $X_{i-1}$ ก็จะเป็นเฟรม $O_i$

การประยุกต์การแปลงเมตริกซ์ในสมการ (61),(62) และ (59) ให้ผลลัพธ์ดังนี้

$$T_{i-1}^{i}=\begin{bmatrix}\cos\theta_i & -\sin\theta_i & 0 & 0\\ \sin\theta_i & \cos\theta_i & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{75}$$

$$\times\begin{bmatrix}1 & 0 & 0 & a_i\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha_i & -\sin\alpha_i & 0\\ 0 & \sin\alpha_i & \cos\alpha_i & 0\\ 0 & 0 & & 1\end{bmatrix}$$

### 2.2.5 การใช้ประโยชน์จากวิธีของ DH (APPICATIONS OF THE DH METHOD) [3; หน้า 105]

ในหัวข้อนี้จะแสดงตัวอย่างซึ่งพิสูจน์การได้มาของเมตริกซ์ของการกระจัด ตามข้อกำหนด DH ที่แสดงในหัวข้อ 2.3.4

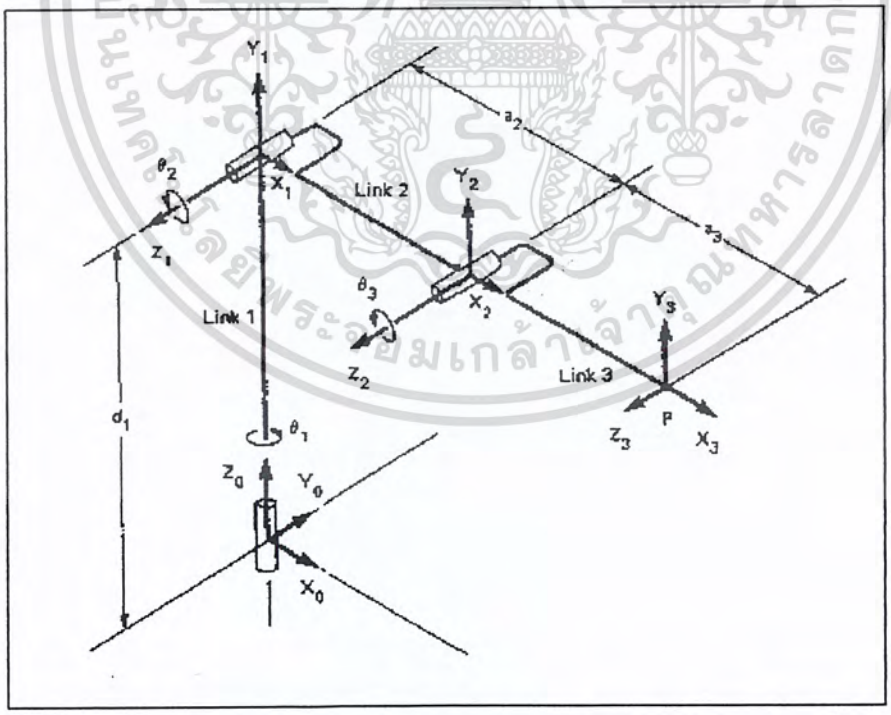
แขนหุ่นยนต์ (robot manipulators) ประกอบด้วยแขนกลและข้อมือ แขนกลเป็นการอ้างถึง link ที่เคลื่อนที่ โดยหนึ่งในสามของข้อต่อ (เริ่มนับจากฐาน) และข้อมือจะอ้างถึงลิงค์ที่เหลือ เมื่อหลายๆชนิดของแขนกลและข้อมือสามารถรวมกันอยู่ในรูปของแขนกล เราจะอภิปรายแยกเป็นส่วนๆการแปลงเมตริกซ์ของแขนจะได้มาจากลูกโซ่ของการแปลงเมตริกซ์ของแขนกลและข้อมือที่มีส่วนร่วม

#### 2.2.5.1 แขนหุ่นยนต์แบบ 3 แกน (Three-Axis Robot Arms) [3; หน้า 106]

แขนกลแบบอาร์ติกูเลทและสเฟอริคอลจะถูกอภิปรายในหัวข้อนี้

**แขนกลแบบอาร์ติกูเลท** สามแกนของแขนกลอาร์ติกูเลทกับสามข้อต่อที่หมุนได้เป็นแผนภาพแสดงในรูปที่ 2.19 ในภาพจะแสดงถึงตำแหน่งของตัวแปรของข้อต่อทั้งหมดที่ตำแหน่งอ้างอิง เช่น $\theta_i = 1$ สำหรับ $i = 1$ ถึง 3 การแปลงเมตริกซ์ที่มีลักษณะเดียวกันจะมาได้มาจากวิธีกำหนดตามหัวข้อ 2.2.4

1. แสดงจำนวนของ link ดังแสดงในรูปที่ 2.19
2. ระบบพิกัดฉากจะถูกกำหนดขึ้นตาม link แกน $Z_i$ จะมีทิศชี้ไปตามแกนของข้อต่อของการหมุน แกน $X_i$ จะถูกเลือกให้ตั้งฉากกับทั้งแกน $Z_{i-1}$ และ $Z_i$ ระบบพิกัดฉาก $O_3$ ที่ปลายสุดของแขนกลจะมีลักษณะเดียวกันกับข้อมือและมีข้อต่อแรกที่หมุนรอบ หรือเลื่อนไปตามแนวแกน $Z_3$




**รูปที่ 2.19** Links' coordinate system and joint parameters for articulated arm [3; หน้า 106]

| ข้อต่อ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $\theta_1$ | $d_1$ | 0 | $+90^\circ$ |
| 2 | $\theta_2$ | 0 | $a_2$ | 0 |
| 3 | $\theta_3$ | 0 | $a_3$ | 0 |

**ตารางที่ 2.1** ตารางตัวแปรของข้อต่อสำหรับแขนกลแบบอาร์ติคิวเลท [3; หน้า 107]

3. ตัวแปรของข้อต่อจะตั้งตามนี้ ระยะทางระหว่าง origin ของ $O_0$ และ $O_1$ ตามแนวแกน $Z_0$ คือ $d_1$ แกน $Z_0$ ตัดกับแกน $Z_1$ ดังนั้น $a_1 = 0$ มุม $\alpha_1$ วัดจากแกน $Z_0$ ไปแกน $Z_1$ รอบแกน $X_1$ (ตามกฎของมือขวา) เท่ากับ $+90^\circ$ มุม $\theta_1$ (วัดจาก $X_0$ ไป $X_1$) เป็นตัวแปรพารามิเตอร์ของข้อต่อที่ 1 โดยอย่างง่าย พารามิเตอร์ของข้อต่ออื่นๆ จะกำหนดและได้มาจากตารางที่ 2.1

4. การแปลงการกระจัดบนเมตริกซ์ กำหนดและถูกแทนด้วยค่าพารามิเตอร์ของข้อต่อ จากตารางที่ 2.1 ลงในสมการ (4-72)

$$T_0^1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{76}$$

$$T_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2S_2 \\ S_2 & C_2 & 0 & a_2C_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{77}$$

$$T_2^3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3S_3 \\ S_3 & C_3 & 0 & a_3C_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{78}$$

ซึ่ง $C_i = \cos\theta_i$ และ $S_i = \sin\theta_i$ (79)

ตอนนี้เมตริกซ์การกระจัดระหว่างเฟรม $O_3$ และ ระบบพิกัดฉากที่ฐาน($O_0$) ถูกตั้งขึ้นและใช้ประโยชน์จากการดำเนินการลูกโซ่ของสมการ (53) เป็นเมตริกซ์ในสมการ (76) ถึง (78) ซึ่งให้ผลดังนี้

$$T_0^3 = T_0^1 T_1^2 T_2^3 = \left[\begin{array}{c|c|c|c} C_1C_2C_3 + C_1S_2S_3 & -C_1C_2S_3 - C_1S_2C_3 & S_1 & a_3C_1C_2C_3 - a_3C_1S_2S_3 + a_2C_1C_2 \\ \hline S_1C_2C_3 - S_1S_2S_3 & -S_1C_2S_3 - S_1S_2C_3 & -C_1 & a_3S_1C_2C_3 - a_3S_1S_2S_3 + a_2S_1C_2 \\ \hline S_2C_3 + C_2S_3 & C_2C_3 - S_2S_3 & 0 & a_3S_2C_3 + a_3C_2S_3 + a_2S_2 + d_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (80)$$

ตำแหน่งของจุด $P$ (ปลายแขนในรูปที่ 2.19) จะอยู่ในเฟรม $O_0$ ได้มาจากเวกเตอร์การเคลื่อนที่ [ดูสมการ (71)]

$$d_0^3 = \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} = \begin{bmatrix} a_3C_1C_2C_3 - a_3C_1S_2S_3 + a_2C_1C_2 \\ a_3S_1C_2C_3 - a_3S_1S_2S_3 + a_2S_1C_2 \\ a_3S_2C_3 + a_3C_2S_3 + a_2S_2 + d_1 \end{bmatrix} \quad (81)$$

เปรียบเทียบระหว่างสมการ (81) และ (5) จะเห็นว่าทั้งคู่จะมี $d_1$ ซึ่งเป็นผลลัพธ์ของผลต่างของตำแหน่ง $O_0$ กับ $O_1$ ในรูปที่ 2.14 และ 2.19

**แขนกลแบบสเฟอริคอล** พิจารณาแขนกลแบบสเฟอริคอลที่มีหนึ่ง prismatic และสองข้อต่อที่หมุนได้ แสดงในรูปที่ 2.14 ตามวิธีการในหัวข้อ 2.2.4 link และข้อต่อจะถูกกำหนดด้วยตัวเลข และระบบพิกัดฉากของ link จะถูกกำหนดระบบพิกัดฉาก $O_3$ ที่ปลายแขนจะเป็นที่เดียวกันกับข้อมือที่มีข้อต่อที่ 1 หมุนรอบแกน $Z_3$

| Joint I | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $\theta_1$ | 0 | 0 | $+90^\circ$ |
| 2 | $\theta_2 + 90^\circ$ | 0 | 0 | $+90^\circ$ |
| 3 | 0 | $d_3$ | 0 | 0 |

**ตารางที่ 2.2** ตารางตัวแปรของข้อต่อสำหรับแขนกลแบบสเฟอริคอล [3; หน้า 108]

ตัวแปรของข้อต่อในกรณีนี้ถูกรวมอยู่ในตารางที่ 2.2 พารามิเตอร์ของข้อต่อที่ 1 จะคล้ายคลึงกับข้อต่อของแขนกลแบบอาร์ติกูเลทแต่ในกรณีนี้ $d_1 = 0$ ตัวแปรการหมุนของข้อต่อที่ 2 คือ $\theta_2 + 90^\circ$ วัดจาก $X_1$ กับ $X_2$ ซึ่ง $\theta_2$ แสดงในรูปที่ 2.14*b* ดังนั้นการแปลงจะเกี่ยวข้องเมื่อการเลือกมุมข้อต่อไม่สามารถใช้ได้โดยตรง เหมือนกับตัวแปรของข้อต่อตัวแปรของ ข้อต่อที่ 3 $d_3$ จะบอกถึงการเคลื่อนที่ของเฟรม $O_3$ ตามแกน $Z_2$ สามการแปลงเมตริกซ์การกระจัดได้มาโดยการแทนพารามิเตอร์ของข้อต่อจากตารางที่ 2.2 ลงในสมการ (72)

$$T_0^1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} -S_2 & 0 & C_2 & 0 \\ C_2 & 0 & S_2 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (82)$$

$$T_2^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ข้อสังเกต $T_0^1$ และ $T_1^2$ เป็น actually rotation matrices และ $T_2^3$ เป็นเมตริกซ์ของการเคลื่อนที่

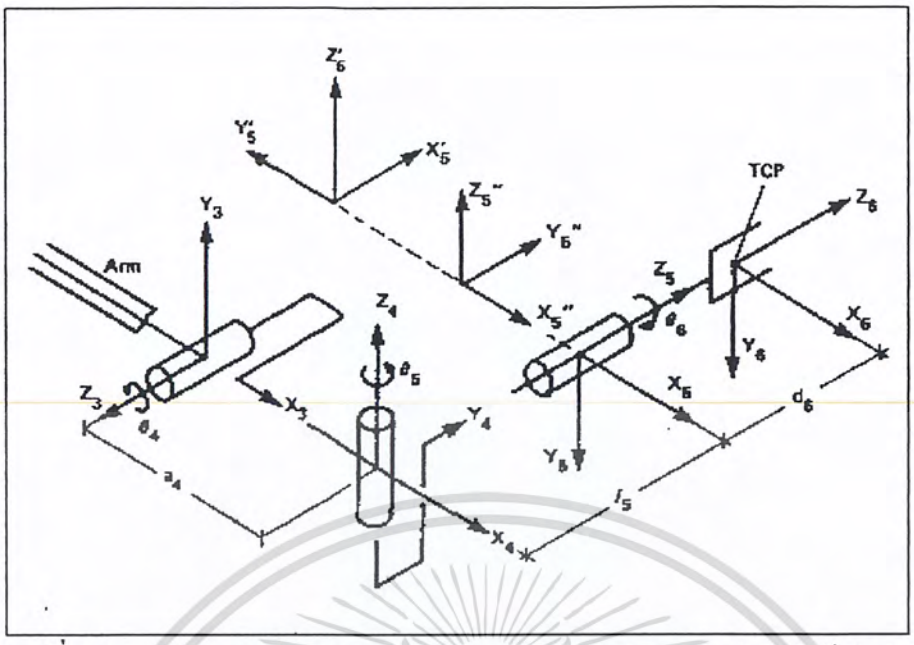
เมตริกซ์การกระจัดระหว่าง $O_3$และเฟรม $O_0$ ได้มาจากการใช้การดำเนินการลูกโซ่ของเมตริกซ์ใน สมการ (82)

$$T_0^3 = T_0^1 T_1^2 T_2^3 = \left[\begin{array}{ccc:c} -C_1S_2 & S_1 & C_1C_2 & d_3C_1C_2 \\ -S_1S_2 & -C_1 & S_1C_2 & d_3S_1C_2 \\ C_2 & 0 & S_2 & d_3S_2 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \qquad (83)$$

ตำแหน่งของ TCP ในรูปที่ 2.14*b* ได้มาจากโดยเวกเตอร์ของการเคลื่อนที่

$$d_0^3 = \begin{bmatrix} X_t \\ Y_t \\ Z_t \end{bmatrix} = \begin{bmatrix} d_3C_1C_2 \\ d_3S_1C_2 \\ d_3S_2 \end{bmatrix} \qquad (84)$$

ซึ่งถูกกำหนดมาจากสมการ (3) ได้มาจากการใช้เรขาคณิตโดยตรง




**รูปที่ 2.20** Links' coordinate and joint parameters for a bend-bend-roll wrist [3; หน้า 110]

**2.2.5.2 ข้อมือแบบสามแกน [3; หน้า 110]**

สองชนิดของข้อมือหุ่นยนต์จะนำมาทดสอบ ได้แก่ชนิด bend- bend-roll (BBR) และ roll- bend-roll (RBR)

**Bend-Bend-Roll Wrist** เป็นดังแผนภาพ รูปที่ 2.20 กับตัวแปร joint ทั้งหมดในตำแหน่งที่อ้างอิงข้อมือแบบ BBR จะใช้ใน $T^3$ robot และมีการเคลื่อนที่แบบ pitch-yaw-roll joint จะถูกนับโดยเริ่มจาก joint ที่ 4 ซึ่งเคลื่อนข้อมือ (wrist) สัมพันธ์กับเฟรม $O_3$ ซึ่งถูกกำหนดเป็นปลายแขน เฟรม $O_6$ จะถูกกำหนดที่ joint ที่ 6 (ได้แก่ end effector) กับ origin ที่ TCP และแกน $Z_6$ จะชี้ตามทิศทางของเครื่องมือ พารานิเตอร์ของ joint สำหรับข้อมือแบบ BBR จะรวบรวมอยู่ในตารางที่ 2.3

| Joint I | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 4 | $\theta_4$ | 0 | $a_4$ | -90° |
| 5 | $\theta_5$ | - | - | -90° |
| 6 | $\theta_6$ | $d_6$ | 0 | 0 |

- ไม่ระบุ

**ตารางที่ 2.3** ตารางตัวแปรของ joint สำหรับข้อมือแบบ BBR Wrist [3; หน้า 110]

จากการกำหนดที่อยู่หัวข้อ 2.2.4 มันเป็นไปไม่ได้ที่ระยะทาง $l_5$ จะเป็น $a_5$ หรือ $d_5$ (ดูรูปที่ 2.20) ตัวแปรของ joint $a_5$ จะบอกถึงระยะทางระหว่าง $Z_4$ และ $Z_5$ และจะเป็นศูนย์เมื่อ $Z_4$ ตัดกับ $Z_5$ นอกจากนั้น ตัวแปรของข้อต่อ $d_5$ จะบอกถึงการเคลื่อนที่ของเฟรม $O_5$ ตามแนวแกน $Z_4$ และเป็นศูนย์ด้วย ความพยายามที่จะละความขัดกันโดยการเลือกระบบพิกัดฉากอื่นสำหรับ $O_5$ หรือ $O_4$ จะไม่มีประโยชน์ในกรณีนี้

สำหรับกรณีนี้ เป็นเหตุการณ์ที่เกิดขึ้นได้ในหลายๆรูปร่างของ joint วิธีการปฏิบัติที่แนะนำคือ :

สมมติระบบพิกัดฉาก $O'_5$ กับแกนที่เหมือนกัน (แต่ไม่ matchingกัน) กับแกนของเฟรม $O_5$ ถูกกำหนดเป็น link ที่ 5 เฟรมนี้จะถูกเลือกแล้วพารามิเตอร์ของ joint จะนำมาใช้กำหนดได้ ซึ่งเมตริกซ์ของการกระจัดระหว่าง $O_4$ และ $O'_5$ จะหาได้ จากนั้นการแปลงเมตริกซ์ของการหมุนจะใช้ในการแปลง $O'_5$ เป็น $O_5$ ในภายหลัง การแปลงเมตริกซ์ระหว่าง $O_4$ และ $O_5$ จะถูกตั้งโดยใช้การดำเนินการลูกโซ่สำหรับตัวอย่างนี้ (รูปที่ 2.20) ระบบพิกัดฉาก $O'_5$ ถูกเลือกและพารามิเตอร์ของ joint ที่ 5 จะถูกกำหนดเป็น $\theta'_5 = \theta_5 + 90°, d'_5 = 0, a'_5 = l_5, \alpha'_5 = 0$ การแปลงเมตริกซ์ของการกระจัดจึงเป็น

$$T_4^{5'} = \begin{bmatrix} -\sin\theta_5 & -\cos\theta_5 & 0 & -l_5\sin\theta_5 \\ \cos\theta_5 & -\sin\theta_5 & 0 & l_5\cos\theta_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{85}$$

ตอนนี้ สองการหมุนที่ต่อเนื่องของ $O'_5$ กำหนดโดยการแปลงเมตริกซ์ระหว่าง $O_5$ และ $O'_5$ ขั้นแรก $O'_5$ จะหมุนรอบแกน $Z'_5$ ใช้สมการ (61) โดย $\theta = -90°$ อยู่ในรูปเฟรม $O''_5$ จากนั้น $O''_5$ จะหมุนรอบแกน $X''_5$ ใช้สมการ (59) โดย $\alpha = -90°$ มาอยู่ในรูปเฟรม $O_5$

$$T_{5'}^{5} = T_{R5'}^{5'} T_{R5''}^{5} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{86}$$

การแปลงระหว่างเฟรม $O_4$ และ $O_5$ คือ

$$T_4^5 = T_4^{5'} T_{5'}^5 \tag{87}$$

ค่าในสมการ (86) สามารถเช็คได้โดย

$$\begin{bmatrix} X'_5 \\ Y'_5 \\ Z'_5 \\ 1 \end{bmatrix} = T_{5'}^5 \begin{bmatrix} X_5 \\ Y_5 \\ Z_5 \\ 1 \end{bmatrix} = \begin{bmatrix} Z_5 \\ -X_5 \\ -Y_5 \\ 1 \end{bmatrix} \tag{88}$$

$Z_5$ จะมีทิศทางเดียวกันกับ $X'_5$; $-X_5 \equiv Y'_5$ และ $-Y_5 \equiv Z'_5$

การแปลงเมตริกซ์สำหรับข้อมือแบบ BBR คือ

$$T_3^4 = \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & a_4\cos\theta_4 \\ \sin\theta_4 & 0 & \cos\theta_4 & a_4\sin\theta_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^5 = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & -l_5\sin\theta_5 \\ \sin\theta_5 & 0 & \cos\theta_5 & l_5\cos\theta_5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (89)$$

$$T_5^6 = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

จากสมการ (89) จะเห็นว่ามันเป็นไปไม่ได้ที่จะหาตัวแปร joint สำหรับหาค่า $T_4^5$ นอกสมการ (72) ตามข้อกำหนด DH

ระยะทางของเมตริกซ์ที่ถูกแปลง (Displacement transformation matrix) ระหว่าง TCP และขอบของแขนหุ่นยนต์ สำหรับ BBR wrist คือ

$$T_3^6 = T_3^4 T_4^5 T_5^6 \qquad (90)$$

**ข้อมือ Roll-Bend-Roll (RBR)** จะถูกใช้ทั่วไปในแขนหุ่นยนต์ (ได้แก่ PUMA, $d_4 = l_5 = 0$) ซึ่งแสดงเป็นตัวอย่างอยู่ในรูปที่ 2.21 ตัวแปรของ joint ทั้งหมดจะแสดงอยู่ในตำแหน่งอ้างอิงของมัน

พารามิเตอร์ของ joint สำหรับข้อมือนี้จะรวมอยู่ในตารางที่ 2.4 การแปลงเมตริกซ์ระหว่าง $O_4$ และ $O_5$ เป็นการแสดงการกระทำของเมตริกซ์ที่ตั้งที่ข้อมือแบบ BBR โดยจะหมายถึงมีสองการแปลง

การแปลงเมตริกซ์สำหรับ RBR wrist คือ

$$T_3^4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

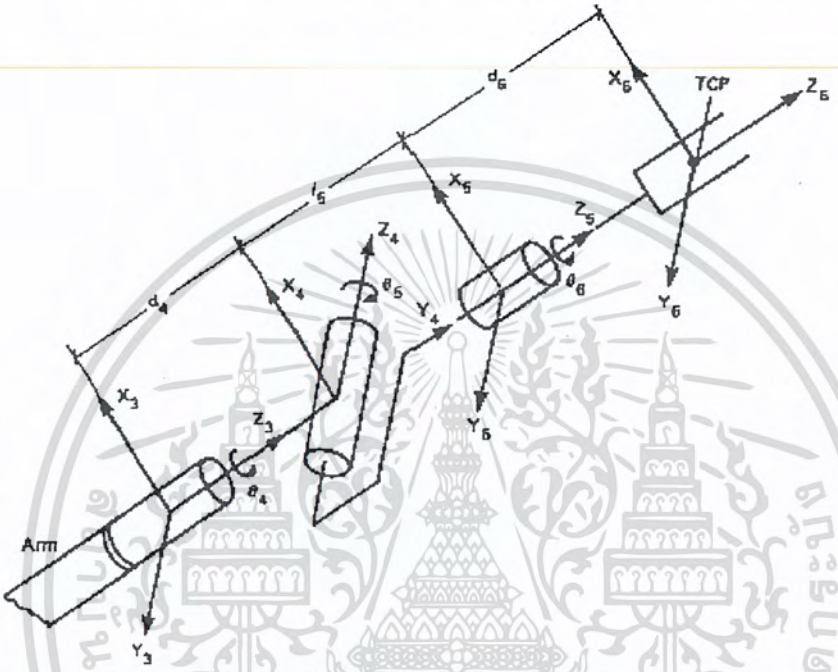
$$T_4^5 = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & l_5\sin\theta_5 \\ \sin\theta_5 & 0 & \cos\theta_5 & l_5\cos\theta_5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (91)$$

$$T_5^6 = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

| ข้อต่อที่ I | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 4 | $\theta_4$ | $d_4$ | 0 | +90° |
| 5 | $\theta_5$ | - | - | -90° |
| 6 | $\theta_6$ | $d_6$ | 0 | 0 |

- ไม่นิยาม

**ตารางที่ 2.4** ตารางตัวแปรของ Joint parameter สำหรับข้อมือแบบ RBR Wrist [3; หน้า 113]




**รูปที่ 2.21** Link's coordinate systems and joint parameters for a roll-bend-roll wrist [3; หน้า 113]

การแปลงเมตริกซ์ของการกระจัดระหว่าง TCP และ ปลายแขนกลได้มาจากสมการ (90) สำหรับเมตริกซ์ของสมการ (91)

**2.2.5.3 แขนหุ่นยนต์แบบ 6 แกน (Six-Axis Robot Manipulators) [3; หน้า 114]**

**แขนกลแบบอาร์ติคิวเลทกับข้อมือแบบ BBR** แขนแบบ 6 แกนสร้างจากการรวมข้อมือแบบ BBR ในรูปที่ 2.20 เป็นแขนแบบอาร์ติคิวเลท ในรูปที่ 2.19 การแปลงเมตริกซ์สำหรับผลลัพธ์ของแขนแบบ 6 แกนได้มาจากแกน ลูกโซ่ของการแปลงเมตริกซ์ของข้อมือกับแขนกล เมื่อเฟรม $O_3$ ในรูปที่ 2.20 match กับเฟรม $O_3$ ในรูปที่ 2.19 เมื่อเราผูกกันได้ การคำเนินการลูกโซ่ก็สามารถนำมาใช้ได้

$$T_0^6 = T_0^3 T_3^6 \tag{92}$$

ซึ่ง $T_0^3$ จะมาจากสมการ (80) สำหรับแขนแบบอาร์ติคิวเลท $T_3^6$ ได้มาจากสมการ (90) สำหรับข้อมือแบบ BBR และ $T_0^6$ เป็นการแปลงเมตริกซ์ของผลลัพธ์ของแขนกลตำแหน่งของ TCP ใน WCS ($O_0$) $d_0^6$ และ การกำหนดทิศทางของ end effector $C_0^6$ ได้มาจาก $T_0^6$ ตามสมการ (71)

**แขนกลแบบสเฟอริคอลกับข้อมือแบบ BBR** ข้อมือแบบ BBR ในรูปที่ 2.20 มาจากการรวมของแขนกลแบบสเฟอริคอล ในรูปที่ 2.14*b* อยู่ในรูปแขนแบบ 6 แกน ลูกโซ่ของการแปลงเมตริกซ์ของข้อมือและแขนกล ไม่

สามารถหาได้โดยตรงจากสมการ (92 ) ในกรณีที่แล้ว สองเฟรม $O_3$ ในรูปที่ 2.20 และ 2.14*b* ไม่สอดคล้องในคำสั่งที่จะทำให้การคำเนินการลูกโซ่ทำได้ ต้องเพิ่มการแปลงเมตริกซ์ระหว่างเฟรม $O_3$ ในรูปที่ 2.14*b* และ $O_3$ ในรูปที่ 2.20

$$T_a^w = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{93}$$

ซึ่ง w และ a เป็นเครื่องหมายของ wrist และแขนกล ตามลำดับ เมตริกซ์ $T_a^w$ (ได้จากการหมุน $O_3$ ในรูปที่ 2.20 เทียบกับ $O_3$ ในรูปที่ 2.14*b*) ซึ่งใช้แปลงจุดโคออร์ดิเนตจากเฟรม $O_3$ ของ wrist เป็น เฟรม $O_3$ ของแขนกล ดังนี้

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_a = T_a^w \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_w = \begin{bmatrix} Y \\ Z \\ X \\ 1 \end{bmatrix}_w \tag{94}$$

ตอนนี้การคำเนินการลูกโซ่จะกระทำดังนี้

$$T_0^6 = T_0^3 T_a^w T_3^6 \tag{95}$$

ซึ่ง $T_0^3$ ได้จากสมการ (83) $T_a^w$ ได้จากสมการ (93) $T_3^6$ ได้จากสมการ (90) และ $T_0^6$ เป็นการแปลงเมตริกซ์สำหรับผลลัพธ์แขนกล

ข้อสังเกต ผลของ $T_a^w$ เป็นการเคลื่อนเฟรม $O_3$ ในรูปที่ 2.14*b* โดยระบบพิกัดอื่นและต่อมาจะเคลื่อน $T_2^3$ ในสมการ (82) โดย

$$T_2^3 T_a^w = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{96}$$

กับสมการ (4-92) สามารถใช้แทนสมการ (4-95) ได้

**แขนกลกับข้อมือแบบ RBR** การรวมข้อมือแบบ RBR ด้วยแขนแบบสเฟอริกอลกระทำตามสมการ (92) เมื่อสองเฟรม $O_3$ ทั้งของแขนและข้อมือสอดคล้องกัน

การรวมข้อมือแบบ RBR ด้วยแขนแบบอาร์ติคิวเลท ดังแสดงในรูปที่ 2.17 (forming a PUMA-type manipulator) กระทำตามสมการ (95) ด้วย

$$T_a^w = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{97}$$

และสำหรับ $T_0^3$ และ $T_3^6$ ได้มาจากสมการ (80) และ (91) ตามลำดับ

**2.2.5.4 การกำหนดระบบพิกัดของเครื่องมือ (Assigning the Tool Coordinate System) [3; หน้า 115]**

จากตัวอย่างของสองชนิดของข้อมือ เฟรม $O_6$ ถูกกำหนด end effector ด้วย origin ที่ TCP และแกน $Z_6$ มีทิศตามทิศทางของ end effector ดังนั้นบ่อยครั้งที่ระบบจะถูกแทนเป็น TCP

ในสอง TCS ที่แสดงในรูปที่ 2.20 และ 2.21 แกน $Z_6$ และ TCPจะถูกแสดงสอดคล้องกับแกน $Z_5$ อย่างไรก็ตามในกรณีที่ทิศทางของแกน $Z_6$(เขียนตามทิศทางของ end effector สัมพันธ์กับขอบของเฟรม $O_5$) และที่ตั้งของ TCP แสดงในเหตุการณ์ที่ต่างกัน สำหรับกรณีนี้ การแปลงเมตริกซ์ระหว่าง $O_6$ และ $O_5$ จะหาได้พื้นฐานบนพารามิเตอร์ในภาษาของโปรแกรม (programming language) จะบอกถึง end effector พารามิเตอร์นี้ (ซึ่งขึ้นอยู่กับภาษาของโปรแกรม) จะถูกใช้โดยตัวกระทำเพื่อบอกตำแหน่งและทิศทางของ end effector เทียบกับปลายของข้อมือ

สำหรับตัวอย่าง พิจารณาภาษาของโปรแกรมซึ่งใช้สามพารามิเตอร์เพื่อบอกถึงที่ตั้งของ TCP สัมพันธ์กับศูนย์กลางของครีบที่ประกอบที่ปลายของข้อมือ สมมติว่าข้อมือแบบ RBR (ตัวอย่างในรูปที่ 2.21) สามพารามิเตอร์จะถูกกำหนดอยู่ในภาษาของโปรแกรมสัมพันธ์กับระบบพิกัดอ้างอิงซึ่งกำหนดเป็นศูนย์กลางของครีบและมีแกน $Z$ ที่ทับสนิทกับแกน $Z_5$ ในกรณีนี้เฟรม $O_5$ สามารถเป็นเหมือนกับระบบพิกัดอ้างอิงกับ $\theta_6 = 0$ และพารามิเตอร์ของ joint $l_5$ เลือกมาจาก origin $O_5$ ถึงขอบของครีบ สามพารามิเตอร์ ($\varepsilon_1, \varepsilon_2$ และ $\varepsilon_3$) จะบอกถึงโคออร์ดิเนตของ TCP ใน $X_5, Y_5$ และ $Z_5$ ตามลำดับ

มันจำเป็นที่จะต้องหาการแปลงเมตริกซ์ระหว่าง $O_6'$ ที่อยู่ที่ TCP และเฟรม $O_5$ เมตริกซ์นี้จะเป็นฟังก์ชันของสามพารามิเตอร์ ($\varepsilon_1, \varepsilon_2$ และ $\varepsilon_3$) และของมุม $\theta_6$ จากรูปที่ 2.21 เมตริกซ์นี้สามารถหาได้จากสอง successive transformations

$$T_5^{6'} = T_5^6 \begin{bmatrix} 1 & 0 & 0 & \varepsilon_1 \\ 0 & 1 & 0 & \varepsilon_2 \\ 0 & 0 & 1 & \varepsilon_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{98}$$

ซึ่ง $T_5^6$ ได้มาจากสมการ (91) กับ $d_6 = 0$ และเมตริกซ์ที่สองจะเป็นเมตริกซ์การเปลี่ยนตำแหน่ง $O_6'$ สัมพันธ์กับเฟรม $O_6$ สามารถแสดงได้หลังจากการคูณเมตริกซ์ และจัดให้อยู่ในเทอมของสมการ (98) จะอยู่ในรูป

$$T_5^{6'} = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & a_6\cos\phi_6 \\ \sin\theta_6 & \cos\theta_6 & 0 & a_6\sin\phi_6 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{99}$$

ซึ่ง $a_6 = (\varepsilon_1^2 + \varepsilon_2^2)^{1/2}$

$\phi_6 = \cos^{-1}(\varepsilon_1 / a_6) + \theta_6$

$d_6 = \varepsilon_3$

สำหรับภาษาของโปรแกรม ซึ่งเป็นไปได้ในการกำหนดทิศทางของ end effector ซึ่งสัมพันธ์กับครีบ การแปลงเมตริกซ์ระหว่าง TCS และเฟรม $O_5$ คือ

$$T_5^{6'} = \left[\begin{array}{cccc} \cos\theta_6 & -\sin\theta_6 & 0 & a_6\cos\phi_6 \\ \sin\theta_6 & \cos\theta_6 & 0 & a_6\sin\phi_6 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc:c} & C* & & \begin{matrix}0\\0\\0\end{matrix} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{100}$$

ซึ่ง $C*$ จะบอกถึงทิศทางของ end effector (หรือเครื่องมือ) สัมพันธ์กับครีบของข้อมือ เมื่อเป็นการกล่าวถึงนอกเหนือจากทิศทางเมตริกซ์จะเป็น DCM programming languages จะต้องมีการกำหนดของสามพารามิเตอร์อิสระ ในทางอื่นภาษาของโปรแกรมสามารถใช้ได้กับพารามิเตอร์อื่นๆ ซึ่งครอบคลุมนอกเหนือจากตัวกระทำ ซึ่ง

$C*$ สามารถสร้างได้โดยอ้อม เมื่อการแสดงใช้พื้นฐานของมุมออยเลอร์หรือเวกเตอร์ของการหมุนตามหัวข้อ 2.2.6

### 2.2.6 QUATERNION AND ROTATION VECTOR PREPRESENTATION [3; หน้า 117]

ในหัวข้อที่แล้ว DCM จะใช้ในการกำหนดความสัมพันธ์ของทิศทางของสองระบบพิกัด วิธีการอื่นๆที่จะใช้กำหนดความสัมพันธ์นี้และการทำการแปลงพิกัดโดยการใช้ quaternion หรือ rotation vector ข้อดีของวิธีการนี้คือการกำหนดความสัมพันธ์ของทิศทางที่ประกอบด้วยสามหรือสี่พารามิเตอร์ที่แทนอยู่ในเมตริกซ์ $3\times3$ เหมือนกับ DCM definition นอกจากนั้นจำนวนของการดำเนินการทางคณิตศาสตร์จะเกี่ยวข้องกับ kinematic solution ของแขนของหุ่นยนต์สามารถลดได้โดยการใช้วิธี quaternion หรือ rotation vector

การแสดงความสัมพันธ์ของทิศทางโดย quaternion และ rotation vector เป็นพื้นฐานบนทฤษฎีขอออยเลอร์ ซึ่งกำหนดการเปลี่ยนตำแหน่งของ rigid body ด้วยหนึ่งจุดที่กำหนดสามารถบอกได้เช่นเดียวกับการหมุนรอบบางแกน (Begges, 1966; Duffy, 1980) เวกเตอร์ของการหมุนเป็นเวกเตอร์ที่ชี้ไปตามแกนนี้ และจะมีขนาดที่เป็นข้อมูลของมุมของการหมุน ต่อมาเวกเตอร์ของการหมุนเป็นเหมือนกับเวกเตอร์ธรรมดาของสองระบบพิกัดใดๆ(ได้แก่ เฟรม) มี origin เหมือนกันแต่มีทิศทางที่ต่างกัน ลักษณะนี้เป็นการประยุกต์ของการกระทำการแปลงเวกเตอร์ระหว่างสองเฟรม ทิศทางของเวกเตอร์ของการหมุนกำหนดแกนโดยรอบซึ่งมีหนึ่ง ระบบพิกัดที่หมุนทิศทางเดียวกับระบบอื่น ขนาดของมันจะกำหนดปริมาณของการหมุน ประโยชน์ของเวกเตอร์ของการหมุนหนึ่งตัวสามารถกำหนดความสัมพันธ์ของสองระบบ และการกระทำการแปลงเวกเตอร์จากหนึ่งระบบพิกัดเป็นอย่างอื่น

quaternion คือ 4 เวกเตอร์หลัก โดยจะมี 3 เวคเตอร์ ที่เป็นส่วนประกอบของเวคเตอร์ระนาบและกำหนดทิศทางของแกนหมุน (ตัวอย่างเช่น การทับกันสนิทของทิศทางของ rotation vector) และองค์ประกอบที่ 4 จะเป็นตัวที่ให้ข้อมูลเกี่ยวกับมุมของการหมุน

เครื่องมือ 2 ตัวของ quaternion และ rotation vector จะถูกอธิบายไว้ดังนี้ ; การเปลี่ยนรูปเวกเตอร์ระหว่าง 2 เฟรมและการแสดงความต่อเนื่องการหมุนของเฟรม โดยการหมุนเชิงเดี่ยว

#### 2.2.6.1 การกำหนดควอเตอร์เนี่ยน (Quaternion Definition) [3, หน้า 118]

ควอเตอร์เนี่ยน $q$ มีสี่ส่วน ซึ่งมีการกำหนดทิศทางอย่างสมบูรณ์ของแกนขแงการหมุนปริมาณของการหมุน มันประกอบด้วยส่วนที่เป็นสกาล่าร์ $q_0$ และเวกเตอร์ q สามารถเขียนได้เป็น

$$\begin{aligned} q &= q_0 + \mathrm{q} = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k} \\ q &= (q_0, \mathrm{q}) \end{aligned} \tag{101}$$

ซึ่ง $\hat{i}, \hat{j}$ และ $\hat{k}$ เป็น unit vector ตามแกน $X, Y$ และ $Z$ ตามลำดับ มันเป็นเครื่องช่วยในการแสดงเวกเตอร์ที่เหมือนกับ quaternion ด้วยส่วนประกอบหลักที่เป็นสกาล่าร์เท่ากับ 0

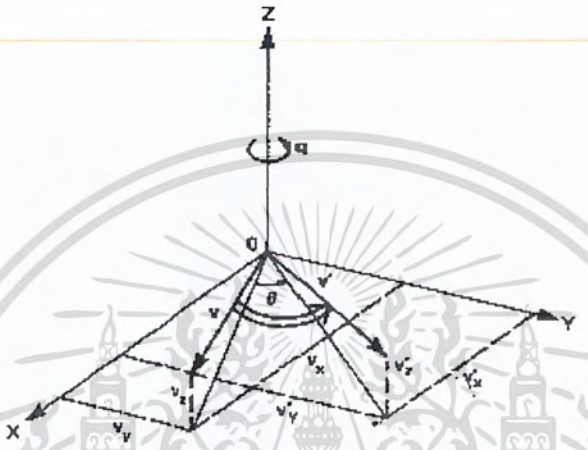
quaternion ซึ่งใช้ในการกำหนดความสัมพันธ์ของการหมุนด้วยมุม $\theta$ รอบแกน ซึ่งทิศทางถูกกำหนดโดย unit vector $\hat{e}$ กำหนดโดยออยเลอร์ดังนี้

$$q = \cos\frac{\theta}{2} + \hat{e}\sin\frac{\theta}{2} \tag{102}$$

หรือ $$q = \cos\frac{\theta}{2} + (e_1\hat{i} + e_2\hat{j} + e_3\hat{k})\sin\frac{\theta}{2}$$

ซึ่ง $e_1, e_2$ และ $e_3$ เป็น direction cosines ของ $\hat{e}$ ในทางอื่น quaternion สามารถเขียนได้เป็น

$$q = q_0 + \mathrm{q} = \cos\frac{\theta}{2} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}\sin\frac{\theta}{2} \qquad (103)$$




**รูปที่ 2.22** การหมุนของเวกเวกเตอร์รอบแกน Z [3; หน้า 119]

**2.2.6.2 การแปลงเวกเตอร์โดยการใช้ควอเตอร์เนี่ยน (Vector Transformation Using the Quaternion) [3; หน้า 119]**

สมมติเวกเตอร์ v ในเฟรมได้ผ่านการหมุนอธิบายโดย quaternion $q$ ในเฟรม $O$ เวกเตอร์ของผลลัพธ์ในเฟรม $O$ คือ v' และมันสัมพันธ์กับ v ได้มาจากสมการ (Hillman, 1964;Shoham, 1982;Wiener, 1962;Whipple, 1971)

$$v' = v + 2q_0(\mathrm{q}\times v) + 2\mathrm{q}\times(\mathrm{q}\times v) \qquad (104)$$

สมการ(104) คือการแสดงเหมือนการแปลงพิกัด ซึ่งหมายถึงการอธิบายของส่วนประกอบของเวกเตอร์ในสองเฟรมกับทิศทางที่แตกต่าง (เวกเตอร์จะถูกกำหนดอยู่ใน space และไม่หมุนเหมือนในตอนแรก) ในสมการ (104) เวกเตอร์ที่ถูกกำหนด v ในเฟรม $O$ จะมองเห็นเหมือน v'ในเฟรม $O'$ ความสัมพันธ์ระหว่างสองเฟรมได้มาโดย quaternion $q$ ใน $O'$

ตัวอย่าง พิจารณาเวกเตอร์ v ในเฟรม $O$ ที่แสดงในรูปที่ 2.22 ซึ่งผ่านการหมุนกำหนดโดย $q$ แกนของการหมุนจะเป็นไปตามสมการ (103) คือ

$$q = \cos\frac{\theta}{2} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\sin\frac{\theta}{2}$$

ซึ่ง $\theta$ เป็นมุมของการหมุนรอบแกน $Z$

ผลลัพธ์ของการหมุนเวกเตอร์ในเวกเตอร์ $v'$ ในเฟรม $O$ ได้จากสมการ (104)

$$\begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} + 2\cos\frac{\theta}{2}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \sin\frac{\theta}{2} + 2\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left( \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \right) \sin\frac{\theta}{2}$$

$$= \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} + 2\cos\frac{\theta}{2}\sin\frac{\theta}{2}\begin{bmatrix} -v_y \\ v_x \\ 0 \end{bmatrix} + 2\sin^2\frac{\theta}{2}\begin{bmatrix} -v_x \\ -v_y \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} v_x\cos\theta - v_y\sin\theta \\ v_y\cos\theta - v_x\sin\theta \\ v_x \end{bmatrix}$$

ผลลัพธ์นี้เป็นผลลัพธ์เดียวกันกับในสมการ (10)

**2.2.6.3 Successive Rotations [3; หน้า 120]**

สอง Successive rotations ซึ่งจะแสดงโดยสอง quaternion $g$ และ $g'$ สามารถแสดงเป็นหนึ่งการหมุนเดี่ยวๆ (single rotation) อธิบายโดยหนึ่ง quaternion ตามสมการ (Iackes, 1970;Shoham, 1982;Whittaker, 1944)

$$\begin{aligned} q = gg' &= (g_0, g)(g_0, g') \\ &= (g_0 g'_0 - g \cdot g', g_0 g' + g'_0 g + g \times g') \end{aligned} \tag{105}$$

แต่ละ quaternion $g$ และ $g'$ ในสมการข้างบนเป็นการแก้ปัญหาในที่ตั้งของระบบพิกัด ซึ่งมีการหมุนอยู่ภายใน

ตัวอย่าง พิจารณาเวกเตอร์ v ในรูปที่ 2.23 เป็นสอง successive rotations ขั้นแรกของการหมุนถูกกำหนดโดย quaternion $g$ หมุนเวกเตอร์ v และระบบพิกัดอ้างอิง $O'$ ที่ติดอยู่กับ v (โดยทั่วไป $O'$ จะทับกันสนิทกับ $O$) เป็นมุม $90°$ รอบแกน $Z$ เกิดเป็นเวกเตอร์ v′ การหมุนครั้งที่สอง กำหนดโดย quaternion $g'$ ในเฟรม $O'$ หมุนเวกเตอร์ v′ และเฟรม $O''$ ที่ติดอยู่กับเวกเตอร์เป็นมุม $90°$ รอบแกน $X'$ เข้าสู่ตำแหน่งสุดท้าย การหาส่วนประกอบของ v ที่ตำแหน่งสุดท้าย ( v″ ) ถ้าจุดเริ่มต้นคือ $\mathrm{v} = [1,0,0]^{\mathrm{T}}$

การแก้ปัญหาจะแบ่งเป็นสองส่วน ขั้นแรกแสดงสอง successive rotation เป็น single composite rotation และการหมุนครั้งที่สองใช้เวกเตอร์ v ตลอดการหมุนนี้หาตำแหน่งสุดท้าย เพื่อหา composite rotation $q$ ในแต่ละ quaternion $g$ และ $g'$ มีการกำหนดที่ตั้งของระบบพิกัดของมัน

quaternion $g$ สัมพันธ์กับเฟรม $O$ และ $O'$ และจะกำหนดตามสมการ (102) สำหรับการหมุนที่ $\theta = 90°$ รอบแกน Z

$$g = (g_0, g) = \left( \frac{1}{\sqrt{2}}, 0\hat{i} + 0\hat{j} + \frac{1}{\sqrt{2}}\hat{k} \right)$$

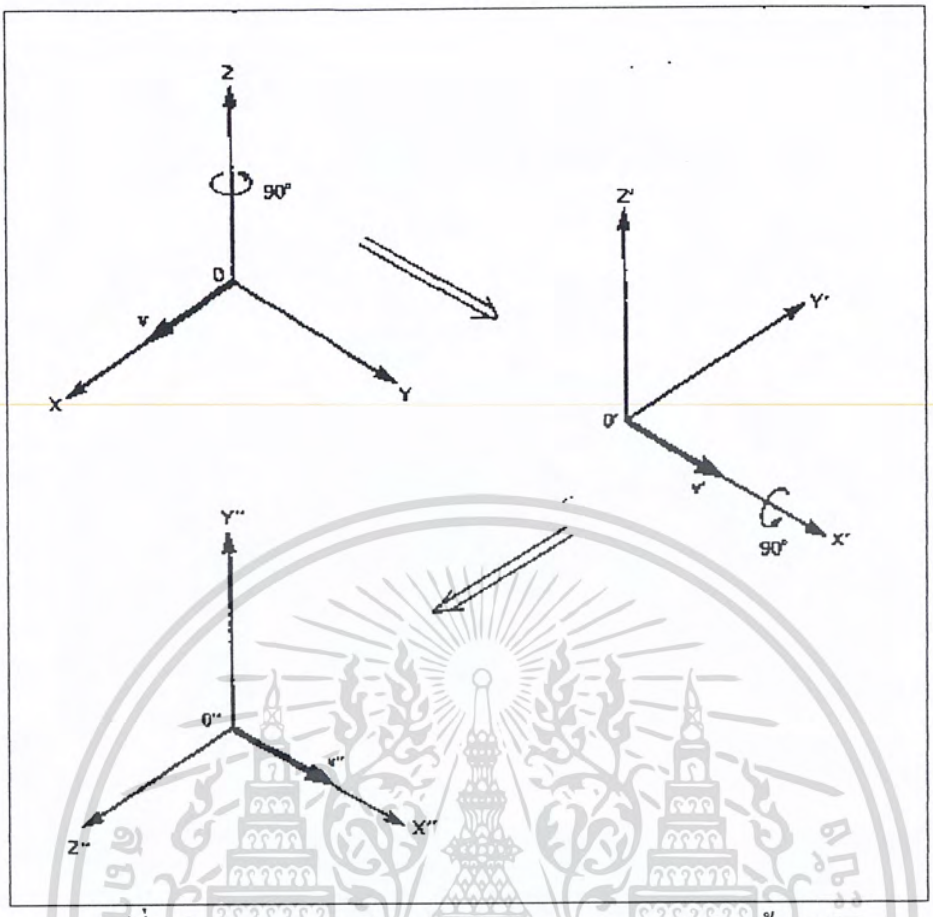
ในทำนองเดียวกัน $g'$ สัมพันธ์กับเฟรม $O'$ และ $O''$ และถูกกำหนดให้หมุน $90°$ รอบแกน $X'$

$$g' = (g'_0, g') = \left( \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\hat{i}' + 0\hat{j}' + 0\hat{k}' \right)$$

composite rotation, $q$ หาได้ตามสมการ (105)

$$q = gg' \left( \tfrac{1}{2}, \tfrac{1}{2}\hat{i} + \tfrac{1}{2}\hat{j} + \tfrac{1}{2}\hat{k} \right) \tag{106}$$

ข้อสังเกต composite rotation $q$ ได้มาจากในเฟรม $O$ และสัมพันธ์กับเฟรม $O''$ ถึง $O$




**รูปที่ 2.23** The vector direction after two successive rotations [3; หน้า 121]

ขั้นต่อมาในการแก้ปัญหา คือการหมุนเวกเตอร์ v ตลอดจน composite rotation ที่ได้จากสมการ (106) โดยใช้สมการ (104)

$$v'' = \begin{bmatrix}1\\0\\0\end{bmatrix} + 2\left(\tfrac{1}{2}\right)\begin{bmatrix}\tfrac{1}{2}\\\tfrac{1}{2}\\\tfrac{1}{2}\end{bmatrix} \times \begin{bmatrix}1\\0\\0\end{bmatrix} + 2\begin{bmatrix}\tfrac{1}{2}\\\tfrac{1}{2}\\\tfrac{1}{2}\end{bmatrix} \times \left(\begin{bmatrix}\tfrac{1}{2}\\\tfrac{1}{2}\\\tfrac{1}{2}\end{bmatrix} \times \begin{bmatrix}1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\1\\0\end{bmatrix} \qquad (107)$$

ผลลัพธ์ของสมการ (107) แสดง unit vector ตามแกน $X$ หลังจากเกิดสองการหมุนจะกลายเป็น unit vector ตามแกน $Y$ สามารถพิสูจน์ได้ในรูปที่ 2.23 ข้อสังเกต ซึ่งการแปลงจาก v เป็น v″ สามารถหาได้โดย single rotation ของ $\theta = 120°$ รอบแกนซึ่งเท่ากับ projection บนแกน $X, Y$ และ $Z$

**2.2.6.4 เวกเตอร์ของการหมุน (Rotation Vector) [3; หน้า 122]**

วิธีการแสดงอื่นๆของการหมุน โดยใช้การแทนค่าด้วยเวกเตอร์สามมิติของ สี่-องค์ประกอบ quaternion พื้นฐานของการแสดงนี้คล้ายกันกับการแสดงของ quaternion และประกอบด้วยการมีอยู่ของแกนสามัญรอบระบบพิกัด ที่มีการหมุนสำเร็จในทิศทางเดียวกับระบบพิกัดอื่นๆ เวกเตอร์ตามแนวแกนสามัญนี้ขนาดของการกำหนดโดยปริมาณของการหมุนรอบแกนจะเรียกว่าเวกเตอร์ของการหมุน หลายๆการกำหนดสำหรับขนาดของเวกเตอร์ของการหมุน (Shoham, 1982;Wiener, 1962) อย่างไรก็ตาม เราจะนิยมที่จะใช้ตามการนิยามสำหรับเวกเตอร์ของการหมุน **R**

$$\mathbf{R} = \hat{e}\tan\frac{\theta}{2} = \hat{e}f \qquad (108)$$

ซึ่ง $\hat{e}$ = unit vector ตามแนวของการหมุน

$\theta$ = มุมของการหมุน $\hat{e}$

$f$ = $\tan\theta/2$ = ขนาดของเวกเตอร์ของการหมุน

เวกเตอร์ของการหมุนสามารถนำมาใช้ได้ในทำนองเดียวกันกับควอเตอร์เนี่ยนสำหรับการแปลงเวกเตอร์ และในการประยุกต์การแสดง single rotation สำหรับหลายๆการหมุน จะถูกอธิบายข้างล่าง

เวกตอร์ v ที่ผ่านการหมุน R เป็น v′ สามารถแสดงได้โดย

$$\mathbf{v}' = \mathbf{v} + \frac{2\mathbf{R}\times(\mathbf{v}+\mathbf{R}\times\mathbf{V})}{1+f^2} \qquad (109)$$

สอง successive rotation $\mathbf{R}_1$ และ $\mathbf{R}_2$ ซึ่งใช้หาที่ตั้งระบบพิกัดสามารถรวมอยู่ในรูปหนึ่ง composite rotation โดย

$$\mathbf{R} = \mathbf{R}_1 * \mathbf{R}_2 = \frac{\mathbf{R}_1 + \mathbf{R}_2 + \mathbf{R}_1\times\mathbf{R}_2}{1-\mathbf{R}_1\cdot\mathbf{R}_2} \qquad (110)$$

ซึ่ง สัญลักษณ์ * จะบอกถึงการคำเนินการของการหมุนจริงๆบนเวกเตอร์ของการหมุนที่กำหนดอยู่ในสมการข้างบน สมการ (109) และ (110) ได้มาโดยการแทน $\mathbf{q} = \mathbf{R}\cos\theta/2$ ลงในสมการ (104) และ (105) ตามลำดับ

เวกเตอร์ที่แสดงการหมุนนี้จะเกี่ยวข้องเฉพาะสามองค์ประกอบหลักมากกว่าที่เป็น สี่องค์ประหลักใน quaternion ซึ่งจะเยอะเกินไปในการแสดงของสี่องค์ประกอบหลัก มันจะกำจัดปัญหาที่แปลกๆที่เกิดขึ้นในการวิธีการเวกเตอร์ของการหมุนที่สอดคล้องกับการหมุนเพียงครึ่งหรือเต็มรอบ ($\theta = n\pi, n = 1, 2, \ldots$)

ตัวอย่างของวิธีการเวกเตอร์ของการหมุนพิจารณาหุ่นยนต์แบบสเฟอริคอล (spherical robot) ที่ถือเครื่องมือที่ end effector ให้หาพิกัดเวกเตอร์ของเครื่องมือ (tool vector coordinates) ใน base frame และทิศทางของเครื่องมือจะเปลี่ยนแปลงเมื่อมีการเคลื่อนที่ของ joint ของหุ่นยนต์

โครงสร้างของหุ่นยนต์และตัวแปร joint จะแสดงอยู่ในรูปที่ 2.24 revolute joint variables $\theta_1$ และ $\theta_2$ เป็นทิศทางของเครื่องมือที่เปลี่ยนแปลง และต่อมาพิกัดเวกเตอร์ของเครื่องมือผันแปรตาม อย่างไรก็ตาม prismatic joint variable $d_3$ จะไม่เปลี่ยนแปลงพิกัดเวกเตอร์ของเครื่องมือเมื่อมันเครื่องมือเคลื่อนขนานกับที่ตั้งในตอนแรก

การแก้ปัญหาประกอบด้วย การแสดงสองการหมุน $\theta_1$ และ $\theta_2$ เป็น single composite rotation และแปลงเวกเตอร์ของเครื่องมือจากตำแน่งเริ่มต้นไปตำแหน่งสุดท้ายโดย composite rotation นี้

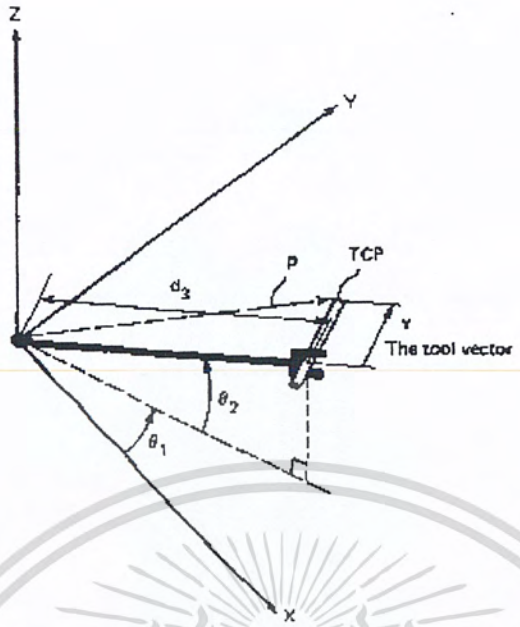
ตามรูปที่ 2.24 การหมุนครั้งแรกเป็นการหมุนรอบแกน Zใน robot base frame ดังนั้นการหมุนเวกเตอร์ครั้งแรกคือ

$$\mathbf{R}_1 = [0,\ 0,\ f_1]^{\mathrm{T}}$$

การหมุนถัดมาคือการเงยของแกน $Y$ ที่ถูกยก และสอดคล้องกันกับเวกเตอร์ของการหมุน คือ

$$\mathbf{R}_2 = [0, -f_2, 0]^{\mathrm{T}}$$

ซึ่ง $f_1 = \tan\theta_1/2$ และ $f_2 = \tan\theta_2/2$




**รูปที่ 2.24** The coordinate of a tool handle by a spherical robot [3; หน้า 123]

Composite rotation **R** หาได้จากสมการ (110)

$$\mathbf{R} = \frac{\begin{bmatrix}0\\0\\f_1\end{bmatrix} + \begin{bmatrix}0\\-f_2\\0\end{bmatrix} + \begin{bmatrix}0\\0\\f_1\end{bmatrix} \times \begin{bmatrix}0\\-f_2\\0\end{bmatrix}}{1 - \begin{bmatrix}0\\0\\f_1\end{bmatrix}\begin{bmatrix}0\\-f_2\\0\end{bmatrix}} = \begin{bmatrix}f_1 f_2\\-f_2\\f_1\end{bmatrix}$$

สำหรับการทำให้ง่ายขึ้น สมมติว่าเครื่องมือ ในรูปที่ 2.24 ในตอนแรกเริ่มอยู่ขนานกับแกน $Y$ และพิกัดเวกเตอร์ของเครื่องมือ ในตอนแรกเท่ากับ $\mathbf{v} = [0, 2, 0]^T$ ตำแหน่งในตอนแรกตัวแปร joint $\theta_1$ และ $\theta_2$ เป็นศูนย์ และที่ตำแหน่งสุดท้าย $\theta_1$ และ $\theta_2$ เป็น $90°$ และ $60°$ ตามลำดับ

ในการหาตำแหน่งสุดท้ายเครื่องมือของหุ่นยนต์ทำการหมุน **R** และเวกเตอร์ของเครื่องมือ $\mathbf{v}'$ สุดท้ายจะกลายเป็นตามสมการ (109)

$$\mathbf{v}' = \begin{bmatrix}0\\2\\0\end{bmatrix} + \frac{2\begin{bmatrix}f_1 f_2\\-f_2\\f_1\end{bmatrix} \times \left(\begin{bmatrix}0\\2\\0\end{bmatrix} + \begin{bmatrix}f_1 f_2\\-f_2\\f_1\end{bmatrix} \times \begin{bmatrix}0\\2\\0\end{bmatrix}\right)}{1 + f_1^2 f_2^2 + f_2^2 + f_1^2}$$

$$= \begin{bmatrix}0\\2\\0\end{bmatrix} + \frac{-4f_1}{\left(f_1^2 + 1\right)}\begin{bmatrix}1\\f_1\\0\end{bmatrix} \qquad (111)$$

สำหรับการหามุมของการหมุน

$$f_1 = \tan\frac{\theta_1}{2} = \tan\frac{90°}{2} = 1$$

$$f_2 = \tan\frac{\theta_2}{2} = \tan\frac{60°}{2} = \frac{1}{\sqrt{3}}$$

ข้อสังเกต ถ้า $f_2$ และ $d_3$ ในสมการ (111) จะไม่ปรากฏพร้อมกัน เพราะว่าแกนของเครื่องมือขนานกับแกนของ $\theta_2$ ของการหมุน และเพราะมันเคลื่อนที่ขนานกับ $d_3$

การหาพิกัดเวกเตอร์ของเครื่องมือที่ตำแหน่งสุดท้ายของเครื่องมือ

$$\mathbf{v} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} + \frac{-4.1}{1+1}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}$$

ผลลัพธ์เป็นเครื่องหมายแสดงเวกเตอร์ของเครื่องมือซึ่งทับกันสนิทกับแกน $Y$ และมีทิศทางในตอนนี้ชี้ไปทาง $-X$

มันจะต้องจดจำขั้นตอนของการแปลงเวกเตอร์ โดยเฉพาะพิกัดเวกเตอร์ของเครื่องมือ ใน base frame ตั้งขึ้น แต่ไม่ใช่ตำแหน่ง TCP (ดูรูปที่ 2.24) ซึ่งเป็นการง่ายในการพิจารณาได้จากสมการ (111) ซึ่ง $\mathbf{v}'$ เป็นฟังก์ชันของ $\theta_1$ เท่านั้นจะไม่ใช่ของ $\theta_2$ และ $d_3$ อย่างไรก็ตาม robot TCP จะขึ้นกับตัวแปรของ joint สามตัวแปร ในคำสั่งที่ได้มาจาก TCP ใน base frame ขั้นแรกกระทำการกับการหมุน R ที่เหมือนกันที่เกิดขึ้นบนเวกเตอร์ ซึ่งคือการวาดจาก origin ของ base frame ไปยัง TCP (เวกเตอร์ p ในรูปที่ 2.24) แล้วแทนการกระทำข้างต้นบนเวกเตอร์ของเครื่องมือ ของตัวมันเอง

## 2.3 Trajectory Interpolators [3; หน้า 127]

kinematic solution ซึ่งจะกำหนดสถานะของเอนด์ เอฟเฟกเตอร์จากสถานะที่ทราบแล้วของหลายๆแกนของการเคลื่อนที่ ดังที่ได้อธิบายไว้ในหัวข้อ 2.3 หัวข้อนี้เป็นการแก้ปัญหาของอินเวอร์ส คิเนเมติกส์ ซึ่งจะกำหนดสถานะของแกนของการเคลื่อนที่ที่ได้จากสถานะที่ให้มาของเอนด์ เอฟเฟกเตอร์ในสเปซ อินเวอร์ส คิเนเมติกส์เป็นเครื่องมือของ trajectory interpolator ของ CP-robots ในหุ่นยนต์นี้ interpolator coordinates เป็นคำสั่งการเคลื่อนที่ของแกนที่เกี่ยวข้องให้ไปยังสถานะของเอนด์ เอฟเฟกเตอร์ที่ต้องการได้ทันที

### 2.3.1 INTRODUCTION [3; หน้า 128]

คำสั่งการเคลื่อนที่ใน CP-Robots สามารถระบุได้ตามแนวแกนของหลายๆระบบพิกัดที่ใช้ ซึ่งแสดงในรูปที่2.25 WCS จะเป็นการอ้างอิงตำแหน่ง stationary base ของแมนิพิวเลเตอร์ (และมันแสดงได้เป็น absolute coordinates) TCS เป็นการอ้างอิงตำแหน่งเอนด์ เอฟเฟกเตอร์ (หรือเซนเซอร์ที่ประกอบอยู่บนข้อมือ) และ SCS เป็นการกำหนดถึงเซนเซอร์ที่ประกอบอยู่เหนือปริมาตรพื้นที่ที่ทำงานของหุ่นยนต์ ตลอดจนรูปที่ 2.25 สามารถใช้ได้กับระบบพิกัดแบบคาร์ทีเซียน (Cartesian coordinate system), ระบบพิกัดแบบไซลินดริคอล (Cylindrical coordinate system), หรือระบบพิกัดแบบสเฟอริคอล (Spherical coordinate system) ก็สามารถใช้ได้ด้วย

ในระบบหุ่นยนต์ การเคลื่อนที่จะกำหนดในระบบใดๆของ above-mentioned coordinate systems ซึ่งสัมพันธ์กับแต่ละการหมุนรอบแกนซึ่งเป็นการหมุนเพียงอย่างเดียว หรือการกระจัดเพียงอย่างเดียวตามแกนหนึ่ง

เพียงอย่างเดียวของแกนพิกัดด้วยคำสั่งของการกระจัด การกำหนดทิศทางของเอนด์ เอฟเฟกเตอร์ยังคงไม่เปลี่ยนใน associated coordinate system และกับคำสั่งของการหมุนซึ่ง absolute position ของ TCP จะไม่เปลี่ยน

**2.3.1.1 ความสำคัญของ Interpolator (Necessity of Interpolators) [3; หน้า 128]**

Trajectory interpolators จะเกี่ยวข้องในทั้งในโหมดของการสอนและโหมดการทำงานของ CP-robots การประยุกต์ใช้งานของ joint command ใน manual teaching mode และ offline programming ของ Robots จะไม่สามารถทำได้ ฉะนั้นจะใช้ WCS หรือ TCS

การใช้ระบบพิกัดที่อ้างอิงใน manual teaching mode คือ TCS ในโหมดนี้ คำสั่งของการกระจัด - การค้นหา, การยกและการกวาด – และคำสั่งของการหมุน – pitch, yaw, และ roll ( ดูรูปที่ 2.25 ) – อ้างถึงแนวทางของตำแหน่งของ TCP และทิศทางที่ไป ซึ่งเอนด์ เอฟเฟกเตอร์ชี้ไป ดังนั้นจึงเป็นการง่ายในการทำความเข้าใจโดยผู้ปฏิบัติ เมื่อ TCS และ WCS ถูกใช้ คำสั่งจะอยู่ในรูปของเวกเตอร์ความเร็วของ TCP ในหนึ่ง above-mention directions

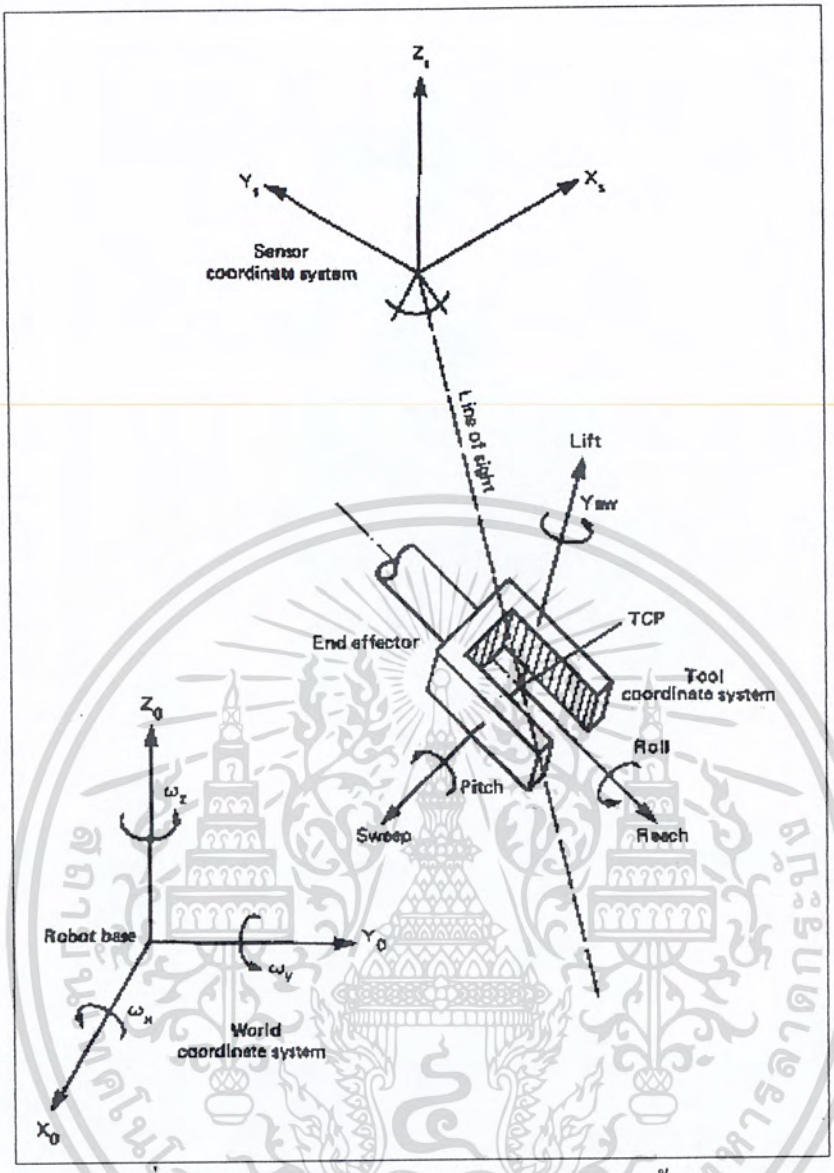
ในโหมดของการดำเนินการ Trajectory interpolators เป็นความต้องการโดยมีจุดประสงค์ คือ

1. ใน CP robots ส่วนใหญ่ในปัจจุบัน interpolators จะถูกใช้ทำให้เกิดการเคลื่อนที่ตามแนวเส้น trajectory ระหว่างการเขียนโปรแกรม absolute end point. Trajectory จะถูกใช้ในบางประโยชน์ของหุ่นยนต์เช่น การโหลดงาน, การประกอบ, การเชื่อมอาร์ค และ deburring ตำแหน่งที่ตั้งของเอนด์ เอฟเฟกเตอร์สามารถระบุอยู่ในหน้าที่ของโปรแกรม (task program) หรือโดยอย่างอื่น คือ สามารถคำนวณในเรียลไทม์ (real-time) ในระหว่างทำการดำเนินการ ตัวอย่างสำหรับกรณีหลัง ได้แก่ เมื่อกล้องโทรทัศน์ (television camera) ถูกใช้ในการตรวจสอบในเรียลไทม์ของที่ตั้งของวัตถุ และต่อมาจะสร้าง trajectory ที่สอดคล้อง ตัวอย่าง ได้แก่ palletizing operation ด้วยหุ่นยนต์แบบ CP มันเพียงพอที่จะโปรแกรมจุดหนึ่งจุดบน pallet ทิศทางของ pallet และความสัมพันธ์ของตำแหน่งของวัตถุ ซึ่งถูก arrange ขึ้นบน pallet ที่ตั้งของแต่ละ absolute position บน pallet และความสอดคล้องของ Trajectory ที่ถูกคำนวณโดยคอมพิวเตอร์ของหุ่นยนต์ ใน real-time และจะถูกแปลงโดย interpolator ไปเป็นค่าของ associate joint ในกรณีนี้ไม่จำเป็นต้องโปรแกรมทุกๆ single location บน pallet ซึ่งกรณีนี้จะเกิดขึ้นเมื่อใช้ PTP robots

2. ในหลายๆ intelligence robots Interpolator จะถูกใช้ทำให้เกิดเคลื่อนที่ตาม trajectory ที่สุ่มขึ้นมา ซึ่งเป็นการคำนวณอย่างต่อเนื่องใน real time สำหรับตัวอย่างคือ ในการ deburring operation Contact force ระหว่างชิ้นงานและเครื่องมือของหุ่นยนต์ สามารถวัดได้อย่างต่อเนื่อง โดย Tactile sensor และต่อมาจะถูกใช้ในการคำนวณระยะการกระจัดที่ต้องการของเครื่องมือ

**2.2.1.2 การสร้างคำสั่งของการเคลื่อนที่ (The Generation of Motion Commands) [3; หน้า 130]**

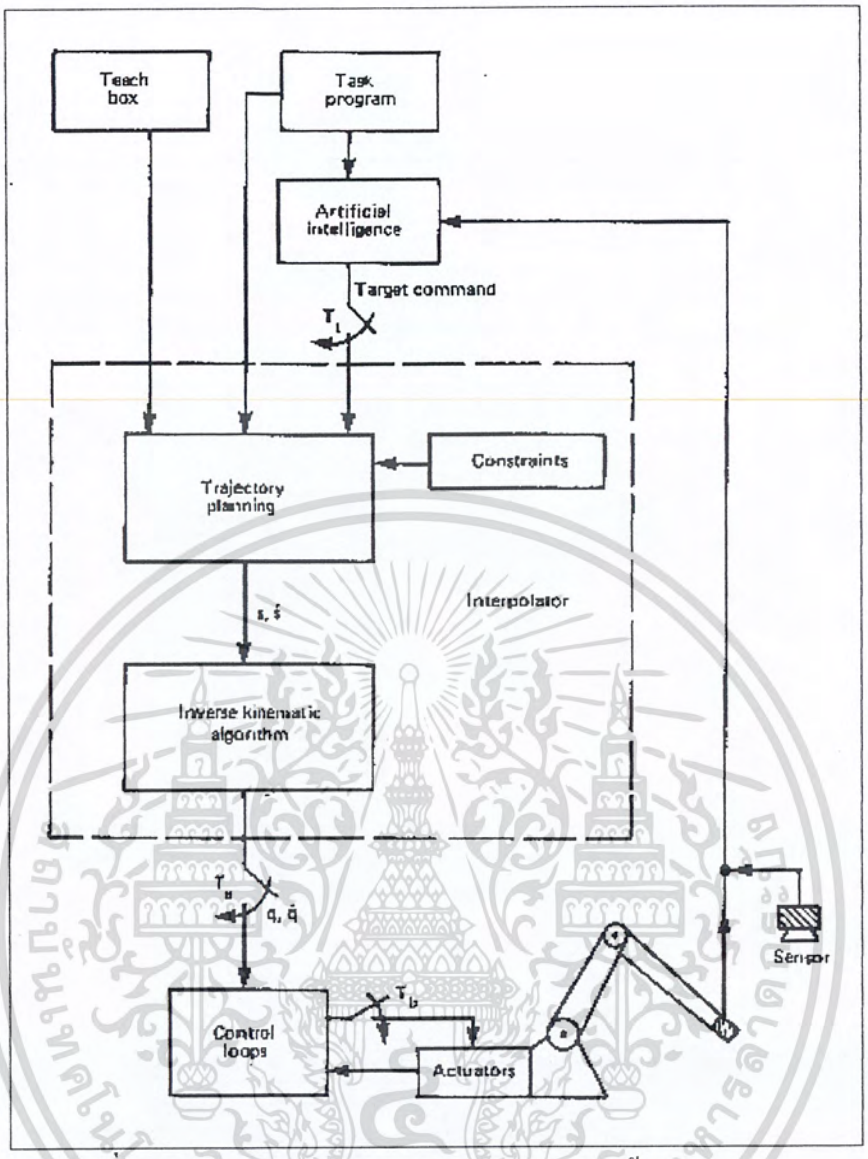
คำสั่งของการเคลื่อนที่ในหุ่นยนต์แบบ CP จะถูกสร้างขึ้นตลอดการควบคุมโดยลำดับขั้นของโครงสร้างดังแสดงในรูปที่ 2.26 ใน intelligent robots ที่ระดับสูง คือ artificial intelligence (AI) algorithm ซึ่งจะรับข้อมูลจาก associate sensor ที่ได้จาก task โปรแกรม AI algorithm จะคำนวณในเรียลไทม์ของตำแหน่งเป้าหมายประกอบด้วยตำแหน่งและทิศทางที่ต้องการของเอนด์ เอฟเฟกเตอร์ และส่งมายัง interpolator ใน nonintelligent robots ตำแหน่งเป้าหมายจะหาได้โดยตรงจาก task โปรแกรม




**รูปที่ 2.25** Useful coordinate systems in CP robot [3; หน้า 129]

ที่ระดับปานกลางคือ interpolator ซึ่งมี 2 งานหลัก : trajectory planning และการแก้อินเวอร์สคิเนเมติก อัลกอริทึ่ม trajectory planning จะกำหนดอนุกรมของ intermediate points ตาม trajectory ระหว่างที่ตั้งของ TCP ในปัจจุบันและตำแหน่งที่ต้องการจะไป จุดนี้จะต้องปราศจากการค้นหาโดยรอบของแมนนิพิวเลเตอร์ และการเคลื่อนที่โดยตลอดจะเป็น subject ของเงื่อนไขของช่องว่างของความเร็วและความเร่ง

การประยุกต์ interpolator ต่อมาจะใช้กับอัลกอริทึ่ม ของอินเวอร์สคิเนเมติก เพื่อแก้ปัญหาสำหรับตำแหน่ง ($q$) และความเร็ว ($\dot{q}$) ตามแกนที่ต้องการ ซึ่งจะส่งจุดที่สร้างขึ้นมาใหม่ไปยังลูปการควบคุม ที่ระดับต่ำสุดจะเป็นตัวควบคุมของลูปเฉพาะ ซึ่งควบคุมการเคลื่อนที่ที่สอดคล้องกับแกน 3 ระดับของการควบคุมโดยลำดับชั้นดำเนินการที่สาม different sampling intervals, $T_b$, $T_a$, และ $T_t$ ซึ่งสัมพันธ์กับ ลูปการควบคุม, interpolator, และ algorithm AI ตามลำดับ ในทุกๆหุ่นยนต์ ความสัมพันธ์ คือ $T_b \leq T_a \leq T_t$ เสมอ




**รูปที่ 2.26** Hierarchical structure of a robot system [3; หน้า 131]

สิ่งหนึ่งของการ design objectives ใน robotics คือ shorten the sampling period เป็นเหมือนกับว่าจะไกลจากความเป็นไปได้ Shorten $T_b$ จะมีความผิดพลาด (errors) ที่เกิดขึ้นตามน้อยกว่าและมี overshoot ที่น้อยกว่า Shorten $T_a$ จะอนุญาตให้มีส่วนที่เล็กกว่าเมื่อหารกับ Trajectory ซึ่งเป็นการปรับปรุง tracking ของเส้นทางที่ต้องการจะไปและเป็นการทำงานที่ดีกว่า เมื่อเวลาขึ้นกับ trajectory ใน intelligent robots Sampling period จะถูกจำกัดโดย execution time ของ algorithm อย่างไรก็ตาม software algorithm ของสามการควบคุม โดยเป็นลำดับชั้นจะเป็นอิสระที่สุด และสามารถคำนวณได้ในทางขนานได้แก่ separate computers Parallel computation จะอนุญาตให้ดำเนินการด้วย shortest sampling periods ที่เป็นไปได้เหมือนเป็นการสนับสนุนคำแนะนำของการดัดแปลงซอฟท์แวร์ (software) ในการพัฒนา

**2.3.1.3 The Trajectory Planning [3; หน้า 132]**

Trajectory Planning Algorithm แบ่งเป็นเส้นทางระหว่างส่วนๆของตำแหน่งสุดท้าย หรือระหว่างตำแหน่งปัจจุบันและตำแหน่งที่ต้องการ ในส่วนที่เล็กโดยเพิ่มจุดกลางตามแนวเส้นทางจุดเหล่านี้จะถูกส่งไปยัง Algorithm ของ inverse kinematics ซึ่งจะส่งคำสั่งที่เกี่ยวกับแกนทุกๆ $T_a$ วินาที

ตำแหน่งที่ต้องการ สามารถนิยามได้ใน 2 ทาง ดังนี้

1. โดยตำแหน่งสุดท้ายของ trajectory หาได้โดย task program หรือโดยเซนเซอร์ ซึ่งสามารถกำหนดตำแหน่งสุดท้ายที่ต้องการ นี่เป็นกรณีที่ธรรมดาสามัญที่สุด ซึ่งพบได้ในหุ่นยนต์เชิงพาณิชย์ในปัจจุบัน และในหลายๆ intelligent robots ในกรณีนี้ target command จะได้มาเมื่อเริ่มต้นทำ trajectory
2. โดยเกี่ยวกับที่ว่างของพื้นที่ของการกระจัดที่เพิ่มขึ้น นิยามโดยทิศทางและขนาดของมันเอง ในกรณีนี้ target command จะเปลี่ยนแปลงอย่างต่อเนื่อง ในลักษณะแบบสุ่มในระหว่างการทำ interpolation นี่เป็นความธรรมดาสามัญใน adaptive robots ซึ่งเซนเซอร์จะถูกใช้เป็นไกด์ให้กับหุ่นยนต์ตามแนว trajectory ที่ต้องการ ในกรณีนี้เซนเซอร์จะถูกใช้อย่างต่อเนื่องเหมือนกับเครื่องมือป้อนกลับ (feedback device) สำหรับระดับ AI level ลักษณะสำคัญที่แตกต่างกันระหว่างใน version นี้ และใน version ก่อน คือ ที่ในระดับ AI level ใช้ประโยชน์เหมือนกับตัวควบคุม ซึ่งรวมถึง overall control loop effect ของ interpolator บน loop ของการควบคุมนี้เป็นเหมือนกับ dead-time element จุดประสงค์โดยการคำนวณเวลาของ algorithm

Target commands จะต้องเป็นการแปลงในคำสั่งเคลื่อนที่ ซึ่งถูกส่งไปยังอัลกอริทึม ของอินเวอร์ส คิเนเมติก ในรูปของแต่ละเวกเตอร์ของตำแหน่งหรือเวกเตอร์ของความเร็ว สำหรับแมนิพิวเลเตอร์แบบ 6 แกนคำสั่งของตำแหน่ง มี 6 component array vector ซึ่งจะมี 3 ส่วนประกอบจะแสดงถึงตำแหน่งของ TCP และอีก 3 ส่วนประกอบจะแสดงถึงทิศทางของเอนด์ เอฟเฟกเตอร์ โดยเหมือนกันคำสั่งของความเร็ว $\dot{s}$ คือ 6 component array vector แสดงถึงอัตราการเปลี่ยนแปลงของ $s$ ทั้ง $s$ และ $\dot{s}$ จะใช้ในการแก้ปัญหาของ spatial velocity และ acceleration constraints

Velocity command vector แสดงในรูปของการกระจัดที่เพิ่มขึ้น $\Delta s$ ดังนี้

$$\Delta s(i) = \dot{s}(i) T_a \tag{112}$$

ซึ่ง $\dot{s}(i)$ เป็นค่าของ $\dot{s}$ ที่ $t = iT$ ($i$ เป็นจำนวนเต็ม) และ $\Delta s(i)$ เป็นการกระจัดที่ต้องการในระหว่างการ sampling period ข้อสังเกต $T_a$ เป็นค่าคงที่ คำสั่งของตำแหน่ง $s(i)$ ที่สอดคล้องกันจะถูกคำนวณโดยการเก็บค่าของการกระจัดที่เพิ่มขึ้น ดังนี้

$$s(i) = s(i-1) + \Delta s(i) \tag{113}$$

ซึ่ง $s(i-1)$ และ $s(i)$ เป็นเครื่องหมายของตำแหน่งก่อนหน้านี้และตำแหน่งต่อมาตามลำดับ

Trajectory planning algorithm จะคำนวณความเร่งที่ต้องการที่การเริ่มต้นของ trajectory และความหน่วงก่อนการเข้าหาตำแหน่งสุดท้าย คำสั่งของความเร็วที่เหมาะสมจะถูกออกแบบด้วยการใช้ $\dot{s}$ แทน $V$

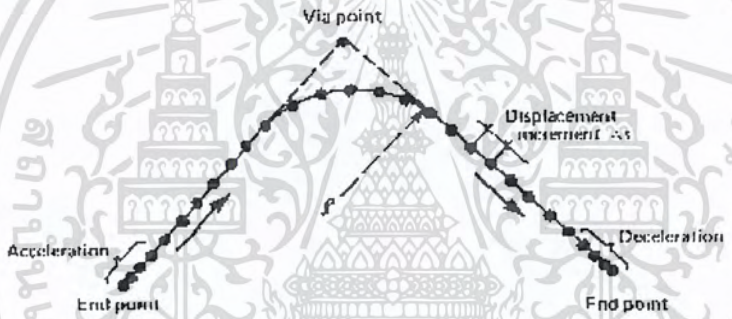
ช่วงของการเร่งสามารถกระทำได้ตามนี้ เริ่มต้นด้วย $\Delta s(0) = 0$ การเพิ่มขึ้นจะถูกคำนวณอีกครั้งในระหว่างแต่ละ sampling period $T_a$ ดังนี้

$$\Delta s(i) = \Delta s(i-1) + a_1 T_a^2 \qquad (114)$$

ซึ่ง $\Delta s(i-1)$ และ $\Delta s(i)$ เป็นเครื่องหมายของการกระจัดในตอนแรก และในตอนต่อมา และ $a_1$ เป็นความเร่งที่ต้องการตามแนวของเส้นทาง ช่วงของความเร่งจะต่อเนื่องกันจนกระทั่งความเร็ว $\dot{s}$ ที่ต้องการมีค่ามากที่สุด (ได้แก่ maximum $\Delta s$) ดังนั้น $\Delta s$ จะยังคงที่จนกระทั่งถึงช่วงของความหน่วง ซึ่งมันจะมีค่าลดลง ตามสมการ (114) ด้วยการนำความหน่วง $-a_2$ แทน $+a_1$

เมื่อเส้นโค้ง trajectory กำเนิดขึ้น ก็จะมีเงื่อนไขของความเร่งหนีศูนย์กลาง เงื่อนไขนี้เป็นจะอยู่ในรูปของรัศมีของความโค้งน้อยที่สุดที่ยินยอมให้มีได้ ซึ่งสอดคล้องกับความเร่งที่ยอมให้มีได้และขนาดของ $\dot{s}$ เมื่อต้องการเส้นโค้งของ trajectory สำหรับตัวอย่าง เมื่อจุดที่อยู่ใกล้กันถูกพิจารณา

จุดที่อยู่ใกล้กันเป็นจุดหนึ่งๆซึ่ง TCP จะผ่านไปอย่างใกล้เคียงโดยปราศจากการหยุด ความสอดคล้องกันของtrajectory จะประกอบด้วยสองเส้นตรงที่ต่อกันตลอดแนวโค้ง ดังแสดงในรูปที่ 2.27 รัศมีของความโค้งของ trajectory นี้, $\rho$, จะสอดคล้องกับความเร่งสูงสุดที่สามารถเกิดได้ของ TCP ข้อจดจำ คือ ถ้าพยายามที่จะกระทำการเปลี่ยนรูปทรงของทิศทาง จะต้องการใช้ความเร่งขนาดสูงที่ TCP ไม่สามารถทำได้ และผลลัพธ์ของการเคลื่อนที่จะไม่เป็นที่ต้องการ




รูปที่ 2.27 The use of a via point in robot trajectory [3; หน้า 133]

**2.3.1.4 Basic Structure of Interpolators [3; หน้า 134]**

ความเกี่ยวข้องกับความซับซ้อนของคณิตศาสตร์ และจำนวนของการดำเนินการทางคณิตศาสตร์ที่เกี่ยวข้องกับอัลกอริทึ่ม ของอินเวอร์ส คิเนเมติกส์ที่พิจารณาเส้นทางหลักของ interpolator ดังนั้นเทอมของ interpolator จะอ้างถึงอัลกอริทึ่มของอินเวอร์ส คิเนเมติกส์ของมันต่อไป

อินพุทของ interpolator เป็น spatial position ($s$) และ/หรือ speed ($\dot{s}$) vectors ซึ่งถูกคำนวณใน trajectory planning stage ใน interpolator เวกเตอร์เหล่านี้จะถูกแปลงเป็น associate axial position ($q$) หรือ ($\dot{q}$) ของแกนของการเคลื่อนที่ และต่อมาจะถูกใช้เป็นสัญญาณอ้างอิงของคอนโทรลลูปในแมนนิพิวเลเตอร์แบบ n-axis $q$ และ $\dot{q}$ เป็น n-component array vectors การแปลงซึ่งเป็นพื้นฐานของการแก้ปัญหาของอินเวอร์ส คิเนเมติกส์ ซึ่งจะกระทำในระหว่างแต่ละ sampling interval $T_a$

Interpolators สามารถแบ่งได้เป็น absolute หรือ incremental interpolator ตำแหน่ง $s$ จะถูกแปลงมาอยู่ใน absolute axial position $q$ The incremental interpolator จะแปลง vector $\dot{s}$ มาอยู่ใน corresponding axial speed $\dot{q}$

**Absolute Interpolators** ตัว interpolator เหล่านี้ ที่แต่ละ sampling period $i$ Absolute axis position $q(i)$ ใหม่นี้จะถูกคำนวณ ดังแสดงในรูปที่ 2.28*a* ส่วนประกอบของ $q(i)$ จะถูกส่งไปเหมือนกับเป็นตำแหน่งอ้างอิงไปยัง control loop เฉพาะ ซึ่งจะถูกคำนวณที่เวลา $t=(i+1)T_a$ ตำแหน่งจริงของแต่ละแกนจะเข้าหาตำแหน่งอ้างอิงที่สอดคล้อง

ถ้าต้องการจะหาความเร็วอ้างอิง จะคำนวณได้จาก absolute axis position ความแตกต่างของตำแหน่งในปัจจุบันกับก่อนหน้านี้ absolute axis position จะเป็นเวกเตอร์ของการกระจัด $\Delta q$ หาได้โดย

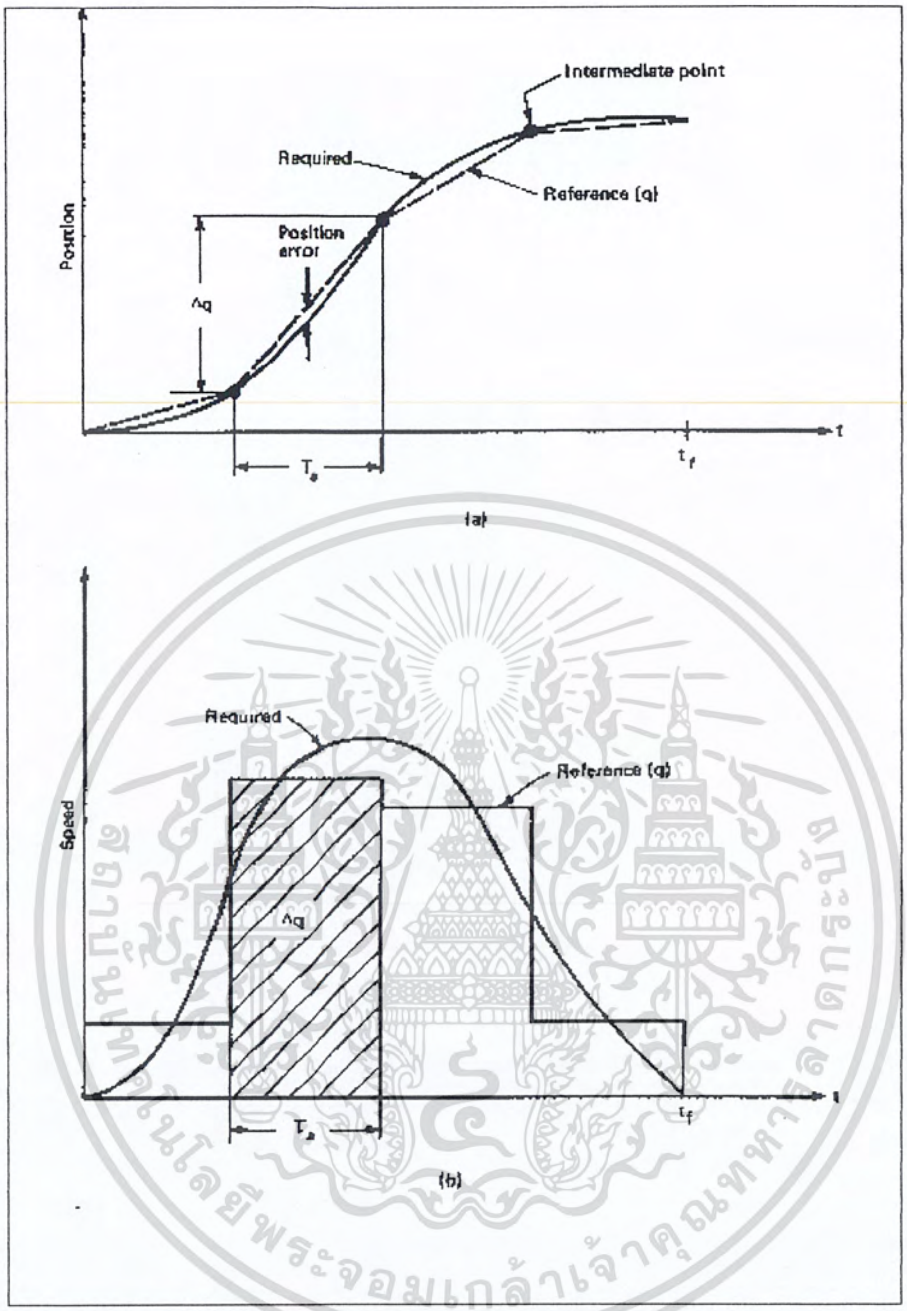
$$\Delta q(i)=q(i)-q(i-1) \tag{115}$$

axis speed vector ที่ต้องการหาจะเป็นสัดส่วนของเวกเตอร์ของการกระจัด

$$\dot{q}(i)=\frac{\Delta q(i)}{T_a} \tag{116}$$

ความเร็วอ้างอิงยังคงจะเป็นค่าคงที่ ในระหว่าง sampling period ต่อมา ดังแสดงในรูปที่ 2.28*b*

Absolute interpolator จะไม่มี reference position error ที่จุดกลาง อย่างไรก็ตามระหว่างจุดเหล่านี้จะเป็น error (ดูรูปที่ 2.27) ซึ่งขึ้นกับขนาดของ $\Delta q$ และต่อมา error จะใหญ่กว่าสำหรับ longer $T_a$ หรือสำหรับ faster speed ดังแสดงในรูปที่ 2.28*b* การเพิ่มขึ้นของ position errors จะเกิดขึ้นโดยการคำนวณ error ใน interpolator algorithm ได้แก่การตัด error และผลลัพธ์ของ error จากการประมาณทางคณิตศาสตร์อื่นๆ อย่างไรก็ตาม เมื่อการคำนวณของตำแหน่งอ้างอิงไม่มีผลโดย previous positions error เหล่านี้จะไม่ถูกเก็บสะสมในระหว่างการทำ interpolator process




**รูปที่ 2.28** The generation of (*a*) position and (*b*) speed commands by absolute interpolator [3; หน้า 135]

ความลำบากในทางคำนวณอีกอย่างหนึ่งคือ ใน absolute interpolator เกิดจากการแก้ปัญหาอินเวอร์สคิเนเมติกส์ที่ไม่ตายตัว หรืออีกนัยหนึ่ง มักจะมีหลายตำแหน่งในแนวแกนที่ทำให้เกิดจุดๆเดียวกัน แต่มีเพียงหนึ่งตำแหน่งเท่านั้นที่เหมาะสม สถานการณ์เหล่านี้อาจเกิดขึ้นได้ เช่นเมื่อมีการทำงานแบบตรีโกณมิติแบบอินเวอร์สและค่ามุมที่ได้ไม่ตายตัว กรณีตัวอย่างที่จะแสดงว่าตำแหน่งที่ตายตัวสามารถหาได้โดยโครงร่างของแขน 2 แบบที่ต่างกันนั้น การแก้ปัญหาของ interpolator จะต้องเลือกวิธีแก้ปัญหาที่เราต้องการ

**Incremental Interpolators** ด้วยเทคนิคนี้ทีแต่ละ sampling period Spatial velocity vector $\dot{s}(i)$ ถูกแปลงโดย interpolator ในแกนของ speed vector $\dot{q}(i)$ ความเร็วนี้เป็นการประมาณที่มีลักษณะรูปร่างเหมือนขั้นบันได ดังแสดงในรูปที่ 2.29*a* และมีการป้อนที่เหมือนกับตัวอ้างอิงของ control loop ตัวอ้างอิงจะยังคงคงที่

ในช่วงกลาง $T_a$ เพื่อรับแกนของตำแหน่ง Digital integration เป็นการกระทำโดยการประมาณพื้นที่ใต้เส้นโค้งของความเร็ว ซึ่งเป็นเหมือนกับผลรวมของพื้นที่สี่เหลี่ยม $\Delta q$ แต่ละตัว

$$\Delta q(i) = \dot{q}(i) T_a \tag{117}$$

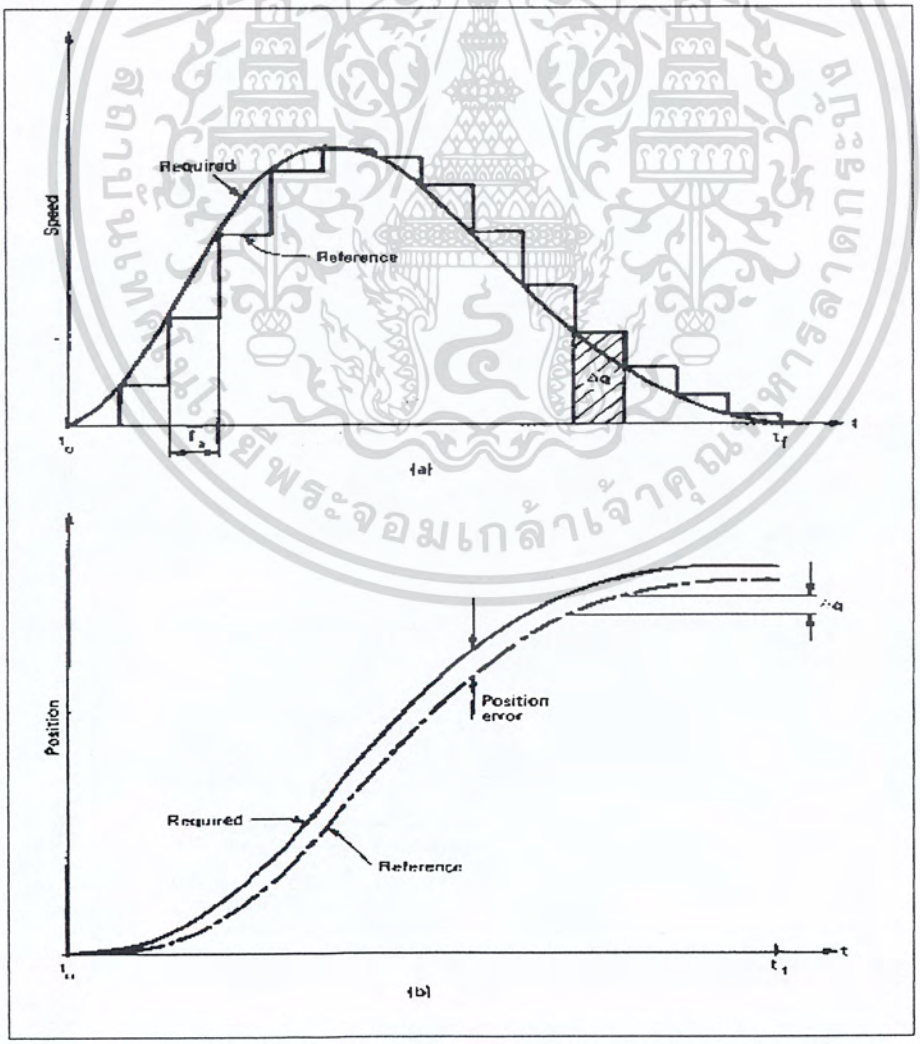
Digital integration ของสมการ (117) ให้ผลดังนี้

$$q(i) = \sum_{j=1}^{i} \Delta q(j) \tag{118}$$

แต่ละส่วนประกอบของ $q$ เป็นแกนอ้างอิงเส้นประในรูปที่ 2.29*b* แสดงถึงตำแหน่งอ้างอิง ตำแหน่งที่ต้องการสามารถแสดงได้โดย

$$q(t) = \int_0^t \dot{q}(\tau) d\tau \tag{119}$$

และแสดงโดยเส้นทึบในรูปที่ 2.29*b* ความแตกต่างกันระหว่างตำแหน่งที่ต้องการ และตำแหน่งอ้างอิง คือ axis position error การรวมกันของ axis position errors ทั้งหมดใน space เป็น trajectory-following error ซึ่ง error นี้จะสะสมในระหว่าง interpolation cycles ผลลัพธ์ใน final position error ไม่สามารถแก้ไขให้ถูกต้องได้โดย incremental interpolator




**รูปที่ 2.29** The generation of (*a*) speed and (*b*) position commands by an incremental interpolator [3; หน้า 137]

เมื่อ position errors ถูกสะสม incremental interpolator จะมีความไหวในการคำนวณ errors (ได้แก่ truncation errors) ข้อสังเกต ถ้าในช่วง sampling ถูกลดลง ในคำสั่งที่ทำ เพื่อให้บรรลุผลมากกว่า accurate integration ในสมการ (118) จำนวนของการ interpolation cycles จะเพิ่มขึ้น และต่อมาการคำนวณ errors จะมีผลมากกว่า

Above-mention position errors เป็นสาเหตุของการเพิ่มขึ้นของ trajectory – following errors ดังแสดงในรูปที่ 2.30 พิจารณาเส้นตรง trajectory จะถูกกำหนดโดยจุดเริ่มและ end point Spatial displacement increments $\Delta s$ ถูกคำนวณใน trajectory – planning level ที่แสดง spatial direction vector ตามแนวของ trajectory สมมติว่าหลังจาก first interpolation cycle Position error จะเกิดขึ้น และใน interpolation cycle ต่อมา $\Delta s$ จะถูกแสดงทิศทาง ซึ่งขนานกับ original trajectory แต่มันจะชี้ไปยังตำแหน่งสุดท้าย Procedure นี้จะเกิดอีกครั้งในระหว่าง interpolation cycles ต่อมา สาเหตุของการเพิ่มของ position errors แสดงในรูปที่ 2.30

Error นี้สามารถแก้ไขได้ใน trajectory- planning level การแก้ไขนี้จะมีอยู่ 2 step (ดูรูปที่ 2.31) ขั้นแรก actual position $s^*(i)$ ถูกคำนวณจาก axial position $q(i)$ ณ ตำแหน่งปัจจุบัน [ได้จากสมการ (118)] ใช้ direct kinematics $s^*(i)$ จะถูกเปรียบเทียบกับ spatial position $s(i+1)$ ถัดไป ก่อให้เกิดการกระจัดที่ต้องการ, $\Delta s(i+1)$ ซึ่งจุดตามตำแหน่งที่ต้องการบน trajectory (ดูรูปที่ 2.30) การกระจัดนี้จะเป็นสาเหตุให้ TCP เบนกลับเข้าหา trajectory เมื่อใช้การแก้ไขนี้แทนสำหรับ position errors โดย incremental interpolator ดังนั้นมันจะไม่สะสมเพิ่ม

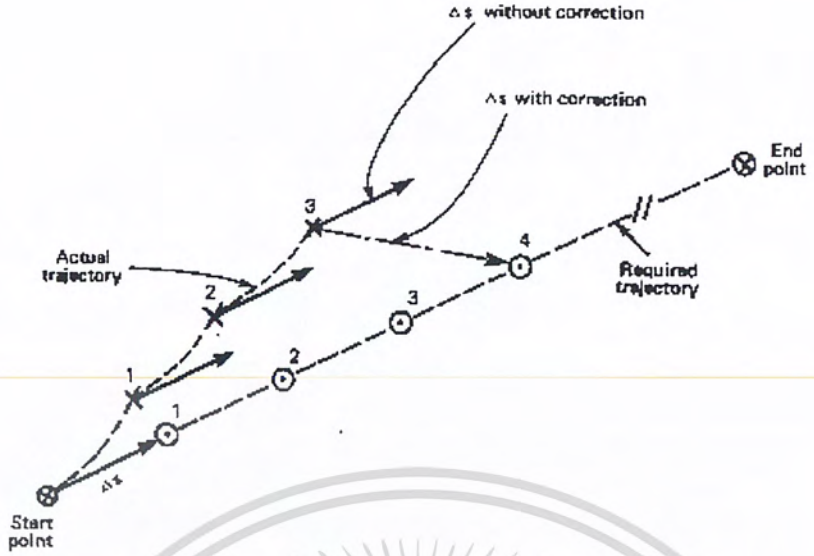
ทั้ง absolute และ incremental algorithm เป็นการลดของ spatial increment $\Delta s$ ซึ่งจะเพิ่มความแม่นยำของ positional accuracy เมื่อ $\Delta s$ เล็กกว่าด้วย traveling speed ที่ช้า [ดูสมการ(112)] ที่นั่นจะเป็นการแลกเปลี่ยนระหว่างความเร็วและ positional accuracy อย่างไรก็ตามด้วย sampling period $T_a$ ที่สั้น ความแม่นยำของ interpolation accuracy สามารถทำให้สำเร็จได้ที่ความเร็วสูง

มันจะจดจำ trajectory – following errors ที่เพิ่มโดย position errors ใน control loops และโดยความไม่เที่ยงตรงทางกล

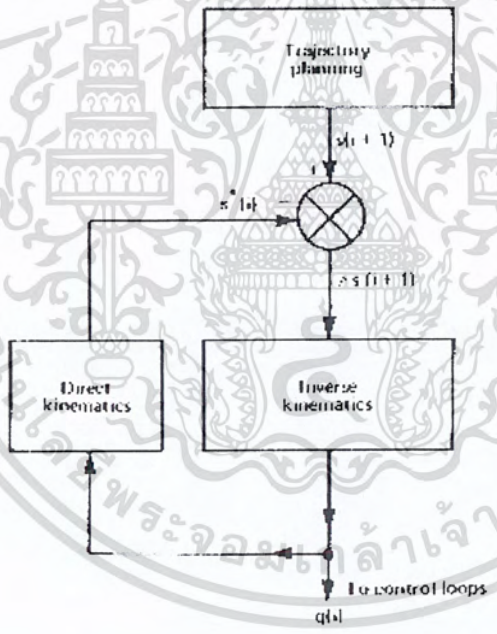
**2.3.1.5 The Solvability of the Inverse Kinematics Problem [3; หน้า 139]**

Direct Kinematics Solution ในระบบหุ่นยนต์ เป็นสิ่งที่ต้องกำหนดขึ้นเสมอ ซึ่งหมายถึงตำแหน่งและทิศทางของเอนด์ เอฟเฟกเตอร์ สามารถคำนวณได้จาก axis position vector $q$ เสมอ โดยชัดเจน ปัญหาของ inverse kinematics จะไม่สามารถแก้ไขได้ ซึ่งหมายถึงมันเป็นไปไม่ได้ที่จะให้ไปในตำแหน่งและทิศทางได้อย่างบรรลุผลด้วยแมนิพิวเลเตอร์ที่ให้มา ความสามารถที่จะไปยังตำแหน่งและทิศทางของเอนด์ เอฟเฟกเตอร์ ใน space ขึ้นกับ จำนวนและชนิดของแกนและโครงสร้างคิเนเมติกส์

ตัวอย่าง พิจารณา Spherical arm แสดงในรูปที่ 2.30 อินเวอร์ส คิเนเมติกส์ของแขนนี้สามารถหาได้โดยตำแหน่งของ TCP เพียงอย่างเดียว แต่ไม่สามารถหาทิศทางได้ มันเป็นไปได้ที่จะคำนวณหาค่าของข้อต่อ ซึ่งนำ TCP ไปยังตำแหน่งที่ต้องการใดๆ โดยปราศจากการยื่นไปรอบๆของแขน มันจะเป็นไปไม่ได้ที่จะหาทิศทาง TCP ถ้าตำแหน่งไม่ถูกกำหนด อย่างไรก็ตาม ถ้าจุดที่ 3 ของ prismatic ในรูปที่ 2.14*b* ถูกแทนโดย revolute joint (รอบแกน Z) มันก็จะกลายเป็นไปได้ในการหาทั้งทิศทางของ TCP หรือ ตำแหน่งของมัน




**รูปที่ 2.30** Trajectory following errors in incremental interpolators [3; หน้า 138]




**รูปที่ 2.31** A correction loop for trajectory errors in robots with incremental interpolators [3; หน้า 139]

เมื่อต้องการทั้งตำแหน่งและทิศทางแมนิพิวเลเตอร์ จะมี 6 แกน ที่เป็นอิสระต่อการเคลื่อนที่ ในหุ่นยนต์แบบ 6 แกนทั้งหมด ปัญหาของอินเวอร์ส คิเนเมติกส์ จะใช้ในการแก้ปัญหาโดยปราศจากปริมาตรที่เกิดจากการเอื้อมของแมนิพิวเลเตอร์ อย่างไรก็ตาม ในกรณีเฉพาะ ซึ่ง ปัญหาของอินเวอร์ส คิเนเมติกส์ ไม่สามารถแก้ได้ ตัวอย่าง เมื่อสอง revolute axes ทับกันสนิท (ได้แก่แกน 4 และ 6 ของ PUMA) ดังนั้นมันจะไม่เป็นอิสระต่อกัน ปรากฏการณ์นี้คล้ายกับเป็นความเสื่อมของ degree of freedom เป็นที่รู้จักกันว่าเป็น singularity ตำแหน่งประหลาดจะพบน้อยในแมนิพิวเลเตอร์ แบบ 4 หรือ 5 แกน แต่จะพบมากในแมนิพิวเลเตอร์แบบ 6 แกน

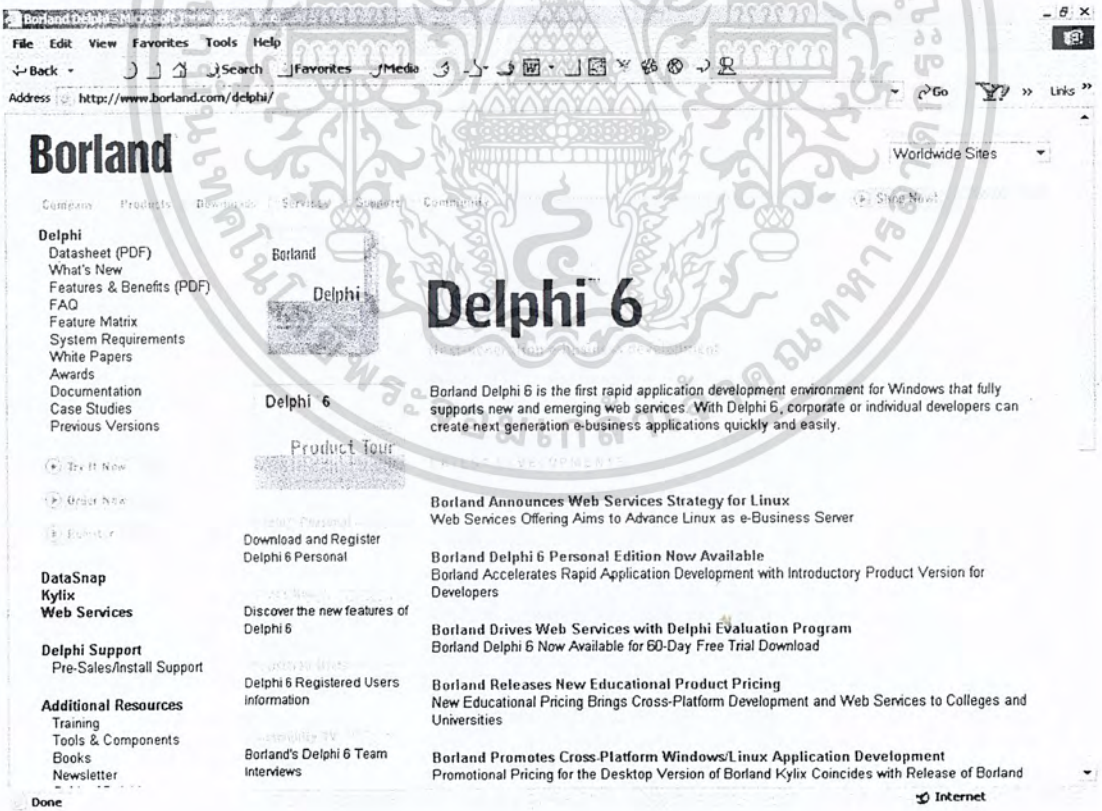
เมื่อ singularity เป็นปรากฏการณ์ในโครงสร้างคิเนเมติกส์ (kinematics configuration) มันเป็นความอิสระของเทคนิคที่ใช้แก้ปัญหาของอินเวอร์ส คิเนเมติกส์ ในหลายๆเทคนิคที่อาจแตกต่างกัน ในความสามารถที่จะกำหนดสถานภาพและนำกลับมาจากสิ่งเหล่านั้น เมื่อใดก็ตามที่แมนนิพิวเลเตอร์ เข้าใกล้ตำแหน่งประหลาด การคำนวณที่จุดนี้อาจจะมี positional errors ที่มีขนาดใหญ่ หรืออาจเป็นอย่างอื่น คือไม่สามารถทำการแปลงพิกัดได้ โดยทั่วไป incremental algorithm จะมีความไวต่อปรากฏการณ์นี้

ซึ่งการใช้ประโยชน์ของ absolute interpolator จะด้อยกว่า incremental Absolute interpolator จะไม่สามารถกระทำได้เสมอไป เพราะความซับซ้อนทางคณิตศาสตร์ที่จะนำมาเก็บค่าอินเวอร์ส คิเนเมติกส์

## 2.4 การเขียนโปรแกรมเดลไฟ 5.0 (Delphi 5.0) [1; หน้า 22]

### 2.4.1 การสร้างโปรแกรมด้วยเดลไฟ [1; หน้า 23]

เดลไฟเป็นเครื่องมือสำหรับสร้างแอพลิเคชันสำหรับรันบนวินโดว์ส (Windows) 95/98/2000 ที่ผลิตโดยบริษัท Inspire (ชื่อเดิมก็คือBorland) ซึ่งเป็นบริษัทที่แวดวงของนักพัฒนาแอพลิเคชันรู้จัก และยอมรับในตัวผลิตภัณฑ์เป็นอย่างดี




**รูปที่ 2.32** เว็บไซต์ของบริษัท Inspire

เดลไฟนั้นเป็นเครื่องมือพัฒนาแอพลิเคชันแบบวิชวลโปรแกรมมิ่ง (เหมือนกับ Visual Basic, Visual C++, PowerBuilder ฯลฯ) ซึ่งทำให้เราสามารถเห็นผลลัพธ์การทำงานไปพร้อมๆกับการลงมือสร้างแอพลิเคชัน

จุดเด่นที่สำคัญมากของความเป็นวิชวลโปรแกรมมิ่ง คือช่วยลดเวลาของการสร้างแอพลิเคชัน นั่นเพราะแทนที่เราจะไปทุ่มเวลาไปปรับแต่งส่วนติดต่อกับผู้ใช้ หรืองานที่ไม่จำเป็น หรืองานซ้ำๆซากๆ เราก็มอบภาระเหล่านี้ให้เดลไฟเสีย สำหรับเราก็มุ่งเข้าไปแก้ไขปัญหาที่เป็นหัวใจการทำงานของแอพลิเคชันดีกว่า

**ความสามารถของเดลไฟ 5.0**

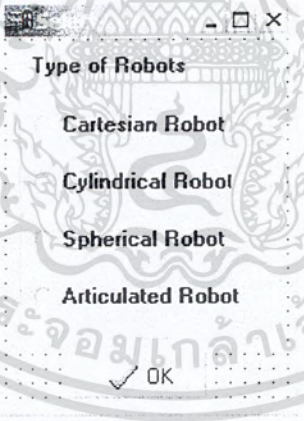
เดลไฟประกอบไปด้วยเครื่องมือช่วยในการออกแบบ, สร้าง และทดสอบแอพลิเคชันที่หลากหลาย ช่วยให้ผลงานออกมาได้อย่างรวดเร็ว, สะดวกสบาย สำหรับความสามารถของเดล 5.0 ที่น่ารู้จักได้แก่

- สร้างแอพลิเคชันจาก VCL
- เครื่องมือสร้างแอพลิเคชันที่เพียบพร้อม
- สร้างแอพลิเคชันใช้งานฐานข้อมูล
- สร้างแอพลิเคชันใช้งานร่วมกับอินเตอร์เน็ต

### 2.4.2 ขั้นตอนการสร้างแอพลิเคชันเบื้องต้นกับ Delphi 5.0 [1; หน้า 43]

**ขั้นแรก :** ออกแบบการทำงาน และหน้าตาของแอพลิเคชัน

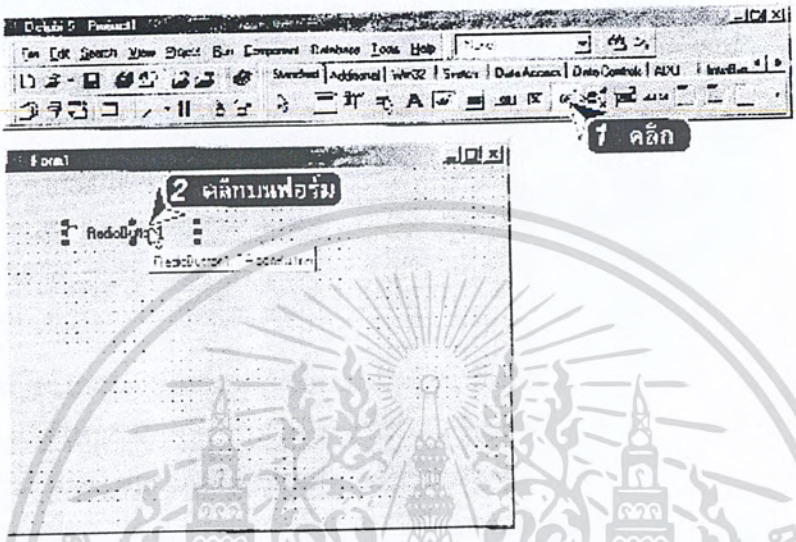
ถือเป็นขั้นตอนแรกที่ต้องทำ ต้องนึกให้ดีว่าอยากจะให้หน้าตาของแอพลิเคชันเป็นอย่างไร ถ้านึกไม่ออกหรือมีความซับซ้อน ก็วาดลงในกระดาษก่อน ดังนี้




**รูปที่ 2.33** หน้าตาของแอพลิเคชันตัวอย่าง

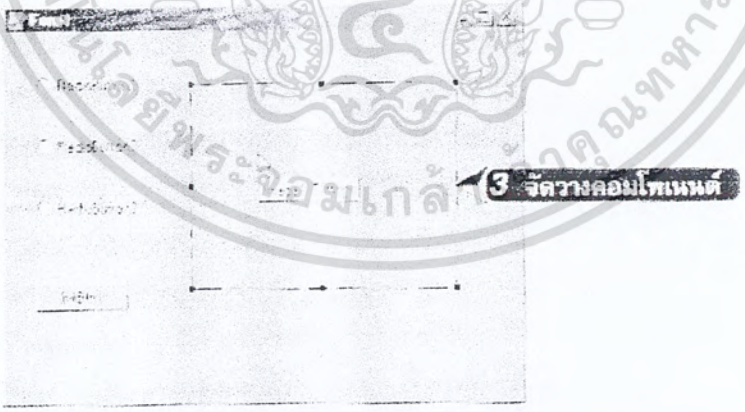
**ขั้นที่สอง :** วางคอมโพเนนต์ที่ต้องการลงในฟอร์ม

ขั้นนี้ จะนำเอาคอมโพเนนต์ต่างๆจาก Component Pallete มาวางบนฟอร์มใน Form Designer โดยให้เราคลิกเลือกคอมโพเนนต์ที่ต้องการ จากนั้นมาคลิกที่ฟอร์ม




**รูปที่ 2.34** เลือกคอมโพเนนต์จาก Component Palette

จัดวางคอมโพเนนต์ต่างๆให้อยู่ในตำแหน่งที่ต้องการสวยงาม สังเกตว่าคอมโพเนนต์ต่างๆไม่ห่างหรือไม่เบียดเสียดกันเกินไป




**รูปที่ 2.35** การจัดวางคอมโพเนนต์ใน Form Designer

**ขั้นที่สาม :** กำหนดพรอพเพอร์ตี้ให้แต่ละคอมโพเนนต์

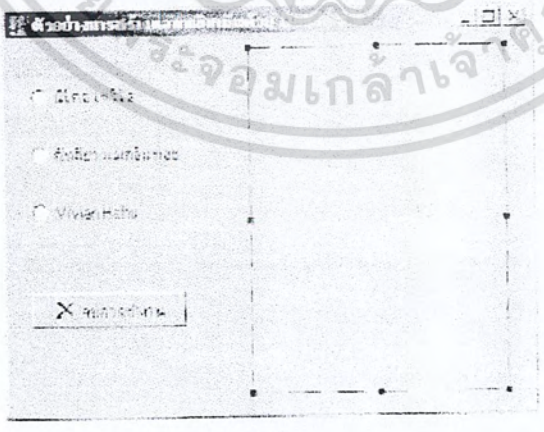
เมื่อได้ตาของแอพลิเคชันแล้ว ก็จะมากำหนดพรอพเพอร์ตี้ให้แต่ละคอมโพเนนต์ โดยใช้ Object Inspector กำหนดค่าที่ต้องการ

สำหรับพรอพเพอร์ตี้ทั้งหมดกำหนดดังนี้

| คอมโพเนนต์ | พรอพเพอร์ตี้ | ค่าที่กำหนด |
|---|---|---|
| Form | Name | FrmFirstApp |
| | Caption | ตัวอย่างการสร้างแอพลิเคชันด้วย Delphi 5.0 |
| BitBtn | Caption | จบการทำงาน |
| | Glyph | Del.bmp |
| RadioButton | Name | rbtNicole |
| | Caption | นิโคล เทรีโอ |
| RadioButton | Name | RbtCathaleeya |
| | Caption | แคทลียา แมคอินทอช |
| RadioButton | Name | RbtVivian |
| | Caption | Vivian Hsu |
| Image | Name | ImgStar |
| | Stretch | Ture |

**ตารางที่ 2.5** ตารางตัวอย่างการกำหนดพรอพเพอร์ตี้ให้แต่ละคอมโพเนนต์ของโปรแกรมเดลไฟ

หลังการกำหนดพรอพเพอร์ตี้กับคอมโพเนนต์ต่างๆ แล้วจะได้หน้าตาแอพลิเคชันดังนี้

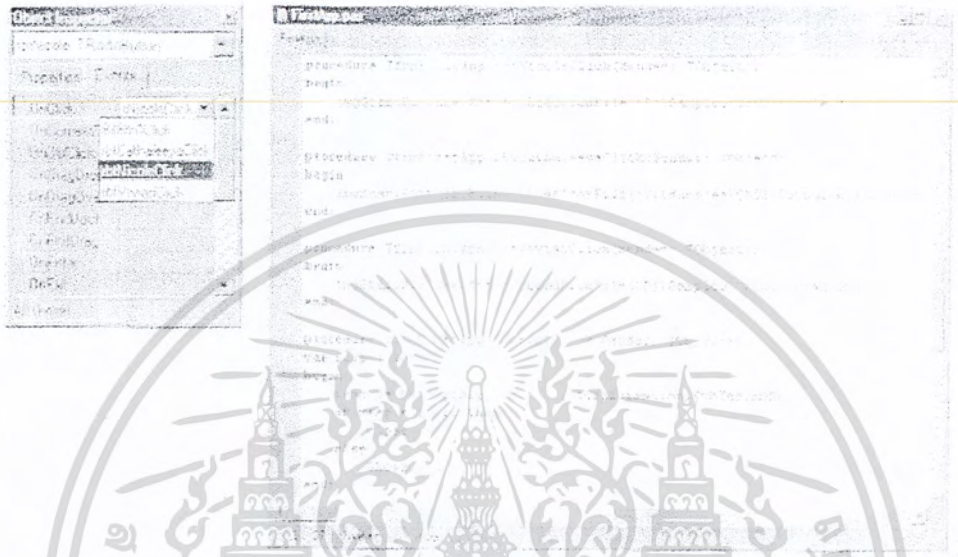


**รูปที่ 2.36** หลังการกำหนดค่าพรอพเพอร์ตี้ให้กับคอมโพเนนต์

**ขั้นที่สี่ :** เขียนคำสั่งกำกับการทำงานของแต่ละคอมโพเนนต์

เพื่อให้การทำงานร่วมกันของออบเจ็กต์ หรือคอมโพเนนต์ต่างๆเป็นไปอย่างราบรื่น จึงจำเป็นต้องขึ้นมากำกับการทำงาน โดยจะเขียนไว้ใน Code Editor

เราเรียก Code Editor มาใช้งานโดยเลือกเมนู View > Code Editor หรือใช้การดับเบิ้ลคลิกที่คอมโพเนนต์ใดๆในฟอร์ม

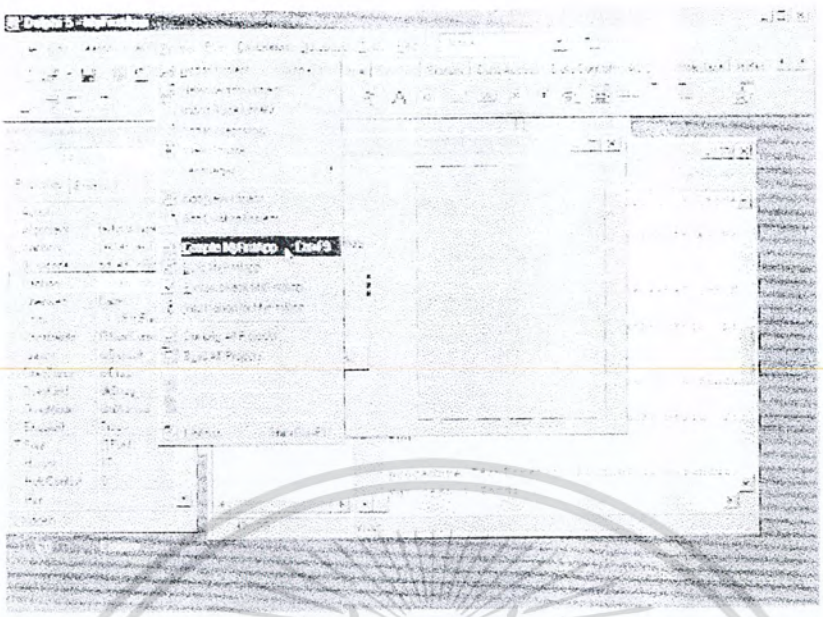


**รูปที่ 2.37** การเขียนคำสั่งภายใน Code Editor

**ขั้นที่ห้า :** ทำการคอมไพล์

เพื่อให้คอมพิวเตอร์ เข้าใจคำสั่งต่างๆที่เขียนขึ้น เราจะต้องทำการคอมไพล์ (Compile) คำสั่งที่เขียนขึ้น ซึ่งก็คือการแปลคำสั่งต่างๆที่เขียนขึ้น (Source Code) จากภาษาปาสคาล ซึ่งมนุษย์อย่างเราเข้าใจ และอ่านได้รู้เรื่อง ให้เป็นภาษาที่เครื่องคอมพิวเตอร์เข้าใจ (Object Code) และทำงานได้

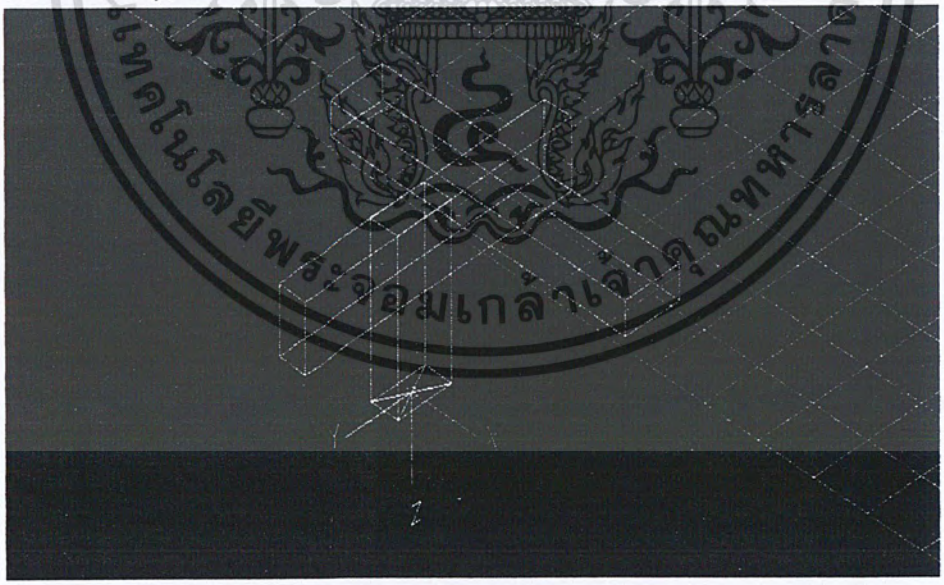
การคอมไพล์ทำได้โดยเลือกเมนู Project > Compile ชื่อโปรเจ็กต์ ในการคอมไพล์นั้น เดลฟี่จะตรวจสอบความถูกต้องของคำสั่งที่ใช้ให้ด้วย ถ้าผิดพลาดจะแจ้งให้ทราบ



**รูปที่ 2.38** การคอมไพล์แอลิเคชัน

**ขั้นที่หก :** ทดสอบการทำงานของแอพลิเคชัน

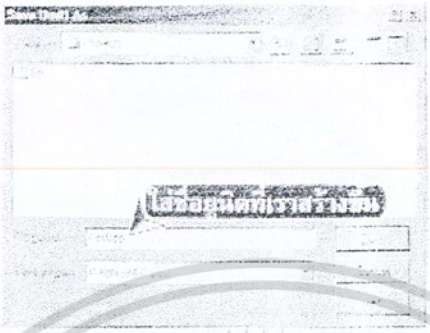
เมื่อคอมไพล์เสร็จ ขั้นตอนถัดไปคือ ทดสอบการทำงานว่ามีความถูกต้อง และตรงกับความต้องการของเราหรือไม่ โดยเลือกเมนู Run > Run ซึ่งจะเป็นการสั่งให้แอพลิเคชันที่สร้างขึ้นเริ่มทำงาน



**รูปที่ 2.39** สั่งรันแอพลิเคชัน

**ขั้นที่เจ็ด :** บันทึกแอพลิเคชันที่สร้างขึ้นเก็บไว้

เมื่อสร้างแอพลิเคชันจนแล้วเสร็จ เราสามารถบันทึกไฟล์ต่างๆเก็บไว้ เพื่อการเรียกใช้งาน หรือแก้ไขในครั้งต่อๆไปได้ ในการบันทึกจะเริ่มจากบันทึกรายละเอียดของฟอร์ม และโปรแกรมที่เราเขียนขึ้นมา โดยคลิกที่ปุ่ม Save หรือเรียกในเมนู File > Save ซึ่งไฟล์ที่บันทึกจะมีนามสกุลเป็น.PAS



**รูปที่ 2.40** บันทึกโปรแกรมที่เขียนขึ้น

จากนั้นจะบันทึกโปรเจ็กต์ ซึ่งได้ชื่อว่าเป็นตัวแทนของการเรียกใช้แอพลิคชัน โดยเรียกเมนู File > Save Project As...

**รูปที่ 2.41** บันทึกโปรเจ็ค

# บทที่ 3
# วิธีการดำเนินงาน

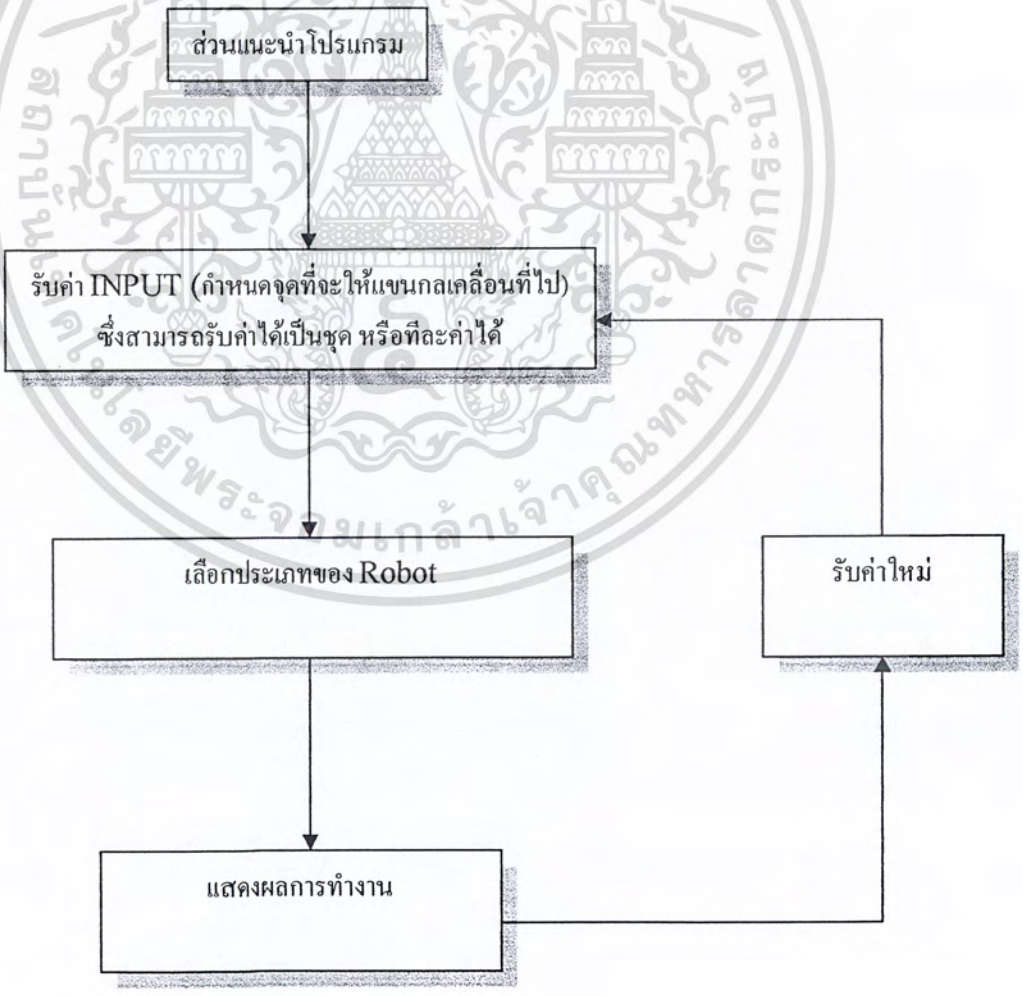
## 3.1 การออกแบบลักษณะของการดำเนินงาน

ลักษณะของตัวโปรแกรมโดยเริ่มจากรับข้อมูลจากผู้ใช้
โดยข้อมูลที่รับประกอบด้วย

- ผู้ใช้กำหนดตำแหน่งเป้าหมายที่ต้องการให้แขนกลจะไป
- เลือกประเภทของหุ่นยนต์ ซึ่งมีอยู่ 4 รูปแบบ

จากนั้นผู้ใช้ก็ทำการกดปุ่มสตาร์ทเพื่อให้โปรแกรมได้โหลดการทำงานขึ้นมา ซึ่งโปรแกรมจะทำการแสดงผลออกมาเป็นกราฟฟิกจำลองการทำงานของหุ่นยนต์ โดยจะสามารถมองเห็นได้ในทุกมุม และภาพที่แสดงผลออกมานี้จะเป็นภาพจำลองการทำงานของแขนกลของหุ่นยนต์ที่จะเข้าหาชิ้นงาน

เพื่อให้ง่ายแก่ความเข้าใจจึงทำเป็น โฟลชาร์ท (Flow Chart) ของโปรแกรมได้ดังนี้




**รูปที่ 3.1** โฟลชาร์ท ของโปรแกรม

## 3.2 วิธีการที่ใช้ในการทำโครงงาน

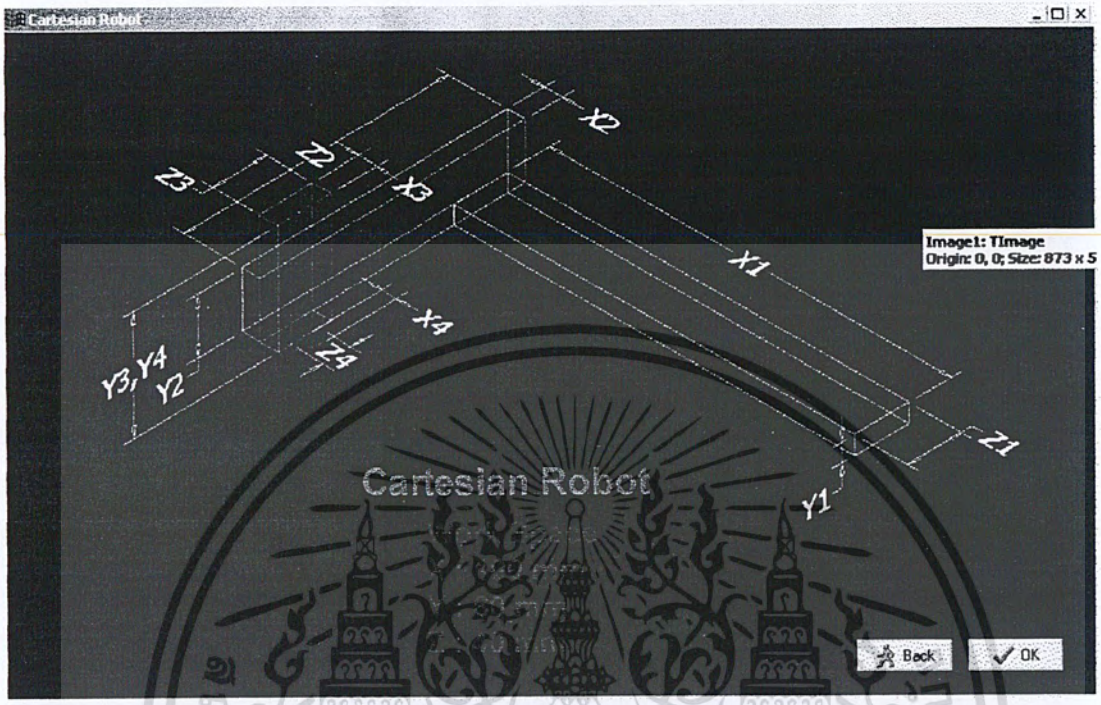
ประกอบด้วยเรื่องดังต่อไปนี้

3.2.1 ทำการศึกษาข้อมูลของหุ่นยนต์ ในเรื่องทฤษฎีของระบบหุ่นยนต์ (Robotics) จากหนังสือ ROBOTCS FOR ENGINEERNG ของ Yoram Koren ซึ่งประกอบด้วยทฤษฎี ดังนี้

1. ทฤษฎีพื้นฐานของหุ่นยนต์ (Basic Concept in Robotics) ประกอบด้วยเรื่อง ดังนี้
   - ประโยชน์และการใช้งานของหุ่นยนต์
   - โครงสร้างพื้นฐานของหุ่นยนต์
   - การจัดประเภทของหุ่นยนต์
2. การวิเคราะห์คิเนเมติกส์และการเปลี่ยนตำแหน่งโคออร์ดิเนต (Kinemaitic Analysis and Coordinate Trandformation) ซึ่งประกอบด้วยเรื่อง ดังต่อไปนี้
   - ปัญหาของ Direct Kinematics ในหุ่นยนต์อุตสาหกรรม (Direct Kinematics Problem in Robotics)
   - การวิเคราะห์ Direct Kinematics โดยใช้พื้นฐานทางเรขาคณิต (Geometry-Based Direct Kinematic Analysis)
   - การแปลงพิกัดและเวกเตอร์โดยการใช้เมตริกซ์ (Coordinate and Vector Transformations Using Matrices)
   - ข้อกำหนดของเดนาวิท-ฮาร์เทนเบิร์ก (Denavit-Hartenberg Convention)
   - การใช้ประโยชน์จากวิธีของ DH (Applications of the DH Method)
   - การแสดง Quaternion และ เวกเตอร์ของการหมุน (Quaternion and Rotation Vector Representations)
3. Trajectory Interpolators ประกอบด้วยหัวข้อ ดังต่อไปนี้
   - ความสำคัญของ Interpolator (The Necessity of Interpolators)
   - การสร้างคำสั่งของการเคลื่อนที่ (The Generation of Motion Commands)
   - The Trajectory Planning
   - Basic Structure of Interpolators
   - The Solvability of the Inverse Kinematics Problem
4. การใช้งานโปรแกรมเคลไฟ 5.0

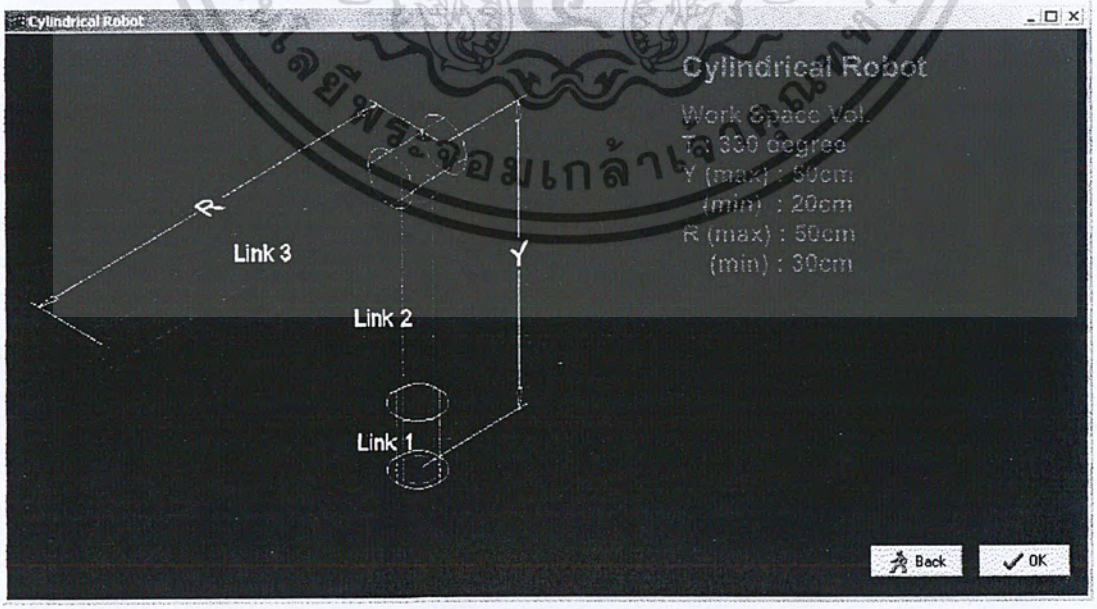
3.2.2 การเขียนโปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรม ซึ่งประกอบด้วย

3.2.2.1 การเขียนโปรแกรมของหุ่นยนต์แบบคาร์ทีเซียน (Cartesian Robot)



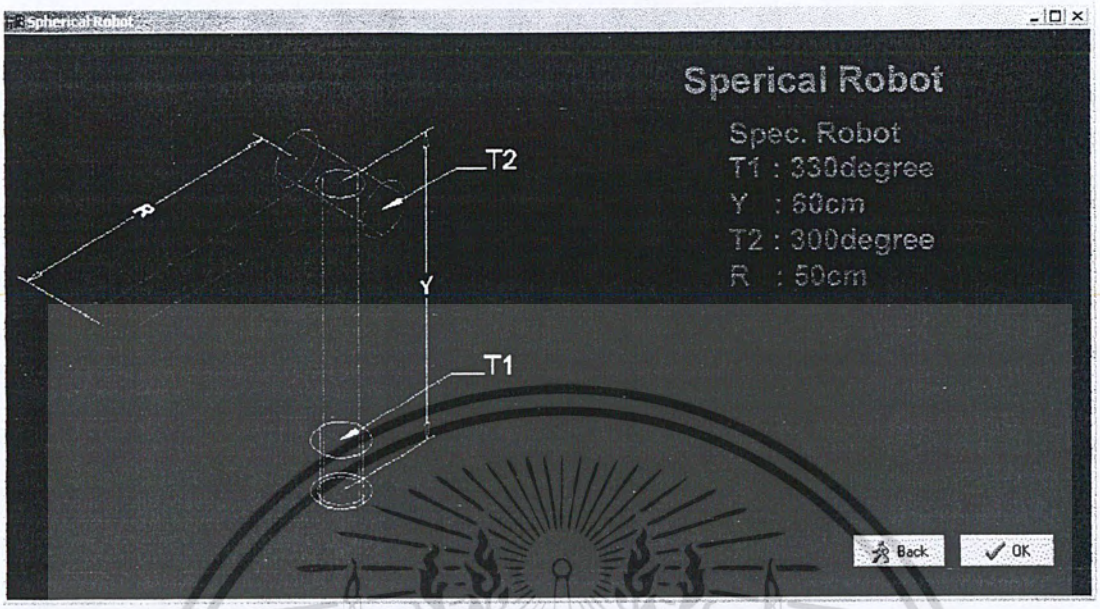

**รูปที่ 3.2** ตัวอย่างของโปรแกรมของหุ่นยนต์แบบคาร์ทีเซียน

3.2.2.2 การเขียนโปรแกรมของหุ่นยนต์แบบไซลินดริคอล (Cylindrical Robot)



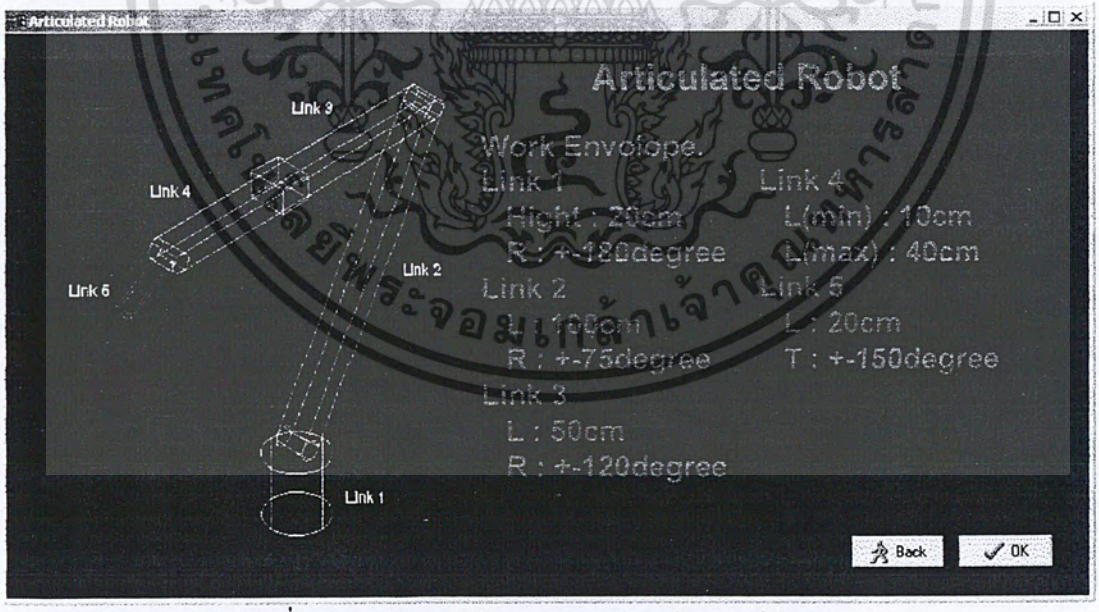

**รูปที่ 3.3** ตัวอย่างของโปรแกรมของหุ่นยนต์แบบไซลินดริคอล

3.2.2.3 การเขียนโปรแกรมของหุ่นยนต์แบบสเฟอริคอล (Spherical Robot)




**รูปที่ 3.4** ตัวอย่างของโปรแกรมของหุ่นยนต์แบบสเฟอริคอล

3.2.2.4 การเขียนโปรแกรมของหุ่นยนต์แบบอาร์ติคิวเลท (Articulated Robot)




**รูปที่ 3.5** ตัวอย่างของโปรแกรมของหุ่นยนต์แบบอาร์ติคิวเลท

3.2.3 ผลการทดสอบโปรแกรมของหุ่นยนต์ทั้ง 4 แบบ

เนื่องจากไม่สามารถทำการเขียนโปรแกรมของหุ่นยนต์ทั้ง 4 แบบ ได้เสร็จสมบูรณ์จึงไม่สามารถทำการทดสอบโปรแกรมได้

# บทที่ 4

# ผลการทดลอง

## 4.1 โครงสร้างการทดลอง

เนื่องจากไม่สามารถทำการเขียนโปรแกรมจำลองการทำงานของหุ่นยนต์ทางอุตสาหกรรมได้สำเร็จ จึงไม่สามารถทำการรันโปรแกรมได้ แต่ผู้จัดทำปริญญานิพนธ์ได้มีตัวอย่างของแนวคิดของการทำงานของโปรแกรมไว้ ดังนี้

ตัวอย่าง ต้องการจำลองการทำงานของแขนหุ่นยนต์แบบต่างๆ ซึ่งมีรายละเอียดดังนี้

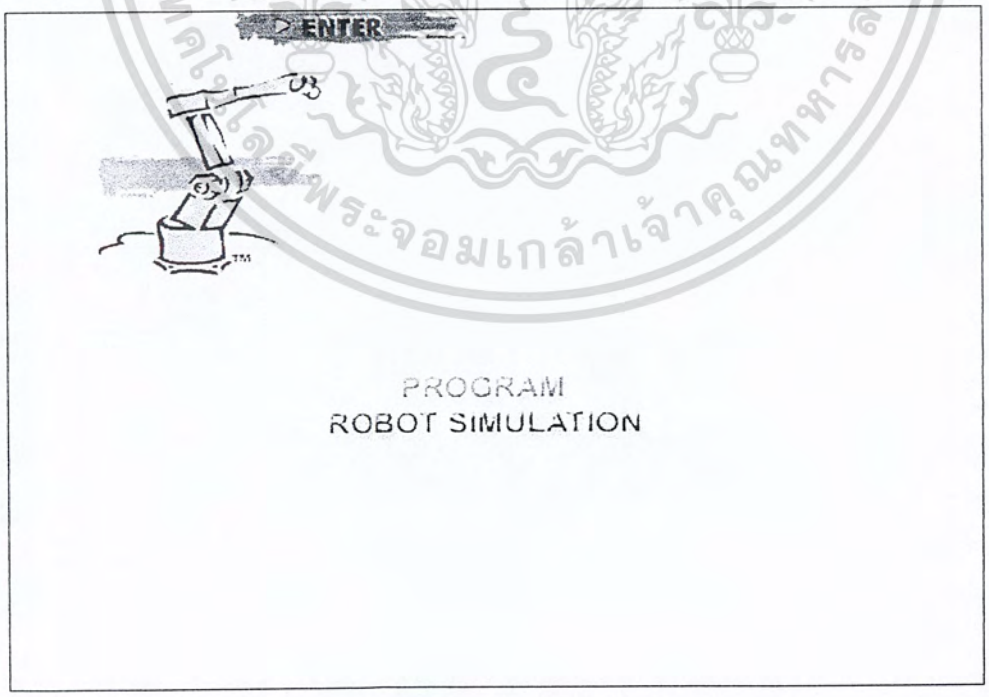
ขั้นที่ 1 กำหนดจุดหมายหรือเส้นทางที่ต้องการให้แขนของหุ่นยนต์เดินทางไป ได้แก่ การกำหนดค่า X, Y และ Z ในระนาบสามมิติ โดยมีเงื่อนไขว่าจะต้องมีค่าไม่เกินขนาดปริมาตรงาน (Space Volume) ของแขนหุ่นยนต์ที่จะสามารถเคลื่อนที่ไปได้

ขั้นที่ 2 เลือกชนิดของแขนหุ่นยนต์ที่ต้องการจำลองการทำงาน ได้แก่

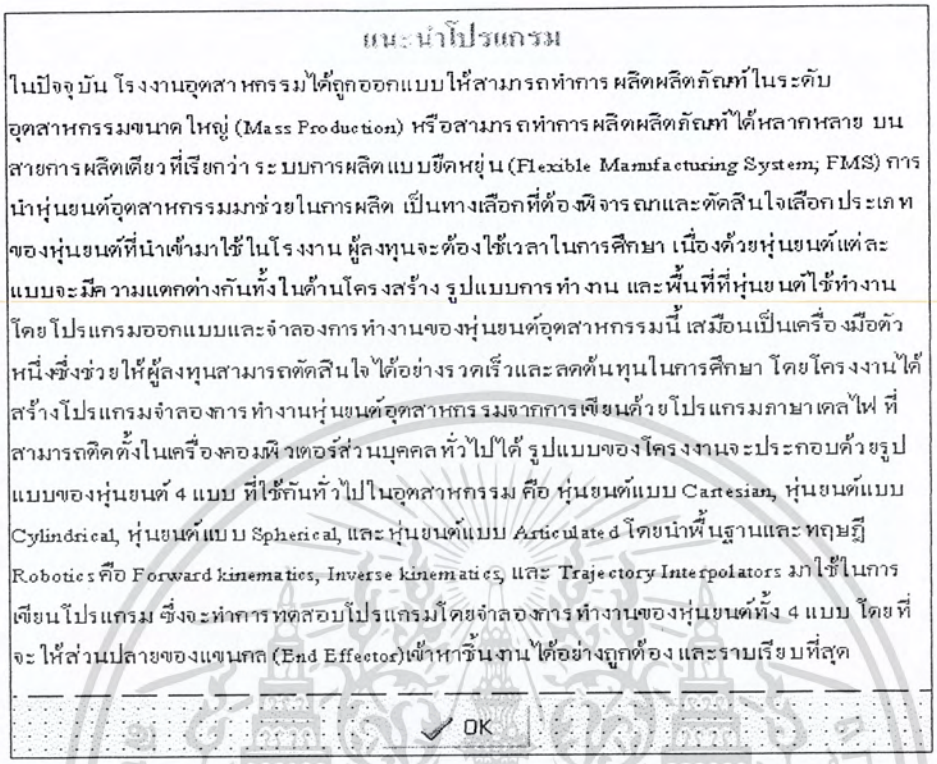
1. แขนหุ่นยนต์แบบคาร์ทีเซียน (Cartesian Robot)
2. แขนหุ่นยนต์แบบไซลินดริคอล (Cylindrical Robot)
3. แขนหุ่นยนต์แบบสเฟอริคอล (Spherical Robot)
4. แขนหุ่นยนต์แบบอาร์ติคิวเลท (Articulated Robot)

ขั้นที่ 3 ตรวจสอบการทำงานของแขนหุ่นยนต์

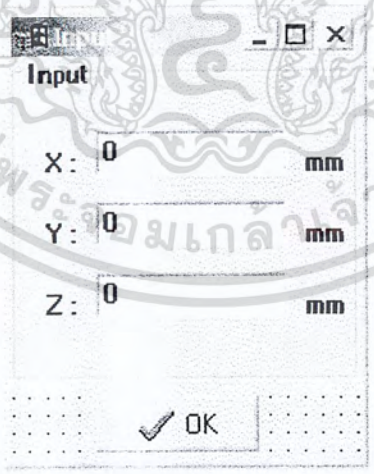
การทำงานของโปรแกรม



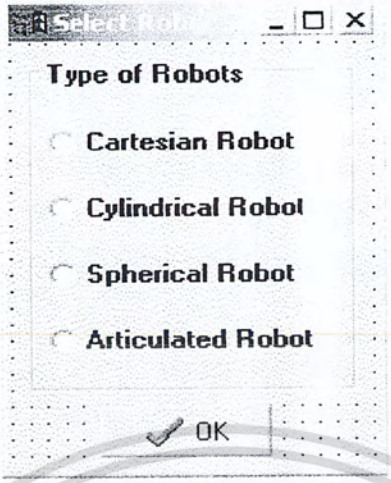

**รูปที่ 4.1** แสดงการเริ่มต้นการทำงานของโปรแกรม



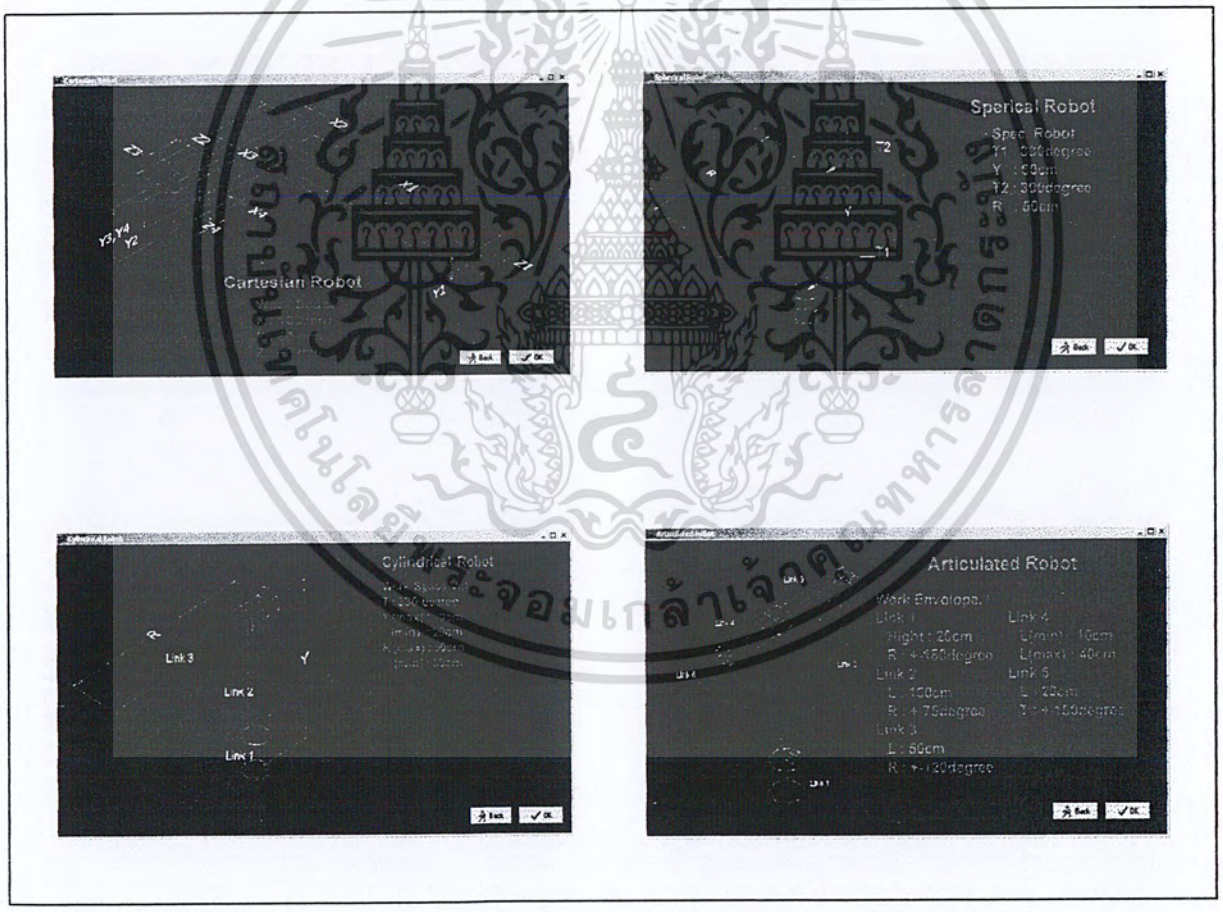

**รูปที่ 4.2** แนะนำโปรแกรม



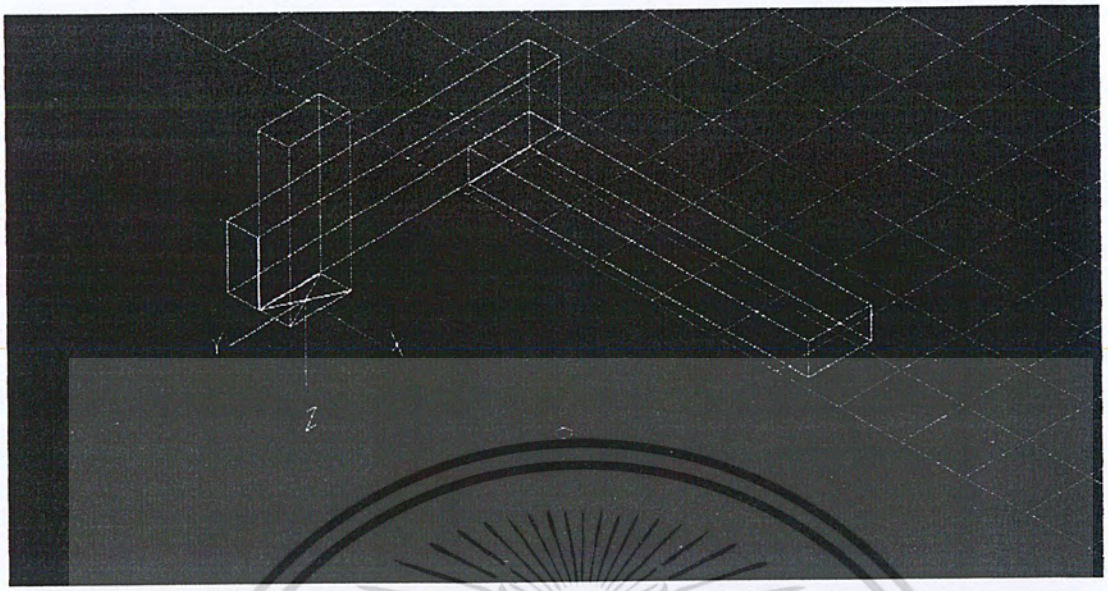

**รูปที่ 4.3** ใส่ค่า INPUT (ตำแหน่งที่ต้องการให้แขนหุ่นยนต์ไป) ซึ่งสามารถใส่ค่าเป็นชุดได้



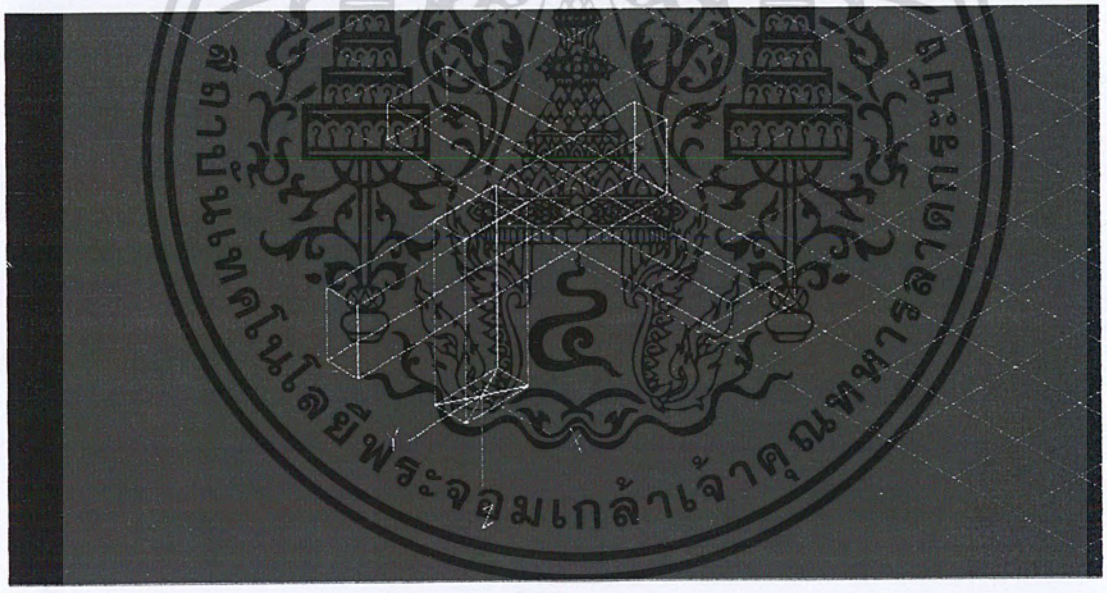

**รูปที่ 4.4** ทำการเลือกประเภทของหุ่นยนต์




**รูปที่4.5** ประเภทของแขนหุ่นยนต์ที่ทำการเลือก



**รูปที่ 4.6** แสดงการเริ่มต้นการจำลองการทำงานของแขนหุ่นยนต์ ในกรณีที่เลือกแขนหุ่นยนต์แบบคาร์ทีเซียน



**รูปที่ 4.7** แสดงการเข้าหาตำแหน่งที่ต้องการของแขนหุ่นยนต์ ในกรณีที่เลือกแขนหุ่นยนต์แบบคาร์ทีเซียน

# บทที่ 5
# วิเคราะห์และสรุป

โปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรม เป็นเครื่องมือที่ใช้และช่วยในการออกแบบและศึกษาการทำงานของหุ่นยนต์อุตสาหกรรม และยังเป็นเครื่องมือที่ช่วยในการตัดสินใจของผู้ที่จะลงทุนทำการจัดซื้อหุ่นยนต์อุตสาหกรรมเพื่อมาใช้งานในโรงงานอุตสาหกรรม โปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาห-กรรมนี้สามารถจะออกแบบ ศึกษาและพัฒนาโดยการจำลองการทำงานผ่านหน้าจอคอมพิวเตอร์ ซึ่งเป็นการประหยัดทั้งเวลาและต้นทุนในการวิจัย โดยสามารถทำให้ผู้ใช้สามารถเลือกรูปแบบของหุ่นยนต์ที่จะมาทำการจำลองการทำงานได้หลากหลายรูปแบบ และสามารถตรวจสอบการทำงาของหุ่นยนต์ได้ทุกมุมมองโดยจะเป็นประโยชน์ต่อระบบอุตสาหกรรมต่อไปในภายภาคหน้า

## 5.1 สรุปผลการทดลอง

เนื่องจากผู้จัดทำไม่สามารถทำการสร้างโปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรมได้เสร็จสมบูรณ์ จึงไม่สามารถทำสรุปผลการทดลองได้ แต่ปริญญานิพนธ์ฉบับนี้ก็เป็นแนวทางหนึ่งในการพัฒนาโปรแกรมโดยฝีมือของคนไทย ซึ่งถ้าหากมีผู้ใดสนใจก็สามารถนำปริญญานิพนธ์ฉบับนี้ไปใช้ประกอบการศึกษาและพัฒนาโปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรมได้ต่อไปในอนาคต ซึ่งแนวทางในการพัฒนาต่อไปนั้น ในส่วนของการแสดงผล นอกจากจะแสดงผลของการทำงานก็อาจจะพัฒนาให้โปรแกรมสามารถแสดงผลของการทำงานในลักษณะ wireframe และลักษณะ surface ได้ นอกจากนี้ผู้จัดทำได้ทำการสร้างโปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรมทั้ง 4 แบบ ในลักษณะที่มีรูปแบบของหุ่นยนต์เพียงรูปแบบเดียว และมีขนาด และโครงสร้างที่คงที่ ซึ่งผู้พัฒนาต่ออาจจะพัฒนาให้โปรแกรมสามารถมีรูปแบบของหุ่นยนต์ได้หลากหลายรูปแบบ และสามารถกำหนดค่าของขนาดและโครงสร้างได้ เพื่อเป็นประโยชน์ในการพัฒนาอุตสาหกรรมและประเทศชาติต่อไปได้ในภายภาคหน้า

## 5.2 ข้อดีของการใช้โปรแกรมจำลองการทำงานของหุ่นยนต์อุตสาหกรรม

1. สามารถช่วยให้ผู้ลงทุนจัดซื้อหุ่นยนต์อุตสาหกรรม สามารถตัดสินใจได้อย่างถูกต้องและรวดเร็ว
2. เป็นการประหยัดทั้งการลงทุนและเวลาในการศึกษาการทำงานของหุ่นยนต์อุตสาหกรรม
3. สามารถออกแบบและจำลองการทำงานของหุ่นยนต์อย่างเห็นได้ชัดทุกมุมมอง
4. เป็นการออกแบบและพัฒนาหุ่นยนต์อุตสาหกรรมได้ทุกรูปแบบและทุกประเภท

## 5.3 ข้อจำกัดของโปรแกรม

1. รูปแบบของหุ่นยนต์อุตสาหกรรมที่นำมาใช้ภายในโปรแกรมนี้ เป็นหุ่นยนต์ที่มีรูปแบบอย่างง่าย มีโครงสร้างไม่ซับซ้อน
2. รูปแบบของการแสดงผลเป็นรูปแบบ wier frame อย่างง่าย ไม่สามารถแสดงผลแบบ surface ได้
3. ขนาดและโครงสร้างของหุ่นยนต์เป็นแบบคงที่ไม่สามารถปรับเปลี่ยนได้

4. โปรแกรมไม่สามารถบักทึกค่าและการจำลองการทำงานหุ่นยนต์ได้

## 5.4 ข้อเสนอแนะและแนวทางการพัฒนา

1. พัฒนาให้โปรแกรมมีรูปแบบของหุ่นยนต์อุตสาหกรรมมีรูปแบบของหุ่นยนต์ได้หลากหลายรูปแบบและมีโครงสร้างการทำงานที่ซับซ้อนได้
2. พัฒนารูปแบบของการแสดงผลให้มีรูปแบบได้ทั้งแบบ wireframe และ surface
3. สามารถเปลี่ยนแปลงขนาดและโครงสร้างของหุ่นยนต์ได้
4. ทำให้โปรแกรมสามารถบันทึกข้อมูลการแสดงผล และสามารถเปิดข้อมูลเก่านำมาใช้ได้อีก
5. ทำให้โปรแกรมสามารถบันทึกค่าวิธีการคำนวณเส้นทางของแขนหุ่นยนต์ที่เคลื่อนที่ไปยังตำแหน่งที่ต้องการ และสามารถเปิดข้อมูลเก่าและพิมพ์ เพื่อใช้เป็นเอกสารอ้างอิงหรือเก็บไว้เป็นข้อมูลต่อไปได้

# เอกสารอ้างอิง

1) กมลมาศ กำจรกิจการ, คู่มือ Borland Delphi 5.0, Provision, 2543

2) สัจจะ จรัสรุ่งรวีวร, เริ่มต้นอย่างมืออาชีพด้วย Delphi 5.0, Info Press, 2543

3) Yoram Koren, Robotic for Engineers, McGRAW-HILL, 2528

4) อ.อุดม จันทร์จรัสสุข, เอกสารประกอบวิชา INDUSTRIAL ROBOTICS, 2543