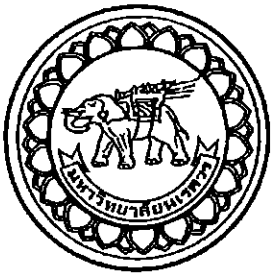


การออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์โดยอาศัยหลักการเวฟเฟรอนของคลื่น

DEVELOPMENT OF PATH PLANNING ALGORITHM USING WAVEFRONT PROPAGATION APPROACH

นายนัฐวัช สุขรวย รหัส52362724

นายวิศรุต พัฒนพงษ์ รหัส 52362922

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์มหาวิทยาลัยนเรศวร

ปีการศึกษา2555

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน	**การออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์โดยอาศัยหลักการแพร่ของคลื่น**		
ผู้ดำเนินโครงงาน	นายนัฐวัช	สุขรวย	รหัส 52362724
	นายวิศรุต	พัฒนพงษ์	รหัส 52362922
ที่ปรึกษาโครงงาน	ดร.พนัส นัถฤทธิ์		
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2555		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

..... ที่ปรึกษาโครงงาน
(ดร.พนัส นัถฤทธิ์)

..... กรรมการ
(ดร.สุรเดช จิตประไพกุลศาล)

..... กรรมการ
(อาจารย์รัฐภูมิ วรานุสาสน์)

..... กรรมการ
(อาจารย์เศรษฐา ตั้งค้าวานิช)

ชื่อหัวข้อโครงงาน	**การออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์โดยอาศัยหลักการแพร่ของคลื่น**		
ผู้ดำเนินโครงงาน	**นายนัฐวัช**	**สุขรวย**	**รหัส 52362724**
	นายวิศรุต	**พัฒนพงษ์**	**รหัส 52362922**
ที่ปรึกษาโครงงาน	**ดร.พนัส**	**นัถฤทธิ์**	
สาขาวิชา	**วิศวกรรมคอมพิวเตอร์**		
ภาควิชา	**วิศวกรรมไฟฟ้าและคอมพิวเตอร์**		
ปีการศึกษา	2555		

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาระบบสำหรับการออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์โดยนำหลักการแพร่ของคลื่นมาประยุกต์ใช้เพื่อเลือกหาเส้นทางที่มีระยาทางสั้นและปลอดภัยสำหรับหุ่นยนต์ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายปลายทางอย่างมีประสิทธิภาพ ซึ่งระบบที่พัฒนาขึ้นสามารถใช้งานกับแผนที่ (Input map) ขนาดต่างๆได้ รวมทั้งสามารถเลือกลักษณะการแพร่ของคลื่นได้ทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทาง โดยผลลัพธ์ของระบบที่พัฒนาขึ้นจะเป็นการสร้างเส้นทางการเคลื่อนที่ของหุ่นยนต์ตั้งแต่จุดเริ่มต้นไปยังจุดหมายปลายทางออกมาในรูปของเป้าหมายย่อยๆ ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหาโดยไม่ชนสิ่งกีดขวาง เพื่อจัดการภารกิจให้เสร็จสิ้นต่อไป

Project title	Development of Path Planning Algorithm using Wavefront Propagation Approach	
Name	Mr. Natthawat Sukruay	ID. 52362724
	Mr. Wisarut Phattanapong	ID. 52362922
Project advisor	Dr. PanusNattharith	
Major	Computer Engineering	
Department	Electrical and Computer Engineering	
Academic Year	2012	

Abstract

This project is development of path planning algorithm using wavefront propagation approach. Wavefront algorithm enabled autonomous path planning in a map field and the wavefront begins at the goal position and propagates outwards until to the start position have a value other than zero. So the wavefront algorithm finds the appropriate path for the robot moving from the start position to the goal position. This program can be used with the map (Input map) in different size . The wavefront's propagates can be choose 4 or 8 neighbor of the wave's spread direction. So the result of development, it's will make the efficient path for the robot's moving from the start position to the goal position until finish the work.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์ฉบับนี้สำเร็จลุล่วงมาได้นั้น เนื่องจากความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษาโครงงาน ดร. พนัส นัถฤทธิ์ ที่กรุณาสละเวลาให้คำแนะนำในการทำงานตลอดถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนวทางการแก้ไขปัญหาตลอดระยะเวลาการทำโครงงานพร้อมทั้งให้คำแนะนำที่เป็นประโยชน์ทำให้การทำโครงงานเป็นไปอย่างราบรื่น ทั้งนี้ต้องขอขอบพระคุณกรรมการทั้งสามท่านอันได้แก่ ดร. สุรเดช จิตประไพกุลศาล อาจารย์รัฐภูมิ วรานุสาสน์ และอาจารย์เศรษฐา ตั้งค้าวานิช อาจารย์ประจำภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์มหาวิทยาลัยนเรศวร ที่สละเวลาอันมีค่าให้ปรึกษาและแนะแนวทางในการแก้ไขปัญหาต่างๆ

สุดท้ายนี้ผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา และอาจารย์ทุกท่าน ที่คอยสั่งสอนให้ความรู้จนผู้จัดทำสำเร็จการศึกษา และขอบคุณเพื่อนๆที่คอยให้กำลังใจ ช่วยให้คำปรึกษาทั้งในเรื่องเรียน เรื่องส่วนตัว จนสำเร็จลุล่วงมาได้ด้วยดี

ขออำนาจคุณพระศรีรัตนตรัย และสิ่งศักดิ์สิทธิ์ทั้งหลายบันดาลให้บิดามารดาและอาจารย์ทุกท่านสุขภาพแข็งแรงและเป็นแรงผลักดันให้คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวรมีความก้าวหน้าต่อไป

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันเทคโนโลยีเข้ามามีบทบาทมากในชีวิตประจำวันและมีความก้าวหน้าอย่างมาก โดยเฉพาะทางด้านการพัฒนาเทคโนโลยีอิเล็กทรอนิกส์ มนุษย์ใช้ความสามารถในการพัฒนาอุปกรณ์และเครื่องมืออำนวยความสะดวกในด้านต่างๆ เพื่อลดการใช้แรงงานมนุษย์ และเพิ่มความสะดวกสบายในการดำเนินชีวิต จึงก่อให้เกิดการแข่งขันในการพัฒนาอุปกรณ์ เครื่องมือและนวัตกรรมใหม่ๆขึ้น หุ่นยนต์นับเป็นนวัตกรรมหนึ่งที่สำคัญและมีใช้กันอย่างแพร่หลาย อาทิเช่น หุ่นยนต์ใช้งานในครัวเรือน หุ่นยนต์สำรวจ หุ่นยนต์รับส่งเอกสารฯลฯ การพัฒนาโปรแกรมสำหรับออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์จึงมีความสำคัญ เพื่อที่จะสามารถควบคุมหุ่นยนต์ให้เคลื่อนที่ไปยังเป้าหมายได้อย่างมีประสิทธิภาพโดยไม่ชนสิ่งกีดขวาง ซึ่งโปรแกรมที่พัฒนาขึ้นนั้นจะให้ผลลัพธ์ออกมาเป็นเส้นทางการเคลื่อนที่ของหุ่นยนต์ที่มีระยะทางสั้นและปลอดภัย หุ่นยนต์จะสามารถเคลื่อนที่เข้าสู่จุดหมายได้อย่างรวดเร็ว ส่งผลให้สิ้นเปลืองพลังงานในการปฏิบัติภารกิจเพียงเล็กน้อย

การพัฒนาโปรแกรมการออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์ในโครงงานนี้ จะนำเอาหลักการแพร่ของคลื่นมาประยุกต์ใช้ซึ่งผลลัพธ์ที่ได้ในเบื้องต้นจะเป็นเส้นทางการเคลื่อนที่ของหุ่นยนต์จำนวนหลายเส้นทาง อย่างไรก็ตามในขั้นตอนต่อมา โปรแกรมจะทำการเลือกเส้นทางที่เหมาะสมกับหุ่นยนต์เพียงเส้นทางเดียว เพื่อนำไปใช้ควบคุมหุ่นยนต์ โดยขั้นตอนดังกล่าวจะอาศัยทฤษฎีการค้น (Searching Technique) มาช่วยในการตัดสินใจ เพื่อหาเส้นทางที่สั้นและปลอดภัยสำหรับหุ่นยนต์ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายปลายทางอย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาและนำทฤษฎีการวางแผนเส้นทางการเคลื่อนที่ของหุ่นยนต์มาใช้ในการพัฒนาโปรแกรม

1.2.2 เพื่อศึกษาและนำหลักการแพร่ของคลื่น (Wavefront algorithm) มาประยุกต์ใช้ในการวางแผนเส้นทางการเคลื่อนที่ของหุ่นยนต์

1.2.4 เพื่อพัฒนาโปรแกรมการออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์ จากจุดเริ่มต้นไปยังจุดหมายปลายทาง โดยเลือกใช้เส้นทางที่เหมาะสมในการเคลื่อนที่

1.3 ขอบข่ายของโครงงาน

1.3.1 พัฒนาโปรแกรมด้วย Microsoft Visual Studio 2010 Express โดยใช้ภาษา Visual C# เพื่อพัฒนาโปรแกรมใช้งานบนระบบปฏิบัติการ Windows

1.3.2 โปรแกรมที่พัฒนาขึ้นสามารถใช้งานกับแผนที่ (Input map) ในมาตราส่วนต่างๆได้

1.3.3 โปรแกรมสามารถสร้างเส้นทางการเคลื่อนที่เหมาะสมให้กับหุ่นยนต์ได้ โดยให้ผลลัพธ์ออกมาในรูปของเป้าหมายย่อยๆ (A Series of Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหาเพื่อทำภารกิจให้เสร็จสิ้น

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาทฤษฎีและหลักการในเรื่องต่างๆ ดังนี้

1) ศึกษาทฤษฎีการวางแผนเส้นทางการเคลื่อนที่ของหุ่นยนต์

2) ศึกษาหลักการแพร่ของคลื่น

1.4.2 ศึกษาการเขียนโปรแกรมด้วยภาษาVisual C#

1.4.3 ออกแบบและพัฒนาโปรแกรมตามทฤษฎีและหลักการที่ได้ศึกษามา

1.4.4 ทดสอบการทำงานของโปรแกรมและปรับปรุง

1.4.5 ตรวจทานความถูกต้องและแก้ไข้ข้อผิดพลาดของโปรแกรม

1.4.6 จัดทำรายงานและสรุปผลการทำงาน

1.4.7 ตรวจทานและแก้ไขข้อผิดพลาดของรายงาน

1.4.8 จัดทำเป็นรูปเล่ม

1.5 ตารางแสดงกิจกรรมการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

กิจกรรม	ปี 2555							ปี 2556		
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
ศึกษาทฤษฎีและหลักการในเรื่องต่างๆ	←	→								
ศึกษาการเขียนโปรแกรมด้วยภาษา Visual C#		←	→							
ออกแบบและพัฒนาโปรแกรม			←		→					

ตารางที่ 1.1 แผนการดำเนินงาน (ต่อ)

กิจกรรม	ปี 2555							ปี 2556		
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
ทดสอบการทำงานของโปรแกรมและปรับปรุง							↔			
ตรวจทานความถูกต้องและแก้ไข้ข้อผิดพลาดของโปรแกรม								↔		
จัดทำรายงานและสรุปผลการทำงาน		↔								
ตรวจทานและแก้ไขข้อผิดพลาดของรายงาน								↔		
จัดทำเป็นรูปเล่ม									↔	

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 ได้รับความรู้เรื่องการเขียนโปรแกรมด้วยภาษา Visual C#

1.6.2 ได้รับความรู้เรื่องการวางแผนเส้นทางการเคลื่อนที่ของหุ่นยนต์

1.6.3 ได้โปรแกรมที่สามารถออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์ โดยไม่ชนสิ่งกีดขวางและไปถึงจุดหมายปลายทางได้

1.6.4 โปรแกรมที่พัฒนาขึ้นสามารถนำไปปรับปรุงเพื่อใช้งานร่วมกับหุ่นยนต์เคลื่อนที่ประเภทต่างๆได้

1.7 งบประมาณ

1.7.1 ค่าถ่ายเอกสารและค่าพิมพ์เอกสาร		500	บาท
1.7.2 ค่าหนังสือคู่มือเกี่ยวกับการใช้ศึกษาค้นคว้า		500	บาท
1.7.3 ค่าจัดทำรูปเล่ม		1000	บาท
	รวมทั้งสิ้น	2000	บาท

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

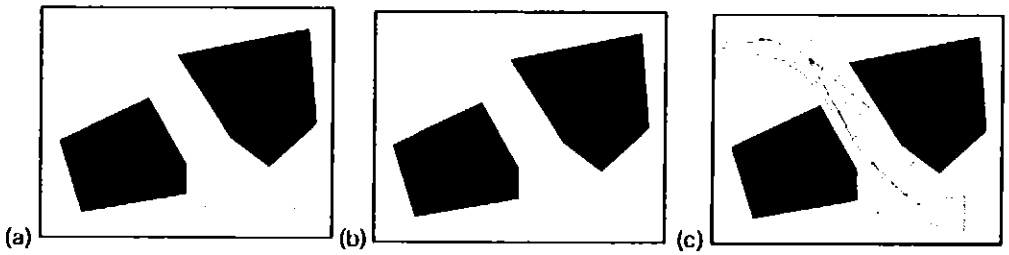
หุ่นยนต์ (Robot) คือ เครื่องจักรกลชนิดหนึ่ง มีลักษณะโครงสร้างและรูปร่างแตกต่างกัน แบ่งออกเป็น 2 ประเภท คือ หุ่นยนต์ชนิดที่ติดตั้งอยู่กับที่ (Fixed robot) และหุ่นยนต์ชนิดที่เคลื่อนที่ได้ (Mobile robot) ปัจจุบันเทคโนโลยีของหุ่นยนต์เจริญก้าวหน้าอย่างรวดเร็วและเริ่มเข้ามามีบทบาทในชีวิตประจำวันของมนุษย์ในด้านต่างๆ เช่น ด้านอุตสาหกรรมการผลิต ด้านการแพทย์ ด้านงานสำรวจ ด้านงานในอวกาศ หรือแม้แต่หุ่นยนต์ที่ถูกสร้างขึ้นเพื่อเป็นของเล่นให้กับมนุษย์ จนกระทั่งในปัจจุบันนี้ได้มีการพัฒนาให้หุ่นยนต์มีลักษณะที่คล้ายมนุษย์ เพื่อให้อาศัยอยู่ร่วมกันกับมนุษย์ในชีวิตประจำวันในบทนี้จะกล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการพัฒนาโปรแกรมเพื่อการออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์โดยอาศัยหลักการแพร่ของคลื่นสำหรับหุ่นยนต์ชนิดเคลื่อนที่ได้

2.1 พื้นฐานของหุ่นยนต์เคลื่อนที่ (Mobile Robot Navigation)

ข้อเสียของหุ่นยนต์ชนิดติดตั้งอยู่กับที่ (Fixed robot) คือ ไม่สามารถเคลื่อนที่ไปมาจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งได้ ทำให้มีข้อจำกัดเกี่ยวกับขอบเขตพื้นที่การทำงาน ซึ่งแตกต่างจากหุ่นยนต์ชนิดเคลื่อนที่ได้ (Mobile robot) ที่สามารถเคลื่อนที่ไปมาได้อย่างอิสระ หรือมีการเคลื่อนที่ไปมาในสถานที่ต่างๆ โดยการเคลื่อนที่ของหุ่นยนต์นั้นส่วนใหญ่เป็นการเคลื่อนที่ในแนวราบ มีอุปกรณ์ที่ใช้ในการเคลื่อนที่ที่ดีที่สุด คือ ล้อ เพราะทำให้มีการเคลื่อนที่ได้อย่างต่อเนื่อง แต่ปัจจุบันก็มีจำนวนไม่น้อยที่มีการนำขามาใช้กับหุ่นยนต์ ซึ่งทำให้การเคลื่อนที่ในทางต่างระดับนั้นดีกว่าล้อ [1] ซึ่งพื้นฐานการเคลื่อนที่ของหุ่นยนต์จะมีหลักการดังนี้

2.1.1 การวางแผนการเคลื่อนที่ (Path Planning)

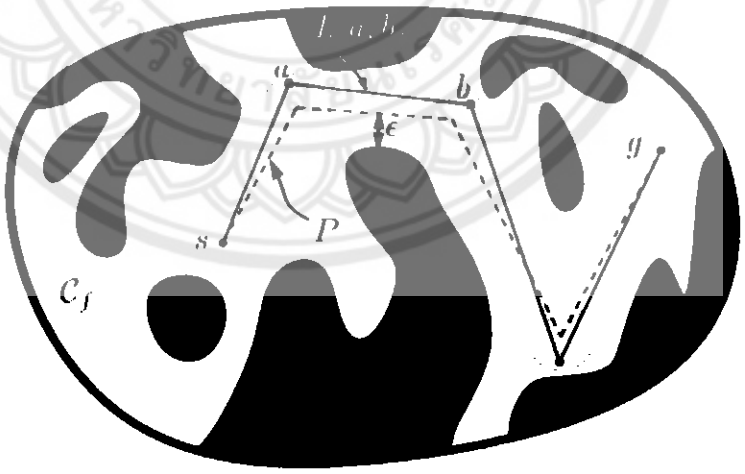
การวางแผนการเคลื่อนที่มีความสำคัญต่อการพัฒนาหุ่นยนต์เป็นอย่างมาก ตัวอย่างการวางแผนการเคลื่อนที่ได้แก่ เมื่อหุ่นยนต์เคลื่อนที่ได้รับคำสั่งให้เดินจากห้องหนึ่งไปยังอีกห้องหนึ่ง หุ่นยนต์จะมีวิธีการอย่างไรเพื่อไปให้ถึงจุดหมาย โดยที่ไม่ชนสิ่งกีดขวางดังตัวอย่างในรูปที่ 2.1 แสดงการเคลื่อนที่ของหุ่นยนต์จากจุดเริ่มต้น (รูปที่ 2.1a) ไปยังจุดที่ต้องการ (รูปที่ 2.1b) จะเห็นว่า หุ่นยนต์ทำการเคลื่อนที่เพื่อให้สามารถลอดผ่านช่องแคบระหว่างสิ่งกีดขวางได้ (รูปที่ 2.1c) จากนั้นเมื่อพ้นสิ่งกีดขวางหุ่นยนต์ก็จะเดินทางไปยังจุดที่ต้องการ การกำหนดว่าจะให้หุ่นยนต์เคลื่อนตัวหรือหมุนตัว คือการวางแผนการเคลื่อนที่ [1]



รูปที่ 2.1 การเคลื่อนที่อย่างต่อเนื่องจากจุดเริ่มต้นไปยังจุดที่ต้องการ [2]

หลักการ

ปัญหาพื้นฐานของการวางแผนการเคลื่อนที่คือการสร้างการเคลื่อนไหวอย่างต่อเนื่องจากจุดเริ่มต้นไปยังจุดเป้าหมายโดยไม่ชนสิ่งกีดขวาง (Obstacle) โดย ณ ตำแหน่งเริ่มต้น ท่าทางหรือโครงแบบ (Configuration) ของหุ่นยนต์มักเขียนด้วย S (Starting configuration) ในขณะที่โครงแบบเป้าหมายนั้นแทนด้วย G (Goal configuration) ดังแสดงในรูปที่ 2.2 ปัญหาการวางแผนการเคลื่อนที่ของหุ่นยนต์ที่ง่ายที่สุดคือ กรณีทราบตำแหน่งของสิ่งกีดขวางอยู่แต่แรก ปกติรูปร่างของหุ่นยนต์และสิ่งกีดขวางจะอยู่ในปริภูมิ 2 มิติหรือ 3 มิติ ที่เรียกว่า บริเวณการทำงาน (Workspace) อย่างไรก็ตามเส้นทาง (path) การเคลื่อนที่ของหุ่นยนต์จะถูกอธิบายในรูปของปริภูมิโครงแบบ (Configuration space) [1]



รูปที่ 2.2 โครงแบบที่มีตำแหน่งเริ่มต้นและตำแหน่งเป้าหมาย [3]

ปริภูมิโครงแบบ (Configuration space)

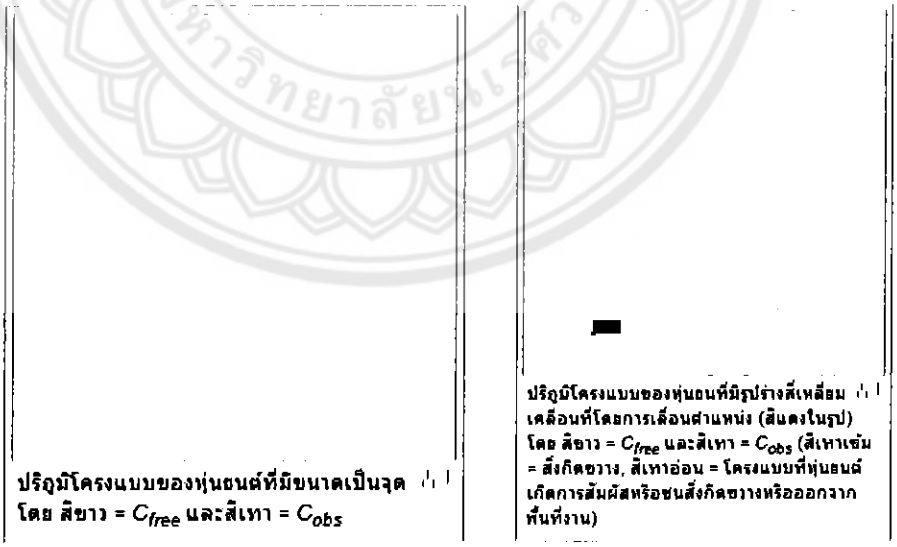
ท่าทางหนึ่งๆ ของหุ่นยนต์ถูกเรียกว่าเป็น "โครงแบบ" (Configuration) ของหุ่นยนต์ เซ็ตของโครงแบบที่เป็นไปได้ทั้งหมดเรียกว่า ปริภูมิโครงแบบ (Configuration space) ดังแสดงในรูปที่ 2.3และนิยมเขียนแทนด้วยเซต C ยกตัวอย่างเช่น [1]

- สมมติหุ่นยนต์มีขนาดเป็นเพียงจุด (ไม่มีขนาด) เคลื่อนที่บนบริเวณการทำงาน (Workspace) ที่เป็นระนาบ 2 มิติ ในกรณีนี้ปริภูมิโครงแบบ C คือระนาบ และโครงแบบของหุ่นยนต์สามารถเขียนอธิบายได้ด้วยพารามิเตอร์ 2 ค่าคือ (x, y)

- ถ้าหุ่นยนต์มีรูปร่างเพียง 2 มิติที่สามารถเลื่อนตำแหน่งและหมุนได้ บริเวณการทำงาน ในกรณีนี้จะยังคงเป็น 2 มิติ แต่ปริภูมิโครงแบบ C จะเป็นยูคลิเดียนกรุ๊ปแบบพิเศษ (special Euclidean group) $SE(2) = R^2 \times SO(2)$ (โดยที่ SO(2) เป็น special orthogonal group ของการหมุนใน 2 มิติ) และโครงแบบในกรณีนี้สามารถอธิบายได้ด้วยพารามิเตอร์ 3 ค่า ได้แก่ (x, y, θ)

- ถ้าหุ่นยนต์มีรูปร่างเพียง 3 มิติที่สามารถเลื่อนตำแหน่งและหมุนได้ บริเวณการทำงาน ในกรณีนี้จะเป็น 3 มิติ แต่ปริภูมิโครงแบบ C จะเป็นยูคลิเดียนกรุ๊ปแบบพิเศษ (special Euclidean group) $SE(3) = R^3 \times SO(3)$ และโครงแบบในกรณีนี้สามารถอธิบายได้ด้วยพารามิเตอร์ 6 ค่า ได้แก่ (x, y, z) สำหรับการเลื่อนตำแหน่ง และ มุมออยเลอร์ (Euler_angles) (α, β, γ)

- ถ้าหุ่นยนต์เป็นหุ่นยนต์แบบแขนกลยึดติดอยู่กับที่ และมีข้อต่อจำนวน N ปริภูมิโครงแบบ C ในกรณีนี้จะเป็นปริภูมิ N มิติ



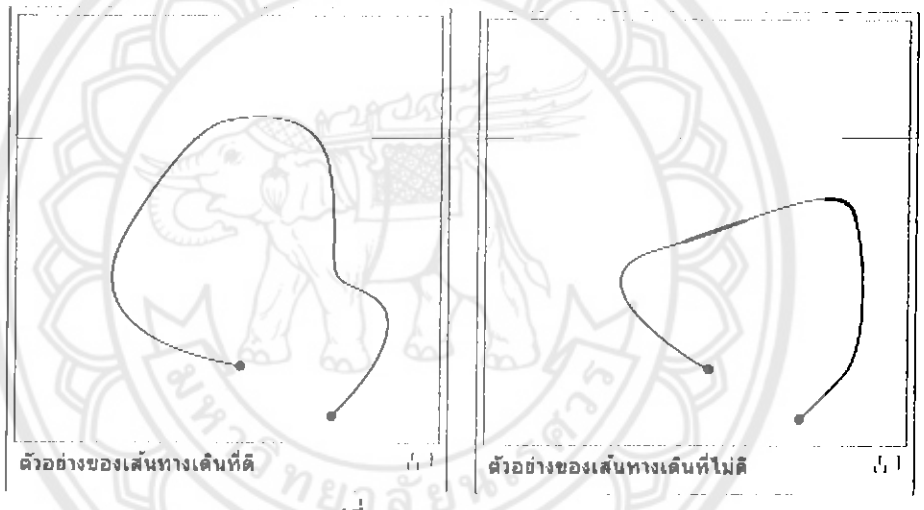
รูปที่ 2.3 Configuration space [1]

ปริภูมิว่าง (Free space)

เป็นโครงแบบที่ไม่มีการชนกับสิ่งกีดขวาง เรียกว่า ปริภูมิว่าง (Free space) เขียนแทนด้วย C_{free} ส่วนสิ่งกีดขวางที่อยู่ใน C_{free} ในปริภูมิโครงแบบ C เรียกว่า Obstacle region เขียนแทนด้วย C_{obs} [1]

การวางแผนเส้นทางการเคลื่อนที่ที่ดี จะต้องควบคุมหุ่นยนต์ไม่ให้ชนสิ่งกีดขวาง หรือใกล้ชิดกับสิ่งกีดขวางมากเกินไป เพื่อให้หุ่นยนต์เคลื่อนที่ไปยังจุดหมายปลายทางได้อย่างปลอดภัย ดังแสดงในรูปที่ 2.4

ในบทนี้การทดสอบโครงแบบของหุ่นยนต์อยู่ในปริภูมิว่าง (C_{free}) หรือไม่ และทดสอบว่าหุ่นยนต์จะเคลื่อนที่ชนสิ่งกีดขวาง (C_{obs}) หรือไม่ จะนำเอาหลักการแพร่ของคลื่น (Wavefront) มาใช้ในการตรวจสอบ



รูปที่ 2.4 Path Planning [1]

2.1.2 การระบุตำแหน่งและการสร้างแผนที่ (Simultaneous Localization and Mapping)

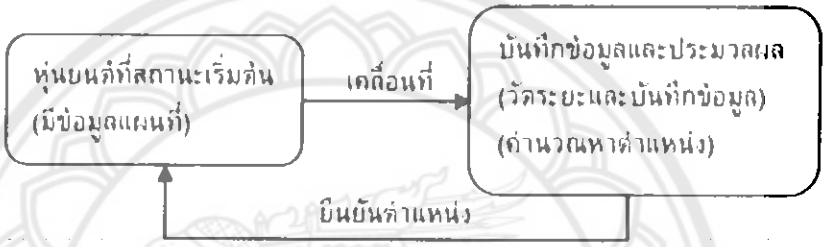
การระบุตำแหน่งและการสร้างแผนที่ (SLAM) มีความสำคัญต่อหุ่นยนต์เคลื่อนที่เป็นอย่างมากและเป็นหนึ่งในหลักการของการควบคุมหุ่นยนต์เคลื่อนที่ หลักการดังกล่าวจะทำให้หุ่นยนต์สามารถปฏิบัติการในสภาพแวดล้อมต่างๆได้ด้วยตนเองสแลมได้รับความสนใจอย่างมากในด้านการวิจัยและพัฒนาในช่วง 20 ปี จุดเริ่มต้นของสแลมเกิดขึ้นในการประชุมระบบอัตโนมัติและหุ่นยนต์ของสถาบันวิชาชีพวิศวกรไฟฟ้าและอิเล็กทรอนิกส์ (IEEE) ณ เมืองซานฟานซิสโก ประเทศสหรัฐอเมริกา ค.ศ.1986 [4]

โดยทั่วไปหุ่นยนต์จะถูกกำหนดเส้นทางหรือมีแผนที่ของการปฏิบัติงานต่างๆไว้ก่อน อย่างไรก็ตามในการปฏิบัติงานสำรวจในสภาวะแวดล้อมที่ไม่รู้จักหรือไม่เคยไปมาก่อน การมีเส้นทางหรือแผนที่ย่อมเป็นไปไม่ได้ ดังนั้นจึงได้มีการศึกษาและมีการนำสแลมมาใช้เป็น

กระบวนการที่ทำให้หุ่นยนต์สามารถระบุตำแหน่งของตัวเองในแผนที่ที่สร้างขึ้นจากสภาพแวดล้อมต่างๆ ที่มีการตรวจสอบได้ ซึ่งจะสามารถนำไปประยุกต์ใช้งานได้ในสภาพแวดล้อมหลายแบบ โดยทั่วไปหลักการของสแลม จะประกอบไปด้วยปัญหาของการระบุตำแหน่งและปัญหาของการสร้างแผนที่ ซึ่งต้องอาศัยวิธีการนำข้อมูลจากเซ็นเซอร์มาคัดกรองเพื่อนำไปใช้แก้ปัญหา [4]

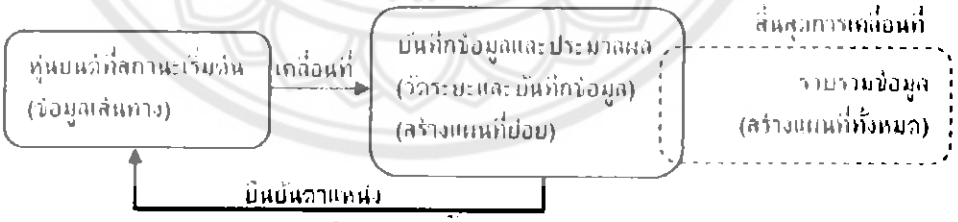
รูปแบบปัญหาของสแลม

1. ปัญหาของการระบุตำแหน่ง [4] หุ่นยนต์ได้รับข้อมูลของแผนที่มาก่อน เมื่อเคลื่อนที่หุ่นยนต์จะประมวลผลหาตำแหน่งของตัวเองโดยใช้ข้อมูลที่วัดได้จากเซ็นเซอร์ต่างๆ และนำไปอ้างอิงกับแผนที่ที่มีอยู่เดิม ซึ่งจะนำไปใช้ประโยชน์ได้ในด้านการติดตาม ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 ปัญหาการระบุตำแหน่ง [4]

2. ปัญหาการสร้างแผนที่ [4] หุ่นยนต์ถูกกำหนดเส้นทางการเคลื่อนที่ในตำแหน่งต่างๆ และระหว่างการเคลื่อนที่นั้น หุ่นยนต์จะใช้เซ็นเซอร์เก็บบันทึกข้อมูลสภาพแวดล้อมในแต่ละตำแหน่ง และนำข้อมูลทั้งหมดมาสร้างเป็นแผนที่ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 ปัญหาการสร้างแผนที่ [4]

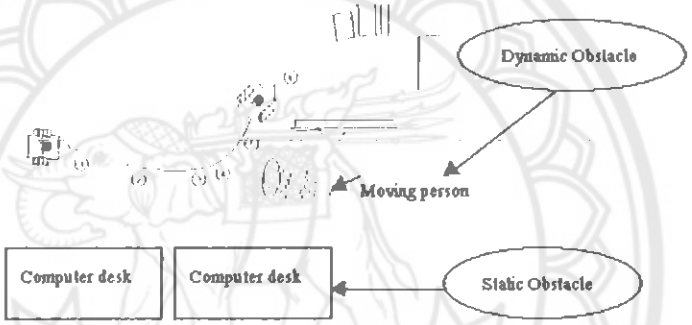
วิธีคัดกรองข้อมูลที่ได้มาจากเซ็นเซอร์มีความสำคัญต่อการทำสแลม วิธีการที่ได้รับการยอมรับและใช้งานกันอย่างแพร่หลายได้แก่

- ตัวกรองคาลมาน (Kalman filter) เหมาะสมกับการใช้ในระบบที่เป็นเชิงเส้น ไม่มีสิ่งรบกวนเข้ามาเกี่ยวข้อง [5]

- ตัวกรองคาลมานแบบขยาย (Extended Kalman filter) เหมาะสมกับการใช้กับระบบที่ไม่เชิงเส้น และมีสิ่งรบกวน [4]

2.1.3 การหลบหลีกสิ่งกีดขวาง (Obstacle Avoidance)

ในหุ่นยนต์เคลื่อนที่ การหลบหลีกสิ่งกีดขวาง เป็นวัตถุประสงค์สำคัญอย่างยิ่งในการควบคุมหุ่นยนต์ไม่ให้เคลื่อนที่เข้าชนสิ่งกีดขวางหรือไม่ไปยังผิดตำแหน่ง เป็นพฤติกรรมที่หลีกเลี่ยงสิ่งกีดขวางทั้งในสถานะที่เป็น Static และ Dynamic โดยการเคลื่อนที่ของหุ่นยนต์จะทำการเลี้ยวซ้ายหรือเลี้ยวขวาไปยังพื้นที่ว่างโดยหลีกเลี่ยงสิ่งกีดขวาง อย่างไรก็ตามการหลบหลีกสิ่งกีดขวางจะต้องมีการวางแผนเส้นทางที่ถูกต้องพร้อมกับการตอบสนองล่วงหน้ากับเส้นทางที่ปราศจากสิ่งกีดขวาง และเส้นทางที่มีสิ่งกีดขวาง เพื่อที่จะควบคุมหุ่นยนต์ไปได้ในทิศทางที่ถูกต้อง [6]



รูปที่ 2.7 การหลบหลีกสิ่งกีดขวาง [6]

จากรูปที่ 2.7 เป็นตัวอย่างของหุ่นยนต์ที่ทำการหลีกเลี่ยงสิ่งกีดขวางทั้งที่เป็นแบบ Static และDynamic โดยในภาพกำหนดให้ Computer desk เป็น Static Obstacle หรือสิ่งกีดขวางที่อยู่กับที่ และ Moving Person เป็น Dynamic Obstacle หรือสิ่งกีดขวางที่เคลื่อนที่ได้ โดยที่ตัวหุ่นยนต์ได้มีการเคลื่อนที่ไปยังพื้นที่ที่ว่างเมื่อหุ่นยนต์ไปถึงพื้นที่ที่มีสิ่งกีดขวางอยู่ก็จะทำการหมุนตัวออกไปเพื่อเป็นการหลีกเลี่ยงสิ่งกีดขวางและเคลื่อนที่ไปยังพื้นที่ว่างต่อไป [6]

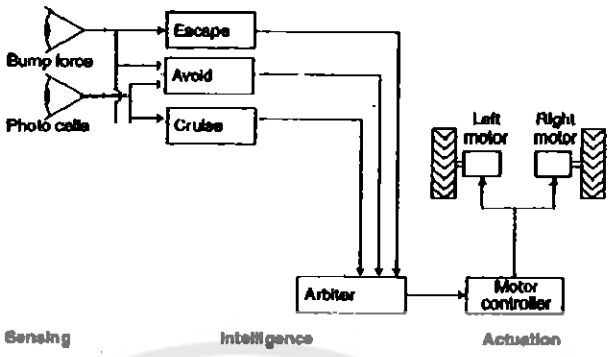
2.2 สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่ (Mobile Robot Architecture)

สถาปัตยกรรมควบคุมหุ่นยนต์ มีอยู่ 3 ระดับ คือ แบบ Deliberative, แบบ Reactive (Behavior-based) และแบบ Hybrid [6]

2.2.1 Deliberative or Planner-based

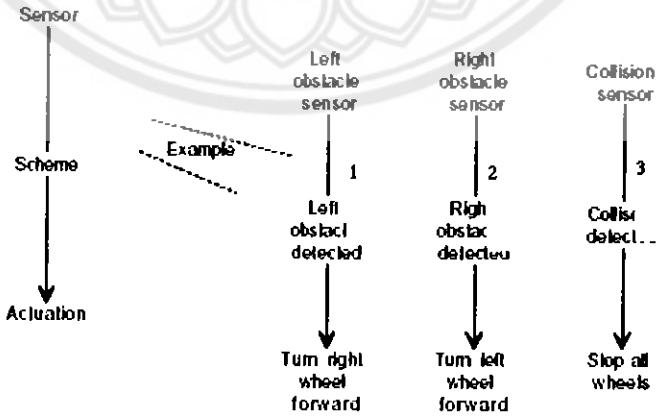
เป็นระบบพื้นฐานของการวางแผนการเคลื่อนที่ของหุ่นยนต์ (Path Planning) โดยหุ่นยนต์จะใช้ข้อมูลที่ได้มีการวางแผนไว้ ตัดสินใจเลือกเส้นทางได้ จากนั้นจะทำการเคลื่อนที่ไปตามเส้นทางที่มีการวางแผนไว้อย่างถูกต้อง

2.2.2 Reactive (Behavior-based)



รูปที่ 2.8 ระบบพื้นฐาน [6]

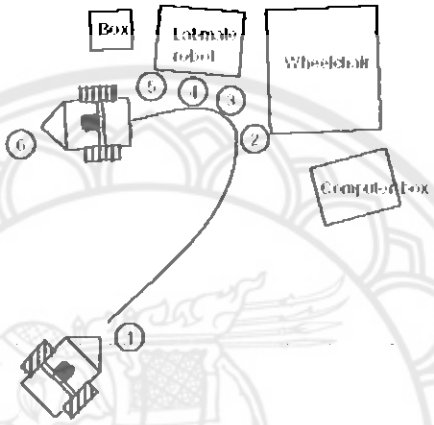
จากรูปที่ 2.8 เป็นภาพของระบบพื้นฐานการวางแผน ซึ่งประกอบไปด้วยสามส่วน คือ ส่วนตรวจรู้ (Sensing) ส่วนความคิด (Intelligence) และส่วนทำงาน (Actuation) โดยจากรูปจะมีการส่งค่าสภาพแวดล้อมมาจากเซ็นเซอร์ จากนั้นจะเข้าสู่กระบวนการคิด เพื่อให้หุ่นยนต์ตัดสินใจในการเลือกเส้นทาง เมื่อตัดสินใจได้ก็จะเข้าสู่ขั้นตอนการทำงานเป็นขั้นตอนสุดท้าย โดยจากภาพประกอบจะมีโมดูลที่รับข้อมูลจากเซ็นเซอร์ทั้งหมดสามตัวซึ่งในแต่ละตัวจะทำงานเป็นอิสระต่อกัน แต่ในบางครั้งโมดูลอาจจะต้องมีการสั่งงานพร้อมๆกันทำให้เกิดความขัดแย้งในการของใช้ทรัพยากร ดังนั้นจึงต้องมีโมดูลพิเศษขึ้นมาคือโมดูล Arbiter เพื่อทำหน้าที่คอยจัดการหรือจัดเรียงระดับความสำคัญของโมดูลต่างๆ โมดูลใดมีความสำคัญกว่าก็จะถูกนำไปใช้งานก่อนหรือทำงานก่อน



รูปที่ 2.9 Reactive [6]

พฤติกรรมพื้นฐานของหุ่นยนต์ ซึ่งมีระบบตอบสนอง (Reactive) ร่วมอยู่ด้วย ซึ่งอาจมีพฤติกรรมพื้นฐานต่างๆ เช่น พฤติกรรมที่เป็นการหลีกเลี่ยงการชน (Collision Avoidance), พฤติกรรมหลบหลีกสิ่งกีดขวาง (Obstacle Avoidance), พฤติกรรมเคลื่อนที่ไปตามเป้าหมายหรือค้นหาเป้าหมาย (Goal Seeking)

- Collision Avoidance เป็นพฤติกรรมที่หลีกเลี่ยงการชนซึ่งจะเป็นการป้องกันหุ่นยนต์ระหว่างทำการเคลื่อนที่ไม่ให้ไปชนสิ่งกีดขวางที่อยู่ข้างหน้าของหุ่นยนต์

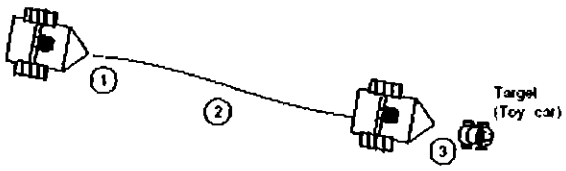


รูปที่ 2.10 Collision Avoidance [6]

จากรูปที่ 2.10 เป็นตัวอย่างของหุ่นยนต์ที่ทำการหมุนตัวออกไปยังพื้นที่ว่าง เมื่อมีการตรวจจับว่าจะมีการชนเกิดขึ้น

- Obstacle Avoidance เป็นพฤติกรรมหลบหลีกสิ่งกีดขวางโดยการเลี้ยวซ้ายเลี้ยวขวาเพื่อเคลื่อนที่ไปยังพื้นที่ว่าง

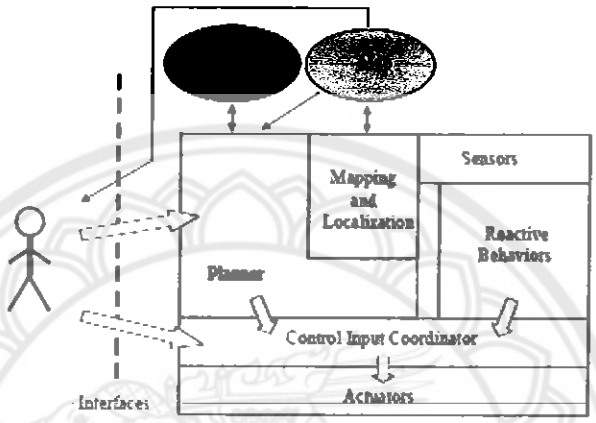
- Goal Seeking เป็นการเคลื่อนที่เข้าหาเป้าหมาย โดยจะมีการตรวจจับหรือค้นหาเป้าหมายก่อนเมื่อเจอเป้าหมายแล้วจะทำการเคลื่อนที่ไปยังเป้าหมายนั้นดังแสดงรูปที่ 2.11



รูปที่ 2.11 Goal Seeking[6]

2.2.3 Hybrid system

เป็นระบบการผสม ระหว่างหลักการหรือทฤษฎีต่างๆ เพื่อให้หุ่นยนต์เคลื่อนที่อย่างมีประสิทธิภาพมากยิ่งขึ้น ดังรูปแสดงในรูปที่ 2.12 จะเห็นได้ว่า มีการนำหลักการต่างๆ ทั้ง Deliberative, Map and Localization, Reactive Behaviors เป็นต้นมาใช้ จะทำให้หุ่นยนต์สามารถเคลื่อนที่ได้อย่างถูกต้อง ไม่คลาดเคลื่อน และมีประสิทธิภาพสมบูรณ์ยิ่งขึ้น [7]



รูปที่ 2.12 Diagram of the developed architecture. [7]

โดยในโครงงานนี้มีการนำเอาสถาปัตยกรรมแบบDeliberativeมาใช้ในการวางแผนเส้นทางการเคลื่อนที่ (Path Planning) เพื่อเลือกเส้นทางที่สั้นและปลอดภัยในการเคลื่อนที่ และมีการนำหลักการแพร่กระจายของคลื่น (Wavefrontalgorithm) มาใช้ในการหลีกเลี่ยงสิ่งกีดขวาง เพื่อสร้างเส้นทางที่หุ่นยนต์สามารถหลบหลีกสิ่งกีดขวางไปได้โดยปลอดภัย

2.3 การวางแผนเส้นทางกับการนำการแพร่กระจายของคลื่นมาประยุกต์ใช้

ในหัวข้อนี้เป็นการนำเอาทฤษฎีของการวางแผนเส้นทางการเคลื่อนที่และหลักการแพร่กระจายของคลื่นมาใช้ร่วมกันในการออกแบบเส้นทางให้หุ่นยนต์เคลื่อนที่

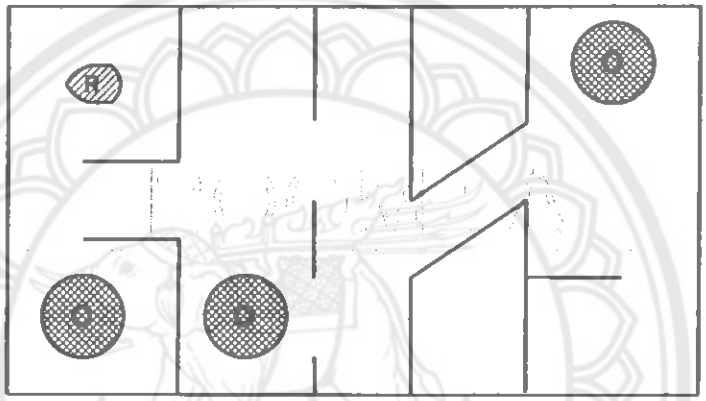
2.3.1 การวางแผนเส้นทาง(Path planning)

2.3.1.1 บริเวณการทำงานและคอนฟิกูเรชัน

เทคโนโลยีด้านหุ่นยนต์จะเหมือนกับศาสตร์ด้านอื่นๆที่มีคำศัพท์เฉพาะทางเพื่อช่วยอำนวยความสะดวกในการกล่าวถึง ซึ่งคำศัพท์ที่จะนำมาใช้สำหรับการวางแผนเส้นทาง จะมีดังนี้

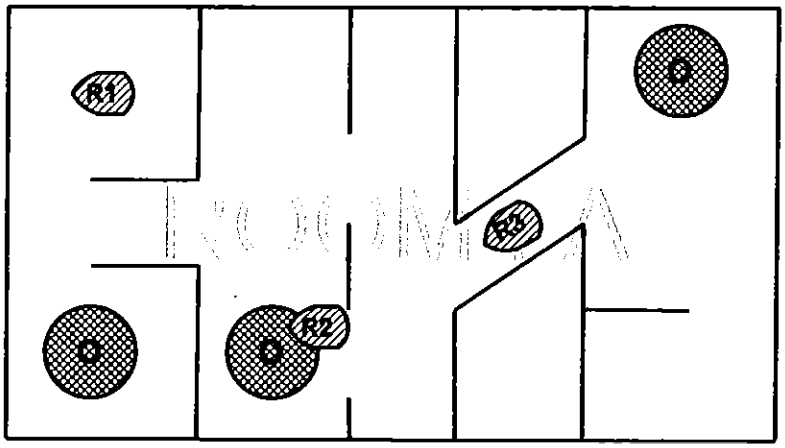
บริเวณการทำงาน (Workspace) [1]คือ ขอบเขตหรือบริเวณที่กำหนดไว้ให้หุ่นยนต์ทำงานดังแสดงรูปที่ 2.13 บริเวณการทำงานคือ ROOM : A

สิ่งกีดขวาง (Obstacle) [1]คือ สิ่งที่ทำให้เกิดความไม่สะดวกหรือติดขัดจนไม่สามารถลุล่วงไปได้ ในที่นี้จะเป็นวัตถุในบริเวณการทำงาน ซึ่งจะมีทั้งที่ควบคุมได้และควบคุมไม่ได้ ดังแสดงรูปที่ 2.14 สิ่งกีดขวางที่ควบคุมได้ เช่น ผนังห้อง กำแพง สิ่งกีดขวางที่ควบคุมไม่ได้ เช่น เก้าอี้ ตู้ ซึ่งสามารถทำการเคลื่อนย้ายได้ หรืออาจจะมีการเคลื่อนที่ของวัตถุที่ไม่สามารถรับรู้ถึงลักษณะการเคลื่อนที่ได้ จะเรียกพื้นที่ที่มีสิ่งกีดขวางนี้ว่า บริเวณต้องห้าม (Forbidden space) ส่วนพื้นที่ที่ไม่มีสิ่งกีดขวางเรียกว่าบริเวณอิสระ (Free space)



รูปที่ 2.13 แผนที่ ROOM : A

คอนฟิกุเรชัน (Configuration) คือ ลักษณะของหุ่นยนต์ในบริเวณการทำงาน ซึ่งก็คือ ตำแหน่ง (Position) และแนวการวางตัว (Orientation) และจะต้องไม่อยู่บนพื้นที่เดียวกับสิ่งกีดขวาง ดังแสดงรูปที่ 2.14 หุ่นยนต์ R1 และ R3 จะมีคอนฟิกุเรชันที่ต่างกัน โดยสังเกตได้อย่างชัดเจนจาก ตำแหน่งและแนวการวางตัวที่ต่างกัน แต่หุ่นยนต์ R2 จะแสดงให้เห็นถึงกรณีที่เป็นไปไม่ได้เมื่อมีการวางแผนเส้นทาง

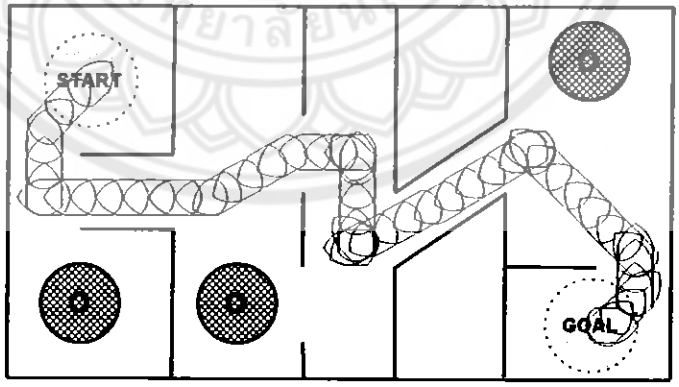


รูปที่ 2.14 แผนที่ ROOM : A(Configuration)

2.3.1.2 การวางแผนเส้นทาง(Path Planning)

การวางแผนเส้นทาง คือ การสร้างการเดินทางที่ดีให้กับหุ่นยนต์ มีความสำคัญต่อการพัฒนาหุ่นยนต์อัตโนมัติเป็นอย่างมากซึ่งในการวางแผนเส้นทางนั้นอาจจะประกอบด้วยหลายองค์ประกอบ เช่น ความสำเร็จในการเดินทางสู่จุดหมายที่ต้องการ ความปลอดภัย ความเหมาะสมของระยะทาง ฯลฯ [1]

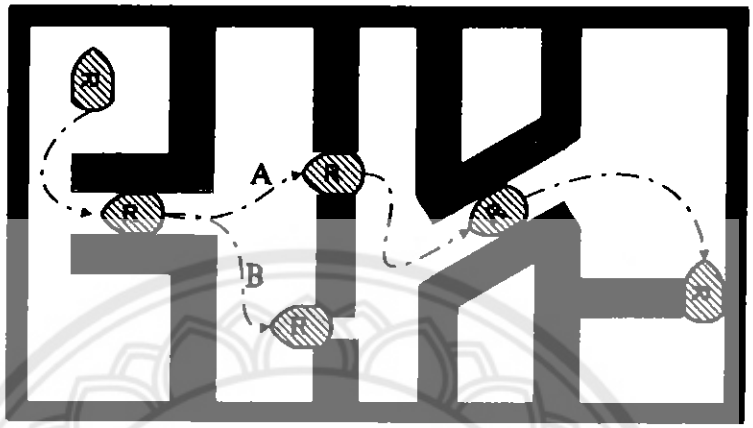
ความสำเร็จในการเดินทางสู่จุดหมายที่ต้องการ (Complete) คือ การกำหนดความเปลี่ยนแปลงคอนฟิกูเรชันของหุ่นยนต์ เพื่อให้หุ่นยนต์เคลื่อนตัวจากจุดเริ่มต้น (Start point) ไปยังจุดที่ต้องการ (Goal point) [1] ดังแสดงรูปที่ 2.15



รูปที่ 2.15 Configuration (Start point to Goal point)

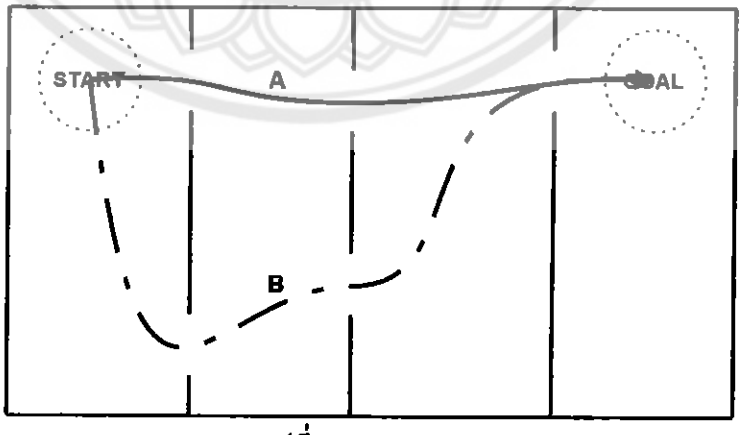
ความปลอดภัย (Safety) คือการที่หุ่นยนต์เคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายที่ต้องการโดยไม่ชนสิ่งกีดขวางหรือเข้าไปในบริเวณต้องห้าม โดยจะใช้การขยายตัว (Expand) ของสิ่งกีดขวาง ซึ่งในการออกแบบเส้นทางจะเป็นสิ่งกีดขวางเสมือนเพื่อกำหนดบริเวณอิสระใหม่ เพื่อ

เพิ่มความปลอดภัยให้กับหุ่นยนต์ ดังแสดงรูปที่ 2.16 จะสังเกตได้ว่าเส้นทาง A เมื่อเกิดการขยายตัวของสิ่งกีดขวางแล้วจะมีความปลอดภัยต่อหุ่นยนต์มากกว่าเส้นทาง B จึงสมควรเลือกเส้นทาง A ในการเคลื่อนที่ไปยังจุดหมาย



รูปที่ 2.16 Expand of obstacle

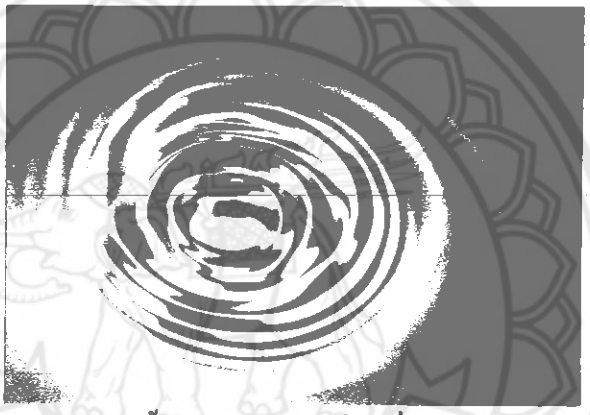
ความเหมาะสมของระยะทาง (Optimize) คือการที่หุ่นยนต์เคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายที่ต้องการ โดยใช้เวลาน้อย มีระยะทางสั้นที่สุด จากเส้นทางทั้งหมดที่เป็นไปได้จากรูปที่ 2.17 จะสังเกตได้ว่ามี 2 เส้นทางหรืออาจมีมากกว่า 2 เส้นทาง ที่จะไปยังจุดหมายที่ต้องการ เส้นทาง A เมื่อเปรียบเทียบกับเส้นทาง B แล้วมีความเหมาะสมกว่าทั้งด้านเวลาและระยะทาง ในการออกแบบเส้นทางจะต้องเลือกเส้นทาง A แต่จะต้องสนใจเรื่องความปลอดภัยของหุ่นยนต์ประกอบไปด้วย



รูปที่ 2.17 Optimize

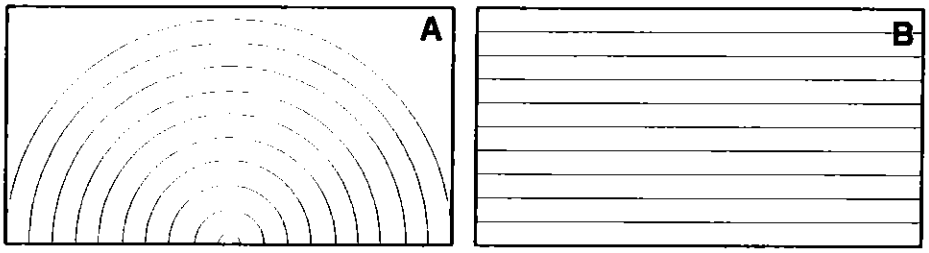
2.3.2 คลื่น (Wave)

คลื่น หมายถึง ลักษณะของการถูกรบกวน ที่มีการแผ่กระจาย เคลื่อนที่ออกไป ในลักษณะของการกวัดแกว่ง หรือกระเพื่อม และมักจะมีการส่งถ่ายพลังงานไปด้วย คลื่นเชิงกลซึ่งเกิดขึ้นในตัวกลาง (ซึ่งเมื่อมีการปรับเปลี่ยนรูป จะมีความแรงยืดหยุ่นในการดึดตัวกลับ) จะเดินทางและส่งผ่านพลังงานจากจุดหนึ่งไปยังอีกจุดหนึ่งในตัวกลาง โดยไม่ทำให้เกิดการเคลื่อนตำแหน่งอย่างถาวรของอนุภาคตัวกลางคือไม่มีการส่งถ่ายอนุภาคนั่นเอง แต่จะมีการเคลื่อนที่แกว่งกวัด (oscillation) ไปกลับของอนุภาค [8] อย่างไรก็ตามสำหรับการแผ่รังสีคลื่นแม่เหล็กไฟฟ้า และการแผ่รังสีแรงดึงดูดนั้นสามารถเดินทางในสุญญากาศได้ โดยไม่ต้องมีตัวกลาง ตัวอย่างของคลื่นแสดงดังรูปที่ 2.18



รูปที่ 2.18 ตัวอย่างผิวน้ำถูกรบกวน เกิดเป็นคลื่นแผ่กระจายออกรอบข้าง

หน้าคลื่น (Wavefront) คือแนวต่อกันของคลื่นที่มีเฟสเดียวกัน โดยในขบวนการใดขบวนการหนึ่งอาจจะมีกี่หน้าคลื่นก็ได้ หน้าคลื่นอาจมีลักษณะเป็นเส้นโค้ง รูปที่ 2.19(A) หรือเส้นตรง รูปที่ 2.19(B) ก็ได้



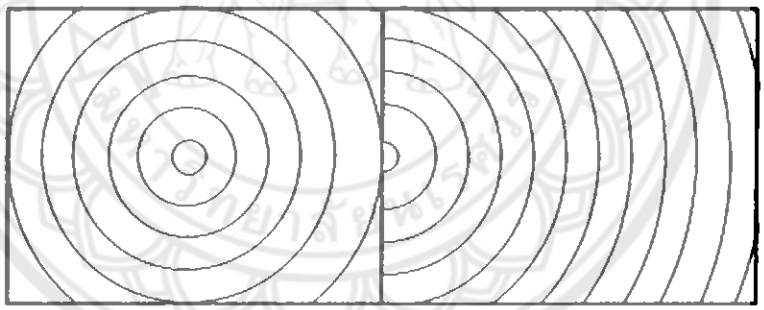
รูปที่ 2.19 ตัวอย่างลักษณะหน้าคลื่น

ลักษณะของหน้าคลื่น

- คลื่นหน้าตรงทิศทางคลื่นขนานกัน
- คลื่นหน้าโค้งวงกลมทิศทางคลื่นเป็นแนวรัศมีของวงกลม
- ทิศทางคลื่นจะตั้งฉากกับหน้าคลื่นเสมอ
- หน้าคลื่นที่ติดกันจะห่างกันเท่ากับความยาวคลื่น

การเลี้ยวเบน หรือการแพร่กระจายคลื่น (diffraction) เกิดจากคลื่นเคลื่อนที่ไปพบสิ่งกีดขวาง ทำให้คลื่นส่วนหนึ่งอ้อมบริเวณของสิ่งกีดขวางแผ่ไปทางด้านหลังของสิ่งกีดขวางนั้น

การเลี้ยวเบนของคลื่นผ่านช่องแคบเดี่ยว (Single Slit) คือ เมื่อคลื่นเคลื่อนที่ผ่านสิ่งกีดขวาง ซึ่งเป็นช่องแคบคลื่นจะเลี้ยวเบนผ่านช่องแคบไป ปรากฏเป็นคลื่นหลังสิ่งกีดขวางได้ ซึ่งการเลี้ยวเบนนี้จะเกิดได้ดี ถ้าหากช่องแคบนั้นมีความกว้างประมาณเท่า หรือน้อยกว่าความยาวคลื่น โดยเสมือนหนึ่งว่าช่องแคบนั้นทำหน้าที่เป็นแหล่งกำเนิดคลื่นใหม่ให้หน้าคลื่นวงกลมออกมารอบช่องแคบนั้น แต่ถ้าช่องแคบนั้นกว้างกว่าความยาวคลื่นจะเกิดการเลี้ยวเบนและเกิดการแทรกสอดขึ้นด้วย ดังรูปที่ 2.20 คลื่นจะค่อยๆ ก่อตัวขึ้นใหม่จากช่องของสิ่งกีดขวาง



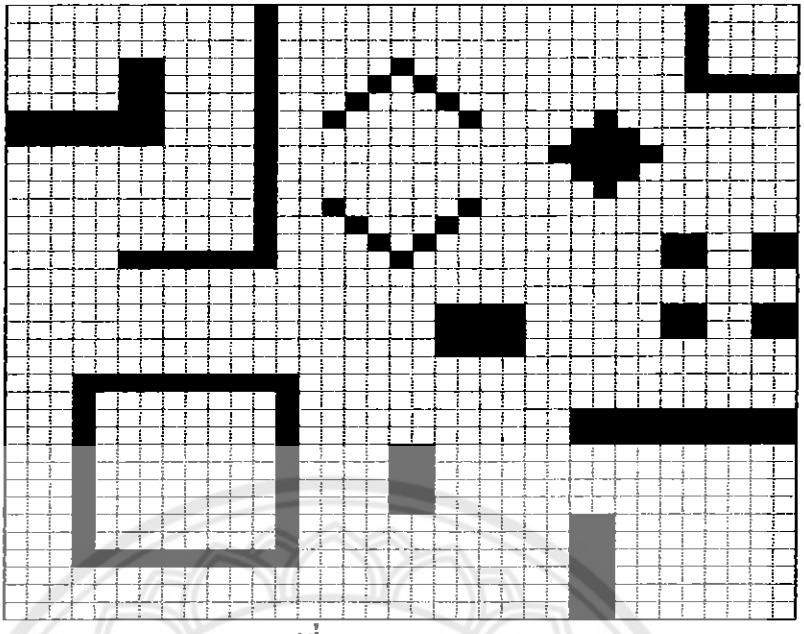
รูปที่ 2.20 ตัวอย่างการเลี้ยวเบนของคลื่นผ่านช่องแคบเดี่ยว

2.3.3 การวางแผนเส้นทางโดยใช้หลักการแพร่ของคลื่น (Path planning using wavefront technique)

สำหรับหัวข้อนี้จะอธิบายถึงขั้นตอนวิธีการ การวางแผนเส้นทางโดยใช้หลักการแพร่ของหน้าคลื่น ซึ่งจะมีขั้นตอนหลักๆ ดังนี้

<u>ขั้นตอนที่ 1</u>

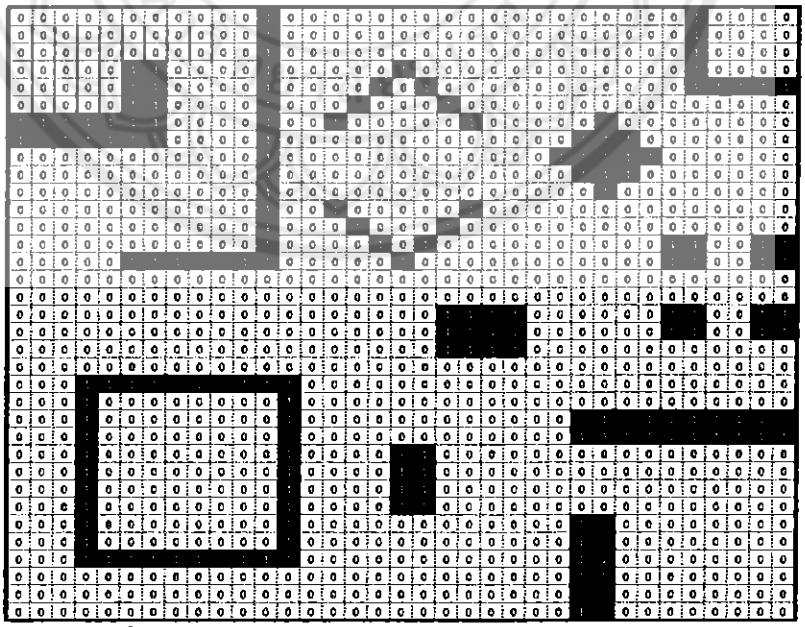
สร้างแผนที่ (Map) ที่ต้องการวางแผนเส้นทางโดยแต่ละส่วนของแผนที่จะต้องมีพิกัดอ้างอิง (x,y) ดังรูปที่ 2.21



รูปที่ 2.21 Grid on Map

ขั้นตอนที่ 2

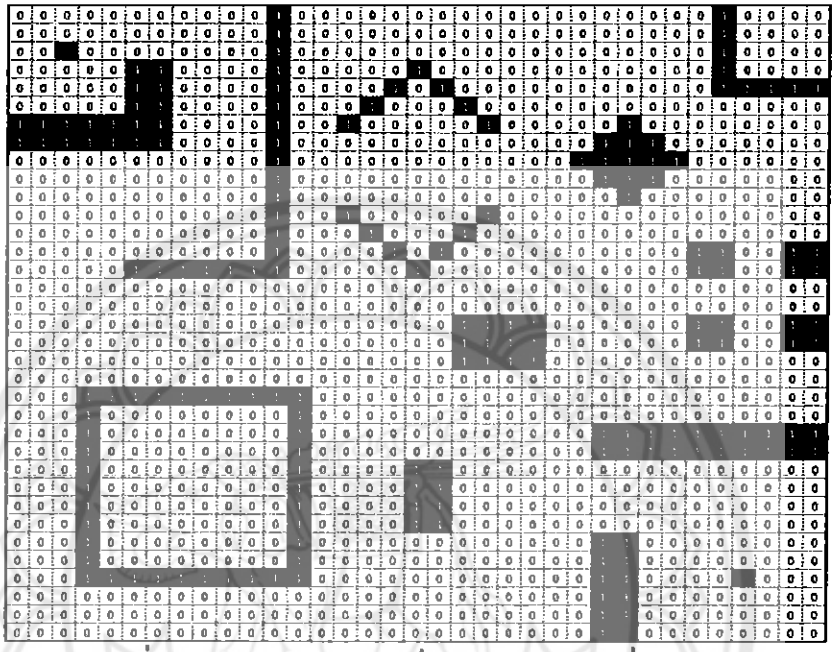
หลังจากสร้างแผนที่แล้ว จากนั้นก็ทำการกำหนดค่าส่วนของบริเวณต้องห้าม (Forbidden space) และบริเวณอิสระ (Free space) โดยจะกำหนดให้บริเวณต้องห้ามมีค่าเท่ากับ 1 และบริเวณอิสระมีค่าเท่ากับ 0 ดังรูปที่ 2.22



รูปที่ 2.22 กำหนดค่าของบริเวณต้องห้ามและบริเวณอิสระ

ขั้นตอนที่ 3

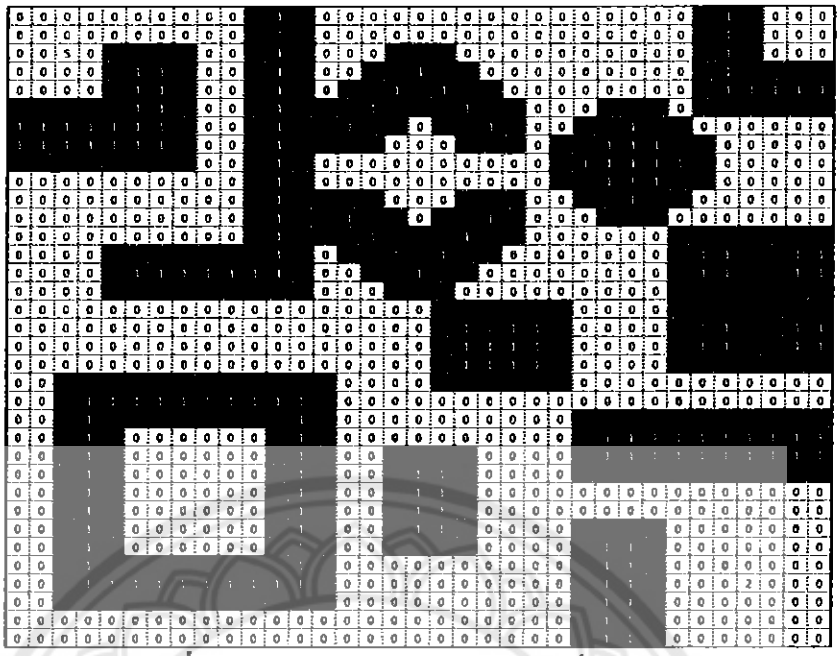
หลังจากกำหนดค่าของบริเวณต้องห้ามและบริเวณอิสระแล้ว จากนั้นจะทำการกำหนดจุดเริ่มต้น(Start point) และกำหนดจุดหมายที่ต้องการ (Goal point) โดยจะกำหนดให้จุดเริ่มต้นเป็น 'S' และกำหนดให้จุดหมายที่ต้องการมีค่าเท่ากับ 2 ดังรูปที่ 2.23



รูปที่ 2.23 กำหนดค่าของจุดเริ่มต้นและจุดหมายที่ต้องการ

ขั้นตอนที่ 4

หลังจากกำหนดจุดเริ่มต้นและจุดหมายที่ต้องการแล้ว จะสร้างการขยายตัวของสิ่งกีดขวาง (Expand) เพื่อเป็นการสร้างปลอดภัยให้กับหุ่นยนต์ ซึ่งในการขยายตัวนี้จะขึ้นอยู่กับความเหมาะสมตามสถานการณ์ โดยส่วนที่ขยายออกมานั้นจะกำหนดค่าเป็น 1 ซึ่งจะเปรียบเสมือนสิ่งกีดขวางนั่นเอง ดังรูปที่ 2.24 จะเห็นว่าบริเวณต้องห้ามจะมีเพิ่มขึ้น แต่ในความเป็นจริงแล้วพื้นที่ที่เพิ่มมาดังกล่าวเป็นส่วนของบริเวณอิสระนั่นเองเพื่อไม่ให้หุ่นยนต์เข้าใกล้สิ่งกีดขวางมากเกินไป

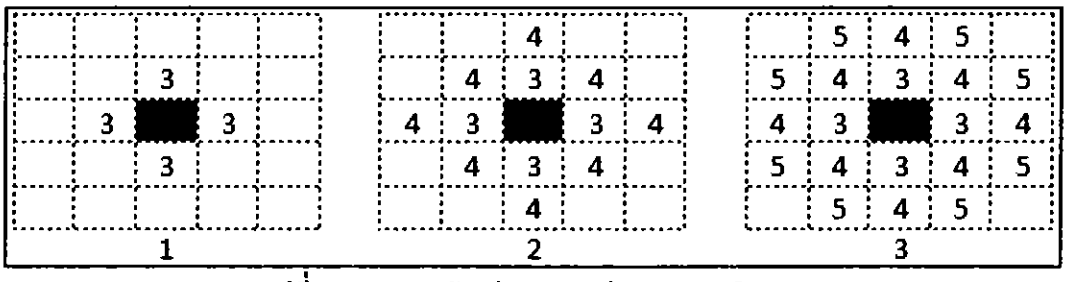


รูปที่ 2.24 แสดงการทำการขยายตัวของสิ่งกีดขวาง

ขั้นตอนที่ 5

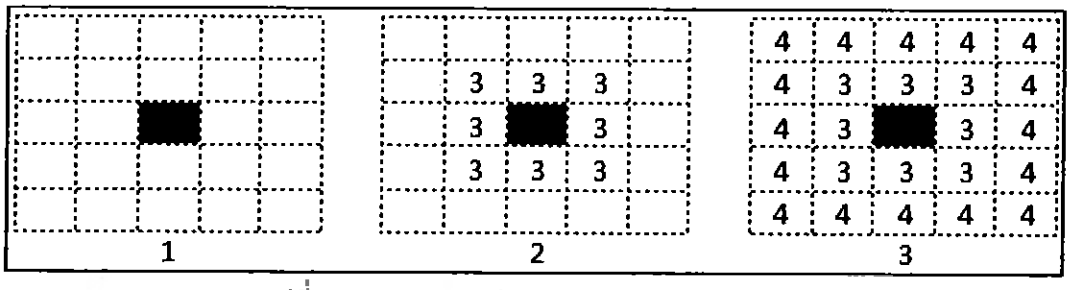
ในขั้นตอนนี้ถือเป็นขั้นตอนสำคัญของการวางแผนเส้นทางโดยการใช้หลักการแพร่ของหน้าคลื่นซึ่งลักษณะของการแพร่จะเป็นการเพิ่มค่าขึ้นเรื่อยๆ โดยเริ่มจากจุดหมายที่ต้องการแพร่กระจายออกเรื่อยๆจนสุดขอบเขตของแผนที่แต่จะไม่แพร่ในบริเวณต้องห้ามหรือบริเวณการขยายตัวของสิ่งกีดขวาง โดยรูปแบบการแพร่กระจายที่จะนำมาแสดงเป็นตัวอย่าง มีอยู่ 2 รูปแบบ ดังนี้

รูปแบบที่ 1 แพร่กระจาย 4 ทิศทาง (4 Direction) ในรูปแบบนี้จะเป็นการแพร่กระจายหรือเพิ่มค่าในตำแหน่งซ้าย (x-1,y) ,ตำแหน่งขวา (x+1,y), ตำแหน่งล่าง(x,y+1) และตำแหน่งบน (x,y-1) ของตำแหน่งปัจจุบัน (x,y) ดังรูปที่ 2.25



รูปที่ 2.25 แสดงตัวอย่างการแพร่กระจาย 4 ทิศทาง

รูปแบบที่ 2 แพร่กระจาย 8 ทิศทาง (8 Direction) ในรูปแบบนี้จะเป็นการแพร่กระจายออกรอบๆ จากตำแหน่งปัจจุบัน โดยจะไม่ทำการแพร่ไปยังตำแหน่งที่ทำการแพร่แล้ว ดังรูปที่ 2.26



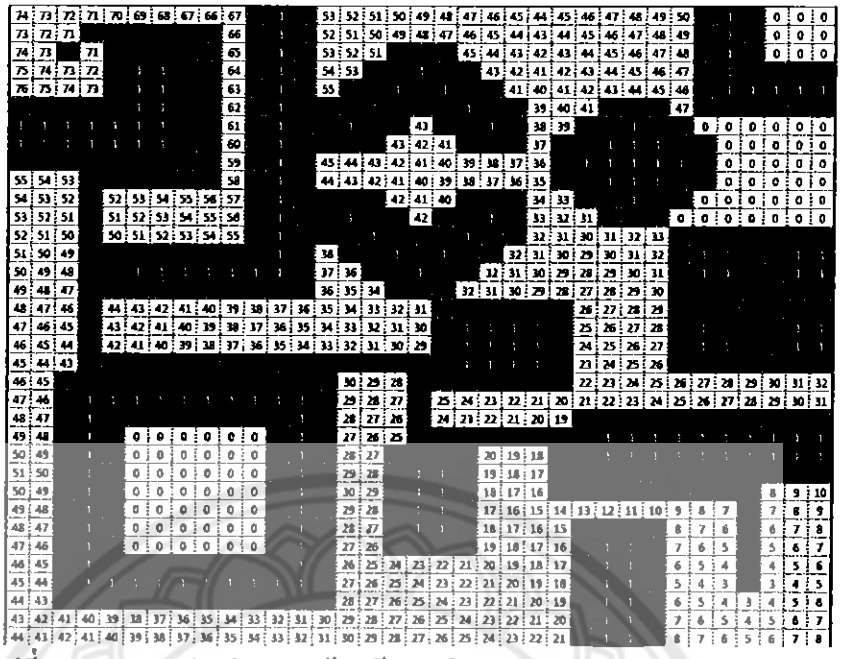
รูปที่ 2.26 แสดงตัวอย่างการแพร่กระจาย 8 ทิศทาง

ขั้นตอนที่ 6

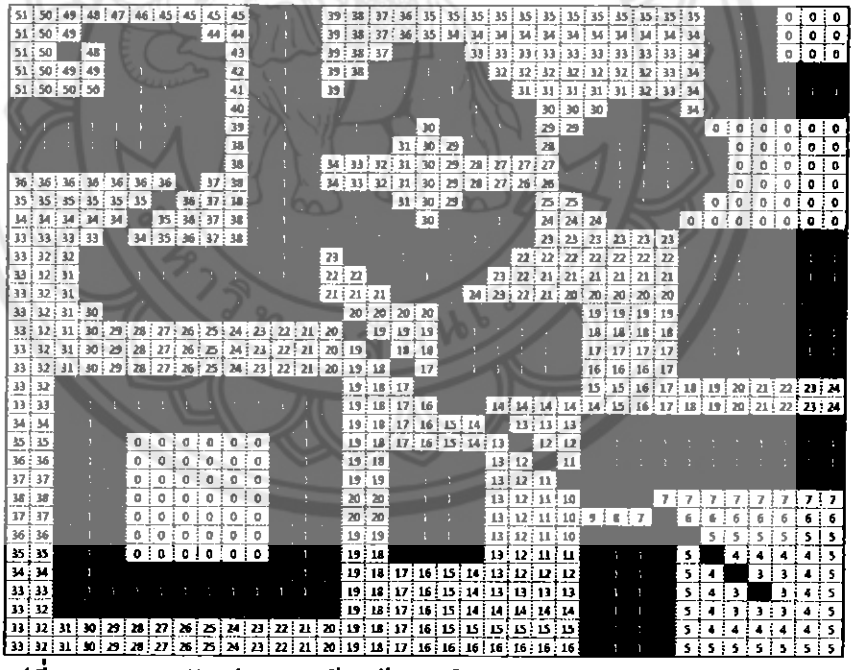
ขั้นตอนนี้จะเป็นการสร้างเส้นทางจากจุดเริ่มต้นไปยังจุดหมายที่กำหนดไว้ โดยเงื่อนไขต่อไปนี้

- ถ้าตำแหน่งต่อไปอยู่ติดกับพิกัดปัจจุบันและมีค่าของพิกัดน้อยกว่าค่าพิกัดปัจจุบัน ให้สร้างเส้นทางได้
- จะไม่มีการสร้างเส้นทางในพิกัดที่มีค่าเท่ากันหรือมากกว่า
- สร้างเส้นทางจนกว่าจะถึงพิกัดที่ต้องการ (พิกัดที่มีค่าเท่ากับ 2)

ดังแสดงในรูปที่ 2.27 และ 2.28 ซึ่งจะเป็นรูปแบบการแพร่กระจายแบบ 4 ทิศทางและ 8 ทิศทาง



รูปที่ 2.27 แสดงตัวอย่างการสร้างเส้นทางในรูปแบบการแพร่กระจาย 4 ทิศทาง

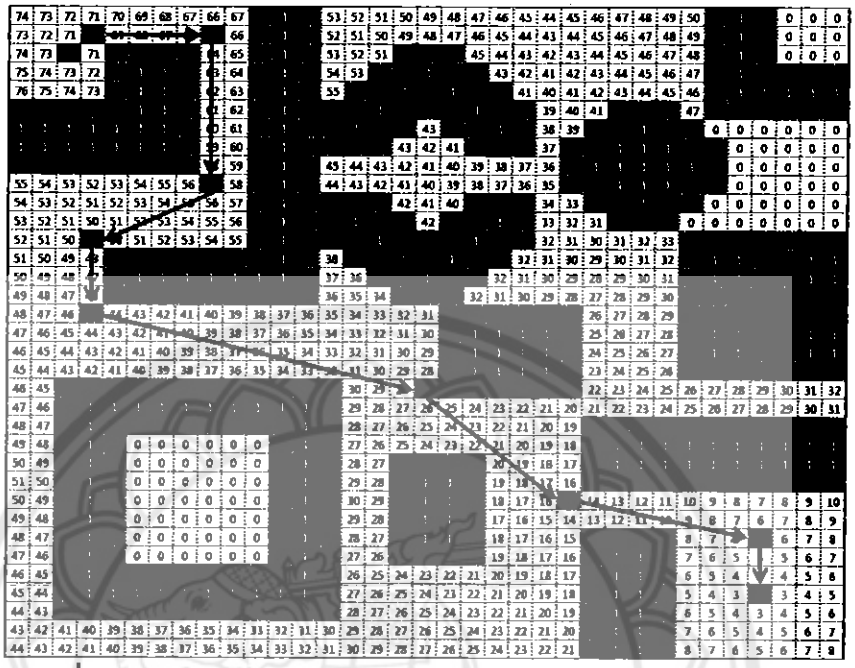


รูปที่ 2.28 แสดงตัวอย่างการสร้างเส้นทางในรูปแบบการแพร่กระจาย 8 ทิศทาง

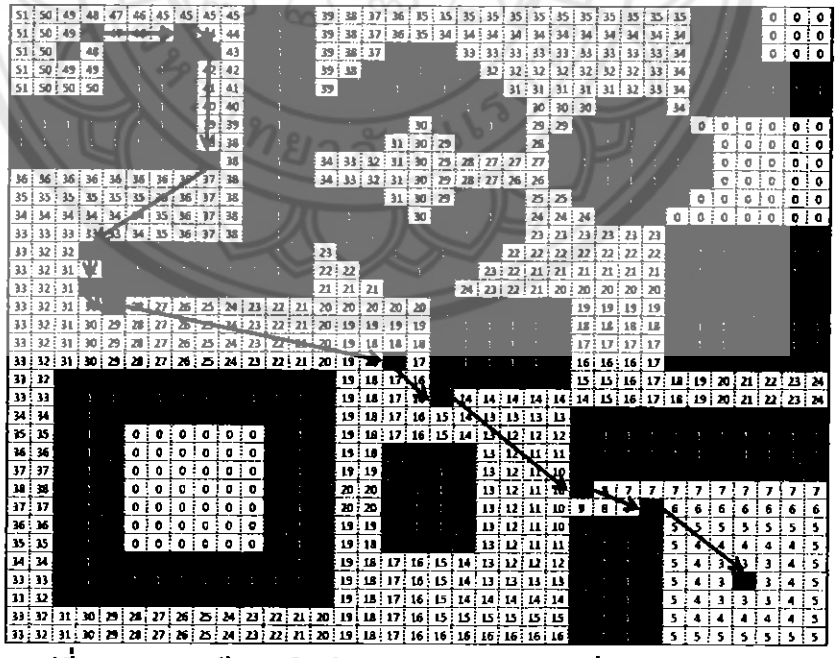
<u>ขั้นตอนที่ 7</u>

ขั้นตอนนี้จะเป็นขั้นตอนสุดท้ายของการวางแผนเส้นทางโดยใช้หลักการแพร่ของหน้าคลื่น ซึ่งจะเป็นการปรับปรุงเส้นทางจากขั้นตอนที่ 6 เพื่อให้มีประสิทธิภาพมากขึ้น โดยจะนำเส้นทางมาวิเคราะห์ดูว่า มีพิกัดใดบ้าง ที่สามารถเชื่อมต่อถึงกันได้โดยไม่เป็นเส้นทางเดิมและมี

ระยะทางสั้นกว่าเส้นทางเดิม โดยจะเรียกพิกัดที่สามารถนำมาสร้างเส้นทางใหม่นี้ว่า จุดสนใจ (Waypoints) ดังรูปที่ 2.29 และรูปที่ 2.30



รูปที่ 2.29 แสดงเส้นทางใหม่หลังจากทำการวิเคราะห์หาชุดของจุดสนใจ ในรูปแบบการแพร่กระจาย 4 ทิศทาง



รูปที่ 2.30 แสดงเส้นทางใหม่หลังจากทำการวิเคราะห์หาชุดของจุดสนใจ ในรูปแบบการแพร่กระจาย 8 ทิศทาง

2.4 ภาษาวิชวลซีชาร์ป(Visual C#)

วิชวลซีชาร์ป (Visual C#) หรือ VC#เป็นภาษาคอมพิวเตอร์ที่พัฒนาโดยบริษัทไมโครซอฟท์ โดยใช้รากฐานของภาษา C/C++ เป็นหลัก ดังนั้นรูปแบบโครงสร้างทางภาษาถึงคล้ายกับ C/C++ แต่ได้ลดความสลับซับซ้อนลง ซึ่งทำให้ภาษา C# นั้นกลายเป็นภาษาที่เรียนรู้ได้ง่ายและสามารถทำงานได้อย่างมีประสิทธิภาพ

ภาษา C# นั้นเกิดขึ้นมาพร้อมกับเทคโนโลยี .NET ดังนั้นการทำงานของ C# จึงขึ้นกับ .NET Framework เป็นหลักโดยมีชุดเครื่องมือที่ใช้ในการพัฒนาแอปพลิเคชั่นด้วยภาษา C# เรียกว่า Visual C# ซึ่งสามารถใช้ในการพัฒนาแอปพลิเคชั่นทั่วๆไปในระดับเดียวกันกับ Visual Basic ซึ่งภาษา C# ในปัจจุบันกำลังได้รับความนิยมมากขึ้นเรื่อยๆ เพราะโครงสร้างของ C# นั้นสั้น, กระชับและเข้าใจได้ง่ายกว่า ซึ่งไม่เพียงสร้างแอปพลิเคชั่นบนวินโดว์เท่านั้น แต่ยังสามารถสร้างแอปพลิเคชั่นอื่นๆ ได้อีกหลากหลาย เช่น Web Application (ASP.NET), Smart Device, WPF, Silverlight เป็นต้น

บทที่ 3
วิธีการดำเนินงาน

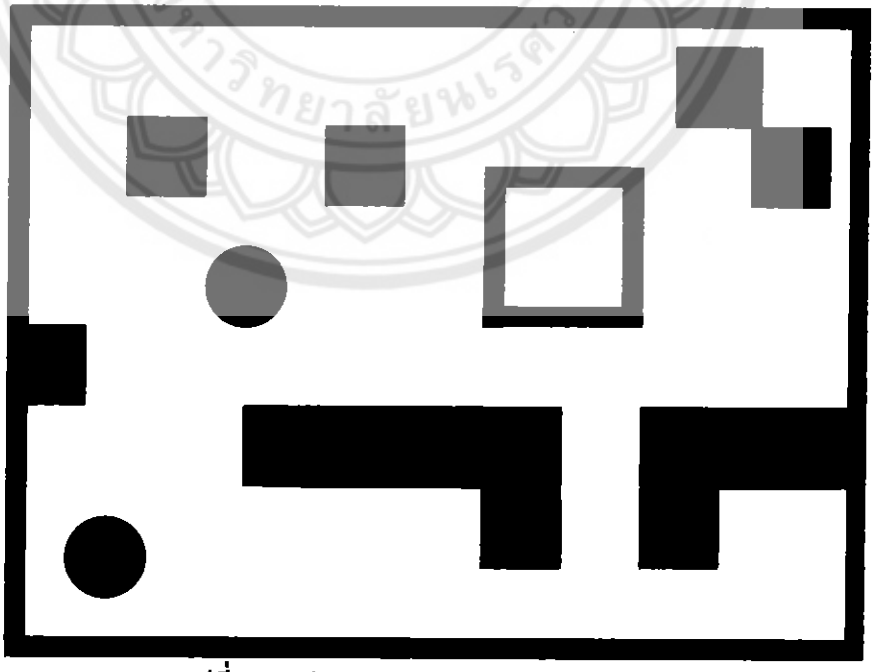
3.1 การสร้างแผนที่สำหรับป้อนให้โปรแกรมที่พัฒนาขึ้น

3.1.1 ศึกษาลักษณะสถานที่

การศึกษารายละเอียดต่างๆของสถานที่ที่ต้องการให้หุ่นยนต์เคลื่อนที่ โดยทำการเก็บรายละเอียดของสถานที่จริงประกอบด้วยการวัดระยะต่างๆทั้งพื้นที่และสิ่งกีดขวางที่อยู่ในสถานที่นั้น เพื่อที่จะนำค่าที่วัดได้ไปสร้าง Input Map สำหรับป้อนให้โปรแกรม เพื่อดำเนินการต่อไป

3.1.2 วาดแผนที่

หลังจากศึกษารายละเอียดของสถานที่จริง ขั้นตอนต่อไปคือการนำค่าระยะทางที่วัดได้ทั้งพื้นที่และสิ่งกีดขวางต่างๆของสถานที่นั้น มาสร้าง Input Map ซึ่งในที่นี้ใช้โปรแกรม Microsoft Visio 2010 โดยขนาดของรูปต้องมีสัดส่วนที่ใกล้เคียงกับสถานที่จริง ซึ่งก็คือกว้างและยาวต้องมีมาตราส่วนเดียวกันตัวอย่าง เช่น จากสถานที่แห่งหนึ่ง กว้าง 4 เมตรสูง 3 เมตรและมีสิ่งกีดขวางสามารถสร้างเป็นแผนที่ซึ่งมีขนาด กว้าง 1024 pixel สูง 768 pixel (อัตราส่วน 4:3) ได้ ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 รูป Input Map ขนาด 1024*768 pixel

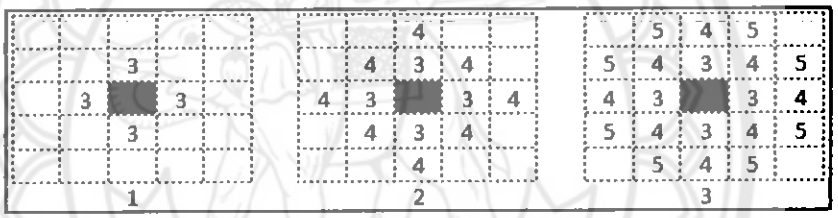
3.2 ออกแบบและพัฒนาโปรแกรม

3.2.1 อัลกอริทึมที่นำมาประยุกต์ใช้

3.2.1.1 Wavefront Algorithm

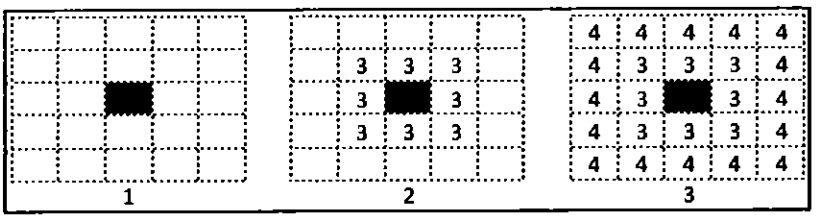
Wavefront Algorithm เป็นอัลกอริทึมที่เลียนแบบการแพร่ของคลื่น ที่เกิดจากคลื่นเคลื่อนที่ไปพบสิ่งกีดขวางทำให้คลื่นส่วนหนึ่งอ้อมบริเวณของสิ่งกีดขวางแผ่ไปยังด้านหลังของสิ่งกีดขวางนั้น

การเลียนแบบหลักการแพร่ของหน้าคลื่น ซึ่งหลักการแพร่จะเป็นการเพิ่มค่าขึ้นเรื่อยๆ โดยเริ่มจากจุดหมายปลายทางแพร่กระจายออกไปเรื่อยๆจนสุดขอบเขตของแผนที่ การแพร่กระจายแบ่งออกเป็นสองแบบคือ การแพร่กระจาย 4 ทิศทาง (4 Direction) และการแพร่กระจาย 8 ทิศทาง (8 Direction) ในส่วนของการแพร่แบบ 4 ทิศทางนั้น จะเป็นการแพร่กระจายหรือเพิ่มค่าในตำแหน่งซ้าย (x-1,y) ,ตำแหน่งขวา (x+1,y), ตำแหน่งล่าง (x,y+1) และตำแหน่งบน (x,y-1) ของตำแหน่งปัจจุบัน (x,y) ดังรูปที่ 3.2



รูปที่ 3.2 แสดงตัวอย่างการแพร่กระจาย 4 ทิศทาง

สำหรับการแพร่กระจายแบบ 8ทิศทาง (8 Direction)นั้นจะเป็นการแพร่กระจายออกรอบๆ จากตำแหน่งปัจจุบัน โดยจะไม่ทำการแพร่ไปยังตำแหน่งที่ทำการแพร่แล้ว และจะไม่แพร่ในบริเวณต้องห้าม การแพร่กระจายในรูปแบบนี้จะเป็นการเพิ่มค่าในตำแหน่งซ้าย (x-1,y) , ตำแหน่งขวา (x+1,y), ตำแหน่งล่าง(x,y+1),ตำแหน่งบน(x,y-1), ตำแหน่งบนซ้าย (x-1,y-1), ตำแหน่งบนขวา (x+1,y-1),ตำแหน่งล่างซ้าย (x-1,y+1), ตำแหน่งล่างขวา (x+1,y+1) ของตำแหน่งปัจจุบัน (x,y) ดังรูปที่ 3.3



รูปที่ 3.3 แสดงตัวอย่างการแพร่กระจาย 8 ทิศทาง

3.2.1.2 การสร้างเส้นทางและการหาเส้นทางที่เหมาะสมจากการใช้ Wavefront algorithm

เป็นการสร้างเส้นทางและหาเส้นทางที่เหมาะสมซึ่งเป็นการวนตรวจสอบตามเข็มนาฬิกาเพื่อหาเส้นทางที่สั้นที่สุดที่ได้จากการแพร่ (Wavefront algorithm)

เปรียบเทียบค่าใน pixel เพื่อหาตำแหน่งต่อไปที่จะเคลื่อนที่

Pseudo code:

```
Point minCW(Point p, intwidth, inthigh)
{
    Point minPoint=null
    int compare, north, north_east, east, south_east,
      south, south_west, west, north_west = width * high
    int current_value = value_of_point[X,Y]

    if(the north point and that point is in the picture)
        north = value of point[p.X, p.Y-1]
    if(the north east point and that point is in the picture)
        north_east = value of point[p.X+1, p.Y-1]
    if(the east point and that point is in the picture)
        east = value of point[p.X+1, p.Y]
    if(the south east point and that point is in the picture)
        south_east = value of point[p.X+1, p.Y+1]
    if(the south point and that point is in the picture)
        south = value of point[p.X, p.Y+1]
    if(the south west point and that point is in the picture)
        south_west = value of point[p.X-1, p.Y+1]
    if(the west point and that point is in the picture)
        west = value of point[p.X-1, p.Y]
    if(the north west point and that point is in the picture)
    north_west = value of point[p.X-1, p.Y-1]

    //เปรียบเทียบค่าทางทิศเหนือโดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง
```

```
if(north > 1 and north != current_value and north < compare)
{
        compare = north
        min_point = new point(p.X, p.Y-1)
}
```

//เปรียบเทียบค่าทางทิศตะวันออกเฉียงเหนือโดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง

```
if(north_east> 1 and north_east != current_value and north_east< compare)
{
        compare = north_east
        minPoint = new point(p.X, p.Y-1)
}
```

//เปรียบเทียบค่าทางทิศตะวันออกโดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง

```
if(east > 1 and east != current_value and east < compare)
{
        Compare = east
        minPoint = new point(p.X+1, p.Y)
}
```

//เปรียบเทียบค่าทางทิศตะวันออกเฉียงใต้โดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง

```
if(south_east and south_east != current_value and south_east< compare)
{
        compare = south_east
        minPoint = new point(p.X+1, p.Y+1)
}
```

//เปรียบเทียบค่าทางทิศใต้โดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง

```
if(south > 1 and south != current_value and south < compare)
{
        compare = south
        minPoint = new point(p.X, p.Y+1)
}
}
```

```
//เปรียบเทียบค่าทางทิศตะวันตกเฉียงใต้โดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง
if(south_west> 1 and south_west != current_value and south_west< compare)
{
        compare = south_west
        minPoint = new point(p.X-1, p.Y+1)
}
//เปรียบเทียบค่าทางทิศตะวันตกโดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง
if(west > 1 and west != current_value and west < compare)
{
        compare = west
        minPoint = new point(p.X-1, p.Y)
}
//เปรียบเทียบค่าทางทิศตะวันตกเฉียงเหนือโดยมีค่ามากกว่าหนึ่ง และน้อยกว่ากว้างคูณสูง
if(north_west> 1 and north_west != current_value and north_west< compare)
{
        Compare = north_west
        minPoint = new point(p.X-1, p.Y-1)
}
```

การเลือกเส้นทางจากจุด Start ไปยังจุด Goal

Pseudo code:

```
Point[] generatePath(point start, point goal, int width, int high)
{   //ถ้าค่าที่อยู่ในจุด (x,y) นั้น ไม่เท่ากับค่าจุด goal ซึ่งเท่ากับ 2
    while(value of point[p.X, p.Y] not equal 2)
    {   //เปรียบเทียบค่าใน pixel เพื่อหาตำแหน่งต่อไปที่จะเคลื่อนที่
        point p = minCW(p, width, high)
        if(p = null)
            break;
        Add p to tmp
    }
}
```

3.2.1.3 Line Drawing Algorithm

Line drawing algorithm เป็นอัลกอริทึมที่ใช้สำหรับใช้วาดเส้นจากจุดหนึ่งไปยังอีกจุดหนึ่ง เช่นจากจุดที่หนึ่ง (x1,y1) ไปยังจุดที่สอง (x2,y2) แล้วทำการหาระยะห่างระหว่างจุดสองจุด จากนั้นทำการหาทิศทางจากจุดที่หนึ่งไปยังจุดที่สองว่า ทิศทางของจุดที่สองอยู่ในทิศทางใด จากนั้นทำการวาดเส้นจากจุดที่หนึ่งไปยังจุดที่สอง [9]

การนำ Line drawing algorithm มาใช้ จะใช้สำหรับการหาเส้นทางย่อยๆ (Series of Waypoint) เป็นการสร้างเส้นทางจากเส้นทางที่ได้จากการแพร่ (Wavefront algorithm) ซึ่งจะทำให้ได้ระยะทางที่สั้นกว่าเส้นทางที่ได้จากการแพร่ โดยจะเช็คตั้งแต่จุด Start ไปยังจุด Goal และจากจุด Goal มายังจุด Start จากนั้นเปรียบเทียบหาเส้นทางที่สั้นที่สุดเพื่อเลือกเส้นทางนั้น

Pseudo Code:

```
Point[] generateOptimalPath(Point[] commonPath)
{
   inti = 1                    //กำหนดค่า i เท่ากับ 1
   point optimal = null                  //กำหนดจุด way point แรก เป็นค่าว่าง
   pointcheckOptimal = null  //กำหนดจุดที่ใช้สำหรับเช็ค waypoint ถัดไป เป็นค่าว่าง
   //กำหนด list point สำหรับจุด Start ไปยังจุด Goal
       List<Point> tmpS2G = new List<Point>();
   //กำหนด list point สำหรับจุด Goal มายังจุด Start
       List<Point> tmpG2S = new List<Point>();

   //กำหนดให้จุด optimal คือจุด Start ที่ได้จากการสร้างเส้นทางที่ได้จากการแพร่ซึ่งเป็น way point แรก
       tmpS2G.Add(commonPath[0]);
   optimal = new point(commonPath[0].X, commonPath[0].Y)

   //เรียกใช้ฟังก์ชัน CheckByDrawLineเพื่อเช็คจุดสองจุด โดยเริ่มจากจุดstart และจุดสุดท้ายของเส้นทางที่ได้จากการแพร่ ซึ่งก็คือ จุด Goal
   while(CheckByDrawline(Optimal, new point(commontPath[commonPath.Length-1].X, [commonPath.Length-1].Y)) is false)
   {  //วนลูปเช็คและขยับจุด commontPath ลงมาเรื่อยๆ จนกระทั่งการเรียกใช้ฟังก์ชั่น CheckByDrawline เป็นจริง
```

```
for(int n=commonPath.Length – 1;n > 1; n--)
        {    //ขยับจุด commonPath
        checkOptimal = new point(commonPath[n].X,commonPath[n].Y)
            if(CheckByDrawline(optimal, checkOptimal) is true)
                {    //ขยับจุด optimal ให้มาอยู่ในตำแหน่งที่
                commonPath[n].X, commonPath[n].Y
                optimal = new point(commonPath[n].X, commonPath[n].Y)
                Add commonPath[n].X, commonPath[n].Y to tmpS2G
                i = n
                break
                }
        }
    }
    tmpS2G.Add(commonPath[commonPath.Length - 1]);
    tmpS2G = AverageWayPoint(tmpS2G);
    OptimalPath = new Point[tmpS2G.Count];
    Array.Copy(tmpS2G.ToArray(), OptimalPath, tmpS2G.Count);

    //กำหนดให้จุด optimal คือจุด Goal ที่ได้จากการสร้างเส้นทางที่ได้จากการแพร่
ซึ่งเป็น way point แรก
    tmpG2S.Add(commonPath[commonPath.Length - 1]);
    Optimal = new Point(commonPath[commonPath.Length - 1].X,
commonPath[commonPath.Length - 1].Y);
```

ทำในลักษณะเดียวกันกับการเช็คจากจุด Start มายังจุด Goal แต่เป็นการเช็คจุด Goal มายังจุด Start

//คำนวณหาระยะทางระหว่างการเช็คจากจุด Start ไปยังจุด Goal กับการเช็คจากจุด Goal ไปยังจุด Start

//ถ้าระยะทางจากจุด Start ไปยังจุด Goal มีระยะทางน้อยกว่าระยะทางจากจุด Goal ไปยังจุด Start ให้เลือกใช้เส้นทางจากจุด Start ไปยังจุด Goal

//ถ้าระยะทางจากจุด Goal ไปยังจุด Startมีระยะทางน้อยกว่าระยะทางจากจุด Startไปยังจุด Goalให้เลือกใช้เส้นทางจากจุด GoalไปยังจุดStart

```
}
```

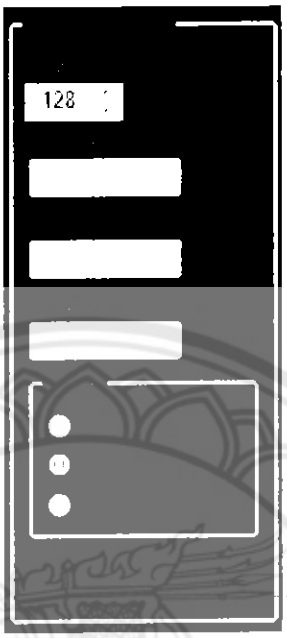
3.2.2 ออกแบบและพัฒนาหน้าต่างของโปรแกรม

หลังจากทำการออกแบบจะได้รูปแบบหน้าต่างของโปรแกรม ดังรูปที่ 3.4

รูปที่ 3.4 รูปแบบหน้าต่างโปรแกรม

การออกแบบหน้าต่างโปรแกรมจะแบ่งออกเป็น 4 ส่วนหลักๆ ได้แก่

3.2.2.1 ส่วนกำหนดข้อมูลของแผนที่ ดังรูปที่ 3.5

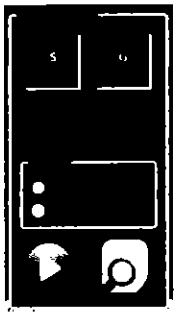


รูปที่ 3.5 ส่วนกำหนดข้อมูลแผนที่

โดยที่

- Grayscale คือ ค่าที่ใช้สำหรับปรับภาพขาวดำ
- Map Width คือ ความกว้างของพื้นที่การทำงานจริง ในระนาบ 2 มิติ
- Map Height คือ ความสูงของพื้นที่การทำงานจริงในระนาบ 2 มิติ
- Object Radius คือ ขนาดรัศมีของหุ่นยนต์ในระนาบ 2 มิติ

3.2.2.2 ส่วนกำหนดจุด start และจุด goal ดังรูปที่ 3.6

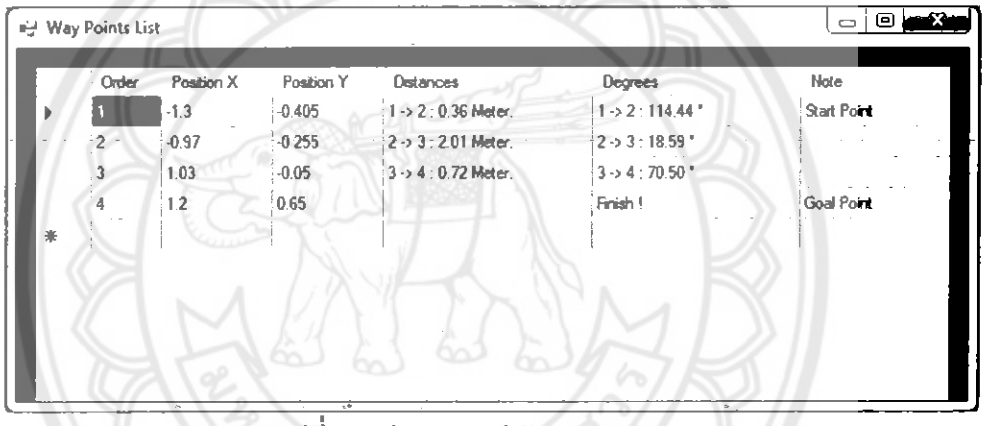


รูปที่ 3.6 ส่วนกำหนดจุด start point และจุด goal point

โดยที่

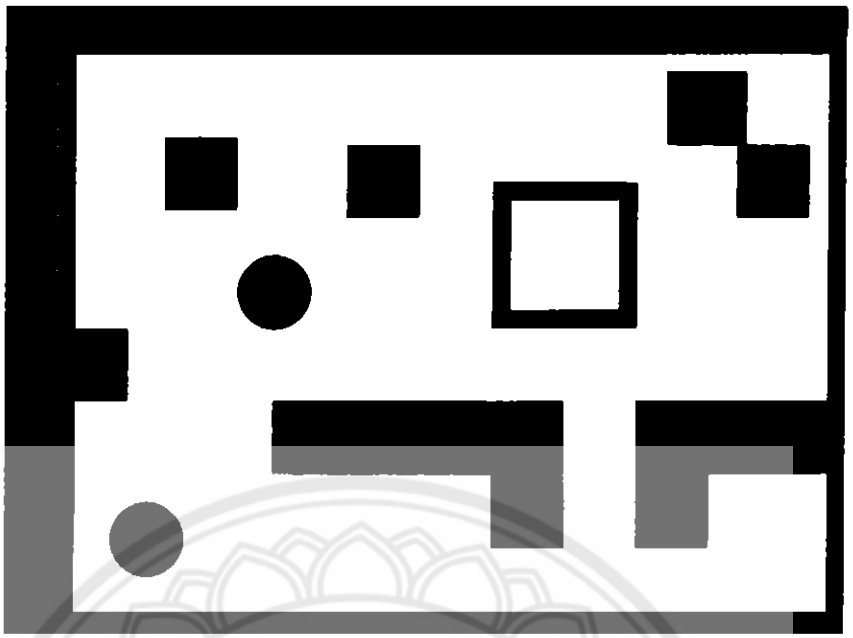
- S แทนสัญลักษณ์จุด start
- G แทนสัญลักษณ์จุด goal
- Start point คือ พิกัดของจุด start
- Goal point คือ พิกัดของจุด goal
- 4 Direction คือ การเพร่แบบ 4 ทิศทาง
- 8 Direction คือ การเพร่แบบ 8 ทิศทาง

3.2.2.3 ส่วนแสดงพิกัด เป็นส่วนแสดงพิกัดของแต่ละ waypoint ทั้งหมดที่ได้จากการรันโปรแกรม ดังรูปที่ 3.7



รูปที่ 3.7 ส่วนแสดงพิกัดของ Waypoint

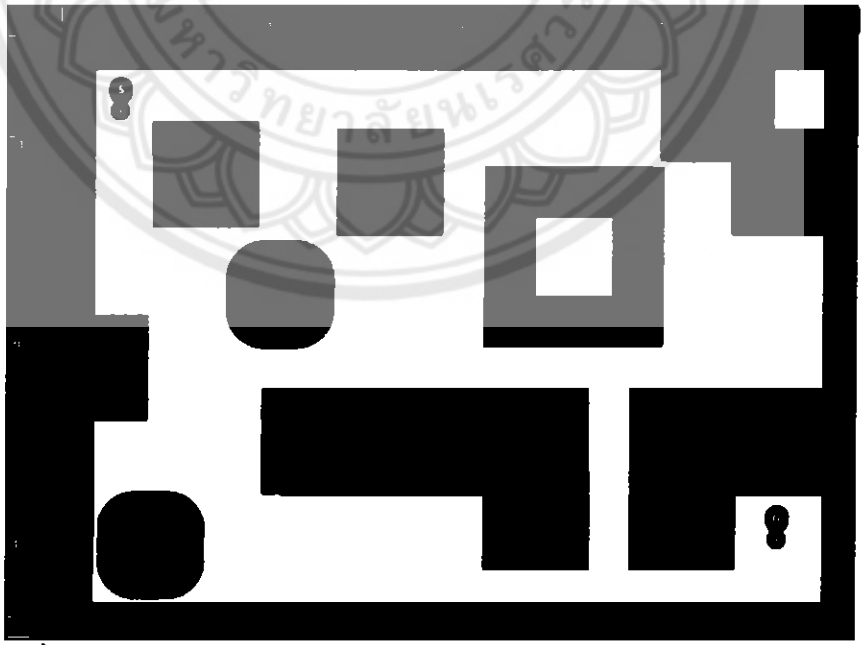
3.2.2.4 ส่วนแสดงแผนที่ เป็นส่วนที่แสดงแผนที่ หลังจาก Import เข้ามาในโปรแกรม ซึ่งในส่วนนี้จะแสดงการทำงานต่างๆของโปรแกรม ทั้งการกำหนดจุด start และจุด goal การแสดงเส้นทาง การแสดงเส้นทางย่อยๆ (A Series of Waypoints) ดังรูปที่ 3.8



รูปที่ 3.8 ส่วนแสดงแผนที่ (Input Map)

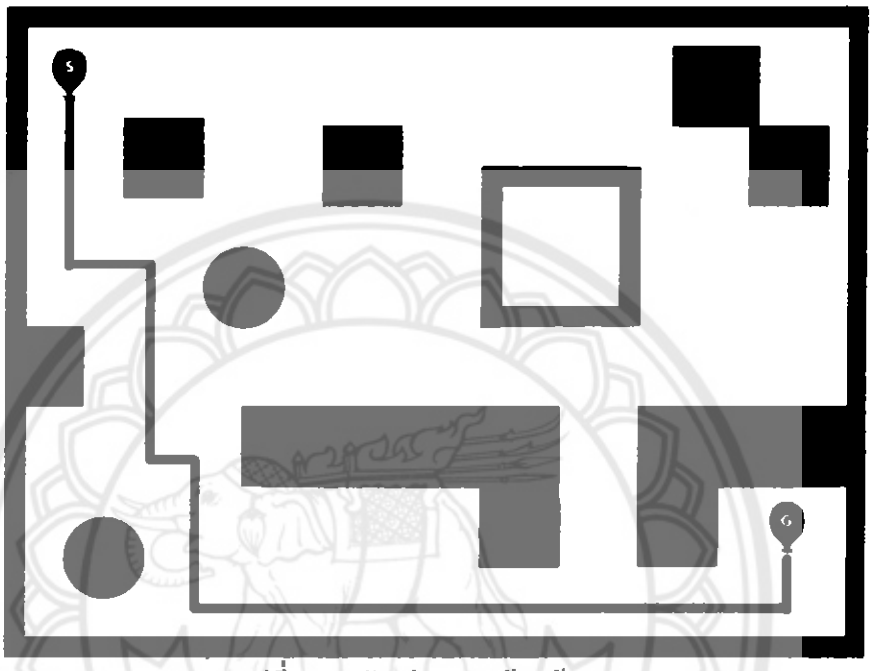
หลักการทำงานของโปรแกรม

เมื่อนำแผนที่เข้ามาในโปรแกรม และใส่ข้อมูลแผนที่ลงในโปรแกรมเรียบร้อยแล้ว โปรแกรมจะทำการขยายสิ่งกีดขวางเพื่อสร้างความปลอดภัยให้กับตัวหุ่นยนต์ จากนั้นให้ผู้ใช้งานกำหนดจุดเริ่มต้น (Start point) และจุดหมายปลายทาง (Goal point) ดังตัวอย่างรูปที่ 3.9



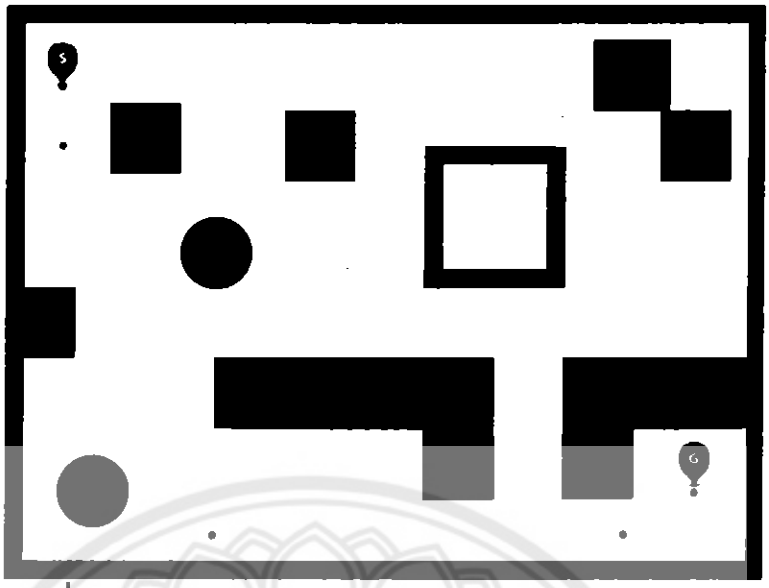
รูปที่ 3.9 กำหนดจุดเริ่มต้น (Start point) และจุดปลายทาง (Goal point) ในแผนที่

เมื่อกำหนดจุดเริ่มต้นและจุดปลายทางเรียบร้อยแล้ว ผู้ใช้งานจะสามารถเลือกลักษณะการแพร่กระจายได้ทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทาง และโปรแกรมจะทำการแพร่โดยการใช้ Wavefront Algorithm จากจุดหมายปลายทางมายังจุดเริ่มต้น จากนั้นโปรแกรมจะทำการหาเส้นทางที่สั้นและปลอดภัยแล้วทำการสร้างเส้นทาง ดังตัวอย่างรูปที่ 3.10



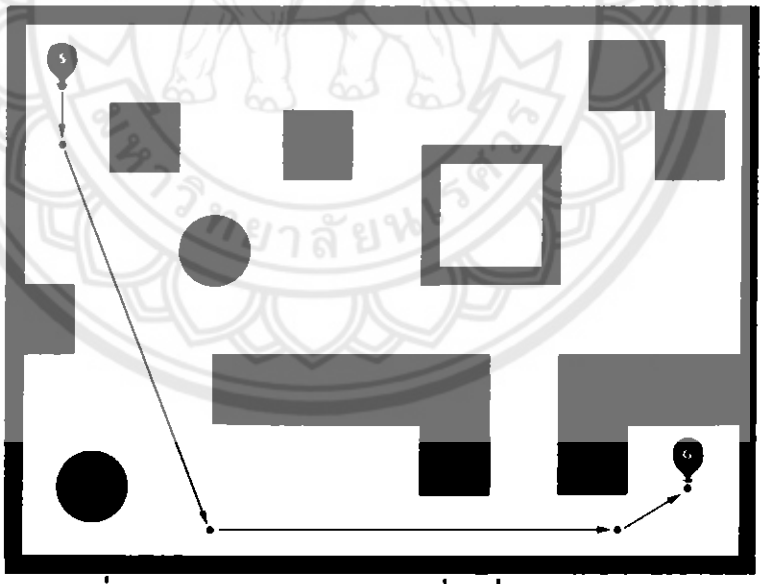
รูปที่ 3.10 ตัวอย่างการสร้างเส้นทาง

เมื่อโปรแกรมค้นหาเส้นทางที่สั้นที่สุดและปลอดภัยแล้ว โปรแกรมจะทำการสร้างเป้าหมายย่อย (A Series of Waypoint) เพื่อเพิ่มประสิทธิภาพในการสร้างการเคลื่อนที่ให้กับหุ่นยนต์ ซึ่งจะทำให้เส้นทางมีระยะทางสั้นมากกว่าเดิม ดังตัวอย่างรูปที่ 3.11



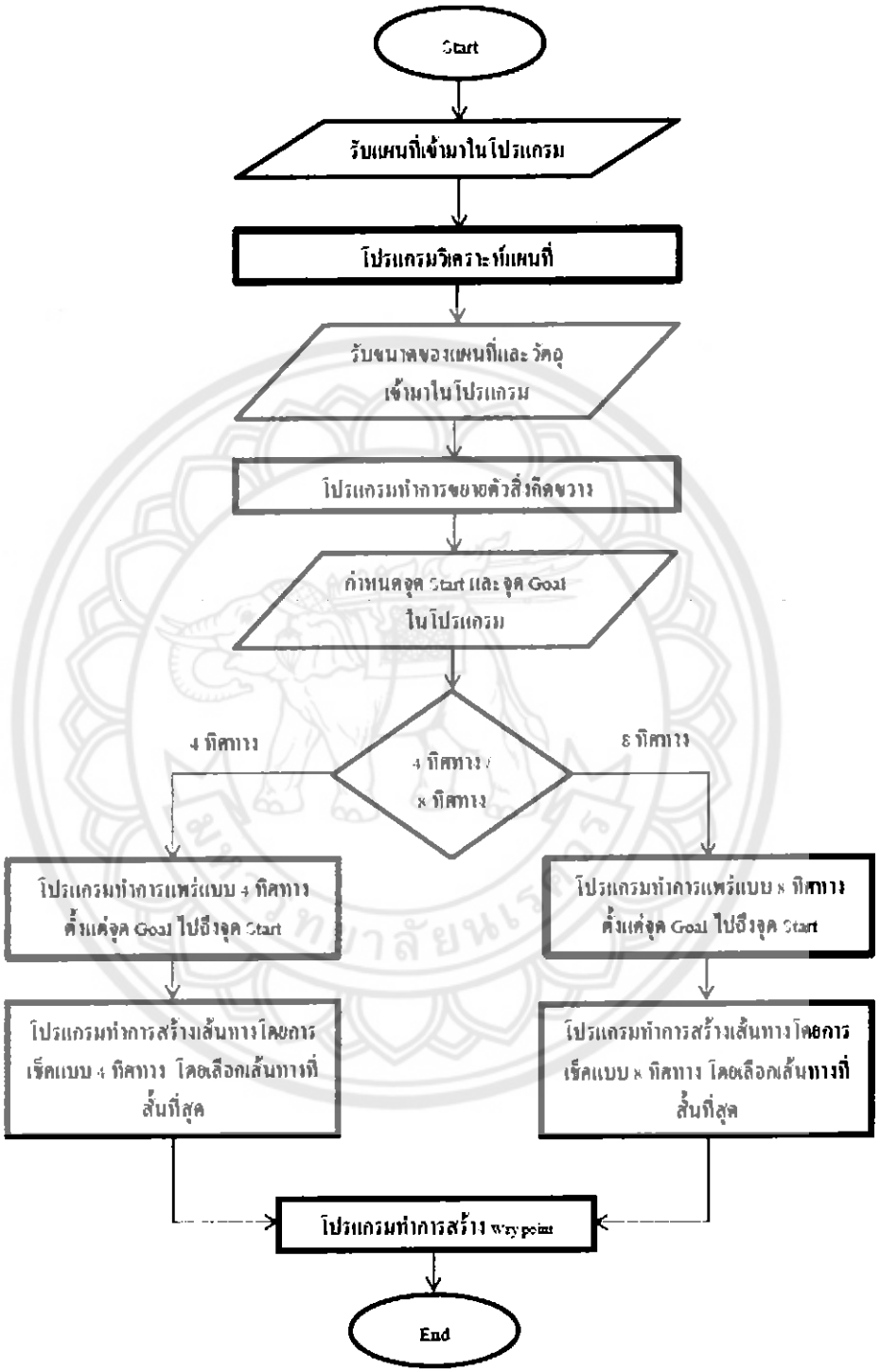
รูปที่ 3.11 ตัวอย่างการสร้างเส้นทางย่อยๆ (A Series of Waypoint)

จะเห็นได้ว่าหากหุ่นยนต์เคลื่อนที่ตามเส้นทางย่อยๆ หุ่นยนต์จะเคลื่อนที่**โดยใช้ระยะทางที่สั้นกว่าเดิม** ดังตัวอย่างรูปที่ 3.12



รูปที่ 3.12 ตัวอย่างการแสดงการเคลื่อนที่ตามเส้นทางย่อยๆ

3.3 แนวคิดของระบบโดยรวม



รูปที่ 3.13 แนวคิดของโปรแกรม

บทที่ 4
ผลการทดลอง

4.1 แผนการทดสอบโปรแกรม

ในขั้นตอนการทดสอบโปรแกรม เป็นขั้นตอนการนำโปรแกรมที่พัฒนาขึ้นมาทำการทดสอบการทำงานในด้านต่างๆ ว่ามีประสิทธิภาพหรือไม่ มีจุดบกพร่องที่ใดบ้าง โดยในการทดสอบได้แบ่งเป็น 2 ส่วน โดยส่วนแรกคือการทดสอบการใช้งานโปรแกรม เป็นการทดสอบการแสดงผล Graphics User Interface (GUI) มีการแสดงผลที่ครบถ้วน โปรแกรมใช้งานได้ง่ายและไม่ซับซ้อน ส่วนที่สองคือการทดสอบประสิทธิภาพของโปรแกรม เป็นการทดสอบการทำงานของโปรแกรม โดยนำตัวอย่างแผนที่ (Input Map) ในขนาดต่างๆ มาทำการทดลองและดูผลลัพธ์ที่ได้ตรงตามจุดประสงค์ของการพัฒนาโปรแกรมนี้ขึ้น

การทดสอบการทำงานของโปรแกรม จะทำการทดสอบกับตัวอย่างแผนที่ขนาดต่างๆ ที่สร้างขึ้น และดูผลลัพธ์ของการทำงานที่ได้ โดยผลลัพธ์ที่ได้ของโปรแกรมคือ เป้าหมายย่อย (Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา ตั้งแต่จุดเริ่มต้นจนถึงจุดหมายปลายทาง เพื่อปฏิบัติภารกิจให้เสร็จสิ้น ซึ่งการทดสอบแผนที่หนึ่งๆ จะทำการทดสอบ ความเสถียรภาพคือการทดสอบหลายครั้ง เพื่อดูผลลัพธ์ที่ได้มีความเหมือนเดิม ไม่เปลี่ยนแปลง และการทดสอบการแพร่กระจายแบบ 4 ทิศทาง และแบบ 8 ทิศทาง ว่าผลลัพธ์ที่ได้มีความแตกต่างกันอย่างไร เช่น จำนวนจุด Waypoint ลักษณะเส้นทางการเคลื่อนที่ ความปลอดภัยของเส้นทาง เป็นต้น

ตารางที่ 4.1 ตารางแผนการทดสอบ

การทดสอบ	รายการ
การใช้งานโปรแกรม	ทดสอบการแสดงผลของ Graphic User Interface (GUI)
การทำงานของโปรแกรม	ทดสอบกับตัวอย่างแผนที่ขนาดกว้าง 25 เมตร สูง 20 เมตร (2500*2000พิกเซล)
	ทดสอบกับตัวอย่างแผนที่ขนาดกว้าง 31.45 เมตร สูง 31.45 เมตร (3000*3000พิกเซล)
	ทดสอบกับตัวอย่างแผนที่ขนาดกว้าง 50.30 เมตร สูง 70.30 เมตร (2012*2812พิกเซล)
	ทดสอบกับตัวอย่างแผนที่ขนาดกว้าง 24.60 เมตร สูง 20.30 เมตร (2460*2030พิกเซล)

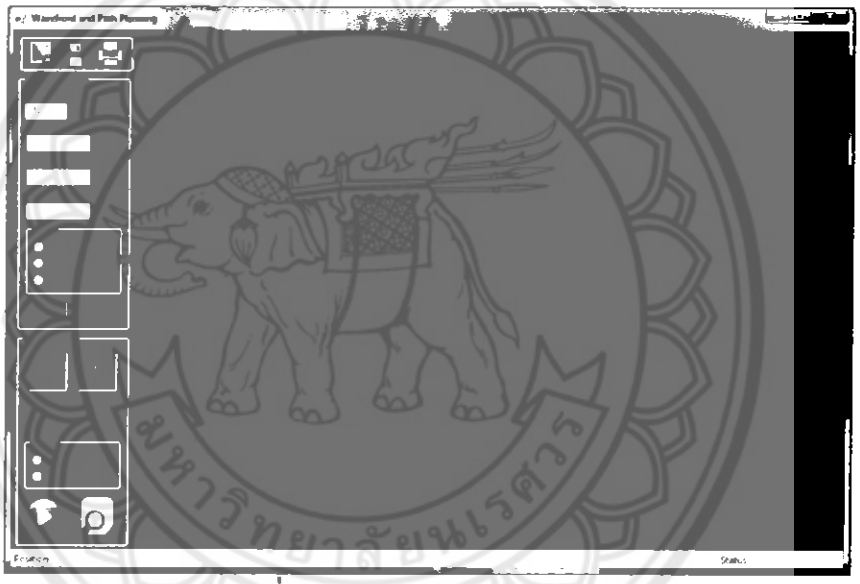
ตารางที่ 4.1 ตารางแผนการทดสอบ(ต่อ)

การทดสอบ	รายการ
การทำงานของโปรแกรม	ทดสอบกับตัวอย่างแผนที่ขนาดกว้าง 25.90 เมตร สูง 13.30 เมตร (2590*1330 พิกเซล)

4.2 การทดสอบโปรแกรม

4.2.1 การใช้งานโปรแกรม

เป็นขั้นตอนการทดสอบการแสดงผลของของโปรแกรม ในหน้าผู้ใช้แบบ Graphic User Interface (GUI) ซึ่งจะมีรูปแบบหน้าตาโปรแกรมเป็นดังที่แสดงในรูปที่ 4.1



รูปที่ 4.1 รูปแบบหน้าตาโปรแกรม

หมายเลข 1 เป็นส่วน Import map, Save map และ Print map

หมายเลข 2 เป็นส่วนกำหนดข้อมูลของแผนที่ ได้แก่

- ค่า Grayscale ค่าที่ใช้ปรับค่าภาพขาวดำ โดยที่ค่ามากขึ้นจะเห็นส่วนที่เป็นสีดำชัดเจนขึ้น ค่าน้อยลงจะเห็นส่วนที่เป็นสีดำน้อยลง
- ขนาดจริงของแผนที่ มีหน่วยเป็นเมตร
- ขนาดรัศมีของหุ่นยนต์ มีหน่วยเป็นเมตร

หมายเลข 3 เป็นส่วนกำหนดระดับความปลอดภัยให้กับหุ่นยนต์ในการเคลื่อนที่โดยสามารถให้ผู้ใช้งานเลือกใช้ได้ดังนี้

- Very Safety มีความปลอดภัยสูงสุด โปรแกรมเลือกเส้นทางที่ปลอดภัยมากที่สุดให้กับหุ่นยนต์ โดยเส้นทางที่ได้อาจไม่ใช่เส้นทางที่สั้นที่สุด เพราะบางเส้นทางหุ่นยนต์อาจเคลื่อนที่ผ่านไปได้ แต่ใกล้ชิดสิ่งกีดขวางมากเกินไป จะทำให้โปรแกรมไม่เลือกเส้นทางดังกล่าว

- Medium Safety มีความปลอดภัยระดับปานกลาง โดยโปรแกรมจะเลือกเส้นทางที่สั้นและปลอดภัยให้กับหุ่นยนต์

- No Safety มีความปลอดภัยน้อยที่สุด โดยโปรแกรมจะเลือกเส้นทางที่สั้นที่สุด แต่เส้นทางที่ได้จะไม่ปลอดภัย ซึ่งอาจทำให้หุ่นยนต์ ชนสิ่งกีดขวาง หรือใกล้ชิดสิ่งกีดขวางในแผนที่มากเกินไป

หมายเลข 4 เป็นส่วนกำหนดจุดเริ่มต้น จุดหมายปลายทาง และรูปแบบการแพร่กระจายของ Wavefront ซึ่งสามารถเลือกได้ทั้งแบบ 4 ทิศทาง และแบบ 8 ทิศทาง โดยที่

- 4 ทิศทาง เป็นการแพร่กระจายแบบ บน ล่าง ซ้าย และขวา

- 8 ทิศทาง เป็นการแพร่กระจายแบบ บน ล่าง ซ้าย ขวา ขวาบน ขวาล่าง ซ้ายบน และซ้ายล่าง

หมายเลข 5 เป็นส่วนแสดงผลของ Map และเส้นทางการเคลื่อนที่ของหุ่นยนต์

4.2.2 การทำงานของโปรแกรม

เป็นขั้นตอนการทดสอบการทำงานของโปรแกรม เมื่อนำตัวอย่างแผนที่เข้าไปในโปรแกรม เพื่อดูผลลัพธ์ที่ได้ โดยจะทำการทดสอบการทำงานของโปรแกรมทั้งแบบการแพร่กระจายแบบ 4 ทิศทาง และการแพร่กระจายแบบ 8 ทิศทาง จากตัวอย่างแผนที่ทั้งหมด 5 แผนที่ ได้แก่

- การทดลองที่ 1 ทดสอบกับตัวอย่างแผนที่ 2500*2000 พิกเซล ซึ่งมีอินพุท คือ ขนาดของความกว้างจริง 25 เมตร สูงจริง 20 เมตร และกำหนดขนาดของหุ่นยนต์เป็นรัศมี 0.2 เมตร ดังนั้นหุ่นยนต์จะมีเส้นผ่านศูนย์กลาง 0.4 เมตรกำหนดจุดเริ่มต้นที่ (-11.23,7.63) และจุดหมายปลายทางที่ (10.23,-2.30) ทำการทดสอบการแพร่กระจายทั้ง 4 ทิศทาง และ 8 ทิศทาง

- การทดลองที่ 2 ทดสอบกับตัวอย่างแผนที่ 3000*3000 พิกเซล ซึ่งมีอินพุท คือ ขนาดของความกว้างจริง 31.45 เมตร สูงจริง 31.45 เมตร และกำหนดขนาดของหุ่นยนต์เป็นรัศมี 0.2 เมตร ดังนั้นหุ่นยนต์จะมีเส้นผ่านศูนย์กลาง 0.4 เมตรกำหนดจุดเริ่มต้นที่ (13.21,11.58) และจุดหมายปลายทางที่ (-12.21,-11.69) ทำการทดสอบการแพร่กระจายทั้ง 4 ทิศทาง และ 8 ทิศทาง

- การทดลองที่ 3 ทดสอบกับตัวอย่างแผนที่ 2012*2812พิกเซล ซึ่งมีอินพุท คือ ขนาดของความกว้างจริง 50.30 เมตร สูงจริง 70.30 เมตร และกำหนดขนาดของหุ่นยนต์เป็นรัศมี 0.2 เมตร ดังนั้นหุ่นยนต์จะมีเส้นผ่านศูนย์กลาง 0.4 เมตรกำหนดจุดเริ่มต้นที่ (-0.35,33.64) และจุดหมายปลายทางที่ (-2.11,-24.85) ทำการทดสอบการแพร่กระจายทั้ง 4 ทิศทาง และ 8 ทิศทาง

- การทดลองที่ 4 ทดสอบกับตัวอย่างแผนที่ 2460*2030 พิกเซล ซึ่งมีอินพุท คือ ขนาดของความกว้างจริง 24.60 เมตร สูงจริง 20.30 เมตร และกำหนดขนาดของหุ่นยนต์เป็นรัศมี 0.2 เมตร ดังนั้นหุ่นยนต์จะมีเส้นผ่านศูนย์กลาง 0.4 เมตรกำหนดจุดเริ่มต้นที่ (-11.37,-5.11) และจุดหมายปลายทางที่ (11.33,9.20) โดยการทดลองนี้จะทดสอบความปลอดภัยของหุ่นยนต์โดยเลือกเส้นทางหุ่นยนต์ไม่สามารถผ่านไปได้

- การทดลองที่ 5 ทดสอบกับตัวอย่างแผนที่ 2590*1330 พิกเซล ซึ่งมีอินพุท คือ ขนาดของความกว้างจริง 25.90 เมตร สูงจริง 13.30 เมตร และกำหนดขนาดของหุ่นยนต์เป็นรัศมี 0.2 เมตร ดังนั้นหุ่นยนต์จะมีเส้นผ่านศูนย์กลาง 0.4 เมตรโดยการทดลองนี้จะทดสอบความปลอดภัยของหุ่นยนต์โดยทดสอบการเลือกระดับความปลอดภัย Very Safety, Medium Safety และ No Safety เพื่อดูการขยายตัวของสิ่งกีดขวางในตัวอย่างแผนที่ดังกล่าว

ตารางที่ 4.2 ตารางการทดสอบโปรแกรม

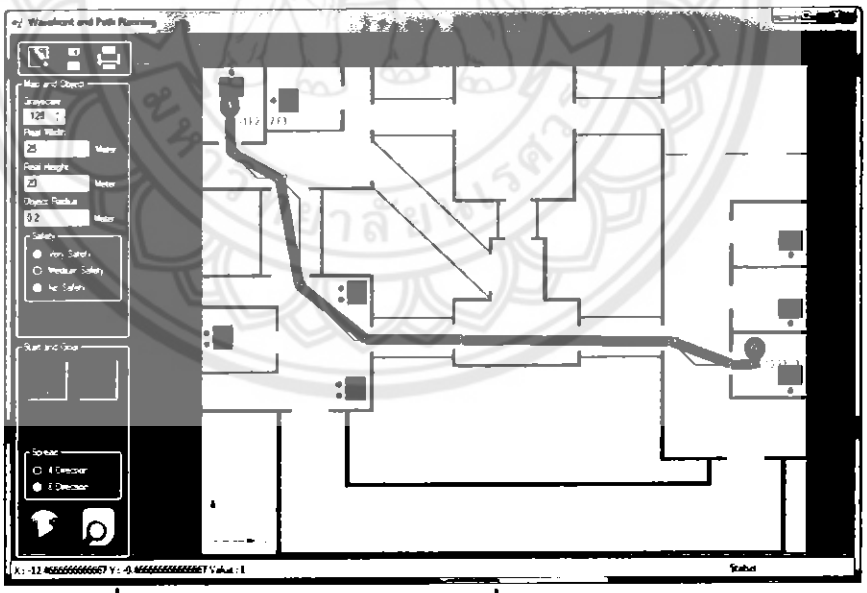
ตัวอย่างแผนที่	Input		
	กว้าง (เมตร)	สูง (เมตร)	ขนาดรัศมีของหุ่นยนต์ (เมตร)
การทดลองที่ 1 ตัวอย่างแผนที่ 2500*2000 พิกเซล	25	20	0.2
การทดลองที่ 2 ตัวอย่างแผนที่ 3000*3000 พิกเซล	31.45	31.45	0.2
การทดลองที่ 3 ตัวอย่างแผนที่ 2012*2812 พิกเซล	50.30	70.30	0.2
การทดลองที่ 4 ตัวอย่างแผนที่ 2460*2030 พิกเซล	24.60	20.30	0.2

ตารางที่ 4.2 ตารางการทดสอบโปรแกรม (ต่อ)

ตัวอย่างแผนที่	Input		
	กว้าง (เมตร)	สูง (เมตร)	ขนาดรัศมีของหุ่นยนต์ (เมตร)
การทดลองที่ 5 ตัวอย่างแผนที่ 2590*1330 พิกเซล	25.9	13.3	0.2

4.2.2.1 ผลการทดสอบกับตัวอย่างแผนที่ กว้าง 25 เมตร สูง 20 เมตร

ทดสอบการแสดงผลการทำงานของโปรแกรมกับตัวอย่างแผนที่ ที่มีขนาดกว้าง 25 เมตร สูง 20 เมตร และมีขนาดรัศมีของหุ่นยนต์ 0.2 เมตร ทำการทดสอบการแพร่กระจายทั้งแบบ 4 ทิศทาง จะปรากฏดังรูปที่ 4.2 และแบบ 8 ทิศทางจะปรากฏดังรูปที่ 4.3 โดยกำหนดจุดเริ่มต้นที่ (-11.23,7.63) และกำหนดจุดหมายปลายทางที่ (10.23,-2.30) โดยเส้นทางสีน้ำเงินคือเส้นทางที่ได้จากการแพร่กระจาย และเส้นทางสีแดงคือเส้นทางที่ได้จาก Waypoint ที่หุ่นยนต์ใช้เคลื่อนที่



รูปที่ 4.2 การทดสอบกับตัวอย่างแผนที่ กว้าง 25 เมตร สูง 20 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

จากรูปที่ 4.2 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 25 เมตร ขนาดสูง 20 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-11.23,7.63) และ

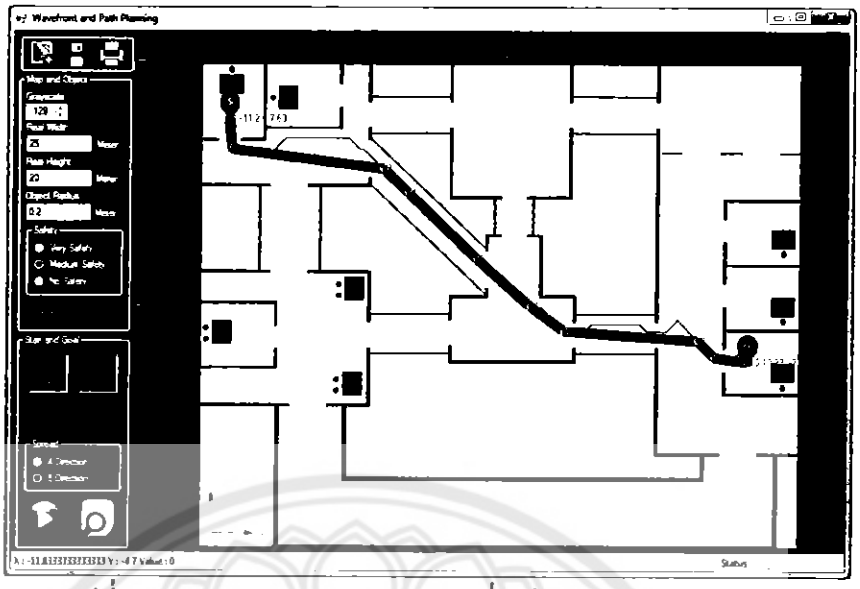
กำหนดจุดหมายปลายทางที่ (10.23,-2.30) และเลือกการแพร่กระจายแบบ 4 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 8 จุด ปรากฏดังตารางที่ 4.3

ตารางที่ 4.3 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 25 เมตร สูง 20 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	-11.23	7.63	Start point
2	-11.13	6.40	
3	-9.17	5.33	
4	-8.37	1.00	
5	-6.33	-1.03	
6	6.87	-1.33	
7	7.40	-1.87	
8	10.23	-2.30	Goal point

จากตารางที่ 4.3 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 8 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

- Waypoint ที่ 1 มีพิกัด (-11.23,7.63)
- Waypoint ที่ 2 มีพิกัด (-11.13,6.40)
- Waypoint ที่ 3 มีพิกัด (-9.17,5.33)
- Waypoint ที่ 4 มีพิกัด (-8.37,1.00)
- Waypoint ที่ 5 มีพิกัด (-6.33,-1.03)
- Waypoint ที่ 6 มีพิกัด (6.87,-1.33)
- Waypoint ที่ 7 มีพิกัด (7.40,-1.87)
- Waypoint ที่ 8 มีพิกัด (10.23,-2.30)



รูปที่ 4.3 การทดสอบกับตัวอย่างแผนที่ กว้าง 25 เมตร สูง 20 เมตร
โดยการแพร่กระจายแบบ 8 ทิศทาง

จากรูปที่ 4.3 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 25 เมตร ขนาดสูง 20 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-11.23,7.63) และกำหนดจุดหมายปลายทางที่ (10.23,-2.30) และเลือกการแพร่กระจายแบบ 8 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 11 จุด ปรากฏดังตารางที่ 4.4

ตารางที่ 4.4 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 25 เมตร สูง20 เมตร
โดยการแพร่กระจายแบบ 4 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	-11.23	7.63	Start point
2	-11.13	6.40	
3	-9.50	6.36	
4	-6.43	6.87	
5	-1.07	2.00	
6	1.23	0.00	
7	2.67	-1.03	
8	3.37	-1.03	

ตารางที่ 4.4 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 25 เมตร สูง20 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง (ต่อ)

Waypoint	X	Y	หมายเหตุ
9	5.23	-0.73	
10	8.87	-2.10	
11	10.23	-2.30	Goal point

จากตารางที่ 4.4 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 11 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

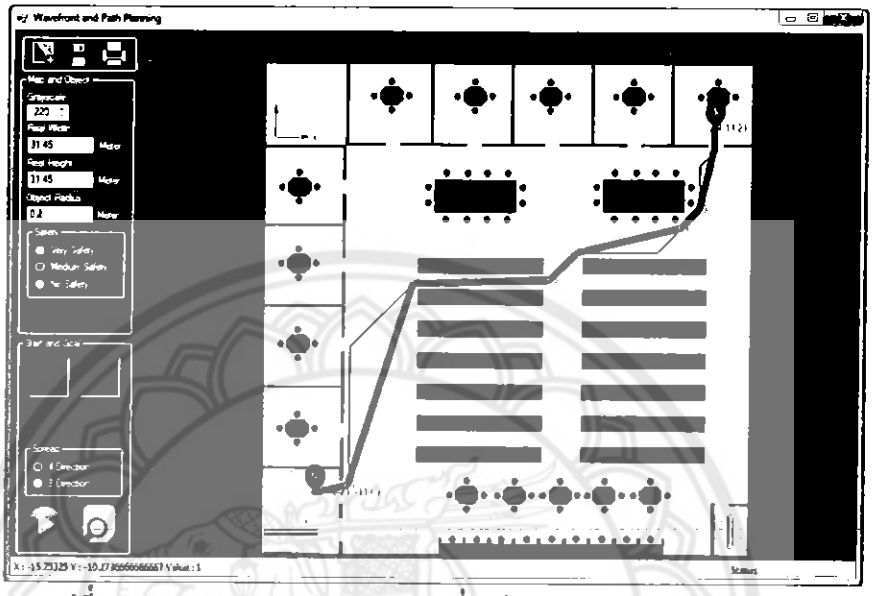
- Waypoint ที่ 1 มีพิกัด (-11.23,7.63)
- Waypoint ที่ 2 มีพิกัด (-11.13,6.40)
- Waypoint ที่ 3 มีพิกัด (-9.5,6.37)
- Waypoint ที่ 4 มีพิกัด (-6.43,6.87)
- Waypoint ที่ 5 มีพิกัด (-1.07,2.00)
- Waypoint ที่ 6 มีพิกัด (1.23,0)
- Waypoint ที่ 7 มีพิกัด (2.67,-1.03)
- Waypoint ที่ 8 มีพิกัด (3.37,-1.03)
- Waypoint ที่ 9 มีพิกัด (5.23,-0.73)
- Waypoint ที่ 10 มีพิกัด (8.87,-2.10)
- Waypoint ที่ 11 มีพิกัด (10.23,-2.30)

จากการทดลองการแพร่กระจายทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทางของตัวอย่างแผนที่กว้าง 25 เมตร สูง 20 เมตร จากรูปที่ 4.2 และรูปที่ 4.3 จะเห็นได้ว่า เส้นทางที่ได้การแพร่กระจายแบบ 8 ทิศทางจะได้เส้นทางที่สั้นกว่าการแพร่กระจายแบบ 4 ทิศทาง แต่จุดเป้าหมายย่อย (Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา การแพร่กระจายแบบ 8 ทิศทางจะมีจำนวนจุดมากกว่าการแพร่กระจายแบบ 4 ทิศทาง ดังนั้นการแพร่กระจายแบบ 8 ทิศทาง จะทำให้หุ่นยนต์มีการเลี้ยวที่บ่อยครั้งกว่าการแพร่กระจายแบบ 4 ทิศทาง

4.2.2.2 ผลการทดสอบกับตัวอย่างแผนที่ กว้าง 31.45 เมตร สูง 31.45 เมตร

ทดสอบการแสดงผลการทำงานของโปรแกรมกับตัวอย่างแผนที่ ที่มีขนาดกว้าง 31.45 เมตร สูง31.45 เมตร และมีขนาดรัศมีของหุ่นยนต์ 0.2 เมตร ทำการทดสอบการแพร่กระจายทั้งแบบ 4 ทิศทาง จะปรากฏดังรูปที่ 4.4 และแบบ 8 ทิศทางจะปรากฏดังรูปที่ 4.5

โดยกำหนดจุดเริ่มต้นที่ (13.21,11.58) และกำหนดจุดหมายปลายทางที่ (-12.21,-11.69) โดยเส้นทางสีน้ำเงินคือเส้นทางที่ได้จากการแพร่กระจาย และเส้นทางสีแดงคือเส้นทางที่ได้จาก Waypoint ที่หุ่นยนต์ใช้เคลื่อนที่



รูปที่ 4.4 การทดสอบกับตัวอย่างแผนที่ กว้าง 31.45 เมตร สูง31.45 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

จากรูปที่ 4.4 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 31.45 เมตร ขนาดสูง 31.45 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (13.21,11.58) และกำหนดจุดหมายปลายทางที่ (-12.21,-11.69) และเลือกการแพร่กระจายแบบ 4 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 9 จุด ปรากฏดังตารางที่ 4.5

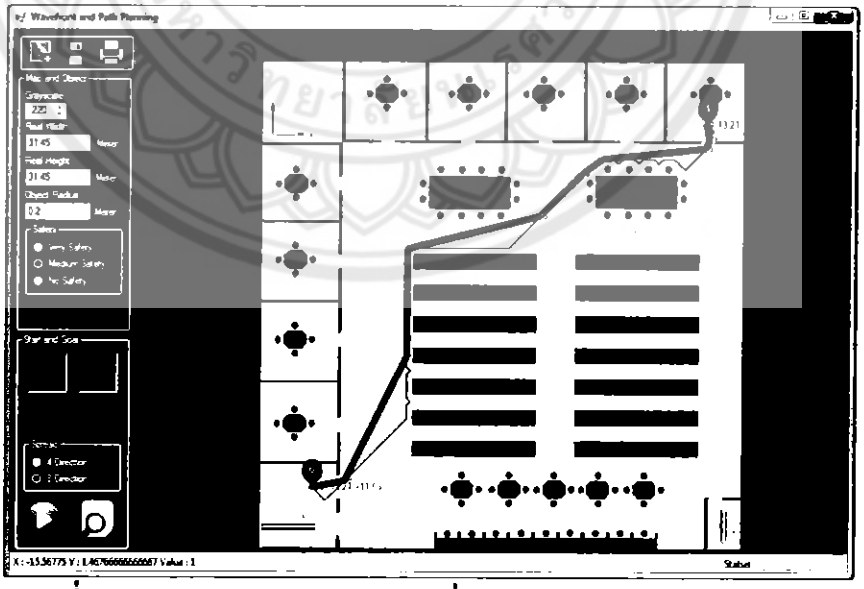
ตารางที่ 4.5 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 31.45 เมตร สูง31.45 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	13.21	11.58	Start point
2	13.46	10.22	
3	12.27	6.45	
4	11.11	5.24	
5	4.46	3.67	

Waypoint	X	Y	หมายเหตุ
6	2.73	1.94	
7	-6.08	1.62	
8	-10.12	-11.22	
9	-12.21	-11.69	Goal point

จากตารางที่ 4.5 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 9 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

- Waypoint ที่ 1 มีพิกัด (13.21,11.58)
- Waypoint ที่ 2 มีพิกัด (13.16,10.22)
- Waypoint ที่ 3 มีพิกัด (12.27,6.45)
- Waypoint ที่ 4 มีพิกัด (11.11,5.24)
- Waypoint ที่ 5 มีพิกัด (4.46,3.67)
- Waypoint ที่ 6 มีพิกัด (2.73,1.94)
- Waypoint ที่ 7 มีพิกัด (-6.08,1.62)
- Waypoint ที่ 8 มีพิกัด (-10.12,-11.22)
- Waypoint ที่ 9 มีพิกัด (-12.21,-11.69)



รูปที่ 4.5 การทดสอบกับตัวอย่างแผนที่ กว้าง 31.45 เมตร สูง 31.45 เมตร โดยการแพร่กระจายแบบ 8 ทิศทาง

จากรูปที่ 4.5 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 31.45 เมตร ขนาดสูง 31.45 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (13.21,11.58) และกำหนดจุดหมายปลายทางที่ (-12.21,-11.69) และเลือกการแพร่กระจายแบบ 8 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 10 จุด ปรากฏดังตารางที่ 4.6

ตารางที่ 4.6 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 31.45 เมตร สูง 31.45 เมตร โดยการแพร่กระจายแบบ 8 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	13.21	11.58	Start point
2	13.37	10.59	
3	13.10	10.06	
4	6.45	9.38	
5	5.19	8.44	
6	2.46	5.71	
7	-6.08	3.30	
8	-6.13	-3.30	
9	-10.12	-11.27	
10	-12.21	-11.69	Goal point

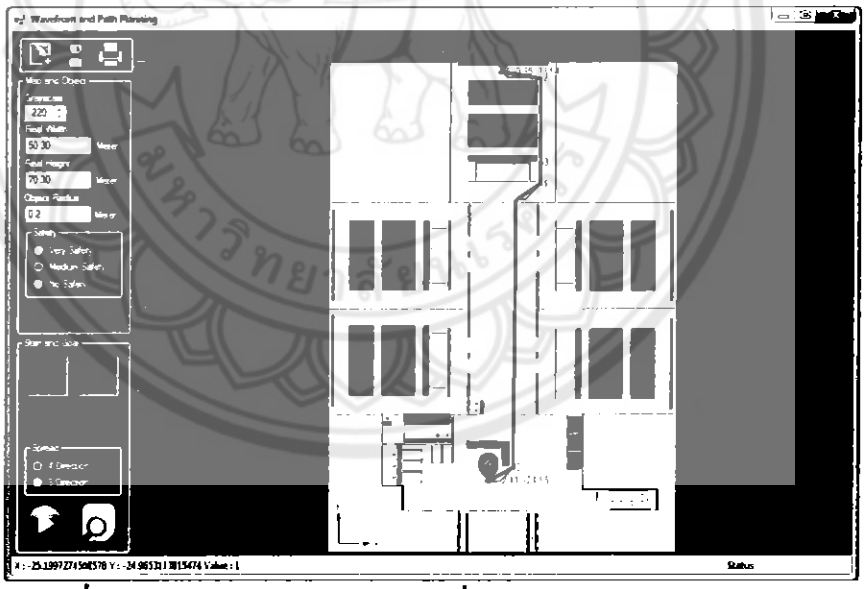
จากตารางที่ 4.6 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 10 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

- Waypoint ที่ 1 มีพิกัด (13.21,11.58)
- Waypoint ที่ 2 มีพิกัด (13.37,10.59)
- Waypoint ที่ 3 มีพิกัด (13.10,10.06)
- Waypoint ที่ 4 มีพิกัด (6.45,9.38)
- Waypoint ที่ 5 มีพิกัด (5.19,8.44)
- Waypoint ที่ 6 มีพิกัด (2.46,5.71)
- Waypoint ที่ 7 มีพิกัด (-6.08,3.62)
- Waypoint ที่ 8 มีพิกัด (-6.13,-3.30)
- Waypoint ที่ 9 มีพิกัด (-10.12,-11.27)
- Waypoint ที่ 10 มีพิกัด (-12.21,-11.69)

จากการทดลองการแพร่กระจายทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทางของตัวอย่างแผนที่กว้าง 31.45 เมตร สูง 31.45 เมตร จากรูปที่ 4.4 และรูปที่ 4.5 จะเห็นได้ว่า เส้นทางที่ได้การแพร่กระจายแบบ 4 ทิศทางจะได้เส้นทางที่สั้นกว่าการแพร่กระจายแบบ 8 ทิศทาง แต่จุดเป้าหมายย่อย (Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา การแพร่กระจายแบบ 8 ทิศทางจะมีจำนวนจุดมากกว่าการแพร่กระจายแบบ 4 ทิศทาง ดังนั้นการแพร่กระจายแบบ 8 ทิศทาง จะทำให้หุ่นยนต์มีการเลี้ยวที่บ่อยครั้งกว่าการแพร่กระจายแบบ 4 ทิศทาง

4.2.2.3 ผลการทดสอบกับตัวอย่างแผนที่ กว้าง 50.30 เมตร สูง 70.30 เมตร

ทดสอบการแสดงผลการทำงานของโปรแกรมกับตัวอย่างแผนที่ ที่มีขนาดกว้าง 50.30เมตร สูง70.30เมตร และมีขนาดรัศมีของหุ่นยนต์ 0.2 เมตร ทำการทดสอบการแพร่กระจายทั้งแบบ 4 ทิศทาง จะปรากฏดังรูปที่ 4.6 และแบบ 8 ทิศทางจะปรากฏดังรูปที่ 4.7 โดยกำหนดจุดเริ่มต้นที่ (-0.35,33.64) และกำหนดจุดหมายปลายทางที่ (-2.11,-24.85) โดยเส้นทางสีน้ำเงินคือเส้นทางที่ได้จากการแพร่กระจาย และเส้นทางสีแดงคือเส้นทางที่ได้จาก Waypoint ที่หุ่นยนต์ใช้เคลื่อนที่



รูปที่ 4.6 การทดสอบกับตัวอย่างแผนที่ กว้าง 50.30 เมตร สูง70.30 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

จากรูปที่ 4.6 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 50.30 เมตร ขนาดสูง 70.30 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-0.35,33.64) และ

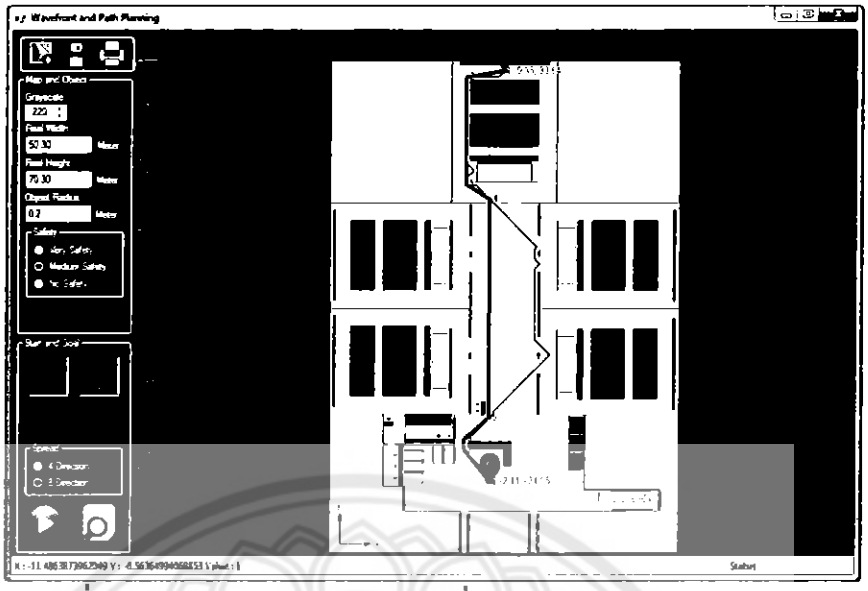
กำหนดจุดหมายปลายทางที่ (-2.11,-24.85) และเลือกการแพร่กระจายแบบ 4 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 7 จุด ปรากฏดังตารางที่ 4.7

ตารางที่ 4.7 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 50.30 เมตร สูง 70.30 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	-0.35	33.64	Start point
2	5.04	32.94	
3	5.27	23.68	
4	5.16	17.58	
5	1.76	15.35	
6	1.41	-22.97	
7	-2.11	-24.85	Goal point

จากตารางที่ 4.7 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 7 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

- Waypoint ที่ 1 มีพิกัด (-0.35,33.64)
- Waypoint ที่ 2 มีพิกัด (5.04,32.94)
- Waypoint ที่ 3 มีพิกัด (5.27,23.68)
- Waypoint ที่ 4 มีพิกัด (5.16,17.58)
- Waypoint ที่ 5 มีพิกัด (1.76,15.35)
- Waypoint ที่ 6 มีพิกัด (1.41,-22.97)
- Waypoint ที่ 7 มีพิกัด (-2.11,-24.85)



รูปที่ 4.7 การทดสอบกับตัวอย่างแผนที่ กว้าง 50.30 เมตร สูง70.30 เมตร โดยการแพร่กระจายแบบ 8 ทิศทาง

จากรูปที่ 4.7 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 50.30 เมตร ขนาดสูง 70.30 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-0.35,33.64) และกำหนดจุดหมายปลายทางที่ (-2.11,-24.85) และเลือกการแพร่กระจายแบบ 8 ทิศทาง ผลลัพธ์ที่ได้จะได้พิกัด (x,y) ของจุดเป้าหมายย่อย (Waypoint) ทั้งหมด 8 จุด ปรากฏดังตารางที่ 4.8

ตารางที่ 4.8 ตารางแสดง Waypoint ตัวอย่างแผนที่ กว้าง 50.30 เมตร ยาว 70.30 เมตร โดยการแพร่กระจายแบบ 8 ทิศทาง

Waypoint	X	Y	หมายเหตุ
1	-0.35	33.64	Start point
2	-5.63	32.82	
3	-5.74	17.58	
4	-2.23	15.35	
5	-1.88	-15.35	
6	-5.74	-18.75	
7	-5.74	-20.39	
8	-2.11	-24.85	Goal point

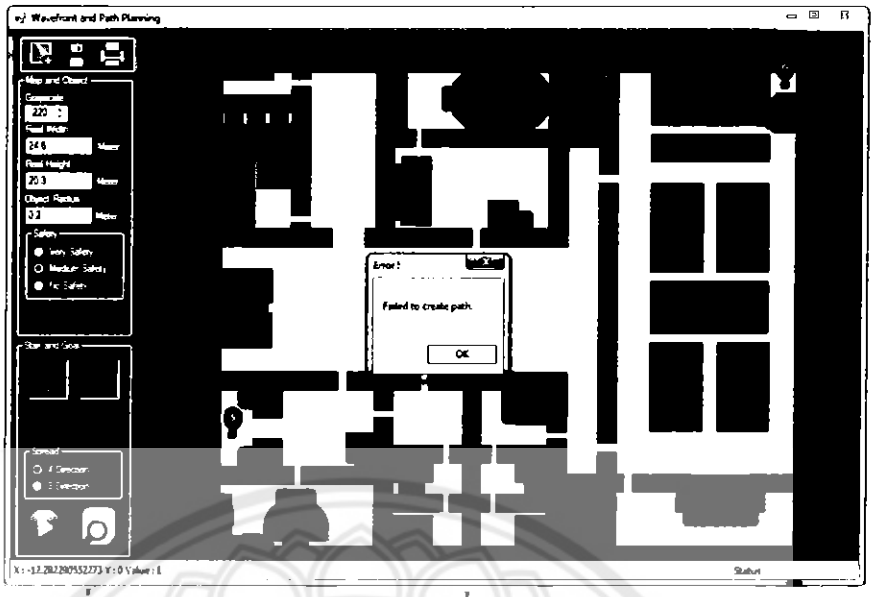
จากตารางที่ 4.8 จะแสดงถึงพิกัด (x,y) ของจุด Waypoint ทั้งหมด 8 จุด ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา โดยเริ่มตั้งแต่จุด Start ไปจนกระทั่งถึงจุด Goal ได้แก่

- Waypoint ที่ 1 มีพิกัด (-0.35,33.64)
- Waypoint ที่ 2 มีพิกัด (-5.63,32.82)
- Waypoint ที่ 3 มีพิกัด (-5.74,17.58)
- Waypoint ที่ 4 มีพิกัด (-2.23,15.35)
- Waypoint ที่ 5 มีพิกัด (-1.88,-15.35)
- Waypoint ที่ 6 มีพิกัด (-5.74,-18.75)
- Waypoint ที่ 7 มีพิกัด (-5.74,-20.39)
- Waypoint ที่ 8 มีพิกัด (-2.11,-24.85)

จากการทดลองการแพร่กระจายทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทางของตัวอย่างแผนที่กว้าง 50.30 เมตร สูง 70.30 เมตร จากรูปที่ 4.6 และรูปที่ 4.7 จะเห็นได้ว่า เส้นทางที่ได้การแพร่กระจายแบบ 8 ทิศทางจะได้เส้นทางที่สั้นกว่าการแพร่กระจายแบบ 4 ทิศทาง แต่จุดเป้าหมายย่อย (Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา การแพร่กระจายแบบ 8 ทิศทางจะมีจำนวนจุดมากกว่าการแพร่กระจายแบบ 4 ทิศทาง ดังนั้นการแพร่กระจายแบบ 8 ทิศทาง จะทำให้หุ่นยนต์มีการเลี้ยวที่บ่อยครั้งกว่าการแพร่กระจายแบบ 4 ทิศทาง

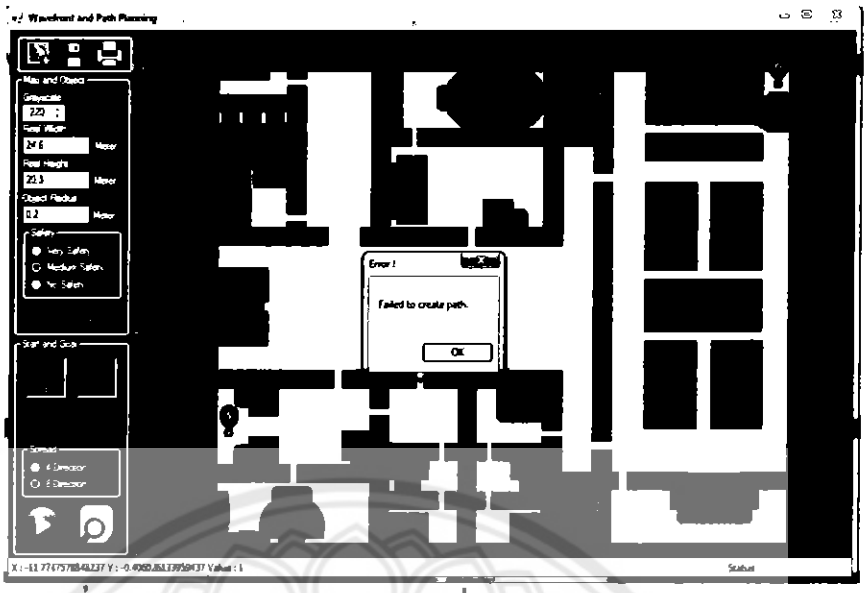
4.2.2.4 ผลการทดสอบกับตัวอย่างแผนที่ กว้าง 24.60 เมตร สูง 20.30 เมตร

ทดสอบการแสดงผลการทำงานของโปรแกรมกับตัวอย่างแผนที่ ที่มีขนาดกว้าง 24.60 เมตร สูง 24.30เมตร และมีขนาดรัศมีของหุ่นยนต์ 0.2 เมตร ทำการทดสอบความปลอดภัยของหุ่นยนต์ โดยเลือกเส้นทางที่หุ่นยนต์ไม่สามารถผ่านไปได้ จะปรากฏดังรูปที่ 4.8 และแบบ 8 ทิศทางจะปรากฏดังรูปที่ 4.9 โดยกำหนดจุดเริ่มต้นที่ (-11.37,-5.11) และกำหนดจุดหมายปลายทางที่ (11.33,9.20)



รูปที่ 4.8 การทดสอบกับตัวอย่างแผนที่ กว้าง 24.60 เมตร สูง20.30 เมตร โดยการแพร่กระจายแบบ 4 ทิศทาง

จากรูปที่ 4.8 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 50.30 เมตร ขนาดสูง 70.30 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-11.37,-5.11) และกำหนดจุดหมายปลายทางที่ (11.33,9.20)ซึ่งเป็นเส้นทางที่หุ่นยนต์ไม่สามารถผ่านไปได้ เพราะระยะห่างระหว่างวัตถุไม่เพียงพอต่อขนาดตัวของหุ่นยนต์และเลือกการแพร่กระจายแบบ 4 ทิศทาง ผลลัพธ์ที่ได้โปรแกรมจะแสดง Message Box เพื่อบอกให้ผู้ใช้งานทราบว่า ไม่สามารถสร้างเส้นทางการเคลื่อนที่ให้กับตัวหุ่นยนต์ได้



รูปที่ 4.9 การทดสอบกับตัวอย่างแผนที่ กว้าง 24.60 เมตร สูง20.30 เมตร โดยการแพร่กระจายแบบ 8 ทิศทาง

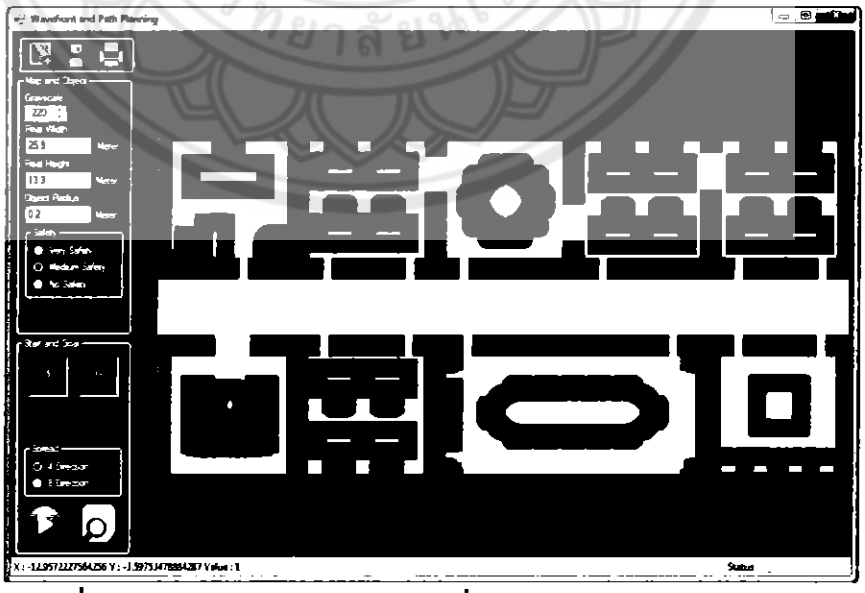
จากรูปที่ 4.9 เมื่อกำหนดอินพุทของโปรแกรม ได้แก่ ขนาดกว้าง 50.30 เมตร ขนาดสูง 70.30 เมตร ขนาดรัศมีของหุ่นยนต์ 0.2 เมตร กำหนดจุดเริ่มต้นที่ (-11.37,-5.11) และกำหนดจุดหมายปลายทางที่ (11.33,9.20) ซึ่งเป็นเส้นทางที่หุ่นยนต์ไม่สามารถผ่านไปได้ เพราะระยะห่างระหว่างวัตถุไม่เพียงพอต่อขนาดตัวของหุ่นยนต์ และเลือกการแพร่กระจายแบบ 8 ทิศทาง ผลลัพธ์ที่ได้โปรแกรมจะแสดง Message Box เพื่อบอกให้ผู้ใช้งานทราบว่า ไม่สามารถสร้างเส้นทางการเคลื่อนที่ให้กับตัวหุ่นยนต์ได้

4.2.2.5 ผลการทดสอบกับตัวอย่างแผนที่ กว้าง 25.90 เมตร สูง 13.30 เมตร

ทดสอบการแสดงผลการทำงานของโปรแกรมกับตัวอย่างแผนที่ 2590*1330 ที่มีขนาดกว้าง 25.90 เมตร สูง 13.30เมตร และมีขนาดรัศมีของหุ่นยนต์ 0.2 เมตร ทำการทดสอบความปลอดภัยของหุ่นยนต์ โดยเลือกทดสอบระดับความปลอดภัยตั้งแต่ No Safety, Medium Safety และ Very Safety จะปรากฏดังรูปที่ 4.10 รูปที่ 4.11 และรูปที่ 4.12ตามลำดับ

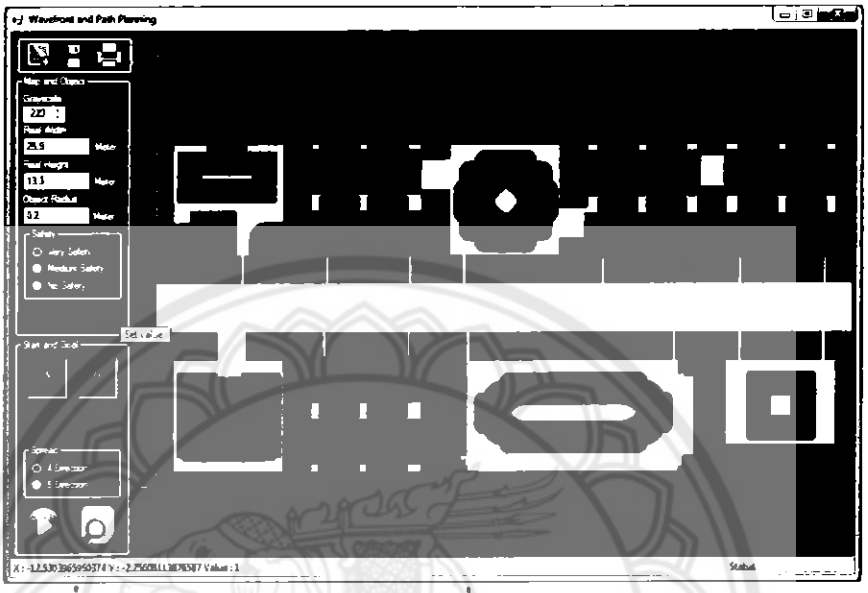
รูปที่ 4.10 การทดสอบกับตัวอย่างแผนที่ กว้าง 25.90 เมตร สูง 13.30 เมตร โดยเลือกระดับความปลอดภัยเป็น No Safety

จากรูปที่ 4.10 การเลือกระดับความปลอดภัยแบบ No Safety เป็นการขยายตัวของสิ่งกีดขวางเป็นครึ่งหนึ่งของขนาดของรัศมีหุ่นยนต์ ซึ่งก็คือพื้นที่สีดำที่ขยายตัวออกมาจากสิ่งกีดขวาง เป็นพื้นที่ที่หุ่นยนต์ไม่สามารถผ่านไปได้หากมีการขยายตัวเป็นครึ่งหนึ่งของรัศมีหุ่นยนต์ จะทำให้เกิดเส้นทางการเคลื่อนที่ได้มาก แต่จะเกิดความเสี่ยงต่อตัวหุ่นยนต์ เพราะจะทำให้หุ่นยนต์เคลื่อนที่ชนสิ่งกีดขวาง หรือเข้าใกล้สิ่งกีดขวางมากเกินไปจนเกิดอันตรายต่อตัวหุ่นยนต์



รูปที่ 4.11 การทดสอบกับตัวอย่างแผนที่ กว้าง 25.90 เมตร สูง 13.30 เมตร โดยเลือกระดับความปลอดภัยเป็น Medium Safety

จากรูปที่ 4.11 การเลือกระดับความปลอดภัยแบบ Medium Safety เป็นการขยายตัวของสิ่งกีดขวางเท่ากับขนาดของรัศมีหุ่นยนต์ พื้นที่สีดำเพิ่มมากขึ้น จะทำให้เกิดเส้นทางการเคลื่อนที่ที่มีความปลอดภัยและมีระยะทางที่สั้นเหมาะสมกับตัวหุ่นยนต์



รูปที่ 4.12 การทดสอบกับตัวอย่างแผนที่ กว้าง 25.90 เมตร สูง 13.30 เมตร โดยเลือกระดับความปลอดภัยเป็น Very Safety

จากรูปที่ 4.12 การเลือกระดับความปลอดภัยแบบ Very Safety เป็นการขยายตัวของสิ่งกีดขวางเป็นสองเท่าของขนาดของรัศมีหุ่นยนต์ พื้นที่สีดำมากจนเกินไปทำให้หุ่นยนต์มีพื้นที่ในการเคลื่อนที่น้อยลง แต่จะทำให้เกิดเส้นทางการเคลื่อนที่ที่มีความปลอดภัยมากที่สุดกับตัวหุ่นยนต์ เพราะบางเส้นทางจะผ่านไปไม่ได้ เส้นทางการเคลื่อนที่ที่ได้ก็จะมีระยะที่ไกลเกินความจำเป็นหรือโปรแกรมอาจสร้างเส้นทางไม่ได้เลย เนื่องจากโปรแกรมจะเลือกเส้นทางที่ปลอดภัยมากที่สุด

4.3 สรุปการทดสอบโปรแกรม

จากการทดสอบโปรแกรม โปรแกรมสามารถเลือกเส้นทางการเคลื่อนที่ที่สั้นและมีความปลอดภัยกับตัวหุ่นยนต์ ซึ่งการเลือกการแพร่กระจายทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทาง ผลลัพธ์ที่ได้ระยะทางของเส้นทางการเคลื่อนที่จะขึ้นอยู่กับรูปแบบของแผนที่ซึ่งจะให้ผลลัพธ์ออกมาแตกต่างกัน ในการทดสอบบางแผนที่การแพร่กระจายแบบ 4 ทิศทาง จะมีระยะทางที่สั้นกว่าการแพร่กระจายแบบ 8 ทิศทาง บางแผนที่การแพร่กระจายแบบ 8 ทิศทาง จะมีระยะทางที่สั้นกว่าการแพร่กระจายแบบ 4 ทิศทาง ซึ่งจากการทดสอบส่วนใหญ่การแพร่กระจายแบบ 8 ทิศทางจะ

ให้ระยะทางที่สั้นกว่า แต่จะเกิดจุดเป้าหมายย่อย (Waypoint)ที่มากกว่า ซึ่งจะทำให้หุ่นยนต์มีการเลี้ยวที่บ่อยครั้งกว่าการแพร่กระจายแบบ 4 ทิศทาง แต่เส้นทางการเคลื่อนที่ของหุ่นยนต์ที่เกิดขึ้นมีความปลอดภัยต่อตัวหุ่นยนต์ หากผู้ใช้งานเลือกระดับความปลอดภัยเป็น Medium Safety และ Very Safety

บทที่ 5
สรุปผลการดำเนินงานและแนวทางการพัฒนา

โครงงานนี้เป็นการพัฒนาระบบสำหรับการออกแบบเส้นทางการเคลื่อนที่ของหุ่นยนต์ โดยนำหลักการแพร่ของคลื่นมาประยุกต์ใช้ เพื่อเลือกหาเส้นทางที่สั้นและปลอดภัยสำหรับหุ่นยนต์ในการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายปลายทางได้อย่างมีประสิทธิภาพ ซึ่งโปรแกรมจะสามารถนำเอาแผนที่ที่สร้างขึ้น มาใช้ในการค้นหาเส้นทางการเคลื่อนที่ที่เหมาะสม โดยอาศัยหลักการแพร่ของคลื่น โปรแกรมมีการแสดงผลแบบ Graphic User Interface (GUI) ทำให้สามารถมองเห็นแผนที่และสิ่งกีดขวางในแผนที่ เส้นทางการเคลื่อนที่จากจุดเริ่มต้นไปยังจุดหมายปลายทาง และให้ผลลัพธ์ออกมาเป็นเป้าหมายย่อย (A Series of Waypoint) สำหรับให้หุ่นยนต์เคลื่อนที่เข้าหา นอกจากนี้โปรแกรมยังสามารถบอกถึงระยะทางที่หุ่นยนต์ทำการเคลื่อนที่เข้าหาเป้าหมายย่อยต่างๆ เพื่อจัดการภารกิจให้เสร็จสิ้นต่อไป

โครงงานนี้ใช้ภาษา Visual C# ในการพัฒนาโปรแกรมเนื่องจากเป็นภาษาที่กระชับ ง่าย และกำลังเป็นที่นิยมต่อการพัฒนาโปรแกรมและการออกแบบหน้าตาของโปรแกรมแบบ Graphic User Interface (GUI) เพื่อให้ง่ายต่อการใช้งาน

5.1 ผลการทดลอง

จากการทดลองโปรแกรมด้วยตัวอย่างแผนที่ขนาดต่างๆ โปรแกรมสามารถเลือกเส้นทางการเคลื่อนที่มีระยะทางสั้นและปลอดภัยให้กับหุ่นยนต์ตามลักษณะการแพร่กระจาย ซึ่งการแพร่กระจายทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทาง ไม่สามารถทราบได้ว่าการแพร่กระจายแบบใดมีประสิทธิภาพมากกว่ากัน ทั้งนี้ขึ้นอยู่กับรูปแบบของแผนที่ การกำหนดจุดเริ่มต้นและจุดหมายปลายทาง บางกรณีอาจได้เส้นทางที่มีระยะทางสั้นกว่าแต่จำนวนเป้าหมายย่อย (Waypoint) ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา มีจำนวนมากกว่า ก็จะทำให้หุ่นยนต์มีการเลี้ยวบ่อยครั้งขึ้น จึงจะทำให้ใช้เวลาในการเคลื่อนที่มากกว่า ดังนั้นการเลือกการแพร่กระจายจึงขึ้นอยู่กับความต้องการของผู้ใช้งานโปรแกรม

การทำงานของระบบที่พัฒนาขึ้นมีประสิทธิภาพในการทำงานที่รวดเร็วหรือล่าช้าขึ้นอยู่กับขั้นตอนการแพร่กระจายของ Wavefront algorithm หากมีการแพร่กระจายที่มาก หรือสิ่งกีดขวางในแผนที่มาก การแพร่กระจายก็จะซับซ้อนยิ่งขึ้น ก็จะทำให้ระบบที่พัฒนาขึ้นช้าลง และระบบมีความเสถียรภาพ เนื่องจากการทดลองโดยกำหนดจุดเริ่มต้นและจุดหมายปลายทางเดียวกัน ระบบจะให้ผลลัพธ์เป็นจุดเป้าหมายย่อย (Waypoint) ออกมาเป็นจำนวนเท่าเดิมและจุดเดิมทุกครั้ง

5.2 ปัญหาและอุปสรรค

จากขั้นตอนการทำงาน ผู้พัฒนาได้เห็นถึงปัญหาที่เกิดขึ้น ได้แก่

- สิ่งกีดขวางในแผนที่ ที่เป็นวงกลมจะทำให้เกิดจุดเป้าหมายย่อย (Waypoint) หลายจุดใกล้เคียงกัน ซึ่งผู้พัฒนาได้ปรับปรุงระบบโดยการทำการเฉลี่ยจุดเป้าหมายย่อย (Waypoint) ที่ใกล้เคียงกัน ที่มีระยะห่างกันน้อยกว่า 10 พิกเซล มารวมกันเป็นจุดๆ เดียว
- มุมของสิ่งกีดขวางเพียงหนึ่งพิกเซล จะทำให้เกิดจุดเป้าหมายย่อย (Waypoint) ที่ไม่จำเป็นขึ้นมาได้ ซึ่งผู้พัฒนาได้ปรับปรุงระบบโดยการเลือกระยะทางที่สั้นจากการเช็คระยะทางการหาจุดเป้าหมายย่อย (Waypoint) ระหว่างเริ่มเช็คจากจุด Start ไปยังจุด Goal กับเริ่มเช็คจากจุด Goal มายังจุด Start
- เส้นทางการเคลื่อนที่ของหุ่นยนต์ที่โปรแกรมให้ผลลัพธ์ออกมาเป็นเส้นทางที่มีระยะทางที่สั้นและปลอดภัย ซึ่งระยะทางที่สั้นกว่า แต่จุดเป้าหมายย่อย (Waypoint) มาก จะทำให้หุ่นยนต์เสียเวลาในการหยุดเลี้ยวมากกว่า ดังนั้นเส้นทางที่ได้ยังไม่คำนึงถึงเวลาที่ใช้ในการเคลื่อนที่

5.3 ข้อเสนอแนะ

จากปัญหาและอุปสรรคที่พบจากหัวข้อ 5.2 ผู้ดำเนินโครงงานจึงมีข้อเสนอแนะสำหรับผู้ที่ต้องการนำโครงงานไปพัฒนาต่อ ควรจะต้องมีความรู้ในสิ่งต่อไปนี้เพื่อที่จะสามารถนำไปพัฒนาเพิ่มเติมได้ คือ

1. ศึกษา Wavefront algorithm ซึ่งเป็นอัลกอริทึมเลียนแบบการแพร่ของคลื่น
2. ศึกษาหลักการของ Waypoint เพื่อการควบคุมการเคลื่อนที่ของหุ่นยนต์ ได้อย่างมีประสิทธิภาพยิ่งขึ้น
3. ศึกษาการสร้าง Graphic User Interface ด้วยภาษา Visual C# เพื่อนำไปใช้ในการแสดงผลที่สมบูรณ์ยิ่งขึ้น

5.4 การพัฒนาโครงงานต่อไปในอนาคต

ระบบที่พัฒนาขึ้นสามารถนำไปใช้ร่วมกับการพัฒนาหุ่นยนต์เคลื่อนที่ประเภทต่างๆ ได้ เพื่อให้หุ่นยนต์เคลื่อนที่ที่พัฒนาร่วมกับระบบ สามารถเลือกเส้นทางที่สั้นและปลอดภัยให้กับตัวหุ่นยนต์เองได้ หรือสามารถพัฒนาระบบให้มีประสิทธิภาพมากขึ้นโดยการใช้ Wavefront แบบ Dualซึ่งระบบที่พัฒนาขึ้นนี้เป็นการแพร่กระจาย เพื่อตรวจสอบสิ่งกีดขวาง จากจุดหมายปลายทางมายังจุดเริ่มต้น แล้วจึงทำการแสดงจุดเป้าหมายย่อย (Waypoint) เพื่อสร้างเส้นทางการเคลื่อนที่

ให้กับหุ่นยนต์ ซึ่งผลลัพธ์ที่ได้จากการแพร่กระจายในทิศทางเดียว ยังทำให้เกิดจุดเป้าหมายย่อยที่ไม่จำเป็นขึ้น ทำให้หุ่นยนต์มีการเลี้ยวที่บ่อยครั้งขึ้น ซึ่งจะทำให้เสียเวลา หากนำระบบไปพัฒนาร่วมกับหุ่นยนต์เคลื่อนที่จริง ดังนั้นการแพร่กระจายแบบ Dual-wavefront จะทำให้ระบบมีประสิทธิภาพมากขึ้น

Dual-wavefront เป็นการแพร่กระจายแบบสองทิศทาง โดยการแพร่กระจายจะเริ่มทั้งจากจุดเริ่มต้นและจุดหมายปลายทางจนกระทั่งการแพร่กระจายจากทั้งสองทิศทางนั้นมาพบกัน ซึ่งเส้นทางที่ได้หลังจากการแพร่กระจายจะช่วยให้ลดระยะทางให้สั้นลงกว่าการแพร่กระจายจากทิศทางเดียว และการแพร่กระจายแบบ Dual-wavefront จะช่วยกำจัดจุดเป้าหมายย่อย (a series of waypoint) ที่ไม่จำเป็นออก และให้ผลลัพธ์เป็นจุดเป้าหมายย่อยที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา ตั้งแต่จุดเริ่มต้นไปยังจุดหมายปลายทาง ซึ่งจะทำให้ระบบมีประสิทธิภาพมากขึ้น [10]

5.5 สรุปผลการดำเนินงาน

จากการทดลองโปรแกรมสามารถสร้างเส้นทางการเคลื่อนที่ของหุ่นยนต์ ซึ่งสามารถกำหนดจุดเริ่มต้นจุดหมายปลายทาง และขนาดของหุ่นยนต์ได้ โปรแกรมยังสามารถรับแผนที่มาตราส่วนต่างๆ ได้และสามารถเลือกใช้เทคนิคการแพร่กระจายด้วย Wavefront algorithm ทั้งแบบ 4 ทิศทางและแบบ 8 ทิศทาง โดยให้ผลลัพธ์จุดเป้าหมายย่อยที่หุ่นยนต์ต้องเคลื่อนที่เข้าหาเพื่อปฏิบัติภารกิจให้เสร็จสิ้น ออกมาเป็นเส้นทางการเคลื่อนที่ที่มีความสั้นและปลอดภัยสำหรับตัวหุ่นยนต์ โดยเส้นทางที่ได้จะไม่ชนสิ่งกีดขวางซึ่งผู้ใช้งานสามารถนำโปรแกรมที่พัฒนาขึ้นไปใช้ปรับปรุงกับหุ่นยนต์เคลื่อนที่ประเภทต่างๆได้

เอกสารอ้างอิง

[1] Wikipedia. (March 10, 2012). **Motion Planning.**Retrieved July 22, 2012, from http://en.wikipedia.org/wiki/Motion_planning

[2] อรรถวิทย์ สุดแสง. (มิถุนายน 47).**การวางแผนการเคลื่อนที่.**สืบค้นเมื่อ 23กรกฎาคม 2555, จาก http://www.cp.eng.chula.ac.th/~attawith/class/motion.pdf

[3] P. Svestka and M. H. Overmars. **Probabilistic Path Planning: Robot Motion Planning and Control.**Retrieved July 23, 2012, from http://ftp.laas.fr/pub/ria/promotion/chap5.pdf

[4] ภัคพงศ์จันทเปรมจิตต์ และ มีเดช เกตุแก้ว. (19-21 ตุลาคม 2554). **การประชุมวิชาการเครือข่ายวิศวกรรมเครื่องกลแห่งประเทศไทย ครั้งที่ 25: การศึกษาการระบุตำแหน่งและสร้างแผนที่ในขณะเดียวกัน.**สืบค้นเมื่อ 24 กรกฎาคม 2555, จาก http://mechatronics.ptwit.ac.th/seksan/menett25/DRC/DRC03.pdf

[5] Wikipedia. (July 7,2012). **Simultaneous localization and mapping (SLAM).**Retrieved July 24, 2012, from http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[6] อัครเดช ม่วงสนิท. หุ่นยนต์ (Robotics).สืบค้นเมื่อ 24 กรกฎาคม 2555, จาก rambutan.net63.net/docs/robotics%202008.doc

[7] ValdirGrassi Junior, Sarangi P. Parikh and Jun Okamoto Junior. (2006). **HYBRID DELIBERATIVE/REACTIVE ARCHITECTURE FOR HUMAN-ROBOT INTERACTION.** Retrieved July 25, 2012, fromhttp://www.abcm.org.br/symposiumseries/SSM_Vol2/Section_VIII_Inteligence_and_Cooperation_in_Robotics/SSM2_VIII_03.pdf

[8] นายสนิท แก้วหนองแสง. **บทเรียนคอมพิวเตอร์ช่วยสอนผ่านเครือข่ายวิชาฟิสิกส์ : คลื่นกล.**สืบค้นเมื่อ 25 กรกฎาคม 2555, จาก http://www.atom.rmutphysics.com/charud/oldnews/0/284/6/wave/wave.html

[9] TomOotjers. **Line Drawing Algorithm Explained.**Retrieved December 19, 2012, from http://www.gamedev.net/page/resources/_/technical/game-programming/line-drawing-algorithm-explained-r1275

เอกสารอ้างอิง (ต่อ)

[10] Lynne E. Parker, Ben Birch, and Chris Reardon. **Indoor Target Intercept Using an Acoustic Sensor Network and Dual Wavefront Path Planning.**Retrieved Febuary 28, 2013, from http://www8.cs.umu.se/kurser/TDBD17/VT04/dl/Assignment%20Papers/Indoor%20Target%20Intercept%20Using%20an%20Acoustic%20Sensor.pdf

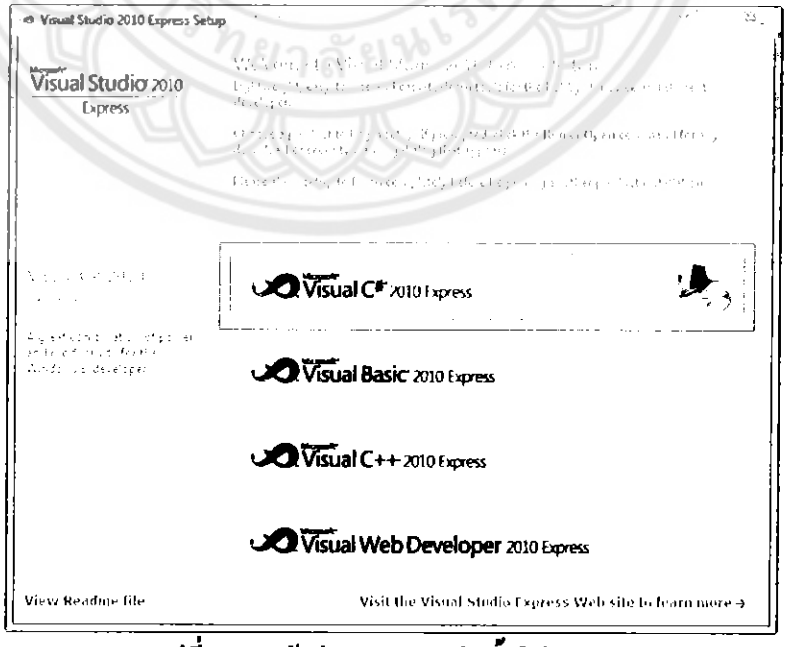
ภาคผนวก
คู่มือการใช้งานโปรแกรม

ก. การติดตั้งโปรแกรม *Microsoft Visual Studio 2010 Express*

การติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express สามารถดาวน์โหลดโปรแกรมได้ที่ http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express

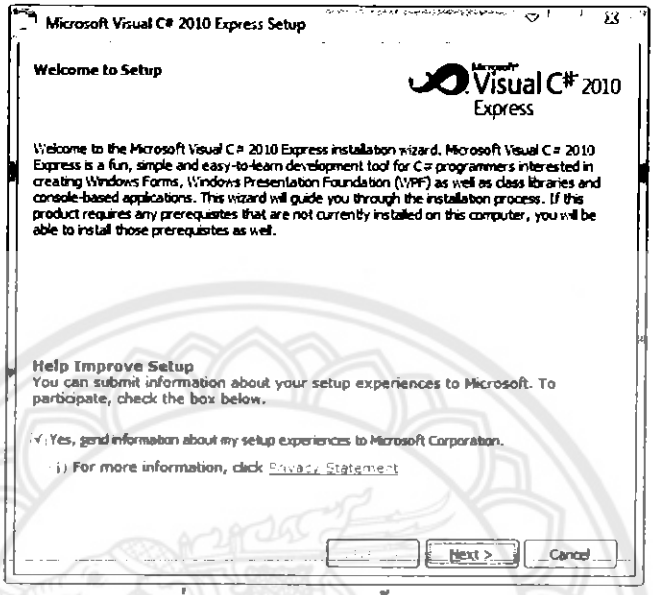
เมื่อผู้ใช้งานทำการดาวน์โหลดโปรแกรม Microsoft Visual Studio 2010 Express ให้ผู้ใช้งานทำการติดตั้งโปรแกรม โดยการดับเบิ้ลคลิกที่ไฟล์ Setup ดังรูปที่ ก.1 จากนั้นจะเข้าสู่การติดตั้งโปรแกรม ดังรูปที่ ก.2

รูปที่ ก.1 การติดตั้งโปรแกรม Microsoft Visual Studio



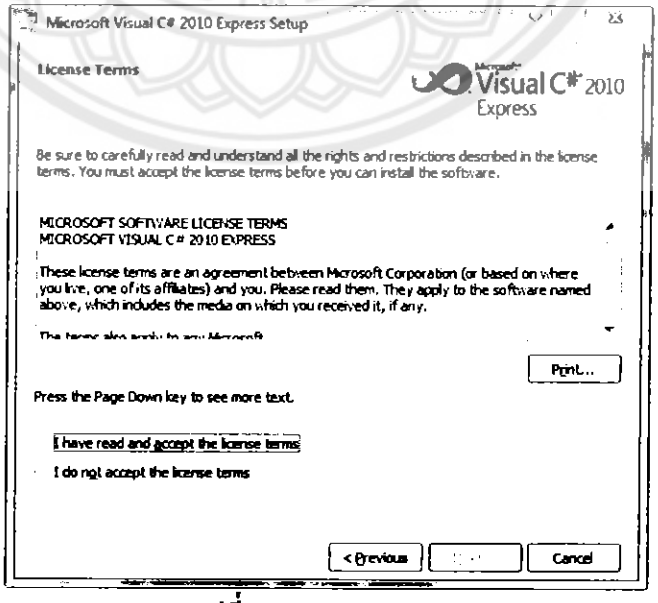
รูปที่ ก.2 หน้าต่างแสดงการติดตั้งโปรแกรม

เมื่อเข้าสู่หน้าต่างการติดตั้งโปรแกรม ดังรูปที่ ก.2 แล้ว ให้ผู้ใช้งานเลือก Microsoft Visual C# 2010 Express จากนั้นรอสักครู่ โปรแกรมจะดำเนินการเข้าสู่ขั้นตอนการติดตั้ง ดังรูปที่ ก.3 ให้ผู้ใช้งานคลิก Next > เพื่อไปยังขั้นตอนถัดไป



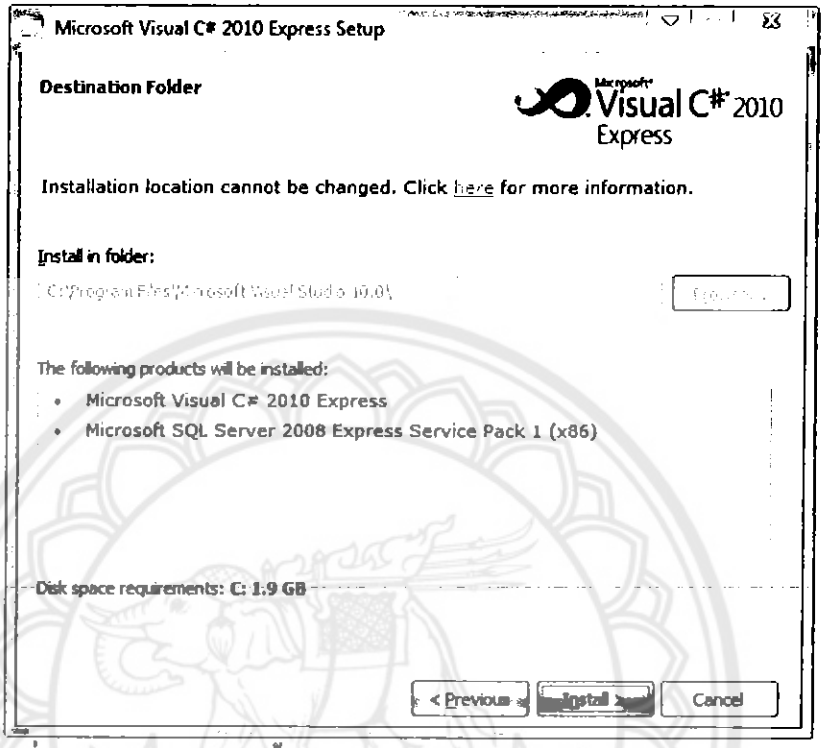
รูปที่ ก.3 เข้าสู่การติดตั้งโปรแกรม

จากนั้นให้ผู้ใช้งานอ่านรายละเอียดของ License Terms เมื่อผู้ใช้งานอ่านเรียบร้อย ให้ผู้งานทำการคลิกที่ I have read and accept the license terms และคลิก Next > ต่อไปเรื่อยๆ เพื่อเข้าสู่ขั้นตอนตำแหน่งการติดตั้งโปรแกรม ดังรูปที่ ก.4



รูปที่ ก.4 License Terms

เมื่อเข้าสู่ขั้นตอนการติดตั้งตำแหน่งของโปรแกรม ให้ผู้ใช้งานคลิก Install เพื่อติดตั้งโปรแกรม ให้ผู้ใช้งานรอสักครู่ จนกระทั่งโปรแกรมทำการติดตั้งจนเสร็จสมบูรณ์



รูปที่ ก.5 ตำแหน่งการติดตั้งโปรแกรม Microsoft Visual Studio 2010 Express

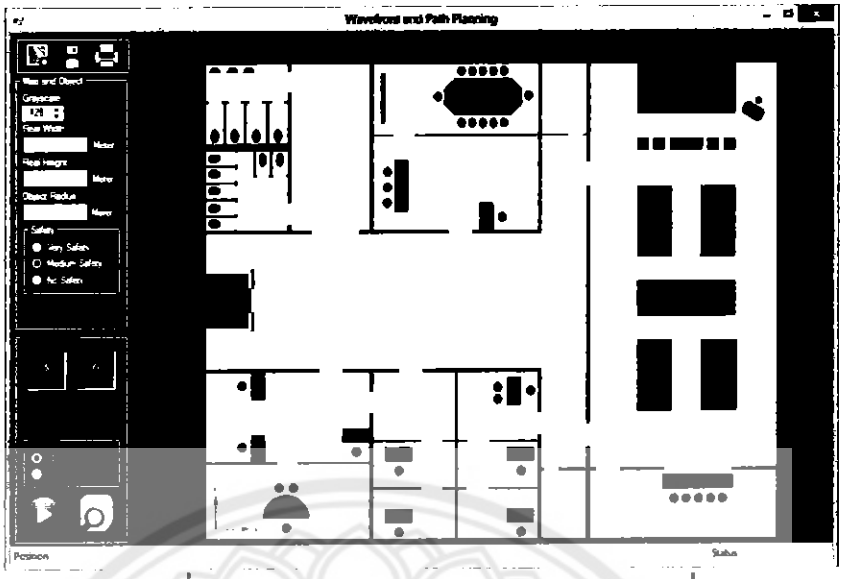
เมื่อโปรแกรมทำการติดตั้งเสร็จสมบูรณ์แล้ว ผู้ใช้งานสามารถใช้ Microsoft Visual Studio 2010 Express ในการพัฒนาโปรแกรมต่อไปได้

ข. ขั้นตอนการใช้งานโปรแกรม

ก่อนการใช้งานโปรแกรม ผู้ใช้งานโปรแกรมจำเป็นต้องทำการสร้างแผนที่ที่ต้องการใช้กับโปรแกรม โดยเป็นภาพขาวดำเท่านั้น เมื่อผู้ใช้งานมีแผนที่ที่ต้องการใช้งานแล้ว ให้ผู้ใช้งานทำการรันโปรแกรม โปรแกรมจะปรากฏหน้าต่าง ดังรูปที่ ข.1

รูปที่ ข.1 แสดงหน้าเริ่มต้นของโปรแกรม

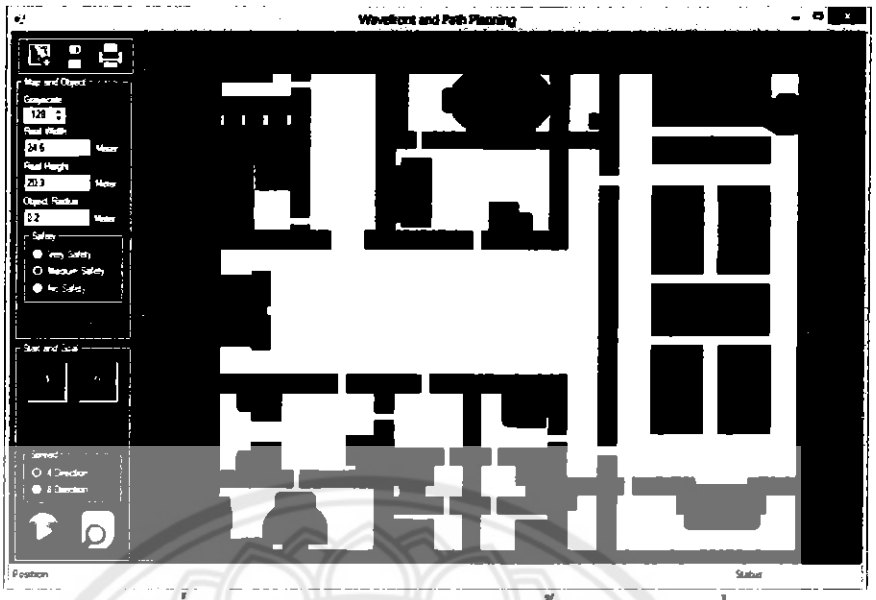
ในหน้าโปรแกรมนี้จะเป็นส่วนตั้งค่าต่างๆให้กับโปรแกรม ให้ผู้ใช้งานทำการ Import แผนที่ที่ได้สร้างขึ้นไว้แล้ว โดยการคลิกที่  เพื่อทำการนำแผนที่ที่สร้างขึ้นเข้าสู่โปรแกรม เพื่อเริ่มต้นการใช้งานและตั้งค่าต่างๆให้กับโปรแกรมต่อไป ดังรูปที่ ข.2



รูปที่ ข.2 หน้าต่างโปรแกรมหลังจาก Import แผนที่

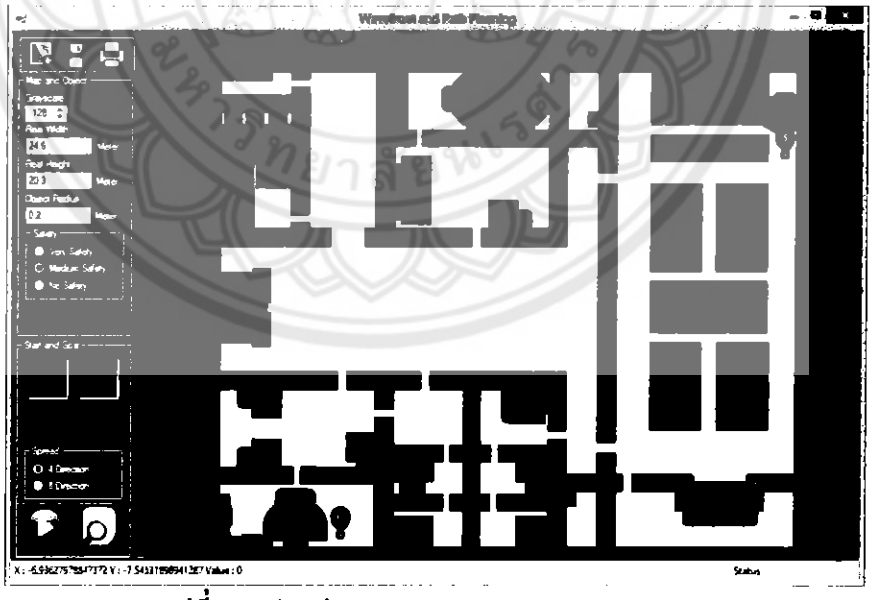
จากนั้นให้ผู้ใช้งานตั้งค่าของแผนที่จริง ได้แก่ กว้างจริง (Real Width) **สูงจริง (Real** Height) และขนาดรัศมีของหุ่นยนต์ (Object Radius) ดังรูปที่ ข.3 เมื่อกำหนดค่า**ต่างๆดังกล่าว**แล้วให้ผู้ใช้งานเลือกระดับความปลอดภัยให้กับตัวหุ่นยนต์ เพื่อที่โปรแกรมจะทำ**การขยายสิ่งกีด**ขวางในแผนที่ จากนั้นให้คลิกที่ [redacted] จะได้ดังรูปที่ ข.4

รูปที่ ข.3 ตัวอย่างการตั้งค่าโปรแกรม



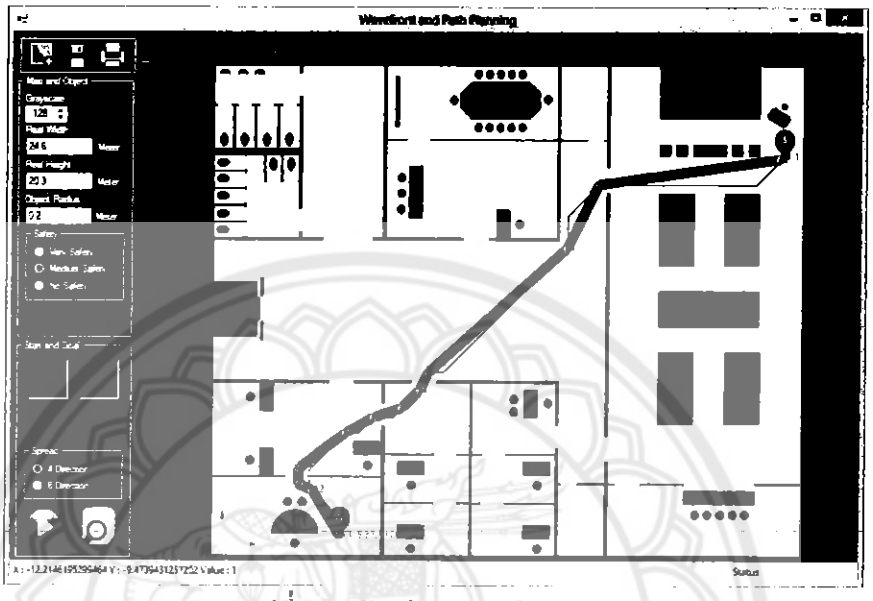
รูปที่ ข.4 หน้าต่างโปรแกรมหลังการตั้งค่าให้กับแผนที่

จากนั้นให้ผู้ใช้งานกำหนดจุดเริ่มต้น โดยการคลิก  ค้างแล้วลากไปไว้ในตำแหน่งที่ต้องการ และกำหนดจุดหมายปลายทางโดยการคลิก  ค้างแล้วลากไปไว้ในตำแหน่งที่ต้องการ ดังรูปที่ ข.5



รูปที่ ข.5 ตัวอย่างการกำหนดจุด Start และจุด Goal

จากนั้นให้ผู้ใช้งานโปรแกรมทำการเลือกการแพร่กระจาย ได้แก่ การแพร่กระจายแบบ 4 ทิศทาง และการแพร่กระจายแบบ 8 ทิศทาง จากนั้นทำการคลิก  เพื่อทำการรันโปรแกรม ดังตัวอย่างรูปที่ ข.6 ซึ่งในที่นี้เลือกการแพร่กระจายแบบ 4 ทิศทาง



รูปที่ ข.6 ตัวอย่างการรันโปรแกรม

จากรูปที่ ข.6 เมื่อผู้ใช้งานทำการรันโปรแกรมแล้ว โปรแกรมจะทำการแพร่กระจายในรูปแบบที่ผู้ใช้งานเลือก ในที่เลือกใช้การแพร่กระจายแบบ 4 ทิศทาง (4 Direction) จากนั้นโปรแกรมจะทำการค้นหาเส้นทาง แล้วจะแสดงผลลัพธ์ออกมาเป็นจุด Waypoint ที่หุ่นยนต์ต้องเคลื่อนที่เข้าหา ตั้งแต่จุดเริ่มต้นจนถึงจุดหมายปลายทาง ผู้ใช้งานสามารถเรียกดูพิกัดของจุด Waypoint ได้โดยการคลิกที่