Institute of Field Robotics, King Mongkut's University of Technology Thonburi
126 Pracha Uthit Rd, Bang Mot, Thung Khru, Bangkok, Thailand 10140
Senior Thesis FRAB (FIBO Robotics and Automation: Bachelor)

Path finding and position trajectory generation to avoid collision in 3 DOF Planar Revolute robot

<กันตพัฒน์ เลาหะพันธ์, 65340500001>
<ชวิศ ทรัพย์เจริญ, 65340500015>
<ธรา ลิ้มไพบูรณ์, 65340500029>

บทคัดย่อ

โครงงานนี้ถูกจัดทำขึ้นเพื่อใช้ในการศึกษาและพัฒนาการสร้างวิถีการเคลื่อนที่ (Trajectory Generation) สำหรับหุ่นยนต์ 3 DOF Planar revolute robot ที่สามารถหลบหลีกสิ่งกีดขวางภายในพื้นที่ทำงาน (Task Space) ได้ โดยการจำลองการเคลื่อนที่และลักษณะของหุ่นผ่านการใช้ Python และอัลกอริทึมที่ใช้ในการค้นหาเส้นทาง (Path finding) ผลลัพธ์ที่คาดหวังคือโปรแกรมจำลองแขนกล 3 DOF ที่สามารถเคลื่อนที่ไปยังเป้าหมายได้โดยหลีกเลี่ยงการชนสิ่งกีดขวางตามข้อกำหนดในขั้นตอนการวางแผนวิถี (Trajectory planning) การทดสอบจะดำเนินการผ่านการจำลองด้วย Pygame ที่จะแสดงให้เห็นถึงการเคลื่อนที่ในพื้นที่ทำงานที่มีสิ่งกีดขวางหลากหลายรูปแบบ ที่ใช้เป็นเกณฑ์ให้เห็นภาพในการทดสอบ

คำสำคัญ: Rapidly exploring random tree, Informed search, Trajectory generation

1. บทนำ (Introduction)

ที่มาและความสำคัญ

เนื่องจากหุ่นยนต์ด้านวิศวกรรมในงานอุตสาหกรรม มีการทำงานที่ต้องการความแม่นยำและการเคลื่อนที่อย่างมีประสิทธิภาพ นี้จึงมุ่งเน้นไปที่การสร้างความเข้าใจในกระบวนการออกแบบและพัฒนาอัลกอริทึมที่ช่วยให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งเป้าหมาย โดยลักษณะของหุ่นยนต์ที่ศึกษาในโครงงานนี้เป็นแบบ 3 DOF Planar Revolute Robot ซึ่งสามารถเคลื่อนที่เชิงมุมบริเวณข้อต่อเพื่อควบคุมตำแหน่งปลายมือ (End Effector) ให้เคลื่อนไปยังตำแหน่งเป้าหมายภายใน Task Space ได้ นอกจากนี้ยังสามารถหลบหลีกสิ่งกีดขวางในพื้นที่ทำงาน และแจ้งเตือนเมื่อเป้าหมายอยู่นอกระยะการทำงานของแขนกลได้อย่างชัดเจน การพัฒนาเทคนิคดังกล่าวจะช่วยเพิ่มขีดความสามารถของหุ่นยนต์ในการทำงานในสภาพแวดล้อมที่ซับซ้อน พร้อมทั้งเปิดโอกาสให้สามารถนำไปประยุกต์ใช้ในงานวิจัยและการออกแบบระบบอัตโนมัติในอนาคต

ในการจำลองหุ่นยนต์ได้เลือกใช้ Python และแสดงผลบน Pygameเพื่อให้สามารถปรับแต่งและพัฒนาอัลกอริทึมได้อย่างยืดหยุ่นและรวดเร็ว ผลลัพธ์ที่ได้จะช่วยสร้างความเข้าใจที่ลึกซึ้งขึ้นเกี่ยวกับการวางแผนการเคลื่อนที่ของหุ่นยนต์ และเป็นพื้นฐานสำคัญสำหรับการพัฒนาหุ่นยนต์ในงานอุตสาหกรรมและระบบอัตโนมัติต่อไป

1.1 จุดประสงค์โครงการ

1) ศึกษาการเคลื่อนที่ของแขนกลแบบ Revolute ในพื้นที่สองมิติ (2D Space)

2) ศึกษาวิธีการสร้างวิถีการเคลื่อนที่ (Trajectory Planning) ให้แขนกลสามารถเคลื่อนที่ไปยังเป้าหมายในขณะที่หลีกเลี่ยงสิ่งกีดขวาง

3) พัฒนาการค้นหาเส้นทาง (Pathfinding) และการวางแผนวิถีการเคลื่อนที่ (Trajectory Planning) ที่เหมาะสมสำหรับการหลบหลีกสิ่งกีดขวางเพื่อไปยังเป้าหมาย

1.2 ขอบเขต

1) ตัวหุ่นยนต์

- กำหนดให้ทุก ๆ ก้านของแขนกลเป็น Rigid body ที่มีเพียงความยาวเท่านั้น ไม่มีความกว้างและลึก
- แขนกลเป็น Joint แบบ Revolute จำนวน 3 Joints (RRR Robot)
- แต่ละก้านของแขนกลมีความยาวเท่ากัน
- ตำแหน่งเริ่มต้น (Home) เหมือนกันทุกการทำงาน
- พิจารณาเฉพาะ Kinematic (ไม่มีการจำลองหรือพิจารณาแรงภายนอก)
- Simulation รับ input เป็น:
  - ตำแหน่งฐานปัจจุบันของ Robot arm
  - ตำแหน่งของ End-effector ณ จุดเริ่มต้น
  - จุดพิกัดที่ต้องการให้ End-effector

2) แผนที่และสิ่งกีดขวาง

- ระบบรู้สภาพแวดล้อมทั้งหมดก่อนเริ่มทำ Path planning
- แผนที่มีขนาดเท่ากันทุกแผนที่
- ขอบของแผนที่นับเป็นสิ่งกีดขวาง
- สิ่งกีดขวางและเป้าหมายจะไม่มีการย้ายตำแหน่งขณะกำลังแสดงผล
- สิ่งกีดขวางจะแสดงผลเป็นพื้นที่สีดำบนแผนที่
- ตำแหน่งเป้าหมายจะอยู่ในพื้นที่แผนที่และไม่อยู่บนสิ่งกีดขวาง
- Simulation รับ input เป็นแผนที่สภาพแวดล้อมที่ระบุสิ่งกีดขวาง

3) การแสดงผล

- ใช้ Pygame ในการแสดงผลการเคลื่อนที่
- Output ของ Simulation เป็น:
  - พิกัด via point
  - การจำลองการเคลื่อนที่ผ่าน Pygame
  - จำลองการเคลื่อนที่ภายใน Simulation ด้วยภาษา Python

2. ทบทวนวรรณกรรม และ ทฤษฎีที่เกี่ยวข้อง (Literature Review)

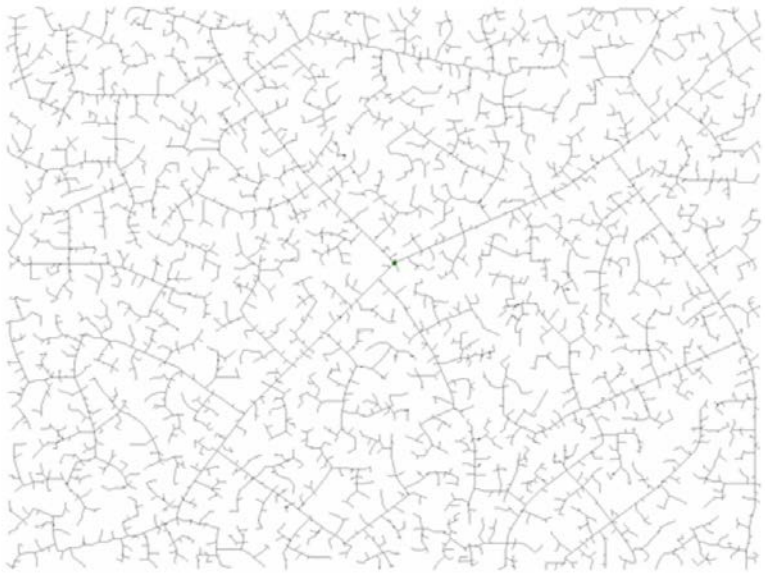
2.1 Recent advance in Rapidly – Exploring random tree: A review (Tong Xu, 2024) [1]

2.1.1 การทำ Map representation

Map representation คือการสร้างแผนที่ที่สามารถทำให้ Agent เข้าใจได้ว่าต้องเคลื่อนที่จากจุดไหนไปจุดไหน บริเวณใดมีสิ่งกีดขวาง Map สามารถสร้างขึ้นเป็นตารางที่มีขนาดเท่าไหร่ก็ได้ แต่ตารางที่ละเอียดกว่าจะส่งผลให้การเคลื่อนที่ของหุ่นยนต์ละเอียดตามไปด้วย แต่แลกกับขนาดของ Map ที่เพิ่มขึ้น การสร้างแผนที่สามารถปรับปรุงได้ด้วยการลดความละเอียดของช่องที่ไม่ได้อยู่ใกล้สิ่งกีดขวาง เก็บไว้เฉพาะพื้นที่ใกล้ขอบของสิ่งกีดขวางที่จำเป็นต้องให้ Agent เคลื่อนที่ด้วยความละเอียดที่สูงกว่าบริเวณที่ไม่มีสิ่งกีดขวาง ข้อมูลของ Map สามารถเก็บในรูป K-dimensional tree

2.1.2 Robotic Path Planning using Rapidly Exploring Random Tree

Path planning เป็นวิธีในการคิดคำนวณเพื่อวางแผนการเคลื่อนที่ของหุ่นยนต์ โดย Rapidly - Exploring Random Tree (RRT) ก็เป็นหนึ่งในอัลกอริทึมที่นิยมเพราะให้ได้เส้นทางคุณภาพสูง (ระยะทางสั้น และหรือ ใช้เวลาน้อยที่สุด) สามารถทำงานได้แบบ real-time, ทำงานในสภาวะแวดล้อมที่มีขนาดใหญ่และไม่แน่นอน ในในช่วงปี 2021 ถึง 2023 มีการพัฒนาปรับปรุง RRT เช่นการปรับปรุงกลยุทธ์การแตกกิ่ง (Branching) การปรับปรุงวิธีสร้างกราฟความเป็นไปได้ในการเคลื่อนที่ โดยการใช้โมเดลมาช่วยในการทำ RRT แต่ก็มีปัญหาเช่น การออกแบบ Hyper - Parameter การทำงานได้ในสภาวะแวดล้อมตามที่ออกแบบ (Weak generalization) พึ่งพาการทำ Localization มาก อีกทั้งยังมีปัจจัยด้านความเสถียรของฮาร์ดแวร์ในระยะยาวที่ส่งผลต่อประสิทธิภาพของ RRT ในสภาพแวดล้อมจริง โดยในอนาคต แนวโน้มการพัฒนาอัลกอริทึมอาจเน้นที่การสร้างความร่วมมือระหว่างหุ่นยนต์ประเภทต่าง ๆ กับมนุษย์ การวางแผนเส้นทางแบบเรียลไทม์ การปรับค่า hyper-parameter แบบอัตโนมัติ และการออกแบบอัลกอริทึมให้เหมาะสมกับงานหรือสภาพแวดล้อมที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง



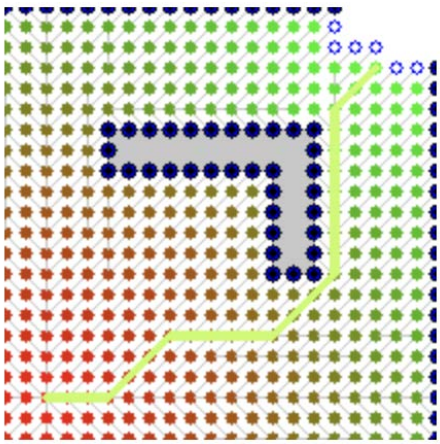
ภาพที่ 1 Path planning random tree

2.2 A Systematic Literature Review of A* Pathfinding (Daniel Foead, 2020) [2]

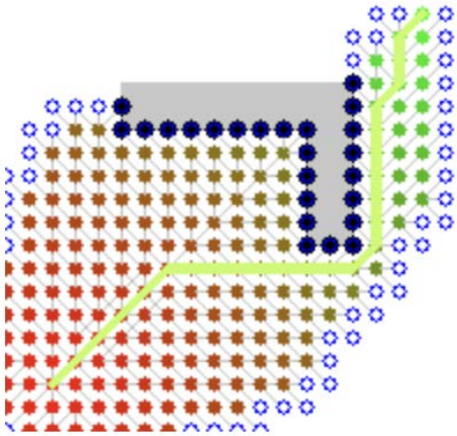
2.2.1 A* Search for Pathfinding

A* Search เป็นอัลกอริทึมที่นิยมสำหรับทำ Path finding มากและถูกพัฒนาต่อมาเรื่อย ๆ โดย A* Search เป็น Informed Searching Algorithm คืออาศัยข้อมูลเพิ่มเติมเรียกว่า Heuristic เพื่อช่วยตัดสินใจการหาเส้นทาง Heuristic ที่นิยมใช้ได้แก่ ระยะทางจาก Agent สู่ Goal แบบ Euclidean และ Manhattan ข้อดีคือสามารถให้ระยะทางที่ดีที่สุด (ใกล้หรือใช้ระยะเวลาไปถึงน้อยที่สุด) ได้ถึง 85% ของโจทย์ที่ได้รับ เพื่อปรับปรุงประสิทธิภาพ สามารถปรับเปลี่ยนการกำหนดค่า Cost ได้ หรือ Heuristic ได้ตัวอย่างคือการเพิ่มข้อมูลทิศทางเข้าไปใน Heuristic ช่วยให้เพิ่ม

ประสิทธิภาพได้ ข้อเสียของ A* Search คือใช้หน่วยความจำมาก (Space Complexity สูง) และใช้เวลาการคำนวณสูง (Time Complexity สูง) เมื่อขนาดข้อมูลใหญ่ขึ้น เวลาที่ใช้คำนวณ Heuristic ก็จะเพิ่มขึ้น นอกจากนี้หาก Agent ขาดข้อมูลเกี่ยวกับ Goal หรือแผนที่ จึงคำนวณค่า Heuristic ไม่ได้ ส่งผลให้ประสิทธิภาพในการหาเส้นทางต่ำลงไปด้วย จากเหตุการณ์ดังกล่าว Uninformed search อย่าง Dijkstra ที่ไม่ได้อาศัย Heuristic สามารถทำงานได้เร็วกว่า ประหยัดหน่วยความจำ แม้ว่าจะไม่มีข้อมูลแผนที่ก็ตาม



Dijkstra's algorithm



A*'s algorithm

2.2.2 Kinematic Modeling

- **Forward Kinematics**: ใช้ในการคำนวณตำแหน่งของตัวกระทำปลายทาง (end-effector) โดยอิงจากมุมของข้อต่อต่าง ๆ ที่กำหนดไว้ สำหรับแขนหุ่นยนต์ที่อยู่ในระนาบที่มี n ข้อต่อ แต่ละข้อต่อจะมีผลต่อการกำหนดตำแหน่งและทิศทางของตัวกระทำปลายทาง

$$P = f(\theta_1, \theta_2, \cdots, \theta_n)$$

  เมื่อ P คือ Position vector ของ End - Effector, และ θ_i คือ Joint angles.

- **Inverse Kinematics**: ใช้สำหรับคำนวณหามุมของข้อต่อต่าง ๆ ที่จำเป็นในการให้ตัวกระทำปลายทางไปถึงตำแหน่งที่ต้องการ การแก้ปัญหาจลนศาสตร์ย้อนกลับจะได้ค่ามุมข้อต่อ θ_i ที่สอดคล้องกับแต่ละจุดบนเส้นทางที่ต้องการ

2.2.3 Trajectory Planning

Trajectory planning คือ การกำหนดเส้นทางที่ End – effector จะเคลื่อนที่ไป ซึ่งมีวิธีการหลักสองแบบคือ Cartesian Space Planning และ Joint Space Planning

- **Cartesian Space Planning**: ในวิธีนี้เส้นทางจะถูกวางแผนเป็นตำแหน่งและทิศทางของ End-Effector โดยใช้การอนุมานตำแหน่งระหว่างจุดเป้าหมาย เช่น Linear Interpolation หรือ Cubic Interpolation เพื่อสร้างเส้นทางที่ราบรื่นระหว่างจุดเป้าหมาย.
- **Joint Space Planning**: ในวิธีการนี้ เส้นทางจะถูกกำหนดในแง่มุมของข้อต่อต่าง ๆ โดยใช้เทคนิค เช่น Polynomial Interpolation, Trapezoidal Velocity Profiles หรือ Cubic Splines เพื่อให้การเปลี่ยนแปลงของมุมข้อต่อเป็นไปอย่างราบรื่น

ตัวอย่างการวางแผนเส้นทางแบบ Polynomial Interpolation

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

เมื่อ $a_0, a_1, a_2, a\_0, a\_1, a\_2, a_3, a\_3$ คือ การคำนวณค่าสัมประสิทธิ์เพื่อให้สอดคล้องกับเงื่อนไขขอบเขต เช่น ตำแหน่งเริ่มต้นและตำแหน่งสุดท้ายความเร็วเริ่มต้นและสุดท้าย รวมถึงความเร่งเริ่มต้นและสุดท้าย

2.2.4 Velocity and Acceleration Profiles

- เพื่อให้การเคลื่อนไหวเป็นไปอย่างราบรื่น ควรกำหนดรูปแบบของความเร็วและความเร่งโดยวิธีนี้นิยมใช้คือ Trapezoidal Velocity Profile หรือการใช้ Cubic Spline ซึ่งช่วยป้องกันการเปลี่ยนแปลงความเร็วหรือความเร่งแบบกระทันหัน
- ความเร็วและความเร่งสามารถหาได้จากเส้นทางของตำแหน่งโดยใช้สมการนี้

$$\dot{\theta}(t) = \frac{d\theta(t)}{dt}, \qquad \ddot{\theta}(t) = \frac{d^2\theta(t)}{dt^2}$$

2.2.5 Control Algorithms

เพื่อให้หุ่นยนต์สามารถติดตามเส้นทางได้ ควรใช้ตัวควบคุม (Control Algorithm) เช่น การควบคุมแบบ PID, การควบคุมแรงบิดคำนวณ (Computed Torque Control), หรือการควบคุมแบบพยากรณ์โมเดล (Model Predictive Control - MPC)

- **PID Control**: เป็นตัวควบคุมที่อาศัย Feedback Controller เพื่อแก้ไขข้อผิดพลาดในตำแหน่งหรือความเร็ว โดยการปรับแรงบิดที่ข้อต่อหรือคำสั่งการทำงานของมอเตอร์
- **Computed Torque Control**: ใช้สำหรับคำนวณแรงบิดที่ต้องการของแต่ละข้อต่อโดยอิงจากแบบจำลองไดนามิกของหุ่นยนต์ เพื่อให้หุ่นยนต์สามารถติดตามเส้นทางได้อย่างใกล้ชิด

2.2.6 How to optimize map (Grid map to K-d Tree)

Map หรือแผนที่ คือการเก็บข้อมูลสภาพแวดล้อมที่หุ่นยนต์อยู่ หนึ่งในประเภทที่นิยมใช้คือ Occupancy grid map คือการเก็บข้อมูลในรูปแบบตาราง 2 มิติ แต่ละช่องของตารางเก็บข้อมูลว่าที่พิกัดดังกล่าวมีสิ่งกีดขวางหรือไม่ ขนาดของตารางและความละเอียดของตารางขึ้นกับขนาดของแผนที่จริงที่นำมาทำ Occupancy grid map และความละเอียดที่ใช้ ในทางทฤษฎีจะใช้หน่วยความจำขนาด $M \times N$ บิต เมื่อ M คือจำนวน Grid ในแนวตั้งและ N คือจำนวน Grid ในแนวนอน วิธีนี้ส่งผลให้ขนาดข้อมูลของ Map มีขนาดใหญ่มีผลต่อหน่วยความจำและเวลาที่ต้องใช้ Searching algorithm ในการหาเส้นทาง ดังนั้นจึงต้องทำการ Optimize เช่น การลดความละเอียดของ Map การรวมช่องที่ต้องการความละเอียดน้อยเช่นบริเวณที่ไกลจากสิ่งกีดขวาง การรวมสิ่งกีดขวางที่ใกล้กันมากๆ จน Agent ไม่สามารถเคลื่อนที่ผ่านไปได้ เข้าด้วยกัน หรือการเก็บข้อมูล map ที่ Optimize ในรูปแบบ K-dimensional tree ภายใน K-dimensional tree เป็นข้อมูลแบบ Binary tree ซึ่งแต่ละ Node จะแบ่งครึ่ง map ลงไปเรื่อย ๆ จนไม่สามารถแบ่งได้ เช่นภายในไม่มีสิ่งกีดขวางอยู่ map จุดนั้นจึงสามารถรวมกันเป็น Grid ที่มีขนาดใหญ่กว่าเพื่อประหยัดหน่วยความจำได้

2.2.7 How A* work

A* เป็น algorithm การหาเส้นทางโดยใช้ข้อมูลช่วยที่เรียกว่า Heuristic มาช่วยในการตัดสินใจเลือกเส้นทาง Algorithm จะเริ่มจาก Grid ที่ Agent อยู่ แล้วเคลื่อนที่ไป Grid ต่อไปที่มีค่า F(...) (ค่า Heuristic รวมกับ Cost) น้อยที่สุด หากมี Grid ที่มีค่า F(...) น้อยกว่า Algorithm จะกลับไปเริ่มหาจาก Grid นั้นก่อน เส้นทางที่ได้คือเส้นทางที่มีระยะทางน้อยที่สุด ข้อดีของ A* search คือไม่จำเป็นต้องตรวจสอบทุกๆ ความเป็นไปได้ ส่งผลให้มีความเร็วมากกว่า แต่ข้อเสียคือ หาก Map มีการเคลื่อนที่ตลอดเวลา หรือ Agent ไม่มีข้อมูลเกี่ยวกับ Map ครบถ้วน มักจะส่งผลให้คำนวณ Heuristic ผิดพลาด เส้นทางที่ได้จึงไม่ได้สั้นที่สุด อีกข้อเสียหนึ่งคือหากขนาด Map ใหญ่จะทำให้ต้องใช้หน่วยความจำเพื่อเก็บข้อมูลมากขึ้นและใช้เวลาคำนวณ Heuristic นานขึ้น

3. เนื้อหาในรายวิชาที่เกี่ยวข้อง

1) Transformation of coordinate frame

2) Forward Position Kinematics

3) Inverse Position Kinematics

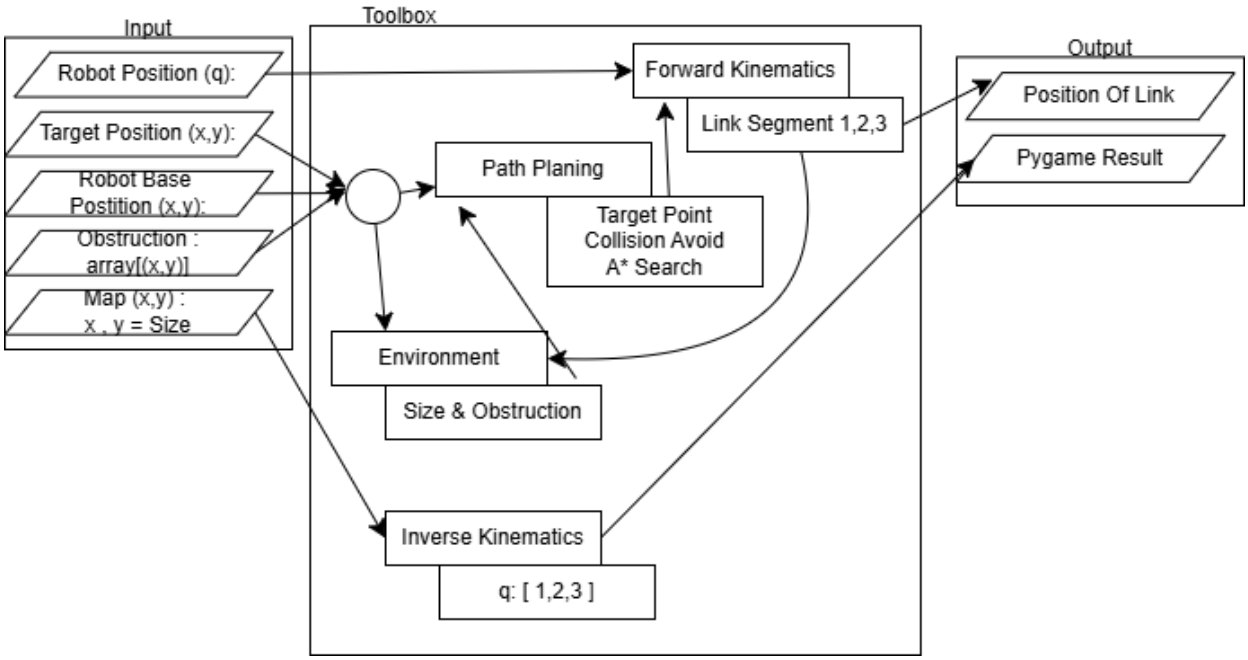
4) Forward Velocity Kinematics

5) Inverse Velocity Kinematics

6) Trajectory generation

7) Path finding & Searching algorithm (วิชา AI ใน Image Processing)

4. System Diagram / System Overview (Function and Argument)



4.1 Input

4.1.1 Robot Pose

เป็นการระบุตำแหน่งและมุมองศาของข้อต่อต่างๆของหุ่นยนต์ในปัจจุบันซึ่งแสดงให้เห็นถึงตำแหน่งและทิศทางของการจัดวางแต่ละจุดเชื่อมต่อของแขนหุ่นยนต์ผ่านตัวแปรที่ใช้ในการเคลื่อนที่แต่ละข้อ (q) ข้อมูลนี้มีความสำคัญในการคำนวณและควบคุมการเคลื่อนที่ของหุ่นยนต์ให้บรรลุเป้าหมายที่ต้องการโดยการนำตำแหน่งของข้อต่อเหล่านี้มาใช้ในการวางแผนการเคลื่อนที่เพื่อให้หลีกเลี่ยงสิ่งกีดขวางในพื้นที่ได้อย่างมีประสิทธิภาพ

4.1.2 Target Position

ตำแหน่งเป้าหมายที่แขนต้องเคลื่อนที่ไป

4.1.3 Robot Base Position

ตำแหน่งที่จุดปลายมือของหุ่นยนต์ ใช้คำนวณหา Invert Kinematics

4.1.4 Obstruction

สิ่งกีดขวางที่อยู่ใน Task space ของหุ่นยนต์ มีผลต่อการวางแผนการเคลื่อนที่ของหุ่นยนต์ โดยจะกำหนดตำแหน่งของสิ่งกีดขวางเพื่อให้ระบบควบคุมสามารถสร้างเส้นทางการหลีกเลี่ยงสิ่งกีดขวางได้อย่างมีประสิทธิภาพ

4.1.5 Map

ขอบเขตและขนาดของพื้นที่ทำงานทั้งหมดของหุ่นยนต์

4.2 Toolbox

ฟังก์ชันสำหรับการหา Forward Kinematics สำหรับ 3 Links เป็นฟังก์ชันที่จะรับค่า Q ของ Actuator และ Segment เพื่อนำไปคำนวณหลังจากได้ค่า Position X-Y จะนำค่าไปใช้เป็น Target Position ของ Path planning

4.2.1 Environment

สำหรับรับค่าขนาดพื้นที่ และจำลองส่งกีดขวางเพื่อจำลองสภาวะการเคลื่อนที่

4.2.2 Searching Algorithm

เป็นวิธีการหาเส้นทางการเคลื่อนที่ของหุ่นยนต์พร้อมการหลบสิ่งกีดขวาง โดยรับค่า Input มาจาก Environment ก่อนส่งค่าไปให้ Path planning

4.2.3 Path Planning

จะรับค่า Position X-Y ที่ได้มาจาก Forward Kinematics และวิถีการเดิน เพื่อนำมาจำลองการเคลื่อนที่ของหุ่นยนต์ และส่งค่าองศาที่เปลี่ยนแปลงไปให้ Invert Kinematics เพื่อหาค่า Q

4.2.4 Inverse Kinematics

เป็นฟังก์ชันสำหรับการหา Inverse Kinematics ของทั้ง 3 Links โดยจะนำลักษณะของ Closed Loop Kinematic Chain ที่ได้รับมาจากค่า Input จาก Path Planning และนำค่า Q ที่หุ่นยนต์สามารถไปได้เข้าไปใน Pygame เพื่อจำลองออดมา

4.3 Output

4.3.1 Trajectory

คือตำแหน่งของค่า Q ที่จะหุ่นยนต์จะมีการเปลี่ยนแปลงแล้วนำไปเช็คกับการจำลองว่าค่า Q ที่ใช้เป็นเท่าไหร่

4.3.2 Pygame Animation

เป็น Output จำลองการเคลื่อนที่ของหุ่นยนต์ที่ได้มาขจาก Toolbox

5. ผลการศึกษาที่คาดหวัง

1) สร้างโปรแกรมที่จำลอง Revolute 3 DOF Planar Robot Arm

2) แขนกลสามารถเคลื่อนที่ไปยังเป้าหมายได้โดยไม่ชนสิ่งกีดขวาง (หากจุดนั้นเป็นจุดที่หุ่นยนต์สามารถเอื้อมไปถึงได้)

3) การเคลื่อนที่ของแขนกลต้องเป็นไปตามข้อจำกัดของสนามที่กำหนดในขั้นตอน Trajectory generation

4) โปรแกรมสามารถส่ง Feedback กลับมาได้หากไม่มีทางที่สามารถไปถึงเป้าหมายได้

6. รายละเอียดโครงการ

ลำดับ	การดำเนินงาน	สัปดาห์ที่ 1	สัปดาห์ที่ 2	สัปดาห์ที่ 3	สัปดาห์ที่ 4	สัปดาห์ที่ 5	สัปดาห์ที่ 6
1	ศึกษาหลักการทำงานและวิธีการทำ	■	■				
2	จัดทำ Proposal	■					
3	ทำ Toolbox ส่วนของ FK และ Path planning		■	■	■		
4	ทำ IK และ Environment				■	■	
5	ทำ Searching Algorithm				■	■	
6	แก้ไขและส่งงาน						■

7. เอกสารอ้างอิง (References)

[1] Tong Xu. “Recent advance in Rapidly – Exploring random tree: A review” (2024).

[2] Daniel Foead. “A Systematic Literature Review of A* Pathfinding” (2020)