ระบบระบุพิกัดและนำทางภายในอาคารโดยวิธีการมองภาพ สำหรับอากาศยานอัตโนมัติไร้นักบิน

นางสาวเบญจมาภรณ์ เณรชู

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเครื่องกลและระบบกระบวนการ
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2563

AN INDOOR LOCALIZATION AND NAVIGATION SYSTEM BY VISUALIZATION METHOD FOR UNMANNED AERIALS VEHICLE (UAV)

Benjamaporn Nenchoo

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Engineering in Mechanical and Process System Engineering

Suranaree University of Technology

Academic Year 2020

ระบบระบุพิกัดและนำทางภายในอาคารโดยวิธีการมองภาพสำหรับอากาศยาน อัตโนมัติไร้นักบิน

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(ผศ. ดร.ชโลธร ธรรมแท้)
ประธานกรรมการ

สุรเดช ตัญตรัยรัตน์

(อ. ดร.สุรเดช ตัญตรัยรัตน์)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(รศ. ดร.จิระพล ศรีเสริฐผล)
กรรมการ

ชินภัทร

(ผศ. ดร.ชินภัทร ทิพโยภาส)
กรรมการ

(อ. ดร.ธีทัต ดลวิชัย)
กรรมการ

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
รองอธิการบดีฝ่ายวิชาการและพัฒนาความเป็นสากล

(รศ. ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

เบญจมาภรณ์ เณรชู : ระบบระบุพิกัดและนำทางภายในอาคารโดยวิธีการมองภาพสำหรับอากาศยานอัตโนมัติไร้นักบิน (AN INDOOR LOCALIZATION AND NAVIGATION SYSTEM BY VISUALIZATION METHOD FOR UNMANNED AERIALS VEHICLE (UAV)) อาจารย์ที่ปรึกษา : อาจารย์ ดร.สุรเดช ตัญตรัยรัตน์, 122 หน้า.

ในปัจจุบันมนุษย์นิยมใช้เทคโนโลยีเข้ามามีบทบาทในการช่วยรับมือต่อภัยพิบัติทางธรรมชาติมากขึ้นและหนึ่งในเทคโนโลยีที่นิยมใช้ก็คือ อากาศยานอัตโนมัติไร้นักบิน (Unmanned Aerials Vehicle: UAV) ซึ่งถูกนิยมใช้ในการค้นหาและช่วยเหลือผู้ประสบภัย ส่วนใหญ่การทำภารกิจโดยใช้ UAV จะนิยมใช้กับเหตุการณ์ที่เกิดขึ้นภายนอกอาคารหรือภายนอกสถานที่ปิดอื่น ๆ เนื่องจากการทำภารกิจภายนอกอาคารสามารถระบุพิกัดของ UAV ได้โดยใช้ระบบบอกพิกัด Global Positioning Systems (GPS) ซึ่งการทำภารกิจภายในอาคารไม่สามารถระบุพิกัด โดยใช้ระบบบอกพิกัด GPS ได้ แต่ในบางครั้งก็ไม่สามารถปฏิเสธได้ว่ามีความจำเป็นที่ต้องช่วยเหลือผู้ประสบภัยในพื้นที่ปิดโดยใช้ UAV ผู้วิจัยได้เล็งเห็นถึงความสำคัญและผลกระทบของปัญหานี้ จึงได้ทำงานวิจัยนี้เพื่อให้สามารถระบุพิกัดและควบคุมการเคลื่อนที่ของโดรนแบบอัตโนมัติภายในอาคารได้โดยใช้วิธีการมองภาพด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ซึ่งงานวิจัยฉบับนี้จะมีการใช้เทคโนโลยีการตรวจจับวัตถุ (Object detection) โดยเทคนิค Deep learning เพื่อทำการเทรนโมเดลตรวจจับวัตถุคือ อากาศยาน 4 ใบพัด (Quadrotor) รุ่น Parrot bebop 2 ซึ่งใช้งานร่วมกับกล้อง ZED เพื่อระบุตำแหน่ง แบบ 3 มิติ โดยมีจุดกำเนิด (Origin point) อยู่ที่ตำแหน่งกล้อง ZED โดยกล้องจะถูกติดตั้งโดยไม่มีการเคลื่อนที่ งานวิจัยฉบับนี้ควบคุมการเคลื่อนที่ของโดรนด้วยระบบควบคุม P โดยมีค่าความคลาดเคลื่อนโดยเฉลี่ยไม่เกิน 0.1 เมตร และโมเดลที่ใช้สำหรับตรวจจับวัตถุมีค่าเฉลี่ยความแม่นยำประมาณ 0.7272 ซึ่งระบบสามารถควบคุมโดรนให้ทำภารกิจได้สำเร็จ

สาขาวิชา วิศวกรรมเครื่องกล ลายมือชื่อนักศึกษา เบญจมาภรณ์

ปีการศึกษา 2563 ลายมือชื่ออาจารย์ที่ปรึกษา สุรเดช ตัญตรัยรัตน์

BENJAMAPORN NENCHOO : AN INDOOR LOCALIZATION AND NAVIGATION SYSTEM BY VISUALIZATION METHOD FOR UNMANNED AERIALS VEHICLE (UAV). THESIS ADVISOR : SURADET TANTRAIRATN, Ph.D., 122 PP.

INDOOR POSITIONING SYSTEM/OBJECT DETECTION/UAV

Nowadays, human beings are increasingly using technology to helping to cope with natural disasters. And one of the most popular technologies is Unmanned Aerials Vehicle (UAV), which is commonly used to search and rescue victims. Most UAV is used for outdoors missions due to, Global Positioning Systems (GPS) able to determine the positioning of UAV in the outdoor environment. However, it cannot be denied that there is a need to help victims in enclosed spaces using UAV. Therefore, this research aims to be able to automatically locate and control the movement of the drone in the indoor environment by using a ZED stereo camera with AI object detection. Deep learning technique used to train the object detection model to detect Parrot Bebop 2 as a target object. This study controls the drone by P controller with an average error of 0.10 meter and an object detection model with an average accuracy of about 0.7272. The system can control the drone to perform a mission accomplished.

School of Mechanical Engineering

Academic year 2020

Student's Signature

Advisor's Signature

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงด้วยดี ทั้งนี้ผู้วิจัยขอขอบพระคุณบุคคลและหน่วยงานต่าง ๆ ที่ให้คำปรึกษา แนะนำ ชี้แนะแนวทาง และให้ความช่วยเหลืออย่างดีเสมอมา ได้แก่

อาจารย์ ดร.สุรเดช ตัญตรัยรัตน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้โอกาสทางการศึกษาระดับบัณฑิตศึกษา ให้คำแนะนำ ปรึกษา และแก้ปัญหาที่เกิดขึ้นโดยตลอด รวมทั้งช่วยตรวจทานแก้ไขวิทยานิพนธ์เล่มนี้จนเสร็จสมบูรณ์

ขอขอบคุณ ผู้ช่วยศาสตราจารย์ ดร.พรเทพ ราชนาวี และนายกีรติ กีรติเรขา สำหรับให้คำแนะนำ และความช่วยเหลือในการใช้อุปกรณ์ Motion capture เป็นอย่างดี

ขอขอบคุณ คณาจารย์ในสาขาวิชาวิศวกรรมเครื่องกลที่คอยติดตามการดำเนินงานให้กำลังใจ และให้คำแนะนำ

ขอขอบคุณ เจ้าหน้าที่เลขานุการ สาขาวิชาวิศวกรรมเครื่องกล ที่คอยให้ความช่วยเหลือและให้คำแนะนำในการเตรียมเอกสารต่าง ๆ ในการทำวิทยานิพนธ์เป็นอย่างดี

ขอขอบคุณ มหาวิทยาลัยเทคโนโลยีสุรนารี และสาขาวิชาวิศวกรรมเครื่องกล สำหรับโอกาสในการศึกษาและทุนการศึกษา

ขอขอบคุณพี่ ๆ และน้อง ๆ บัณฑิตศึกษา รวมทั้งน้อง ๆ นักศึกษาปริญญาตรีสาขาวิชาวิศวกรรมเครื่องกลทุกคนที่คอยให้ความช่วยเหลือ ให้กำลังใจ และให้คำแนะนำในการเรียนและการทำวิจัยด้วยดีเสมอมา

ขอขอบคุณ นายกีรติ เลิศทินรัตน์ ที่ให้คำแนะนำทางด้านการเขียนโปรแกรม ให้กำลังใจ และคอยช่วยสนับสนุนการทำวิจัยจนสำเร็จด้วยดี

สุดท้ายนี้ ขอขอบพระคุณบิดา มารดา ที่ให้การอุปการะอบรมเลี้ยงดู ผลักดัน อดทน และส่งเสริมการศึกษา รวมถึงเป็นกำลังใจที่สำคัญในการฝ่าฟันอุปสรรค์ที่เกิดขึ้นจนสำเร็จการศึกษา

เบญจมาภรณ์ เณรชู

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหาการวิจัย

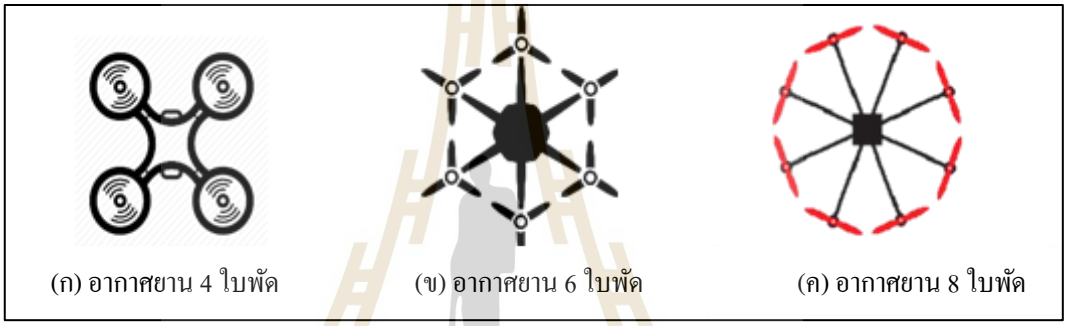
ภัยพิบัติทางธรรมชาตินั้นมีมาตั้งแต่อดีตจนกระทั่งปัจจุบันและยังคงเกิดขึ้นอยู่อย่างต่อเนื่อง ไม่ว่าจะเป็นแผ่นดินไหว อุทกภัย หรืออัคคีภัยก็ตาม ซึ่งเป็นหน้าที่ของมนุษย์ที่ต้องปรับตัวและหาวิธีรับมือให้สามารถอยู่รอดจากภัยพิบัติทางธรรมชาติในกรณีฉุกเฉินได้ ในยุคปัจจุบันมนุษย์นิยมใช้เทคโนโลยีเข้ามามีบทบาทในการช่วยรับมือต่อภัยพิบัติทางธรรมชาติมากขึ้น และหนึ่งในเทคโนโลยีที่เป็นที่นิยมใช้ได้แก่ อากาศยานอัตโนมัติไร้นักบิน (Unmanned Aerials Vehicle: UAV) ซึ่งถูกใช้ในการค้นหาและช่วยเหลือผู้ประสบภัย โดยจากบทความวิชาการเรื่อง “The Future of Drone Use, Information Technology and Law Series” ของ Bart Custers และคณะ (2016) ได้ทำการแบ่งประเภทของ UAV ตามลักษณะเฉพาะออกเป็น 3 ประเภทหลัก ๆ ได้แก่ ประเภทปีกตรึง (Fixed wing) ประเภทปีกหมุน (Rotorcraft) ซึ่งจะแบ่งออกได้อีก 2 ประเภท คือ อากาศยานหลายใบพัด (Multi rotor) และเฮลิคอปเตอร์ ส่วนประเภทสุดท้าย ได้แก่ ประเภทผสม (Hybrid) เช่น อากาศยานแบบปรับมุมเอียงมอเตอร์ (Tilt rotor) เป็นต้น ประเภทของ UAV ที่กล่าวมาข้างต้นแสดงตัวอย่างดังรูปที่ 1.1

(ก) ประเภทปีกหมุน (ข) ประเภทปีกตรึง (ค) ประเภทผสม

รูปที่ 1.1 ตัวอย่างประเภทของอากาศยานอัตโนมัติไร้นักบิน (UAV)

บทความวิชาการของ Bart Custers และคณะ (2016) ยังได้กล่าวอีกว่า UAV แต่ละประเภทนั้นมีลักษณะที่แตกต่างกันทำให้มีข้อได้เปรียบและข้อจำกัดแตกต่างกันไป ซึ่งความแตกต่างของ UAV

ทั้ง 3 ประเภท คือ ประเภทปีกหมุนสามารถขึ้น-ลงในแนวดิ่งได้ แต่สำหรับแบบปีกตรึงจำเป็นต้องใช้พื้นที่ในการขึ้นบินและลงจอดหรือก็คือจำเป็นต้องมีการใช้รันเวย์ในการขึ้นบินและลงจอด แต่ความเร็วในการเดินทางจะเร็วกว่าแบบปีกหมุน ส่วนประเภทผสมจะรวมข้อดีของแต่ละประเภทไว้ก็คือ สามารถขึ้น-ลงในพื้นที่จำกัดได้ และเมื่อปรับมุมมอเตอร์จะสามารถทำความเร็วในการเดินทางได้เร็วเหมือนแบบปีกตรึง ซึ่งประเภทของ UAV ที่พบเห็นได้บ่อยคือ ประเภทปีกหมุนแบบหลายใบพัด (Multi rotor) เช่น อากาศยาน 4 ใบพัด (Quadrotor) อากาศยาน 6 ใบพัด (Hexarotor) อากาศยาน 8 ใบพัด (Octorotor) เป็นต้น ตัวอย่างรูปแบบของอากาศยานประเภทปีกหมุนแบบหลายใบพัดแสดงดังรูปที่ 1.2



รูปที่ 1.2 ตัวอย่างรูปแบบของอากาศยานประเภทปีกหมุนแบบหลายใบพัด (Multirotor)

การใช้งานของ UAV ที่มีรูปแบบและประเภทที่หลากหลายนี้ขึ้นอยู่กับลักษณะและวัตถุประสงค์ของการนำไปใช้งาน เช่น การถ่ายภาพยนต์ การสำรวจ การกู้ชีพฉุกเฉิน การขนส่ง เป็นต้น โดยส่วนใหญ่การทำภารกิจโดยใช้ UAV จะนิยมใช้กับเหตุการณ์ที่เกิดขึ้นภายนอกอาคารหรือภายนอกสถานที่ปิดอื่น ๆ เนื่องจากการทำภารกิจภายนอกอาคารสามารถระบุพิกัดของ UAV ได้โดยใช้ระบบ Global Navigation Satellite System (GNSS) ซึ่งโดยทั่วไปจะนิยมใช้เครือข่ายดาวเทียมของระบบ Global Positioning Systems (GPS) ซึ่งเป็นดาวเทียมของสัญชาติสหรัฐอเมริกา การทำภารกิจภายในอาคารไม่สามารถระบุพิกัดโดยใช้ระบบบอกพิกัด GPS ได้ แต่ในบางครั้งก็ไม่สามารถปฏิเสธได้ว่ามีความจำเป็นที่ต้องช่วยเหลือผู้ประสบภัยหรือทำภารกิจอื่น ๆ ในพื้นที่ปิดโดยใช้ UAV และปัญหาเรื่องของระบบระบุพิกัดในพื้นที่ปิดก็เป็นปัญหาที่สำคัญมากในการปฏิบัติภารกิจ เนื่องจากมีผลต่อการนำทางและการปฏิบัติภารกิจให้สำเร็จตามที่ตั้งเป้าหมายไว้

ดังนั้น ทางผู้วิจัยได้เล็งเห็นถึงความสำคัญและผลกระทบของปัญหาเรื่องการระบุพิกัดในอาคารจึงได้ทำงานวิจัยนี้เพื่อให้สามารถระบุพิกัดภายในอาคารได้โดยใช้วิธีการระบุพิกัดด้วย

กล้องถ่ายภาพสามมิติ (Stereo camera) และเพื่อนำทาง UAV ให้เข้าสู่พื้นที่ปฏิบัติภารกิจภายในอาคารได้ จากการศึกษาทำให้ผู้วิจัยเลือกที่จะใช้วิธี Visualization เนื่องจากเป็นวิธีการแก้ปัญหาที่เหมาะกับการช่วยเหลือผู้ประสบภัยภายใต้เงื่อนไขที่กล่าวมาข้างต้นมากที่สุด ซึ่งงานวิจัยฉบับนี้จะมีการใช้เทคโนโลยีการตรวจจับวัตถุ (Object detection) โดยเทคนิคการเรียนรู้แบบเชิงลึก (Deep learning: DL) ด้วยโครงสร้างโมเดล YOLO (You Only Look Once) เพื่อทำการเรียนรู้โมเดลที่ใช้ในการตรวจจับวัตถุอากาศยาน 4 ใบพัด (Quadrotor) รุ่น Parrot Bebop 2 ซึ่งใช้งานร่วมกับกล้อง ZED เพื่อระบุตำแหน่งของ Parrot Bebop 2 แบบ 3 มิติ โดยมีจุดกำเนิด (Origin point) พิกัด X, Y และ Z เป็น 0, 0, 0 ตามลำดับ อยู่ที่ตำแหน่งกล้อง ZED โดยกล้องจะถูกติดตั้งโดยไม่มีการเคลื่อนที่

งานวิจัยฉบับนี้ผู้วิจัยคาดหวังว่าจะสามารถระบุพิกัดและนำทาง Parrot bebop 2 ภายในอาคารได้ นอกจากนั้นงานวิจัยฉบับนี้สามารถนำไปพัฒนาต่อโดยประยุกต์ใช้กับเทคโนโลยี SLAM และนำไปใช้ประโยชน์ในการทำภารกิจต่าง ๆ เช่น ช่วยเหลือผู้ประสบภัยพิบัติในพื้นที่ปิด การจัดการโกดังสินค้า การตรวจสอบภายในอาคาร เป็นต้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อให้สามารถระบุพิกัดของ Parrot Bebop 2 ภายในอาคารโดยใช้วิธีการระบุพิกัดด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ได้

1.2.2 เพื่อให้สามารถพัฒนาระบบนำทางแบบอัตโนมัติของ Parrot Bebop 2 ภายในอาคารโดยใช้วิธีการระบุพิกัดด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ได้

1.3 ขอบเขตของการวิจัย

1.3.1 การทดลองทำในสภาพแวดล้อมแบบปิด (Indoor environment) โดยมีขนาดพื้นที่ทำการทดลองที่มีความกว้าง 3 เมตร ความลึก 3 เมตร และความสูง 1.6 เมตร โดยประมาณ

1.3.2 UAV เป้าหมายที่ใช้ในการทำการทดลองในงานวิจัยนี้คือ Parrot Bebop 2

1.3.3 กล้องที่ใช้ในการทดลองเป็นกล้องถ่ายภาพสามมิติยี่ห้อ Stereo labs รุ่น ZED

1.3.4 การระบุพิกัดของ UAV จะระบุพิกัดเป็น 3 มิติ คือ X, Y, และ Z เทียบกับตำแหน่งของกล้องโดย X แทนระยะตามความกว้างของพื้นที่ Y แทนระยะตามความสูงของพื้นที่ และ Z แทนระยะตามความลึกของพื้นที่เทียบกับตำแหน่งของกล้อง ZED

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ระบบสามารถระบุพิกัดและควบคุมการเคลื่อนที่ของ Parrot Bebop 2 ภายในอาคารโดยใช้วิธีการระบุพิกัดด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ได้

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

การพัฒนาระบบระบุพิกัดและนำทางอากาศยานอัตโนมัติด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ผู้วิจัยได้ทำการศึกษาค้นคว้าวรรณกรรมและงานวิจัยต่างๆ ที่เกี่ยวข้องเพื่อใช้ในการวิจัยและพัฒนา ได้แก่ ระบบควบคุมพื้นฐานของอากาศยานอัตโนมัติประเภท 4 ใบพัด ระบบปฏิบัติการของหุ่นยนต์ (ROS) ระบบระบุพิกัดในอาคาร การตรวจจับวัตถุในภาพด้วย Machine learning กล้องถ่ายภาพสามมิติ (Stereo camera) เป็นต้น ดังอธิบายในหัวข้อต่อไปนี้

2.1 ระบบพื้นฐานของอากาศยานอัตโนมัติประเภท 4 ใบพัด

โดยปกติแล้วนักบินสามารถควบคุมการเคลื่อนที่และท่าทางการบินของ UAV ได้โดยการบังคับวิทยุ ซึ่งส่งคำสั่งไปควบคุมระบบควบคุมการบินในบอร์ดควบคุม ระบบควบคุมการบินจะส่งคำสั่งควบคุมไปยังพื้นบังคับ (Control surface) ที่ใช้สำหรับการควบคุมท่าทางการบินของ UAV ให้เป็นตามคำสั่งของนักบิน สำหรับอากาศยานหลายใบพัดจะใช้การควบคุมความเร็วรอบของมอเตอร์โดยการส่งกระแสไฟที่ไม่เท่ากัน เกิดเป็นความเร็วรอบที่ต่างกัน เมื่อมอเตอร์หมุนด้วยความเร็วรอบที่ต่างกันทำให้แรงยก (Lift force) ที่เกิดขึ้นนั้นต่างกันไปด้วย จึงทำให้เกิดแรงคู่ควบที่สามารถควบคุมท่าทาง Roll, Pitch และ Yaw ของอากาศยานหลายใบพัดแบบหมุนได้ ซึ่งสามารถอธิบายหลักการทำงานได้ตามหัวข้อดังต่อไปนี้

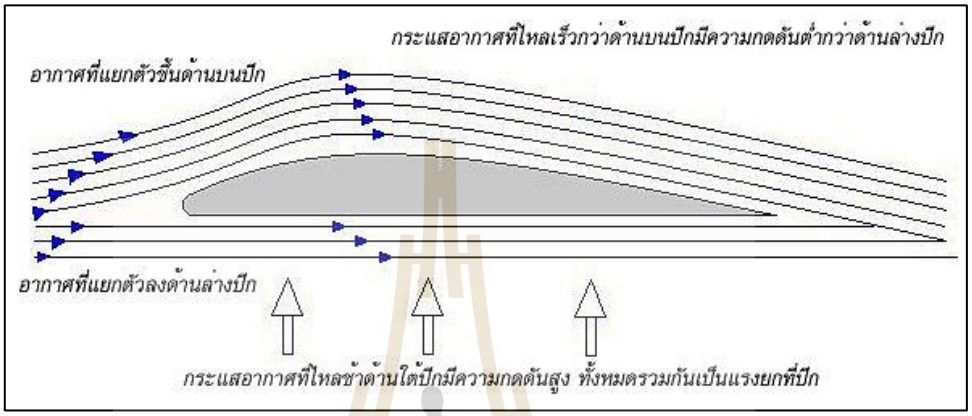
2.1.1 แรงทางอากาศพลศาสตร์

แรงทางอากาศพลศาสตร์มีความเกี่ยวข้องในด้านการบินเป็นอย่างมาก เนื่องจากเป็นแรงที่กระทำและส่งผลโดยตรงกับอากาศยานขณะที่ทำการเคลื่อนที่ผ่านอากาศ ซึ่งเกิดจากการที่ความดันและแรงเฉือนที่เกิดขึ้นจากกระแสอากาศที่ไหลผ่านวัตถุในบริเวณต่าง ๆ นั้นไม่เท่ากัน จึงเกิดเป็นแรงและโมเมนต์ขึ้นมา (John Anderson, 2017) โดยแรงที่กระทำต่ออากาศยานขณะลอยตัวประกอบด้วยแรงหลัก ๆ ดังนี้

2.1.1.1 แรงยก (Lift)

แรงยกเป็นแรงที่เกิดเมื่อเกิดความแตกต่างของความดันระหว่างด้านบนและด้านล่างของปีกตามทฤษฎีของเบอร์นูลลี่ (Bernoulli's Principle) คือเมื่อกระแสอากาศมีความเร็วเพิ่มขึ้นจะส่งผลให้ความดันอากาศลดลง การที่ปีกของเครื่องบินหรือใบพัดของ UAV มีรูปทรงแอร์ฟอล์ยทำให้กระแสอากาศที่ไหลผ่านมีความเร็วที่แตกต่างกัน โดยที่บริเวณด้านบนจะมี

ความเร็วมากกว่าด้านล่าง ซึ่งส่งผลให้ความดันอากาศด้านบนน้อยกว่าด้านล่างและเกิดเป็นแรงยกในทิศทางที่ตั้งฉากกับทิศทางของความเร็วสามารถแสดงได้ดังรูปที่ 2.1 นอกจากนั้นทฤษฎีนี้ยังสามารถใช้อธิบายแรงยกที่เกิดขึ้นจากการหมุนของใบพัดของ UAV ได้เช่นกันเนื่องจากลักษณะรูปทรงของใบพัดมีลักษณะใกล้เคียงกับรูปทรงแอร์ฟอล์ยของปีกเครื่องบิน



รูปที่ 2.1 ลักษณะการไหลของกระแสอากาศผ่านรูปทรงแอร์ฟอล์ย

2.1.1.2 แรงต้าน (Drag)

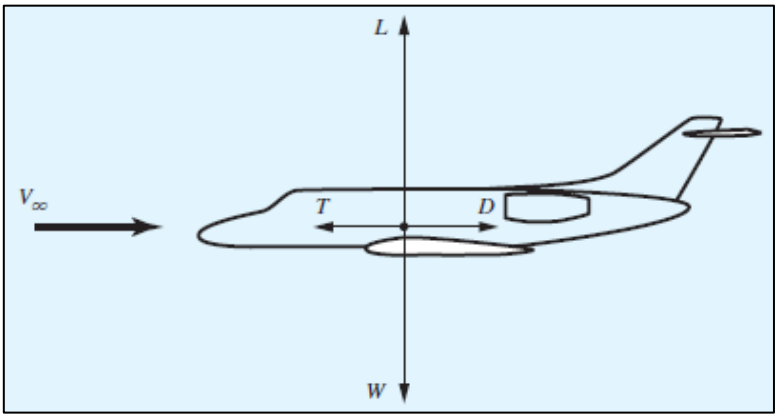
จากเอกสารวิชาการเรื่องอากาศพลศาสตร์ โดย อิศระ เชิดชู (2005) ได้ให้นิยามไว้ว่า “แรงต้านเป็นแรงที่มีทิศทางขนานกับแนวการเคลื่อนที่ของกระแสอากาศ และกระทำกับอากาศยานในทิศทางตรงข้ามกับทิศทางการบิน (Flight path) โดยทั่วไปแล้วรูปทรงของอากาศยานจะถูกออกแบบให้มีแรงต้านเกิดขึ้นน้อยที่สุดเพื่อให้เครื่องบินทำงานได้เต็มสมรรถนะ”

2.1.1.3 แรงขับ (Thrust)

แรงขับเป็นแรงที่เกิดจากกำลังของเครื่องยนต์ทำให้อากาศยานสามารถเคลื่อนที่ไปข้างหน้าหรือในทิศทางที่สวนกับทิศทางของความเร็วกระแสอากาศได้ โดยสำหรับอากาศยานประเภทหลายใบพัดนั้น แรงขับจะเป็นแรงที่สามารถทำให้เคลื่อนที่ไปในทิศทางต่าง ๆ ได้ โดยแรงขับนั้นจะต้องมีมากกว่าแรงต้านจึงจะสามารถทำให้อากาศยานเคลื่อนที่ได้

2.1.1.4 น้ำหนัก (Weight)

น้ำหนักเป็นแรงที่เกิดจากน้ำหนักของอากาศยานโดยกระทำอยู่ที่จุดศูนย์กลางมวลของอากาศยานและมีทิศพุ่งลงตั้งฉากเข้าสู่พื้นโลก โดยที่แรงทั้ง 4 ทางอากาศพลศาสตร์นสามารถแสดงและอธิบายได้ดังรูปที่ 2.2



รูปที่ 2.2 ทิศทางของแรงทางอากาศพลศาสตร์ทั้ง 4 แรง (John Anderson, 2017)

2.1.2 การเคลื่อนที่ในแกนต่าง ๆ ของอากาศยาน

อากาศยานขณะทำการบินจะสามารถทำการเคลื่อนที่ได้ทั้งหมด 3 แกน คือ X, Y และ Z โดยการเคลื่อนที่รอบแกนต่าง ๆ ของอากาศยานจะมีชื่อเรียกที่แตกต่างกันไป ดังนี้

2.1.2.1 แกน Longitudinal หรือแกน X

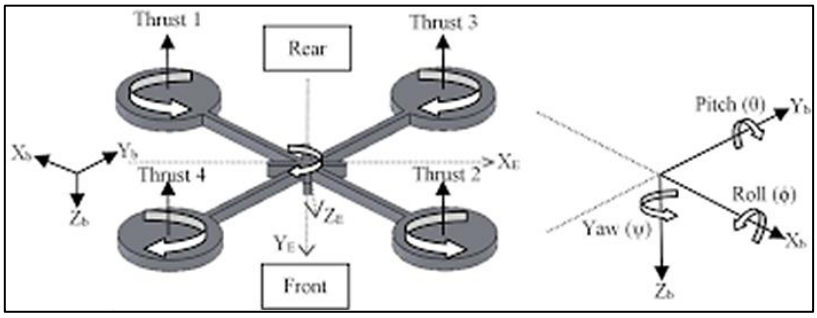
การหมุนรอบแกนจะเรียกว่าการ Roll โดยที่หมุนรอบแกนตามยาวไปทางซ้ายมีค่าเป็นลบ และไปทางขวามีค่าเป็นบวก การเคลื่อนที่ตามแกน X ไปข้างหน้าจะเป็นบวก ถอยหลังจะเป็นลบ เช่นเดียวกับการเคลื่อนที่ตามแกนของ Parrot Bebop 2

2.1.2.2 แกน Lateral หรือแกน Y

การหมุนรอบแกนจะเรียกว่า การ Pitch โดยที่การเงยขึ้นมีค่าเป็นลบ และก้มลงมีค่าเป็นบวก การเคลื่อนที่ตามแกน Y ไปทางขวาจะเป็นบวก ทางซ้ายจะเป็นลบ แต่สำหรับการเคลื่อนที่ของ Parrot Bebop 2 ไปทางซ้ายจะเป็นบวก ทางขวาจะเป็นลบ

2.1.2.3 แกน Vertical หรือแกน Z

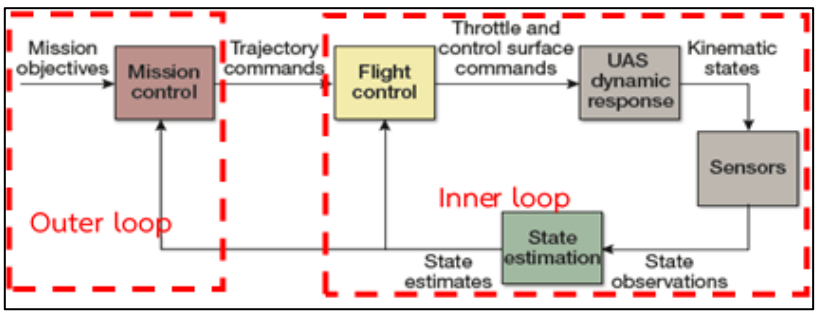
การหมุนรอบแกนจะเรียกว่า การ Yaw โดยที่การหมุนไปทางซ้ายมีค่าเป็นลบ และหมุนไปทางขวามีค่าเป็นบวก การเคลื่อนที่ตามแกน Z เคลื่อนที่ลงจะเป็นบวก เคลื่อนที่ขึ้นจะเป็นลบ แต่สำหรับการเคลื่อนที่ของ Parrot Bebop 2 เคลื่อนที่ขึ้นจะเป็นบวก เคลื่อนที่ลงจะเป็นลบ โดยที่การเคลื่อนที่ตามแกนและรอบแกนต่าง ๆ ของอากาศยาน 4 ใบพัด แสดงตัวอย่างดังรูปที่ 2.3



รูปที่ 2.3 การเคลื่อนที่ตามแนวแกนและการหมุนรอบแกนของอากาศยาน 4 ใบพัด (Endrowednes Kuantama, 2017)

2.1.3 ระบบควบคุมพื้นฐานของ UAV

เมื่อ UAV เคลื่อนที่ไปตามคำสั่งของนักบินแล้วจะมีการวัดค่าสถานะต่าง ๆ ของ UAV เช่น ความเร่ง ความสูง ความเร็วเชิงมุม เป็นต้น ด้วยอุปกรณ์เซ็นเซอร์ที่ติดตั้งอยู่ภายในบอร์ดควบคุมหรือที่ถูกนำมาติดตั้งเพิ่ม แต่ค่าสถานะบางอย่างไม่สามารถวัดได้โดยตรงด้วยอุปกรณ์เซ็นเซอร์จึงมีการใช้ระบบประมาณค่าสถานะ (State estimation) เพื่อทำให้สามารถส่งค่าสถานะกลับไปยังบอร์ดควบคุมเพื่อใช้ในการคำนวณ ให้ระบบรักษาเสถียรภาพได้ ซึ่งการควบคุมดังที่กล่าวมาข้างต้นนี้เป็นการควบคุมภายใน (Inner loop) หากมีการทำระบบการบินแบบอัตโนมัติ (Autopilot) ด้วยการกำหนดภารกิจและเส้นทางการบินเกิดขึ้น ระบบควบคุมจึงจำเป็นต้องมีการควบคุมภายนอก (Outer loop) เพื่อทำการควบคุมภารกิจ (Mission Control) และส่งคำสั่งเป็นพิกัดไปควบคุมระบบควบคุมภายในอีกทีเพื่อให้ UAV มุ่งหน้าไปยังตำแหน่งที่ต้องการ (Jeffrey D. Barton, 2012) โดยที่การทำงานของการควบคุมภายในและภายนอกสามารถอธิบายได้ด้วยแผนภาพดังรูปที่ 2.4

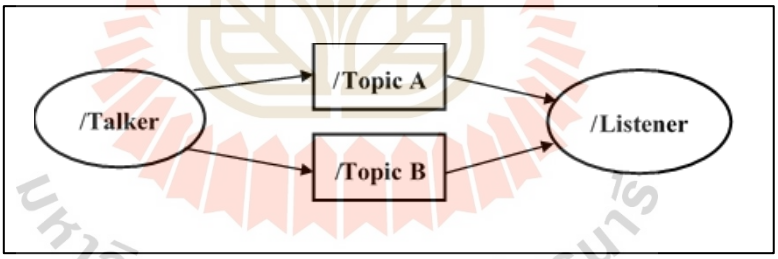


รูปที่ 2.4 การทำงานของระบบควบคุมของ UAV (Jeffrey D. Barton, 2012)

สำหรับในงานวิจัยนี้การควบคุมจะเน้นไปที่การควบคุมภายนอก แต่การควบคุมการเคลื่อนที่ของ Parrot Bebop 2 นั้นจะใช้การควบคุมที่แตกต่างจาก UAV โดยทั่วไปเพียงเล็กน้อย กล่าวคือ การควบคุมายนอกของ UAV โดยทั่วไปนั้นสามารถส่งค่าพิกัดไปยังการควบคุมภายในเพื่อให้ UAV เคลื่อนที่ไปที่ เป้าหมายได้เลย แต่สำหรับ Parrot Bebop 2 การควบคุมภายนอกนั้นจะทำได้โดยการส่งเป็นค่าความเร็วในแกนต่าง ๆ ไปให้แก่บอร์ดควบคุมแทนการส่งค่าเป็นพิกัด โดยที่ค่าของความเร็วในแต่ละแกนนั้นจะมีค่าอยู่ระหว่าง -1 ถึง 1

2.2 ระบบปฏิบัติการของหุ่นยนต์ (ROS)

ROS ย่อมาจาก “Robot Operation System” คือ เฟรมเวิร์คของการพัฒนาหุ่นยนต์ ซึ่งเฟรมเวิร์คคือระเบียบวิธีการทำงานของโปรแกรมที่ช่วยให้ทำงานได้สะดวกขึ้น ROS ทำงานด้วยการแบ่งหน้าที่เป็นหน่วยซึ่งถูกเรียกว่า “node” ในแต่ละ node จะมีโค้ดทำงานเฉพาะ node นั้น ๆ และสามารถรับ-ส่งข้อมูลของแต่ละ node ได้ผ่าน topics ซึ่งเป็นเสมือนช่องทางการสื่อสารของ node ซึ่งสามารถมี topics ได้มากกว่า 1 topics โดยที่หน่วยส่งจะเรียกว่า “Publisher” ทำหน้าที่ในการส่ง Message หรือข้อความที่จะสื่อสารไปยังหน่วยรับที่เรียกว่า “Subscriber” โดยหลักการทำงานโดยสังเขปของ ROS สามารถอธิบายได้ดังรูปที่ 2.5



รูปที่ 2.5 แผนภาพอธิบายการทำงานของ ROS โดยสังเขป

การส่งข้อความในรูปแบบ node ของ ROS ดังที่กล่าวมาข้างต้นนั้นจะช่วยให้การทำงานง่ายขึ้นเนื่องจากไม่จำเป็นต้องเขียนโค้ดควบคุมการทำงานของแต่ละส่วนรวมกันไว้ในโค้ดเดียว ดังนั้นการทำงานจึงสามารถทำได้อย่างสะดวกและรวดเร็วขึ้น โดยที่ ROS นั้นมีรูปแบบของการส่งข้อความระหว่าง node ที่แตกต่างกันไปตามประเภทของข้อความ (Message) ที่ต้องการส่ง ซึ่งสามารถตรวจสอบได้ที่เว็บไซต์ ROS.org โดยรูปแบบการส่งนั้นจะมีทั้งหมด 3 รูปแบบ ได้แก่

1) ROS Message เป็นการส่งในรูปแบบ Stream คือ การส่งข้อมูลแบบต่อเนื่อง และมีการส่งผลการตอบสนองกลับมาผ่านฟังก์ชั่น Receive callback

2) ROS Service เป็นการส่งข้อมูลแบบไม่ต่อเนื่องคือ ส่งข้อความ (Request) ไปแล้วรอผลลัพธ์ (Response) ตอบกลับมา และอาจกำหนดเวลาให้มีการรอการตอบกลับของผลลัพธ์ไม่เกินเวลาที่ตั้งค่าไว้เพื่อไม่ให้ใช้ระยะเวลาในการรอมากเกินไป และระหว่างรอผล node ที่ส่งจะไม่สามารถทำงานอย่างอื่นได้

3) ROS Action Library เป็นการส่งข้อมูลที่คล้ายกับ ROS Service แต่จะมีการตอบกลับเป็นความก้าวหน้าอยู่เรื่อย ๆ และ node ที่ส่งก็ยังสามารถทำงานอื่น ๆ ได้ (Theppasith N, 2018)

การที่ใช้ ROS ในการช่วยพัฒนาหุ่นยนต์ทำให้การพัฒนานั้นสามารถใช้ภาษาในการเขียนโปรแกรมได้หลากหลาย นอกจากนั้นสามารถนำมาประยุกต์ใช้งานได้หลากหลาย เช่น การควบคุมหุ่นยนต์ผ่านแอพพลิเคชั่นบนมือถือ การควบคุมหุ่นยนต์ผ่าน Arduino การควบคุมโดรนผ่าน PX4 เป็นต้น

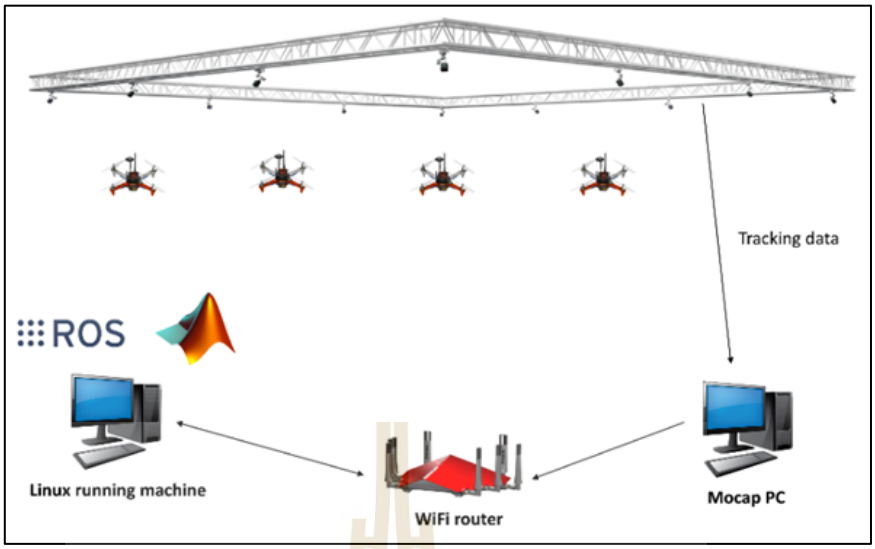
สำหรับในงานวิจัยฉบับนี้นั้น ROS จะถูกใช้เป็นเครื่องมือในการรับและส่งคำสั่งหรือข้อมูลต่าง ๆ ในการควบคุม Parrot bebop 2 อันได้แก่ คำสั่งควบคุมการเคลื่อนที่ของ Parrot bebop 2 รับและส่งข้อมูลพิกัดของ Parrot bebop 2 จากการประมวลผลของกล้อง ZED ไปยังระบบควบคุมการเคลื่อนที่ ส่งข้อมูลจากการตรวจจับวัตถุด้วย Artificial intelligence (AI) ไปยังระบบแปลงพิกัด

2.3 ระบบระบุพิกัดในอาคาร

ในปัจจุบันมีงานวิจัยจำนวนมากที่ได้ทำการแก้ปัญหาเรื่องระบบระบุพิกัดในอาคาร ซึ่งสามารถทำได้หลายวิธี โดยแต่ละวิธีจะมีข้อดีและข้อจำกัดที่แตกต่างกัน ดังกล่าวต่อไปนี้

2.3.1 เทคโนโลยีกล้องตรวจจับการเคลื่อนไหว (Motion capture)

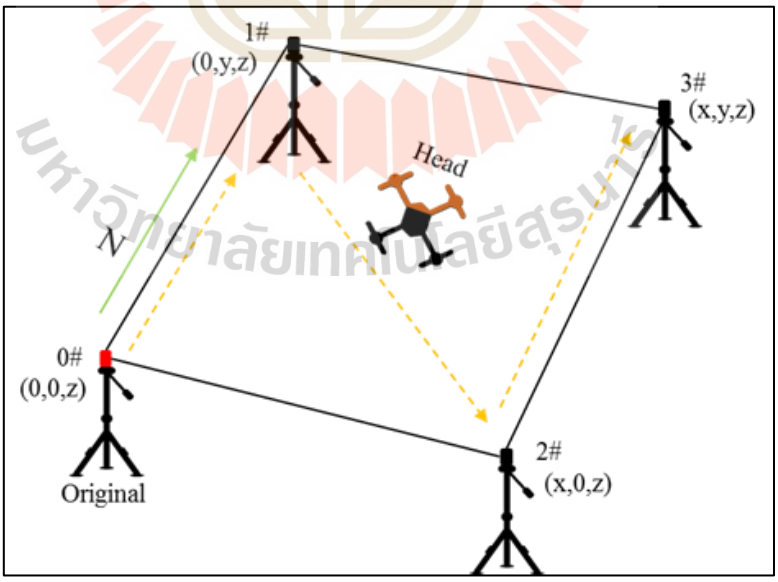
จากตัวอย่างงานวิจัยของ Shaima Al H. และคณะ (2015) และ Bo L. และ Normand P. (2018) ที่มีการใช้เทคโนโลยีกล้องตรวจจับการเคลื่อนไหว (Motion capture) ในการระบุพิกัดของอากาศยานอัตโนมัติในอาคาร นอกจากจะสามารถระบุพิกัดได้แล้วเทคโนโลยีนี้ยังสามารถบอกความเร็ว มุมในการหมุน และทิศทางของวัตถุได้ด้วย ซึ่งทำได้โดยมีกล้องตรวจจับการเคลื่อนที่ติดตั้งอยู่รอบบริเวณพื้นที่การทดสอบ 4 เครื่องขึ้นไป และมีการทำสัญลักษณ์สะท้อน (Markers) ติดไว้ที่ตัวเป้าหมายที่ต้องการ จะทำให้สามารถทราบตำแหน่ง ท่าทางการเคลื่อนที่ของวัตถุได้ ซึ่งวิธีนี้มีความแม่นยำสูง แต่มีข้อจำกัดคือการติดตั้งอุปกรณ์ทำได้ค่อนข้างใช้เวลานานไม่เหมาะสำหรับใช้ในกรณีฉุกเฉิน อีกทั้งการเคลื่อนย้ายยังทำได้ยากและค่าบำรุงรักษาอุปกรณ์มีราคาสูง รูปแสดงตัวอย่างระบบการระบุพิกัดและควบคุมอากาศยานสี่ใบพัดด้วยเทคโนโลยี Motion capture โดยสังเขป แสดงดังรูปที่ 2.6



รูปที่ 2.6 การทำงานของระบบระบุพิกัดในอาคารด้วยเทคโนโลยี Motion Capture (Rioku, 2019)

2.3.2 เทคโนโลยี Ultra-Wideband (UWB)

จากตัวอย่างงานวิจัยของ S. J. Ingram และคณะ (2004) และ A. Alarifi และคณะ (2016) ได้มีการใช้เทคโนโลยี Ultra-Wideband ในการระบุพิกัด ดังตัวอย่างรูปที่ 2.7



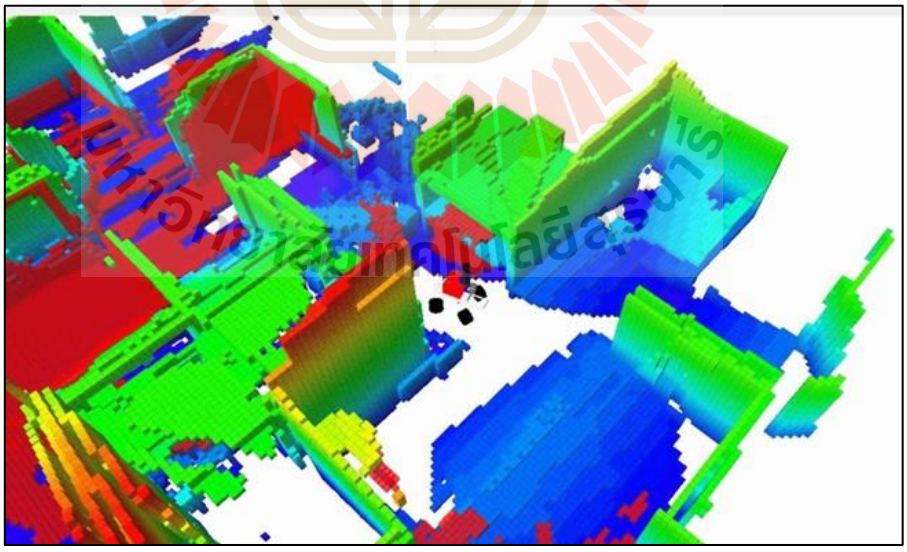
รูปที่ 2.7 การทำงานของระบบระบุพิกัดในอาคารด้วยเทคโนโลยี Ultra-Wideband (Debupt, 2018)

จากรูปที่ 2.7 สามารถอธิบายหลักการทำงานของการระบุพิกัดโดยใช้เทคโนโลยี Ultra-Wideband ได้ว่าเทคโนโลยีนี้จำเป็นต้องมีอุปกรณ์ปล่อยสัญญาณคลื่น UWB อย่างน้อย 4 ชิ้น โดยแต่ละชิ้นจะมีหมายเลขระบุอุปกรณ์ (ID number) เพื่อบอกจุดกำเนิด (Origin point) เมื่อคลื่น UWB ถูกส่งออกมากระทบกับอุปกรณ์รับสัญญาณที่ติดตั้งบนหุ่นยนต์หรือวัตถุที่ต้องการจึงสามารถทราบพิกัดของวัถุได้คล้ายกับการทำงานของระบบ GPS แต่อยู่ในอาคาร และมีความแม่นยำมากกว่า เทคโนโลยีนี้มีข้อจำกัดคือ การติดตั้งอุปกรณ์ทำได้ค่อนข้างใช้เวลานานไม่เหมาะสำหรับใช้ในกรณีฉุกเฉิน

2.3.3 เทคโนโลยีการทำแผนที่สามมิติ (Simultaneous localization and mapping: SLAM)

จากตัวอย่างงานวิจัยของ Sergio G. และคณะ (2016) และ Yuki E. และคณะ (2017) ได้มีการใช้เทคโนโลยีการทำแผนที่สามมิติ (Simultaneous localization and mapping: SLAM) เพื่อใช้ในการทำแผนที่และนำทางหุ่นยนต์ในพื้นที่ปิด แสดงตัวอย่างดังรูปที่ 2.8

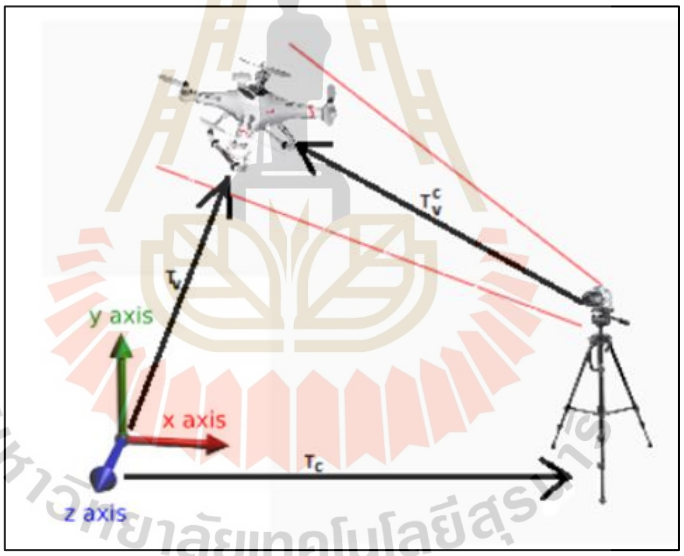
เทคโนโลยี SLAM นั้นมี 2 รูปแบบ คือ 2 มิติ และ 3 มิติ โดยรูปแบบ 2 มิติ จะใช้เซนเซอร์ Lidar ในการสแกนพื้นที่ ส่วนแบบ 3 มิติ ใช้กล้องถ่ายภาพสามมิติ (Stereo camera) หรืออาจใช้ร่วมกับ Lidar ก็ได้ เมื่อทำการสำรวจพื้นที่ด้วยเซนเซอร์แล้วจะได้แผนที่เพื่อใช้สำหรับการทำระบบนำทางถัดไป แต่การใช้เทคโนโลยี SLAM มีข้อจำกัดคือหากพื้นที่ที่ต้องการนั้นมีขนาดใหญ่จำเป็นต้องใช้ระยะเวลานานในการประมวลผลแผนที่



รูปที่ 2.8 การทำงานของระบบระบุพิกัดในอาคารด้วยเทคโนโลยีการทำแผนที่สามมิติ Simultaneous localization and mapping (SLAM) (S. HIGGINS, 2016)

2.3.4 เทคโนโลยีกล้องถ่ายภาพสามมิติ (Stereo camera)

ดังตัวอย่างงานวิจัยของ Samvram S. และคณะ (2019) ซึ่งจะสามารถระบุพิกัดแบบสามมิติได้โดยใช้เทคนิคการมองภาพหรือVisualization ด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ร่วมกับการตรวจจับวัตถุ (Object detection) ด้วยเทคนิคการเรียนรู้แบบเชิงลึก โมเดล Fast R-CNN ซึ่งย่อมาจาก Fast Region Convolution Neural Network ซึ่งหากทำการเปรียบเทียบกับเทคโนโลยีที่ได้กล่าวมาก่อนหน้าแล้วนั้นจะพบว่า เทคโนโลยีนี้มีข้อดีคือ สามารถเคลื่อนย้ายอุปกรณ์ได้สะดวกและรวดเร็วกว่าวิธีการใช้ Motion capture และวิธีการใช้ UWB นอกจากนั้นการประมวลผลข้อมูลนั้นยังสามารถทำได้รวดเร็วกว่าวิธีการใช้ SLAM แต่ถึงอย่างไรก็ตามเทคโนโลยีนี้ยังมีข้อจำกัดในด้านการมองภาพที่ต้องให้วัตถุที่ต้องการระบุพิกัดนั้นอยู่ภายในขอบเขตการมองของกล้องตลอดเวลา หรือจะต้องไม่มีสิ่งกีดขวางมาบดบังการมองเห็นวัตถุเป้าหมายของกล้อง การทำงานของระบบระบุพิกัดด้วยกล้องถ่ายภาพสามมิตินั้น สามารถแสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 การทำงานของระบบระบุพิกัดในอาคารด้วยเทคโนโลยีกล้องถ่ายภาพสามมิติร่วมกับการตรวจจับวัตถุด้วย AI (Sahu Samvram, 2019)

จากการศึกษาข้อมูลตามที่กล่าวมาข้างต้นนั้น เพื่อให้สอดคล้องกับวัตุประสงค์ของงานวิจัยที่เน้นช่วยเหลือในพื้นที่ประสบภัยที่ต้องการความรวดเร็วและใช้เวลาในการเตรียมการน้อย ดังนั้น ในงานวิจัยฉบับนี้จึงเลือกใช้วิธีการระบุพิกัดภายในอาคารด้วยเทคโนโลยีกล้องถ่ายภาพสามมิติ (Stereo Camera) เนื่องจากเป็นวิธีที่มีความแม่นยำสูงและเคลื่อนย้ายได้สะดวกเหมาะแก่การใช้ทำภารกิจตามวัตถุประสงค์

2.4 กล้องถ่ายภาพสามมิติ (Stereo vision system)

เทคโนโลยีกล้องถ่ายภาพสามมิติเป็นเทคโนโลยีที่พัฒนาขึ้นเพื่อให้สามารถมองภาพแบบ 3 มิติได้ โดยจำลองการทำงานคล้ายกับการรับรู้ระยะด้วยตาทั้ง 2 ข้างของมนุษย์ ซึ่งตาแต่ละข้างจะรับภาพในมุมมองที่ต่างกันทำให้ได้ระยะของวัตถุที่แตกต่างกัน และทำการส่งภาพที่ได้จากตาทั้งสองข้างไปประมวลผลที่สมอง ซึ่งความแตกต่างของระยะที่ต่างกันจากสองภาพนี้ทำให้สมองสามารถประมวลผลระยะของวัตถุได้ เช่นเดียวกันกับการทำงานของกล้องถ่ายภาพสามมิติที่ประกอบด้วยกล้องความละเอียดสูง 2 ชิ้น ทำให้ได้ภาพสองภาพที่มีระยะแตกต่างกันและคำนวณออกมาเป็นระยะความลึกได้โดยใช้ทฤษฎีสามเหลี่ยมมุมฉาก (Triangulation Theory) เกิดเป็นภาพแบบ 3 มิติขึ้นมากระบวนการนี้จะถูกเรียกว่า “Stereo Reconstruction” (Luis E. Ortiz และคณะ, 2018)

ในปัจจุบันกล้องถ่ายภาพสามมิตินั้นสามารถคำนวณระยะห่างและความลึกของวัตถุโดยไม่จำเป็นต้องให้ผู้ใช้งานเขียนโปรแกรมขึ้นมาใหม่ เนื่องจากมีไลบรารี่ให้สามารเรียกใช้งานได้ ดังนั้นผู้ใช้งานเพียงแค่ทำการเรียกใช้ไลบรารี่ของกล้องก็จะได้ข้อมูล ระยะ X, Y และ Z รวมทั้งความลึกของวัตถุด้วย นอกจากนั้นกล้องถ่ายภาพสามมิติในตลาดปัจจุบันก็มีให้เลือกใช้งานหลากหลาย ซึ่งคุณสมบัติของกล้องถ่ายภาพสามมิติที่นิยมในปัจจุบันแสดงดังตารางที่ 2.1

ตารางที่ 2.1 ตารางเปรียบเทียบคุณสมบัติของกล้องถ่ายภาพสามมิติรุ่นต่าง ๆ

รุ่น	**Depth FOV**	**ระยะความลึก (เมตร)**	**Frame rate (FPS)**
Intel RealSense Depth Camera D435i (Intel Cooperation, 2018)	Horizontal: 90° Vertical: 59°	0.105 - 10	90
Microsoft Kinect v2. (Al-Naji, 2017)	Horizontal: 70° Vertical: 60°	0.5 - 4.5	30
Stereolabs ZED camera (StereoLabs, 2018)	Horizontal: 90° Vertical: 60°	0.3 - 25	100

สำหรับในงานวิจัยนี้เลือกใช้กล้อง Stereolabs ZED ในการระบุพิกัดตัวอย่างดังรูปที่ 2.10 โดยขั้นตอนการคำนวณระยะความลึกและพิกัดต่าง ๆ ของวัตถุประกอบไปด้วยขั้นตอนสำคัญ ได้แก่ การคาลิเบรต การแก้ไขภาพ และการประเมินความลึก โดยสามารถอธิบายได้ดังต่อไป

รูปที่ 2.10 ตัวอย่างกล้องถ่ายภาพสามมิติ Stereolabs ZED (Luis E. Ortiz และคณะ, 2018)

2.4.1 การคาลิเบรต (Calibration)

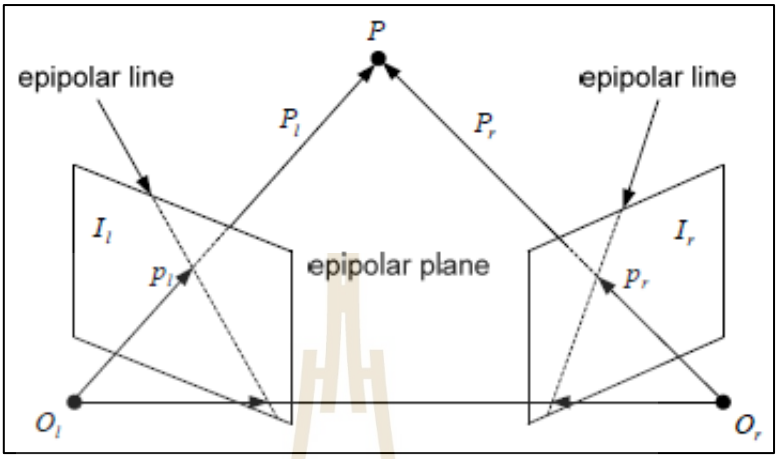
ขั้นตอนการคาลิเบรตจะเป็นขั้นตอนการประเมินค่าตัวแปรทั้งภายในและภายนอกของกล้อง ตัวแปรภายในจะประกอบด้วย ระยะโฟกัส (Focal length: *fx* ,*fy*) ตำแหน่งจุดศูนย์กลางเลนส์ (Principal point coordinates: *Cx, Cy*) ขนาดรัศมี (Radial: *K1*, *K2*, *K3*) และเส้นสัมผัสวง (Tangential: *P1, P2*) ซึ่งค่าเหล่านี้จะช่วยลดการบิดของรูปเนื่องมาจากผลกระทบของเลนส์ที่มีความโค้ง และช่วยให้สามารถแสดงภาพแบบ 3 มิติได้ ส่วนตัวแปรภายนอกจะเป็นตัวแปรที่สัมพันธ์กับระบบอ้างอิงของโลกจริง เช่นทิศทางการเคลื่อนที่ การหมุน เป็นต้น โดยปกติแล้วค่าตัวแปรต่าง ๆ เหล่านี้ของกล้อง ZED จะมีค่าตั้งต้นที่ทางผู้ผลิตได้ทำการตั้งค่าไว้ให้สามารถใช้งานได้แล้ว ดังนั้นการคาลิเบรตกล้อง ZED จึงเป็นการคาลิเบรตแบบ Self-calibration ที่ผู้ใช้งานนั้นสามารถทำได้โดยง่ายผ่าน ZED SDK (Luis E. Ortiz และคณะ, 2018)

2.4.2 การแก้ไขภาพ (Rectification)

การแก้ไขภาพเป็นอีกขั้นตอนสำคัญขั้นตอนหนึ่งเพื่อนำไปสู่การคำนวณพิกัดของวัตถุได้อย่างแม่นยำ เนื่องจากภาพที่ได้จากกล้องทั้ง 2 ข้างมองภาพในมุมมองที่แตกต่างกันและบางครั้งภาพที่ได้อาจเกิดการเอียง แสดงดังตัวอย่างรูปที่ 2.12 ดังนั้นจึงต้องมีการปรับแก้ไขภาพเพื่อให้ขั้นตอนการคำนวณพิกัดของวัตถุแม่นยำมากที่สุด และนอกจากนั้นขั้นตอนนี้ยังช่วยในการประมาณค่าความต่าง (Disparity) ของตำแหน่งวัตถุจากทั้งสองภาพในขั้นตอนถัดไปด้วย การปรับแก้ไขภาพนั้นสามารถทำได้ โดยทฤษฎีเรขาคณิตอีพิโพลาร์ (Epipolar Geometry) (Luis E. Ortiz และคณะ, 2018)

เรขาคณิตอีพิโพลาร์ช่วยให้หาตำแหน่งของภาพที่มาจากจุดเดียวกันบนวัตถุได้ด้วยการสร้างเส้นอ้างอิงอีพิโพลาร์ (Epipolar line) ผ่านตำแหน่งอ้างอิงของภาพแต่ละภาพ โดยที่เส้นอีพิโพลาร์คือเส้นที่ตัดกันระหว่างระนาบของภาพกับระนาบอีพิโพลาร์ซึ่งระนาบอีพิโพลาร์นี้

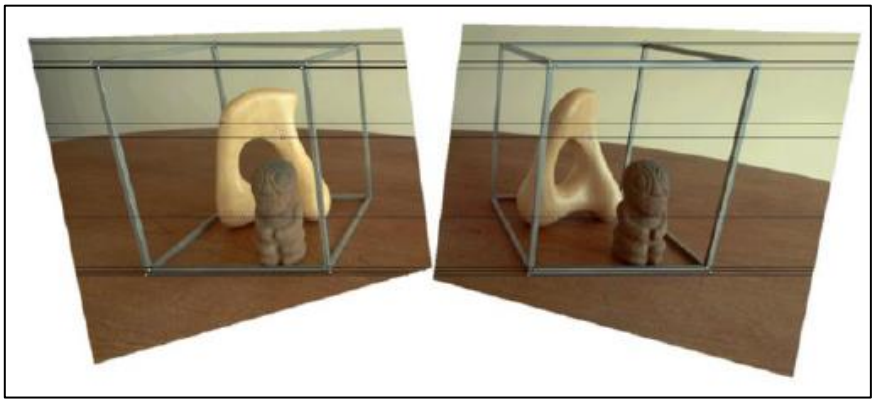
เกิดจากจุดศูนย์กลางของกล้องทั้งสองและจุดบนวัตถุ (วิบูลย์ แสงวีระพันธุ์ศิริ และคณะ, 2548) แสดงดังรูปที่ 2.11



รูปที่ 2.11 ลักษณะของรูปเรขาคณิตอีพิโพลาร์ (วิบูลย์ แสงวีระพันธุ์ศิริ และคณะ, 2548)

หลังจากนั้นจะทำการจับคู่พิกเซลที่ตรงกันจากทั้งสองภาพและทำการปรับภาพให้เส้นอ้างอิงของแต่ละภาพอยู่ในระดับเดียวกัน โดยภาพที่ผ่านกระบวณการปรับแก้แล้วแสดงดังรูปที่ 2.13 สำหรับกระบวณการปรับแก้ภาพของกล้อง ZED นั้น สามารถเรียกใช้ภาพที่ผ่านการปรับแก้แล้วได้เลย เนื่องจาก ZED SDK มีการคำนวณภาพให้พร้อมใช้งานเรียบร้อย

รูปที่ 2.12 ภาพตัวอย่างก่อนการทำการปรับแก้ภาพ (Kris Kitani, 2017)



รูปที่ 2.13 ภาพตัวอย่างหลังการทำการปรับแก้ภาพ (Kris Kitani, 2017)

2.4.3 การจับคู่สเตอริโอ (Stereo matching)

Stereo matching คือ กระบวนการเทียบหาจุดเหมือนโดยจะทำการเทียบในหน่วยพิกเซล แต่เนื่องจากการเทียบหาทีละ 1 พิกเซลนั้นใช้เวลานานและอีกทั้งยังไม่แม่นยำอีก ด้วยจึงต้องทำการเทียบหาเป็นกรอบทีละหลายพิกเซล หรือ window size (เช่น 7x7 พิกเซล) จากนั้น เมื่อได้ตำแหน่งพิกเซลที่เหมือนกันจะทำการหาค่าระยะห่างของพิกเซลที่เหมือนกันระหว่างรูป ทางด้านซ้ายและด้านขวา (Disparity) โดยวัตถุที่อยู่ไกลค่าของระยะห่างของภาพทั้งสองจะน้อยกว่า วัตถุที่อยู่ใกล้ หลังจากนั้นนำค่าความต่างที่ได้ไปพล็อทลงใน Disparity map ได้เป็นความลึก ของวัตถุ ซึ่งเรียกวิธีนี้ว่า “Local stereo matching” นอกจากนั้นยังมีการจับคู่แบบ Global stereo matching ด้วย ซึ่งวิธีนี้จะทำให้พื้นผิวของวัตถุที่ได้นั้นเรียบกว่า (Satyarth Praveen, 2019)

2.4.4 การประมาณความลึก (Depth Estimation)

การประมาณความลึกของกล้อง ZED นั้นสามารถทำได้โดยใช้ทฤษฎีสามเหลี่ยม มุมฉาก (Triangulation) จากแบบจำลองทางเรขาคณิตของภาพที่ผ่านการแก้ไข (Rectification) เพื่อไม่ให้ภาพเกิดการบิดงอ จากรูปที่ 2.14 เป็นแบบจำลองทางเรขาคณิตโดยที่ให้ระยะโฟกัส ของทั้งกล้องซ้ายและขวามีค่าเท่ากัน ($fl = fr = f$)

โดย X_L, Y_L, Z_L คือ พิกัดของวัตถุที่จุด L

B คือ ระยะห่างระหว่างจุดศูนย์กลางของกล้องทั้งสองข้าง

x_l^l คือ ระยะตามแนวแกน X ระหว่างจุดกึ่งกลางของภาพถึงจุดบนวัตถุของกล้องทางด้านซ้าย

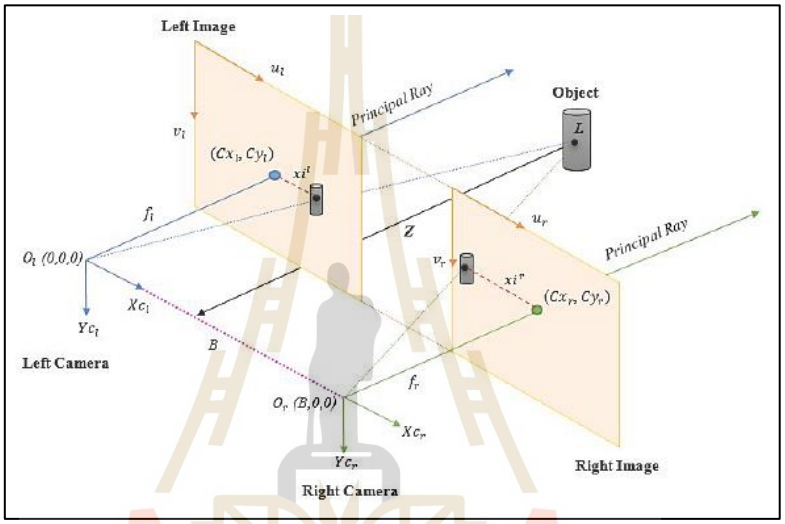
x_l^r คือ ระยะตามแนวแกน X ระหว่างจุดกึ่งกลางของภาพถึงจุดบนวัตถุของกล้องทางด้านขวา

y_l^l คือ ระยะตามแนวแกน Y ระหว่างจุดกึ่งกลางของภาพถึงจุดบนวัตถุของกล้องทางด้านซ้าย

y_l^r คือ ระยะตามแนวแกน Y ระหว่างจุดกึ่งกลางของภาพถึงจุดบนวัตถุของกล้องทางด้านขวา

Cx_l และ Cy_l คือ พิกัดจุดศูนย์กลางของกล้องทางด้านซ้าย

Cx_r และ Cy_r คือ พิกัดจุดศูนย์กลางของกล้องทางด้านขวา



รูปที่ 2.14 แบบจำลองเรขาคณิตหลักการทำงานของกล้อง ZED (Luis E. Ortiz และคณะ, 2018)

หากพิจารณารูปที่ 2.14 ตามทฤษฎีสามเหลี่ยมมุมฉาก (Triangulation) จะสามารถเขียนได้ดังสมการที่ 2.1 และสมการที่ 2.2 ดังนี้

$$\frac{X_L}{x_I^l} = \frac{Z_L}{f} \tag{2.1}$$

$$\frac{X_L - B}{x_I^r} = \frac{Z_L}{f} \tag{2.2}$$

ปรับรูปสมการที่ 2.1 และสมการที่ 2.2 เพื่อหาค่าความลึกของวัตถุให้สมการอยู่ในรูป Z_L จะได้ดังสมการที่ 2.3

$$Z_L = \frac{fX_L}{x_I^l} = \frac{fX_L - fB}{x_I^r} \tag{2.3}$$

พิจารณาสมการที่ 2.3 และปรับรูปให้สามารถคำนวณหาค่า X_L จะสามารทำได้ดังแสดงต่อไปนี้

$$\frac{fX_L}{x_I^l} = \frac{fX_L - fB}{x_I^r} \tag{2.4}$$

$$X_L = \frac{X_L - B}{x_I^r} x_I^l \tag{2.5}$$

$$X_L = \frac{X_L x_I^l}{x_I^r} - \frac{Bx_I^l}{x_I^r} \tag{2.6}$$

$$X_L - \frac{X_L x_I^l}{x_I^r} = -\frac{Bx_I^l}{x_I^r} \tag{2.7}$$

$$X_L (x_I^r - x_I^l) = -Bx_I^l \tag{2.8}$$

จากสมการที่ 2.4-2.8 ดังแสดงมาข้างต้นนั้น จะสามารถหาพิกัด X ของวัตถุที่จุด L ได้โดยใช้สมการที่ 2.9 โดยแสดงดังต่อไปนี้

$$X_L = \frac{Bx_I^l}{(x_I^l - x_I^r)} \tag{2.9}$$

จากสมการที่ 2.9 จะพบว่า ค่าความต่างระหว่างระยะตามแนวแกน X ระหว่างจุดกึ่งกลางของภาพถึงจุดบนวัตถุของกล้องทางด้านซ้ายและขวา $x_I^r - x_I^l$ นั้นคือ ค่าความต่าง

ของวัตถุที่วัดโดยกล้องทั้ง 2 ข้างหรือก็คือ ค่า Disparity นอกจากนั้นหากนำสมการที่ 2.9 แทนค่าในสมการที่ 2.3 จะทำให้สามารถหาค่าความลึก Z ของวัตถุที่จุด L ได้โดยใช้สมการที่ 2.10 โดยแสดงดังต่อไปนี้

$$Z_L = \frac{fB}{(x_I^l - x_I^r)} \tag{2.10}$$

จากสมการที่ 2.10 พบว่า ค่าความลึกที่วัดได้จากกล้องสเตอริโอนั้นมีความสัมพันธ์กับค่า Disparity โดยที่ความลึกที่ได้นั้นจะแปรผกผันกับค่าความต่างนี้ กล่าวคือ หากวัตถุอยู่ไกลจากกล้องมีค่า Z_L มากจะทำให้ค่าความต่างนั้นน้อยลง ซึ่งเป็นไปดังที่กล่าวไว้ในหัวข้อ 2.4.3 นอกจากนั้น หากทำการพิจารณาแบบจำลองเรขาคณิตนี้ในแนวแกน Y เพื่อหาระยะ Y_L โดยที่กำหนดให้ภาพที่ผ่านกระบวนการแก้ไขภาพ (Rectification) แล้วจะมีค่า $y_I^l = y_I^r = y_I$ เนื่องจากภาพพิกเซลเดียวกันจะอยู่ในระนาบเดียวกัน และเมื่อทำการพิจารณาในทำนองเดียวกันกับการหาค่า X_L และ Z_L แล้ว จะทำให้สามารถหาค่า Y_L ได้ดังสมการที่ 2.11 โดยแสดงดังต่อไปนี้

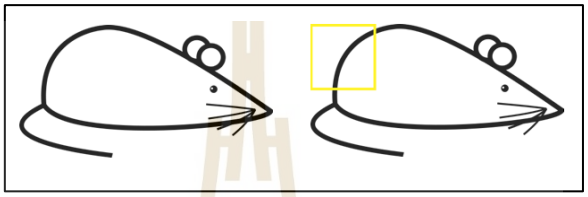
$$Y_L = \frac{y_I B}{(x_I^l - x_I^r)} \tag{2.11}$$

2.5 การตรวจจับวัตถุในภาพด้วยการเรียนรู้ของเครื่อง (Machine learning)

การเรียนรู้ของเครื่อง (Machine Learning: ML) เป็นเครื่องมือหนึ่งของปัญญาประดิษฐ์ (Artificial Intelligence: AI) ที่เน้นการใช้ตัวอย่างหรือประสบการณ์ในการเรียนรู้ โดยที่มนุษย์มีส่วนร่วมในการออกแบบระบบ และระบบจะเลือกจุดสำคัญจากตัวอย่างหรือประสบการณ์เหล่านั้น หลังจากที่ผ่านการเรียนรู้ด้วยตัวอย่างที่มากเพียงพอ เครื่องมือหรือระบบที่เรียนรู้แล้วจะสามารถนำไปใช้ในการประมวลผลตัวอย่างที่ไม่เคยพบมาก่อนได้ ซึ่งเครื่องมือการเรียนรู้ของเครื่อง (Machine Learning: ML) เป็นเครื่องมือที่นิยมใช้แก้ปัญหาในปัจจุบันโดยเฉพาะการเรียนรู้แบบเชิงลึก (Deep Learning: DL) ซึ่งเป็นหนึ่งในวิธีของการเรียนรู้ของเครื่อง ที่มีพื้นฐานจากโครงข่ายประสาทเทียม (Artificial Neural Networks) หลักการเบื้องต้นของโครงข่ายประสาทเทียมมาจากการจำลองการทำงานของเซลล์ประสาทของมนุษย์ (ปริญญา สงวนสัตย์, 2562) สำหรับการวิเคราะห์รูปภาพจะนิยมใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network: CNN)

2.5.1 โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network: CNN)

โครงข่ายประสาทเทียมแบบคอนโวลูชันเป็นโครงข่ายประสาทเทียมแบบ Bio-Inspired โดยจำลองการมองภาพเป็นพื้นที่ย่อยเป็นพิกเซลในการวิเคราะห์เมตริกซ์ของรูปภาพ และนำกลุ่มพื้นที่ย่อยนั้นมารวมกันเพื่อทำการแยกลักษณะเด่นของพื้นที่ย่อยนั้น เช่น ลายเส้น เป็นต้น (Natthawat Phongchit, 2018; PradyaSin, 2019) ตัวอย่างพื้นที่ย่อยบริเวณกรอบสีเหลือง ดังรูปที่ 2.15

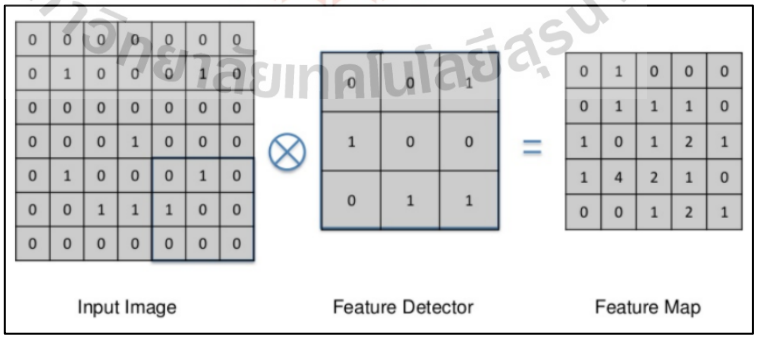


รูปที่ 2.15 ตัวอย่างการแบ่งรูปเป็นพื้นที่ย่อย (Natthawat Phongchit, 2018)

ขั้นตอนในการทำโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network: CNN) มีขั้นตอนดังนี้

2.5.1.1 Convolution

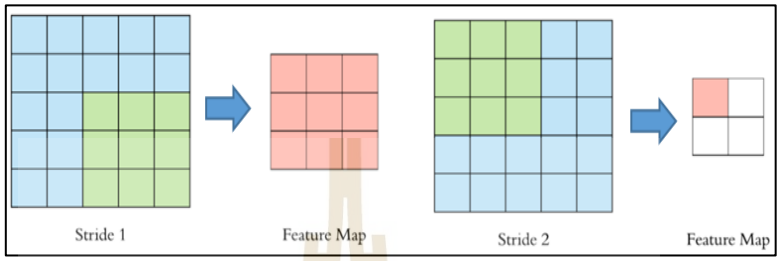
Convolution เป็นการทำการคูณเมทริกซ์ระหว่างรูปภาพ (Input image) กับตัวจับลักษณะของภาพ (Feature detector) ทำให้ได้ Feature map ดังตัวอย่างรูปที่ 2.16



รูปที่ 2.16 กระบวนการทำ Convolution (PradyaSin, 2019)

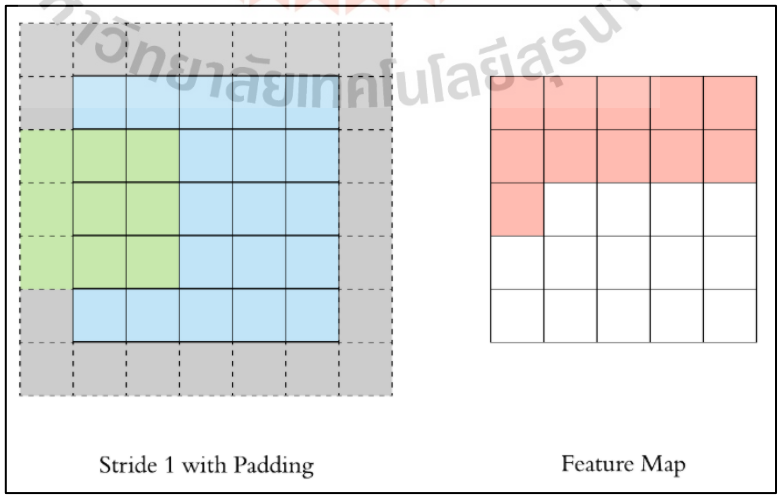
ตัวจับลักษณะภาพ (Feature Detector) หรือเรียกอีกชื่อหนึ่งว่า “ตัวกรอง (Filter)” เป็นเมทริกซ์ที่ช่วยคัดลักษณะเด่นของภาพออกมา ซึ่งตัวกรองนั้นมีลักษณะหลายแบบ

ขึ้นอยู่กับความต้องการของการคัดประเภทลักษณะเด่น นอกจากนั้นการเลื่อนตัวกรองบนภาพจำเป็นต้องมีเทคนิค 2 อย่างคือ Stride และ Padding โดยที่ Stride เป็นการกำหนดขั้น (Step) ของการเลื่อนตัวกรองบนรูปภาพ ถ้าขั้นมากขึ้นส่งผลให้ Feature map มีขนาดเล็กลงและพื้นที่ทับซ้อนน้อยลงด้วย ดังตัวอย่างรูปที่ 2.17



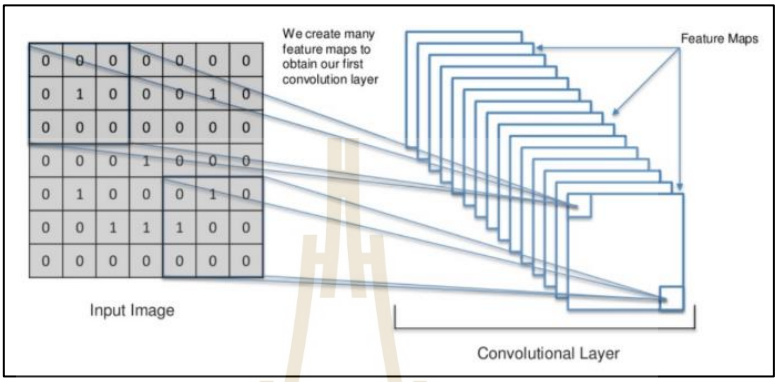
รูปที่ 2.17 ความสัมพันธ์ของการกำหนดค่าของ Stride กับขนาดของ Feature map (Natthawat Phongchit, 2018)

เนื่องจากการเลื่อนตัวกรองนั้นจะไม่ทำการเลื่อนไปนอกบริเวณที่เกินขอบเขตของภาพซึ่งจะส่งผลทำให้ภาพมีขนาดเล็กลง และอาจทำให้พลาดลักษณะเด่นของภาพที่อยู่บริเวณขอบด้วย ดังนั้นจึงต้องมีการทำ Padding คือการเติมค่ารอบพื้นที่เดิมเพื่อช่วยลดปัญหาที่กล่าวมาข้างต้น โดยปกติแล้วค่าที่เติมคือ 0 แสดงดังรูปที่ 2.18 บริเวณที่ถูกเติมจะแทนด้วยสีเทา (Natthawat Phongchit, 2018)



รูปที่ 2.18 การทำ Stride และ Padding (Natthawat Phongchit, 2018)

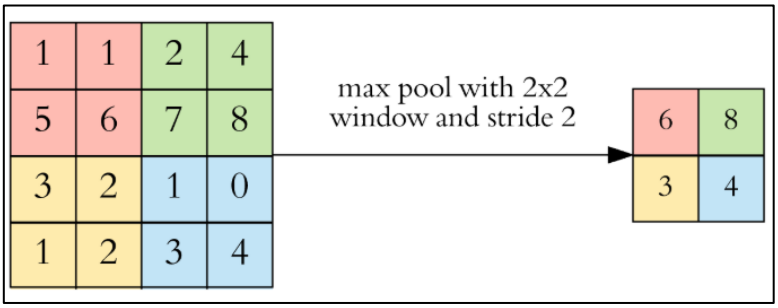
เมื่อทำการคูณเมทริกซ์กับตัวกรองแล้วจะได้ Feature map มาจำนวนหนึ่ง ซึ่งจะทำการเรียก Feature map เหล่านี้ว่า “Convolutional Layer” ดังรูปที่ 2.19 หลังจากนั้นจะนำมาผ่านกระบวนการปรับแต่ง (Rectifier) ด้วย ReLu Layer (PradyaSin, 2019) เพื่อให้โมเดลนั้นถูกเทรนและลู่เข้าเป้าหมายได้เร็วขึ้น (Keng Surapong, 2019)



รูปที่ 2.19 Feature map ที่ถูกเรียกว่า “Convolution layer” (PradyaSin, 2019)

2.5.1.2 Pooling

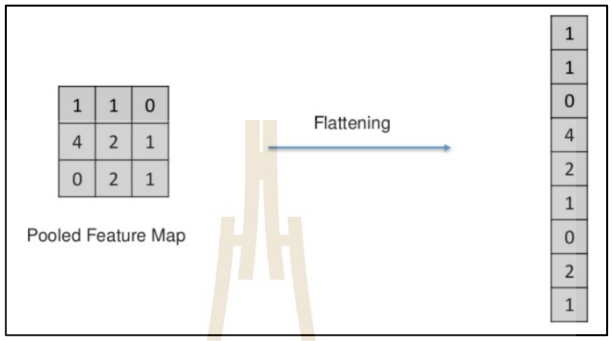
Pooling เป็นความสามารถของการย่อรูปชนิดหนึ่งเพื่อให้ภาพมีขนาดที่หลากหลายทำให้การจำแนกวัตถุทำได้ดีขึ้น Pooling มี 2 ประเภท คือ Max Pooling และ Mean Pooling ปกติจะนิยมใช้ Max Pooling เป็นตัวกรอง โดย Max Pooling เป็นตัวกรองที่จะทำการเลือกค่าที่มากที่สุดในบริเวณเมทริกซ์เดียวกันเพื่อคงเอกลักษณ์ของจุดเด่นในบริเวณนั้นไว้เพื่อให้ได้ Pooled Feature Map โดยขนาดตัวกรองของการทำ Max Pooling จะนิยมเรียกว่า “Pool size” ดังรูปที่ 2.20



รูปที่ 2.20 ตัวอย่างกระบวนการ Max Pooling (Natthawat Phongchit, 2018)

2.5.1.3 Flattening

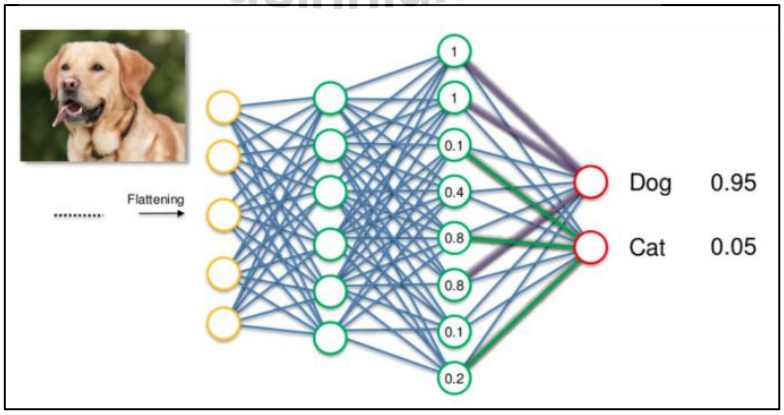
Flattening เป็นกระบวนการแปลง Pooled Feature Map ให้เป็นเมทริกซ์คอลัมน์เดียวเพื่อให้สะดวกต่อการวิเคราะห์ข้อมูลในขั้นตอนถัดไป ตัวอย่างการทำ Flattening แสดงดังรูปที่ 2.21



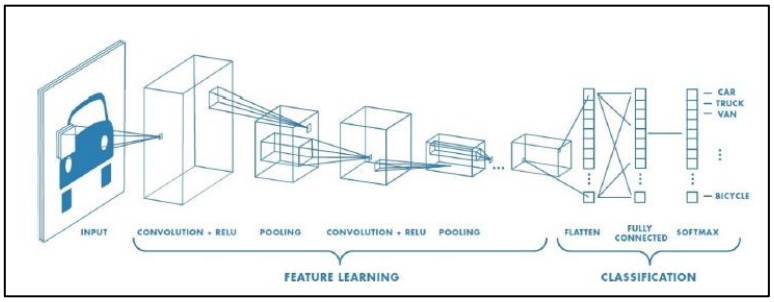
รูปที่ 2.21 ตัวอย่างกระบวนการ Flattening (PradyaSin, 2019)

2.5.1.4 Full Connection

Full Connection เป็นกระบวนการนำเมทริกซ์ Flattening ที่ได้จากการผ่านกระบวนการข้างต้นแล้วนั้นให้เข้าสู่กระบวนการเรียนรู้แบบเชิงลึก (Deep Learning) เพื่อทำการวิเคราะห์ความน่าจะเป็นของผลลัพธ์ของภาพ (Input image) ดังรูปที่ 2.22 ที่แสดงความเป็นไปได้ว่าเป็นสุนัขร้อยละ 95 และตัวอย่างภาพรวมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network: CNN) แสดงดังรูปที่ 2.23

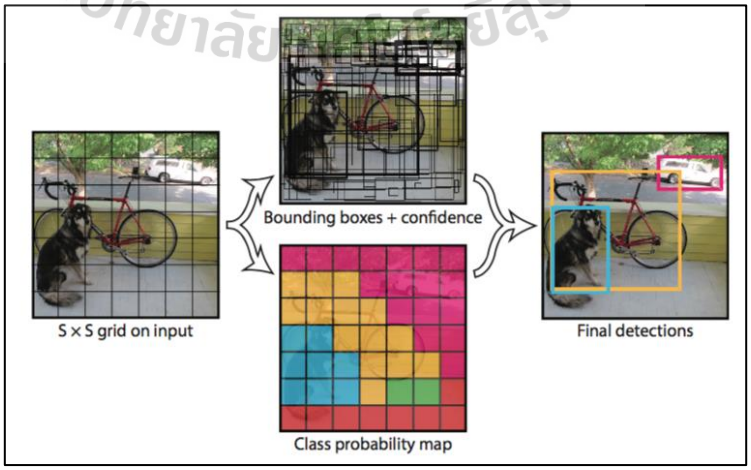


รูปที่ 2.22 กระบวนการวิเคราะห์ผลลัพธ์ในขั้นตอน Full connection (PradyaSin, 2019)



รูปที่ 2.23 ภาพรวมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Natthawat Phongchit, 2018)

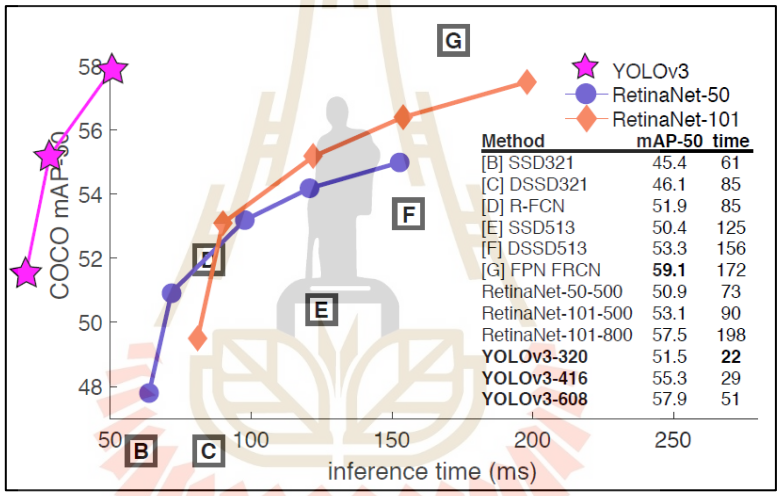
ในปัจจุบันโครงค่ายประสาทเทียมแบบคอนโวลูชันถูกนำไปพัฒนาจนเกิดเป็นอัลกอริทึมรูปแบบต่าง ๆ จำนวนมาก เช่น โครงข่ายเสนอพื้นที่ (Region Proposal Network: R-CNN) ที่นำเอาวิธีการเลือกลักษณะเด่นของภาพมาจาก CNN, Fast R-CNN และ Faster R-CNN ที่เร็วกว่า Fast R-CNN และ Single Shot Detector (SSD) นอกจากนั้นยังมีงานวิจัยของ J. Redmon และคณะ (2016) ที่ได้ทำการพัฒนาอัลกอริทึม YOLO (You Only Look Once) ขึ้นมาสำหรับใช้ในการทำการตรวจจับวัตถุ ซึ่งประสิทธิภาพในการทำงานนั้นดีกว่าและรวดเร็วกว่าโครงข่ายประเภท R-CNN ที่เป็นการทำงานแบบ 2 ขั้น (Two-stage Object-Detection Network) เนื่องจาก YOLO นั้นเป็นอัลกอริทึมการทำงานแบบ 1 ขั้น (Single stage Object-Detection Network) คือ ทำการทำนายตำแหน่งและขนาดของวัตถุพร้อมกับความน่าจะเป็นของประเภทต่าง ๆ พร้อมกัน (Sanparith Marukatat, 2018) ดังรูปที่ 2.24 หลักการทำงานของ YOLOv3 โดยละเอียดสามารถอธิบายดังหัวข้อที่ 2.5.2



รูปที่ 2.24 ขั้นตอนการทำงานของอัลกอริทึม YOLO (Marko B, 2019)

2.5.2 YOLOv3

YOLOv3 หรือ You Only Look Once Version 3 เป็นอัลกอริทึมที่สร้างมาเพื่อใช้ในการตรวจจับวัตถุในรูปภาพ (Object Detection) ซึ่งมีความแตกต่างกับการทำ Classification เนื่องจาก Object detection จะสามารถบอกตำแหน่งและชนิดของวัตถุในภาพได้ แต่ Classification นั้นจะสามารถทำได้แค่การบอกชนิดของวัตถุเท่านั้น โดยพื้นฐานการทำงานนั้นจะมีการใช้ CNN ทำการแยกลักษณะเด่นของวัตถุจากภาพในช่วงแรกโดยไม่มีการทำ Pooling ซึ่งการทำงานจะแบ่งภาพเป็น Grid cell เช่นเดียวกันกับการทำงานของ CNN อัลกอริทึมนี้เป็นงานวิจัยและพัฒนาโดย Joseph Redmon และคณะ ซึ่งจากการทดลองผู้พัฒนาได้อ้างว่าประสิทธิภาพในการทำงานนั้นมีความรวดเร็วกว่าอัลกอริทึมอื่น ๆ ซึ่งแสดงดังรูปที่ 2.25

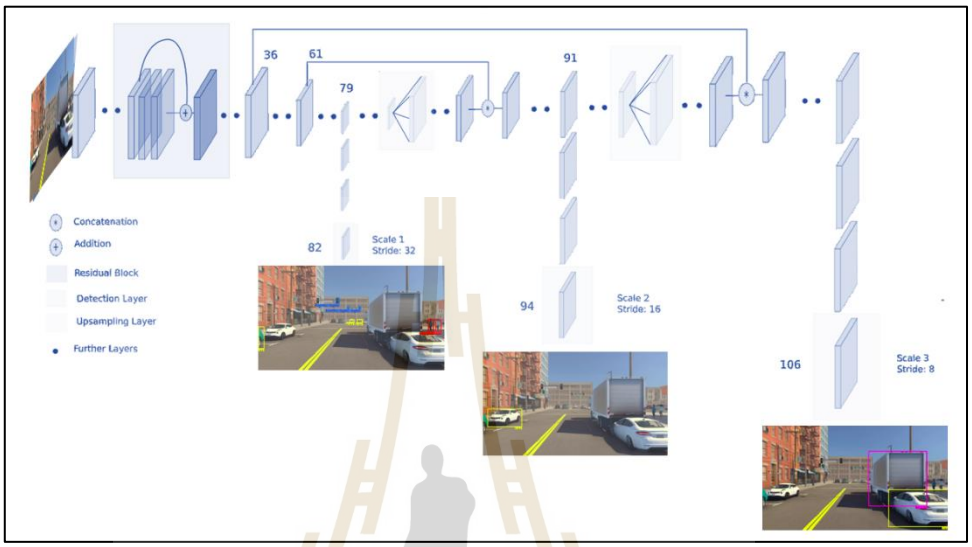


รูปที่ 2.25 กราฟระหว่างความแม่นยำ (Mean Average Precision: mAP) และเวลาในการประมวลผล (Inference time: ms) ของ YOLOv3 เทียบกับอัลกอริทึมอื่น ๆ (Joseph Redmon และคณะ, 2018)

2.5.2.1 โครงสร้างสถาปัตยกรรมของ YOLOv3

โครงสร้างสถาปัตยกรรมของ YOLOv3 ประกอบด้วย CNN ทั้งหมด 53 ชั้น และ อื่น ๆ อีก 53 ชั้น รวมเป็น 106 ชั้นของการทำงาน ซึ่งในชั้นการทำงาน 106 ชั้นนี้จะมีชั้นสำหรับการทำ Detection 3 ชั้น คือ ชั้นที่ 82, 94 และ 106 เนื่องจากโครงสร้างจะทำการ Down sample ด้วยวิธี Convolutional แทนการ Pooling เพื่อป้องกันการสูญเสียจากคุณลักษณะที่มีค่าต่ำ ๆ ทำให้ตรวจจับวัตถุขนาดเล็กได้ดีขึ้น การ Down sample ทำทั้งหมด 3 ครั้ง ซึ่งมี Stride คือ 32, 16 และ 8 ตามลำดับ จึงทำให้ได้ภาพ 3 ขนาด ซึ่งภาพทั้ง 3 ขนาดนี้จะใช้ในการหาวัตถุที่มีขนาดแตกต่างกัน

โดยที่ภาพขนาดเล็กสุดใช้ตรวจจับวัตถุขนาดใหญ่ ภาพขนาดกลางใช้ตรวจจับวัตถุขนาดกลาง และภาพขนาดใหญ่สุดใช้ตรวจจับวัตถุขนาดเล็ก ซึ่งเมื่อผ่านกระบวนการทำงานของทั้ง 3 ชั้นนี้แล้ว จะทำให้สามารถตรวจจับวัตถุที่ขนาดหลากหลายได้ แสดงตัวอย่างดังรูปที่ 2.26



รูปที่ 2.26 โครงสร้างทางสถาปัตยกรรมของ YOLOv3 (Sanjay Dulepet และคณะ, 2020)

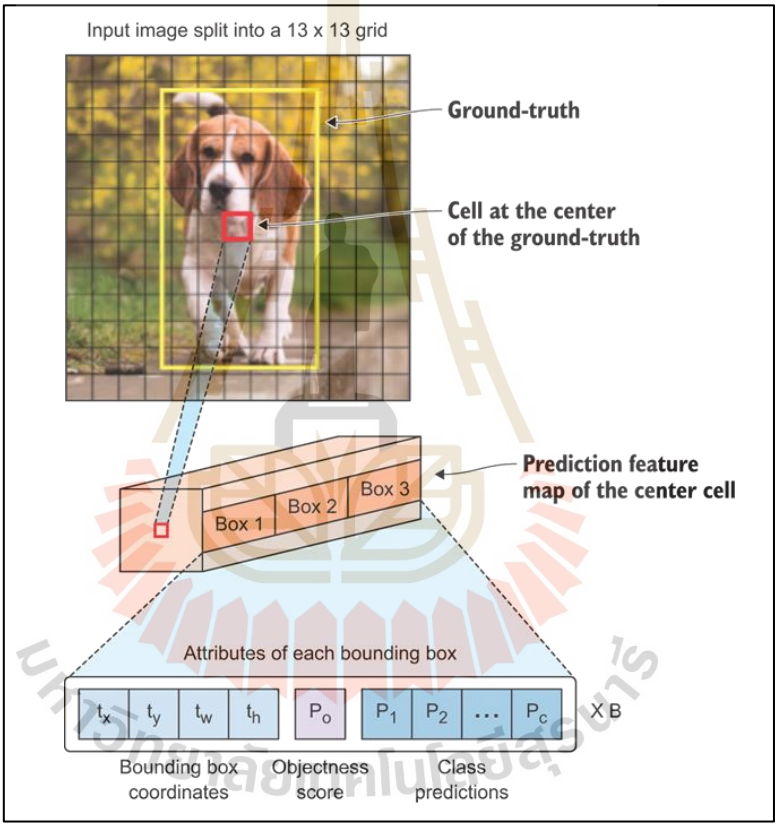
2.5.2.2 หลักการทำงานของ YOLOv3

YOLOv3 สามารถรองรับรูปภาพได้หลายขนาดโดยรูปภาพที่เป็นอินพุตเข้ามานั้นจะมาในรูปของชุดข้อมูลภาพ (Batch of images) ซึ่งมีขนาด (n, 416, 416, 3) ซึ่งประกอบไปด้วย n จำนวนรูปภาพ ความกว้างและความสูงของรูปภาพในหน่วยพิกเซล ซึ่งเป็นได้หลายขนาดที่สามารถหารด้วย 32 ลงตัว และจำนวนชั้นของสี RGB ตามลำดับ

หลังจากนั้นภาพจะถูกนำมาแยกลักษณะเด่นด้วยการทำ Convolutional ที่ขนาด Stride 32 จะทำให้ได้ภาพขนาด 13×13 พิกเซลในกรณีที่ภาพเริ่มต้นมีขนาด 416×416 เพื่อนำเข้าสู่ชั้น Detection ต่อไป

ในชั้น Detection นั้นจะมี Detection Kernel ซึ่งเป็นเมทริกซ์ที่ใช้ในการตรวจจับวัตถุในภาพ เพื่อนำไปใช้ในการสร้าง Feature map ซึ่งในแต่ละเซลล์ของภาพนั้นจะมีการทำนายขอบเขตของวัตถุ (Bounding Box) ทั้งหมด 3 อันและทำนายหาจุดศูนย์กลางของวัตถุจากค่าความเป็นกลาง (Objectless Score) โดยอิงจากขอบเขตที่ถูกต้อง (Ground truth bounding box) ซึ่งทำไว้ก่อนที่จะเริ่มเรียนรู้โมเดล โดยที่ถ้าเซลล์อยู่ตรงกลางวัตถุมากจะทำให้ค่าที่ได้ใกล้เคียง 1 และเซลล์โดยรอบจุดศูนย์กลางจะมีคะแนนลดลงไป ซึ่ง kernel มีขนาด $1\times1\times(B(5+c))$ โดยที่ B

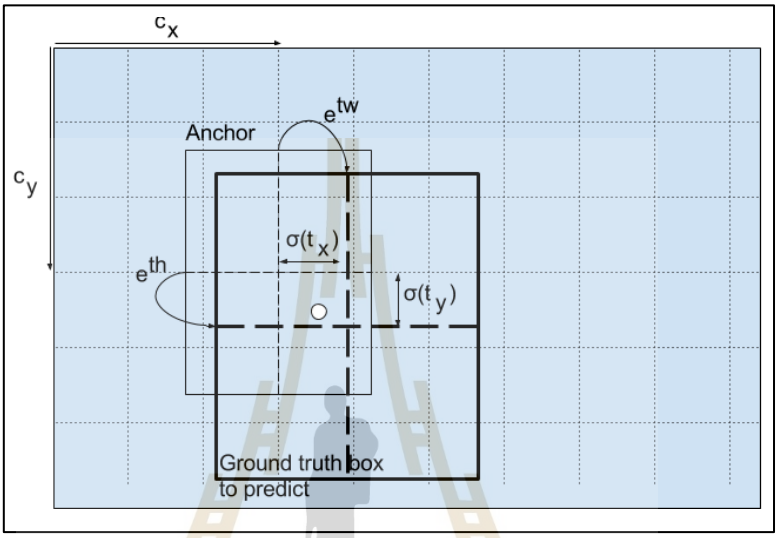
คือ จำนวน Bounding box ในแต่ละเซลล์ ซึ่งมีค่าเป็น 3 และ c คือ จำนวนคลาสทั้งหมดในโมเดล ซึ่งปกติแล้วการเทรนโมเดล YOLOv3 จะใช้ข้อมูลของ COCO ซึ่งมีทั้งหมด 80 คลาส แต่หากในกรณีที่ทำการสร้างโมเดลเองจำนวนคลาสอาจจะมากหรือน้อยกว่านี้ได้ขึ้นอยู่กับการนำไปใช้ของผู้พัฒนาซึ่งหากอิงตามข้อมูลของ COCO จะทำให้ Kernel นี้มีความลึก 255 ทำให้ Kernel มีขนาด $1 \times 1 \times 255$ ดังนั้น Feature map ที่สร้างจาก Detection kernel ทั้ง 3 ชั้น และใช้ข้อมูลของ COCO จะมีรูปร่างเป็น (13, 13, 255), (26, 26, 255) และ (52, 52, 255) ตามลำดับ สามารถอธิบายได้ดังรูปที่ 2.27



รูปที่ 2.27 Detection kernel ของแต่ละเซลล์ในภาพ (Robertson Davies, 2019)

การทำนายหาขนาดความกว้างและความสูงของขอบเขตวัตถุจำเป็นต้องอาศัย Anchor Box ช่วยในการทำนายซึ่ง Anchor box นี้จะใช้ 3 อันในรูปชั้น Detection แต่ละขนาด ซึ่งถ้ารวมทั้งหมด 3 ขนาด จะใช้ Anchor box 9 อัน ซึ่งนั่นเป็นเหตุผลทำให้ในแต่ละเซลล์ของรูปสามารถหา Bounding Box ได้ 3 อันในแต่ละเซลล์ การหา Anchor Box ของแต่ละรูปนั้นใช้ K-mean clustering ในการคำนวณหา เมื่อได้ Anchor Box แล้วสามารถนำไปคำนวณหา Bounding

Box ได้โดยอาศัยการคำนวณค่าความต่างระหว่าง Anchor Box และ Predicted Bounding Box โดยแสดงดังรูปที่ 2.24 การคำนวณโดยอาศัยค่าความต่างหรือ Offset มีข้อดีคือ ช่วยลดการเกิด Unstable gradient ระหว่างที่โมเดลกำลังเรียนรู้ ในส่วนของการคำนวณจุดศูนย์กลางของขอบเขตวัตถุนั้นจะใช้ Sigmoid function ในการคำนวณ ซึ่งทำให้ค่าที่ได้นั้นอยู่ระหว่าง 0-1



รูปที่ 2.28 ลักษณะการหา Bounding Box จาก Anchor Box ด้วยการหาค่า Offset ในรูปของ Sigmoid function (Rokas Balsys, 2019)

จากรูปที่ 2.28 สามารถอธิบายประกอบสมการการคำนวณขนาดของ Bounding Box ได้ดังสมการต่อไปนี้

$$b_x = \sigma(t_x) + c_x \tag{2.12}$$

$$b_y = \sigma(t_y) + c_y \tag{2.13}$$

$$b_w = p_w \times e^{tw} \tag{2.14}$$

$$b_h = p_h \times e^{th} \tag{2.15}$$

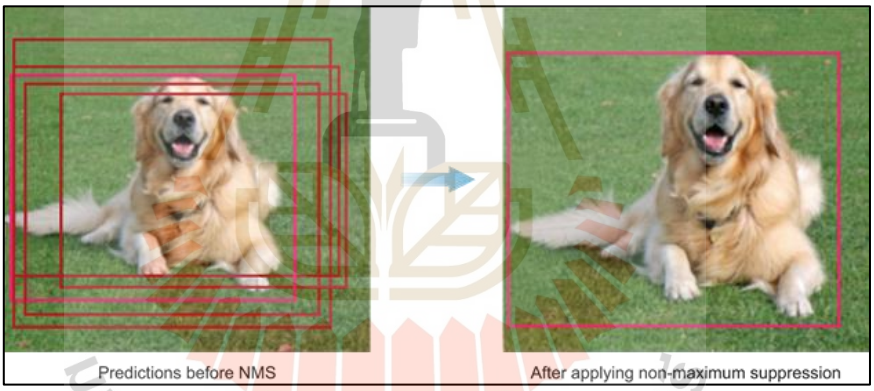
เมื่อ b_x และ b_y คือ จุดศูนย์กลางของ Bounding Box

b_w และ b_h คือ ความกว้างและความสูงของ Bounding Box ตามลำดับ

c_x และ c_y คือ ตำแหน่งอ้างอิงมุมบนซ้ายของ Grid

t_x, t_y, t_w และ t_h คือ ผลลัพท์ของเครือข่าย Neural network

การหาตำแหน่งจุดศูนย์กลางของ Bounding Box นั้นจะนำค่า t_x และ t_y ผ่าน Sigmoid function เพื่อให้ได้ค่าระหว่าง 0-1 นำไปบวกกับตำแหน่ง c_x และ c_y ตามสมการที่ 2.12 และสมการที่ 2.13 ดังที่กล่าวไว้ข้างต้น สำหรับการหาขนาดของ Bounding Box นั้นจะใช้ log-space transform กับผลลัพท์ของเครือข่ายและนำไปคูณด้วย ขนาดของ Anchor Box ตามสมการที่ 2.14 และสมการที่ 2.15 การคำนวณหา Bounding Box ทั้งหมดตั้งแต่ต้นจนจบกระบวณการนี้จะทำให้ได้ Bounding Box ทั้งหมดจำนวน 10,647 ซึ่งจะถูกทำให้เหลืออันที่ถูกต้องมากที่สุดโดยใช้เทคนิค Non-maximum suppression (NMS) แสดงตัวอย่างดังรูปที่ 2.29



รูปที่ 2.29 ตัวอย่างรูปก่อนและหลังการทำ Non-maximum suppression (Mohamed Elgendy, 2020)

จากข้อมูลที่กล่าวมาข้างต้นนั้น ทำให้ตัดสินใจได้ว่าในงานวิจัยฉบับนี้จึงเลือกใช้อัลกอริทึมของ YOLOv3 เนื่องจากมีความแม่นยำและความรวดเร็วมากกว่าอัลกอริทึมอื่น นอกจากนั้น YOLOv3 ยังได้ทำการพัฒนาอัลกอริทึม Darknet ROS เพื่อให้สามารถเชื่อมต่อการทำงานกับ ROS ได้ซึ่งง่ายต่อการวิจัยและพัฒนา

2.5.3 การประเมินผลโมเดลที่ผ่านการเรียนรู้ (Evaluate model)

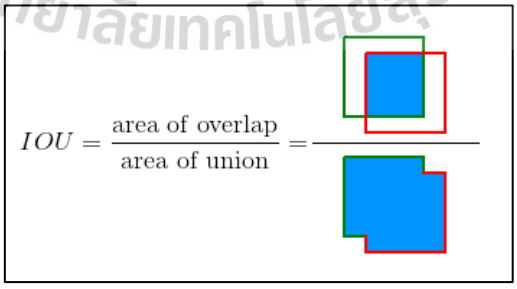
หลังจากที่ผ่านการเรียนรู้ด้วยตัวอย่างที่มากเพียงพอ เครื่องมือหรือโมเดลที่เรียนรู้แล้วนั้นจะสามารถนำไปใช้ในการประมวลผลได้ แต่ก่อนการนำไปใช้งานจริงนั้นต้องทำการประเมินประสิทธิภาพการทำงานของโมเดล เพื่อให้ทราบว่าโมเดลนั้นมีประสิทธิภาพที่ดีพอ

ต่อการนำไปใช้งาน สำหรับการบ่งบอกประสิทธิภาพของโมเดลมีตัวแปรที่เกี่ยวข้องทั้งหมด 4 ตัวแปร ได้แก่ T_P คือ จำนวนข้อมูลที่ทำนายถูกและค่าจริงนั้นถูก F_N คือจำนวนข้อมูลที่ทำนายผิดแต่ค่าจริงนั้นถูก F_P คือ จำนวนข้อมูลที่ทำนายถูกแต่ค่าจริงนั้นผิด และ T_N คือ จำนวนข้อมูลที่ทำนายผิดและค่าจริงนั้นผิด โดยสามารถอธิบายได้จาก Confusion Matrix ดังรูปที่ 2.30 จากนั้นจึงนำตัวแปรมาคำนวนค่าต่าง ๆ เพื่อวัดประสิทธิภาพของโมเดล (Mr.P L, 2018 และ Keng Surapong, 2019)

	Predicted class	
True class	Positive	Negative
Positive	*TP*	*FN*
Negative	*FP*	*TN*

รูปที่ 2.30 Confusion Matrix สำหรับการประเมินผลของโมเดล (Atish P. Sinha และคณะ, 2004)

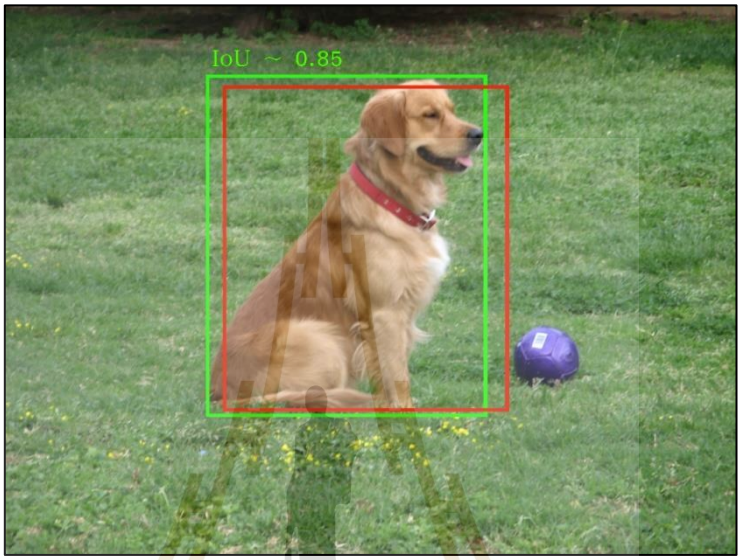
สำหรับการวัดประสิทธิภาพของโมเดลที่ใช้ในการตรวจจับวัตถุในภาพนั้นการได้มาซึ่งตัวแปรทั้ง 4 อันได้แก่ T_P, T_N, F_P และ F_N นั้นต้องใช้ทฤษฎีเรื่อง Intersection over Union (IOU) ช่วย โดยที่ IOU นั้นจะเป็นการคำนวณพื้นที่ทับซ้อนกันระหว่าง 2 Bounding box ซึ่งได้แก่ Ground truth bounding box คือ Bounding box เฉลยที่ถูกต้อง และ Predicted bounding box คือ Bounding box ที่ได้จากการทำนายของแบบจำลอง ซึ่ง IOU นั้นสามารถคำนวณได้จากการนำพื้นที่ในส่วนที่ซ้ำกันหารด้วยพื้นที่ทั้งหมด โดยแสดงดังรูปที่ 2.31



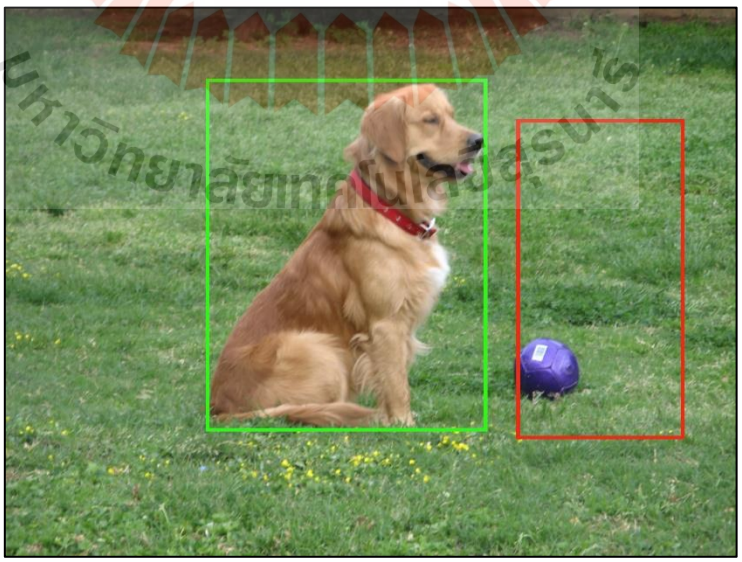
รูปที่ 2.31 การคำนวณหาค่า IOU เพื่อใช้ในการประเมินโมเดล (Supervisely, 2021)

เมื่อคำนวณค่า IOU ได้แล้วจะนำมาพิจารณาเพื่อหาว่าผลลัพธ์ที่ได้ในรูปนั้นเป็น T_P, F_N หรือ F_P ซึ่งทำได้โดยการกำหนดค่าที่ยอมรับได้ โดยปกติตามมาตรฐานหาก

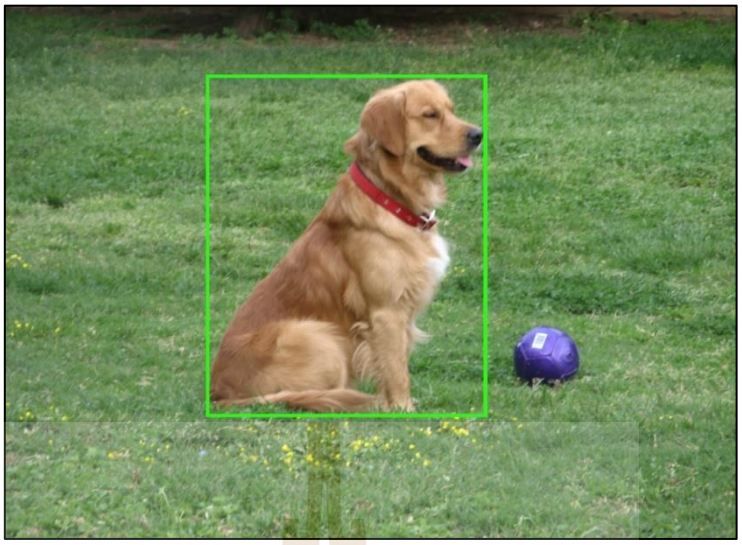
ค่า IOU มากกว่า 0.5 จะถือว่าเป็น T_P ถ้าน้อยกว่ามาตรฐานจะเป็น F_P และไม่สามารถตรวจจับวัตถุในภาพได้จะเป็น F_N สำหรับ T_N ในการประเมินโมเดลสำหรับการตรวจจับวัตุนั้นจะไม่มีการนำมาคิดเนื่องจากไม่สามารถทำการหาข้อมูลที่ทำนายผิดได้ ตัวอย่างการตรวจจับในรูปแบบต่าง ๆ แสดงดังรูปที่ 2.32-2.34



รูปที่ 2.32 ประเมินโมเดลโดยค่า IOU มากกว่า 0.5 ผลลัพธ์แบบ True Positive (Supervisely, 2021)



รูปที่ 2.33 ประเมินโมเดลโดยค่า IOU น้อยกว่า 0.5 ผลลัพธ์แบบ False Positive (Supervisely, 2021)



รูปที่ 2.34 โมเดลไม่สามารถตรวจจับวัตถุได้ ผลลัพธ์แบบ False Negative (Supervisely, 2021)

2.5.3.1 Precision

Precision คือ ค่าความแม่นยำ เกิดจากการนำค่าของข้อมูลที่ทำนายแล้วถูกต้องเมื่อเทียบกับเฉลย (t_p) เทียบกับค่าของข้อมูลที่อยู่ในเฉลยแต่ไม่มีในการทำนาย (f_p) โดยการคำนวนค่าความแม่นยำสามารถคำนวนได้จากสมการที่ 2.16 ดังนี้ (K. Satangmongkol, 2019)

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{2.16}$$

2.5.3.2 Recall

Recall คือ ค่าความน่าจะเป็นที่โมเดลนั้นทำนายถูกต้อง เกิดจากการนำค่าของข้อมูลที่ทำนายแล้วถูกต้องเมื่อเทียบกับเฉลย (t_p) เทียบกับค่าของข้อมูลที่ทำนายแล้วไม่ถูกต้องเมื่อเทียบกับเฉลย (f_n) โดยการคำนวนค่าความถูกต้องสามารถคำนวนได้จากสมการที่ 2.17 ดังนี้

$$\text{Pecall} = \frac{t_p}{t_p + f_n} \tag{2.17}$$

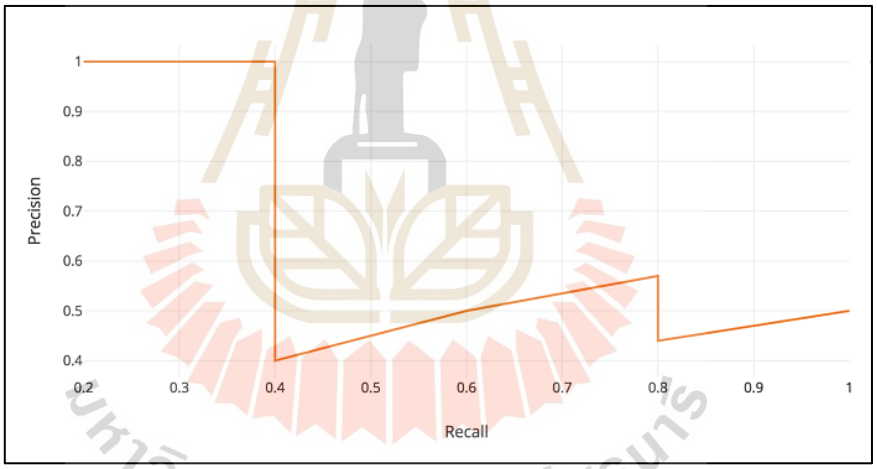
2.5.3.3 F1 Score

F1 Score คือ ค่าเฉลี่ยแบบ harmonic mean ของ Precision และ Recall เพื่อเป็นเมทริกซ์เดี่ยวที่สามารถวัดความสามารถของโมเดลได้ คำนวนได้จากสมการที่ 2.18 ดังนี้

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{2.18}$$

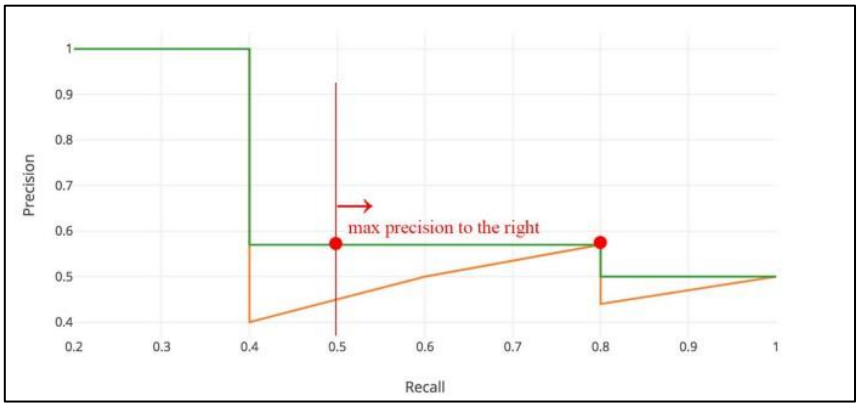
2.5.3.4 Average Precision (AP)

Average Precision เป็นค่าที่นิยมใช้ในการเปรียบเทียบความแม่นยำของแต่ละโมเดล AP เป็นค่าเฉลี่ยของความแม่นยำที่เกิดจากการคำนวณพื้นที่ใต้กราฟระหว่างค่า Precision และค่า Recall ซึ่งกราฟที่ได้จะมีรูปแบบซิกแซก แสดงตัวอย่างสมมติดังรูปที่ 2.35



รูปที่ 2.35 ตัวอย่างกราฟระหว่าง Precision และ Recall (Jonathan Hui, 2018)

ค่า Precision และ Recall จะอยู่ระหว่าง 0-1 ดังนั้นค่า AP จึงอยู่ในช่วง 0-1 เช่นกัน การหาค่า AP นั้นควรจะต้องทำให้กราฟที่ได้นั้นมความเรียบมากขึ้นโดยการใช้ค่า Precision สูงสุดของแต่ละช่วงแทนทำให้กราฟจากเส้นสีส้มเป็นสีเขียว และการแทนที่ด้วยค่า Precision สูงสุดในแต่ละค่า Recall นี้สามารเขียนในรูปสมการได้ดังสมการที่ 2.36



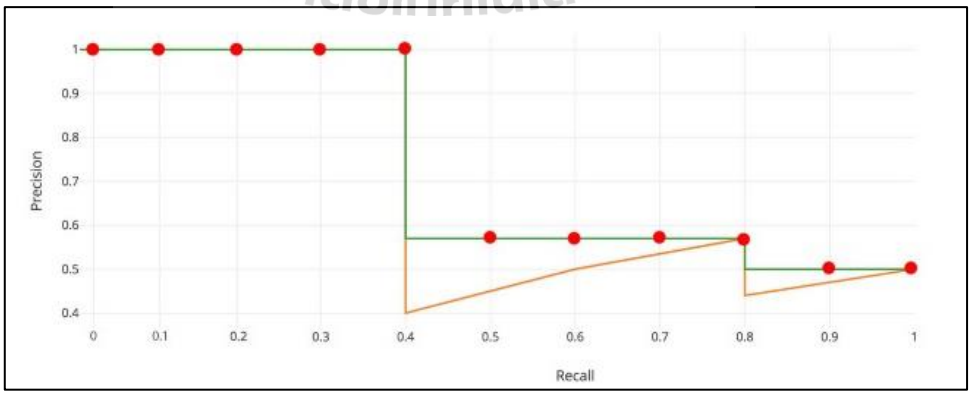
รูปที่ 2.36 ตัวอย่างกราฟระหว่าง Precision และ Recall หลังจากการทำ Max precision (Jonathan Hui, 2018)

$$P_{interp}(r) = \max p(\tilde{r}) \quad ; r \geq \tilde{r} \tag{2.19}$$

เมื่อ $p(\tilde{r})$ คือ ค่า Precision ที่ค่า Recall

จากนั้นจะทำการหาค่าเฉลี่ยของค่า Precision ที่ค่า Recall ต่าง ๆ โดยแบ่งเป็นทั้งหมด 11 ค่า ซึ่งสามารคำนวณได้ดังสมการที่ 2.20 และแสดงตัวอย่างดังรูปที่ 2.37

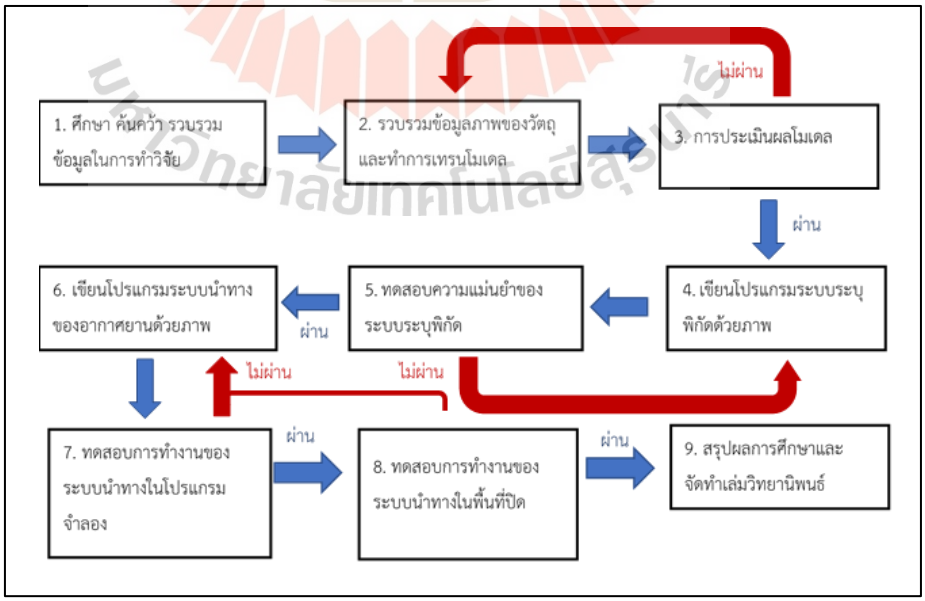
$$AP = \sum_{r \in \{0, 0.1, \ldots, 1.0\}} P_{interp}(r) \tag{2.20}$$



รูปที่ 2.37 ตัวอย่างกราฟระหว่าง Precision และ Recall แบ่งเป็น 11 จุด (Jonathan Hui, 2018)

บทที่ 3
วิธีการดำเนินการวิจัย

งานวิจัยฉบับนี้นำเสนอระบบระบุพิกัดและควบคุมการเคลื่อนที่ของ Parrot Bebop 2 ภายในอาคารโดยใช้วิธีการระบุพิกัดด้วยกล้องถ่ายภาพสามมิติ (Stereo camera) ซึ่งการดำเนินงานวิจัยเริ่มจากการศึกษาข้อมูลที่เกี่ยวข้องกับงานวิจัย ซึ่งได้แก่ การทำงานของกล้องถ่ายภาพสามมิติ การเชื่อมต่อควบคุมหุ่นยนต์ด้วยระบบปฏิบัติการหุ่นยนต์ (Robot Operating Systems: ROS) วิธีการรวบรวมข้อมูลภาพของวัตถุเพื่อทำการเทรนโมเดล การทำการตรวจจับวัตถุ (Object detection) เป็นต้น หลังจากนั้นจึงเริ่มทำการเก็บรวบรวมข้อมูลภาพของวัตถุที่เราต้องการซึ่งในที่นี้คือ Parrot Bebop 2 เพื่อใช้ในการเทรนโมเดลทำระบบตรวจจับภาพด้วยเทคนิค Deep learning โดยใช้โครงสร้างอัลกอริทึมของ YOLOv3 ในการตรวจจับวัตถุเป้าหมาย จากนั้นจะทำการเขียนโปรแกรมระบุพิกัดของวัตถุด้วยภาษา C++ ทำการทดสอบความแม่นยำของระบบระบุพิกัดโดยใช้ Motion capture ในการเทียบ เมื่อระบบมีความแม่นยำที่รับได้แล้วจึงทำการเขียนโปรแกรมการควบคุมการเคลื่อนที่ของโดรน ทำการทดสอบระบบพร้อมกับปรับปรุงแก้ไข สรุปผลและจัดทำวิทยานิพนธ์ฉบับไป ซึ่งแผนภาพขั้นตอนการดำเนินงานแสดงดังรูปที่ 3.1 ดังนี้



รูปที่ 3.1 แผนภาพขั้นตอนวิธีการดำเนินงานวิจัย

3.1 การสร้างโมเดลสำหรับตรวจจับวัตถุ

3.1.1 การรวบรวมและจัดเตรียมชุดข้อมูลภาพ

ขั้นตอนการเตรียมข้อมูลเพื่อทำการเทรนนั้นเป็นขั้นตอนที่สำคัญมากขั้นตอนหนึ่ง เนื่องจากข้อมูลนั้นส่งผลสำคัญถึงความแม่นยำของโมเดลที่จะนำไปใช้ โดยเฉพาะกับงานที่โมเดลนั้นค่อนข้างเฉพาะ การเตรียมข้อมูลจะยิ่งยุ่งยากมากเพราะนอกจากการเตรียมรูปภาพจำนวนมากแล้ว รูปภาพที่นำมายังต้องถูกทำการคัดกรองและปรับแต่งภาพ ซึ่งขั้นตอนการปรับแต่งภาพนี้ส่งผลต่อความแม่นยำของโมเดลที่จะนำไปใช้ ในปัจจุบันขั้นตอนการเตรียมข้อมูลและการเทรนโมเดลนี้สามารถทำได้ในแพลตฟอร์มออนไลน์ต่าง ๆ ซึ่งจะช่วยลดการทำงานของนักวิจัยลงไปมาก โดยในงานวิจัยนี้จะใช้ Supervise.ly ซึ่งเป็นแพลตฟอร์มออนไลน์ช่วยในการเตรียมเซ็ทข้อมูลและเทรนโมเดล ขั้นตอนการเตรียมข้อมูลและการเทรนโมเดล มีดังต่อไปนี้

1. รวบรวมชุดข้อมูลภาพ ในงานวัยฉบับนี้ทำการรวบรวมภาพโดยการบันทึกวิดีโอด้วยกล้อง ZED ในขณะที่โดรนมีการเคลื่อนที่ด้วยการบังคับจากนักบิน การเก็บชุดข้อมูลภาพในครั้งแรก ณ อาคารเครื่องมือ 5 (F5) ดังตัวอย่างรูปที่ 3.2 และครั้งที่สองเก็บข้อมูลเพิ่ม ณ อาคารสุรเริงไชย ห้องทดลอง Motion capture ดังตัวอย่างรูปที่ 3.3

รูปที่ 3.2 ตัวอย่างข้อมูลภาพที่บันทึก ณ อาคารเครื่องมือ 5 (F5)

รูปที่ 3.3 ตัวอย่างข้อมูลภาพที่บันทึก ณ อาคารสุรเริงไชย ห้องทดลอง Motion capture

2. อัพโหลดวิดีโอไปยังแพลตฟอร์ม Supervise.ly และทำการแบ่งรูปภาพจากวิดีโอทุก ๆ 30 เฟรม เนื่องจากความเร็ววิดีโอคือ 60 เฟรมต่อวินาที ทำให้ใน 1 วินาที สามารถเก็บข้อมูลได้ 2 ดังนั้นจึงได้ข้อมูลภาพทั้งหมดจำนวน 1,493 ภาพ เนื่องจากเป็นการเรียนรู้แบบ Supervised ดังนั้นผู้พัฒนาจึงทำการให้ข้อมูลโดยการล้อมรอบวัตถุและระบุชื่อของวัตถุที่ต้องการจะให้เรียนรู้ในข้อมูลภาพทั้งหมด ตัวอย่างดังรูปที่ 3.4

รูปที่ 3.4 ตัวอย่างข้อมูลภาพหลังจากการระบุตำแหน่งและชื่อของวัตถุที่ต้องการให้เรียนรู้

3. ทำการปรับแต่งรูปภาพเพื่อให้รูปภาพมีความหลากหลายโมเดลที่ได้จะสามารถเรียนรู้ได้ดีขึ้น โดยการปรับแต่งนี้จะทำในแพลตฟอร์ม Supervise.ly ในขั้นตอน DTL ซึ่งเป็นขั้นตอนการเตรียมข้อมูลสำหรับการเทรนนิ่งโดยการใช้ภาษาพิเศษที่ทางผู้พัฒนาได้คิดค้นขึ้นมาสามารถทำให้การเตรียมข้อมูลทำได้เร็วขึ้น การปรับแต่งรูปภาพในงานวิจัยนี้จะมีการปรับความเข้มและความสว่างของภาพ (Contrast) ตัวอย่างดังรูปที่ 3.5 การกลับภาพในแนวตั้ง (Vertical) และแนวนอน (Horizontal) ตัวอย่างดังรูปที่ 3.6 การลดความคมชัดของภาพ (Blur) ตัวอย่างดังรูปที่ 3.7 และการแบ่งข้อมูลเป็น 2 ส่วน คือ ส่วนสำหรับการเทรนโมเดล (Training set) เป็นร้อยละ 80 ของจำนวนข้อมูล และส่วนสำหรับการเทียบความผิดพลาด (Validation set) เป็นร้อยละ 20 ของจำนวนข้อมูล โดยหลังจากผ่านกระบวนการปรับแต่งรูปภาพตามขั้นตอนทั้งหมดข้างต้นแล้วทำให้มีจำนวนภาพสำหรับการเทรนทั้งสิ้นจำนวน 35,832 ภาพ

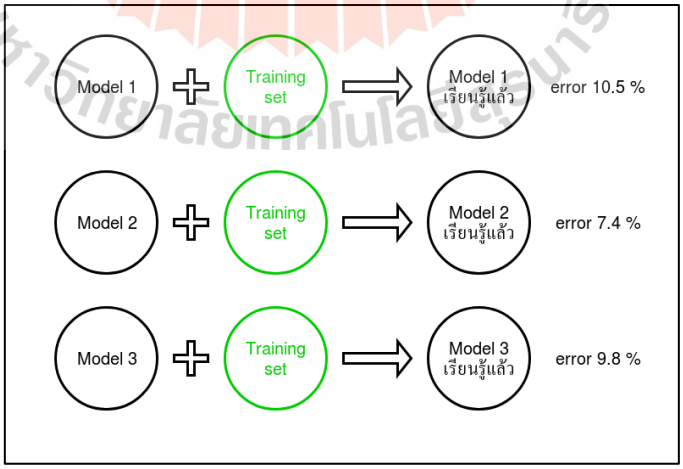
รูปที่ 3.5 ตัวอย่างภาพที่ผ่านการเพิ่มและลดความสว่างของรูปภาพ

รูปที่ 3.6 ตัวอย่างภาพที่ผ่านการหมุนภาพในแนวนอน (Horizontal) และแนวตั้ง (Vertical)

รูปที่ 3.7 ตัวอย่างภาพที่ผ่านการลดความคมชัด (Blur)

3.1.2 การเทรนโมเดล

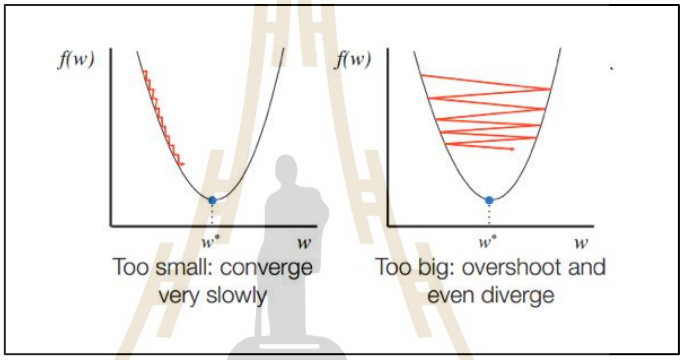
การเทรนโมเดลสำหรับการตรวจจับวัตถุจะทำผ่านโครงสร้างของ YOLOv3 ในแพลตฟอร์ม Supervise.ly โดยใช้ Training set ในการเทรนโมเดลด้วยขั้นตอน Gradient descent ให้โมเดลทำนายคำตอบของข้อมูลใน Training set เทียบคำตอบจากโมเดลกับคำตอบจริงเพื่อวัดค่าความผิดพลาดแล้วทำการปรับค่าตัวแปรของโมเดล เพื่อให้ค่าความผิดพลาดรอบถัดไปลดลงจนได้โมเดลและค่าความผิดพลาดสุดท้ายของโมเดล ตัวอย่างการเทรนโมเดลด้วยขั้นตอน Gradient Descent แสดงในรูปที่ 3.8



รูปที่ 3.8 การเทรนโมเดลด้วยขั้นตอน Gradient descent (Suphan Fayong, 2018)

ส่วนที่สำคัญที่สุดสำหรับการเทรนโมเดลคือ ไฮเปอร์พารามิเตอร์ต่าง ๆ ที่ใช้สำหรับการตั้งค่าการเทรน ซึ่งมีผลต่อประสิทธิภาพของโมเดลที่สร้างขึ้น โดยการปรับค่าไฮเปอร์พารามิเตอร์ จะต้องคำนึงถึงความสามารถของ GPU ด้วย ซึ่งในการวิจัยนี้ใช้ GPU รุ่น GTX 1050 2 GB และไฮเปอร์พารามิเตอร์ที่สำคัญมีดังต่อไปนี้

1. อัตราการเรียนรู้ (Learning rate: lr) เนื่องจากการเทรนโมเดลใช้วิธี Gradient decent อัตราการเรียนรู้จึงเปรียบเสมือนความเร็วในการเปลี่ยนแปลงในแต่ละขั้นของการคำนวณ ถ้ามีค่าน้อยจะใช้ระยะเวลาในการเทรนมาก ถ้าค่ามากจะใช้เวลาเร็วแต่เสี่ยงที่โมเดลจะไม่ลู่เข้า แสดงตัวอย่างดังรูปที่ 3.9



รูปที่ 3.9 ลักษณะการทำ Gradient decent ที่มีอัตราการเรียนรู้น้อยเกินไปและมากเกินไป (Subir Varma และคณะ, 2018)

2. จำนวนรอบในการเรียนรู้ (Epochs) ประสิทธิภาพของโมเดลขึ้นอยู่กับจำนวนรอบในการเรียนรู้ด้วยยิ่งมีการเรียนรู้หลายรอบทำให้โมเดลมีประสิทธิภาพมากขึ้น แต่การเรียนรู้ที่มีจำนวนรอบมากเกินไปอาจทำให้โมเดลเกิดการ Over fitting ได้

3. ขนาดของกลุ่มข้อมูล (Batch size) การปรับค่าให้ชุดข้อมูลมีขนาดเล็กจะทำให้การเรียนรู้ช้าลง และใช้ความจุของ GPU น้อยลงด้วย ซึ่งจากงานวิจัยของ Dominic Mastersและคณะ 2018 ได้ทำการวิจัยและทดลองได้ผลสรุปว่าการปรับค่าไม่เกิน 32 จะทำให้ได้ประสิทธิภาพที่ดีกว่า นอกจากนั้นการปรับขนาดต้องคำนึงถึงความจุของ GPU ด้วย เพื่อไม่ให้เกิดปัญหาด้านพื้นที่ความจุไม่พอ

4. ขนาดของรูปภาพ (Input size) ต้องทำการกำหนดความกว้างและความสูงของรูปภาพในหน่วยพิกเซล ขนาดของรูปภาพสามารถเป็นได้ตั้งแต่ 256×256, 416×416 หรือมากกว่า

นั้นแต่ต้องเป็นเลขที่หารด้วย 32 ลงตัว ขนาดรูปภาพที่ใหญ่จะทำให้เทรนช้าลงและใช้พื้นที่ความจุ GPU มาก

จากการสืบค้นข้อมูลในการปรับตั้งค่าไฮเปอร์พารามิเตอร์ที่เกี่ยวข้องกับการเทรนโมเดล ทำให้ผู้วิจัยได้ทำการทดลองปรับตั้งค่าที่เหมาะสม โดยการปรับค่าไฮเปอร์พารามิเตอร์แสดงดังตารางที่ 3.1 สำหรับผลการทดลองจากการปรับค่าตามตารางจะอภิปรายในหัวข้อที่ 4.1.1

ตารางที่ 3.1 การทดลองปรับค่าไฮเปอร์พารามิเตอร์สำหรับเทรนโมเดล

ที่	อัตราการเรียนรู้ (Learning rate)	จำนวนรอบ (Epochs)	ขนาดของกลุ่มข้อมูล (Batch size)	ขนาดของรูปภาพ (Input size)
1	0.0001 (ค่าตั้งต้น)	10000	8 (ค่าตั้งต้น)	416×416 (ค่าตั้งต้น)
2	0.0001 (ค่าตั้งต้น)	10000	2	416×416 (ค่าตั้งต้น)
3	0.0001 (ค่าตั้งต้น)	10000	1	416×416 (ค่าตั้งต้น)
4	0.0001 (ค่าตั้งต้น)	2000	1	416×416 (ค่าตั้งต้น)
5	0.0001 (ค่าตั้งต้น)	700	1	416×416 (ค่าตั้งต้น)
6	0.0001 (ค่าตั้งต้น)	500	1	416×416 (ค่าตั้งต้น)
7	0.0001 (ค่าตั้งต้น)	100	1	416×416 (ค่าตั้งต้น)
8	0.0001 (ค่าตั้งต้น)	100	1	608×608
9	0.0001 (ค่าตั้งต้น)	150	1	416×416 (ค่าตั้งต้น)
10	0.0001 (ค่าตั้งต้น)	200	1	416×416 (ค่าตั้งต้น)
11	0.0001 (ค่าตั้งต้น)	100	1	416×416 (ค่าตั้งต้น)

3.1.3 การเลือกใช้โมเดลและการทดสอบความแม่นยำ

การพิจารณาเลือกใช้โมเดลหลังจากการเทรนเสร็จสิ้นนั้นต้องมีการพิจารณา 2 ครั้งในครั้งแรก พิจารณาจากค่าความผิดพลาดที่เกิดขึ้นจากการเทรนในแต่ละรอบโดยเป็นค่าความผิดพลาดที่ทดสอบเทียบกับชุดข้อมูลสำหรับการเทียบความผิดพลาด (Validation set) ซึ่งชุดทดสอบนี้เป็นชุดทดสอบที่อยู่ในกระบวนการเรียนรู้ของโมเดล การเลือกโมเดลในครั้งแรกนี้ทำโดยการเลือกรอบการคำนวณที่มีค่าความผิดพลาดน้อยที่สุด แสดงตัวอย่างดังรูปที่ 3.10

รูปที่ 3.10 การเลือกโมเดลโดยพิจารณาจากค่าความผิดพลาดที่น้อยที่สุด

การพิจารณาเลือกใช้โมเดลในครั้งที่ 2 จะพิจารณาโดยทำการทดสอบโมเดลที่ได้จากการเลือกครั้งแรก ทดสอบเทียบกับชุดข้อมูลทดสอบ (Test set) ซึ่งเป็นชุดข้อมูลที่ไม่มีใน Training set หรือเป็นชุดข้อมูลที่โมเดลไม่เคยเจอมาก่อน การทดสอบนี้จำเป็นต้องทำเฉลยชุดทดสอบ (Ground truth) ซึ่งสามารถทำโดยการระบุตำแหน่งกรอบของวัตถุ (Bounding Box) บนรูปภาพโดยผู้ทำการทดลอง จากนั้นจึงนำโมเดลที่ผ่านการพิจารณาในครั้งที่ 1 มาทดสอบร่วมกับชุดทดสอบที่มีเฉลยนี้ หลังจากการทดสอบจะได้กรอบของวัตถุซึ่งเป็นเฉลยและกรอบวัตถุที่ได้จากการทำนายจากโมเดล ตัวอย่างรูปที่ 3.11 กรอบสีเขียวคือกรอบที่ได้จากการทำนาย (Predicted bounding box) ส่วนกรอบสีแดงคือกรอบที่ได้จากเฉลย (Ground truth bounding box)

รูปที่ 3.11 ตัวอย่างรูปภาพชุดทดสอบหลังจากการทำนายด้วยโมเดล

หลังจากการทดสอบด้วยชุดทดสอบ (Test set) ชุดข้อมูลนี้จะนำไปคำนวณหาค่า IOU เพื่อพิจารณาว่าผลจากการทำนายเป็นประเภทใดและสร้าง Confusion matrix ขึ้นมา หลังจากนั้นการพิจารณาโมเดลในครั้งที่ 2 นี้ทำโดยการประเมินผลของโมเดลจากค่าความแม่นยำ (Precision) ค่าความถูกต้อง (Recall) และค่าเฉลี่ยความแม่นยำ (Average Precision) ตามทฤษฎีในหัวข้อที่ 2.5.3 โดยในงานวิจัยนี้จะเน้นที่การพิจารณาค่าเฉลี่ยความแม่นยำเป็นหลัก เนื่องจากเป็นค่ามาตรฐานที่นิยมใช้ในการเปรียบเทียบประสิทธิภาพของโมเดล โดยทำการเลือกโมเดลที่มีค่าเฉลี่ยความแม่นยำมากที่สุด ตัวอย่าง Confusion matrix ที่ได้จากการคำนวณ IOU ดังรูปที่ 3.12

```
***************** Result metrics values for 0.5 IoU threshold ******************
****************************** Confusion matrix *******************************
                    bebop2_predicted    False Negatives
      bebop 2              1169                32

 False Positives           733                 0
```

รูปที่ 3.12 ตัวอย่าง Confusion matrix ที่ได้จากการคำนวณค่า IOU

3.2 การพัฒนาระบบระบุพิกัดด้วยภาพ

3.2.1 ออกแบบระบบการทำงาน

การเขียนโปรแกรมระบบระบุพิกัดด้วยภาพ ในงานวิจัยนี้เขียนโปรแกรมด้วยภาษา C++ เพื่อใช้ในการเชื่อมต่อและดึงข้อมูลระหว่างเครื่องมือในการตรวจจับวัตถุ (Darknet ROS) และกล้องถ่ายภาพสามมิติ (Stereo camera) ผ่าน ROS ภาพที่มีการตรวจจับวัตถุบนหน้าจอได้ ดังรูปที่ 3.13

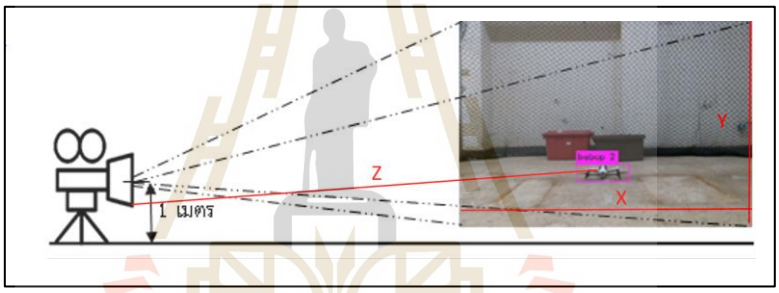
รูปที่ 3.13 ตัวอย่างการตรวจจับวัตถุบนหน้าจอ

หลังจากมีการตรวจจับวัตถุบนหน้าจอได้แล้ว Darknet จะส่งข้อมูลพิกัดบนหน้าจอของเป้าหมายซึ่งเป็นพิกัด 2 มิติในหน่วยพิกเซล เข้ามาในระบบที่ได้ทำการพัฒนาไว้ผ่านหัวข้อของ ROS จากนั้นข้อมูลจะถูกส่งไปเทียบกับตำแหน่งของ point cloud แต่ละจุด ซึ่งได้จากการคำนวณผ่าน SDK ของกล้อง ZED โดย point cloud แต่ละจุดนั้นมีพิกัด 3 มิติในหน่วยเมตรของจุดนั้น ๆ อยู่ ซึ่งการแปลงพิกัดจากหน่วยพิกเซลเป็นหน่วยเมตรทำได้โดยใช้ Point cloud library (PCL) กระบวนการประมวลผลการประมาณค่าตำแหน่งของจุดแบบ 3 มิติ เป็นไปดังทฤษฎีในหัวข้อที่ 2.4 สัญลักษณ์แกน X แทนแกนแนวนอนของภาพ Y แทนแกนแนวตั้งของภาพ และ Z แทนความลึกของภาพ หลังจากได้ตำแหน่งเป็นเมตรแล้วจึงเข้าสู่กระบวนการกรองสัญญาณรบกวนออก (Noise filtration) โดยการนำค่าที่ไม่สามารถอ่านได้หรือค่าที่ผิดพลาดออกจากข้อมูลและเฉลี่ยชุดข้อมูลที่ได้มาทุก ๆ 5 ค่า เพื่อให้ค่าที่ได้เป็นค่าที่มีการแกว่งน้อยลง เพิ่มความแม่นยำในการตรวจจับ การทำงานของระบบระบุพิกัดสามารถอธิบายเป็นแผนภาพการทำงานดังรูปที่ 3.14

รูปที่ 3.14 แผนภาพการทำงานของระบบระบุพิกัด

3.2.2 การประเมินความสามารถของระบบระบุพิกัด

ขั้นตอนการประเมินความสามารถของระบบระบุพิกัด จะใช้อุปกรณ์ระบุพิกัดภายในอาคารที่มีความแม่นยำสูงอย่าง Motion capture เป็นอุปกรณ์ที่ใช้เทียบเพื่อหาค่าความผิดพลาด (Error) ของระบบและทำการปรับปรุงแก้ไขระบบจนกระทั่งได้ค่าความคลาดเคลื่อนที่ยอมรับได้หรือไม่เกิน 10 ซม. โดยค่านี้อ้างอิงจากระบบระบุพิกัดในอาคาร POZYX ซึ่งใช้คลื่นสัญญาณ UWB ในการระบุพิกัดและมีค่าเฉลี่ยความคลาดเคลื่อนไม่เกิน 10 ซม. ดังนั้นผู้วิจัยจึงเห็นสมควรให้ใช้ค่ายอมรับได้ไม่เกิน 10 ซม. จึงจะสามารถถือได้ว่าระบบมีความแม่นยำในเกณฑ์ที่ดี สำหรับการตั้งการทดลองเบื้องต้น เป็นดังรูปที่ 3.15 ระยะใกล้และไกลสุดในการทดสอบคือ 2.0 และ 3.0 เมตร ตามลำดับ และมีความสูงจากพื้นไม่เกิน 1.6 เมตร เนื่องจากถูกกำหนดขอบเขตด้วยขนาดห้องทดลอง ตำแหน่งจุดกำเนิด X, Y และ Z เป็น 0 ที่ตำแหน่งของกล้อง จากนั้นจะทำการประเมินผลโดยการหาค่าความแม่นยำและค่าความแปรผัน

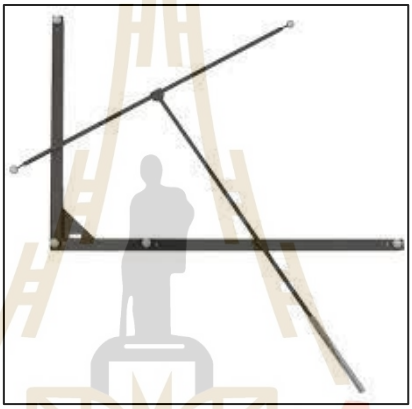


รูปที่ 3.15 การตั้งการทดสอบการระบุพิกัดเบื้องต้น

การทดสอบเก็บข้อมูลจากเครื่องมือระบุพิกัดทั้ง 2 ชนิด ให้ได้ผลที่แม่นยำที่สุด ก่อนการทำงานควรที่จะต้องมีการเทียบวัดอุปกรณ์ (Calibration) ก่อนทุกครั้ง เนื่องจากอุปกรณ์การระบุพิกัดทั้ง 2 ชนิด มีหลักการทำงานที่แตกต่างกัน ดังนั้นกระบวนการเทียบวัดอุปกรณ์ทั้ง 2 ชนิดนี้จึงแตกต่างกันไปด้วย ซึ่งกระบวนการเทียบวัดและจัดเตรียมอุปกรณ์ทั้ง 2 ชนิด ให้พร้อมใช้งาน สามารถอธิบายได้ดังหัวข้อต่อไปนี้

1. กล้อง Motion capture การระบุพิกัดใช้หลักการตามทฤษฎีในหัวข้อที่ 2.3.1 โดยการทดลองนี้ใช้ของยี่ห้อ Qualisys จำนวน 6 เครื่อง มีโปรแกรมควบคุมและตรวจสอบการทำงานชื่อว่า Qualisys Track Manager ซึ่งใช้ในการตั้งค่าและดำเนินการทำงานต่าง ๆ การเทียบวัดอุปกรณ์นั้นทำโดยการนำชุดอุปกรณ์เทียบวัดมาตรฐาน (Calibration kit) มีลักษณะเป็นแท่งเหล็กรูปตัว T และแท่งเหล็กรูปตัว L ที่มีวัสดุสะท้อนแสงติดอยู่ ดังตัวอย่างรูปที่ 3.16 อุปกรณ์ที่มีลักษณะรูปตัว L ใช้สำหรับวางที่ตำแหน่งตั้งต้น (Origin point) โดยด้านสั้นแทนแกน Y ด้านยาว แทน

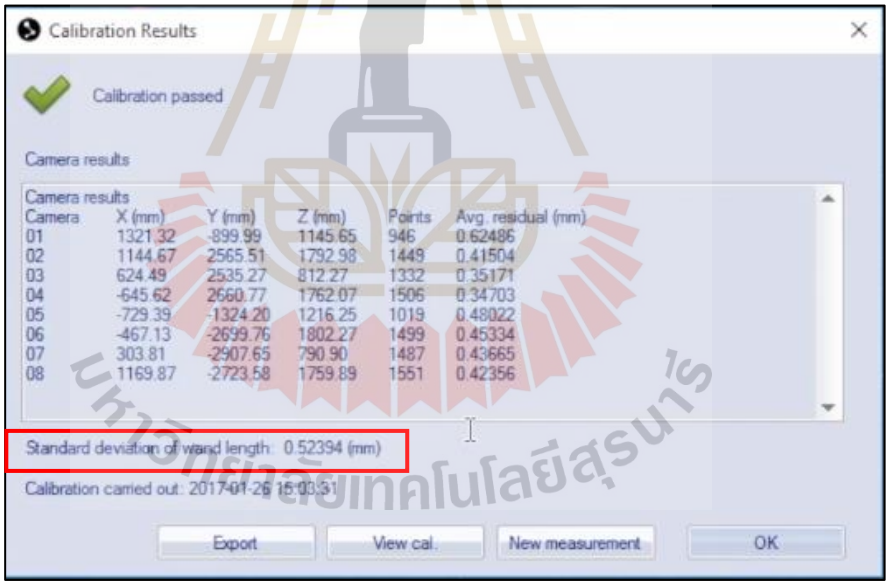
แกน X แสดงดังรูปที่ 3.17 ในการทดลองนี้จะให้ด้านแกนสั้นชี้เข้าหากล้องถ่ายภาพสามมิติ ส่วนอุปกรณ์รูปตัว T ใช้สำหรับการกวัดแกว่งเพื่อเทียบวัดปริมาตรของบริเวณที่ต้องการทดลอง แสดงดังรูปที่ 3.18 การแกว่งอุปกรณ์ต้องไม่เร็วและไม่ช้าเกินไป ระยะเวลาในการทำการเทียบวัดอุปกรณ์ผู้ทำการทดลองสามารถกำหนดได้ โดยเริ่มตั้งแต่ 10 วินาที เป็นต้นไป การเทียบวัดในพื้นที่การทดลองที่มีขนาดใหญ่จะทำให้ค่าความคลาดเคลื่อนนั้นมีมากกว่าพื้นที่ทดลองขนาดเล็ก แต่อย่างไรก็ตามค่าความคลาดเคลื่อนจากการเทียบวัดไม่ควรเกิน 0.6 มม. ซึ่งเป็นค่ามาตรฐานที่ยอมรับได้จากทางผู้ผลิต โดยสามารถตรวจสอบได้จากหน้าต่างที่แสดงในโปรแกรมหลังจากการเทียบวัดดำเนินการแล้วเสร็จ ดังตัวอย่างรูปที่ 3.19



รูปที่ 3.16 ตัวอย่างอุปกรณ์มาตรฐานสำหรับการเทียบวัดระบบ Motion capture ยี่ห้อ Qualisys (Kersting UG และคณะ, 2016)

รูปที่ 3.17 ตัวอย่างการวางอุปกรณ์เทียบวัดรูปตัว L ที่ตำแหน่งจุดกำเนิด

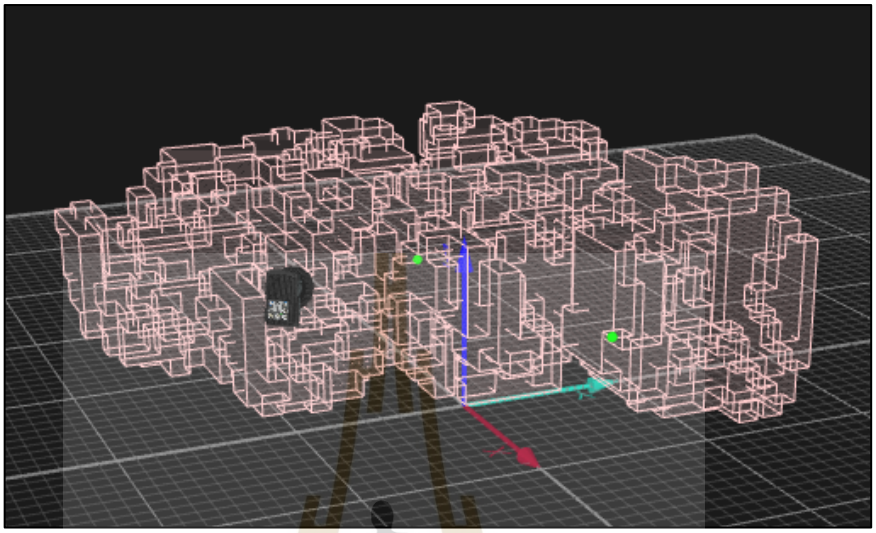
รูปที่ 3.18 ตัวอย่างขั้นตอนการทำการเทียบวัดบริเวณพื้นที่ทดลองด้วยอุปกรณ์เทียบวัดรูปตัว T



รูปที่ 3.19 ตัวอย่างหน้าต่างแสดงผลการเทียบวัดจากโปรแกรม Qualisys Track Manager (Bassett Biomechanics, 2017)

เมื่อดำเนินการเทียบวัดอุปกรณ์ให้ค่าความคลาดเคลื่อนไม่เกินค่ามาตรฐานแล้วเสร็จ สามารถตรวจสอบปริมาตรที่ถูกทำการเทียบวัด (Calibrated volume) ดังรูปที่ 3.20 เพื่อตรวจสอบการกวัดแกว่งอุปกรณ์ให้อยู่ในพื้นที่ทดลองที่ต้องการ นอกจากนั้นผู้ทดลองควรกำจัด

แสงตกกระทบจากสิ่งรบกวนอื่น ๆ ที่แสดงบนจอภาพจากกล้องทั้ง 6 เครื่อง ดังรูปที่ 3.21 เพื่อไม่ให้เกิดการคำนวณที่ผิดพลาดและพร้อมสำหรับการทดลอง



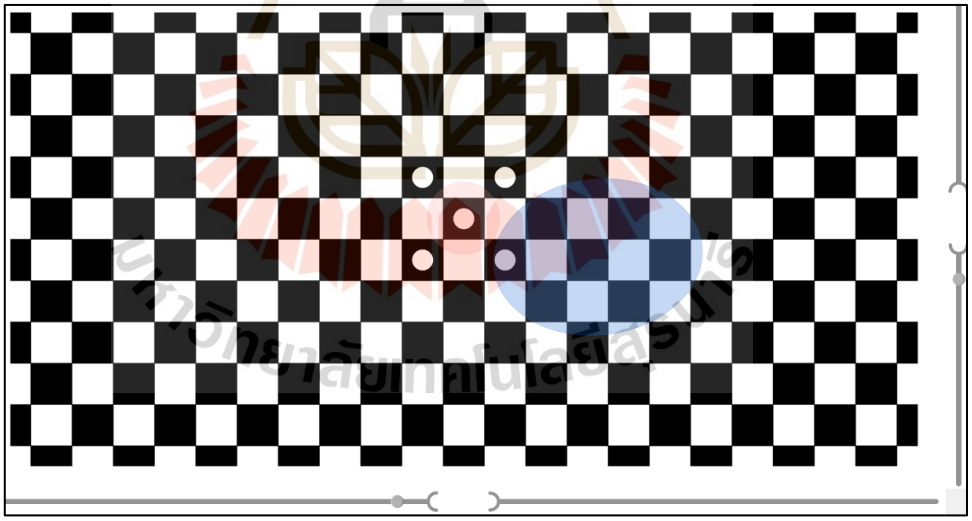
รูปที่ 3.20 ตัวอย่างปริมาตรที่ถูกเทียบวัดแบบ 3 มิติ

รูปที่ 3.21 ตัวอย่างหน้าจอแสดงผลจากกล้องทั้ง 6 เครื่อง

การอ่านพิกัดด้วยระบบที่ผู้วิจัยพัฒนาขึ้นได้กำหนดตำแหน่งจุดกำเนิดให้อยู่ที่ตำแหน่งของกล้องถ่ายภาพ 3 มิติ แต่ทางห้องทดลองได้มีการตั้งค่าตำแหน่งการวางจุดกำเนิดมาเรียบร้อย ซึ่งตำแหน่งจุดกำเนิดที่ผู้วิจัยต้องการกับตำแหน่งจุดกำเนิดที่ได้มีการตั้งค่าไว้ไม่ตรงกัน และไม่สามารถแก้ไขตำแหน่งได้ เนื่องจากอุปกรณ์มีผู้ใช้งานนอกเหนือจากผู้วิจัยอยู่ด้วย ซึ่งการตั้ง

ค่าดังกล่าวจำเป็นที่จะต้องให้ทางผู้ผลิตดำเนินการทำให้ส่งผลกระทบต่อผู้ใช้งานท่านอื่น อีกทั้งโปรแกรมที่ใช้ไม่มีการคำนวณหาค่าพิกัดเทียบกับตำแหน่งจุดกำเนิดโดยตรง ดังนั้นผู้วิจัยจึงทำการทดลองหาพิกัดโดยการหาระยะห่างระหว่างอุปกรณ์สะท้อนแสง (Marker) 2 ชิ้น ชิ้นแรกติดไว้กับกล้องถ่ายภาพ 3 มิติ ซึ่งอยู่กับที่เปรียบเสมือนเป็นตำแหน่งจุดกำเนิด และชิ้นที่ 2 ติดบริเวณด้านหลังของโดรนที่ทำการเคลื่อนที่ จึงทำให้สามารถได้ข้อมูลที่ต้องการเพื่อนำไปเปรียบเทียบกับข้อมูลที่ได้จากกล้องถ่ายภาพ 3 มิติถัดไป

2. กล้องถ่ายภาพ 3 มิติ การเตรียมความพร้อมสำหรับการระบุพิกัดด้วยวิธีนี้มีขั้นตอนน้อยกว่าวิธีใช้กล้อง Motion capture การเทียบวัดอุปกรณ์สามารถทำได้โดยใช้โปรแกรม ZED calibration ซึ่งเป็นโปรแกรมที่ติดตั้งมาพร้อมกับ SDK ของอุปกรณ์เรียบร้อยแล้ว การเทียบวัดในโปรแกรมจะมีลักษณะเป็นตารางหมากรุกเพื่อตรวจสอบการบิดงอของภาพ (Distortion) ผู้วิจัยต้องหันกล้องเข้าหาตารางหมากรุกที่ปรากฏบนหน้าจอคอมพิวเตอร์และทำการเคลื่อนที่เข้าและออก เพื่อให้พื้นที่สีน้ำเงินและสีแดงซ้อนทับกันให้มากที่สุด โดยสีน้ำเงินคือ พื้นที่ที่เกิดจากกล้อง ส่วนสีแดงคือ พื้นที่ที่โปรแกรมกำหนดขึ้น ตัวอย่างดังรูปที่ 3.22 ทำเช่นนั้นจนเสร็จสิ้นกระบวนการโปรแกรมจะทำการปรับค่าเทียบวัดให้กล้องอ่านค่าได้แม่นยำมากขึ้น



รูปที่ 3.22 ตัวอย่างการเทียบวัดกล้องถ่ายภาพ 3 มิติ ZED (StereoLABS)

การประเมินประสิทธิภาพของระบบระบุพิกัดด้วยกล้องถ่ายภาพ 3 มิติ พิจารณาจากค่ามาตรฐาน 2 อย่าง คือ ค่าความถูกต้อง และค่าความผันผวนของจุดศูนย์กลางของตำแหน่งวัตถุที่สนใจ เนื่องจากการระบุพิกัดโดยใช้กล้องถ่ายภาพ 3 มิตินั้นมีความละเอียดใน

หน่วยเซนติเมตร ดังนั้น การเปลี่ยนแปลงเพียงเล็กน้อยของตำแหน่งจุดศูนย์กลางอาจส่งผลถึงค่าพิกัดที่วัดได้จึงต้องมีการประเมินค่าความผันผวนเพื่อเพิ่มความน่าเชื่อถือของงานวิจัย

การทดสอบเพื่อหาค่าความถูกต้องของการระบุพิกัด ผู้วิจัยได้แบ่งการทดลองเป็น 2 กรณี ดังนี้

1) การทดลองโดยให้โดรนไม่เคลื่อนที่ (Static test) โดยผู้วิจัยได้ทำการทดลองโดยการแบ่งพื้นที่ห้องทดลองออกเป็นตาราง 0.5×0.5 ม. ทั้งแกน x และ z ส่วนความสูง y จะเพิ่มขึ้นทุก ๆ 50 ซม. จนถึงความสูง 160 ซม. จากพื้น และให้กล้องทำการอ่านค่าพิกัดและคำนวณหาค่าเฉลี่ยความคลาดเคลื่อน

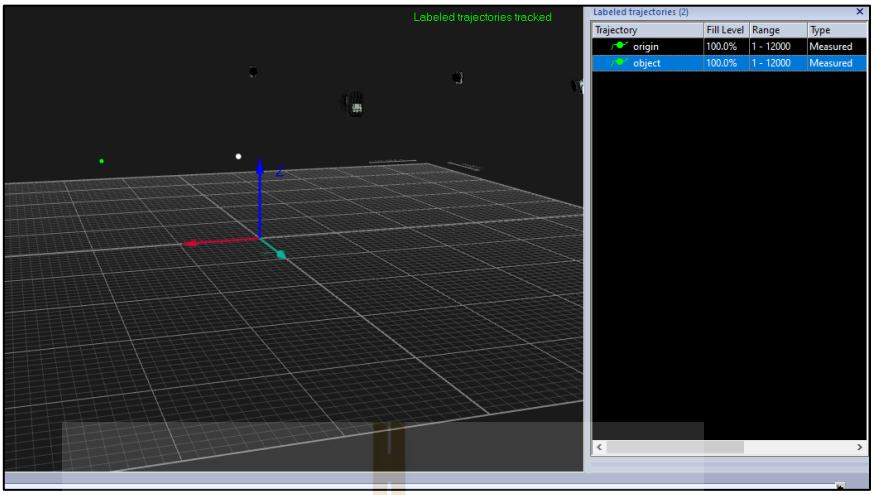
2) การทดลองโดยให้โดรนเคลื่อนที่ (Dynamic test) ซึ่งในการทดลองกรณีที่ 2 ผู้วิจัยได้ตั้งสมมติฐานในการทดลองไว้ 2 ข้อ ได้แก่

- ตำแหน่งตามระยะแกน Z มีผลต่อตำแหน่งตามระยะแกน Y กล่าวคือ ถ้าระยะตามแกน Z มากขึ้นทำให้ค่าพิกัดตามแกน Y มีความคลาดเคลื่อนเพิ่มขึ้น

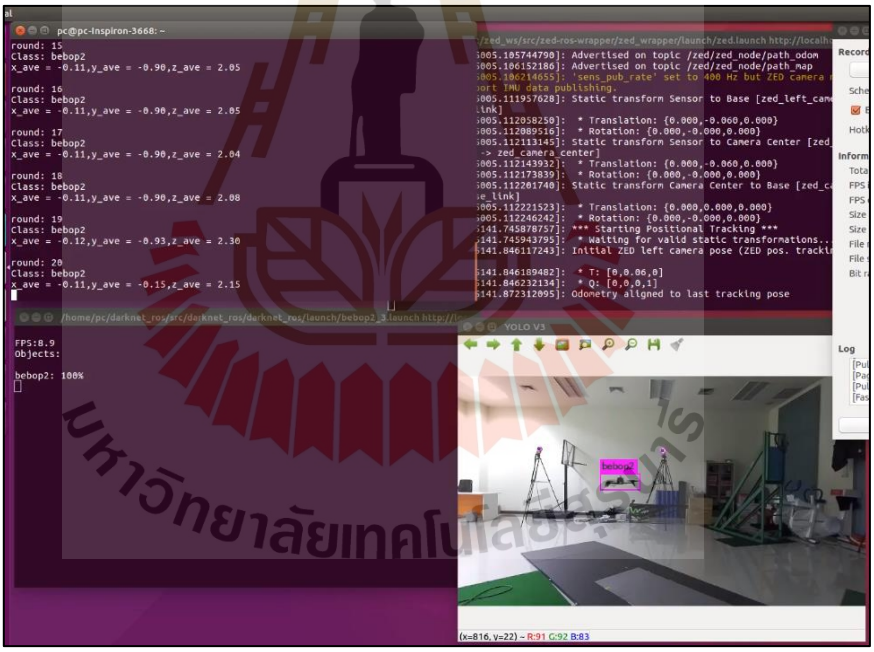
- ตำแหน่งตามระยะแกน X มีผลต่อค่าตำแหน่งตามระยะแกน Z กล่าวคือค่าความถูกต้องของตำแหน่งในแกน Z จะเพิ่มขึ้นเมื่อตำแหน่งวัตถุห่างจากตำแหน่งศูนย์กลางของภาพไปทางขวาและซ้าย เนื่องจากความโค้งของเลนส์ภาพ

ดังนั้น ผู้วิจัยจึงออกแบบการทดลองให้ทดสอบหาค่าความคลาดเคลื่อนที่ตำแหน่งความลึก 3 ตำแหน่ง คือ 2.0, 2.5 และ 3.0 เมตร โดยที่ความลึกต่าง ๆ โดรนจะเคลื่อนที่ไปตามระนาบ X และ Y ให้มีการเคลื่อนที่ตามแนวแกน Z ให้น้อยที่สุด เพื่อพิจารณาผลที่ได้ว่าเป็นไปตามสมมติฐานหรือไม่ โดยทดสอบตำแหน่งละ 3 ครั้ง ใช้เวลาครั้งละ 60 วินาที จากนั้นจึงประเมินผลจากการทดลองที่ได้เทียบกับค่าพิกัดที่ได้จากระบบ Motion capture ตัวอย่างจอแสดงผลจากระบบ Motion capture แสดงดังรูปที่ 3.23 และตัวอย่างจอแสดงผลจากระบบกล้องถ่ายภาพ 3 มิติ แสดงดังรูปที่ 3.24

การทดสอบค่าความผันผวนของจุดศูนย์กลางของตำแหน่งวัตถุเป็นการทดสอบ เพื่อประกอบการพิจารณาว่าระบบตรวจจับวัตถุว่ามีความผันผวนมากหรือน้อยเพียงใด การทดสอบทำโดยการให้โดรนอยู่ที่ตำแหน่งเดิมและมีการวัดค่าพิกัดอย่างต่อเนื่องเป็นเวลา 60 วินาที ตำแหน่งในการทดลองของโดรนจะแบ่งเป็นกริด 0.5×0.5 ม. ให้ครอบคลุมภายในบริเวณพื้นที่ทดลองและมีความลึกตั้งแต่ 2.0-3.0 เมตร โดยทำการทดลองโดยเปลี่ยนตำแหน่งทั้งสิ้น 11 ตำแหน่ง และทำการเก็บข้อมูลตำแหน่งละ 3 ครั้ง จากนั้นจึงนำข้อมูลที่ได้มาคำนวณค่าความผันผวนต่อไป



รูปที่ 3.23 ตัวอย่างจอแสดงผลด้วยระบบระบุพิกัดจากระบบ Motion capture



รูปที่ 3.24 ตัวอย่างจอแสดงผลด้วยระบบระบุพิกัดจากระบบกล้องถ่ายภาพ 3 มิติ

3.3 การพัฒนาระบบนำทางของอากาศยานด้วยภาพ

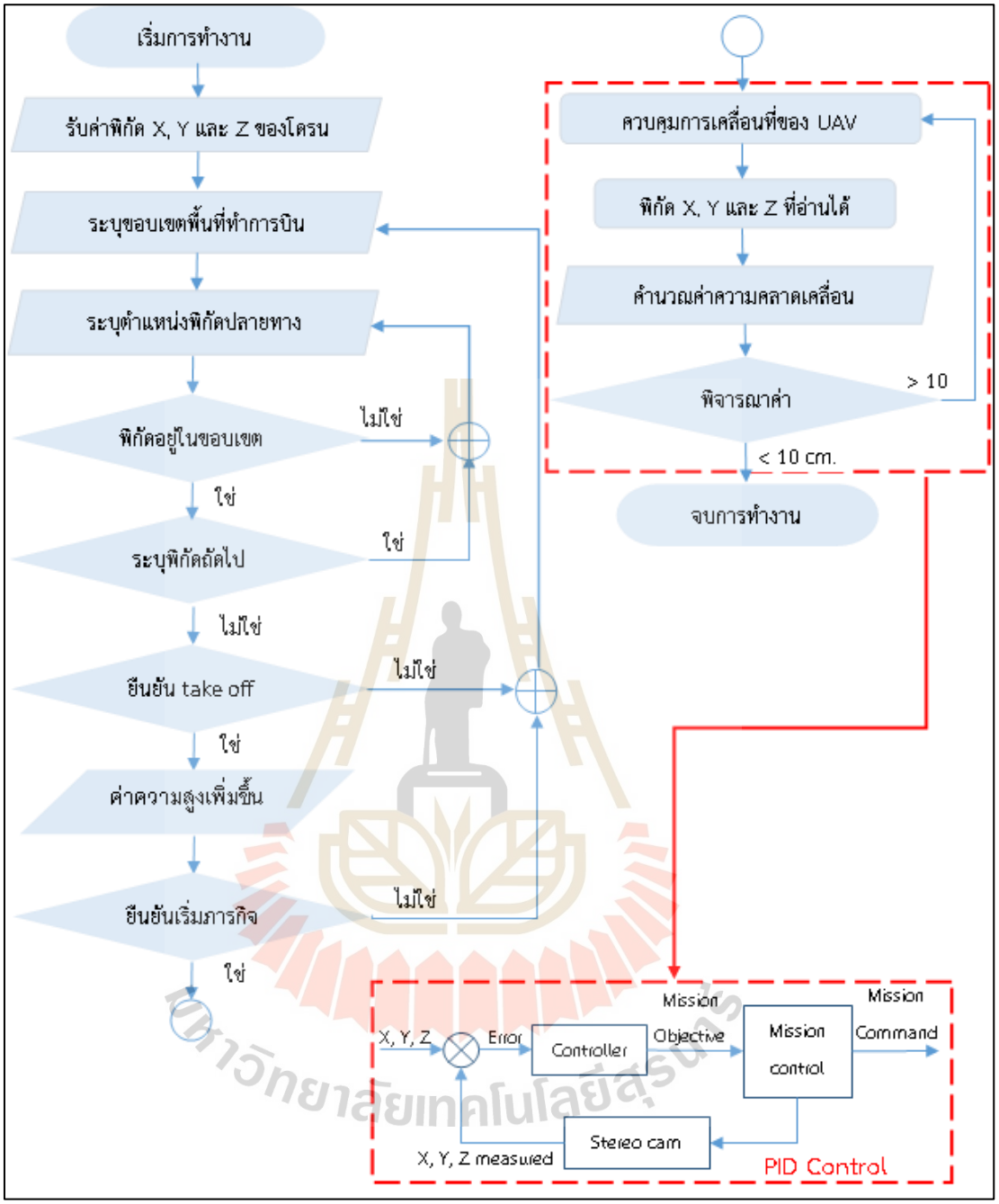
3.3.1 ออกแบบระบบการทำงาน

ระบบนำทางอากาศยานถูกพัฒนาเพื่อใช้สำหรับควบคุมการเคลื่อนที่ของ Parrot bebop 2 ให้เคลื่อนที่ไปยังตำแหน่งที่ต้องการแบบอัตโนมัติด้วยการระบุพิกัดจากกล้องถ่ายภาพ

3 มิติ ซึ่งเป็นการควบคุมแบบ High level ด้วย ROS การเชื่อมต่อและรับ-ส่งข้อมูลระหว่างโดรนและคอมพิวเตอร์สำหรับประมวลผลทำได้โดยอาศัยการทำงานผ่านสัญญาณ Wi-fi

การทำงานเริ่มเมื่อระบบได้รับค่าพิกัดของวัตถุจากระบบระบุพิกัด จากนั้นจึงเริ่มให้ผู้ใช้ระบุขอบเขตพื้นที่ทำการบินในหน่วยเมตรเพื่อป้องกันการป้อนค่าตำแหน่งที่นอกเหนือขอบเขตพื้นที่ทดลอง จากนั้นจึงทำการป้อนค่าพิกัดปลายทางที่ต้องการ โดยค่าพิกัดนั้นจะต้องอยู่ภายในขอบเขตพื้นที่ที่ระบุไว้ และอยู่ในขอบเขตการมองเห็นของกล้อง (Field of view) โดยที่ผู้ใช้สามารถป้อนค่าพิกัดที่ต้องการได้มากกว่า 1 พิกัด เมื่อป้อนค่าพิกัดตามเงื่อนไขข้างต้นเรียบร้อยแล้วระบบจะทำการสั่งให้โดรนทำการ Take off ซึ่งก่อนการ Take off ผู้ใช้จะต้องทำการยืนยันก่อน เมื่อโดรนทำการ Take off เรียบร้อย ระบบจะขึ้นข้อความให้ผู้ใช้ยืนยันการทำภารกิจ เมื่อทำการยืนยันแล้วระบบจะควบคุมโดรนให้เคลื่อนที่ไปยังพิกัดเป้าหมาย โดยใช้การควบคุมความเร็วในการเคลื่อนที่ของแต่ละแกน ซึ่งกล้องถ่ายภาพ 3 มิติ ทำงานคล้ายกับอุปกรณ์เซ็นเซอร์ในการระบุพิกัดเสมือนการใช้ GPS นอกอาคาร จากนั้นนำพิกัดที่วัดได้จากกล้องถ่ายภาพ 3 มิติ มาเทียบกับพิกัดเป้าหมายที่ระบุไว้ หากไม่ตรงกันจะทำการปรับให้โดรนมีการเคลื่อนที่เข้าหาจุดนั้น ซึ่งในขั้นตอนนี้จะใช้ตัวควบคุม PID ในการควบคุมการเคลื่อนที่ของโดรนให้ไปยังตำแหน่งเป้าหมายได้ดียิ่งขึ้น นอกจากนั้นมีการกำหนดบริเวณที่ยอมรับได้ (Death Zone) เป็นรัศมี 10 เซนติเมตร รอบพิกัดเป้าหมาย เพื่อไม่ให้โดรนเกิดการแกว่ง เนื่องจากค่าพิกัดไม่ตรงกันพอดี เมื่อโดรนอยู่ที่ตำแหน่งเป้าหมายแล้ว ระบบจะทำการตรวจสอบว่ามีพิกัดเป้าหมายถัดไปหรือไม่ หากระบบตรวจสอบแล้วไม่มีพิกัดเป้าหมายถัดไปจะส่งคำสั่งให้โดรนทำการ Landing ลงที่ตำแหน่งเป้าหมายสุดท้ายทันที

ทั้งนี้ หากเกิดปัญหาระหว่างการทำภารกิจอันเนื่องจาก ระบบระบุพิกัดไม่สามารถตรวจจับวัตุเป้าหมายได้ หรือวัตถุเป้าหมายเคลื่อนที่ออกนอกบริเวณพื้นที่การมองเห็นของกล้อง ระบบจะทำการสั่งการให้โดรนหยุดเคลื่อนที่และลอยตัวอยู่ที่ตำแหน่งนั้นทันทีจนกว่าผู้ใช้จะส่งคำสั่งไป ซึ่งการทำงานของระบบควบคุมการเคลื่อนที่ที่กล่าวมาข้างต้นสามารถอธิบายได้ด้วยแผนภาพการทำงานแสดงดังรูปที่ 3.25



รูปที่ 3.25 แผนภาพการทำงานของระบบนำทางด้วยกล้อง Stereo camera

3.3.2 ทดสอบขอบเขตการมองเห็นของกล้องถ่ายภาพ 3 มิติ

ก่อนเริ่มภารกิจระบบได้ให้ผู้ใช้ทำการกำหนดขอบเขตพื้นที่ดำเนินการเพื่อป้องกันการระบุพิกัดนอกขอบเขตพื้นที่ทดลอง แต่การกำหนดพื้นที่ทดลองเพียงอย่างเดียวอาจไม่เพียงพอต่อความปลอดภัย เนื่องจากมีบางมุมในพื้นที่ทดลองที่อยู่นอกขอบเขตการมองเห็นของกล้อง

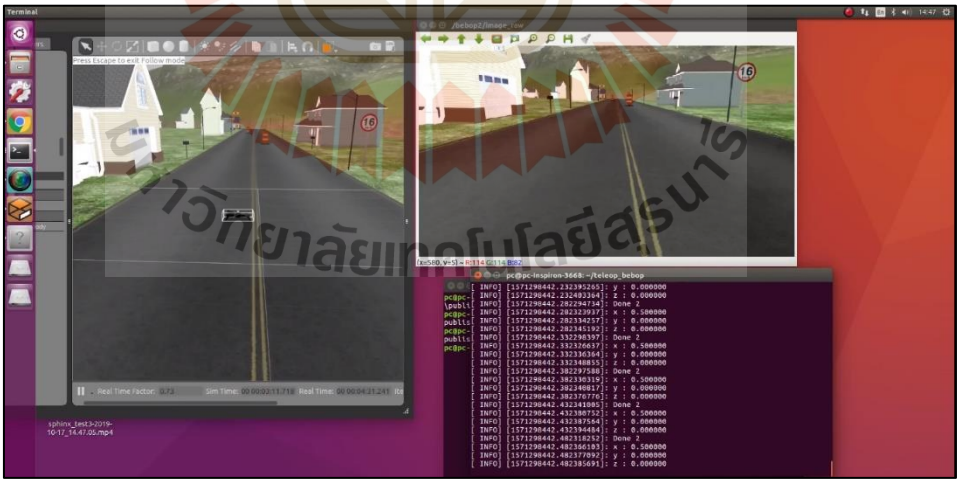
(Field of view) ดังนั้นผู้วิจัยจึงทำการทดสอบขอบเขตการมองเห็นของกล้องและสร้างสมการที่ได้จากการทดสอบเพิ่มในเงื่อนไข เพื่อป้องกันผู้ใช้ไม่ให้ระบุพิกัดเป้าหมายในบริเวณจุดอับของกล้อง

การทดลองทำโดยการวัดความกว้างที่กล้องมองเห็น (X_{FOV}) ที่ระยะความลึก 6 ตำแหน่ง คือ 1.0, 1.5, 2.0, 2.5, 3.0 และ 4.0 เมตร ตำแหน่งละ 3 ครั้ง ด้วยตลับเมตร หลังจากนั้นจึงประเมินข้อมูลที่ได้พล็อทเป็นกราฟ และสร้างสมการใส่เพิ่มในเงื่อนไขการทำงาน

3.3.3 ทดสอบการทำงานของระบบนำทางในโปรแกรมจำลอง

การทดสอบด้วยโปรแกรมจำลองเป็นอีกขั้นตอนหนึ่งที่จำเป็น ก่อนการนำโดรนไปทดลองบินจริง เพื่อลดความเสียหายที่อาจเกิดขึ้นจากความผิดพลาดของอัลกอริทึม การคำนวณในโปรแกรม หรือความผิดพลาดอื่น ๆ

โปรแกรมจำลองที่ใช้คือ Parrot sphinx เป็นโปรแกรมจำลองที่พัฒนาต่อจากโปรแกรม Gazebo เพื่อใช้สำหรับการจำลองการบินของโดรนในค่าย Parrot โดยเฉพาะ รูปตัวอย่างแสดงลักษณะของโปรแกรมดังรูปที่ 3.26 การจำลองการบินมีความสมจริงผู้ใช้งานสามารถปรับหรือตั้งค่าพารามิเตอร์ต่าง ๆ และสิ่งแวดล้อมได้ นอกจากนั้นโปรแกรมถูกพัฒนาให้สามารถใช้งานร่วมกับ ROS ได้ จึงทำให้การทดสอบสามารถทำได้เสมือนจริงมากขึ้น แต่เนื่องจากระบบระบุพิกัดใช้กล้องถ่ายภาพ 3 มิติ ในการระบุพิกัด ซึ่งไม่สามารถออกแบบการทดสอบให้ใช้กล้องถ่ายภาพ 3 มิติ ในระบบโปรแกรมจำลองได้



รูปที่ 3.26 การจำลองการบิน Parrot Bebop 2 ด้วยโปรแกรมจำลอง Parrot sphinx

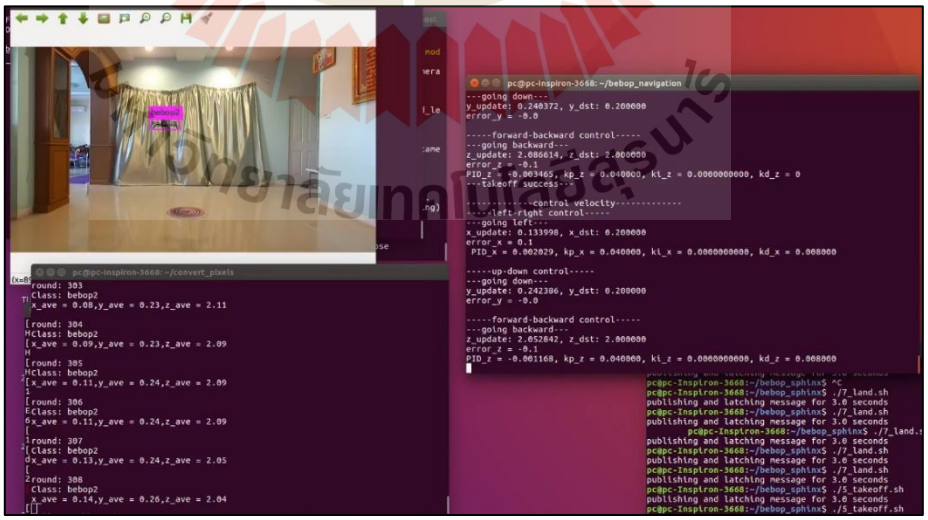
ดังนั้น ผู้วิจัยจึงทำการระบุค่าพิกัดจำลองแทนค่าพิกัดที่วัดได้จากกล้องถ่ายภาพ 3 มิติ แต่อย่างไรก็ตาม การที่ไม่สามารถใช้กล้องถ่ายภาพ 3 มิติ ในระบบทดสอบจำลองได้ไม่ใช่

ปัญหาสำคัญ เนื่องจากการทดสอบด้วยโปรแกรมจำลองนี้ มีวัตถุประสงค์เพียงเพื่อทดสอบความถูกต้องของอัลกอริทึมการควบคุมโดรน เพื่อพิจารณาว่าโดรนสามารถเคลื่อนที่ไปตามทิศทางที่กำหนดได้ถูกต้องหรือไม่ และสามารถหยุดเมื่อถึงพิกัดเป้าหมายแล้วได้หรือไม่ หากการทดสอบด้วยโปรแกรมจำลองมีอัลกอริทึมการควบคุมที่ถูกต้องจึงสามารถนำไปทดสอบจริงในส่วนถัดไป

3.3.4 ทดสอบการทำงานของระบบนำทางด้วยกล้อง Stereo camera

การทดสอบในขั้นตอนนี้เป็นการทดสอบระบบการทำงานแบบเต็มรูปแบบ กล่าวคือ มีการตรวจจับวัตถุจากกล้องถ่ายภาพ 3 มิติ และส่งคำสั่งควบคุมโดรนให้ทำการเคลื่อนที่ตามภารกิจ โดยก่อนทำการทดสอบระบบนำทางผู้วิจัยได้ทำการทดลองหาค่าตัวควบคุม K_P, K_I และ K_D ที่เหมาะสมสำหรับควบคุมการเคลื่อนที่ของโดรนให้มีเสถียรภาพตามต้องการ การทดลองใช้วิธี Trial and error โดยเริ่มจากการสุ่มเลือกค่า K_P ของทั้งสามแกนจนได้ค่าที่เหมาะสม จึงเพิ่ม K_I และ K_D ตามลำดับ การพิจารณาถึงค่าที่เหมาะสมจะพิจารณาจากพฤติกรรมของโดรนที่เกิดขึ้นเมื่อค่าตัวควบคุมเปลี่ยนไป และพิจารณาจากการนำผลที่ได้วิเคราะห์พฤติกรรมของโดรนผ่านกราฟเพื่อทำให้สามารถพิจารณาได้อย่างละเอียดมากขึ้น

เมื่อได้ค่าตัวควบคุมที่เหมาะสมกับระบบนำทางจึงเริ่มทำการทดสอบระบบแบบเต็มรูปแบบ โดยการทดสอบนี้ผู้วิจัยได้ป้อนค่าพิกัดเป้าหมาย 4 พิกัด ให้โดรนบินเป็นรูป 4 เหลี่ยม และเมื่อภารกิจสำเร็จ ผู้วิจัยได้นำผลการระบุพิกัดที่ได้จากการทดสอบมาพิจารณาและวิเคราะห์ผลในรูปแบบของกราฟในส่วนถัดไป รูปตัวอย่างระหว่างการทดสอบแสดงดังรูปที่ 3.27



รูปที่ 3.27 ตัวอย่างการทดสอบระบบระบุพิกัดแบบเต็มรูปแบบ

บทที่ 4
ผลการดำเนินการวิจัย

4.1 ผลการทดลองการเรียนรู้ของโมเดล

4.1.1 ผลการรวบรวมและจัดเตรียมชุดข้อมูลภาพ

จากการรวบรวมข้อมูลภาพด้วยวิธีการที่กล่าวไว้ดังหัวข้อ 3.1.1 จึงได้ข้อมูลภาพทั้งสิ้นจำนวน 1,493 ภาพ ผู้วิจัยได้นำภาพจำนวนดังกล่าวผ่านกระบวนการจัดเตรียมดังนี้

1. การกลับภาพในแนวตั้งและแนวนอน (Vertical/Horizontal flip) ผลจากการกลับภาพในแนวตั้งและแนวนนอน ทำให้จำนวนรูปภาพที่ได้มีจำนวนเพิ่มขึ้นเป็น 8 เท่าจากจำนวนเดิม ดังนั้นจำนวนรูปภาพหลังผ่านกระบวนการนี้จึงมีจำนวนทั้งสิ้น 11,944 ภาพ

2. การปรับความเข้มและความสว่างของภาพ (Contrast) ชุดข้อมูลภาพจำนวน 11,944 ภาพ ที่ผ่านกระบวนการดังหัวข้อก่อนหน้าจะถูกนำเข้าสู่กระบวนการการปรับความเข้มและความสว่างของภาพ ซึ่งหลังจากที่ชุดข้อมูลภาพได้ผ่านกระบวนการดังกล่าวทำให้จำนวนชุดข้อมูลภาพเพิ่มขึ้นเป็น 2 เท่า ดังนั้นจึงมีจำนวนรูปภาพหลังผ่านกระบวนการทั้งสิ้น 23,888 ภาพ

3. การปรับความคมชัดของภาพ (Blur) ชุดข้อมูลภาพจำนวน 23,888 ภาพ ถูกนำมาผ่านกระบวนการปรับความคมชัด ทำให้จำนวนข้อมูลภาพหลังผ่านกระบวนการเพิ่มขึ้นเป็น 1.5 เท่า ดังนั้นจึงมีจำนวนรูปภาพหลังผ่านกระบวนการทั้งสิ้นจำนวน 35,832 ภาพ

4. การแบ่งชุดข้อมูล ผู้วิจัยได้ทำการแบ่งชุดข้อมูลภาพสำหรับการเทรนโมเดล (Training set) และการเทียบความผิดพลาด (Validation set) ด้วยการติดป้ายกำกับในแพลทฟอร์ม supervise.ly โดยข้อมูลจำนวน 28,666 ภาพเป็นข้อมูลสำหรับการเทรนโมเดลคิดเป็นร้อยละ 80 ของจำนวนภาพทั้งหมด และข้อมูลภาพจำนวน 7,166 ภาพ เป็นข้อมูลสำหรับการเทียบความผิดพลาดคิดเป็นร้อยละ 20 ของจำนวนภาพทั้งหมด

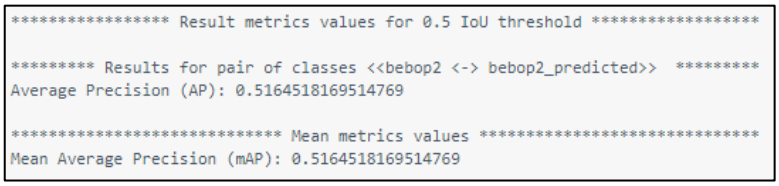
4.1.2 ผลการทดลองการปรับค่าไฮเปอร์พารามิเตอร์

การทดลองปรับค่าไฮเปอร์พารามิเตอร์ทำการปรับพารามิเตอร์ ซึ่งได้แก่ จำนวนรอบ (Epochs) ขนาดข้อมูล (Batch size) และขนาดของรูปภาพ โดยทำการปรับค่าตามตารางที่ 4.1 โดยเมื่อการทดลองเสร็จสิ้นได้โมเดลที่สมบูรณ์ทั้งสิ้น จำนวน 4 โมเดล

ตารางที่ 4.1 ตารางแสดงการปรับค่าไฮเปอร์พารามิเตอร์สำหรับเทรนโมเดล

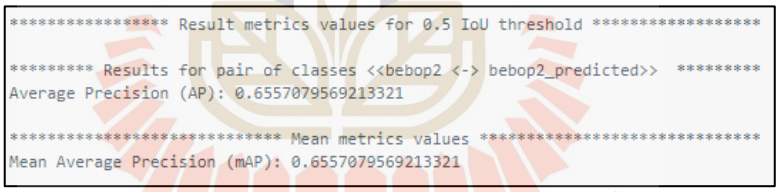
ที่	อัตราการเรียนรู้ (Learning rate)	จำนวนรอบ (Epochs)	ขนาดข้อมูล (Batch size)	ขนาดรูปภาพ (Input size)	หมายเหตุ
1	0.0001	10000	8	416×416	
2	0.0001	10000	2	416×416	
3	0.0001	10000	1	416×416	
4	0.0001	2000	1	416×416	
5	0.0001	700	1	416×416	
6	0.0001	500	1	416×416	
7	0.0001	100	1	416×416	โมเดลที่ 1
8	0.0001	100	1	608×608	
9	0.0001	150	1	416×416	โมเดลที่ 2
10	0.0001	200	1	416×416	โมเดลที่ 3
11	0.0001	100	1	416×416	โมเดลที่ 4

จากตารางที่ 4.1 การทดลองครั้งที่ 1 โมเดลไม่สามารถเทรนสำเร็จได้ เนื่องจากปัญหาเรื่องขนาดของชุดข้อมูลที่มีขนาดใหญ่เกินกว่าความสามารถของ GPU ผู้วิจัยจึงได้ทำการลดขนาดของชุดข้อมูลลงเหลือ 2 ในการทดลองครั้งที่ 2 แต่ในครั้งนี้โมเดลก็ไม่สามารถเทรนได้สำเร็จเช่นกันเนื่องจากปัญหาของขนาดชุดข้อมูล การทดลองครั้งที่ 3 จึงทำการลดขนาดให้เหลือเพียง 1 แต่ก็ยังไม่สามารถเทรนได้สำเร็จ เนื่องจากจำนวนรอบที่เยอะเกินไป ในการทดลองครั้งที่ 4, 5 และ 6 จึงได้ทำการปรับลดรอบลง จนกระทั่งการทดลองที่ 7 จึงสามารถเทรนโมเดลได้สำเร็จเป็นโมเดลแรกโดยที่ใช้เวลาในการเทรนทั้งสิ้นประมาณ 14 วัน จากนั้นผู้ทดลองจึงนำโมเดลที่ 1 ทดสอบกับชุดทดสอบ (Test set) ซึ่งเป็นข้อมูลภาพที่ไม่อยู่ในชุดข้อมูลสำหรับการเทรนจำนวนทั้งสิ้น 3,510 ภาพ โดยภาพเป็นการบันทึกจากสถานที่ทดลองทั้ง 2 แห่ง ดังที่กล่าวไว้ก่อนหน้า ซึ่งผลของการทดสอบของโมเดลที่ 1 กับชุดทดสอบพบว่า โมเดลมีค่าเฉลี่ยความแม่นยำ (Average Precision) โดยประมาณอยู่ที่ 0.516 แสดงผลดังรูปที่ 4.1 ซึ่งประสิทธิภาพของโมเดลที่ 1 นั้นอยู่ในระดับต่ำและยังไม่สามารถนำไปใช้งานได้



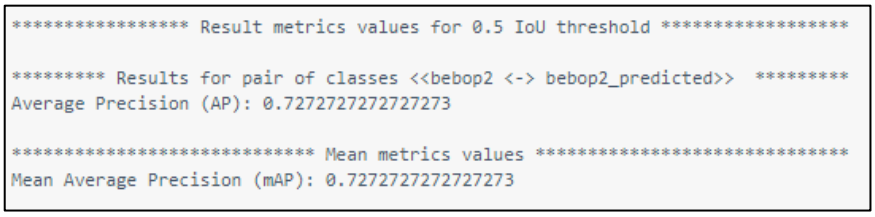
รูปที่ 4.1 ผลการทดสอบโมเดลที่ 1 กับชุดข้อมูลทดสอบ

ในการทดลองครั้งที่ 8 จึงได้นำโมเดลที่ 1 มาเทรนต่อโดยใช้ข้อมูลชุดเดิม และเพิ่มขนาดรูปภาพเป็น 608×608 พิกเซล เพื่อให้รูปมีความละเอียดขึ้น แต่การเทรนนั้นไม่สำเร็จ เนื่องจากขนาดความจุของ GPU ไม่เพียงพอต่อการเพิ่มขนาดรูปภาพ ทำให้ในครั้งที่ 8 ได้ลดขนาดรูปภาพลงเหลือเท่าเดิมและเพิ่มรอบของการเทรนเป็น 150 รอบ ซึ่งสามารถเทรนได้สำเร็จโดยใช้เวลาทั้งสิ้นประมาณ 22 วัน จากนั้นจึงนำโมเดลที่ได้ทดสอบกับชุดทดสอบและได้ค่าเฉลี่ยความแม่นยำอยู่ที่ประมาณ 0.655 แสดงผลดังรูปที่ 4.2 ซึ่งค่าที่ได้เพิ่มขึ้นจากการเทรนโมเดลที่ 1 ซึ่งสามารถวิเคราะห์ได้ว่าโมเดลที่ 2 นั้นมีประสิทธิภาพมากกว่า แต่ประสิทธิภาพที่ได้ยังอยู่ในระดับกลางและยังไม่เพียงพอต่อการตัดสินใจนำไปงานจริง



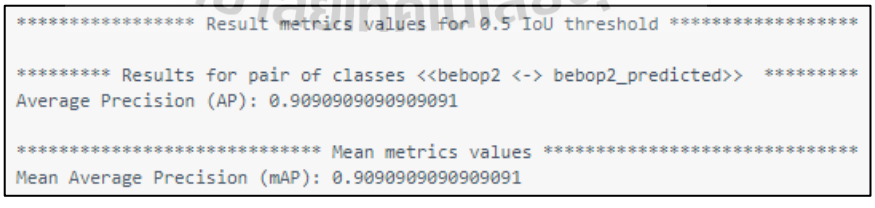
รูปที่ 4.2 ผลการทดสอบโมเดลที่ 2 กับชุดข้อมูลทดสอบ

การทดลองครั้งที่ 10 ได้นำโมเดลที่ 9 มาพัฒนาต่อโดยในครั้งนี้ได้ทำการเทรนที่จำนวน 200 รอบ และใช้ข้อมูลชุดเดิมในการเทรน ซึ่งสามารถเทรนได้สำเร็จเป็นโมเดลที่ 3 และเมื่อนำไปทดสอบกับชุดทดสอบแล้วพบว่าโมเดลที่ 10 มีค่าเฉลี่ยความแม่นยำเพิ่มขึ้นจากโมเดลที่ 9 โดยมีค่าเฉลี่ยความแม่นยำอยู่ที่ 0.7272 โดยประมาณ แสดงผลดังรูปที่ 4.3 ซึ่งโมเดลที่ได้ถือว่ามีประสิทธิภาพอยู่ในระดับดีและสามารถนำไปใช้งานได้



รูปที่ 4.3 ผลการทดสอบโมเดลที่ 3 กับชุดข้อมูลทดสอบ

อย่างไรก็ตาม แม้ว่าโมเดลที่ 3 เมื่อทดสอบแล้วจะได้ค่าเฉลี่ยความแม่นยำอยู่ในระดับที่ดีและสามารถนำไปใช้งานได้แล้ว แต่ผู้วิจัยยังคงทำการเทรนโมเดลต่อไปตามแนวโน้มของประสิทธิภาพโมเดลที่เพิ่มขึ้นเมื่อมีการเทรนมากขึ้นเพื่อให้โมเดลนั้นมีประสิทธิภาพที่ดียิ่งขึ้น การทดลองครั้งที่ 11 จึงเป็นการนำโมเดลที่ 3 มาเทรนต่อด้วยข้อมูลชุดเดิมเป็นจำนวน 100 รอบ เพื่อลดระยะเวลาในการเทรนให้ลดลง ซึ่งการเทรนโมเดลที่ 4 ในครั้งนี้สำเร็จภายในระยะเวลาประมาณ 14 วัน เมื่อทำการทดสอบด้วยชุดทดสอบแล้วพบว่าค่าเฉลี่ยความแม่นยำอยู่ที่ประมาณ 0.90 ซึ่งเป็นโมเดลที่มีประสิทธิภาพอยู่ในระดับที่ดีมาก แสดงผลดังรูปที่ 4.4 ดังนั้น หากทำการพิจารณาผลของการปรับค่าไฮเปอร์พารามิเตอร์จะพบว่า จำนวนรอบที่ใช้ในการเทรนมีผลต่อประสิทธิภาพของโมเดล กล่าวคือ เมื่อโมเดลมีการเรียนรู้ด้วยจำนวนรอบที่มากขึ้นประสิทธิภาพของโมเดลจึงเพิ่มขึ้น นอกจากนั้นจำนวนรอบที่ใช้ในการเทรนส่งผลถึงระยะเวลาที่ใช้สำหรับการเทรนด้วย โดยที่จำนวนรอบที่มากขึ้นทำให้ใช้เวลานานขึ้น สำหรับไฮเปอร์พารามิเตอร์อื่น ๆ นั้นในงานวิจัยนี้ไม่สามารถทดลองปรับค่าได้มากนัก เนื่องจากข้อจำกัดทางด้านอุปกรณ์ฮาร์ดแวร์ แต่ถึงอย่างไรโมเดลที่เทรนได้สำเร็จนั้นก็มีประสิทธิภาพที่สามารถนำไปใช้งานได้อย่างดี



รูปที่ 4.4 ผลการทดสอบโมเดลที่ 4 กับชุดข้อมูลทดสอบ

4.1.3 ผลการทดสอบการใช้งานแบบเรียลไทม์

ถึงแม้ว่าผลการทดสอบโมเดลทั้ง 4 โมเดล ด้วยชุดทดสอบทำให้ทราบถึงผลค่าเฉลี่ยความแม่นยำที่สามารถประกอบการพิจารณานำไปใช้ได้และบางโมเดลอยู่ในระดับ

ที่ประสิทธิภาพของโมเดลอยู่ในระดับที่ดีถึงดีมาก แต่อย่างไรก็ควรนำโมเดลทดสอบการใช้งานแบบเรียลไทม์ เพื่อประกอบการพิจารณานำไปใช้จริง โดยการทดสอบแบบเรียลไทม์จะทดสอบเฉพาะโมเดลที่ 3 และ 4 เนื่องจากมีประสิทธิภาพที่ผ่านเกณฑ์การพิจารณานำไปใช้งานได้ การทดสอบจะให้โดรนทำการบินและใช้กล้องถ่ายภาพ 3 มิติ ตรวจจับวัตถุแบบเรียลไทม์ โดยใช้สถานที่บริเวณบ้านของผู้วิจัยในการทดสอบ เพื่อให้ภาพที่ได้มีความแตกต่างจากชุดข้อมูลภาพสำหรับการเทรนและชุดทดสอบโดยสิ้นเชิง

การทดสอบโมเดลที่ 3 เมื่อทดสอบด้วยการตรวจจับวัตถุแบบเรียลไทม์แล้ว พบว่า การตรวจจับวัตถุมีประสิทธิภาพดี กล่าวคือ เมื่อโดรนเคลื่อนที่โมเดลสามารตรวจจับวัตถุได้ตลอดและตรงตำแหน่ง มีการจับผิดพลาดบ้างบางจุดแต่ผิดพลาดเพียงเล็กน้อย หากทำการประเมินโดยผู้วิจัยโมเดลนี้เมื่อนำไปใช้จริงแบบเรียลไทม์เป็นโมเดลที่มีความสามารถในการตรวจจับที่อยู่ในระดับดีมาก ผลการทดสอบแบบเรียลไทม์ด้วยโมเดลที่ 3 แสดงดังรูปที่ 4.5

รูปที่ 4.5 ตัวอย่างผลการทดสอบการตรวจจับแบบเรียลไทม์ด้วยโมเดลที่ 3

การทดสอบการตรวจจับวัตถุแบบเรียลไทม์ด้วยโมเดลที่ 4 ที่มีความแม่นยำสูงสุดนั้นพบว่า เมื่อเริ่มการตรวจจับมีการตรวจจับติดตามวัตถุได้ดีในขณะที่วัตถุหยุดนิ่ง แต่มีความผิดพลาดในการตรวจจับวัตถุที่ไม่ใช่วัตถุเป้าหมายมากกว่าโมเดลที่ 3 และในบางครั้งยังไม่สามารถตรวจจับวัตถุในขณะเคลื่อนที่เร็วได้ ผลการทดสอบแบบเรียลไทม์ด้วยโมเดลที่ 3 แสดงดังรูปที่ 4.6

รูปที่ 4.6 ตัวอย่างผลการทดสอบการตรวจจับแบบเรียลไทม์ด้วยโมเดลที่ 4

สาเหตุที่ทำให้โมเดลที่ 4 เมื่อใช้งานจริงมีประสิทธิภาพที่ด้อยกว่าโมเดลที่ 3 ผู้วิจัยได้ทำการพิจารณาผลและวิเคราะห์ว่าเกิดจากการที่โมเดลมีความ Overfitting ซึ่งการเกิด Overfitting model สามารถเกิดได้จากหลายปัจจัยเช่น การปรับค่าไฮเปอร์พารามิเตอร์ การเทรนด้วยจำนวนข้อมูลที่เยอะเกินไป เป็นต้น ซึ่งสำหรับสาเหตุหลัก ๆ ที่เกิดขึ้นกับปัญหาของโมเดลที่ 4 นั้นคือ การปรับค่าไฮเปอร์พารามิเตอร์รอบของการเทรนซึ่งเป็นการเทรนโมเดลต่อโมเดลทำให้การเรียนรู้ของโมเดลสะสมเรื่อย ๆ และเมื่อเกิดการเรียนรู้ที่มากเกินไปจึงทำให้โมเดลเกิดปัญหาเรื่อง Over fit ได้ ส่วนสาเหตุที่เกิดเพราะจำนวนข้อมูลที่มากเกินไปนั้น ผู้วิจัยวิเคราะห์ว่าไม่ได้เกิดจากสาเหตุนี้ เนื่องจากปัญหา Overfitting model ที่เกิดจากจำนวนข้อมูลนั้นสามารถแก้ได้ด้วยการทำ Cross validation คือ การแบ่งข้อมูลเป็นชุดย่อย ๆ 3 ชุด ซึ่งผู้วิจัยได้มีการทำวิธี Cross validation ในงานวิจัยนี้แล้ว ดังนั้นจึงสามารถวิเคราะห์ได้ว่าไม่ใช่สาเหตุของการเกิดปัญหา Overfitting ของโมเดลที่ 4

4.2 ประสิทธิภาพของระบบระบุพิกัดด้วยกล้องถ่ายภาพ 3 มิติ

4.2.1 ผลการทดสอบค่าความถูกต้องกรณีที่โดรนไม่เคลื่อนที่

จากการทดสอบหาค่าความถูกต้องโดยทดสอบแบบ Static test และเปรียบเทียบกับระบบระบุพิกัดด้วยระบบ Motion capture ได้ผลการทดลองบางส่วน ดังตารางที่ 4.2 ซึ่งหากทำการคำนวนหาค่าเฉลี่ยความคลาดเคลื่อนของแต่ละแกนแล้วพบว่า แกน x มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.05 เมตร แกน y มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.04 เมตร และแกน z มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.03 เมตร ซึ่งจากการพิจารณาผลการทดลองพบว่า ค่าเฉลี่ยความคลาดเคลื่อนของทั้ง 3 แกน มีค่า ไม่เกิน 0.10 เมตร ตามที่ตั้งไว้ ดังนั้นจึงถือว่าระบบระบุพิกัดมีความแม่นยำ

ตารางที่ 4.2 ตารางแสดงผลการทดสอบความแม่นยำของระบบระบุพิกัดด้วยกล้องถ่ายภาพสามมิติเทียบกับระบบ Motion capture ในกรณีที่โดรนไม่เคลื่อนที่

ค่าพิกัดที่วัดได้จากระบบ Motion capture (เมตร)			**ค่าเฉลี่ยพิกัดที่วัดได้จากกล้องถ่ายภาพสามมิติ (เมตร)**		
X	**Y**	**Z**	**X**	**Y**	**Z**
1	1.5	2.5	0.99	1.5	2.55
0.5	1	2	0.58	1.09	1.97
-0.5	0.5	1.5	0.46	0.48	1.5
0	0	1	0.04	-0.06	0.97

4.2.2 ผลการทดสอบค่าความถูกต้องกรณีที่โดรนเคลื่อนที่

การทดสอบค่าความถูกต้องของระบบระบุพิกัดด้วยภาพถ่าย 3 มิติ จะทำการพิจารณาและวิเคราะห์ผลเทียบกับการระบุพิกัดด้วยระบบ Motion capture และได้ทำการทดลองตามสมมติฐานที่ได้ตั้งไว้ และได้ผลการทดลองตามสมมติฐาน ดังนี้

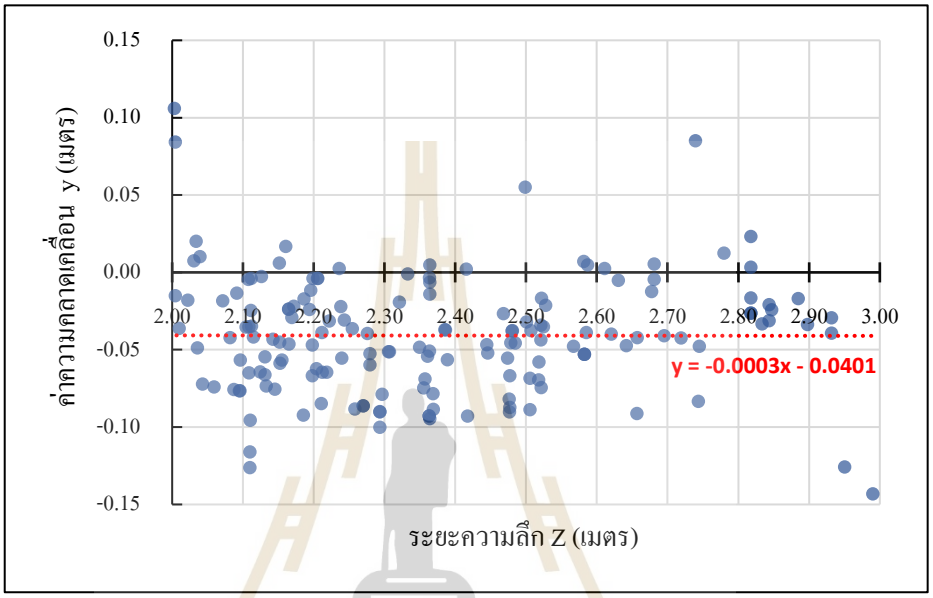
1. ตำแหน่งตามระยะแกน Z มีผลต่อตำแหน่งตามระยะแกน Y กล่าวคือ ถ้าระยะตามแกน Z มากขึ้นทำให้ค่าพิกัดตามแกน Y มีความคลาดเคลื่อนเพิ่มขึ้น

การทดลองตามสมมติฐานที่ 1 แสดงดังรูปที่ 4.7 จากรูปเป็นกราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อนของค่าความสูง y และความลึก z โดยสามารถพิจารณาได้ว่าค่าความคลาดเคลื่อน y ที่ได้จากการทดลองนั้นมีแนวโน้มการเปลี่ยนแปลงที่น้อยมาก โดยมีอัตราการเปลี่ยนแปลงของค่าความคลาดเคลื่อนอยู่ที่ประมาณ 0.0003 ซึ่งสามารถอนุมานได้ว่าค่าความคลาดเคลื่อน y ไม่เปลี่ยนแปลงตามระยะความลึก z และมีค่าความคลาดเคลื่อนโดยสูงสุด 0.11 และค่าความคลาดเคลื่อนต่ำสุดคือ -0.14 ส่วนค่าความคลาดเคลื่อนเฉลี่ยมีค่าประมาณ -0.04 ซึ่งค่าความคลาดเคลื่อนที่ได้โดยเฉลี่ยนั้นเป็นค่าที่อยู่ในช่วงที่ยอมรับได้ดังนั้นจึงไม่มีการปรับปรุงการระบุค่าพิกัด จากผลการทดลองที่กล่าวมาข้างต้นสามารถสรุปได้ว่าสมมติฐานที่ 1 ไม่เป็นจริง

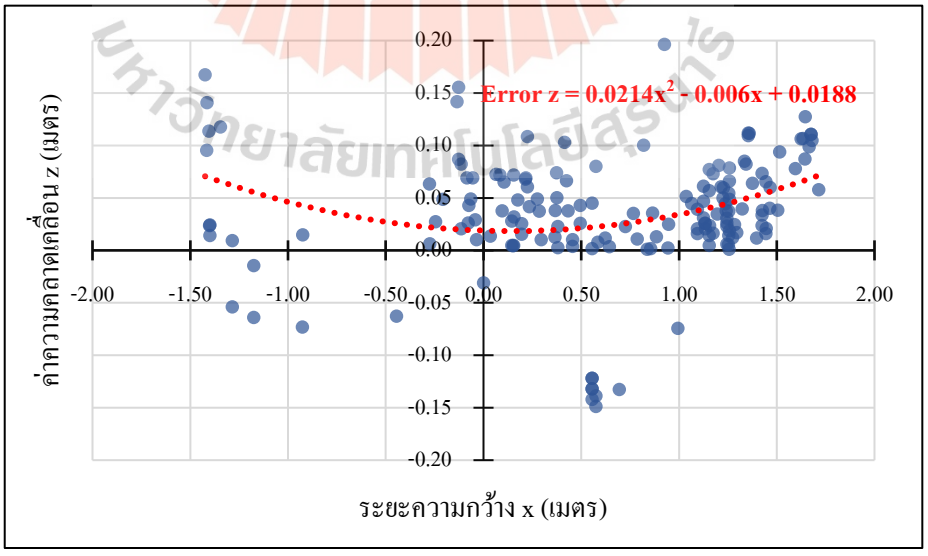
2. ตำแหน่งตามระยะแกน X มีผลต่อค่าตำแหน่งตามระยะแกน Z กล่าวคือ ค่าความถูกต้องของตำแหน่งในแกน Z จะเพิ่มขึ้นเมื่อตำแหน่งวัตถุห่างจากตำแหน่งศูนย์กลางของภาพไปทางขวาและซ้าย เนื่องจากความโค้งของเลนส์ภาพ

การทดลองตามสมมติฐานที่ 2 แสดงผลการทดลองดังรูปที่ 4.8 จากรูปเป็นกราฟแสดงผลระหว่างค่าความคลาดเคลื่อนของค่าความลึก (Z) และระยะความกว้าง (X) จากกราฟพบว่าเมื่อพิจารณาเส้นแนวโน้มของกราฟที่เกิดขึ้น ซึ่งเป็นแนวโน้มในลักษณะพาราโบร่าหงาย

ทำให้สามารถพิจารณาได้ว่าค่าความคลาดเคลื่อนของ Z จะมีแนวโน้มลดลงเมื่อค่า X มีค่าเข้าใกล้ 0 หรือเป็นบริเวณกึ่งกลางของกล้องและมีแนวโน้มเพิ่มขึ้นเมื่อค่า X มีค่าห่างจาก 0 ทั้งทางซ้ายและขวา โดยสามารถวิเคราะห์ได้ว่าเกิดขึ้นเนื่องจากความโค้งของเลนส์ภาพทำให้ค่าที่ไม่ได้อยู่บริเวณกึ่งกลางมีความคลาดเคลื่อนสูงตามสมมติฐาน



รูปที่ 4.7 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน y กับระยะ z



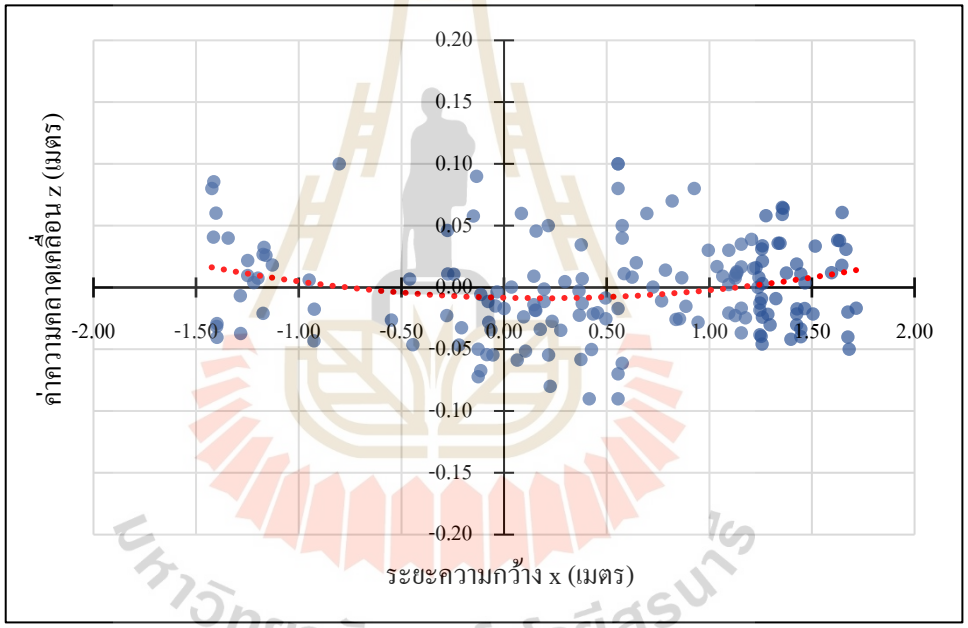
รูปที่ 4.8 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน z กับระยะ x

เมื่อผลลัพธ์ที่ได้จากการทดลองเป็นไปดังที่อธิบายข้างต้น ผู้วิจัยจึงได้ทำการปรับปรุงค่าด้วยการนำสมการที่ได้จากกราฟช่วยให้ผลลัพธ์ที่ได้ดียิ่งขึ้นและทำให้ค่าความคลาดเคลื่อนบริเวณที่ห่างจาก 0 ลดลง โดยสมการที่เป็นไปดังสมการที่ 4.1 และผลการทดลองหลังจากปรับปรุงการระบุค่าพิกัด แสดงดังรูปที่ 4.9

$$\text{Error } z = 0.0214x^2 - 0.006x + 0.0188 \quad (4.1)$$

เมื่อ Error z คือ ค่าความคลาดเคลื่อนของค่าความลึก

x คือ ค่าพิกัดตามแนวแกน x

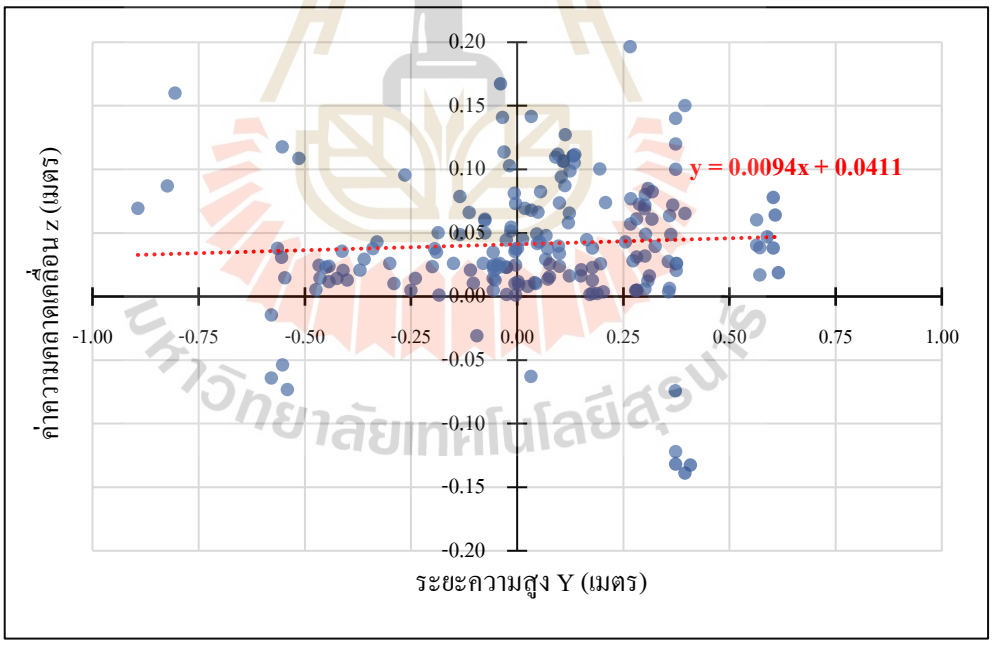


รูปที่ 4.9 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน z กับระยะ x หลังจากทำการปรับปรุงค่า

หลังจากปรับปรุงค่าพิกัดความลึกที่อ่านได้ด้วยสมการที่ 4.1 ได้ผลการทดสอบแสดงดังรูปที่ 4.9 จากกราฟพบว่าค่าความคลาดเคลื่อนของค่าความลึกในบริเวณที่ห่างจากตำแหน่งศูนย์กลางมีแนวโน้มลดลงอย่างเห็นได้ชัด นอกจากนั้นการปรับปรุงค่าในครั้งนี้ทำให้ค่าความคลาดเคลื่อนที่ได้ส่วนใหญ่มีค่าไม่เกิน 0.1 เมตร ซึ่งได้ผลลัพธ์เป็นที่น่าพอใจ

เมื่อการทดลองถูกต้องตามสมมติฐานที่ 2 ทำให้ผู้วิจัยได้เกิดข้อสงสัยว่าผลการทดลองที่ได้จากการพล็อทกราฟระหว่างค่าความคลาดเคลื่อน z และตำแหน่งความสูง y จะมีผลคล้ายกับการทดลองดังรูปที่ 4.8 ซึ่งเกิดเนื่องจากผลของความโค้งนูนของเลนส์หรือไม่ ดังนั้นผู้วิจัยจึงได้ทำการพิจารณาผลโดยการพล็อทกราฟระหว่างค่าความคลาดเคลื่อน z และตำแหน่งความสูง y ซึ่งแสดงผลดังรูปที่ 4.10

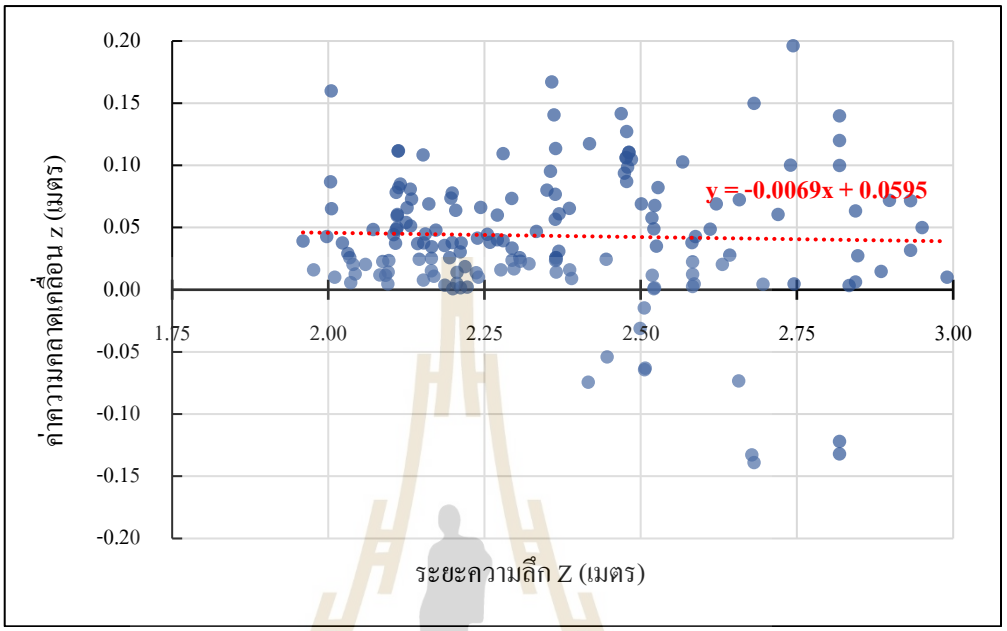
จากรูปที่ 4.10 สามารถพิจารณาได้ว่าแนวโน้มของกราฟเมื่อความสูงเพิ่มขึ้นมีแนวโน้มเพิ่มขึ้นเล็กน้อยโดยมีอัตราการเปลี่ยนแปลงอยู่ที่ประมาณ 0.0094 ซึ่งสามารถอนุมานได้ว่าความสูงที่เพิ่มขึ้นหรือลดลง ไม่ส่งผลต่อค่าความคลาดเคลื่อน z และแนวโน้มของกราฟที่ได้ไม่เป็นไปในทิศทางเดียวกับรูปที่ 4.8 ซึ่งผู้วิจัยวิเคราะห์ว่าสาเหตุที่แกน y ไม่ได้รับผลกระทบจากความโค้งของเลนส์เช่นเดียวกับแกน x เนื่องจากช่วงการทดลองตามแกน y อยู่ระหว่าง -1 ถึง 1 เมตร ซึ่งเป็นช่วงการทดลองที่แคบกว่าการทดลองตามแกน x จึงอาจส่งผลให้เส้นแนวโน้มที่มีลักษณะรูปพาราโบร่า ที่ค่าพิกัด z มีความคลาดเคลื่อนสูงเมื่ออยู่ห่างจากจุดศูนย์กลางนั้น ไม่มีแนวโน้มที่ชัดเจนปรากฏขึ้น



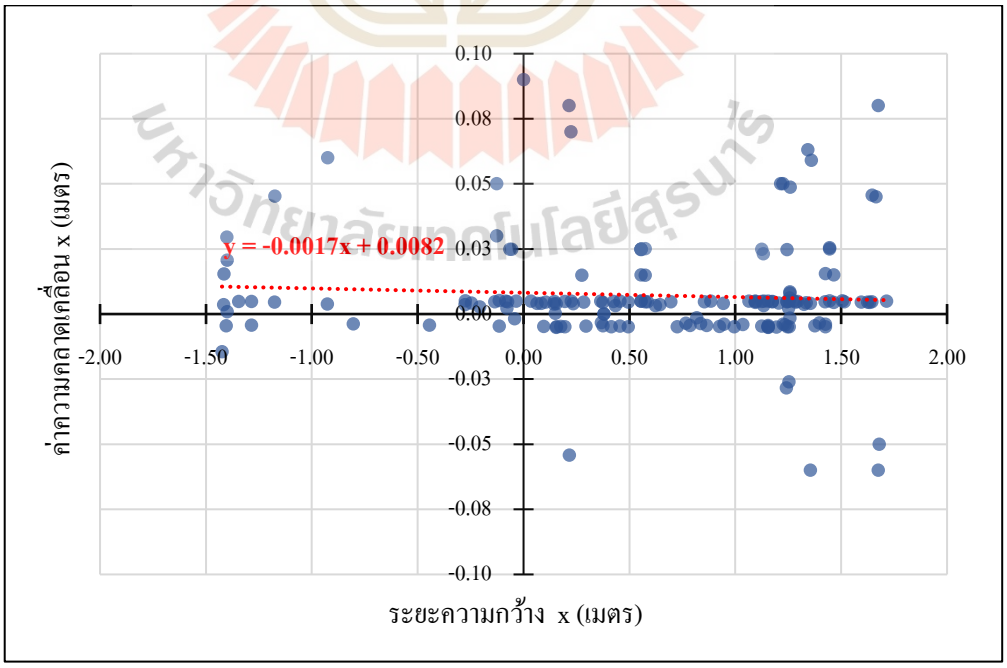
รูปที่ 4.10 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน z กับระยะ y

นอกจากนั้นผู้วิจัยได้ทำการพิจารณาผลความคลาดเคลื่อนของทั้ง 3 แกน เทียบกับพิกัดของทั้ง 3 แกน ซึ่งแนวโน้มทั้งหมดจะเป็นในทิศทางเดียวกันคือค่าความคลาดเคลื่อน

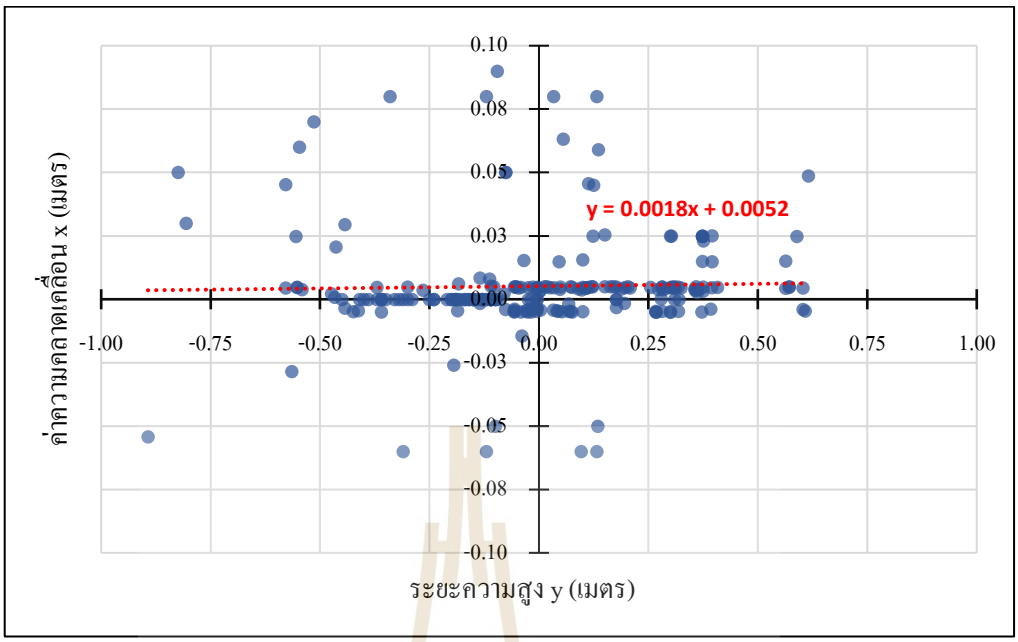
มีแนวโน้มเพิ่มขึ้นหรือลดลงเล็กน้อยซึ่งทั้งหมดนี้สามารถอนุมานได้ว่ามีแนวโน้มที่คงที่ ซึ่งผลการทดลองเป็นไปดังรูปที่ 4.11 ถึงรูปที่ 4.17 ดังนี้



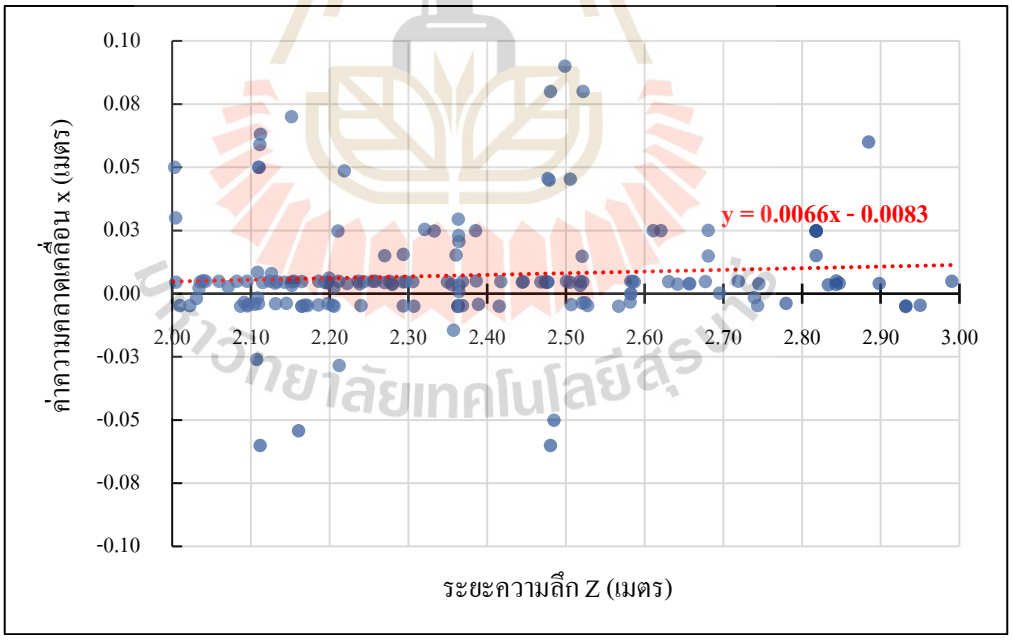
รูปที่ 4.11 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน z กับระยะ z



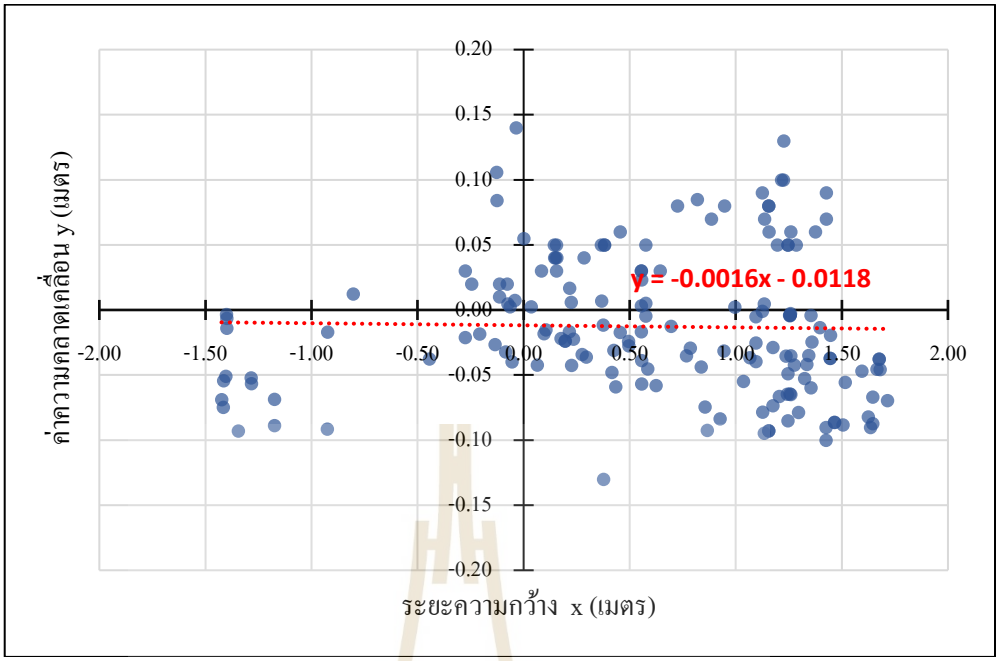
รูปที่ 4.12 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน x กับระยะ y



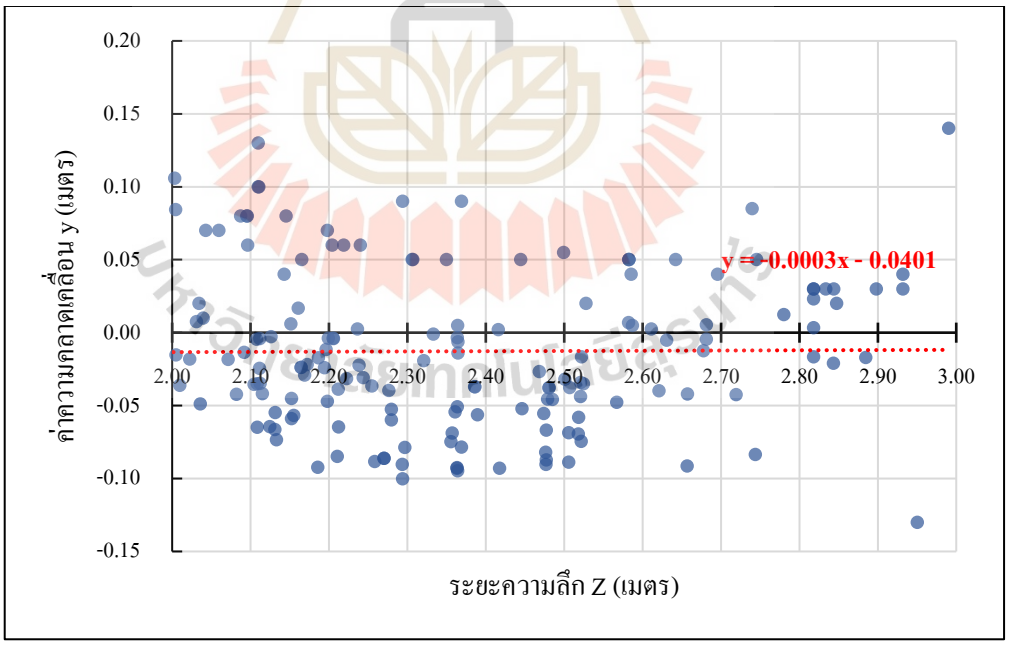
รูปที่ 4.13 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน x กับระยะ y



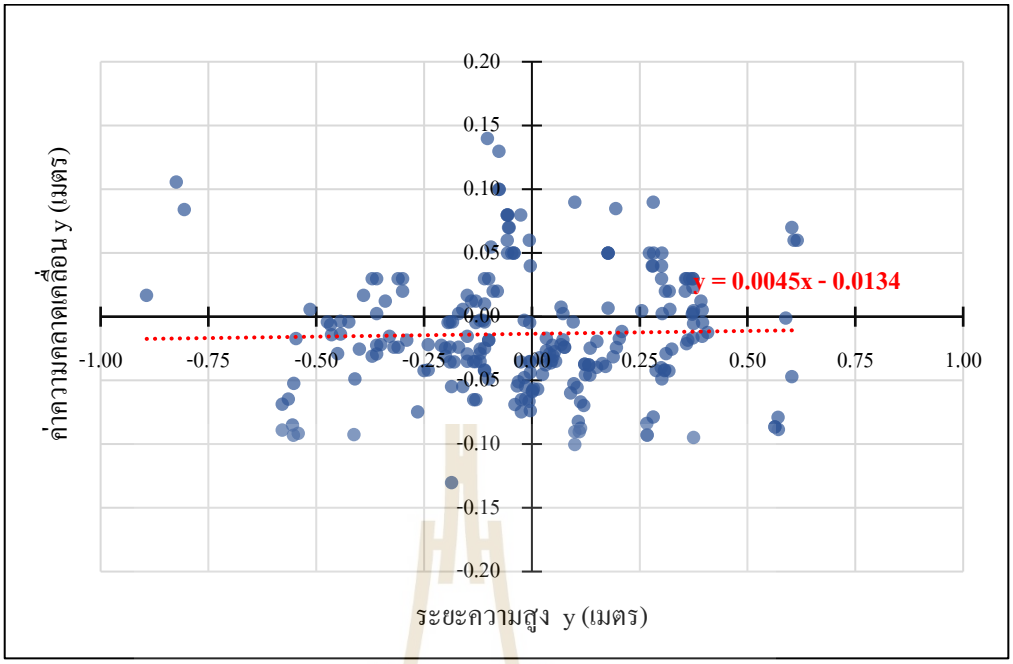
รูปที่ 4.14 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน x กับระยะ z



รูปที่ 4.15 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน y กับระยะ x



รูปที่ 4.16 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน y กับระยะ z

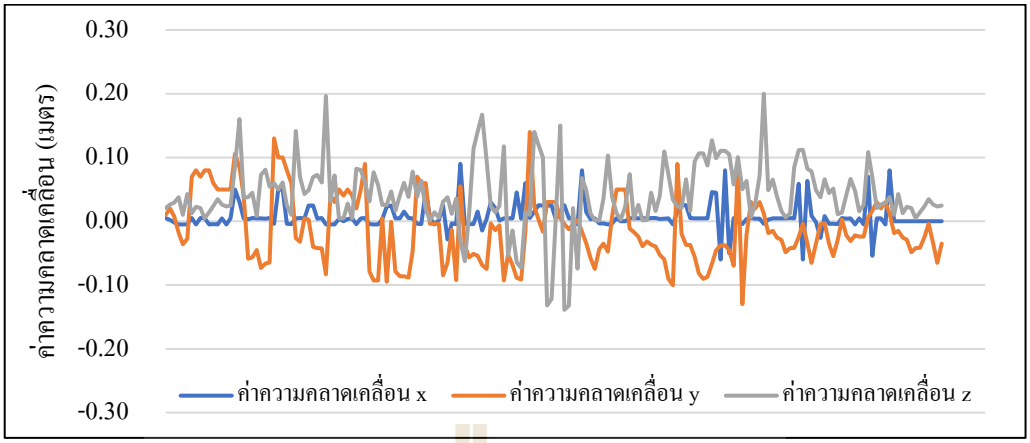


รูปที่ 4.17 กราฟแสดงความสัมพันธ์ระหว่างค่าความคลาดเคลื่อน y กับระยะ y

จากผลการทดลองข้างต้นสามารถพิจารณาได้ว่า ตำแหน่งพิกัดใด ๆ ไม่ส่งผลต่อค่าความคลาดเคลื่อนของความกว้าง (x) และความสูง (y) ซึ่งผู้วิจัยวิเคราะห์ว่าเกิดจากกระบวนการเตรียมรูปภาพในขั้นตอน rectification ก่อนเข้ากระบวนการคำนวณค่าของกล้อง ZED ดังทฤษฎีหัวข้อที่ 2.4.2 จึงทำให้ช่วยลดผลความคลาดเคลื่อนในแกนระนาบ x และ y ได้

4.2.3 ผลการทดสอบค่าความผันผวน

จากการทดลองทดสอบค่าความผันผวนพบว่าค่าเฉลี่ยของความคลาดเคลื่อนของค่า x อยู่ที่ 0.01 เมตร มีค่าความคลาดเคลื่อนสูงสุดอยู่ที่ 0.09 เมตร โดยมีค่าเบี่ยงเบนมาตรฐานอยู่ที่ 0.02 ค่าเฉลี่ยของความคลาดเคลื่อนของค่า y อยู่ที่ -0.01 เมตร มีค่าความคลาดเคลื่อนสูงสุดอยู่ที่ 0.14 เมตร ค่าเบี่ยงเบนมาตรฐานอยู่ที่ 0.05 และค่าเฉลี่ยของความคลาดเคลื่อนของค่า z อยู่ที่ 0.04 เมตร มีค่าความคลาดเคลื่อนสูงสุดอยู่ที่ 0.20 เมตร ส่วนค่าเบี่ยงเบนมาตรฐานมีค่าประมาณ 0.05 ซึ่งสามารถแสดงผลสรุปได้ดังรูปที่ 4.18



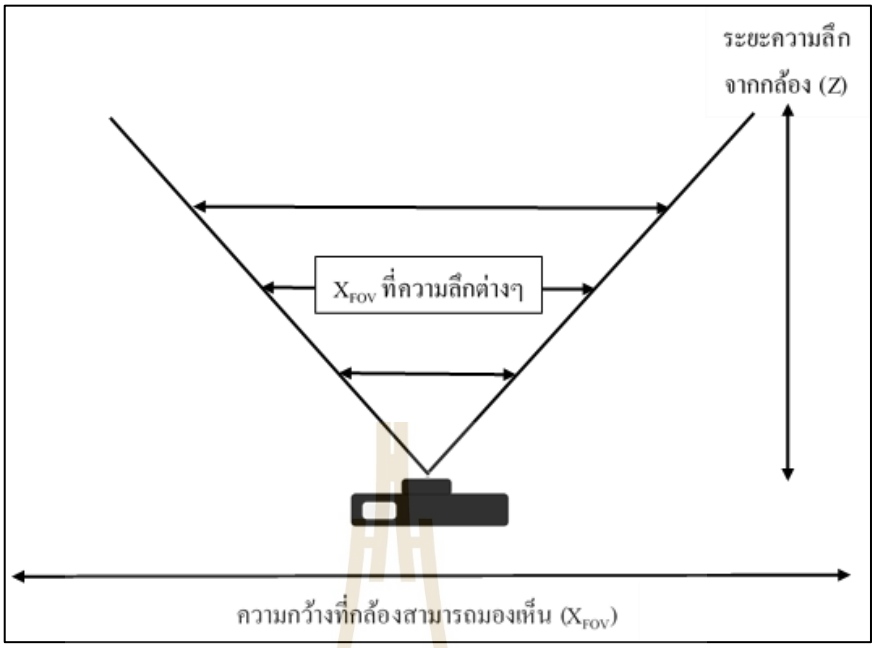
รูปที่ 4.18 กราฟแสดงผลสรุปโดยการเปรียบเทียบค่าความคลาดเคลื่อนของทั้ง 3 แกน

เมื่อนำข้อมูลที่ได้คำนวณหาค่าความแปรปรวน (Variance) พบว่าค่าพิกัด y มีความแปรปรวนสูงสุดโดยมีค่าอยู่ที่ 0.0027 รองลงมาคือค่าพิกัด z ที่มีค่าความแปรปรวนประมาณ 0.0026 และค่าพิกัด x มีค่าความแปรปรวนน้อยที่สุดอยู่ที่ 0.0004 จากค่าความแปรปรวนที่ได้นั้นสามารถวิเคราะห์ได้ว่าระบบการระบุพิกัดมีความแปรปรวนที่น้อยมาก ค่าพิกัดที่ได้จึงมีความเที่ยงตรงสูง

4.3 ผลการทดลองการพัฒนาระบบนำทาง

4.3.1 ผลการทดสอบขอบเขตการมองเห็นของกล้อง

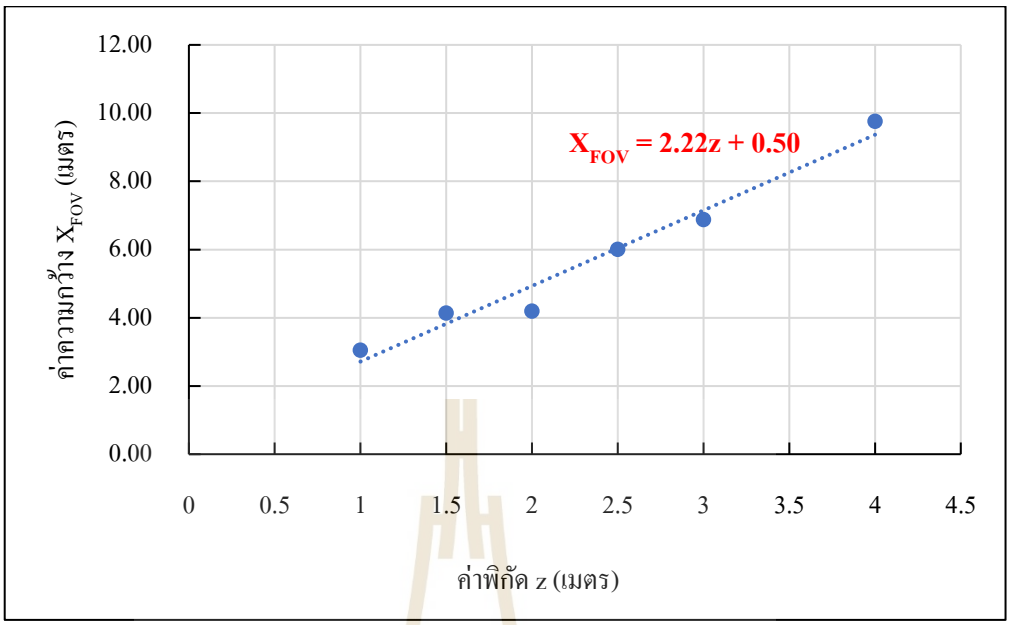
จากการทดสอบเพื่อหาขอบเขตความสามารถในการมองเห็นของกล้อง ซึ่งการทำการทดสอบสามารถจำลองได้ดังรูป 4.19 เมื่อทำการทดสอบครบทั้ง 3 ครั้ง ได้ข้อมูลจากการทดสอบแสดงดังตารางที่ 4.3 และทำการคำนวณค่าเฉลี่ยจากการทดลองทั้ง 3 ครั้ง จากนั้นจึงนำข้อมูลที่เฉลี่ยแล้วพล็อทกราฟระหว่างค่าความกว้าง x และพิกัด z ดังรูปที่ 4.20 เพื่อทำการพิจารณาหาสมการสำหรับกำหนดของเขตการบินให้อยู่ในพื้นที่การมองเห็นของกล้อง



รูปที่ 4.19 รูปจำลองการทดสอบหาค่า X_{FOV} ที่ระยะความลึกต่าง ๆ

ตารางที่ 4.3 ตารางแสดงข้อมูลการทดสอบขอบเขตการมองเห็นของกล้องถ่ายภาพ 3 มิติ ZED

ค่าพิกัด z (เมตร)	**ค่าความกว้าง X_{FOV} (เมตร)**			
	1	**2**	**3**	**เฉลี่ย**
1	3.05	3.03	3.07	3.05
1.5	4.14	4.15	4.13	4.14
2	4.2	4.2	4.19	4.20
2.5	6.00	6.00	6.01	6.00
3	6.88	6.9	6.86	6.88
4	9.76	9.75	9.76	9.76



รูปที่ 4.20 กราฟแสดงความสัมพันธ์ระหว่างค่าความกว้าง X_{FOV} และค่าพิกัด z (เมตร) สำหรับพิจารณาขอบเขตการมองเห็นของกล้องถ่ายภาพ 3 มิติ ZED

จากรูปที่ 4.20 สามารถพิจารณาได้ว่าเมื่อค่าพิกัด z เพิ่มขึ้น ความกว้างในการมองเห็นสำหรับแกน x จะมีค่าเพิ่มมากขึ้น สามารถกล่าวได้ว่าเมื่อวัตถุอยู่ห่างจากกล้องมากขึ้นจะสามารถเคลื่อนที่ในแนวแกน x โดยที่อยู่ในขอบเขตการมองเห็นของกล้องได้มากขึ้น ซึ่งการกำหนดขอบเขตการมองเห็นในโปรแกรมทำโดยการเพิ่มสมการที่ 4.2 ช่วยให้การกำหนดขอบเขตครอบคลุมมากยิ่งขึ้น

$$X_{FOV} = 2.22z + 0.50 \quad (4.2)$$

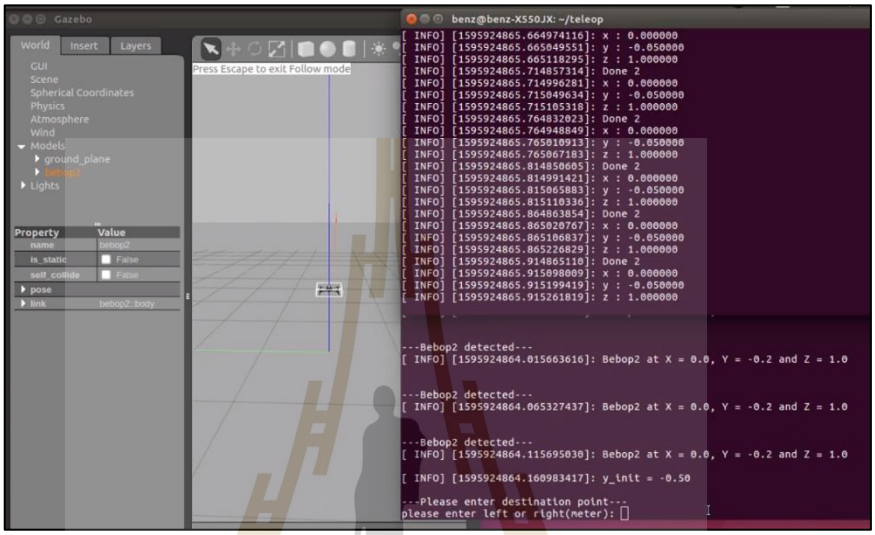
เมื่อ X_{FOV} คือ ความกว้างในการมองเห็นของกล้องในแกน x

z คือ พิกัดความลึกในแกน z

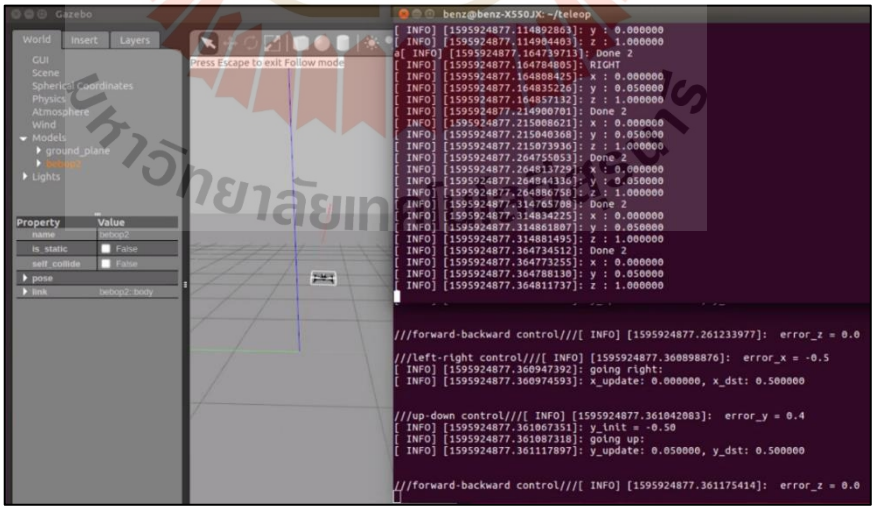
4.3.2 ผลการทดสอบอัลกอริทึมด้วยระบบจำลอง

ในการทดสอบอัลกอริทึมควบคุมโดรนด้วยระบบจำลอง เริ่มทำการทดสอบตั้งแต่การส่งคำสั่ง Take off จนกระทั่ง Landing โดยการทดสอบแบ่งเป็น 5 การทดสอบย่อย ได้แก่ ทดสอบการขึ้นบิน ดังรูปที่ 4.21 การเคลื่อนที่หน้า-หลัง การเคลื่อนที่ซ้าย-ขวา การเคลื่อนที่ขึ้น-ลง

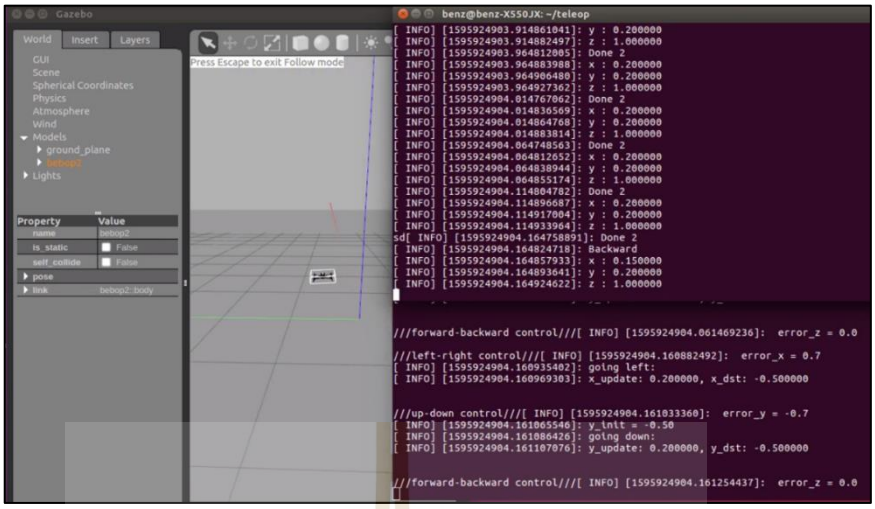
ดังรูปที่ 4.22 และ 4.23 และการลงจอดเมื่อจบกระบวณการ ดังรูปที่ 4.24 เมื่อทำการพิจารณาผลการทดสอบที่ได้จากการทดสอบด้วยระบบจำลองพบว่า โดรนสามารถเคลื่อนที่ไปยังทิศทางทั้ง 6 ได้ถูกต้อง รวมทั้งอัลกอริทึมที่ใช้ในการตรวจสอบและสั่งการขึ้นบินและลงจอดก็สามารถทำงานได้ถูกต้องตามที่กำหนด



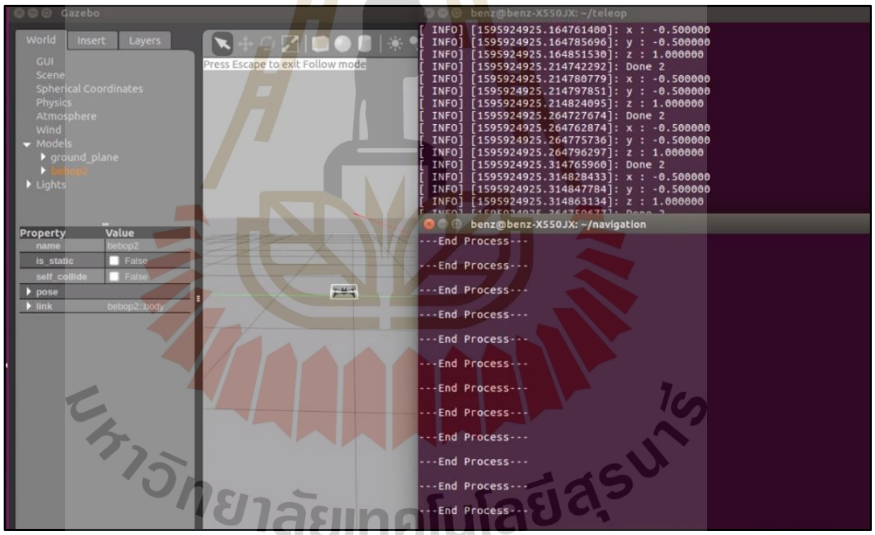
รูปที่ 4.21 รูปแสดงผลการทดสอบคำสั่งการขึ้นบิน (Take off)



รูปที่ 4.22 รูปแสดงผลการทดสอบคำสั่งการเคลื่อนที่ไปทางขวาและเพิ่มความสูง



รูปที่ 4.23 รูปแสดงผลการทดสอบคำสั่งการเคลื่อนที่ไปทางซ้ายและลดความสูง



รูปที่ 4.24 รูปแสดงผลการทดสอบคำสั่งการลงจอด (Landing)

4.3.3 ผลการทดลองหาค่าตัวควบคุม PID

ในการทดลองปรับค่าตัวควบคุม PID ใช้วิธี Trial and error ในการหาค่าทดลองที่เหมาะสม โดยเริ่มจากการพิจารณาปรับค่า k_p ของทั้ง 3 แกนเพื่อพิจารณาปฏิกิริยาการตอบสนองของโดรน หลังจากนั้นจึงทำการปรับเพิ่มค่า k_I และ k_D ตามลำดับ การพิจารณาปรับค่า k_p เนื่องจากผู้วิจัยมีประสบการณ์ในการทดลองปรับค่า k_p ของโดรน Bebop ในงานวิจัยก่อนหน้าจงทำให้ผู้วิจัยมีค่า k_p

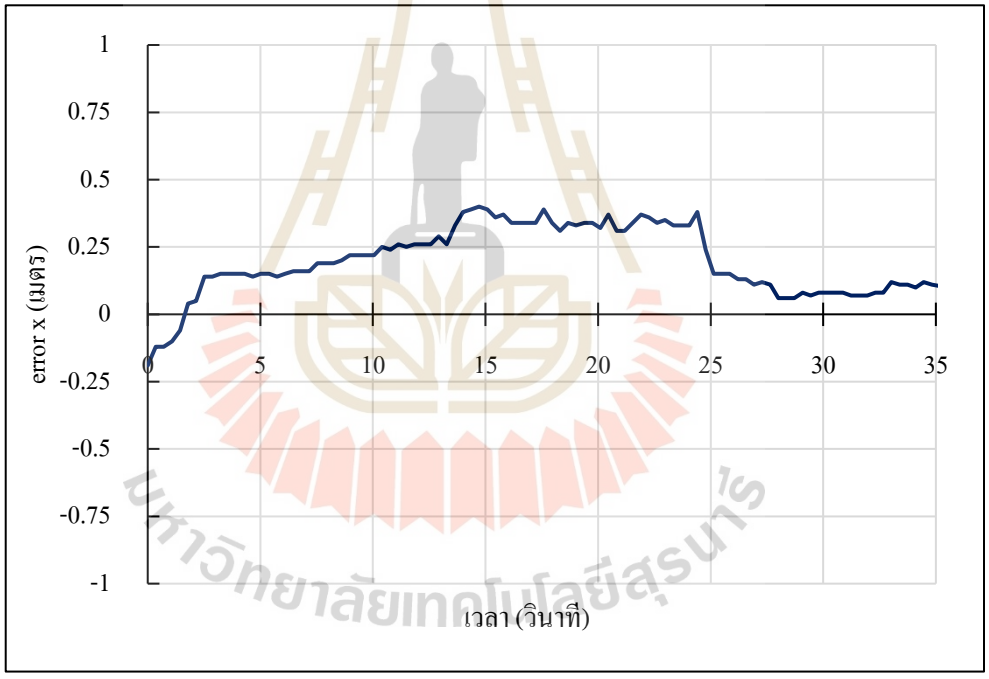
ตั้งต้น สำหรับการทดสอบอยู่ที่ 0.0048 จึงใช้ค่าดังกล่าวเป็นค่าตั้งต้น แต่เมื่อทำการทดสอบแล้วพบว่าค่า k_p ที่เคยใช้กับงานวิจัยก่อนหน้ามีค่าน้อยเกินไปโดรนจึงไม่มีปฏิกิริยาตอบสนอง ผู้วิจัยจึงได้ทำการเพิ่มค่า k_p ครั้งละ 0.005 และพบว่าโดรนเริ่มมีปฏิกิริยาตอบสนองที่ค่า k_p เท่ากับ 0.03 และเริ่มทำการแกว่งที่ค่าประมาณ 0.05 ผู้วิจัยจึงกำหนดช่วงทดสอบค่า k_p อยู่ระหว่าง 0.03 และ 0.05 ซึ่งเป็นช่วงที่เหมาะสมสำหรับการเคลื่อนที่ในแนวแกน x สำหรับการทดสอบในแนวแกน z ค่า k_p อยู่ระหว่าง 0.03 และ 0.045 สำหรับแกน y ไม่มีการตอบสนองในช่วงค่าดังกล่าว ผู้วัจัยจึงทำการเพิ่มค่า k_p ของแกน y ครั้งละ 0.005 และพบว่าโดรนเริ่มมีปฏิกิริยาตอบสนองเมื่อค่าเท่ากับ 0.095 และเริ่มมีการแกว่งเมื่อค่าเท่ากับ 0.15 ดังนั้นผู้วิจัยจึงเลือกช่วงทดลองค่า k_p ของแกน y อยู่ระหว่าง 0.095 ถึง 0.15 โดยการทดลอง เป็นไปดังตารางที่ 4.4 ดังนี้

ตารางที่ 4.4 ตารางแสดงการปรับค่า k_p สำหรับควบคุมการเคลื่อนที่ในแกน x, y และ z

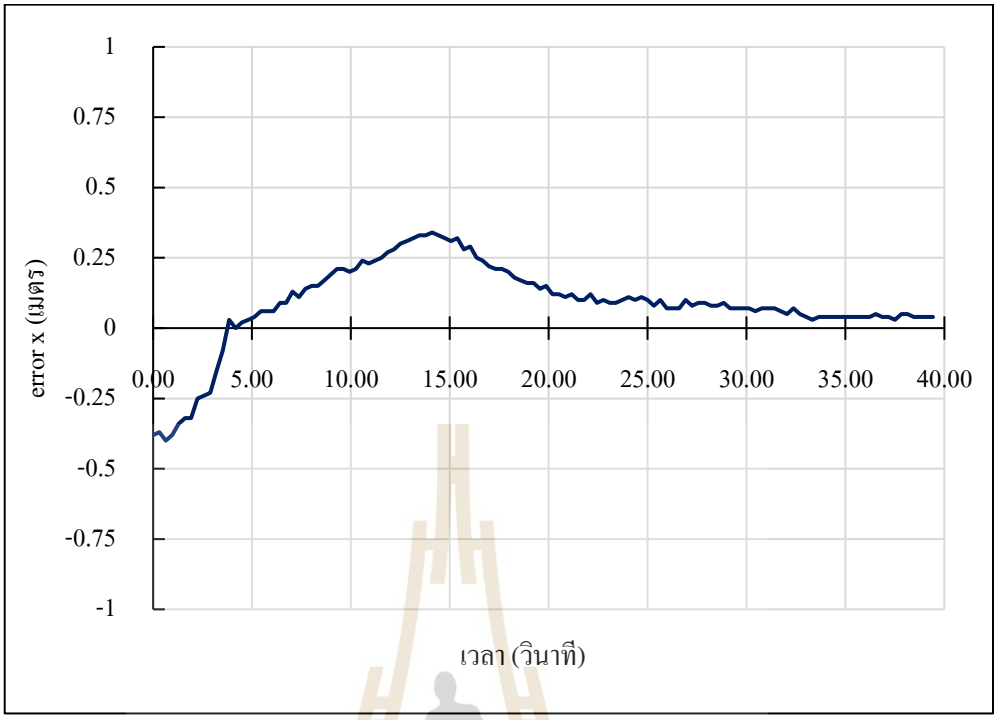
การทดลองครั้งที่	**ค่า k_p**		
	X	**Y**	**Z**
1	0.03	0	0
2	0.035	0	0
3	0.04	0	0
4	0.045	0	0
5	0.05	0	0
6	0	0.095	0
7	0	0.1	0
8	0	0.15	0
9	0	0	0.03
10	0	0	0.035
11	0	0	0.04
12	0	0	0.045

การทดลองครั้งที่ 1 ถึง 5 เป็นการทดลองเพื่อหาค่า k_p ของการเคลื่อนที่ในแกน x ซึ่งผลการทดลองที่ 1 และ 2 พบว่า การตอบสนองของโดรนให้เข้าสู่พิกัดเป้าหมายนั้นทำได้ค่อนข้างดีกล่าวคือโดรนใช้ระยะเวลาประมาณ 40 วินาที ในการปรับแก้ให้เคลื่อนที่ให้เข้าสู่เป้าหมายและมีค่าโอเวอร์ชูตประมาณ 0.35 เมตร ทั้ง 2 การทดลอง แต่จากการวิเคราะห์รูปที่ 4.25

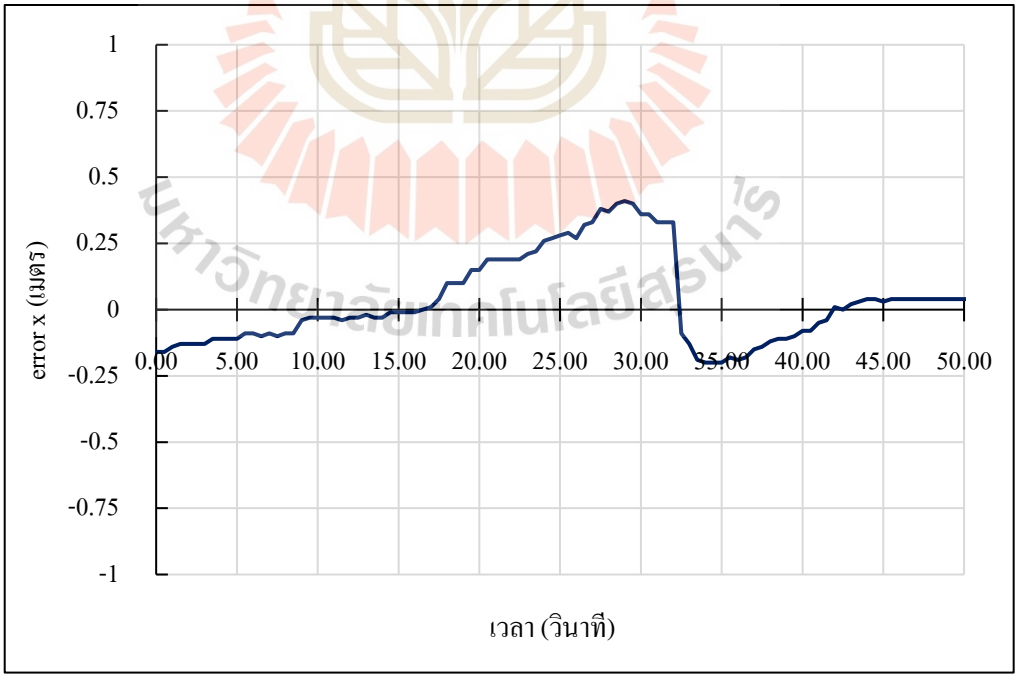
และรูปที่ 4.26 พบว่า ในการทดลองที่ 2 การปรับแก้การเคลื่อนที่เร็วกว่าการทดลองที่ 1 โดยการทดลองที่ 2 เริ่มปรับแก้ที่เวลา 15 วินาที ส่วนการทดลองที่ 1 เริ่มทำการปรับแก้ที่เวลา 20-25 วินาที โดยประมาณ ในส่วนการทดลองที่ 3, 4 และ 5 จากการพิจารณาพบว่าการทดลองทั้ง 3 มีค่าโอเวอร์ชูตเกิดขึ้น โดยที่ผลการทดลองที่ 3 แสดงดังรูปที่ 4.27 พบว่า เริ่มเกิดโอเวอร์ชูตมีค่าประมาณ 0.4 เมตร และเกิดโอเวอร์ชูตอีกครั้งที่เวลาประมาณ 35 วินาที มีค่าประมาณ -0.2 เมตร การทดลองที่ 4 โอเวอร์ชูตเพิ่มสูงขึ้นโดยค่าสูงสุดประมาณ 0.75 เมตร และโดรนใช้เวลาในการปรับแก้เข้าสู่พิกัดเป้าหมายประมาณ 100 วินาที ผลการทดลองครั้งที่ 4 แสดงดังรูปที่ 4.28 และการทดลองครั้งที่ 5 โดรนใช้เวลาในการปรับแก้ประมาณ 300 วินาที เนื่องจากค่า k_p ที่ใช้ในการทดลองครั้งที่ 5 ส่งผลให้โดรนเกิดการแกว่งทำให้ใช้เวลานานในการปรับแก้เข้าสู่จุดพิกัดเป้าหมาย ผลการทดลองครั้งที่ 5 แสดงดังรูปที่ 4.29



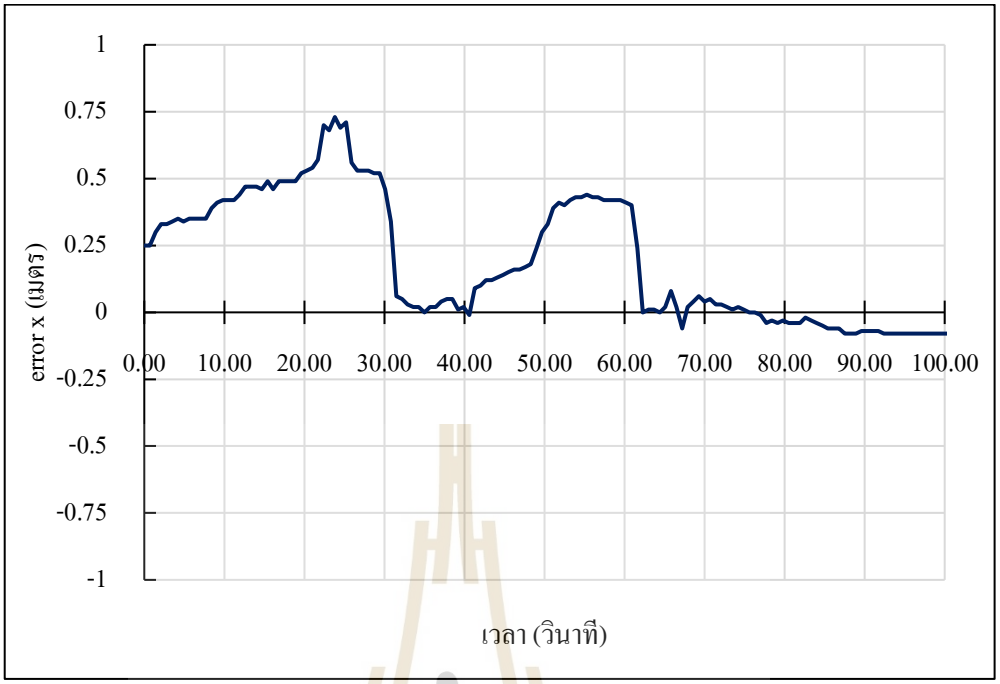
รูปที่ 4.25 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน x เมื่อค่า k_p เท่ากับ 0.03



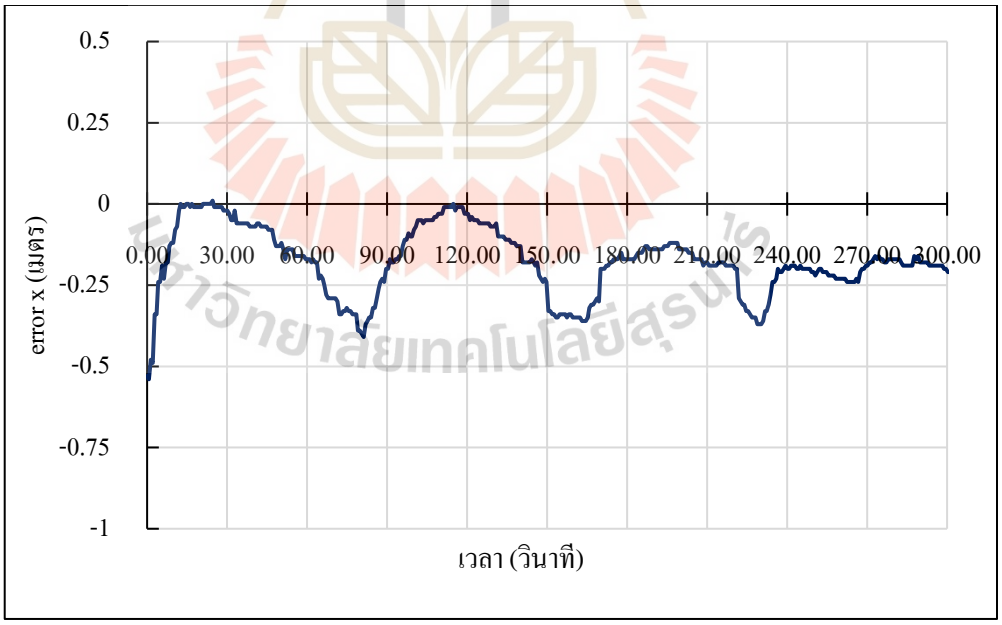
รูปที่ 4.26 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน x เมื่อค่า k_p เท่ากับ 0.035



รูปที่ 4.27 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน x เมื่อค่า k_p เท่ากับ 0.04

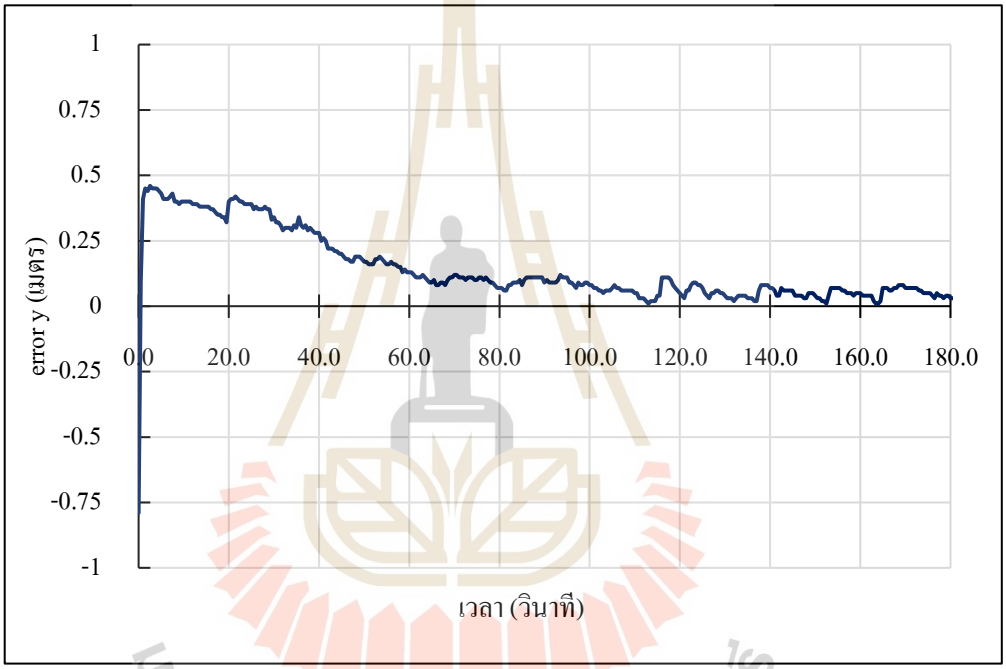


รูปที่ 4.28 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน x เมื่อค่า k_p เท่ากับ 0.045

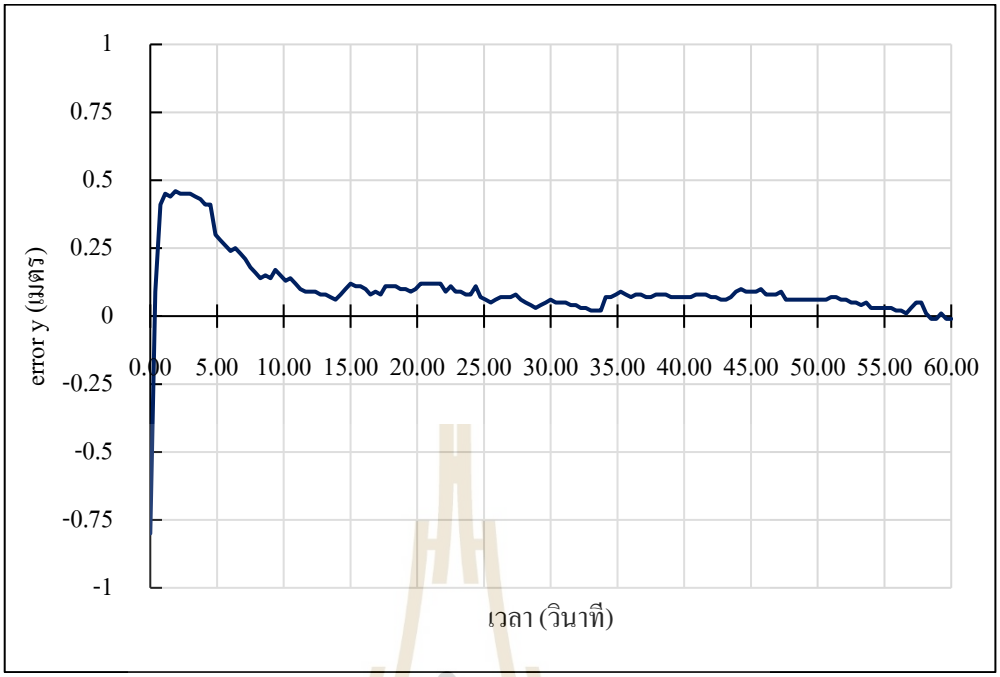


รูปที่ 4.29 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน x เมื่อค่า k_p เท่ากับ 0.05

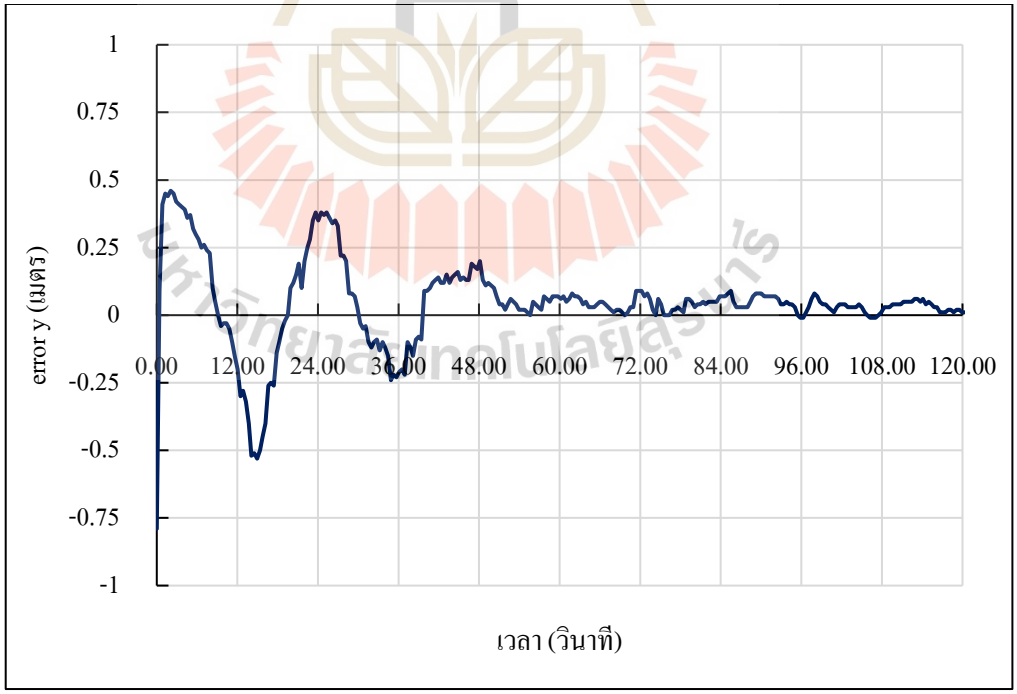
การทดลองครั้งที่ 6, 7 และ 8 เป็นการทดลองเพื่อหาค่า k_p ที่เหมาะสมสำหรับการควบคุมการเคลื่อนที่ตามแนวแกน y จากการทดลองพบว่าการทดลองที่ 6 และ 7 โดรนเคลื่อนที่เข้าสู่เป้าหมายได้โดยไม่เกิดโอเวอร์ชูต โดยการทดลองครั้งที่ 6 ผลการทดลองแสดงดังรูปที่ 4.30 ใช้เวลาเข้าสู่พิกัดเป้าหมายประมาณ 100 วินาที ส่วนการทดลองครั้งที่ 7 ผลการทดลองแสดงดังรูปที่ 4.31 ใช้เวลาประมาณ 55 วินาที ซึ่งเร็วกว่าการทดลองที่ 6 สำหรับการทดลองครั้งที่ 8 พบว่าโดรนใช้เวลาในการเข้าสู่พิกัดเป้าหมายประมาณ 55 วินาที แต่โดรนเกิดการแกว่งรุนแรงกว่าการทดลองที่ 6 และ 7 โดยผลการทดลองที่ 8 แสดงดังรูปที่ 4.32



รูปที่ 4.30 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน y เมื่อค่า k_p เท่ากับ 0.095

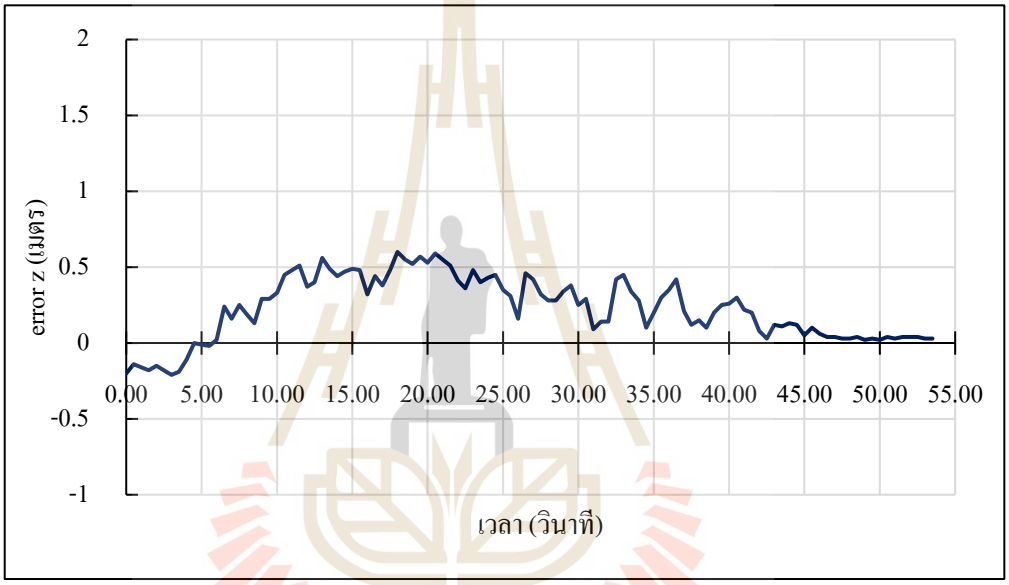


รูปที่ 4.31 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน y เมื่อค่า k_p เท่ากับ 0.1

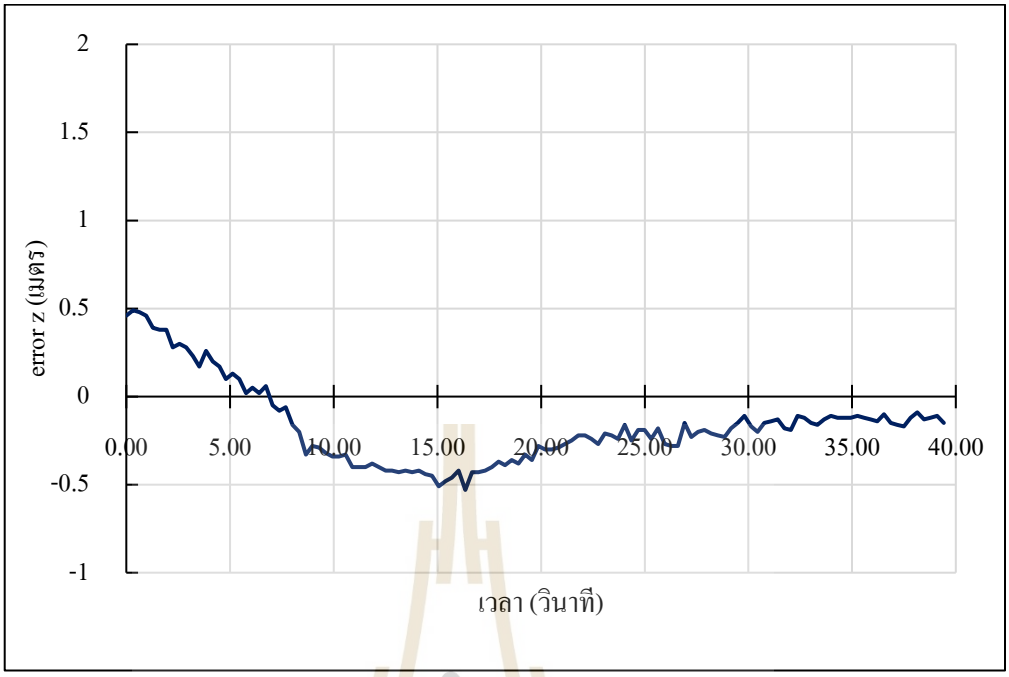


รูปที่ 4.32 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน y เมื่อค่า k_p เท่ากับ 0.15

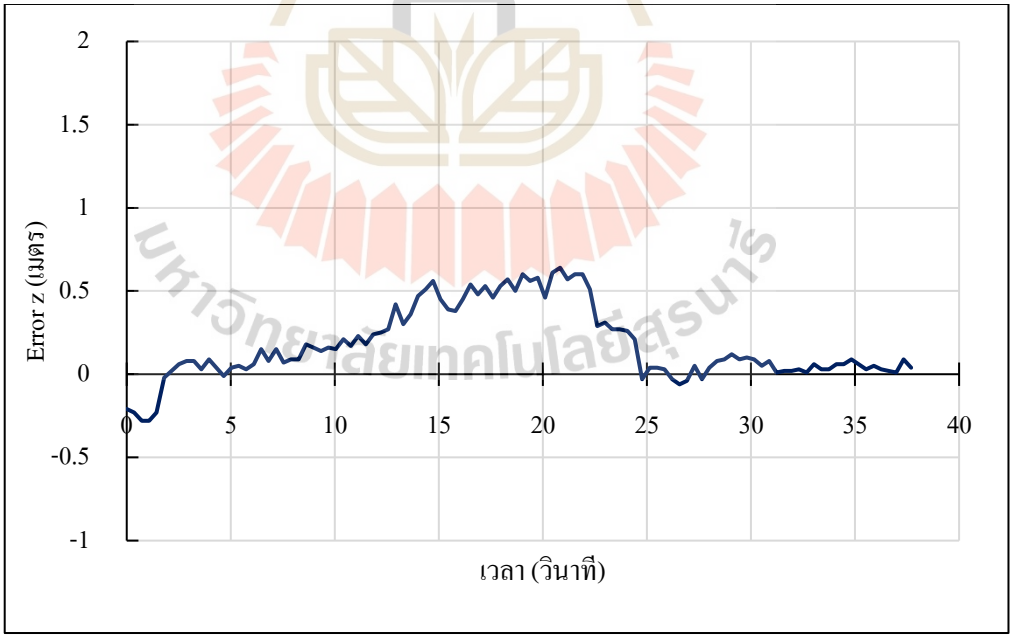
การทดลองที่ 9 ถึง 12 เป็นการทดลองเพื่อหาค่า k_p ที่เหมาะสมสำหรับการควบคุมการเคลื่อนที่ของโดรนโดยการทดลองที่ 9, 10 และ 11 เกิดโอเวอร์ชูตที่ใกล้เคียงกัน โดยมีค่าประมาณ 0.5 เมตร การทดลองที่ 9 ใช้เวลาเข้าสู่พิกัดเป้าหมายที่เวลาประมาณ 45 วินาที ผลการทดลองแสดงดังรูปที่ 4.33 การทดลองที่ 10 ใช้เวลาประมาณ 40 วินาที ผลการทดลองแสดงดังรูปที่ 4.34 และการทดลองที่ 11 ใช้เวลาประมาณ 25 วินาที ผลการทดลองดังรูปที่ 4.35 การทดลองที่ 12 ค่า k_p ส่งผลให้โดรนมีเสถียรภาพลดลงและเกิดการแกว่งสูง โดยผลการทดลองแสดงดังรูปที่ 4.36



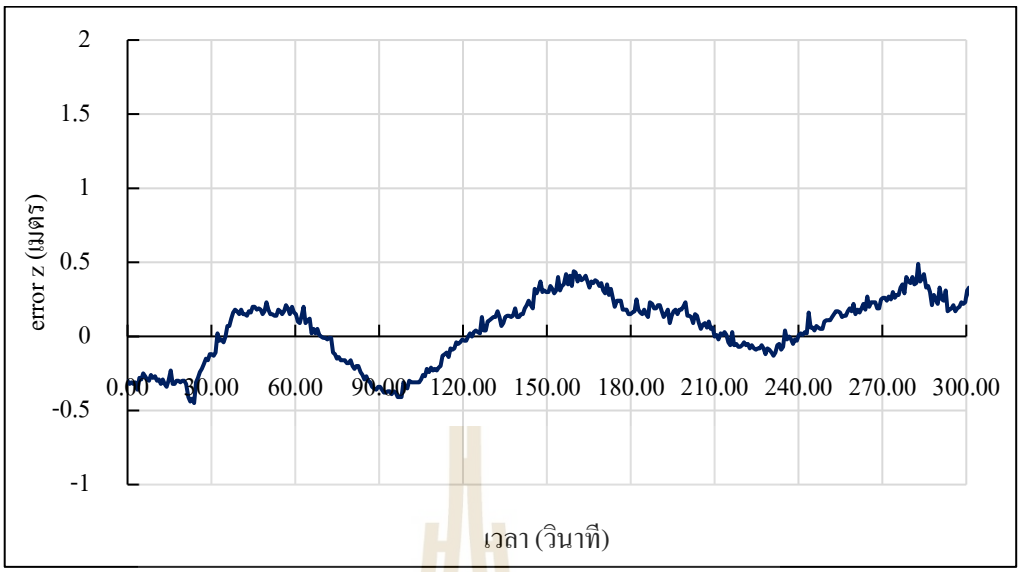
รูปที่ 4.33 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน z เมื่อค่า k_p เท่ากับ 0.03



รูปที่ 4.34 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน z เมื่อค่า k_p เท่ากับ 0.035



รูปที่ 4.35 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน z เมื่อค่า k_p เท่ากับ 0.04



รูปที่ 4.36 ผลการทดลองการตอบสนองของการเคลื่อนที่ตามแกน z เมื่อค่า k_p เท่ากับ 0.045

การควบคุมการเคลื่อนที่ของโดรนเน้นการควบคุมการเคลื่อนที่อย่างรวดเร็วในช่วง transient response เป็นหลัก อีกทั้งระบบมีช่วงค่าความคลาดเคลื่อนที่สามารถยอมรับได้ ดังนั้นผู้วิจัยจึงตัดการควบคุมด้วยค่า k_I ที่ส่งผลต่อการควบคุมในช่วง Steady state ออก เพื่อลดจำนวนในการทดลอง และทำการทดสอบค่า k_D ที่เหมาะสมกับระบบ แต่เมื่อทดสอบแล้วพบว่าเมื่อเพิ่มค่า k_D ในการทดลองส่งผลให้ระบบการควบคุมโดรนมีเสถียรภาพในการทำงานลดลง เมื่อเทียบกับการควบคุมด้วย k_p เพียงอย่างเดียว ดังนั้นผู้วิจัยจึงไม่ทำการควบคุมการเคลื่อนที่ของโดรนด้วยค่า k_D และเลือกใช้การควบคุมด้วยค่า k_p เพียงอย่างเดียวโดยค่า k_p ของการเคลื่อนที่แกน x, y และ z คือ 0.035, 0.1 และ 0.04 ตามลำดับ

4.3.4 ผลการทดสอบระบบนำทาง

เมื่อได้ค่าตัวควบคุมการเคลื่อนที่ที่เหมาะสม ผู้วิจัยจึงได้ทำการทดลองระบบควบคุมการเคลื่อนที่แบบเต็มรูปแบบ ผู้วิจัยได้ทำการกำหนดจุดหมายให้แก่โดรนทั้งหมด 4 จุด โดยในเบื้องต้นเป็นการกำหนดจุดหมายที่ระยะห่างประมาณ 0.5 เมตร และมีระยะทางโดยรวมทั้งสิ้นประมาณ 2 เมตร ซึ่งผลปรากฏว่าโดรนสามารถทำภารกิจเคลื่อนที่ไปยังจุดหมายทั้ง 4 จุด ได้ด้วยเวลา 38.40 วินาที หากพิจารณาตามรูปที่ 4.37 ซึ่งแสดงการเคลื่อนที่แบบ 2 มิติของโดรน เมื่อมองจากมุมมองด้านบน (Top view) พบว่า กราฟมีการแกว่งระหว่างทางเล็กน้อยซึ่งเกิดจากการแกว่งในการตรวจจับของกล้อง และเนื่องจากสเกลมีขนาดละเอียดจึงทำให้มองเห็นค่าการแกว่งนี้ชัดเจน ซึ่งหากพิจารณาแล้วจะพบว่าค่าความคลาดเคลื่อนนี้มีค่าไม่เกิน 0.10 เมตร ซึ่งมีค่าน้อยมาก

และอยู่ในขอบเขตที่ยอมรับได้ รวมทั้งการเคลื่อนที่ตามแกน y ดังรูปที่ 4.38 ซึ่งสามารถควบคุมได้ดี และมีค่าความคลาดเคลื่อนที่ไม่เกิน 0.10 เมตร เช่นกัน

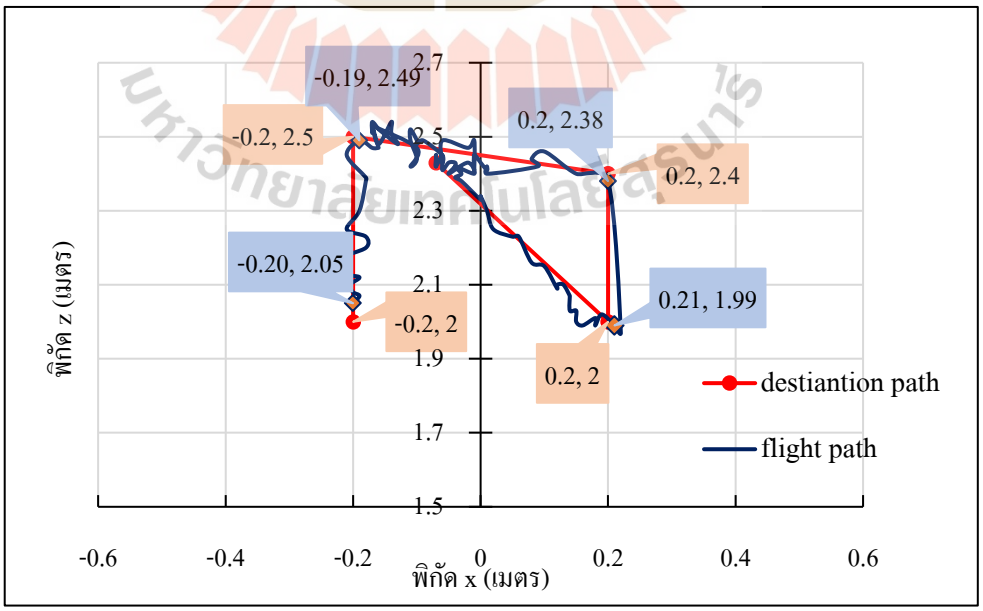
หลังจากทำการทดสอบการเคลื่อนที่ระยะใกล้แล้ว ผู้วิจัยจึงได้ทำการทดลองเพิ่มระยะของพิกัดเป้าหมายแต่เนื่องจากระหว่างการทดลองได้เกิดอุบัติเหตุเนื่องจากโดรนขาดการเชื่อมต่อกับคอมพิวเตอร์ควบคุมทำให้โดรนชนผนังและมอเตอร์เกิดความเสียหายทำให้ผู้วิจัยไม่สามารถทำการวิจัยต่อในส่วนนี้ได้ แต่อย่างไรก็ตามงานวิจัยนี้ก็ได้บรรลุวัตถุประสงค์ในการควบคุมการเคลื่อนที่ของโดรนภายในอาคารด้วยระบบระบุพิกัดด้วยกล้องสามมิติ ซึ่งผลการทดลองได้แสดงดังที่กล่าวข้างต้น

จากการทดสอบระบบนำทางทั้งหมดประมาณ 30 ครั้ง พบว่า ระบบสามารถควบคุมโดรนให้เข้าสู่เป้าหมายจนภารกิจสำเร็จทั้งหมด 22 ครั้ง คิดเป็นร้อยละ 73.33 โดยสาเหตุที่ทำให้เกิดความผิดพลาดของระบบ ผู้วิจัยได้วิเคราะห์สาเหตุ ดังนี้

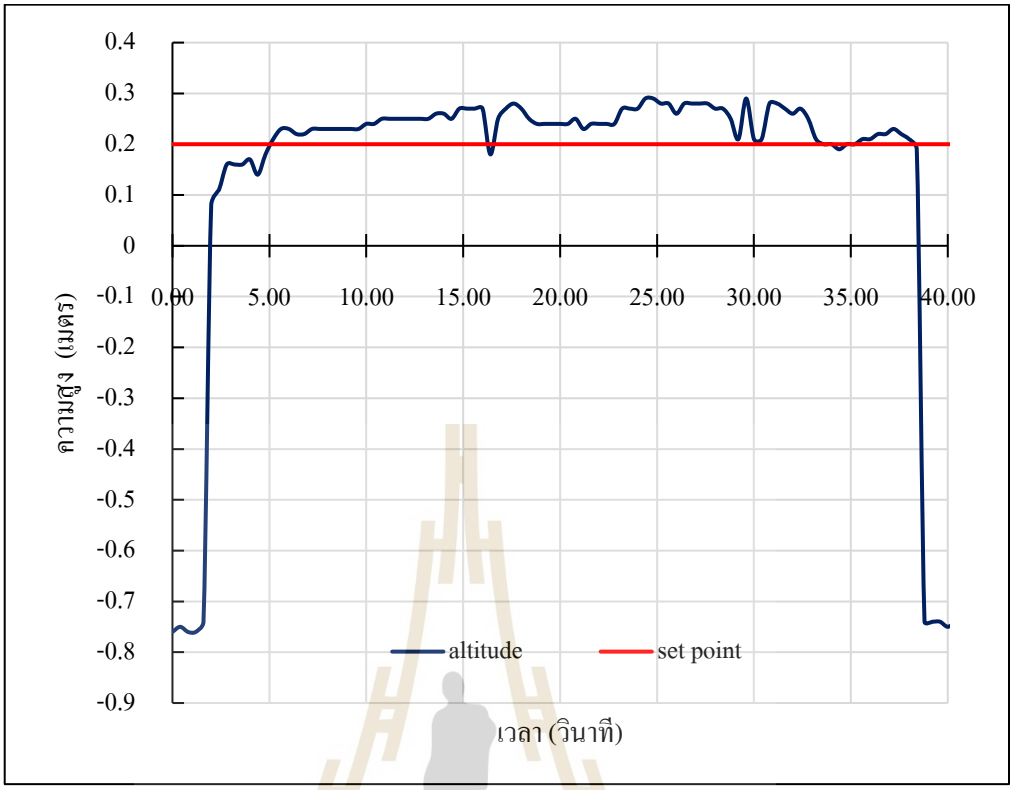
1) การเชื่อมต่อระหว่างโดรนและคอมพิวเตอร์ ซึ่งตัวโมดูลการรับสัญญาณของโดรนอาจมีปัญหาจึงทำให้ในบางครั้ง คอมพิวเตอร์ส่งคำสั่งไปแต่โดรนอยู่กับที่และไม่ทำตามคำสั่ง

2) การดีเลย์ของสัญญาณภาพที่ส่งมาสำหรับการคำนวณพิกัดและควบคุม ซึ่งมีการดีเลย์อยู่ประมาณ 1-2 วินาที

3) ความคลาดเคลื่อนของการระบุพิกัดที่เกิดจากกรอบที่ใช้ตรวจจับโดรนมีขนาดที่เปลี่ยนไปจึงทำให้เกิดความคลาดเคลื่อนของพิกัดได้



รูปที่ 4.37 รูปแสดงการเคลื่อนที่อัตโนมัติแบบ 2 มิติ ของโดรน



รูปที่ 4.38 รูปแสดงการเคลื่อนที่ในแนวแกน y ที่ความสูงเป้าหมาย 0.20 เมตร

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้นำเสนอการระบุพิกัดด้วยกล้องถ่ายภาพสามมิติ ZED ร่วมกับการตรวจจับวัตถุด้วย AI และควบคุมการเคลื่อนที่ของโดรน Parrot Bebop 2 แบบอัตโนมัติภายในอาคาร เพื่อให้โดรนสามารถทำภารกิจแบบอัตโนมัติภายในอาคารหรือพื้นที่ปิดต่าง ๆ ได้ในกรณีฉุกเฉินงานวิจัยมีรายละเอียดโดยสรุปดังนี้

1. เทรนนิ่งโมเดลเพื่อใช้สำหรับการตรวจจับโดรนรุ่น Parrot bebop 2 โดยโมเดลที่ได้นำมาใช้ในการตรวจจับวัตถุแบบเรียลไททม์มีค่าเฉลี่ยความแม่นยำ (Average precision) อยู่ที่ 0.7272 โดยประมาณ ซึ่งหลังจากนำไปใช้งานแบบเรียลไทม์ก็พบว่าโมเดลที่ได้สามารถตรวจจับวัตถุได้โดยมีการรบกวนน้อยกว่าโมเดลอื่น ๆ

2. การทดสอบความแม่นยำของระบบระบุพิกัดด้วยกล้องถ่ายภาพสามมิติร่วมกับ AI พบว่า กรณีที่ทดสอบโดยโดรนไม่เคลื่อนที่ แกน x มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.05 เมตร แกน y มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.04 เมตร และแกน z มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.03 เมตร ในกรณีที่โดรนมีการเคลื่อนที่ค่าพิกัดแกน z มีค่าเฉลี่ยความคลาดเคลื่อนสูงสุดจากทั้ง 3 แกน โดยมีค่าอยู่ที่ 0.04 เมตร และมีค่าความแปรปรวนอยู่ที่ 0.0026 สำหรับพิกัดแกน x มีค่าเฉลี่ยความคลาดเคลื่อนอยู่ที่ 0.01 เมตร และมีค่าความแปรปรวนอยู่ที่ 0.0004 และค่าเฉลี่ยของความคลาดเคลื่อนของพิกัดแกน y อยู่ที่ -0.01 เมตร ค่าความแปรปรวนอยู่ที่ 0.0027 โดยรวมแล้วค่าเฉลี่ยความคลาดเคลื่อนไม่เกิน 0.10 เมตร

3. ระบบควบคุมที่ใช้เป็นระบบควบคุม P เนื่องจากการควบคุมการเคลื่อนที่ของโดรนเน้นที่ช่วง Transient response จึงไม่ได้ใช้ตัวควบคุม I ในระบบ และเมื่อทำการทดสอบร่วมกับตัวควบคุม D แล้วพบว่า เสถียรภาพการทำงานของระบบควบคุมลดลง ดังนั้นจึงใช้เพียงระบบควบคุม P ในการควบคุมการเคลื่อนที่ของโดรนทั้ง 3 แกน โดยที่ค่า K_p ที่เหมาะสมสำหรับควบคุมการเคลื่อนที่แกน x, y และ z คือ 0.035, 0.1 และ 0.04 ตามลำดับ

4. ผลการทดสอบระบบระบุพิกัดและระบบนำทางด้วยกล้องถ่ายภาพสามมิติในงานวิจัยนี้พบว่า ระบบสามารถควบคุมโดรนให้เข้าสู่เป้าหมายและทำภารกิจสำเร็จลุล่วงโดยคิดเป็นร้อยละ 73.3

5.2 ข้อเสนอแนะ

1. ระบบตัวควบคุมโดรนนอกจากตัวควบคุม PID แล้วสามารถพัฒนาใช้ระบบตัวควบคุม Adaptive เพื่อให้การควบคุมโดรน ทำได้ดียิ่งขึ้นในช่วงการเคลื่อนที่ต่าง ๆ

2. การกรองสัญญาณรบกวนที่ได้จากการอ่านค่าพิกัดด้วยกล้องถ่ายภาพสามมิติ สามารถพัฒนาโดยการใช้ตัวกรอง Karman ช่วยทำให้ค่าที่ได้แม่นยำมากขึ้นได้

3. นอกจากการใช้ Parrot Bebop 2 ร่วมกับระบบแล้วสามารถนำไปพัฒนาต่อยอดโดยการใช้ร่วมกับโดรนอื่น ๆ ได้

รายการอ้างอิง

ปริญญา สงวนสัตย์. (2019). **Artificial Intelligence with Machine Learning, AI สร้างได้ด้วยแมชชีนเลิร์นนิ่ง**. นนทบุรี: ไอดีซี พรีเมียร์. 356 หน้า. ISBN 978-616-487-071-0.

Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M. A., & Al-Khalifa, H. S. (2016). Ultra-wideband indoor positioning technologies: Analysis and recent advances. **Sensors**, 16(5), 707.

Adept Turnkey Pty Ltd. **Making it work with the stereo-matching technique and texture projection**. [Online]. Available: http://www.adept.net. [Accessed 12 FEB 2020].

Al-Naji, A. A., Gibson, K., Lee, S.-H., & Chahl, J. (2017). Real Time Apnoea Monitoring of Children Using the Microsoft Kinect Sensor: A Pilot Study. **Sensors**, 17, 286.

Atish P. Sinha and Jerrold H. May. (2004). Evaluating and Tuning Predictive Data Mining Models Using Receiver Operating Characteristic Curves. **Journal of Management Information Systems**. 21(3). 249-280.

Nenchoo, B., & Tantrairatn, S. (2020). Real-Time 3D UAV Pose Estimation by Visualization. **Proceedings**, 39, 18.

Debupt. (2018). **Indoor Positioning with ArduPilot based on UWB modules**. [Online]. Available: https://discuss.ardupilot.org. [Accessed 27 Jan 2020].

Endo, Y., Sato, K., Yamashita, A., & Matsubayashi, K. (2017). Indoor positioning and obstacle detection for visually impaired navigation system based on LSD-SLAM. **In 2017 International Conference on Biometrics and Kansei Engineering (ICBAKE)**. 158-162.

Intel Cooperation. **Intel® RealSense™ Depth Camera D435i**. [Online]. Available: https://www.intelrealsense.com. [Accessed 12 Feb 2020].

Barton, J. (2012). Fundamentals of Small Unmanned Aircraft Flight. **Johns Hopkins Apl Technical Digest**, 31, 132-149.

Jonathan Hui. (2018). **Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)**. [Online]. Available: https://medium.com. [Accessed 04 Feb 2020].

K. Satangmongkol. (2019). **อธิบาย 10 Metrics พื้นฐานสำหรับวัดผลโมเดล Machine Learning**. [Online]. Available: https://datarockie.com. [Accessed 27 Jan 2020].

Kasidis Satangmongkol. (2017). Machine Learning 101- ทดสอบ model accuracy in Excel. [Online]. Available: https://medium.com. [Accessed 31 Jan 2020].

Keng Surapong. (2019). **Activation Function คืออะไร ใน Artificial Neural Network, Sigmoid Function คืออะไร-Activation Function ep.1** [Online]. Available: https://www.bualabs.com. [Accessed 04 FEB 2020].

Keng Surapong. (2019). **Metrics คืออะไร Confusion Matrix คืออะไร Accuracy, Precision, Recall, F1 Score ต่างกันอย่างไร-Metrics ep.1**. [Online]. Available: https://www.bualabs.com. [Accessed 27 Jan 2020].

Kuantama, E., Vesselenyi, T., Dzitac, S., & Tarca, R. (2017). PID and Fuzzy-PID Control Model for Quadcopter Attitude with Disturbance Parameter. **International Journal of Computers, Communications and Control**. 12. 519-532.

Li, Y., Scanavino, M., Capello, E., Dabbene, F., Guglieri, G., & Vilardi, A. (2018). A novel distributed architecture for UAV indoor navigation. **Transportation Research Procedia**, 35, 13-22.

Liu, B., & Paquin, N. (2018). Viconmavlink: A software tool for indoor positioning using a motion capture system. **CoRR abs/1811.11878**.

Marko B. (2016). **YOLO ROS: Real-Time Object Detection for ROS**. [Online]. Available: https://github.com. [Accessed 01 Nov 2019].

Mr.P L. (2018). **Evaluate Model นั้นสำคัญอย่างไร? Machine Learning 101**. [Online]. Available: https://medium.com. [Accessed 31 Jan 2020].

Morat, J., Devernay, F., Ibanez-Guzman, J., & Cornou, S. (2007). Evaluation Method for Automotive Stereo-Vision Systems. **2007 IEEE Intelligent Vehicles Symposium**. 202-208.

Natthawat Phongchit. (2018). **Convolutional Neural Network (CNN) คืออะไร**. [Online]. Available: https://blog.datawow.io. [Accessed 12 Feb 2020].

Natthawat Phongchit. (2019). **ย้อนรอย Object Detection และเจาะลึก RetinaNet**. [Online]. Available: https://blog.datawow.io. [Accessed 04 Feb 2020].

Phyblas. (2018). **[python] วาดเส้นกราฟ ROC เพื่อประเมินผลการทำนาย**. [Online]. Available: https://phyblas.hinaboshi.com. [Accessed 27 Jan 2020].

PradyaSin. (2019). **Convolution Neural Network คืออะไร**. [Online]. Available: https://medium.com. [Accessed 12 Feb 2020].

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. 779-788.

Rodríguez-Quiñonez, J., Sergiyenko, O., Flores-Fuentes, W., Lopez, M., Hernandez-Balbuena, D., Rascon, R., & Mercorelli, P. (2017). Improve a 3D distance measurement accuracy in stereo vision systems using optimization methods' approach. **Opto-Electronics Review**, 25. 24-32.

S. Al Habsi, M. Shehada, M. Abdoon, A. Mashood and H. Noura. (2015). Integration of a Vicon camera system for indoor flight of a Parrot AR Drone. **10th International Symposium on Mechatronics and its Applications (ISMA)**. 1-6.

S. J. Ingram, D. Harmer and M. Quinlan. (2004). Ultrawideband indoor positioning systems and their use in emergencies. PLANS 2004. **Position Location and Navigation Symposium (IEEE)**. 706-715.

Sahu, Samvram. (2019). 3D Pose Estimation of UAVs using Stereo-vision. **Department of Avionics Indian Institute of Space Science and Technology Thiruvananthapuram**. India.

Sanparith Marukatat. (2018). **โลกหมุนไปงานวิจัยก็หมุนตาม**. [Online]. Available: https://medium.com. [Accessed 04 Feb 2020].

Sergio G., M. Elena L., et al. (2016). Indoor SLAM for Micro Aerial Vehicles Control using Monocular Camera and Sensor Fusion. **2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)**. 205-210.

Sik-Ho Tsang. (2018). **Review: SSD-Single Shot Detector (Object Detection)**. [Online]. Available: https://towardsdatascience.com. [Accessed 04 Feb 2020].

Stereo Labs. (2021). **Meet ZED**. [Online]. Available: https://www.stereolabs.com/zed/. [Accessed 04 Feb 2020].

Suphan Fayong. (2018). **Machine Learning ตอนที่ 4 เตรียมข้อมูลเพื่อสร้าง Model**. [Online]. Available: http://codeonthehill.com. [Accessed 31 Jan 2020].

Theppasith N. (2018). **Introduction + Installation**. [Online]. Available: https://medium.com. [Accessed 16 June 2019].

Theppasith N. (2018). **Concept of Modularity**. [Online]. Available: https://medium.com. [Accessed 16 Jun 2019].

Yasir Mohd Mustafah, Amelia Wong Azman, Fajril Akbar. (2012). Indoor UAV Positioning Using Stereo Vision Sensor. **Procedia Engineering**. 41. 575-579.

ภาคผนวก ก

โปรแกรมระบุพิกัดและโปรแกรมควบคุมการเคลื่อนที่ของโดรน

```
#include <ros/ros.h>
#include <std_msgs/Int8.h>
#include <std_msgs/String.h>
#include <std_msgs/Empty.h>
#include <sensor_msgs/Imu.h>
#include <nav_msgs/Odometry.h>
#include <geometry_msgs/Twist.h>
#include <geometry_msgs/Pose.h>
#include <geometry_msgs/PoseStamped.h>
#include <darknet_ros_msgs/ObjectCount.h>

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <time.h>

using namespace std;
//using for cout/cin

float x_max, y_max, z_max, x_dst, y_dst, z_dst, x_total,
y_total, x_min, y_min, x_fov_min, x_fov_max;
float error_right, error_left,  diff_error_x, diff_times_x,
times_x, times_error_x;
float error_up, error_down, diff_error_y, diff_times_y, times_y,
times_error_y ;
float error_fwd, error_bwd, diff_error_z, diff_times_z, times_z,
times_error_z;

double P_x, I_x, D_x, PID_x, i_x,  derivative_x;
//variable of x  kp_x,kd_x,ki_x,
double P_y, I_y, D_y, PID_y, i_y,  derivative_y;
//variable of y  kp_y,kd_y, ki_y,
double P_z, I_z, D_z, PID_z, i_z,  derivative_z;
//variable of z  kp_z,kd_z,  ki_z,
double pre_error_x = 0.0;
double pre_error_y = 0.0;
double pre_error_z = 0.0;
double kp_x = 0.035;
double kp_y = 0.1;
double kp_z = 0.04;
double ki_x = 0.0;
double ki_y = 0.0;
double ki_z = 0.0;
double kd_x = 0.0;
double kd_y = 0.0;
double kd_z = 0.0;
float x_update ;                                          //new x
pose
float y_update ;                                          //new y
pose
float z_update ;                                          //new z
pose
float y_init;
//initial y position at floor
```

```
int detect_num;
int dummy = 0;
int dst_set = 1;
int count_first = 0;
char status_ready;
char status_continue;
clock_t times = clock();
float sample_time = times*0.03/10000;
// times in sec.
bool already_takeoff = false;

class SubscribeAndPublish
{
public:
    SubscribeAndPublish(){
        // publish and subscibe topic
        sub_position = n.subscribe("/zed_position", 10,
&SubscribeAndPublish::get_position,this);
//subscribe topic from point2.cpp
        sub_detection = n.subscribe("/darknet_ros/found_object",
10, &SubscribeAndPublish::detection_check,this);     //subscribe
topic for detection check
        sub_gain_tunning = n.subscribe("/tunning_PI", 10,
&SubscribeAndPublish::tunning_gain,this);
        sub_gain_tunning_D = n.subscribe("/tunning_D", 10,
&SubscribeAndPublish::tunning_gain_D,this);


        velocity_pub = n.advertise<geometry_msgs::Twist>
("bebop/cmd_vel", 10);                  // publish topic
        takeoff_pub = n.advertise<std_msgs::Empty>
("bebop/takeoff", 10);                          // publish takeoff
topic
        land_pub = n.advertise<std_msgs::Empty>("bebop/land",
10);                                    // publish land topic
        dst_pub = n.advertise<geometry_msgs::Twist>
("/dst_point", 10);                             // publish destination
point topic to point2.cpp
    };

    void set_area(){
        //setting flying area
        printf("---Please define the flying area---\n");
        cout << "The maximum depth from camera(meter): ";
//receive depth input
        cin >> z_max;
//collect value to z_max
        cout << "The maximum width (meter): ";
//receive width input
        cin >> x_total;
//collect value to x_max
        cout << "The maximum height(meter): ";
//receive height input
        cin >> y_total;
//collect value to y_max

        //Calculation boundary areas
        x_min = -(x_total/2);
        x_max = x_total/2;
        y_min = -(y_total/2);
        y_max = y_total/2;

        count_first++;
    }
```

```
void get_position(const geometry_msgs::PoseStamped::ConstPtr&
pose)         //receive position function
    {
        x_update = pose->pose.position.x;
        y_update = pose->pose.position.y;
        z_update = pose->pose.position.z;

        if(dst_set == 1){
            printf("\n---Bebop2 detected---\n");
            ROS_INFO("Bebop2 at X = %.1f, Y = %.1f and Z = %.1f
\n", x_update, y_update, z_update);
        }
    }

    void tunning_gain(const geometry_msgs::Twist::ConstPtr&
gain)               //tunning kp, ki gain controller function
    {
        kp_x = gain->linear.x;
        kp_y = gain->linear.y;
        kp_z = gain->linear.z;

        ki_x = gain->angular.x;
        ki_y = gain->angular.y;
        ki_z = gain->angular.z;
    }

     void tunning_gain_D(const geometry_msgs::Twist::ConstPtr&
gain)               //tunning kp, ki gain controller function
    {
        kd_x = gain->linear.x;
        kd_y = gain->linear.y;
        kd_z = gain->linear.z;

        // ROS_INFO("This is tuningD");
        // printf("kd_x = %f \n", kd_x);
        // printf("kd_y = %f \n", kd_y);
        // printf("kd_z = %f \n", kd_z);
    }

    void control_velocity(){
        //control bebop to destination
        geometry_msgs::Twist velocity;
//variable
        clock_t times = clock();
        float sample_time = times*0.03/10000;

            ///left-right control///
            if (x_update || x_update == 0){

                // printf("\n///left-right control///");
                if (x_update < x_dst)
// bebop have to go left
                {
                    error_left = x_dst - x_update;
// calculation error left +
                    times_error_x = sample_time;
                    // ROS_INFO("going right:\n");
                }
                else if (x_update > x_dst)
// bebop have to go right
                {
                    error_right = x_dst - x_update;
// calculation error right -
                    times_error_x = sample_time;
                    // ROS_INFO("going left:\n");
                }
                else
                {
                    //errorleft=0;
                    //errorrigth=0;
                    times_x = sample_time;
                }

                diff error x = error right + error left;
```

```
///up-down control///
            if (y_update || y_update == 0){
                // printf("\n///up-down control///");
                if (y_update < y_dst)
// bebop have to go up
                {
                    error_up = y_dst - y_update;
// calculation error up +
                    times_error_y = sample_time;
                    // ROS_INFO("going up:\n");

                }
                else if (y_update > y_dst)
// bebop have to go down
                {
                    error_down = y_dst - y_update;
// calculation error down -
                    times_error_y = sample_time;
                    // ROS_INFO("going down:\n");
                }
                else
                {
                    //errorleft=0;
                    //errorrigth=0;
                    times_y = sample_time;
                }

                diff_error_y = error_up + error_down;
// calculation delta error between up and down
                ROS_INFO(" error_y = %.1f", diff_error_y);
                // printf("  y = %.1f\n", y_update);


                if (diff_error_y > 0.05 || diff_error_y
< -0.05 )
                {
                //// PID control calculation ////

                    //calculation of integral term
                    diff_times_y = times_error_y - times_y;
//calculation delta times
                    i_y += diff_error_y * diff_times_y;
                    I_y = ki_y * i_y;

                    //calculation of proportional term
                    P_y = kp_y * diff_error_y;

                    //calculation of Derivative term
                    derivative_y = (diff_error_y -
pre_error_y)/diff_times_y;
                    D_y = kd_y * derivative_y;
                    pre_error_y = diff_error_y;

                    //summation of proportional and integral
term
                    PID_y = P_y + I_y + D_y;

                    ROS_INFO(" PID_y = %f, kp_y = %f, ki_y =
%.10f, kd_y = %f\n",PID_y, kp_y, ki_y, kd_y);
                    // ROS_INFO(" P_y = %f, I_y = %f, D_y = %f
\n", P_y, I_y, D_y);

                    ROS_INFO("y_init = %.2f", y_init);
                    if (y_update < y_dst)
{                                       // bebop have to
go up
                        ROS_INFO("going up: ");
                        ROS_INFO("y_update: %f, y_dst: %f
\n",y_update, y_dst);
                    }
```

```
else if (y_update > y_dst)
{                                           // bebop have to go
down
                        ROS_INFO("going down: ");
                        ROS_INFO("y_update: %f, y_dst: %f
\n",y_update, y_dst);
                    }
                }
                if (y_update < y_dst || y_update > y_dst)
                {
                    velocity.linear.z = PID_y;
//+ ascend, - descend
                    // PID_y = 0.0;
                }
                velocity_pub.publish(velocity);
            }
            else if (!y_update)
// cannot detect
            {
                error_up = 0;
                error_down = 0;
                velocity.linear.z = 0;
// velocity command = 0 bebop not move up or down
                velocity_pub.publish(velocity);
            }

            ///backward-forward control///
            if (z_update || z_update == 0){
                // printf("\n///forward-backward control///");
                if (z_update < z_dst)
// bebop have to go forward
                {
                    error_fwd = z_dst - z_update;
// calculation error forward +
                    times_error_z = sample_time;
                    // ROS_INFO("going forward:\n");
                }
                else if (z_update > z_dst)
// bebop have to go backward
                {
                    error_bwd = z_dst - z_update;
// calculation error backward -
                    times_error_z = sample_time;
                    // ROS_INFO("going backward:\n");
                }
                else
                {
                    //errorleft=0;
                    //errorrigth=0;
                    times_z = sample_time;
                }

                diff_error_z = error_fwd + error_bwd;
// calculation delta error between forward and backward
                ROS_INFO(" error_z = %.1f", diff_error_z);
                // ROS_INFO("  z = %.1f\n", z_update);

                if (diff_error_z > 0.05 || diff_error_z
< -0.05  )
                {
                //// PID control calculation ////
                    //calculation of integral term
                    diff_times_z = times_error_z - times_z;
//calculation delta times
                    i_z += diff_error_z * diff_times_z;
                    I_z = ki_z * i_z;

                    //calculation of proportional term
                    P_z = kp_z * diff_error_z;

                    //calculation of Derivative term
                    derivative_z = (diff_error_z -
pre_error_z)/diff_times_z;
                    D_z = kd_z * derivative_z;
                    pre_error_z = diff_error_z;
```

```
//summation of proportional and integral term
                    PID_z = P_z + I_z + D_z;

                    ROS_INFO(" PID_z = %f, kp_z = %f, ki_z =
%.10f, kd_z = %f\n\n",PID_z, kp_z, ki_z, kd_z);
                    // ROS_INFO(" P_z = %f, I_z = %f, D_z = %f
\n", P_z, I_z, D_z);

                    if (z_update < z_dst)
{                                       // bebop have to go
forward
                        ROS_INFO("going forward: ");
                        ROS_INFO("z_update: %f, z_dst: %f
\n",z_update, z_dst);

                    }
                    else if (z_update > z_dst)
{                                       // bebop have to go
backward
                        ROS_INFO("going backward: ");
                        ROS_INFO("z_update: %f, z_dst: %f
\n",z_update, z_dst);

                    }
                }
                if (z_update < z_dst || z_update > z_dst)
                {
                    velocity.linear.x = PID_z;
//+ forward, - backward
                    // PID_z = 0.0;

                }
                velocity_pub.publish(velocity);
            }
            else if (!z_update)
// cannot detect
            {
                error_fwd = 0;
                error_bwd = 0;
                velocity.linear.x = 0;
// velocity command = 0 bebop not move backward or forward
                velocity_pub.publish(velocity);
            }
        }

    void detection_check(const darknet_ros_msgs::ObjectCount&
check)                  //detection check function
    {
        std_msgs::Empty takeoff;
//variable
        std_msgs::Empty land;
//variable
        geometry_msgs::Twist dst;
        geometry_msgs::Twist velocity;

        detect_num = check.count;
//mockup value for simulation test

        if(dst_set != 3){

            if (dummy == 0){
                set_area();
            }
            //Detection check
            if (detect_num == 1) {
```

```
// Ready for takeoff check
                if (dummy == 0){
                    printf("\n---Pre armming---\n");
                    cout << "Ready for takeoff?(y/n): ";
//receive ready command y = yes,ready/ n = not ready
                    cin >> status_ready;
                    if (count_first == 1){
                        y_init = y_update;
//setting for checking takeoff condition
                        ROS_INFO("y_init = %.2f", y_init);
                    }

                    // ROS_INFO("y_update = %f", y_update);

                    if (status_ready == 'n' || status_ready ==
'N'){
                        dummy = 0;
                    }
                }
                if (status_ready == 'y'||status_ready == 'Y')
//status ready
                {
                    dummy = 1;
                    takeoff_pub.publish(takeoff);

                    if (y_update > (y_init + 0.1)){
                        already_takeoff = true;
                    }

                    //takeoff checking condition
                    if ( already_takeoff ){

                        //setting destination point
                        if (dst_set == 1 || dst_set == 2 ) {
                            ROS_INFO("y_init = %.2f", y_init);
                            printf("\n---Please enter
destination point---\n");
                            cout << "please enter left or
right(meter): ";              //receive left or right
destination
                            cin >> x_dst;
//collect value to destination of x
                            cout << "please enter height from
camera's center(meter): ";       //receive height of
destination
                            cin >> y_dst;
//collect value to destination of y
                            cout << "please enter depth from
camera(meter): ";                 //receive height input
                            cin >> z_dst;
//collect value to destination os z
                            dst_set = 0;

                            dst.linear.x = x_dst;
                            dst.linear.y = y_dst;
                            dst.linear.z = z_dst;
                            dst_pub.publish(dst);
                        }
                        x_fov_min = -(2.22*z_dst + 0.5)/2;
                        x_fov_max = (2.22*z_dst + 0.5)/2;
                        if (x_dst < x_min || x_dst > x_max ||
x_dst < x_fov_min || x_dst > x_fov_max || y_dst < y_min
                            || y_dst > y_max ||  z_dst < 2.0  ||
z_dst > z_max){
                            printf("\nOut of flying area, please
try again.\n\n");
                            printf("x_min = %f, x_max = %f,
y_min = %f, y_max = %f,x_fov_min = %f, x_fov_max = %f\n\n",
                            x_min, x_max, y_min,y_max,x_fov_min,
x_fov_max);
                            dst_set = 1;
                        }
```

```
if (dst_set == 0){

                              control_velocity();

                         }
                         ///landing after arrives to
destination///
                         if ((diff_error_x <= 0.05 &&
diff_error_x >= -0.05) && (diff_error_y <= 0.05 &&
diff_error_y >= -0.05) &&
                              (diff_error_z <= 0.05 &&
diff_error_z >= -0.05)){

                              //Clear error//
                              diff_error_x = 0;
                              diff_error_y = 0;
                              diff_error_z = 0;
                              PID_x = 0.0;
                              PID_y = 0.0;
                              PID_z = 0.0;

                              //Continue condition//
                              cout << "Do you want to continue?
(y/n): ";

                              cin >> status_continue;


                              if(status_continue == 'y' ||
status_continue == 'Y'){          // continue go to destination
set

                                   printf("---Next
destination---\n");

                                   dst_set = 2;


                              }


                              if(status_continue == 'n' ||
status_continue == 'N'){          // not continue landing
                                   printf("\n---Bebop is
landing---\n");

                                   land_pub.publish(land);
                                   dst_set = 3;
                              }
                         }
                    }
               }

else{
//else for not ready to takeoff or not arming
                    printf("\nArmming denied, please try again.
\n\n");
               }
          }
```

```
    else{
                    velocity.linear.x = 0.0;
                    velocity.linear.y = 0.0;
                    velocity.linear.z = 0.0;
                    velocity_pub.publish(velocity);
                }
            }
            else{
                printf("\n---End Process---\n");
            }

        }


    private:
        ros::NodeHandle n;
        ros::Publisher velocity_pub;
        ros::Publisher takeoff_pub;
        ros::Publisher land_pub;
        ros::Publisher dst_pub;
        ros::Subscriber sub_position;
        ros::Subscriber sub_detection;
        ros::Subscriber sub_gain_tunning;
        ros::Subscriber sub_gain_tunning_D;
    };


    int main(int argc, char **argv)
    {
    ros::init(argc, argv, "navigation_bebop2");


    SubscribeAndPublish SAPObject;
    ros::spin();
    return 0;

    }
```

```
#include <iostream>
#include <bits/stdc++.h>
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <pcl_ros/point_cloud.h>
#include <pcl/point_types.h>
#include <darknet_ros_msgs/BoundingBoxes.h>
#include <darknet_ros_msgs/BoundingBox.h>
#include <geometry_msgs/PoseWithCovariance.h>
#include <geometry_msgs/Point.h>
#include <geometry_msgs/Twist.h>


#include <boost/foreach.hpp>
#include <math.h>
#include <fstream>
#include <stdio.h>
#include <iomanip>
#include <stack>
#include <vector>
#include <string>

typedef pcl::PointCloud<pcl::PointXYZ> PointCloud;
using namespace std;
    float x = 0.0;
    float y = 0.0;
    float z = 0.0;
    float sum_x = 0.0;
    float sum_y = 0.0;
    float sum_z = 0.0;
    double x_obj, y_obj, z_obj;
    double x_dst, y_dst, z_dst;
    int i;
    int rounds;
    int x_center_screen = 640;
    int y_bottom_screen = 360;
    int x_error, y_error;
    int dummy = 0;
    clock_t times = clock();
    float sample_time = times;
    FILE * pFileTXT;
    // float cam_height = 0.0; //0.125; //metre
    string classes;
```

```
class SubscribeAndPublish
{
public:
  SubscribeAndPublish()
  {
    //Topic you want to publish
    pose_pub = n_.advertise<geometry_msgs::PoseStamped>
("/zed_position", 10);
    // gps = n_.advertise<geometry_msgs::PoseWithCovariance>
("/gps", 10);

    //Topic you want to subscribe
    sub_ =
n_.subscribe("/zed/zed_node/point_cloud/cloud_registered", 1,
&SubscribeAndPublish::callback, this);
    sub1_ = n_.subscribe("/darknet_ros/bounding_boxes", 1,
&SubscribeAndPublish::calculate, this); // for yolo detect
method
    dst_sub = n_.subscribe("/dst_point", 1,
&SubscribeAndPublish::setText, this);

    // image_sub = n_.subscribe("/blob/point_blob", 1,
&SubscribeAndPublish::calculate, this); //for image processing
method
  }
    int x_center,y_center, x_min, x_max, y_min, y_max;
    float Y_cali,X_cali,Z_cali,x_ave,y_ave,z_ave,y_ave_reverse;
    float round(float var)
    {
      float value = (int)(var * 10);
      return (float)value / 10;
    }

// // Calculate Pixel centroid from yolo

  void calculate(const darknet_ros_msgs::BoundingBoxes& msg)
  {
    if (msg.bounding_boxes[0].id == 0){
    classes = msg.bounding_boxes[0].Class;
    x_center = ((msg.bounding_boxes[0].xmax -
msg.bounding_boxes[0].xmin)/2)+ msg.bounding_boxes[0].xmin;
    y_center = ((msg.bounding_boxes[0].ymax -
msg.bounding_boxes[0].ymin)/2) + msg.bounding_boxes[0].ymin;

    dummy =1;
  //  ROS_INFO("\ncalculate function\n");
      }
    }
```

```
void setText(const geometry_msgs::Twist& msg)
  {
    x_dst = msg.linear.x;
    y_dst = msg.linear.y;
    z_dst = msg.linear.z;

    pFileTXT = fopen("Position_log.txt", "a");
    fprintf (pFileTXT, "\ntimes=  %f",sample_time);
    fprintf (pFileTXT, "\ndestination point: x =  %.2f,y =
%.2f , z = %.2f m.\n",x_dst, y_dst, z_dst);
    fclose (pFileTXT);
  }

//// convert to meter ////
  void callback(const PointCloud::ConstPtr& msg)
  {
    clock_t times = clock();
    float sample_time = times*0.03/10000;
  if (dummy == 1){
    geometry_msgs::PoseStamped pose;
    x_obj = msg->points[(y_center*1280)+x_center-1].y;
    y_obj = msg->points[(y_center*1280)+x_center-1].z;
    z_obj = msg->points[(y_center*1280)+x_center-1].x;
    // printf("\nx_obj =%.2f, y_obj=%.2f, z_obj=%.2f\n", x_obj,
y_obj, z_obj);
    dummy = 0;
    //// REDUCE NOISE PROCESS BY AVERAGE ////
    // summation of value
    if((x_obj == x_obj && y_obj == y_obj && z_obj == z_obj) &&
      (x_obj != -INFINITY && y_obj != -INFINITY && z_obj != -
INFINITY && x_obj != INFINITY && y_obj != INFINITY && z_obj !=
INFINITY)&&
      (x_obj > -2.0 && x_obj < 2.0) && (y_obj > -1.0 && y_obj
< 1.0) && (z_obj > 0.0 && z_obj < 4.0)){
        // printf("\nPart1\n");
        // summation
        sum_x = x_obj + sum_x;
        sum_y = y_obj + sum_y;
        sum_z = z_obj + sum_z;
        // printf("summation of x = %f\n", sum_x);
        i++;

      // average value
      if(i % 5 == 0 && i != 0){
        x_ave = sum_x/5;
        y_ave = sum_y/5;
        z_ave = sum_z/5;
        rounds++;
        printf("\nround: %d\n",rounds);
        printf("Class: %s\n", classes.c_str());
        printf("x_ave = %.2f,y_ave = %.2f,z_ave = %.2f\n",x_ave,
y_ave, z_ave);
```

```
  //clear value
          sum_x = 0;
          sum_y = 0;
          sum_z = 0;

          // logfile
          pFileTXT = fopen("Position_log.txt", "a");
          fprintf (pFileTXT, "times=  %f round:%d x =  %.2f,y =
%.2f , z = %.2f m.\n",sample_time, rounds, x_ave, y_ave, z_ave);
          fclose (pFileTXT);

          pFileTXT = fopen("Bounding_box_center_log.txt", "a");
          fprintf (pFileTXT, "round:%d center =  %d,%d \n",rounds,
x_center, y_center);
          fclose (pFileTXT);

          pose.pose.position.x = x_ave;
          pose.pose.position.y = y_ave;
          pose.pose.position.z = z_ave;
          pose_pub.publish(pose);
        }
      }
    }
    }

  private:
    ros::NodeHandle n_;
    ros::Publisher pose_pub;
    ros::Publisher gps;
    ros::Subscriber sub_;
    ros::Subscriber sub1_;
    ros::Subscriber dst_sub;

    // ros::Subscriber image_sub;


  };//End of class SubscribeAndPublish


  int main(int argc, char **argv)
  {
    //Initiate ROS
    ros::init(argc, argv, "subscribe_and_publish");

    //Create an object of class SubscribeAndPublish that will take
  care of everything
    SubscribeAndPublish SAPObject;
    ros::Rate rate(10);
    ros::spin();

    return 0;
  }
```

ภาคผนวก ข

บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

รายชื่อบทความวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

Nenchoo, B., & Tantrairatn, S. (2020). Real-Time 3D UAV Pose Estimation by Visualization. **Proceedings**, 39, 18.

Chaisena, K., Nenchoo, B., & Tantrairatn, S. (2020). Automatic Balancing System in Quadcopter with Change in Center of Gravity. **IOP Conference Series: Materials Science and Engineering**, 886, 012006.

proceedings

MDPI

Proceedings

Real-Time 3D UAV Pose Estimation by Visualization †

Benjamaporn Nenchoo and Suradet Tantrairatn *

Institute of Engineering, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand; benz_2469@hotmail.com

* Correspondence: suradetj@sut.ac.th; Tel.: +66-955-623-555

† Presented at the Innovation Aviation & Aerospace Industry—International Conference 2020 (IAAI 2020), Chumphon, Thailand, 13–17 January 2020.

Published: 7 January 2020

Abstract: This paper presents an estimation of 3D UAV position in real-time condition by using Intel RealSense Depth camera D435i with visual object detection technique as a local positioning system for indoor environment. Nowadays, global positioning system or GPS is able to specify UAV position for outdoor environment. However, for indoor environment GPS hasn't a capability to determine UAV position. Therefore, Depth stereo camera D435i is proposed to observe on ground to specify UAV position for indoor environment instead of GPS. Using deep learning for object detection to identify target object with depth camera to specifies 2D position of target object. In addition, depth position is estimated by stereo camera and target size. For experiment, Parrot Bebop2 as a target object is detected by using YOLOv3 as a real-time object detection system. However, trained Fully Convolutional Neural Networks (FCNNs) model is considerably significant for object detection, thus the model has been trained for bebop2 only. To conclude, this proposed system is able to specifies 3D position of bebop2 for indoor environment. For future work, this research will be developed and apply for visualized navigation control of drone swarm.

Keywords: deep learning; indoor positioning system; visualization

1. Introduction

The application to deploy drone in indoor environments has been needed, for instance using UAV to silos inspection, disaster relief, warehouse management etc., Nevertheless, positioning for indoor environment is the main challenge for researcher. Because GPS hasn't a capability to determine UAV position as discussed by Mainetti et al. (2014) [2] and Mautz et al. (2009) [4]. Thus, sensors with high precision, accuracy performance (centimeter level) and low latency are necessary for UAV indoor applications.

Motion tracking systems use reflected markers to detect position, speed and orientation of object. UAVs are able to flight at the same time by using markers with different shapes. Each camera has coverage area depending on the field of view. A ground station receives and processes camera data, allowing motion-tracking reconstruction. For conclude, Motion tracking systems work as an artificial GPS for indoor environment. Therefore, with high performance and accuracy they have to exchange with high cost for maintenance and calibration [7].

Stereo or IR depth perception cameras technologies in robotics are able to reconstruct 3D model and understand the environment. The high accuracy position and orientation accuracy make these sensors suitable to perform visual odometry algorithms [7]. Intel RealSense Depth camera D435i combines the robust depth sensing capabilities of the D435 with the addition of an inertial measurement unit (IMU). The maximum range Approx. 10 m. Accuracy depending on calibration, scene, and lighting condition. Depth field of view approx. 87° ± 3° × 58° ± 1° × 95° ± 3° with Intel

RealSense Vision Processor D435i as a vision processor board. Due to lower cost than motion tracking systems stereo depth camera used in this research as a sensor.

Therefore, this paper present 3D position estimation by using object detection with Intel RealSense Depth camera D435i as sensor for indoor environment application. For conclusion, this proposed system is able to specifies 3D position of bebop2 for indoor environment.

The paper is organized as follows: Section 2 Materials and Method describe about model training, detection, connection and working flow chart. Section 3 Results and Discussion describes testing result in static and flying test. Section 4 Conclusion describes conclusions of the work and future work.

2. Materials and Methods

This section provides a description of the Model Training and Detection and Positioning Method used in this research.

2.1. Model Training and Detection

Model training is one of procedure for deep learning. Accuracy of trained model depends on dataset, epochs, learning rates and other parameters. In this research, model is trained to detect bebop2 while flying, take-off and landing with 8000 images of dataset that based on YOLOv3(coco) model with Supervisely platform. Dataset's preparation is important procedure before training. After images was labeled, these have to do data transformation language (DTL) process viz vertical flip and horizontal flip. This process is defined with a JSON based config file. YOLO architecture is provided FCNN (fully convolutional neural network). The images (nxn) are passed through the FCNN, then output is (mxm) prediction. The convolutional neural network has 24 convolutional layers with 2 fully connected layers. Alternating 1 × 1 convolutional layer reduce the features space from preceding layers. The convolutional layers pretrained on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection [1] as shown in Figure 1.

However, the connection of YOLOv3 with Intel RealSense Depth camera D435i aren't able to directly connect each other. By the way, this problem solved by using YOLOv3 base on Robot Operating Systems (ROS) or Darknet ROS package work as ROS bridge.

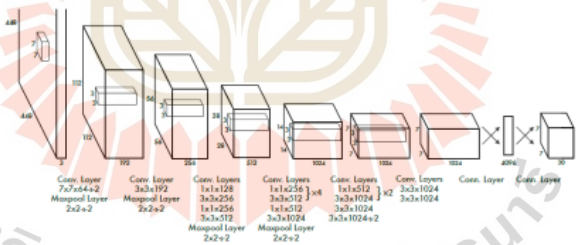


Figure 1. YOLO's architecture [1].

2.2. Positioning Method

3D positioning systems for indoor environment using Intel RealSense Depth camera D435i is utilized as a positioning sensor which set origin coordinates X = 0 at middle of camera's frame, Y = 0 at ground level and Z = 0 at position of depth camera and detect Bebop2 with YOLOv3 object detection. The bounding box pixels of target object on screen size 640 × 480 pixels send to points cloud topics of Intel RealSense Depth camera D435i that collect the coordinate x, y and z in every single pixel. For receives the coordinate x, y and z in meter unit, coordinates pixel center x and y from darknet ros [3] send to points cloud topic with point cloud library (PCL) for transformation of pixel to meter. Moreover, there is some noise from the position x, y and z in meter after transformation

that could filter. This research, using average method with 50 window size of data as a Noise filtration method. In additional the schematic diagram of working flow shown in Figure 2.

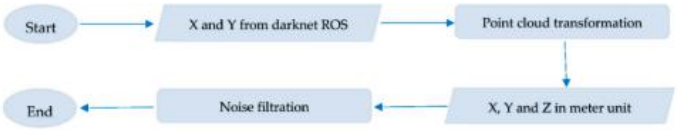


Figure 2. The schematic diagram of working flow.

3. Results and Discussion

In test implementation, Bebop2 was employed as a target object the camera by setting above the ground 45 cm. Testing area for indoor environment with area size width (X) 2.5 m., depth (Z) 2.5 m. and height (Y) 1.5 m. and minimum depth 0.5 m. The camera calibrated before testing in 6 positions randomly. The coordinate X and Z is the camera position and Y is ground. Setting up for testing is illustrated in Figure 3.

Figure 3. Setting up of camera for testing.

3.1. Static Testing

The testing performed to estimate the indoor position of UAVs. The estimated x, y and z position of the UAV at several positions are collected and the example results is shown in the Table 1.

Table 1. The indoor positioning system result. x, y and z between the measure and actual position of the UAV.

X (m.)	Y(m.)	Z(m.)	Mean X(m.)	Mean Y(m.)	Mean Z (m.)
1	1.5	2.5	0.99	1.5	2.55
0.5	1	2	0.58	1.09	1.97
−0.5	0.5	1.5	0.46	0.48	1.5
0	0	1	0.04	−0.06	0.97

The system able to measure accurate position in x, y and z where the mean of the estimated position values is close to the actual values. The system has standard deviation about 0 due to noise filtration from calculation process in coding. Moreover, comparison with GPS system for outdoor environment which has accuracy about ±3 m. Thus, this positioning system is more reliable. The additional results of object detection are demonstrated in Figure 4.

Figure 4. Result of object detection while static testing.

3.2. Flying Testing

Testing perform to estimate indoor position of UAV while flying and 3D trajectory plot with MATLAB program. Height (Y values from system) is altitude from ground in unit of meter, depth (Z from system) is distance from depth camera to UAV in unit of meter and X value is width from middle of camera. Moreover, the calculation process with fast process since position is update at a frequency less than 10 Hz and variance is acceptable. However, there is some mistake during testing due to model that is sometime unable to continuously detect. The result of flying test is shown in Figure 5.

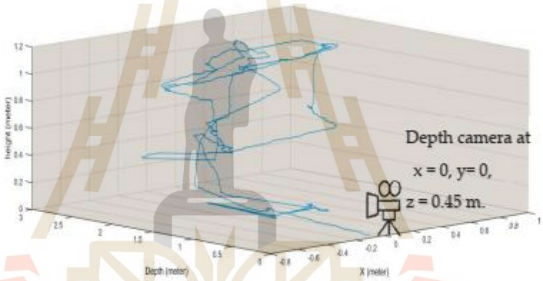


Figure 5. 3D trajectory plot of UAV by MATLAB with indoor positioning system.

4. Conclusions

UAV for indoor application has become increasing popular in application viz disaster release, mapping, indoor inspection etc. This research present one of choices to specify 3D position of UAV for indoor environment by using deep learning with Intel RealSense Depth camera D435i. The testing results show satisfy performance of system with low variance, high accuracy with maximum error about 10 cm. and fast by updating position at frequency less than 10 Hz. However, sometimes the camera system was unable to detect object due to trained model. Therefore, for future work, the trained model performance need to more improve by training more accuracy model. Moreover, filtration process will be applied with advanced techniques including Karman filter for more accuracy and this system will be applied to specify 3D position and navigation of swarm drone.

References

1 Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2016**, arXiv:1506.02640v5.

2 Mainetti, L.; Luigi, P.; Ilaria, S. A survey on indoor positioning systems. In Proceedings of the 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 17–19 September 2014.
3 Marko B. YOLO ROS: Real-Time Object Detection for ROS. 2017. Available online: https://github.com/leggedrobotics/darknet_ros (accessed on 1 November 2019).
4 Mautz, R. Overview of current indoor positioning systems. *Geodezija ir kartografija* **2009**, *35*, 18–22.
5 Yasir, M.; Amelia, W.; Fajril, A. Indoor UAV Positioning Using Stereo Vision Sensor. *Procedia Eng.* **2012**, *41*, 575–579.
6 Yohanes, K.; Izabela, N. A system of UAV application in indoor environment. *Prod. Manuf. Res. Open Access J.* **2016**, *4*, 2–22.
7 Li, Y.; Scanavino, M.; Capello, E.; Dabbene, F.; Guglieri, G.; Vilardi, A. A novel distributed architecture for UAV indoor navigation. *Transp. Res. Procedia* **2018**, *35*, 13–22.

IOP Conference Series: Materials Science and Engineering

PAPER • OPEN ACCESS

Automatic Balancing System in Quadcopter with Change in Center of Gravity

To cite this article: Kamolwat Chaisena *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **886** 012006

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

Automatic Balancing System in Quadcopter with Change in Center of Gravity

Kamolwat Chaisena[1], Benjamaporn Nenchoo[1] and Suradet Tantrairatn[1,*]

[1]Suranaree University of Technology, Nakhonratchasima, 30000, Thailand

* Corresponding Author: suradetj@sut.ac.th

Abstract
The purpose of this paper is to explore an automatic balancing system using the proportional integral derivative or PID feedback control algorithm in quad-rotor Helicopter, so-called quadcopter, with the center of gravity, moved on account of payload. Nowadays, the quadcopter widely used for a variety of applications due to high stability performance. In many applications in quadcopter usually carried a payload in the middle of its frame to maintain the balance during flight. However, the application of quadcopter could be a contributing factor to increase a payload on one side and can lead to an improper position of the center of gravity. This study set out to investigate the use of the mass of battery and landing gear to create a counterbalance to compensate for the payload. The stabilization of the balancing system was implemented by the PID control algorithm to control the counterbalancing of the quadcopter. The current study found that the PID control algorithm works well to maintain the balance in a quadcopter with the center of gravity moving during flight.

Keywords: Balancing System, PID control, Unbalance of Quadcopter

1. Introduction

Since it reported in the early 1900s, a multi-rotor helicopter has been attracting a lot of interest. In the 1920s, there has been an increasing interest in multi-rotor helicopter since it has been used in the military. One of the greatest challenges of the application of a multi-rotor helicopter is its huge, overweight, and instability in control. In a few decades, the technology in the field of Unmanned Aerial Vehicle or UAV has been developing continuously. This type of UAV aircraft was developed to a smaller size to suit the application. It also used in military and civil missions.

Nowadays, a quadcopter is considered as multi-rotor aircraft and has been widely used due to its lightweight and high-lift force, vertically take-off and landing. It has become a central issue for hovering in the air and remains stable for flight control. This type of UAV aircraft consists of four rotors at the end of the arm. Its movement is allowed an adjustment of the rotor speed independently so that it can control a pitch, roll and yaw. The quadcopter can play an important role in addressing the issue of carrying a payload. For instance, it can be used as a camera for aerial video and photography, gripper for pick and place the object, transportation of goods. Generally, the payload is

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

usually attached in the center of mass of the aircraft body to maintain stability during flight. However, in some applications, the payload is mounted out in front of the body because there is not enough space in the center of the aircraft frame.

Studies of quadcopter show the importance of improving PID control so that it can increase the stability in aircraft control [1-4]. Moreover, several attempts have been made to investigate an effect of variations in center of gravity such as the response of rotating speed of the rotors [5], the PID control used to stabilize the quadrotor with dynamic change in center of gravity on account of payload [6]. In particular [7], this research has established the solution to maintain the stability of multi-rotor during flight using landing gear. And, it has been acting as a tail in a mechanical system to create a counterbalance to compensate the payload.

In previous research [8], we have presented an automatic balancing system in quadcopter using the aircraft component as lithium polymer battery, which energizes the whole system to create a counterbalance to compensate the payload attached on the front of the quadcopter frame. The method of the balancing system implemented by PID, which was used to control the stabilization. The battery had linearly driven a servomotor to create a counterbalance. As a result, the response of the previous system was quite slow because the movement was changed from rotation to translation. In the previous work, it had not been mentioned about the improvement of PID control.

This paper is to explore the relationship between a new model of a quadcopter and automatic balancing system using a combination of the mass of battery and landing gear. It is also to create a counterbalance to have a faster response and to present the process of improvements in PID control of stabilization.

2. Concept and Method

This section discusses the model and design of the quadcopter and concept of balancing system. The S-500 PCB quad-rotor frame was used. The quadcopter consists of four motors, there are two motors rotate clockwise (CW) direction and two motors rotate counterclockwise (CCW) direction as shown in Figure1. The moment force around the vertical axis of the aircraft body was compensated for each other.

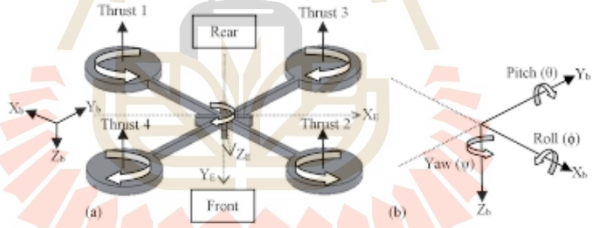


Figure 1. A free body diagram of quadcopter X [9]

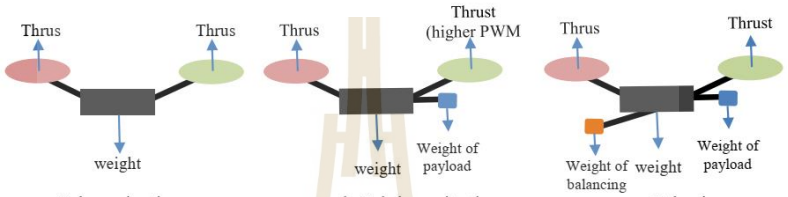
In this study, the balance situation with the equally PWM signal value of four motors used as a condition. The unbalance situation simulated by installing a payload on the front of the aircraft frame. The PWM signal value of four motors was unable to have equal values. From this data, the PWM signal values of a pair of front motors showed greater value than a pair of rear motors. The lightweight aluminum bar is attached to the front of the aircraft frame so that it was able to adjust the position of aluminum block to simulate the movement of the center of gravity when carrying a payload (see Figure 2). Figure 3 provides a free body diagram of a quadcopter with three situations.

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

a. Unbalance system in a real situation

b. Unbalance system in a simulate situation

Figure 2. Qu.adcopters with payload



a. Balance situation b. Unbalance situation c. Balancing

Figure 3. Free body diagram of a quadcopter with three situations

For the balancing mechanism, a couple of high - torque RC servo motor was used as an actuator at the landing gear joints to fold the rear of the aircraft frame and to compensate the payload. As can be seen from Figure 4 below, the battery was mounted at the bottom of the landing gear to increase a balancing capacity.

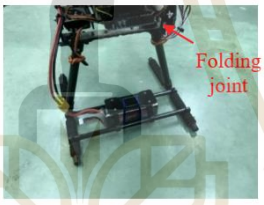


Figure 4. Battery attached with landing gear

The advantage of this balancing mechanism is a faster response than previous research [8] because the folding angle of landing gear depends on the angular angle of faster servomotor. Table 1 shows the aircraft specifications of some parameters.

Table 1. Quadcopter Specifications

Parameter	Size (grams)
Take-off Weight	1,695
Battery	235
Payload	330
Landing Gear	240

3. System Integration

The method of the study based on the main hardware of the quadcopter using the lithium battery as a payload and power source to create the balancing force. Pixhawk board for control stability and movement of quadcopter and receive PWM signal from the transmitter through the receiver.

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

Moreover, there is an Arduino board as a processing unit to control the servo motor for controlling and balancing the system (see Figure 5 below).

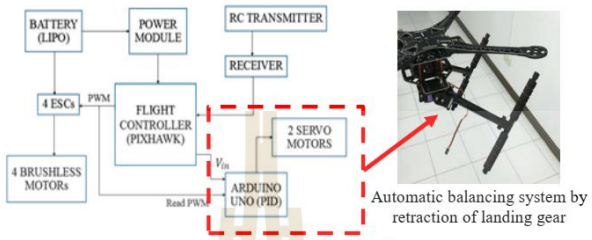
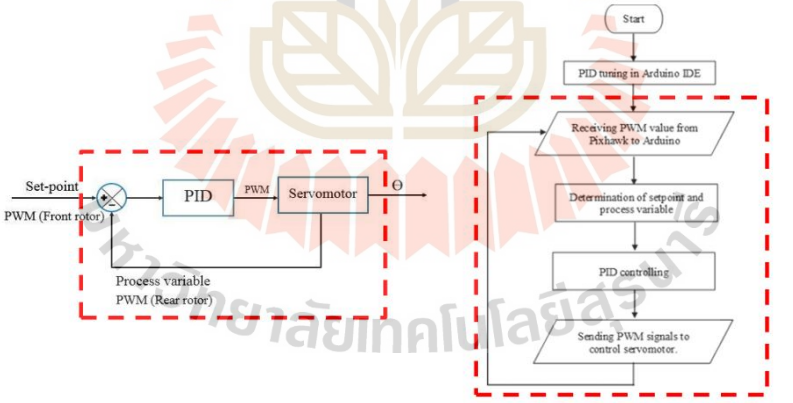


Figure 5. A diagram of a hardware system

By the way, it can be seen that complete systems need more than one hardware system, it also needs software and controller to control the systems. It is apparent from the diagram that the proportional – integral – derivative or PID control is applicable in many industrials and a wide range of research. This controller response is able to correct by tuning three values as Kp, Ki and Kd to create minimum errors.

It suggested that PWM values of a pair of front rotors have equal value and set as a set point. It also has equal value and set as a process variable. As shown in Figure 6a, the setpoint value as an input was sent to the PID controller to control servomotor until a process variable converges effectively to a set point. Besides, the process balancing system starts with tuning PID value with trial and error method in the Arduino board. Arduino board receives PWM values from Pixhawk and sets these values as setpoint and process variable. These values sent to the PID controlling process to control servomotor for balancing system. The schematic diagram provided in Figure 6b.



a. A PID block diagram

b. A schematic diagram of the balancing system

Figure 6. An automatic balancing system diagram

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

4. Experiments and Results analysis

This section shows the experiments and results of the study in four cases. It consists of a test that is with a payload and the test without a payload. It also comprises of a test that is with balancing and a test with a static balancing. To study the behavior of the PWM signal in various cases, a flight plotter program was used to analyze the data.

4.1 Testing without a payload

From the data in Figure 7, it is apparent that the stability of quadcopter is the most important issue since the position of any devices that set in quadcopter has an effect on PWM signals. We can see that the PWM signal values almost have equal values about 1700 (see Figure 7 below).

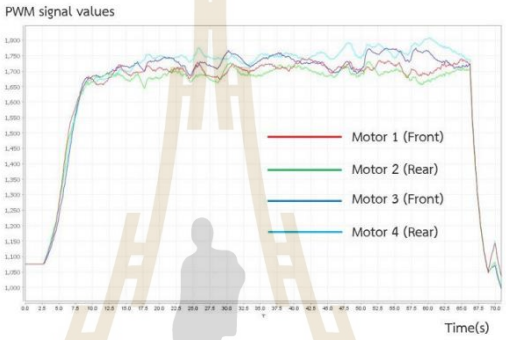


Figure 7. PWM values response in case with balance.

4.2 Testing with a payload

As Figure 8 shows, there is the 330 grams payload attached in front of the frame (15 cm), and it displaces between payload and center of gravity. The result shows that the PWM mean value of a pair of front motors (approx. 1700) was greater than a pair of rear motors (approx. 1550). Hence, the system of quadcopter has to maintain stability by increasing PWM values of the heavier side, as shown in Figure 8.

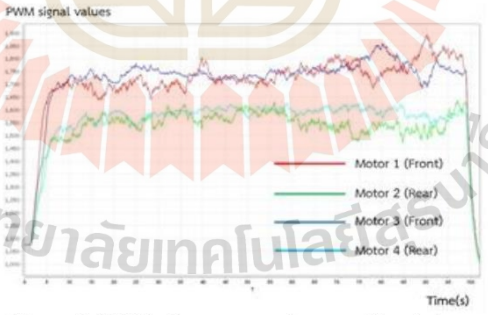


Figure 8. PWM values response in case with unbalance

4.3 Testing with balancing

As seen from the data in Figure 9, the folding of landing gear to the backside used as a counterbalance for the balancing system. The trial and error method used to find PID gain value (Kp, Ki, and Kd). It

10th TSME-International Conference on Mechanical Engineering (TSME-ICoME 2019) IOP Publishing
IOP Conf. Series: Materials Science and Engineering **886** (2020) 012006 doi:10.1088/1757-899X/886/1/012006

also used to control the rotation of the servo motor. After trial and error tuning, there are Kp, Ki, and Kd values that suit this system and they have chosen as 0.0375, 0.0175 and 0, respectively.

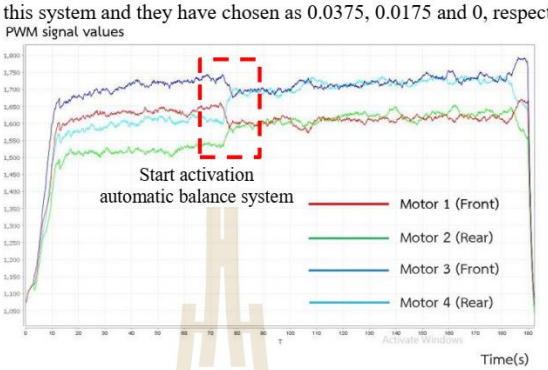


Figure 9. PWM values response in case with balancing.

As can be seen from Figure 8, the result demonstrates that the PWM value of a pair of the motor in the same diagonal line has converged to each other. The settling time is about three seconds and without overshoot of servomotor.

4.4 Static balancing test

After testing all cases, there is a static balancing test to make sure that these gain values are able to maintain the stability of the quadcopter. By testing the tilt angle of the quadcopter in three cases, the data shows balance, unbalance, and balancing. The 330 grams payload was attached at front of the frame in unbalance and balancing test with displacement 15, 17.5 and 20 cm, respectively. A payload at the center of gravity of quadcopter (Kp, Ki, and Kd values) was chosen in testing 4.3, and used in balancing test (see Table 2)

Table 2. Result of static balancing test

Displacement (cm.)	Angle (Deg.)				
	Balance	Unbalance	Balancing	error	Landing gear retraction
15	0.91	15.09	2.70	1.78	44.95
17.5	0.93	17.88	1.86	0.92	45.58
20	0.54	20.54	1.60	1.06	76.08

As shown in Table 2, it is apparent that the displacement of the payload has an effect on the angle of landing gear retraction by increasing displacement. Also, it is able to increase the angle of landing gear retraction. From the data, we can see that displacement of the payload has direct variation to an angle of landing gear retraction (see Figure 10 below).

Figure 10. Tilt angle in any static testing case.

5. Conclusion

This study has found that generally, the balancing system by folding landing gear with 475 grams counterbalance payload due to the building of quadcopter with 330 grams of payload. The most obvious finding to emerge from this study is that the PID controller used in this balancing system for maintaining the stability of quadcopter with Kp, Ki and Kd values as 0.0375, 0.0175 and 0, respectively. The results can also be summarised as follow:

5.1 In balance case (without payload): PWM signal values of 4 motor have equal value about 1700.

5.2 In unbalance case (with payload): PWM signal values of a pair of the front motor is greater than a pair of the rear motor.

5.3 In balancing case: PID controller able to control a servo motor with three seconds of settling time, without overshoot and maintain the stability of a quadcopter.

References

[1] Li J and Li Y 2011 *Proc. 2011 IEEE Int. Conf. on Mechatronics and Automation (Beijing)* (China)

[2] Praveen V and Pillai A S 2016 *IJCTA*. **9**(15) 7151-7158

[3] Wang P Man Z Cao Z Zeng J and Zhao Y 2016 *Proc. 2016 Int. Conf. on Advanced Mechatronics Systems (Melbourne)* (Australia) pp 498-503

[4] Qasim M Susanto E and Wibowo A S 2017 *Proc. 5th Int. Conf. on Instrumentation, Control, and Automation* pp 109-114

[5] Xu X Liu C and Ye B 2016 *Proc. 8th Int. Conf. on Modelling, Identification and Control* pp 1348-353

[6] Ariyanto M Paryanto M and Naniwa T 2016 *Proc. 8th Int. Conf. on Information Technology and Electrical Engineering (Yogyakarta)* (Indonesia) pp 1-6

[7] Molina J and Hirai S 2017 *Proc. Int. Conf. on Unmanned Aircraft Systems (Miami)* (USA) pp1731-1736

[8] Chaisena K Chamniprasart K and Tantrairatn S 2018 *Proc. Int. Conf. on Engineering Science and Innovative Technology (Phang-Nga)* (Thailand) pp 1-5

[9] Kuantama E 2017 *Int. J. Computers Communication and Control* **12**(4) 519-532

ประวัติผู้เขียน

นางสาวเบญจมาภรณ์ เณรชู เกิดเมื่อวันที่ 7 ธันวาคม พ.ศ. 2538 ที่อำเภอเมือง จังหวัดนครราชสีมา เริ่มต้นชั้นประถมศึกษาที่โรงเรียนเมืองนครราชสีมา จังหวัดนครราชสีมา จนกระทั่งระดับมัธยมศึกษา ได้เข้าศึกษาต่อที่โรงเรียนสุรนารีวิทยา จังหวัดนครราชสีมา ต่อมาเมื่อปี พ.ศ. 2557 ได้ศึกษาต่อในระดับอุดมศึกษาที่มหาวิทยาลัยเทคโนโลยีสุรนารี สาขาวิชาวิศวกรรมเครื่องกล หลักสูตรวิศวกรรมอากาศยาน สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิตในปี พ.ศ. 2561 ต่อมาได้เข้าศึกษาต่อในระดับบัณฑิตศึกษา สาขาวิชาวิศวกรรมเครื่องกลและระบบกระบวนการ ในปี พ.ศ. 2562 ได้ร่วมงานวิจัยกับ Dr. Sutthiphong Srigrarom ที่ Singapore Polytechnic ประเทศสิงคโปร์ เป็นเวลา 1 เดือน

ขณะศึกษาระดับบัณฑิตศึกษาได้รับมอบหมายให้เป็นผู้สอนในสาขาวิชาวิศวกรรมเครื่องกล จำนวน 2 รายวิชา ดังต่อไปนี้

1. Aircraft Maintenance and Aircraft System Laboratory
2. Aerodynamics and Aircraft Structure Laboratory

ผลงานวิจัย: ได้นำเสนอบทความจำนวน 2 บทความ ดังต่อไปนี้

1. บทความเรื่อง “Real-Time 3D UAV Pose Estimation by Visualization” การประชุมวิชาการระดับนานาชาติ Innovation Aviation & Aerospace Industry International Conference 2020 (IAAI 2020) 13-17 มกราคม 2563 จังหวัดชุมพร ดังปรากฏในภาคผนวก ข.

2. บทความเรื่อง “Automatic Balancing System in Quadcopter with Change in Center of Gravity” การประชุมวิชาการนานาชาติเครือข่ายวิศวกรรมเครื่องกลแห่งประเทศไทย ครั้งที่ 10 (TSME-ICOME 2019) 10-13 ธันวาคม 2562 พัทยา ดังปรากฏในภาคผนวก ข.