การปรับปรุงสถาปัตยกรรม PointPillars ด้วยการประมวลผลข้อมูล LiDAR เพื่อลดภาระการคำนวณของระบบตรวจจับ 3 มิติ

นายนัฐพงษ์ พวงมาลี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเมคคาทรอนิกส์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2568

ENHANCING POINTPILLARS ARCHITECTURE WITH LIDAR DATA PROCESSING MINIMIZE COMPUTATIONAL LOAD OF 3D DETECTION SYSTEM

NATTAPONG PHUANGMALEE

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Mechatronics Engineering
Suranaree University of Technology
Academic Year 2025

การปรับปรุงสถาปัตยกรรม PointPillars ด้วยการประมวลผลข้อมูล LiDAR
เพื่อลดภาระการคำนวณของระบบตรวจจับ 3 มิติ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

.....

(รศ. ดร.จิระพล ศรีเสริฐผล)
ประธานกรรมการ

.....

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

.....

(ผศ. ดร.โศรฎา แข็งการ)
กรรมการ

.....

(ดร.สุพัฒน์ กลิ่นเขียว)
กรรมการ

.....

(รศ. ดร.ยุพาพร รักสกุลพิวัฒน์)
รักษาการแทนรองอธิการบดีฝ่ายวิชาการและประกันคุณภาพ

.....

(รศ. ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

นัฐพงษ์ พวงมาลี : การปรับปรุงสถาปัตยกรรม PointPillars ด้วยการประมวลผลข้อมูล LiDAR เพื่อลดภาระการคำนวณของระบบตรวจจับ 3 มิติ
(ENHANCING POINTPILLARS ARCHITECTURE WITH LIDAR DATA PROCESSING MINIMIZE COMPUTATIONAL LOAD OF 3D DETECTION SYSTEM)
อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.สุรเดช ตัญตรัยรัตน์, 109 หน้า

คำสำคัญ: การตรวจจับวัตถุสามมิติ/ไลดาร์/ปัญญาประดิษฐ์

วิจัยนี้มุ่งเน้นการพัฒนาและปรับปรุงสถาปัตยกรรมของโมเดล PointPillars สำหรับการตรวจจับวัตถุสามมิติจากข้อมูล LiDAR โดยมีวัตถุประสงค์เพื่อลดภาระการคำนวณให้เหมาะสมกับการใช้งานบนหุ่นยนต์อัตโนมัติขนาดเล็กที่มีข้อจำกัดด้านทรัพยากรการประมวลผล งานวิจัยนี้ตระหนักถึงข้อได้เปรียบของเซ็นเซอร์ LiDAR ในการให้ข้อมูลเชิงลึกที่แม่นยำกว่าเซ็นเซอร์ภาพทั่วไป แต่การนำมาใช้งานกับโมเดลเชิงลึกที่ซับซ้อนอาจไม่สอดคล้องกับข้อจำกัดของฮาร์ดแวร์ระดับฝังตัว ดังนั้นจึงได้มีการปรับโครงสร้างของโมเดล PointPillars โดยเน้นการจำกัดขอบเขตการตรวจจับให้แคบลงตามระยะปฏิบัติงานจริงของหุ่นยนต์ และเสริมด้วยกลไกความสนใจ (Attention Mechanism) เพื่อเน้นเฉพาะบริเวณที่สำคัญต่อการตรวจจับ ส่งผลให้ระบบสามารถทำงานได้อย่างแม่นยำและมีประสิทธิภาพด้านเวลาแม้บนแพลตฟอร์มที่มีพลังการประมวลผลต่ำ งานวิจัยนี้จึงมีบทบาทสำคัญต่อการยกระดับความสามารถของหุ่นยนต์อัตโนมัติในสภาพแวดล้อมจริง เช่น การใช้งานบนหุ่นยนต์ส่งของที่ต้องตรวจจับสิ่งกีดขวางอย่างรวดเร็วและปลอดภัยบนทางเท้าหรือพื้นที่สาธารณะ

สาขาวิชาวิศวกรรมเมคคาทรอนิกส์ ลายมือชื่อนักศึกษา นัฐพงษ์ พวงมาลี

ปีการศึกษา 2568 ลายมือชื่ออาจารย์ที่ปรึกษา สุรเดช ตัญตรัยรัตน์

NATTAPONG PHUANGMALEE : ENHANCING POINTPILLARS ARCHITECTURE WITH LIDAR DATA PROCESSING MINIMIZE COMPUTATIONAL LOAD OF 3D DETECTION SYSTEM)
THESIS ADVISOR : ASST. PROF. SURADET TANTRAIRATN, Ph.D., 109 PP.

Keywords: 3D Object Detection/LiDAR/Artificial Intelligence

This research focuses on the development and enhancement of the PointPillars architecture for 3D object detection using LiDAR data, with the objective of reducing computational load for deployment on small-scale autonomous robots with limited processing resources. Recognizing the advantages of LiDAR sensors in providing precise depth information—superior to traditional 2D imaging sensors—this study addresses the challenge of applying complex deep learning models on resource-constrained embedded platforms. To tackle this, the architecture of the PointPillars model has been optimized by narrowing the detection range to align with the robot's actual operating scope and integrating an attention mechanism to emphasize critical regions for detection. These improvements enable the model to operate accurately and efficiently in real time, even on low-power platforms. This research is particularly significant for enhancing the real-world capabilities of autonomous robotic systems, such as delivery robots, which require fast and reliable obstacle detection on sidewalks or other public pathways.

School of Mechatronics Engineering
Academic Year 2025

Student's Signature ณัฐพงษ์ พวงมาลี
Advisor's Signature สุรเดช ตันตรัยรัตน์

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณา แนะนำให้คำปรึกษา ตรวจสอบแก้ไข ทั้งทางด้านวิชาการและการดำเนินงานวิจัยจาก คณาจารย์ บุคลากร นักศึกษาที่เกี่ยวข้องของมหาวิทยาลัยเทคโนโลยีสุรนารีข้าพขอขอบคุณอย่างสูงสุดไว้ ณ โอกาสนี้

ผู้ข้าพเจ้า ขอบกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สุรเดช ตัญตรัยรัตน์ อาจารย์ที่ปรึกษาในการทำวิทยานิพนธ์ ที่ให้โอกาสทางการศึกษาระดับบัณฑิตศึกษา ผู้ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้า และคำแนะนำในการดำเนินการวิจัยทุกขั้นตอนเป็นอย่างดียิ่ง ตลอดจนความช่วยเหลือในการตรวจทานแก้ไขวิทยานิพนธ์นี้จนเสร็จสมบูรณ์

ขอบพระคุณคณาจารย์และกรรมการสอบวิทยานิพนธ์อันประกอบด้วย รองศาสตราจารย์ ดร.จิรพล ศรีเสริฐผล ประธานกรรมการสอบ ผู้ช่วยศาสตราจารย์ ดร.โศรฎา แข็งการ และ ดร.สุพัฒน์ กลิ่นเขียว กรรมการสอบวิทยานิพนธ์ทุกท่านที่ให้คำแนะนำอย่างดียิ่งในการพัฒนาผลงานวิจัยตลอดจนข้อเสนอแนะที่เป็นประโยชน์ในการปรับปรุงและพัฒนาร่างวิทยานิพนธ์ฉบับนี้

ขอบพระคุณมหาวิทยาลัยเทคโนโลยีสุรนารี ที่ได้มอบทุน OROG ให้แก่ผู้วิจัย สำหรับการสนับสนุนค่าเล่าเรียนและค่าบัณฑิตศึกษา รวมถึงทีมงานวิจัยที่อาคารเครื่องมือ F11 ที่ให้ผู้วิจัยสามารถดำเนินการเก็บผลการทดลองได้ด้วยความสะดวก

สุดท้ายนี้ ขอขอบคุณ บิดา มารดา ที่ให้การเลี้ยงดู ขอขอบคุณครอบครัวที่ให้กำลังใจอย่างเต็มที่มาโดยตลอดและเพื่อนนักศึกษาทุกคนที่ให้ความช่วยเกลือและสามารถทำงานร่วมกันได้อย่างราบรื่น สำเร็จวิทยานิพนธ์ฉบับนี้ได้เพราะความร่วมมือของหลายฝ่ายที่ใช้เวลาร่วมกันตลอดจนจบการศึกษาในครั้งนี้

นัฐพงษ์ พวงมาลี

สารบัญ

หน้า

บทคัดย่อ (ภาษาไทย).....ก
บทคัดย่อ (ภาษาอังกฤษ).....ข
กิตติกรรมประกาศ.....ค
สารบัญ.....ง
สารบัญตาราง.....ช
สารบัญรูป.....ซ

บทที่

1 บทนำ 1
1.1 ความสำคัญและที่มาของปัญหาการวิจัย 1
1.2 วัตถุประสงค์ของการวิจัย 2
1.3 ขอบเขตของงานวิจัย 2
1.4 ประโยชน์ที่คาดว่าจะได้รับ 2
2 ปริทัศน์วรรณกรรมและทฤษฎีที่เกี่ยวข้อง 3
2.1 ทฤษฎีที่เกี่ยวข้อง 3
2.1.1 หุ่นยนต์ขนส่งอัตโนมัติ 3
2.1.2 เซ็นเซอร์ LiDAR 4
2.1.3 การเรียนรู้ของเครื่องจักร 6
2.1.4 สถาปัตยกรรม PointPillars 11
2.1.5 โครงข่ายพีระมิดความสนใจ (Lightweight Attention Pyramid Network) 13
2.1.6 การวิเคราะห์เชิงประจักษ์ของระยะการตรวจจับ (An Empirical Analysis of Detection Range) 15
2.1.7 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System – ROS) 16
2.1.8 Jetson Xavier AGX และ Jetson Stats 19

สารบัญ (ต่อ)

หน้า

2.1.9 คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (Onboard Computer of Autonomous Golf Cart) 20
2.1.10 เครื่องมือสำหรับการทำจัดการข้อมูลและการติดป้ายกำกับข้อมูลจุดสามมิติ (labelCloud) 22
2.1.11 งานวิจัยที่เกี่ยวข้อง 22
2.2 การประยุกต์ใช้งานเทคโนโลยี LiDAR และการตรวจจับวัตถุสามมิติ 25
3 วิธีการดำเนินงานวิจัย 28
3.1 การเตรียมการ 28
3.1.1 การศึกษาค้นคว้าและรวบรวมงานวิจัยที่เกี่ยวข้อง 28
3.1.2 การเตรียมพื้นที่ที่จะทำการทดสอบ 29
3.1.3 ทดสอบการทำงานของ LiDAR 31
3.2 การปรับปรุงสถาปัตยกรรม PointPillars 32
3.2.1 การเก็บข้อมูลพอยคลาวด์ 32
3.2.2 การกรองข้อมูล 36
3.2.3 การปรับแต่งโครงสร้าง 38
3.2.4 การตรวจจับวัตถุ 3 มิติจากชุดข้อมูลพอยคลาวด์ 44
3.2.5 การสร้างโมเดลการเรียนรู้ 46
3.2.6 การทดสอบภาระการคำนวณบนหุ่นยนต์ 48
3.2.7 การทดสอบการวิเคราะห์เพื่อหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ 50
4 ผลการวิจัยและอภิปรายผล 52
4.1 ผลการทดสอบระบบตรวจจับวัตถุ 3 มิติ 52
4.2 ผลการทดสอบภาระการคำนวณบนหุ่นยนต์ 68
4.3 ผลการทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ 84
5 สรุปและข้อเสนอแนะ 93
5.1 สรุปผลการวิจัย 93
5.2 ข้อเสนอแนะ 98
รายการอ้างอิง 100

สารบัญ (ต่อ)

หน้า

ภาคผนวก ก 103
ประวัติผู้เขียน 109

สารบัญตาราง

ตารางที่ **หน้า**

2.1 การประยุกต์ใช้งานเทคโนโลยี LiDAR และ PointPillars 26
4.1 แสดงค่า loss และ parameter จากการฝึกสอนโมเดล 53
4.2 แสดงค่า AP ของโมเดลตรวจจับวัตถุของโมเดลดั้งเดิม 62
4.3 แสดงค่า AP ของโมเดลตรวจจับที่ปรับปรุงแล้ว 62
4.4 แสดงค่า Precison ,Recall และ F1score ของโมเดลตรวจจับวัตถุของโมเดลดั้งเดิม 63
4.5 แสดงค่า Precison ,Recall และ F1score ของโมเดลตรวจจับที่ปรับปรุงแล้ว 63
4.6 สรุปคุณลักษณะหลักของ Jetson Xavier AGX และคอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ 66
4.7 แสดงค่า FPS และเวลาที่โมเดลใช้ในการทำนาย ขณะรันโมเดลอย่างเดียว 67
4.8 แสดงค่า FPS และเวลาที่โมเดลใช้ในการทำนาย พร้อมระบบขับเคลื่อนอัตโนมัติเต็มระบบ . 68
4.9 ทดสอบการตรวจจับ 3 มิติ 84
4.10 ทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ 88
5.1 สรุปเปอร์เซ็นต์การลดลงของการใช้งาน CPU และ GPU หลังการปรับปรุงโมเดล PointPillars 96

สารบัญรูป

รูปที่ **หน้า**

2.1 หุ่นยนต์ขนส่งอัตโนมัติ 4
2.2 เซ็นเซอร์ LiDAR 5
2.3 พอยต์คลาวด์ 6
2.4 โครงสร้าง ANN 7
2.5 โครงสร้างของ PointPillars (Lang, A. H., Vora, S., Caesar, H., Zhou, O. 2019.)..... 11
2.6 โครงสร้าง LAPN..... 14
2.7 การปรับระยะการตรวจจับ 15
2.8 แผนภาพอธิบายการทำงานของ ROS 17
2.9 เครื่องมือสำหรับปรับแต่งชุดข้อมูลพอยคลาวด์ (Rusu, R. B., & Cousins, S. 2011). 17
2.10 เครื่องมือสำหรับแสดงชุดข้อมูลพอยคลาวด์ (Zhou, Q. Y., Park, J., V. 2018)..... 18
2.11 Jetson Xavier AGX..... 19
2.12 หน้าจอแสดงค่าการใช้ทรัพยากรของระบบ (Bonghi, R. 2020). 20
3.1 พื้นที่ทดสอบ..... 29
3.2 พื้นที่ทดสอบที่มีเครื่องหมายวัดระยะ 30
3.3 พื้นที่ทดสอบที่มีวัตถุวางอยู่ที่ใช้ในการตรวจจับ..... 30
3.4 เซนเซอร์ LiDAR 31
3.5 ภาพการทดสอบเปิด LiDAR โดยใช้ C++ และ ไลบรารี่ velodyne บนซอฟต์แวร์ rviz..... 32
3.6 วางแผนการเก็บข้อมูล 33
3.7 ภาพกลุ่มพอยคลาวด์ของกรวยจากเซนเซอร์ LiDAR และภาพกรวยจากกล้อง 34
3.8 ภาพกลุ่มพอยคลาวด์ของคนจากเซนเซอร์ LiDAR และภาพคนจากกล้อง 34
3.9 ภาพกลุ่มพอยคลาวด์ของคนขี่จักรยานจากเซนเซอร์ LiDAR และภาพคนขี่จักรยาน
จากกล้อง..... 35
3.10 ภาพกลุ่มพอยคลาวด์ของรถจากเซนเซอร์ LiDAR และภาพรถจากกล้อง..... 36
3.11 ภาพก่อนการกรองข้อมูล LiDAR บนซอฟต์แวร์ rviz..... 37
3.12 ภาพหลังจากการกรองข้อมูล LiDAR บนซอฟต์แวร์ rviz..... 37

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

3.13 กระบวนการของ Pointpillar ก่อนปรับปรุง 38
3.14 กระบวนการของ Pointpillar ที่ปรับปรุงแล้ว 39
3.15 พื้นที่การตรวจจับพอยคลาวด์และวอกเซลขนาดใหญ่ 39
3.16 พื้นที่การตรวจจับพอยคลาวด์และวอกเซลขนาดเล็ก 40
3.17 กระบวนการของ Backbone ใน Pointpillar ก่อนปรับปรุง 41
3.18 กระบวนการของ Backbone ใน Pointpillar ที่ปรับปรุงแล้ว 41
3.19 กระบวนการของ Attention Pyramid Network หรือ EFA ใน Pointpillar 42
3.20 โครงสร้าง 2D Backcone ของ Pointpillar ก่อนปรับปรุง 43
3.21 โครงสร้าง 2D Backcone ของ Pointpillar ที่ปรับปรุงแล้ว 43
3.22 ขั้นตอนการติดป้ายกำกับโดยใช้เครื่องมือ LabelCloud 45
3.23 ขั้นตอนการแบ่งชุดข้อมูลก่อนนำไปสร้างโมเดลการเรียนรู้ 45
3.24 คำสั่งที่ใช้ในการฝึกสอนโมเดล 9000 รอบ (Epochs) และปรับขนาดแบตช์เป็น 7 46
3.25 โครงสร้างของ Pointpillar ที่ปรับปรุงแล้ว 47
3.26 ทดสอบการใช้โมเดลตรวจจับวัตถุ 3 มิติที่ได้จากการฝึกฝน 48
3.27 แสดงหน้าต่าง Jetson stats ที่ใช้วัดภาระการคำนวณของค่า CPU และ GPU 50
3.28 สมการหาระยะห่างระหว่างจุดสองจุดในระบบพิกัดคาร์ทีเซียน 50
4.1 กราฟจากผลลัพธ์การฝึกของโมเดล 53
4.2 Confusion Matrix Unnormalized แสดงผลการทดสอบของโมเดลดั้งเดิม 54
4.3 Confusion Matrix Unnormalized แสดงผลการทดสอบของโมเดลที่ปรับปรุงแล้ว 55
4.4 Confusion Matrix Normalized แสดงผลการทดสอบของโมเดลดั้งเดิม 56
4.5 Confusion Matrix Normalized แสดงผลการทดสอบโมเดลที่ปรับระยะการตรวจจับแล้ว 56
4.6 กราฟ Precision-Recall Curve ของคลาสคนจากโมเดลดั้งเดิม 57
4.7 กราฟ Precision-Recall Curve ของคลาสคนขี่จักรยานจากโมเดลดั้งเดิม 58
4.8 กราฟ Precision-Recall Curve ของคลาสกรวยจากโมเดลดั้งเดิม 58
4.9 กราฟ Precision-Recall Curve ของคลาสรถยนต์จากโมเดลดั้งเดิม 59
4.10 กราฟ Precision-Recall Curve ของคลาสคนจากโมเดลที่ปรับปรุงแล้ว 60

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

4.11 กราฟ Precision-Recall Curve ของคลาสคนขี่จักรยานจากโมเดลที่ปรับปรุงแล้ว 60
4.12 กราฟ Precision-Recall Curve ของคลาสกรวยจากโมเดลที่ปรับปรุงแล้ว 61
4.13 กราฟ Precision-Recall Curve ของคลาสรถยนต์จากโมเดลที่ปรับปรุงแล้ว 61
4.14 ผลการตรวจจับวัตถุจากพอยคลาวด์(1)..... 64
4.15 ผลการตรวจจับวัตถุจากพอยคลาวด์(2)..... 64
4.16 ผลการตรวจจับวัตถุจากพอยคลาวด์(3)..... 65
4.17 ผลการตรวจจับวัตถุจากพอยคลาวด์(4)..... 65
4.18 การใช้งาน CPU ในสถานะฮาร์ดแวร์ ไม่รันโปรแกรมใดๆ..... 69
4.19 การใช้งาน GPU ในสถานะฮาร์ดแวร์ ไม่รันโปรแกรมใดๆ 70
4.20 การใช้งาน CPU ในสถานะระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดลตรวจจับ..... 70
4.21 การใช้งาน GPU ในสถานะระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดลตรวจจับ 71
4.22 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุง 72
4.23 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุง 73
4.24 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว 74
4.25 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว 75
4.26 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Onboard Computer..... 76
4.27 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Onboard Computer 77
4.28 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว Onboard 78
4.29 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว Onboard Computer..... 79
4.30 การใช้งาน CPU ในสถานะรันโมเดล PointPillars ก่อนปรับปรุงพร้อมระบบอัตโนมัติ 80
4.31 การใช้งาน GPU ในสถานะรันโมเดล PointPillars ก่อนปรับปรุงพร้อมระบบอัตโนมัติ 81
4.32 การใช้งาน CPU ในสถานะรันโมเดล PointPillars ที่ปรับปรุงแล้วพร้อมระบบอัตโนมัติ 82
4.33 การใช้งาน GPU ในสถานะรันโมเดล PointPillars ที่ปรับปรุงแล้วพร้อมระบบอัตโนมัติ 83

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

5.1 แสดงการเปรียบเทียบค่าเฉลี่ยการใช้งานของ CPU และ GPU ระหว่างโมเดล PointPillars ก่อนและหลังการปรับปรุง บนแพลตฟอร์ม Jetson Xavier และ Onboard Compute 96

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ปัจจุบันเทคโนโลยีหุ่นยนต์ขนส่งอัตโนมัติมีบทบาทสำคัญในภาคอุตสาหกรรมและบริการ โดยเฉพาะการขนส่งสิ่งของภายในอาคารอย่างมีประสิทธิภาพ ผู้วิจัยได้เข้าร่วมทีมวิจัยที่พัฒนาหุ่นยนต์ขนส่งอัตโนมัติ ณ ชั้นใต้ดิน อาคาร F11 ซึ่งโครงการนี้เป็นการต่อยอดจากวิทยานิพนธ์ระดับปริญญาโทที่พัฒนารถกอล์ฟไฟฟ้าอัตโนมัติสำหรับขนส่งผู้โดยสารระหว่างอาคาร โดยรถกอล์ฟดังกล่าวสามารถเคลื่อนที่จากจุดเริ่มต้นไปยังเป้าหมาย พร้อมตรวจจับวัตถุ เช่น คนเดินเท้า เพื่อหยุดหรือหลบหลีกสิ่งกีดขวางโดยอัตโนมัติ (วงศธร, 2565)

หุ่นยนต์ขนส่งที่ใช้ในโครงการวิจัยนี้ได้นำระบบอัตโนมัติมาจากรถกอล์ฟเดิม แต่เนื่องจากตัวหุ่นยนต์มีขนาดเล็กกว่า จึงไม่สามารถใช้คอมพิวเตอร์ประสิทธิภาพสูงได้เหมือนเดิม ทีมวิจัยจึงเลือกใช้หน่วยประมวลผลขนาดเล็กอย่าง NVIDIA Jetson AGX Xavier ซึ่งมีขนาดกะทัดรัดแต่ให้พลังการประมวลผลที่เหมาะสมสำหรับงานแบบเรียลไทม์ อย่างไรก็ตาม จากการทดสอบพบว่าการใช้โมเดลตรวจจับเดิมที่รวมข้อมูลจากกล้องและ LiDAR มีภาระการประมวลผลสูงเกินไปสำหรับบอร์ดนี้ โดยเฉพาะเมื่อต้องทำงานร่วมกับระบบนำทางที่ใช้ LiDAR อยู่แล้ว ทำให้ระบบตรวจจับล้มเหลวและไม่สามารถใช้งานได้จริง

ในการพัฒนาระบบที่ใช้ได้จริงบนหุ่นยนต์ขนาดเล็ก การเลือกใช้การตรวจจับวัตถุแบบสามมิติ (3D Object Detection) โดยใช้ข้อมูลจาก LiDAR เพียงอย่างเดียวจึงเป็นทางเลือกที่เหมาะสม เนื่องจากสามารถให้ข้อมูลเชิงลึกทั้งตำแหน่ง ระยะห่าง และรูปร่างของวัตถุในพื้นที่จริงได้แม่นยำกว่าการตรวจจับแบบสองมิติที่อาศัยเพียงกล้อง ซึ่งไม่สามารถประเมินความลึกหรือรูปร่างเชิงพื้นที่ได้อย่างมีประสิทธิภาพ โดยเฉพาะในสภาพแวดล้อมที่มีแสงน้อยหรือซับซ้อน เช่น ทางเดินในอาคารหรือพื้นที่ที่มีสิ่งกีดขวางหลากหลาย

ข้อมูลจาก LiDAR ยังสามารถประมวลผลในรูปแบบ Bird’s Eye View (BEV) ซึ่งเหมาะกับการนำมาใช้ร่วมกับโมเดล Deep Learning ที่เรียนรู้ตำแหน่งของวัตถุในพื้นที่แบบแผนที่ได้อย่างแม่นยำ ช่วยเพิ่มประสิทธิภาพในการหลบหลีกสิ่งกีดขวางในขณะเคลื่อนที่ และลดภาระของระบบโดยไม่ต้องอาศัยกล้องหรือการประมวลผลภาพเพิ่มเติม ซึ่งสอดคล้องกับแนวทางของหุ่นยนต์ที่ต้องการลดทรัพยากรการประมวลผลให้เหมาะสมกับขนาดและข้อจำกัดของฮาร์ดแวร์

จากการศึกษางานวิจัยพบว่าโมเดล PointPillars ที่พัฒนาโดย Zhou et al. (2018) เป็นหนึ่งในโมเดลที่เหมาะสมกับงานตรวจจับ 3 มิติจากข้อมูล LiDAR โดยใช้ Bird's Eye View ทำให้ลดภาระการประมวลผลและสามารถทำงานแบบเรียลไทม์ได้ อย่างไรก็ตามโมเดลดั้งเดิมของ PointPillars ได้รับการออกแบบเพื่อทำงานบนอุปกรณ์ประสิทธิภาพสูง ดังนั้นเพื่อตอบโจทย์การใช้งานบนหุ่นยนต์ขนาดเล็ก ผู้วิจัยจึงมีแนวคิดในการปรับปรุงสถาปัตยกรรมของโมเดลดังกล่าว เพื่อลดภาระการประมวลผลและเพิ่มประสิทธิภาพในการทำงานบนบอร์ด Jetson AGX Xavier อย่างเหมาะสม

1.2 วัตถุประสงค์ของการวิจัย

1) เพื่อปรับปรุงปัญญาประดิษฐ์สถาปัตยกรรม PointPillars
2) เพื่อใช้งานปัญญาประดิษฐ์ให้หุ่นยนต์จัดส่งอัตโนมัติสามารถตรวจจับวัตถุ 3 มิติ
3) เพื่อลดภาระการคำนวณของระบบตรวจจับ 3 มิติ ให้หุ่นยนต์จัดส่งอัตโนมัติ

1.3 ขอบเขตของงานวิจัย

1) พื้นที่ทดสอบชั้น G อาคาร F11 ห้องศูนย์ถ่ายทอดเทคโนโลยีอากาศยานไร้นักบิน
2) ใช้เซ็นเซอร์ Velodyne Lidar VLP-16
3) ใช้ค่า Confusion matrix ในการประเมินโมเดล
4) ประเภทของวัตถุที่ตรวจจับ คน,จักรยานยนต์,รถ,กรวยจราจร
5) วัตถุที่ตรวจจับจะหยุดนิ่ง
6) ตรวจจับวัตถุแต่ละประเภทในระยะ 2-7 เมตร
7) หุ่นยนต์อัตโนมัติจะใช้ความเร็วคงที่
8) หุ่นยนต์อัตโนมัติวิ่งตรงเข้าหาวัตถุ
9) ความแม่นยำในตรวจจับของแต่ละประเภทของวัตถุต้องไม่ต่ำกว่า 60%
10) ใช้บอร์ด Jetson AGX Xavier เป็นหน่วยประมวลผล

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) หุ่นยนต์จัดส่งอัตโนมัติสามารถใช้ปัญญาประดิษฐ์สถาปัตยกรรม PointPillars ที่ปรับปรุงได้

2) หุ่นยนต์จัดส่งอัตโนมัติสามารถลดภาระการคำนวณจากการใช้ปัญญาประดิษฐ์สถาปัตยกรรม PointPillars ที่ปรับปรุงได้

3) ความปลอดภัยที่เพิ่มขึ้นสำหรับทั้งหุ่นยนต์และสิ่งแวดล้อมโดยรอบ ด้วยการตรวจจับและจำแนกวัตถุอย่างแม่นยำ หุ่นยนต์จึงสามารถหลีกเลี่ยงการชนได้

บทที่ 2
ปริทัศน์วรรณกรรมและทฤษฎีที่เกี่ยวข้อง

งานวิจัยนี้มุ่งเน้นการศึกษาและปรับปรุงสถาปัตยกรรม PointPillars โดยประยุกต์ใช้ข้อมูลจากเซ็นเซอร์ LiDAR เพื่อพัฒนาระบบตรวจจับวัตถุสามมิติที่มีประสิทธิภาพและใช้ทรัพยากรคำนวณน้อย เหมาะสำหรับการใช้งานบนหุ่นยนต์จัดส่งอัตโนมัติ งานวิจัยได้ทำการปรับแต่งบริเวณพื้นที่พิจารณาข้อมูลหรือกริด ของ PointPillars โดยใช้การวิเคราะห์เชิงประจักษ์ของระยะการตรวจจับเพื่อจำกัดขอบเขตข้อมูลและลดภาระการประมวลผล จากนั้นจะใช้ปรับแต่งโครงข่ายประสาทเทียมแบบ Convolutional ที่มีโครงสร้างเป็นเครือข่ายพีระมิดความสนใจน้ำหนักเบา (Lightweight Attention Pyramid Network) เพื่อคัดกรองคุณลักษณะที่สำคัญและลดข้อมูลที่ไม่จำเป็น ผู้วิจัยได้ทำการศึกษาค้นคว้าวรรณกรรมและงานวิจัยต่างๆที่เกี่ยวข้องเพื่อใช้ในการวิจัย โดยอธิบายในหัวข้อต่อไปนี้

2.1 ทฤษฎีที่เกี่ยวข้อง

งานวิจัยนี้อาศัยพื้นฐานทางทฤษฎีจากหลายสาขาเพื่อพัฒนาระบบตรวจจับวัตถุสามมิติที่มีประสิทธิภาพ โดยใช้ข้อมูลจากเซ็นเซอร์ LiDAR ร่วมกับเทคนิคการเรียนรู้เชิงลึก (Deep Learning) เพื่อให้หุ่นยนต์ขนส่งอัตโนมัติสามารถรับรู้สภาพแวดล้อมและหลบหลีกสิ่งกีดขวางได้อย่างแม่นยำ การศึกษาครอบคลุมทั้งหลักการทำงานของหุ่นยนต์อัตโนมัติ การประมวลผลข้อมูลจาก LiDAR สถาปัตยกรรมการเรียนรู้ของเครื่องจักร เช่น PointPillars และโครงข่ายพีระมิดความสนใจน้ำหนักเบา (Lightweight Attention Pyramid Network) รวมถึงการวิเคราะห์เชิงประจักษ์ของระยะการตรวจจับ ระบบปฏิบัติการหุ่นยนต์ (ROS) และแพลตฟอร์มประมวลผลแบบฝังตัว NVIDIA Jetson Xavier AGX โดยองค์ความรู้ทั้งหมดนี้เป็นรากฐานสำคัญในการพัฒนาและปรับปรุงโมเดลตรวจจับวัตถุให้เหมาะสมกับข้อจำกัดของหุ่นยนต์อัตโนมัติในสภาพแวดล้อมจริง

2.1.1 หุ่นยนต์ขนส่งอัตโนมัติ

หุ่นยนต์จัดส่งอัตโนมัติ หรือ Delivery Robot เป็นหุ่นยนต์ที่ถูกพัฒนามาเพื่อใช้ขนส่งของจากจุดรับของไปที่จุดเป้าหมายที่จะส่งของ แบบอัตโนมัติโดยไม่มีการควบคุมจากคน ซึ่งบนหุ่นยนต์ได้ใส่เซนเซอร์วัดระยะด้วยแสง LiDAR เซนเซอร์วัดความเฉื่อย IMU ตัวประมวลผลกลางที่ใช้คำนวณและควบคุมการเคลื่อนที่ของหุ่นยนต์ ระบบช่วงล่างที่จะรับคำสั่งจากตัวประมวลผลกลางเพื่อสั่งมอเตอร์ให้ทำงาน ทำให้หุ่นยนต์ตัวนี้สามารถใช้ระบบอัตโนมัติที่รวมหลายระบบได้ เช่น ระบบระบุ

ตำแหน่ง ระบบนำทาง ระบบความปลอดภัย จึงทำให้หุ่นยนต์ตัวนี้สามารถขนส่งของได้อย่างแม่นยำและปลอดภัย ดังรูปที่ 2.1

รูปที่ 2.1 หุ่นยนต์ขนส่งอัตโนมัติ

2.1.2 เซ็นเซอร์ LiDAR

ไลดาร์ (Lidar) คือ ไลดาร์ (LiDAR) เป็นตัวย่อจากคำว่า “Light detection and ranging” เป็นเทคโนโลยีที่ใช้สำหรับการวัดระยะทางหรือความสูงของวัตถุและพื้นผิวโดยอาศัยหลักการสะท้อนของแสงเลเซอร์ ระบบจะปล่อยแสงเลเซอร์ออกไปยังวัตถุเป้าหมาย และเมื่อแสงสะท้อนกลับมายังตัวรับสัญญาณ (Receiver) ระบบจะคำนวณระยะทางโดยใช้ระยะเวลาในการเดินทางของแสงร่วมกับค่าความเร็วของแสงตามสมการ:

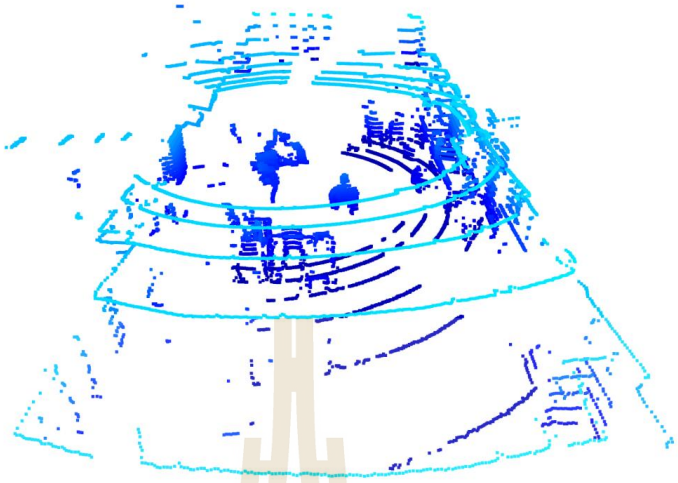
$$\text{ระยะทาง} = \frac{c \times t}{2}$$

โดยที่ **c** คือ ความเร็วของแสงเลเซอร์ และ t คือเวลาที่แสงเดินทางไปกลับจากแหล่งกำเนิดถึงวัตถุและกลับมายังตัวรับสัญญาณ เทคโนโลยี LiDAR มีความสามารถในการเก็บข้อมูลเชิงลึก (Depth Information**)** ได้อย่างแม่นยำ และได้รับความนิยมอย่างแพร่หลายในการใช้งานด้านระบบช่วยเหลือผู้ขับขี่ (ADAS), ยานยนต์ไร้คนขับ (Autonomous Vehicles) รวมถึงงานวิจัยด้านการรับรู้สภาพแวดล้อมของหุ่นยนต์อัตโนมัติ

รูปที่ 2.2 เซ็นเซอร์ LiDAR

ในภาพ 2.2 คือ เซ็นเซอร์ LiDAR ที่ใช้งานในระบบอัตโนมัติและหุ่นยนต์เคลื่อนที่แบบไร้คนขับ Velodyne VLP-16 (Velodyne LiDAR Inc, 2020) ถือเป็นเซ็นเซอร์ที่ได้รับความนิยมสูง เนื่องจากมีขนาดกะทัดรัด น้ำหนักเบา และสามารถให้ข้อมูลแบบ 3 มิติความละเอียดสูงแบบเรียลไทม์ โดย VLP-16 ใช้หลักการหมุนเลเซอร์ 360 องศาในแนวนอน และปล่อยลำแสงเลเซอร์ออกในแนวตั้งจำนวน 16 ลำ ซึ่งครอบคลุมมุมการมองในแนวตั้ง (Vertical Field of View) ประมาณ 30 องศา ตั้งแต่ -15° ถึง +15° ด้วยความถี่การสแกนสูงสุดที่ 20 Hz และสามารถวัดระยะทางได้ถึง 100 เมตรด้วยความแม่นยำระดับเซนติเมตร งานวิจัยโดย Zhang et al. (2017) ได้นำเซ็นเซอร์รุ่นนี้มาใช้ในระบบรับรู้ของหุ่นยนต์เคลื่อนที่เพื่อสร้างแผนที่ 3 มิติของสภาพแวดล้อมแบบเรียลไทม์ และพบว่าสามารถเพิ่มความแม่นยำในการระบุวัตถุและตรวจจับสิ่งกีดขวางได้อย่างมีประสิทธิภาพ แม้ในพื้นที่ที่มีความซับซ้อนสูง เช่น พื้นที่ที่มีการเคลื่อนไหวของมนุษย์หรือยานพาหนะ นอกจากนี้ VLP-16 ยังถูกนำมาใช้อย่างแพร่หลายในงานด้านการตรวจจับวัตถุ การประมาณระยะ และการระบุตำแหน่งในระบบ SLAM (Simultaneous Localization and Mapping) รวมถึงระบบยานยนต์ไร้คนขับ

พอยต์คลาวด์ (Point Cloud) คือชุดข้อมูลที่ประกอบด้วยตำแหน่งของจุดจำนวนมากในพื้นที่สามมิติ ซึ่งมักได้มาจากการวัดระยะด้วยเทคโนโลยีเลเซอร์ เช่น เลเซอร์สแกนเนอร์ (Laser Scanner) หรือไลดาร์ (LiDAR) โดยใช้หลักการปล่อยลำแสงเลเซอร์จากอุปกรณ์ส่งสัญญาณไปยังวัตถุ และคำนวณตำแหน่งของจุดจากระยะเวลาที่ลำแสงสะท้อนกลับมายังตัวรับสัญญาณ ข้อมูลพอยต์คลาวด์ที่ได้จะสามารถแสดงรูปร่าง พื้นผิว และโครงสร้างเชิงพื้นที่ของวัตถุต่าง ๆ ได้อย่างแม่นยำ ระบบเลเซอร์ในปัจจุบันสามารถวัดระยะได้ตั้งแต่ประมาณ 100 เมตร ไปจนถึงมากกว่า 300 เมตร ขึ้นอยู่กับรุ่นของเซ็นเซอร์ และสามารถสร้างจุดข้อมูลได้ในอัตราความถี่สูง ตั้งแต่ประมาณ 70,000 จุดต่อวินาที ไปจนถึงมากกว่า 1,000,000 จุดต่อวินาที ซึ่งส่งผลให้ได้ข้อมูลที่มีความละเอียดสูง เหมาะสำหรับนำไปใช้ในงานวิเคราะห์สามมิติ เช่น การสร้างแผนที่ การตรวจจับวัตถุ และการประมวลผลสภาพแวดล้อมในระบบหุ่นยนต์หรือยานยนต์ไร้คนขับ



รูปที่ 2.3 พอยต์คลาวด์

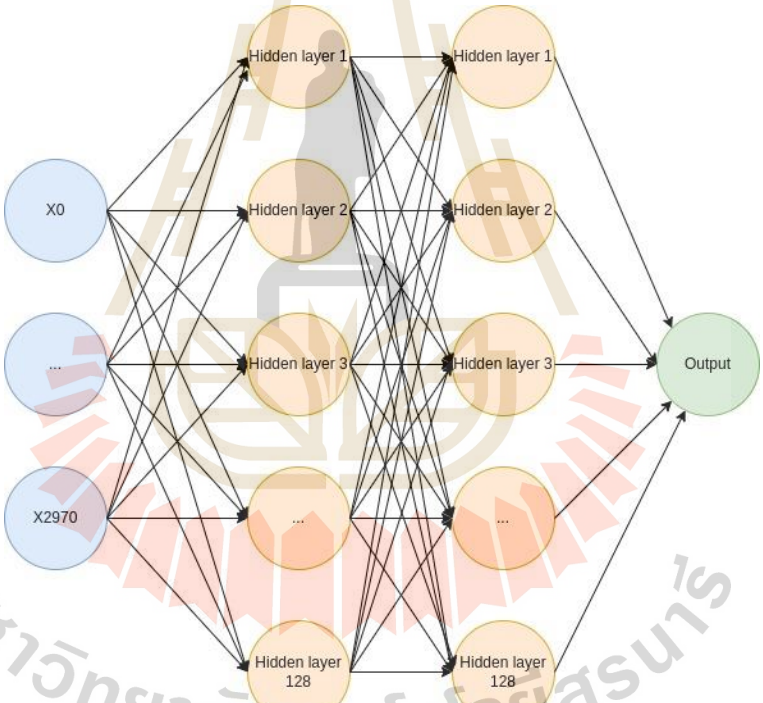
ในภาพที่ 2.3 จะแสดงกลุ่มพอยต์คลาวด์ที่ใช้เลเซอร์สแกนรอบทิศทางของเซนเซอร์ LiDAR บนซอฟต์แวร์ที่ใช้แสดงพอยต์คลาวด์

2.1.3 การเรียนรู้ของเครื่องจักร

การเรียนรู้ของเครื่อง (Machine Learning: ML) เป็นเทคโนโลยีหลักแขนงหนึ่งของปัญญาประดิษฐ์ (Artificial Intelligence: AI) ซึ่งมุ่งเน้นการเรียนรู้จากข้อมูลตัวอย่างหรือประสบการณ์ในอดีต โดยระบบจะสามารถเลือกคุณลักษณะที่สำคัญจากข้อมูลเหล่านั้นได้โดยอัตโนมัติภายใต้การออกแบบของมนุษย์ เมื่อได้รับการฝึกด้วยข้อมูลจำนวนมากเพียงพอ ระบบที่ผ่านการเรียนรู้จะสามารถประมวลผลและตัดสินใจกับข้อมูลใหม่ที่ไม่เคยพบมาก่อนได้อย่างมีประสิทธิภาพ ปัจจุบันการเรียนรู้ของเครื่องได้รับความนิยมอย่างแพร่หลายในหลากหลายสาขา โดยเฉพาะการเรียนรู้เชิงลึก (Deep Learning: DL) ซึ่งเป็นแขนงหนึ่งของการเรียนรู้ของเครื่องที่มีพื้นฐานจากโครงข่ายประสาทเทียม (Artificial Neural Networks) ซึ่งได้รับแรงบันดาลใจจากการจำลองกระบวนการทำงานของเซลล์ประสาทในระบบประสาทของมนุษย์ (ปริญญา สงวนสัตย์, 2562) โดยเฉพาะในงานด้านการประมวลผลภาพ โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks: CNN) ได้รับความนิยมสูง เนื่องจากสามารถเรียนรู้ลักษณะเชิงพื้นที่ของภาพได้อย่างมีประสิทธิภาพ และช่วยลดความซับซ้อนของข้อมูลภาพดิบให้เหมาะสมกับการนำไปวิเคราะห์ต่อไป

Artificial Neural Network หรือชื่อภาษาไทยคือโครงข่ายประสาทเทียม (Artificial Neural Network: ANN) คือรูปแบบหนึ่งของแบบจำลองทางคอมพิวเตอร์ที่ได้รับแรงบันดาลใจจาก

โครงสร้างและการทำงานของระบบประสาทในมนุษย์ โดยในแต่ละโหนด (Node) ของเครือข่ายจะสามารถรับข้อมูล ประมวลผล และส่งข้อมูลต่อไปยังโหนดถัดไปได้อย่างเป็นระบบ ANN ถือเป็นพื้นฐานสำคัญของเทคนิคการเรียนรู้เชิงลึก (Deep Learning: DL) ซึ่งในเชิงโครงสร้างสามารถอธิบายได้ว่า DL คือโครงข่ายประสาทเทียมที่มีหลายชั้น (Layers) เรียงซ้อนกัน ทำให้สามารถเรียนรู้ข้อมูลที่มีความซับซ้อนและนามธรรมได้ลึกยิ่งขึ้น ปัจจุบันเทคนิคที่พัฒนาโดยใช้โครงข่ายประสาทเทียมสามารถนำไปประยุกต์ใช้ในงานหลากหลายด้านภายใต้ขอบเขตของการเรียนรู้ของเครื่อง (Machine Learning) ไม่ว่าจะเป็นปัญหาที่มีขนาดใหญ่หรือซับซ้อนระดับสูง โดยเฉพาะในสาขาที่เกี่ยวข้องกับการประมวลผลภาพ (Computer Vision) และการประมวลผลภาษาธรรมชาติ (Natural Language Processing: NLP) ซึ่งโครงข่ายประสาทเทียมถือเป็นเทคนิคระดับแนวหน้า (State-of-the-art) ที่ได้รับความนิยมอย่างแพร่หลาย



รูปที่ 2.4 โครงสร้าง ANN

ในภาพที่ 2.4 แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบป้อนตรง (Feedforward Neural Network) ซึ่งประกอบด้วยชั้นอินพุตที่รับข้อมูลนำเข้าจำนวน 2,971 คุณลักษณะ (features) เช่น X_0 ถึง X_{2970} ข้อมูลเหล่านี้จะถูกส่งผ่านไปยังชั้นซ่อน (Hidden Layers) ซึ่งมีจำนวน 128 ชั้น โดยแต่ละชั้นประกอบด้วยนิวรอนที่เชื่อมโยงกันผ่านค่าน้ำหนัก (Weights) และกระทำการคำนวณผลรวมถ่วงน้ำหนักร่วมกับการผ่านฟังก์ชันกระตุ้น (Activation Function) เช่น

ReLU, Sigmoid หรือ Tanh เพื่อเพิ่มความไม่เชิงเส้นให้กับแบบจำลอง ส่งผลให้สามารถเรียนรู้ลักษณะเชิงลึกของข้อมูลได้ดีขึ้น ในระหว่างกระบวนการฝึกโมเดล (Training) ระบบจะทำการปรับค่าน้ำหนักโดยใช้กระบวนการถอยกลับเพื่อปรับพารามิเตอร์ (Backpropagation) ร่วมกับเทคนิคการหาค่าต่ำสุดของฟังก์ชันสูญเสีย (Loss Function) เช่น Gradient Descent จนกระทั่งแบบจำลองสามารถเรียนรู้และให้ผลลัพธ์ที่แม่นยำยิ่งขึ้น และในท้ายที่สุด ข้อมูลที่ผ่านทุกชั้นจะส่งต่อไปยังชั้นเอาต์พุต (Output Layer) เพื่อให้ผลลัพธ์สุดท้ายของการทำนายตามลักษณะของปัญหาที่ต้องการ เช่น การจำแนก (Classification) หรือการประมาณค่า (Regression)

โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks: CNNs) (Chauhan et al., 2018)เป็นหนึ่งในสถาปัตยกรรมหลักที่ได้รับความนิยมอย่างแพร่หลายในงานด้านการประมวลผลภาพ (Image Processing) และมีบทบาทสำคัญในการสกัดคุณลักษณะ (Feature Extraction) จากข้อมูลภาพและข้อมูลสามมิติ เช่น พอยต์คลาวด์ (Point Cloud) โดย CNN สามารถเรียนรู้ลักษณะเฉพาะเชิงลึกจากข้อมูลที่มีความซับซ้อนได้อย่างมีประสิทธิภาพ ซึ่งนำไปสู่การพัฒนาเทคนิคที่สามารถประยุกต์ใช้ CNN กับข้อมูล Point Cloud ได้โดยตรง เช่น PointNet (Qi et al., 2017) ซึ่งเป็นแนวทางแรกที่ออกแบบให้สามารถประมวลผลจุดในอวกาศสามมิติได้โดยไม่ต้องแปลงเป็นรูปภาพ หรือ PointCNN (Li et al., 2018) ที่ปรับโครงสร้างการคอนโวลูชันให้เหมาะกับการเรียงลำดับข้อมูลแบบไม่สม่ำเสมอของ Point Cloud รวมถึง PV-RCNN (Shi et al., 2020) ที่ผสานแนวคิดของ PointNet กับ CNN เชิงปริภูมิ (Voxel-based CNN) เพื่อให้ได้ทั้งประสิทธิภาพในการสกัดคุณลักษณะเชิงลึกและความแม่นยำในการตรวจจับวัตถุแบบสามมิติ โดยกระบวนการทำงานของ CNN เริ่มจากการนำข้อมูลนำเข้า ไม่ว่าจะเป็นภาพหรือพอยต์คลาวด์ ผ่านเลเยอร์คอนโวลูชัน (Convolution Layer) ซึ่งจะเรียนรู้และดึงคุณลักษณะที่สำคัญของข้อมูล จากนั้นจะผ่านกระบวนการลดขนาด (Pooling) เช่น Max Pooling หรือ Average Pooling เพื่อลดมิติของข้อมูล ลดการใช้ทรัพยากรในการประมวลผล และป้องกันการเกิด overfitting หลังจากผ่านกระบวนการสกัดคุณลักษณะแล้ว ข้อมูลจะถูกส่งผ่านเข้าสู่เลเยอร์แบบเต็ม (Fully Connected Layer) หรือทำการจัดเรียงข้อมูลให้อยู่ในรูปแบบเวกเตอร์หนึ่งมิติด้วยเลเยอร์ Flatten และใช้ฟังก์ชันกระตุ้น (Activation Function) เช่น Softmax เพื่อแปลงค่าให้เป็นความน่าจะเป็นสำหรับการจำแนกประเภท ซึ่งในท้ายที่สุดจะได้ผลลัพธ์ที่ชัดเจนจากเลเยอร์เอาต์พุต (Output Layer) เพื่อใช้ในการตัดสินใจหรือการทำนายในแต่ละประเภทของวัตถุ แนวทางดังกล่าวถือเป็นรากฐานสำคัญของระบบการตรวจจับวัตถุอัตโนมัติสมัยใหม่ โดยเฉพาะในแอปพลิเคชันที่ต้องการความแม่นยำและประสิทธิภาพสูง เช่น ระบบขับเคลื่อนอัตโนมัติ หุ่นยนต์เคลื่อนที่ และระบบประมวลผลข้อมูลจากกล้องหรือเซ็นเซอร์ LiDAR

เพื่อประเมินความแม่นยำของโมเดลในการเรียนรู้ของเครื่อง (Machine Learning) และการวิเคราะห์ข้อมูลในโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks: CNNs) มีเครื่องมือและเทคนิคหลายประเภทที่สามารถนำมาใช้ในการวัดประสิทธิภาพของโมเดล โดยหนึ่งในเครื่องมือพื้นฐานและได้รับความนิยมสูงคือ Confusion Matrix ซึ่งเป็นตารางที่แสดงผลการทำนายของโมเดลในแต่ละคลาส โดยแบ่งออกเป็นสี่ประเภท ได้แก่ True Positive (TP), False Positive (FP), True Negative (TN), และ False Negative (FN) ซึ่งค่าทั้งหมดนี้เกิดจากการเปรียบเทียบระหว่างผลลัพธ์ที่โมเดลทำนายกับค่าความจริง (Ground Truth) โดยทั่วไปตาราง Confusion Matrix จะมีลักษณะดังนี้:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

การวิเคราะห์จาก Confusion Matrix สามารถนำไปใช้คำนวณตัวชี้วัดอื่น ๆ ที่สำคัญ เช่น Accuracy, Precision, Recall และ F1-score ซึ่งช่วยสะท้อนถึงประสิทธิภาพของโมเดลในมิติต่าง ๆ (Powers, 2011) ทั้งในด้านความถูกต้องโดยรวม ความสามารถในการระบุวัตถุเป้าหมาย และการลดความผิดพลาดในการทำนาย โดยเฉพาะอย่างยิ่งในกรณีที่ข้อมูลไม่สมดุล การใช้ Confusion Matrix จะช่วยให้สามารถระบุจุดอ่อนของโมเดลในแต่ละคลาสได้อย่างชัดเจน และนำไปสู่การปรับปรุงโมเดลให้มีประสิทธิภาพยิ่งขึ้น

Precision (ค่าความแม่นยำ) หมายถึงสัดส่วนของจำนวนตัวอย่างที่โมเดลทำนายว่าเป็นบวก (Positive) และทำนายถูกจริง เมื่อเทียบกับจำนวนทั้งหมดที่โมเดลทำนายว่าเป็นบวก ซึ่งคำนวณได้จาก

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

โดยที่ TP (True Positives) คือ จำนวนตัวอย่างที่เป็นบวกจริงและโมเดลทำนายถูกต้อง, ส่วน FP (False Positives) คือจำนวนตัวอย่างที่เป็นลบแต่โมเดลทำนายผิดว่าเป็นบวก

Recall (ค่าความไว) หมายถึงสัดส่วนของตัวอย่างที่เป็นบวกจริงซึ่งโมเดลสามารถตรวจจับได้อย่างถูกต้อง เมื่อเทียบกับจำนวนตัวอย่างที่เป็นบวกทั้งหมด โดยคำนวณได้จาก

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

เมื่อ FN (False Negatives) คือ จำนวนตัวอย่างที่เป็นบวกจริงแต่โมเดลทำนายผิดว่าเป็นลบ

F1-Score (Sasaki, 2007) เป็นค่าเฉลี่ยเชิงฮาร์มอนิกส์ของ Precision และ Recall ซึ่งใช้วัดความสมดุลระหว่างสองค่าดังกล่าว โดยเฉพาะในกรณีที่ข้อมูลมีการกระจายไม่สมดุล (imbalanced dataset) โดยคำนวณจากสูตร

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

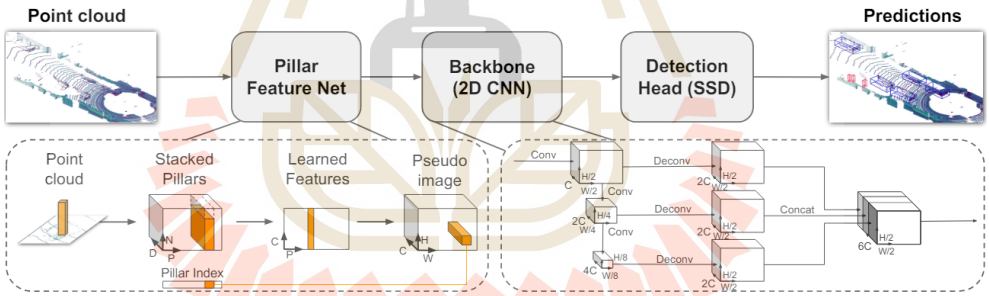
สำหรับการประเมินผลในงานตรวจจับวัตถุ (Object Detection) โดยเฉพาะในชุดข้อมูลมาตรฐาน เช่น COCO และ KITTI dataset มักนิยมใช้ค่าเฉลี่ยของ Average Precision (AP) หรือที่เรียกว่า Mean Average Precision (mAP) ซึ่งเป็นตัวชี้วัดที่สะท้อนประสิทธิภาพของโมเดลในแต่ละคลาส โดย Average Precision (AP) คำนวณจากพื้นที่ใต้กราฟ Precision-Recall (PR Curve) ของแต่ละคลาส และนำค่า AP เหล่านั้นมาหาค่าเฉลี่ยเพื่อได้ค่า mAP โดยสูตร คือ

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

โดยที่ N คือ จำนวนคลาสทั้งหมด และ AP_i คือค่า Average Precision ของคลาสที่ i สำหรับชุดข้อมูล COCO จะใช้การประเมินแบบ mAP@IoU = 0.5:0.95 ซึ่งหมายถึงการคำนวณค่า AP ซ้ำหลายครั้งที่ค่าเกณฑ์การทับซ้อนของวัตถุ (Intersection over Union: IoU) ตั้งแต่ 0.50 ถึง 0.95 โดยเพิ่มทีละ 0.05 แล้วหาค่าเฉลี่ย ส่วนใน KITTI dataset ซึ่งถูกออกแบบมาสำหรับการประเมินระบบรับรู้ในยานยนต์ไร้คนขับ จะใช้ค่าเกณฑ์ IoU = 0.7 สำหรับคลาส Car และ IoU = 0.5 สำหรับคลาส Pedestrian และ Cyclist ตามเกณฑ์มาตรฐานของการจัดอันดับในชุดข้อมูลนี้ (Geiger et al., 2012; KITTI Benchmark, 2023) ซึ่งสะท้อนถึงระดับความแม่นยำที่ต้องการสำหรับวัตถุแต่ละประเภทในสภาพแวดล้อมจริง

2.1.4 สถาปัตยกรรม PointPillars

PointPillars (Lang et al., 2019) เป็นหนึ่งในวิธีการที่ได้รับความนิยมในการตรวจจับวัตถุแบบสามมิติจากข้อมูลพอยต์คลาวด์ (Point Cloud) โดยมีแนวคิดหลักในการแปลงข้อมูลสามมิติให้อยู่ในรูปแบบสองมิติผ่านการสร้างมุมมองจากด้านบน หรือ Bird's Eye View (BEV) เพื่อลดความซับซ้อนของข้อมูลและลดภาระในการประมวลผล ซึ่งแนวคิดการใช้ BEV ในการประมวลผลข้อมูลพอยต์คลาวด์ได้รับการเสนอและพัฒนาอย่างแพร่หลายในงานวิจัยก่อนหน้า เช่น ในงานของ Chen et al. (2017) ที่นำเสนอโมเดล MV3D โดยผสานข้อมูลจากกล้องและ LiDAR ผ่านการแปลงเป็นภาพแบบ BEV เพื่อเพิ่มความแม่นยำในการตรวจจับวัตถุในมุมมองสามมิติ วิธีการของ PointPillars จะจัดโครงสร้างของข้อมูลพอยต์คลาวด์ให้อยู่ในรูปของ พิลลาร์ (Pillar) ซึ่งแต่ละเสาเป็นตัวแทนของข้อมูลในพื้นที่เฉพาะ จากนั้นนำข้อมูลพิลลาร์เหล่านี้เข้าสู่โครงข่ายประสาทเทียมแบบคอนโวลูชัน (CNN) เพื่อสกัดคุณลักษณะสำคัญ และส่งผลลัพธ์ที่ได้เข้าสู่โครงข่ายประสาทเทียมแบบทั่วไป (ANN) สำหรับกระบวนการจำแนกประเภท (Classification) และการระบุตำแหน่งเชิงสามมิติ (3D Bounding Box Regression) ของวัตถุที่ตรวจจับได้ โดยอัลกอลิทึม Pointpillar มีกระบวนการทำงานดังนี้



รูปที่ 2.5 โครงสร้างของ PointPillars (Lang, A. H., Vora, S., Caesar, H., Zhou, O. 2019.)

จากภาพที่ 2.5 กระบวนการทำงานของอัลกอริธึม PointPillars เริ่มต้นจากการนำเข้าข้อมูลพอยต์คลาวด์ที่ได้จากการสแกนด้วยเซนเซอร์ LiDAR ซึ่งมีลักษณะเป็นชุดของจุดสามมิติในเชิงพิกัด (x,y,z) พร้อมด้วยคุณลักษณะอื่น ๆ เช่นค่าความเข้ม (Intensity) ข้อมูลนี้จะถูกประมวลผลในโมดูลแรกคือ Pillar Feature Net (PFN) ซึ่งมีหน้าที่แปลงข้อมูลพอยต์คลาวด์แบบสามมิติให้เป็นรูปแบบสองมิติหรือมุมมองจากด้านบน (Bird's Eye View: BEV) เพื่อให้เหมาะสมกับการประมวลผลด้วยโครงข่ายคอนโวลูชัน 2 มิติ (2D CNN)

กระบวนการใน PFN เริ่มจากการกำหนด ขอบเขตการตรวจจับ (Detection Range) และ ขนาดของเวกเซล (Voxel Size) ตามแกน x,y,z เพื่อแบ่งพื้นที่การตรวจจับออกเป็นกริด

สม่ำเสมอในแนวระนาบ โดยข้อมูลในแต่ละกริดจะถูกรวมกันเป็นหน่วยข้อมูลที่เรียกว่า **Pillar** ซึ่งเป็นตัวแทนของข้อมูลในแนวตั้งของพื้นที่นั้น ๆ โดยจะเลือกเฉพาะ Pillar ที่มีจุดอยู่จริงเท่านั้น จากนั้นข้อมูลในแต่ละ Pillar จะถูกจัดเรียงใหม่เป็นเทนเซอร์สามมิติที่มีโครงสร้าง [D,P,N][D, P, N][D,P,N] ซึ่ง D คือ จำนวนฟีเจอร์ต่อจุด เช่น [x,y,z,intensity,Δx,Δy,Δz], P คือจำนวน Pillar ทั้งหมดในฉาก และ N คือ จำนวนจุดสูงสุดต่อ Pillar ที่กำหนดไว้ล่วงหน้า

ถัดจากนั้น แต่ละ Pillar จะถูกป้อนเข้าสู่ชั้นเรียนรู้ฟีเจอร์แบบ PointNet ซึ่งทำการรวมข้อมูลของจุดภายใน Pillar ด้วยการใช้ฟังก์ชัน Max Pooling เพื่อให้ได้ฟีเจอร์รวมของเสานั้นในรูปของเวกเตอร์ขนาด C (เช่น 64 ช่องสัญญาณ) ผลลัพธ์ที่ได้จะอยู่ในรูปเทนเซอร์ขนาด [C,P] ซึ่งแต่ละเวกเตอร์แทนฟีเจอร์ของเสาหนึ่งต้น

ในขั้นตอนถัดไป ข้อมูล [C,P] จะถูกแปลงกลับให้อยู่ในตำแหน่งที่สอดคล้องกับพิกัด x,y ของแต่ละ Pillar ในแผนที่ BEV ซึ่งจะสร้างเป็นภาพเทียม (Pseudo Image) ขนาด [C,H,W] ที่ H และ W คือจำนวนเซลล์ในแนวแกน y และ x ตามลำดับ ขึ้นอยู่กับค่าของ Point Cloud Range และ Voxel Size ที่กำหนด ตัวอย่างเช่น หากพื้นที่ครอบคลุมถูกแบ่งเป็นกริดขนาด 400×400 ช่อง และใช้ฟีเจอร์ขนาด C=64 จะได้ภาพ Pseudo Image ขนาด [64,400,400] ซึ่งสามารถนำไปผ่านโครงข่ายคอนโวลูชันแบบสองมิติต่อไปได้ สำหรับพิกัดที่ไม่มีข้อมูล Pillar อยู่ ค่าฟีเจอร์จะถูกแทนด้วยศูนย์ ทั้งหมดนี้คือกระบวนการในโมดูล PointPillars Feature Net ซึ่งเป็นขั้นตอนสำคัญในการแปลงข้อมูลพอยต์คลาวด์ไปสู่รูปแบบที่สามารถใช้ร่วมกับโครงข่ายประสาทเทียมแบบภาพได้อย่างมีประสิทธิภาพ

หลังจากผ่านขั้นตอนการเข้ารหัสฟีเจอร์ในโมดูล Pillar Feature Net แล้ว ข้อมูลจะอยู่ในรูปของเทนเซอร์ขนาด [C,H,W] ซึ่งเรียกว่า Pseudo Image ข้อมูลนี้จะถูกป้อนเข้าสู่โมดูล Backbone 2D Convolutional Neural Network (2D CNN) ซึ่งประกอบด้วยชุดของเลเยอร์คอนโวลูชัน (Conv2D) หลายชั้นที่ทำหน้าที่เรียนรู้คุณลักษณะเชิงลึกของข้อมูล ในแต่ละชุดของเลเยอร์จะมีการใช้กระบวนการ Downsampling ผ่านการคอนโวลูชันที่มี stride > 1 เพื่อลดขนาดของฟีเจอร์แมป และในขณะเดียวกันเพิ่มจำนวน channel ของฟีเจอร์เพื่อลดความซับซ้อนเชิงพื้นที่และเพิ่มความสามารถในการเรียนรู้ฟีเจอร์ระดับสูง จากนั้นจะทำการ Upsampling ด้วยเทคนิค Deconvolution (หรือ Transposed Convolution) เพื่อนำฟีเจอร์แมปกลับมาใกล้เคียงขนาดเดิม โดย Backbone แบ่งออกเป็น 3 กลุ่มของบล็อก (Block 0–2) ซึ่งฟีเจอร์ที่ได้จากแต่ละบล็อกจะถูก Upsample ให้มีขนาดเดียวกัน ([2C,H/2,W/2] แล้วนำมารวมกัน (Concatenate) ในมิติของ channel เพื่อได้เทนเซอร์สุดท้ายขนาด [6C,H/2,W/2] ขั้นตอนนี้มีวัตถุประสงค์เพื่อนำฟีเจอร์จากหลายระดับความละเอียดมาใช้ร่วมกัน ซึ่งช่วยให้โมเดลสามารถตรวจจับวัตถุที่มีขนาดหลากหลายได้อย่างมีประสิทธิภาพ ทั้งวัตถุขนาดเล็กที่ต้องการรายละเอียดสูง และวัตถุขนาดใหญ่ที่ต้องการบริบท

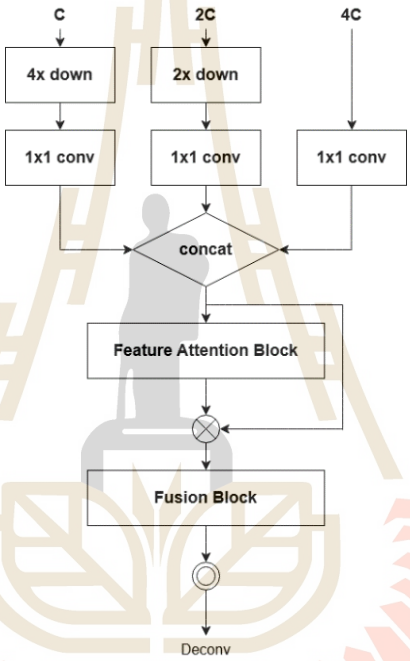
ภาพที่กว้างขึ้น นอกจากนี้ยังช่วยรับมือกับลักษณะที่เปลี่ยนแปลงของข้อมูลพอยต์คลาวด์ ซึ่งอาจเกิดจากลักษณะการสแกนของเซนเซอร์ LiDAR ที่เปลี่ยนไปในแต่ละรอบ กระบวนการนี้จึงมีบทบาทสำคัญในการเพิ่มความแม่นยำของโมเดลก่อนที่จะเข้าสู่ขั้นตอนถัดไปคือ Detection Head สำหรับการคาดการณ์ผลลัพธ์การจำแนกและระบุตำแหน่งวัตถุในรูปแบบสามมิติ

สุดท้ายของสถาปัตยกรรม PointPillars คือโมดูล Detection Head ซึ่งทำหน้าที่ในการจำแนกประเภทของวัตถุและคาดการณ์พิกัดของกล่องสามมิติที่ล้อมรอบวัตถุในฉาก กระบวนการนี้จะรับข้อมูลเทนเซอร์จาก Backbone 2D CNN ซึ่งเป็นฟีเจอร์เชิงลึกในรูปแบบภาพมุมมอง Bird's Eye View โดย Detection Head ที่ใช้ใน PointPillars มีพื้นฐานจากสถาปัตยกรรม Single Shot Detector (SSD) (W. Liu et al., 2016) ซึ่งเป็นเทคนิคที่สามารถทำการจำแนกและประมาณตำแหน่งวัตถุได้ในขั้นตอนเดียว (single forward pass) โดยไม่ต้องแยกกระบวนการออกเป็นสองส่วน เช่น Region Proposal และ Classification ดังใน Faster R-CNN (Ren et al., 2015) การทำงานของ SSD Detection Head จะใช้เลเยอร์คอนโวลูชันหลายระดับในการประมวลผลฟีเจอร์ที่มีความละเอียดแตกต่างกัน เพื่อตรวจจับวัตถุที่มีขนาดหลากหลาย โดยมีการลดขนาดของภาพฟีเจอร์ (feature map) ลงเรื่อย ๆ ในแต่ละเลเยอร์ ซึ่งช่วยให้ครอบคลุมบริบทของวัตถุขนาดใหญ่ ขณะที่เลเยอร์ที่มีความละเอียดสูงสามารถใช้ตรวจจับวัตถุขนาดเล็กได้ดี ในแต่ละตำแหน่งของฟีเจอร์แมป โมเดลจะทำนายพิกัดของกล่อง (bounding box regression) และค่าความน่าจะเป็นของแต่ละคลาส (classification scores) และจะมีการใช้กระบวนการ Non-Maximum Suppression (NMS) เพื่อตัดกล่องที่ซ้อนกันและเลือกเฉพาะกล่องที่มีคะแนนสูงสุดเท่านั้น การรวมฟีเจอร์จากหลายสเกลเช่นนี้ช่วยเพิ่มประสิทธิภาพของโมเดลในการตรวจจับวัตถุที่หลากหลายทั้งในแง่ของขนาดและความซับซ้อนเชิงพื้นที่

2.1.5 โครงข่ายพีระมิดความสนใจ (Lightweight Attention Pyramid Network)

ในงานวิจัยนี้มีการประยุกต์ใช้เครือข่ายพีระมิดความสนใจน้ำหนักเบา (Lightweight Attention Pyramid Network: LAPN) ซึ่งเป็นโครงข่ายประสาทเทียมที่ได้รับการออกแบบมาเพื่อรองรับภารกิจด้านการตรวจจับวัตถุ (Object Detection) และการแบ่งแยกวัตถุเฉพาะราย (Instance Segmentation) โดย LAPN ถูกพัฒนาขึ้นเพื่อตอบโจทย์ข้อจำกัดของแนวทางการสร้างฟีเจอร์หลายระดับแบบดั้งเดิม ไม่ว่าจะเป็นการใช้ภาพหลายสเกลในแนวทาง Featurized Image Pyramid ที่แม้จะรองรับวัตถุหลากหลายขนาดได้ดี แต่กลับมีต้นทุนการคำนวณสูง หรือแนวทาง Single Feature Map ที่ประหยัดทรัพยากรแต่ไม่รองรับการตรวจจับวัตถุขนาดเล็กและใหญ่ได้พร้อมกัน อีกทั้ง Pyramidal Feature Hierarchy แม้จะสร้างฟีเจอร์จากหลายชั้นแต่ก็เพิ่มความซับซ้อนของโมเดล และ Deep Supervision ที่ช่วยเพิ่มประสิทธิภาพการเรียนรู้แต่ยังคงมีข้อจำกัดด้านการรวมข้อมูลข้ามระดับ ส่วน Feature Pyramid Network (FPN) (Lin et al., 2017) นั้น แม้จะได้รับ

ความนิยมสูงในการรวมฟีเจอร์แบบ top-down ร่วมกับ lateral connections แต่การรวมฟีเจอร์จำกัดอยู่เพียงชั้นที่อยู่ติดกันเท่านั้น จากข้อจำกัดดังกล่าว LAPN จึงถูกเสนอขึ้นโดย Zhang et al. (2020) ซึ่งเพิ่มกลไกความสนใจ (Attention Mechanism) เข้ามาในโครงสร้างพีระมิดเพื่อให้โครงข่ายสามารถเลือกให้ความสำคัญกับฟีเจอร์ในแต่ละระดับได้อย่างเหมาะสมผ่านโมดูล Feature Attention Fusion Block (FAFB) ส่งผลให้ LAPN สามารถสกัดคุณลักษณะได้อย่างมีประสิทธิภาพจากหลายระดับความละเอียดภายในเครือข่ายเดียว ทั้งยังลดขนาดโมเดลและภาระการคำนวณลงอย่างมีนัยสำคัญ



รูปที่ 2.6 โครงสร้าง LAPN

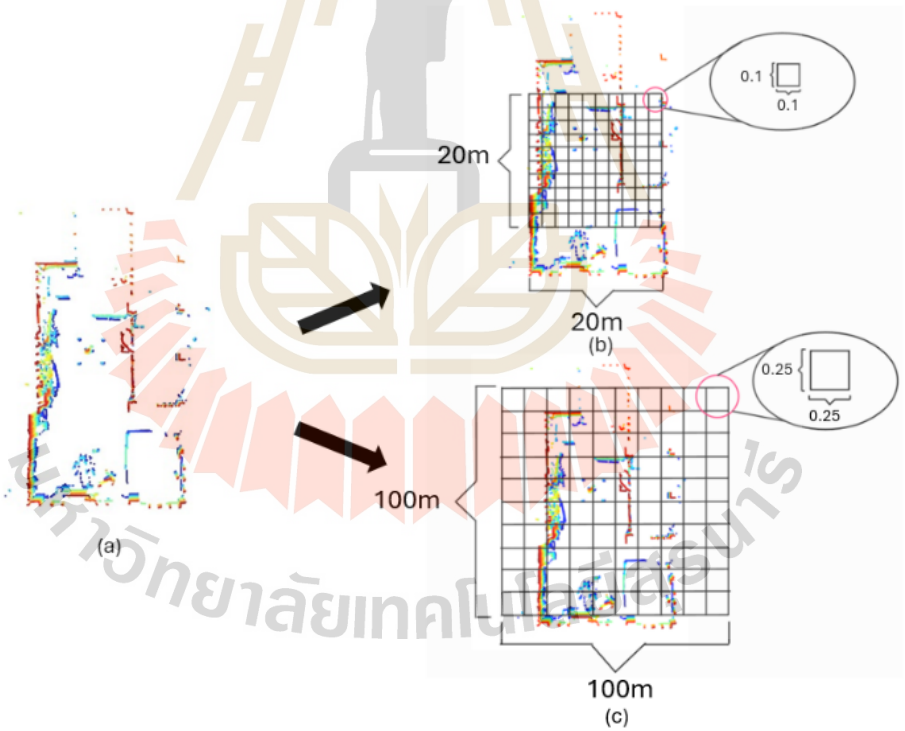
จากภาพที่ 2.6 คือ โครงข่าย LAPN ประกอบด้วยองค์ประกอบสำคัญ 2 ส่วน ได้แก่ หน่วยความสนใจต่อคุณลักษณะ (Feature Attention Block): โมดูลนี้ทำหน้าที่เน้นช่องสัญญาณ (feature channels) ที่มีความสำคัญ โดยลดความสำคัญของช่องสัญญาณที่ไม่เกี่ยวข้องลง กลไกนี้ช่วยให้เครือข่ายสามารถโฟกัสไปที่ข้อมูลที่มีประโยชน์มากที่สุดในแต่ละชั้นของพีระมิด ส่งผลให้การสกัดคุณลักษณะมีความแม่นยำมากยิ่งขึ้น โดยเฉพาะอย่างยิ่งในสภาพแวดล้อมที่มีวัตถุจำนวนมากหรือมีความซับซ้อนสูง

การรวมคุณลักษณะหลายระดับ (Feature Fusion): เครือข่ายจะรวมข้อมูลจากชั้นความละเอียดต่ำ (low-level features) ซึ่งมีข้อมูลเชิงพื้นที่ที่ละเอียด เข้ากับข้อมูลจากชั้นความ

ละเอียดสูง (high-level features) ซึ่งมีข้อมูลเชิงนามธรรมที่สื่อถึงบริบททั่วไป การหลอมรวมคุณลักษณะทั้งสองระดับนี้ช่วยให้ระบบสามารถจดจำวัตถุได้อย่างมีประสิทธิภาพ แม้ว่าวัตถุจะมีขนาด รูปร่าง หรือมุมมองที่แตกต่างกัน

2.1.6 การวิเคราะห์เชิงประจักษ์ของระยะการตรวจจับ (An Empirical Analysis of Detection Range)

จากงานวิจัยของ Peri และคณะ (2023). ได้เสนอแนวทางที่เรียกว่า Range Expert ได้รับการเสนอเพื่อเพิ่มประสิทธิภาพของโมเดลโดยการปรับค่า ระยะตรวจจับ (Detection Range) และ ความละเอียดของเวกเซล (Voxel Resolution) ให้เหมาะสมกับลักษณะของงานในแต่ละระยะ ซึ่งสามารถกำหนดได้ในระหว่างขั้นตอนการกำหนดค่าระบบ (Configuration Phase) กลยุทธ์นี้ช่วยให้สามารถออกแบบระบบที่ตอบสนองต่อภารกิจเฉพาะ เช่น การหลีกเลี่ยงสิ่งกีดขวางระยะใกล้หรือการตรวจการณ์ระยะไกล โดยการเลือกค่าพารามิเตอร์ให้สอดคล้องกับความต้องการของระบบ ส่งผลให้เพิ่มทั้งความแม่นยำและประสิทธิภาพในการประมวลผลของโมเดล



รูปที่ 2.7 การปรับระยะการตรวจจับ

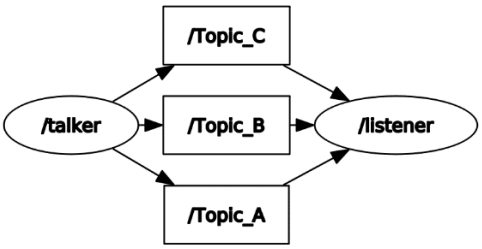
ดังที่แสดงในภาพที่ 2.7(a) ซึ่งเป็นข้อมูลพอยต์คลาวด์ที่ได้จากเซนเซอร์ LiDAR ในพื้นที่ทดสอบ สำหรับกรณีที่เน้นการตรวจจับวัตถุระยะใกล้ เช่น การหลีกเลี่ยงการชนในหุ่นยนต์

เคลื่อนที่อัตโนมัติ จะเลือกใช้เวกเซลขนาดเล็ก เช่น 0.1 เมตร และจำกัดระยะตรวจจับไว้ที่ 20 เมตร ตามภาพที่ 2.6(b) เพื่อให้สามารถสกัดรายละเอียดของวัตถุขนาดเล็กหรือการเคลื่อนไหวได้แม่นยำ และมุ่งเน้นการใช้ทรัพยากรคำนวณเฉพาะพื้นที่สำคัญใกล้ตัว ในขณะที่สำหรับงานที่ต้องการตรวจสอบสิ่งแวดล้อมระยะไกล เช่น การตรวจจับยานพาหนะที่อยู่ห่างออกไป จะเลือกใช้เวกเซลขนาดใหญ่กว่า (0.25 เมตร) และขยายระยะตรวจจับเป็น 100 เมตร ตามภาพที่ 2.6(c) เพื่อควบคุมภาระการคำนวณโดยรวม วิธีการนี้ช่วยให้สามารถสร้างจุดสมดุลระหว่าง ความละเอียดเชิงพื้นที่ และขอบเขตการตรวจจับ ทำให้สามารถทำงานแบบเรียลไทม์ได้ดีในระบบที่มีข้อจำกัดด้านทรัพยากร (Peri et al., 2023)

2.1.7 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System – ROS)

Robot Operating System (ROS) เป็นเฟรมเวิร์กที่ได้รับความนิยมอย่างแพร่หลายในการพัฒนาและควบคุมระบบหุ่นยนต์ โดย ROS ไม่ใช่ระบบปฏิบัติการในเชิงเทคนิค แต่เป็นชุดของไลบรารี เครื่องมือ และโครงสร้างพื้นฐานที่ช่วยสนับสนุนการสร้างซอฟต์แวร์สำหรับหุ่นยนต์นี้ออกแบบมาเพื่ออำนวยความสะดวกในการพัฒนาแอปพลิเคชันหุ่นยนต์ผ่านแนวคิดการทำงานแบบแยกหน้าที่เป็นหน่วยย่อยที่เรียกว่า Nodes ซึ่งแต่ละ Node จะทำหน้าที่เฉพาะด้าน เช่น การควบคุมเซนเซอร์ การประมวลผลภาพ หรือการเคลื่อนที่ โดย Nodes ต่าง ๆ สามารถสื่อสารกันผ่านระบบที่เรียกว่า Topics ซึ่งทำหน้าที่เป็นช่องทางกลางในการแลกเปลี่ยนข้อมูลผ่านข้อความ (Messages) หน่วยที่ส่งข้อมูลจะถูกเรียกว่า Publisher ขณะที่หน่วยที่รับข้อมูลจะเรียกว่า Subscriber โดยระบบสามารถมีหลาย Topic และหลาย Node ทำงานร่วมกันได้แบบขนาน (asynchronous) ช่วยให้สามารถพัฒนาและทดสอบระบบย่อยแต่ละส่วนได้อย่างอิสระและมีความยืดหยุ่นสูง

ROS ได้รับการพัฒนาอย่างต่อเนื่องในหลายเวอร์ชัน โดยเฉพาะ ROS Melodic Morenia ซึ่งเป็นเวอร์ชันที่เสถียรสำหรับระบบ Ubuntu 18.04 และเหมาะสำหรับระบบฝังตัว และ ROS Noetic Ninjemys ซึ่งเป็นเวอร์ชัน LTS รุ่นสุดท้ายใน ROS 1 ที่ออกแบบมาสำหรับ Ubuntu 20.04 โดยเน้นการสนับสนุน Python 3 และมีการปรับปรุงด้านประสิทธิภาพและความปลอดภัยที่ดีขึ้น ช่วยเพิ่มความเหมาะสมต่อการใช้งานในระบบหุ่นยนต์อัตโนมัติสมัยใหม่ โดยเฉพาะในงานที่ต้องการการทำงานร่วมกันของโมดูลหลายตัว เช่น ระบบนำทาง การหลีกเลี่ยงสิ่งกีดขวาง และการตรวจจับวัตถุแบบเรียลไทม์ (Quigley et al., 2009; ROS Wiki, 2020a; ROS Wiki, 2020b) โดยหลักการทำงานโดยสังเขปของ ROS สามารถอธิบายได้ดังรูปที่ 2.7



รูปที่ 2.8 แผนภาพอธิบายการทำงานของ ROS

การส่งข้อความในรูปแบบ node ของ ROS ดังที่กล่าวมาข้างต้นนั้นจะช่วยให้การทำงานง่ายขึ้นเนื่องจากไม่จำเป็นต้องเขียนโค้ดควบคุมการทำงานของแต่ละส่วนรวมกันไว้ในโค้ดเดียว ดังนั้นการทำงานจึงสามารถทำได้อย่างสะดวกและรวดเร็วขึ้น

ROS Bag File เป็นรูปแบบไฟล์ที่ใช้ในระบบ ROS (Robot Operating System) เพื่อการบันทึกและการเล่นซ้ำข้อมูลจากเซนเซอร์และการสื่อสารภายในระบบ ROS ไฟล์นี้เป็นเครื่องมือที่มีประโยชน์อย่างยิ่งในการทดสอบและการวิเคราะห์ระบบหุ่นยนต์ เนื่องจากสามารถบันทึกข้อมูลจำนวนมากเช่น ข้อมูลจากเซนเซอร์ สถานะของหุ่นยนต์ และข้อความแลกเปลี่ยนระหว่างโหนดใน ROS การใช้ ROS Bag File ช่วยให้นักพัฒนาสามารถวิเคราะห์และทำซ้ำการทดสอบในสภาพแวดล้อมที่ควบคุมได้ รวมทั้งเป็นเครื่องมือในการแชร์ข้อมูลกับผู้อื่นในชุมชน ROS

PCL หรือ Point Cloud Library (Rusu et al., 2011) เป็นไลบรารีโอเพนซอร์สขนาดใหญ่สำหรับการประมวลผลพอยต์คลาวด์ 2D/3D และการมองเห็นด้วยคอมพิวเตอร์ ซึ่งเป็นส่วนสำคัญในการพัฒนาแอปพลิเคชั่นหุ่นยนต์และการมองเห็นด้วยคอมพิวเตอร์ ในไลบรารี PCL ใช้เพื่อการวิเคราะห์และการประมวลผลข้อมูลพอยต์คลาวด์ที่ได้รับจากเซนเซอร์ต่างๆ เช่น เซนเซอร์ไลดาร์หรือกล้อง RGB-D คุณสมบัติหลักของ PCL ใน ROS ได้แก่ การตรวจจับวัตถุ การจำแนกประเภท การติดตามวัตถุ การสร้างแบบจำลอง 3D และการประมวลผลข้อมูลพอยต์คลาวด์ในเวลาจริง

รูปที่ 2.9 เครื่องมือสำหรับปรับแต่งชุดข้อมูลพอยคลาวด์ (Rusu, R. B., & Cousins, S. 2011).

ใน PCL นั้นมีไลบรารีที่ช่วยสามารถปรับแต่งชุดข้อมูลพอยคลาวด์ได้ดังนี้

Voxel Grid filter วิธีนี้ทำงานโดยการแบ่งพื้นที่ข้อมูลออกเป็นกลุ่มเล็กๆ (voxels) และในแต่ละกลุ่มจะเลือกจุดข้อมูลเพียงจุดเดียวเพื่อแทนที่จุดทั้งหมดในกลุ่มนั้น ทำให้จำนวนของพอยต์คลาวด์จะลดลงอย่างมาก เนื่องจากจุดข้อมูลที่ซ้ำซ้อนหรืออยู่ใกล้เคียงกันมากจะถูกลดทอนลงให้เหลือเพียงจุดเดียวต่อหนึ่งกลุ่ม

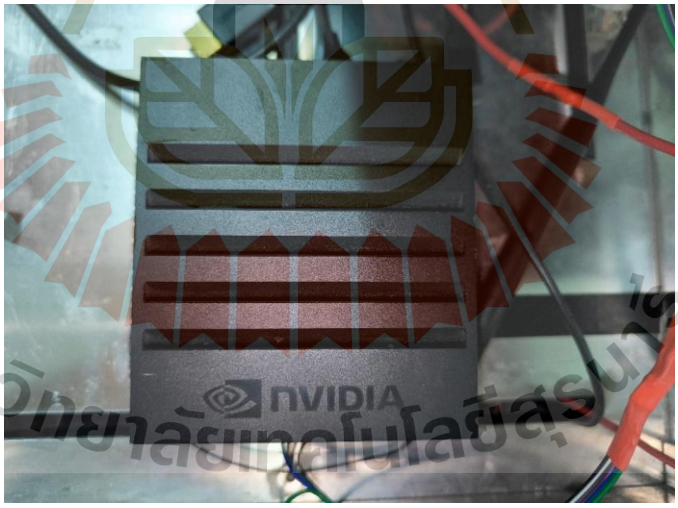
Passthrough filter กรองพอยต์คลาวด์โดยการกำหนดขอบเขตในแกน X, Y, หรือ Z และเฉพาะจุดข้อมูลที่ตกอยู่ในขอบเขตที่กำหนดเท่านั้นที่จะถูกเก็บไว้ ทำให้ช่วยลดจุดข้อมูลที่อยู่นอกขอบเขตที่สนใจ ทำให้ชุดข้อมูลมีความเข้มข้นและเฉพาะเจาะจงมากขึ้น

Open3D (Zhou et al., 2018) เป็นไลบรารีโอเพ่นซอร์สที่ถูกพัฒนาโดยทีมวิจัยจาก Intelligent Systems Lab (ISL) เพื่อรองรับการประมวลผลข้อมูลสามมิติ (3D Data Processing) ที่หลากหลาย เช่น พอยต์คลาวด์ (Point Cloud), เมช (Mesh), และกริดแบบเวกเซล (Voxel Grid) จุดเด่นของ Open3D คือความสามารถในการใช้งานได้ทั้งในภาษา Python และ C++ พร้อมด้วยชุดฟังก์ชันที่ครอบคลุมการใช้งานในหลายขั้นตอนของงานด้านคอมพิวเตอร์กราฟิกส์และวิสัยทัศน์ของเครื่อง (Computer Vision) ตัวอย่างฟังก์ชันที่สำคัญ ได้แก่ การสร้างกล่องครอบวัตถุ (Bounding Box) ทั้งแบบที่ขนานกับแกน (Axis-Aligned Bounding Box: AABB) และแบบหมุนได้ (Oriented Bounding Box: OBB), การโหลดและแสดงผลข้อมูลจากไฟล์ .pcd, .ply, .xyz และรูปแบบอื่น ๆ ได้อย่างมีประสิทธิภาพ ด้วยความสามารถเหล่านี้ Open3D จึงได้รับความนิยมอย่างแพร่หลายในการใช้งานวิจัย เช่น การประมวลผลข้อมูลจาก LiDAR, การเตรียมข้อมูลนำเข้าสำหรับโมเดลการตรวจจับวัตถุสามมิติ (3D Object Detection) และการแสดงผลผลลัพธ์ของการตรวจจับในรูปแบบเชิงภาพอย่างมีประสิทธิภาพและยืดหยุ่น

รูปที่ 2.10 เครื่องมือสำหรับแสดงชุดข้อมูลพอยคลาวด์ (Zhou, Q. Y., Park, J., V. 2018).

2.1.8 Jetson Xavier AGX และ Jetson Stats

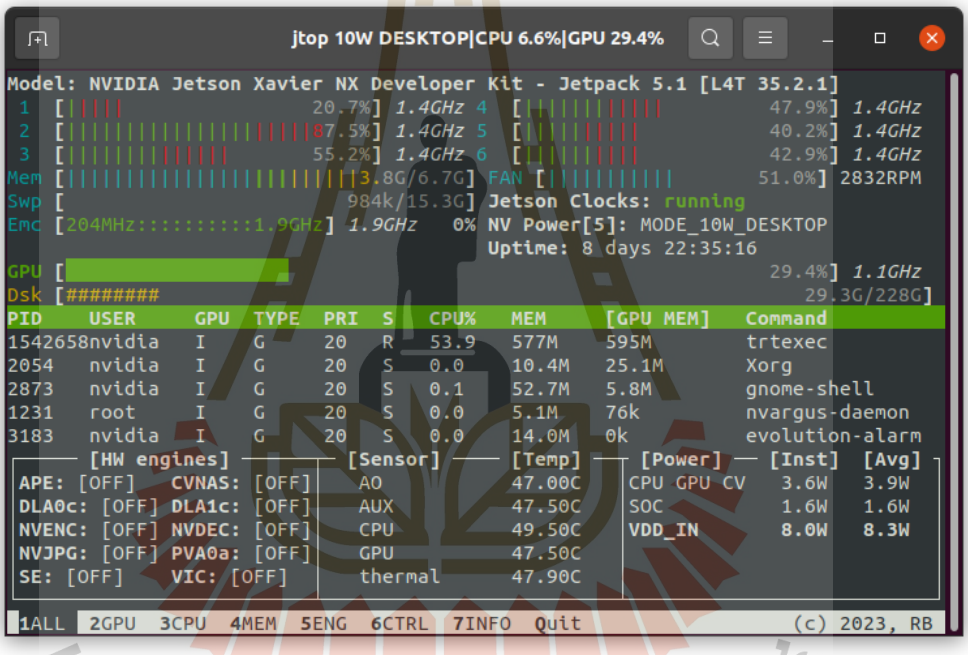
Jetson Xavier AGX เป็นตัวประมวลผลสมรรถนะสูงพลังงานต่ำที่พัฒนาโดยบริษัท NVIDIA ซึ่งถูกออกแบบมาเพื่อรองรับการประมวลผลเชิงลึกด้านปัญญาประดิษฐ์ (Artificial Intelligence: AI) และระบบหุ่นยนต์อัตโนมัติในลักษณะ Edge Computing โดยเฉพาะ อุปกรณ์นี้เป็นหนึ่งในกลุ่มผลิตภัณฑ์ NVIDIA Jetson ซึ่งมีจุดมุ่งหมายเพื่อรองรับการประยุกต์ใช้งาน AI ภาคสนามด้วยขนาดที่กะทัดรัดและประสิทธิภาพที่สูง Jetson Xavier AGX ประกอบด้วยหน่วยประมวลผลกลางแบบ 8 คอร์ ARM v8.2 64-bit, หน่วยประมวลผลกราฟิก Volta GPU ที่มาพร้อม Tensor Cores สำหรับการเร่งการประมวลผลแบบขนาน, หน่วยประมวลผลเฉพาะทางสำหรับ AI อย่าง Deep Learning Accelerator (DLA), และหน่วยความจำ LPDDR4x ขนาด 32 GB อุปกรณ์ยังรองรับอินเทอร์เฟซการเชื่อมต่อที่หลากหลาย เช่น Ethernet, USB 3.1, PCIe และ HDMI นอกจากนี้ยังมาพร้อมชุดพัฒนา JetPack SDK ที่รวมไลบรารีและเครื่องมือสำคัญ เช่น CUDA, cuDNN, TensorRT, และ DeepStream SDK เพื่อสนับสนุนการพัฒนาแอปพลิเคชัน AI และการประมวลผลภาพขั้นสูง ด้วยคุณสมบัติเหล่านี้ Jetson Xavier AGX จึงเหมาะสำหรับการใช้งานในระบบหุ่นยนต์อัตโนมัติ, ยานยนต์ไร้คนขับ, ระบบการมองเห็นของเครื่องจักร (Machine Vision) และแอปพลิเคชัน AI แบบเรียลไทม์ภายใต้ข้อจำกัดด้านพลังงานและทรัพยากร (NVIDIA, 2020)



รูปที่ 2.11 Jetson Xavier AGX

Jetson Stats (Bonghi et al., 2020) เป็นเครื่องมือที่ออกแบบมาเฉพาะสำหรับอุปกรณ์ Jetson ของ NVIDIA โดยมีจุดประสงค์เพื่อใช้ในการตรวจสอบและติดตามการทำงานของทรัพยากรระบบแบบเรียลไทม์ เครื่องมือนี้สามารถแสดงข้อมูลการใช้งานของหน่วยประมวลผลกลาง

(CPU), หน่วยประมวลผลกราฟิก (GPU), หน่วยความจำ (RAM), รวมถึงอัตราการใช้พลังงาน (Power Usage) และอุณหภูมิของฮาร์ดแวร์อย่างต่อเนื่อง Jetson Stats จึงมีบทบาทสำคัญในการวิเคราะห์ประสิทธิภาพของโมเดลที่รันบนอุปกรณ์ฝังตัว (Embedded Systems) โดยเฉพาะอย่างยิ่งในบริบทของงานประมวลผลที่ต้องการประสิทธิภาพสูงภายใต้ข้อจำกัดด้านทรัพยากร เช่น งานด้านการตรวจจับวัตถุแบบเรียลไทม์บน Jetson Xavier AGX หรือ Jetson Nano การใช้เครื่องมือนี้สามารถช่วยให้ผู้พัฒนาระบบสามารถปรับแต่งการทำงานของโมเดลให้เหมาะสมกับข้อจำกัดด้านฮาร์ดแวร์และประเมินผลกระทบจากการปรับเปลี่ยนโมเดลต่อการใช้งานทรัพยากรได้อย่างแม่นยำและเป็นระบบ



รูปที่ 2.12 หน้าจอแสดงค่าการใช้ทรัพยากรของระบบ (Bonghi, R. 2020).

2.1.9 คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (Onboard Computer of Autonomous Golf Cart)

คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (วงศธร, 2565) เป็นหน่วยประมวลผลหลักสำหรับการควบคุมระบบการขับเคลื่อน การรับรู้สภาพแวดล้อม และการตัดสินใจแบบเรียลไทม์ โดยทำหน้าที่เป็น ศูนย์กลางการประมวลผล (Central Processing Unit) ของระบบขับเคลื่อนอัตโนมัติทั้งหมด อุปกรณ์ที่ใช้ในงานวิจัยนี้ประกอบด้วยหน่วยประมวลผลกลาง (CPU) แบบ 11th Gen Intel® Core™ i7-11700K @ 3.60 GHz ซึ่งมีจำนวนคอร์ทั้งหมด 16 คอร์ (8 คอร์จริงและ 8 เธรด

เสมือน) สถาปัตยกรรม “Rocket Lake” ของ Intel ให้ประสิทธิภาพสูงในด้านการคำนวณแบบหลายเธรดและการประมวลผลเชิงลึก พร้อมเทคโนโลยี Intel Turbo Boost ที่ช่วยเพิ่มความเร็วสัญญาณนาฬิกาอัตโนมัติเมื่อต้องการประสิทธิภาพสูงสุด

สำหรับการประมวลผลเชิงกราฟิกและงานด้านปัญญาประดิษฐ์ คอมพิวเตอร์ดังกล่าวใช้หน่วยประมวลผลกราฟิก NVIDIA GeForce RTX 3080 Ti ซึ่งมีสถาปัตยกรรม Ampere ที่มาพร้อม Tensor Cores และ RT Cores สำหรับการเร่งการประมวลผลแบบขนานและการเรียนรู้เชิงลึก (Deep Learning) โดยเฉพาะ GPU รุ่นนี้รองรับเทคโนโลยี CUDA, cuDNN, และ TensorRT ทำให้สามารถประมวลผลแบบขนานขนาดใหญ่และรันโมเดลปัญญาประดิษฐ์ที่ซับซ้อนได้อย่างมีประสิทธิภาพ

ด้วยคุณสมบัติดังกล่าว คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติสามารถรองรับการทำงานของโมเดลตรวจจับวัตถุสามมิติ (3D Object Detection) และการประมวลผลภาพจากเซนเซอร์ LiDAR และกล้องได้แบบเรียลไทม์ โดยยังคงมีเสถียรภาพในการทำงานอย่างต่อเนื่องในสภาพแวดล้อมจริง ทั้งนี้ ระบบยังสามารถเชื่อมต่อกับอุปกรณ์ภายนอกผ่านพอร์ตมาตรฐาน เช่น USB 3.1, Ethernet, HDMI, และ PCIe เพื่อเชื่อมต่อกับกล้อง, เซนเซอร์ และระบบควบคุมการขับเคลื่อนอื่น ๆ

คอมพิวเตอร์นี้จึงถือเป็นแพลตฟอร์มประมวลผลหลักของระบบรถกอล์ฟอัตโนมัติที่ใช้ในการทดลองในงานวิจัยนี้ โดยใช้เปรียบเทียบกับแพลตฟอร์ม Jetson Xavier AGX เพื่อประเมินประสิทธิภาพของโมเดลปัญญาประดิษฐ์ที่ได้รับการปรับปรุงภายใต้สภาพแวดล้อมการประมวลผลที่แตกต่างกัน

รูปที่ 2.12 คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ

2.1.10 เครื่องมือสำหรับการทำจัดการข้อมูลและการติดป้ายกำกับข้อมูลจุดสามมิติ (labelCloud)

ในการเตรียมข้อมูลสำหรับการฝึกสอนโมเดลตรวจจับวัตถุสามมิติจากข้อมูลเซนเซอร์ LiDAR ได้เลือกใช้ซอฟต์แวร์ labelCloud (Sager et al., 2022) ซึ่งเป็นเครื่องมือแบบโอเพ่นซอร์สที่ได้รับการพัฒนาขึ้นมาเพื่อรองรับการสร้างป้ายกำกับ (Labeling) แบบกล่องสามมิติ (3D Bounding Box) บนข้อมูลพอยต์คลาวด์โดยเฉพาะ ซอฟต์แวร์นี้รองรับข้อมูลจากหลากหลายรูปแบบ เช่น .pcd, .ply, .xyz และ .bin (ตามรูปแบบของ KITTI dataset) อีกทั้งยังสามารถกำหนดค่ากล่อง Bounding Box ได้อย่างยืดหยุ่นในเชิงสามมิติ รวมถึงการหมุนในแต่ละแกน (yaw, pitch, roll) นอกจากนี้ labelCloud ยังเปิดโอกาสให้สามารถกำหนดคลาสของวัตถุ และสีของกล่องกำกับที่แตกต่างกันได้ตามต้องการ ซึ่งช่วยเพิ่มความสะดวกในการจัดการข้อมูลที่มีหลายประเภทและหลายฉากการณ์ ความสามารถเหล่านี้ทำให้ labelCloud เป็นเครื่องมือที่มีประสิทธิภาพในการสร้างชุดข้อมูลแบบ Annotated Point Cloud ซึ่งเหมาะสมอย่างยิ่งสำหรับการใช้งานร่วมกับโมเดล Deep Learning ในการตรวจจับวัตถุสามมิติจากข้อมูล LiDAR ในบริบทของการวิจัยด้านหุ่นยนต์อัตโนมัติหรือยานพาหนะไร้คนขับ

รูปที่ 2.13 เครื่องมือสำหรับจัดการข้อมูลจากเซนเซอร์ (LiDAR Sager, C., N. 2022).

2.1.11 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องจะกล่าวถึงงานวิจัยอื่น ๆ ที่ใช้การปรับปรุงโมเดล Pointpillar และการใช้ Attention Network ปรับปรุงโมเดล เป็นต้น

Zhang et al., 2021 เสนอ Pyramid Squeeze Attention (PSA) Module ซึ่งเป็นโมดูลเบาสำหรับใช้แทน convolution แบบ 3×3 ภายในโครงข่ายลึก โดย PSA ออกแบบมาเพื่อสกัดฟีเจอร์หลายระดับอย่างมีประสิทธิภาพ โดยไม่เพิ่มพารามิเตอร์หรือค่าใช้จ่ายในการคำนวณ โมดูลนี้สามารถรวมบริบทเชิงพื้นที่และเชิงช่องสัญญาณเข้าด้วยกันในลักษณะของพีระมิด ทำให้เหมาะสำหรับโมเดลแบบ lightweight ที่ต้องการความแม่นยำสูงแต่ทำงานบนฮาร์ดแวร์จำกัด ซึ่งสอดคล้องกับแนวทางการใช้ Attention ที่พัฒนาใน LAPN

Zhang, L. et al., 2023 นำเสนอโมเดล Transformer-Based Global PointPillars (TGPP) ที่เสริมความสามารถของ PointPillars เดิมโดยการแทรก Multi-Head Attention (MHA) จากแนวคิดของ Transformer เข้าไปในขั้นตอนของการ encoding ฟีเจอร์ภายในแต่ละ pillar เพื่อให้โมเดลสามารถเข้าใจ global context ได้ควบคู่กับ local structure ส่งผลให้ความแม่นยำในการตรวจจับดีขึ้นประมาณ 2.6% บนชุดข้อมูล KITTI ซึ่งแสดงให้เห็นว่าแนวทางการเพิ่ม Attention Module ในโครงสร้าง PointPillars มีศักยภาพสูงในการเพิ่มประสิทธิภาพการตรวจจับ โดยเฉพาะในงานที่ต้องรองรับวัตถุหลายประเภทและหลายขนาด

Zhai et al., 2024 เสนอโครงข่าย ASCA-PointPillars ที่ใช้ adaptive-scale pillars (ASP) ร่วมกับ correlative point attention (CPA) เพื่อปรับปรุง pillar encoding แบบ multi-scale ทำให้โมเดลสามารถรับมือกับวัตถุห่าง/เล็กได้อย่างมีประสิทธิภาพ

Konrad Lis & Tomasz Kryjak., 2022 ได้ทำการศึกษาเปรียบเทียบ Backbone Architectures สำหรับ PointPillars โดยทดลองแทนที่ CNN ดั้งเดิมด้วยโครงข่ายเบา เช่น MobileNetV1 และ CSPDarknet ผลลัพธ์แสดงให้เห็นว่าสามารถเพิ่มความเร็วในการทำ inference ได้มากถึง 1.5–4 เท่า ขณะที่ค่าความแม่นยำลดลงเพียงเล็กน้อย (ไม่เกิน 1.2%) แนวทางนี้แสดงให้เห็นว่า การเลือก backbone ที่เหมาะสมสามารถลดภาระการประมวลผลได้อย่างมาก โดยยังคงรักษาคุณภาพของการตรวจจับ ซึ่งแนวคิดนี้มีความสอดคล้องกับงานวิจัยที่ลดจำนวน convolution layers และแทนที่ด้วย attention module ที่มีน้ำหนักเบา เพื่อให้เหมาะสมกับการทำงานบน Jetson AGX Xavier

Shi et al., 2022 ได้นำเสนอโครงข่าย PillarNet ซึ่งเป็น backbone ใหม่ที่ออกแบบมาเฉพาะสำหรับข้อมูลแบบ pillar โดยเน้นการปรับสมดุลระหว่าง ความเร็วในการประมวลผล และความแม่นยำในการตรวจจับ โดยไม่ต้องใช้โครงสร้างแบบ voxel-based ที่ซับซ้อน PillarNet ใช้โครงสร้างเบาแต่สามารถสกัดฟีเจอร์ได้ลึก ทำให้ได้ผลลัพธ์ใกล้เคียงกับโมเดลขนาดใหญ่ที่ใช้ voxel แต่ออกแบบให้เหมาะสำหรับการใช้งานที่ต้องการ real-time และทรัพยากรจำกัด แนวคิดนี้สนับสนุนแนวทางที่ออกแบบ LAPN โดยเน้น lightweight feature extraction และการใช้ attention modules แทน convolution blocks ที่หนักและใช้ทรัพยากรมาก

Lin et al., 2017 ได้นำเสนอ Feature Pyramid Network (FPN) ซึ่งเป็นโครงข่ายสำหรับการรวมคุณลักษณะจากหลายระดับความละเอียด โดยใช้การเชื่อมต่อจากล่างขึ้นบนและจากบนลงล่างร่วมกัน ทำให้สามารถตรวจจับวัตถุได้ดีทั้งในขนาดเล็กและใหญ่ โครงข่าย FPN กลายเป็นพื้นฐานสำคัญในงานตรวจจับวัตถุหลายสาขา และมีอิทธิพลต่อการพัฒนาโครงข่ายในลักษณะพีระมิด เช่น Lightweight Attention Pyramid Network (LAPN) ที่ใช้ในงานวิจัยนี้ ซึ่งแทรก Attention Module แบบเบาเข้าไปในทุกระดับชั้นของโครงสร้าง CNN เพื่อเน้นฟีเจอร์สำคัญแบบ Multi-Scale โดยยังคงความเบาและเร็วสำหรับระบบหุ่นยนต์ที่รันบน Xavier

Lang et al., 2019 ได้นำเสนอโมเดล PointPillars ที่เป็นหนึ่งในวิธีการแปลงข้อมูล Point Cloud ให้อยู่ในรูปแบบ 2D pseudo-image เพื่อใช้กับ 2D CNN โดยมีการกำหนดขอบเขตของ Point Cloud ล่วงหน้า (Point Cloud Range) เพื่อควบคุมพื้นที่การตรวจจับให้เน้นเฉพาะบริเวณสำคัญ เช่น ด้านหน้ารถยนต์หรือระยะที่สัมพันธ์กับความปลอดภัย ซึ่งแนวคิดนี้ได้กลายเป็นแนวปฏิบัติพื้นฐานในการปรับปรุงโมเดลตรวจจับวัตถุแบบ 3 มิติที่ใช้บนระบบที่มีข้อจำกัดด้านการประมวลผล เช่น Jetson AGX Xavier

Hu et al., 2021 ยังได้เสนอ EPSANet ที่ใช้ Pyramid Squeeze Attention Module แทน convolution ซ้ำในแต่ละบล็อก ซึ่งแนวทางนี้มีแนวคิดใกล้เคียงกับ LAPN โดยใช้ attention แบบเบาในการสกัดฟีเจอร์หลายระดับ ทำให้โมเดลมีประสิทธิภาพดีขึ้นโดยไม่เพิ่มพารามิเตอร์มากเกินไป และสามารถนำไปประยุกต์ใช้กับโครงข่ายเบาสำหรับ edge device ได้อย่างมีประสิทธิภาพ
Zhou et al., 2018 ได้พัฒนาโมเดล VoxelNet ซึ่งถือเป็นต้นแบบของการใช้การแบ่งพื้นที่ Point Cloud ด้วย Voxel Grid และใช้ขนาดของ Voxel เป็นตัวควบคุม resolution ของข้อมูลฟีเจอร์ โดยพบว่าการปรับขนาด Voxel ให้ละเอียดขึ้นในแนวราบ (X, Y) ช่วยเพิ่มความสามารถในการจำแนกวัตถุขนาดเล็ก และทำให้โมเดลตอบสนองกับวัตถุใกล้ได้ดีขึ้น แนวคิดนี้มีอิทธิพลอย่างมากต่อการปรับโครงสร้างการเรียนรู้ของโมเดล PointPillars เพื่อให้สอดคล้องกับการทำงานแบบเรียลไทม์บน Edge Device

Zhang et al., 2020 นำเสนอ Lightweight Attention Pyramid Network (LAPN) ซึ่งเป็นโครงข่ายแบบพีระมิดที่ผสมผสานกลไก Attention เพื่อเน้นคุณลักษณะที่สำคัญในแต่ละระดับของฟีเจอร์ และรวมข้อมูลจากหลายระดับความละเอียด ช่วยให้โมเดลสามารถตรวจจับวัตถุขนาดเล็กและรายละเอียดเฉพาะจุดได้ดีขึ้น โดยไม่เพิ่มภาระการคำนวณมากนัก โครงข่ายนี้จึงเหมาะอย่างยิ่งสำหรับการใช้งานในระบบที่มีข้อจำกัดด้านทรัพยากร เช่น Embedded Device หรือหุ่นยนต์อัตโนมัติ

Peri et al., 2023 ได้ศึกษาอิทธิพลของ ช่วงการตรวจจับ (Point Cloud Range) และ ขนาด Voxel ที่มีต่อประสิทธิภาพของโมเดลตรวจจับวัตถุแบบ 3 มิติ โดยทดสอบการจำกัดช่วงการประมวลผลให้อยู่ในระยะใกล้ เช่น ด้านหน้าและด้านข้างของยานพาหนะ ซึ่งเป็นระยะที่เกี่ยวข้องกับ

การตัดสินใจของระบบนำทาง พบว่าการจำกัดขอบเขตเชิงพื้นที่ร่วมกับการเลือกขนาด voxel ที่เหมาะสม สามารถลดภาระการประมวลผลและเพิ่มความแม่นยำในการตรวจจับได้อย่างมีนัยสำคัญ แนวทางนี้สอดคล้องกับการออกแบบระบบที่เน้นประสิทธิภาพสำหรับ edge computing platform เช่น Jetson AGX Xavier

2.2 การประยุกต์ใช้งานเทคโนโลยี LiDAR และการตรวจจับวัตถุสามมิติ

เทคโนโลยีการตรวจจับวัตถุสามมิติจากข้อมูล LiDAR (Light Detection and Ranging) ได้รับความสนใจอย่างแพร่หลายทั้งในภาคการวิจัยและภาคอุตสาหกรรม เนื่องจากสามารถให้ข้อมูลเชิงลึกเชิงพื้นที่ (depth information) ที่แม่นยำและไม่ขึ้นกับสภาพแสงเหมือนกล้องทั่วไป ข้อมูลจาก LiDAR จึงถูกนำมาใช้ร่วมกับเทคนิคการประมวลผลเชิงลึก (Deep Learning) เพื่อพัฒนาระบบตรวจจับวัตถุแบบสามมิติที่สามารถทำงานได้อย่างแม่นยำในสภาพแวดล้อมจริง โดยเฉพาะในงานที่ต้องการความปลอดภัยสูง เช่น ยานยนต์อัตโนมัติ หุ่นยนต์ขนส่ง และระบบอุตสาหกรรมอัจฉริยะ

หนึ่งในสถาปัตยกรรมที่ได้รับความนิยมสูงคือ PointPillars ซึ่งสามารถแปลงข้อมูลจุดสามมิติ (Point Cloud) ให้เป็นลักษณะของพิลลาร์ (Pillar) ที่มีประสิทธิภาพในการจัดการข้อมูลแบบ Sparse ทำให้สามารถนำข้อมูลเข้าโมเดลเชิงลึกได้รวดเร็วกว่าแบบ PointNet หรือ VoxelNet แบบดั้งเดิม จุดเด่นนี้ทำให้ PointPillars ถูกนำไปใช้อย่างกว้างขวางในระบบ ยานยนต์อัตโนมัติ (Autonomous Vehicle) เช่น งานของ Lang et al. (2019) ซึ่งพัฒนา PointPillars เพื่อใช้ตรวจจับรถยนต์และคนเดินเท้าในชุดข้อมูล KITTI ได้แบบเรียลไทม์ และงานต่อเนื่องในอุตสาหกรรมยานยนต์ เช่น Waymo, Cruise, และ Baidu Apollo ที่นำเทคนิคนี้ไปใช้ในระบบตรวจจับสิ่งกีดขวาง (Obstacle Detection) เพื่อเพิ่มความปลอดภัยของการขับเคลื่อน

นอกจากนี้ PointPillars ยังถูกนำไปใช้ในงาน หุ่นยนต์ขนส่งอัตโนมัติ (Autonomous Delivery Robot) และ หุ่นยนต์อุตสาหกรรม (Industrial Robot) เพื่อให้สามารถตรวจจับสิ่งกีดขวางหรือวัตถุในพื้นที่จำกัดได้อย่างแม่นยำ ตัวอย่างเช่น งานของ Wang et al. (2021) ที่ใช้ PointPillars ในการตรวจจับกล่องพัสดุและคนในคลังสินค้าแบบเรียลไทม์ ซึ่งช่วยลดอุบัติเหตุจากการชนและเพิ่มประสิทธิภาพในการจัดเส้นทางเดินของหุ่นยนต์ ในภาคอุตสาหกรรมการผลิต เทคโนโลยีการตรวจจับวัตถุสามมิติจาก LiDAR ยังถูกใช้ในสายการผลิตอัจฉริยะเพื่อระบุตำแหน่งของชิ้นส่วนหรือตรวจสอบความถูกต้องของชิ้นงาน โดยใช้ร่วมกับแขนกลอัตโนมัติ (Industrial Manipulator) ซึ่งช่วยลดต้นทุนแรงงานและเวลาในการตรวจสอบคุณภาพ

ในส่วนของ Attention Mechanism ซึ่งเป็นเทคนิคที่ช่วยให้โมเดลสามารถโฟกัสเฉพาะบริเวณที่สำคัญของข้อมูลได้ ถูกนำมาใช้เพื่อเพิ่มประสิทธิภาพของโมเดลตรวจจับสามมิติอย่างต่อเนื่อง งานของ Li et al. (2022) นำ Self-Attention มาปรับใช้ในโครงสร้างของ PointPillars เพื่อให้ระบบ

สามารถเน้นจุดข้อมูลที่มีลักษณะโดดเด่น ส่งผลให้เพิ่มความแม่นยำในการจำแนกวัตถุโดยใช้เวลาประมวลผลเท่าเดิม อีกทั้งงานของ Zhang et al. (2023) ยังแสดงให้เห็นว่า การเพิ่ม Attention Layer ในส่วนของ Feature Encoding ช่วยให้โมเดลสามารถทำงานได้ดีขึ้นในสภาพแวดล้อมที่มีจุดรบกวนจำนวนมาก เช่น ในโรงงานหรือพื้นที่สาธารณะ

อย่างไรก็ตาม งานวิจัยส่วนใหญ่ที่กล่าวมามักมุ่งเน้นการพัฒนาโมเดลสำหรับระบบคอมพิวเตอร์ประสิทธิภาพสูง (High-Performance Computing) หรือแพลตฟอร์มยานยนต์ขนาดใหญ่ที่มีทรัพยากรเพียงพอ ซึ่งไม่เหมาะกับอุปกรณ์ฝังตัว (Embedded System) ที่มีข้อจำกัดด้านพลังงานและหน่วยความจำ เช่น หุ่นยนต์ขนาดเล็กหรือโดรนขนส่งขนาดเล็ก ดังนั้นงานวิจัยนี้จึงมุ่งเน้นการปรับปรุงสถาปัตยกรรมของ PointPillars ให้มีความเบา (Lightweight Architecture) และลดขอบเขตการตรวจจับให้แคบลงตามระยะการปฏิบัติงานจริงของหุ่นยนต์ พร้อมทั้งเสริมด้วย Attention Mechanism เพื่อช่วยให้โมเดลโฟกัสเฉพาะพื้นที่ที่สำคัญต่อการตรวจจับ

การปรับปรุงดังกล่าวมีความสำคัญต่อการประยุกต์ใช้งานในอุตสาหกรรมหุ่นยนต์ขนาดเล็กและระบบอัตโนมัติในเมือง (Urban Robotics) เช่น หุ่นยนต์ขนส่งของบนทางเท้าหรือในอาคาร ที่ต้องตรวจจับสิ่งกีดขวางและผู้คนอย่างรวดเร็วภายใต้ข้อจำกัดด้านพลังงานและหน่วยประมวลผล งานวิจัยนี้จึงถือเป็นแนวทางที่สอดคล้องกับแนวโน้มของอุตสาหกรรมยุคใหม่ที่มุ่งสู่การพัฒนา ระบบอัตโนมัติประสิทธิภาพสูงแต่ใช้พลังงานต่ำ (High-Efficiency, Low-Power Robotics) ซึ่งมีศักยภาพต่อการนำไปใช้งานจริงทั้งในด้านความปลอดภัยและการลดต้นทุนการดำเนินงาน

ตารางที่ 2.1 การประยุกต์ใช้งานเทคโนโลยี LiDAR และ PointPillars

อุตสาหกรรม	ตัวอย่างการประยุกต์ใช้	ประโยชน์และผลลัพธ์ที่ได้รับ
ยานยนต์อัตโนมัติ (Autonomous Vehicle)	Lang และคณะ (2019) ใช้ LiDAR ร่วมกับ PointPillars ในการตรวจจับรถยนต์และคนเดินเท้าแบบเรียลไทม์เพื่อการขับขี่อัตโนมัติที่ปลอดภัย	เพิ่มความแม่นยำในการตรวจจับสิ่งกีดขวาง ลดอุบัติเหตุจากระบบขับเคลื่อนอัตโนมัติ
หุ่นยนต์ขนส่ง (Delivery Robot)	Wang และคณะ (2021) ใช้โมเดลตรวจจับวัตถุแบบเบาในการหลบหลีกสิ่งกีดขวางและเคลื่อนที่ในพื้นที่แคบ เช่น ทางเดินหรือคลังสินค้า	ลดต้นทุนฮาร์ดแวร์และพลังงาน ทำงานได้แบบเรียลไทม์และปลอดภัยยิ่งขึ้น

ตารางที่ 2.1 การประยุกต์ใช้งานเทคโนโลยี LiDAR และ PointPillars (ต่อ)

อุตสาหกรรม	ตัวอย่างการประยุกต์ใช้	ประโยชน์และผลลัพธ์ที่ได้รับ
ความปลอดภัยและ Smart City	Zhang และคณะ (2023) ใช้ LiDAR ตรวจจับคนหรือยานพาหนะในพื้นที่สาธารณะ เพื่อตรวจสอบการจราจรหรือความปลอดภัย	เพิ่มความปลอดภัยในพื้นที่ชุมชน และช่วยจัดการการจราจรอย่างมีประสิทธิภาพ
อุตสาหกรรม	**ตัวอย่างการประยุกต์ใช้**	**ประโยชน์และผลลัพธ์ที่ได้รับ**
หุ่นยนต์อัตโนมัติขนาดเล็ก (Embedded Robotics)	งานวิจัยนี้ได้ใช้โมเดลที่ปรับปรุงแล้วบน Jetson Xavier ในการตรวจจับสิ่งกีดขวางและผู้คนรอบตัว	ประมวลผลได้รวดเร็ว ใช้พลังงานต่ำ เหมาะสำหรับหุ่นยนต์ภาคสนามและระบบบริการอัตโนมัติ

ที่มา: ผู้วิจัยสรุปจาก Lng และคณะ (2019), Wang และคณะ (2021), Zhang และคณะ (2023)

บทที่ 3
วิธีการดำเนินงานวิจัย

งานวิจัยฉบับนี้นำเสนอการปรับปรุงโครงสร้างปัญญาประดิษฐ์ของสถาปัตยกรรม PointPillars ที่ใช้ข้อมูล LiDAR ในระบบตรวจจับวัตถุ 3 มิติ เพื่อสร้างโครงสร้างที่ลดภาระการคำนวณของหุ่นยนต์ ทำให้หุ่นยนต์สามารถใช้การตรวจจับนี้ในการระบุประเภทและตำแหน่งวัตถุได้ แล้วนำข้อมูลเหล่านี้ไปใช้ในการหลีกเลี่ยงการชนกับวัตถุนั้นได้ ทำให้เพิ่มความปลอดภัยให้กับหุ่นยนต์ได้ ทั้งนี้ในการปรับปรุงโครงสร้างจะเริ่มการเตรียมการที่ครอบคลุมตั้งแต่การรวบรวมข้อมูลและการศึกษาค้นคว้าที่เกี่ยวข้อง การเตรียมพื้นที่ทดสอบ ไปจนถึงการทดสอบการทำงานของอุปกรณ์ การปรับปรุงโครงสร้างนี้จะเกี่ยวข้องกับการกรองข้อมูลจาก LiDAR การเก็บข้อมูล การแปลงข้อมูล การปรับแต่งโครงสร้าง การเทรนโมเดล การประเมินผลโมเดล การทดสอบภาระการคำนวณบนหุ่นยนต์ การทดสอบตรวจจับวัตถุ และการทดสอบหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้

3.1 การเตรียมการ

ในขั้นตอนการเตรียมการได้ดำเนินการตามลำดับต่อไปนี้

3.1.1 การศึกษาค้นคว้าและรวบรวมงานวิจัยที่เกี่ยวข้อง

การเตรียมการสำหรับงานวิจัยนี้ได้เริ่มต้นด้วยการทำวิจัยเชิงสำรวจเกี่ยวกับเทคโนโลยีการมองเห็นของเครื่องจักร (Machine Vision) และปัญญาประดิษฐ์ (Artificial Intelligence) จาก แหล่งข้อมูลที่หลากหลาย เช่น งานวิจัย และบทความที่เกี่ยวข้อง ด้วยเป้าหมายที่จะปรับปรุงโครงสร้างปัญญาประดิษฐ์ของสถาปัตยกรรม PointPillars ให้หุ่นยนต์ส่งของอัตโนมัติ การปรับแต่งขอบเขตระยะตรวจจับ (Detection Range) และความละเอียดของเวกเซล (Voxel Resolution) ของสถาปัตยกรรม PointPillars เป็นขั้นตอนในการสร้างกริดหรือพื้นที่ในการตรวจจับของปัญญาประดิษฐ์เป็นขั้นตอนสำคัญที่ช่วยกำหนดขนาดกริดที่เหมาะสมซึ่งจะช่วยจัดการลดภาระการคำนวณของปัญญาประดิษฐ์และยังคงการตรวจจับอย่างแม่นยำ การใช้โครงสร้างพีระมิดความสนใจน้ำหนักเบา (Lightweight Attention Pyramid Network) สำหรับการลดคุณลักษณะที่ไม่เกี่ยวข้องลง ช่วยให้เครือข่ายสามารถโฟกัสไปที่ข้อมูลที่มีประโยชน์มากที่สุด ส่งผลให้การสกัดคุณลักษณะมี

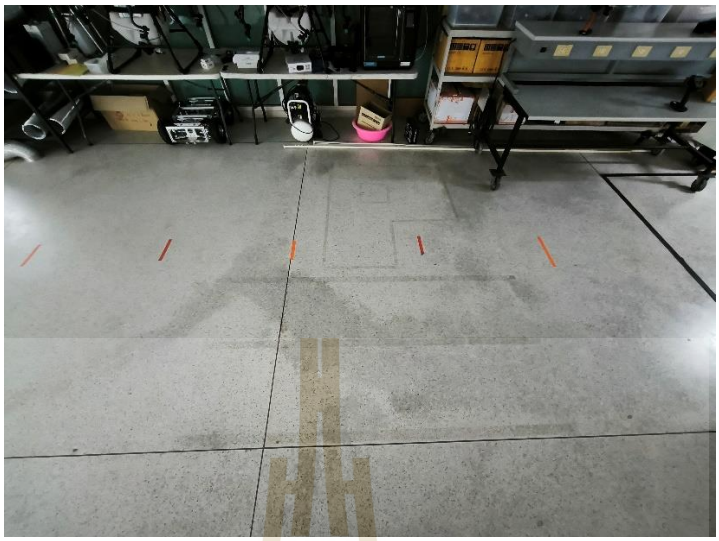
ความแม่นยำมากยิ่งขึ้น ช่วยให้เครือข่ายสามารถจดจำวัตถุได้อย่างมีประสิทธิภาพ แม้ว่าวัตถุจะมีขนาด รูปร่าง หรือมุมมองที่แตกต่างกัน สิ่งเหล่านี้ได้เป็นพื้นฐานในปรับปรุงโครงสร้างปัญญาประดิษฐ์ของสถาปัตยกรรม PointPillars ซึ่งช่วยลดภาระการคำนวณของหุ่นยนต์ การวิจัยยังครอบคลุมถึงหลักการเขียนและวิธีการใช้ภาษาทางคอมพิวเตอร์สำหรับซอฟแวร์ Robot Operating System คือ ภาษา C++ และ ภาษา Python

3.1.2 การเตรียมพื้นที่ที่จะทำการทดสอบ

พื้นที่ทดสอบที่เลือกสำหรับการทดสอบนี้เป็นพื้นที่ภายในที่ปิด ซึ่งจำลองเป็นบริเวณทางตรงเพื่อจำลองสภาพแวดล้อมจริงที่รถอัตโนมัติอาจจะเจอ พื้นที่นี้มีลักษณะเป็นพื้นผิวแบนที่กว้างขวาง ไม่ขรุขระ ไม่มีเนิน ช่วยให้สามารถทำการทดสอบการเคลื่อนที่ของหุ่นยนต์ได้โดยไม่มี ซึ่งการที่ใช้เซนเซอร์ LiDAR จะไม่ได้ผลกระทบจากปัจจัยรบกวนจากภายนอก เช่น แสงแดดที่แปรปรวนหรือสภาพอากาศ เพราะใช้เลเซอร์ในการทำงานต่างจากกล้องที่มีผลกระทบเมื่อแสงเปลี่ยนแปลง ดังนั้น แสงจึงไม่มีกระทบผลต่อประสิทธิภาพของเซ็นเซอร์ LiDAR และระบบการทำงานของ รถอัตโนมัติ

รูปที่ 3.1 พื้นที่ทดสอบ

ในภาพที่ 3.1 มีหุ่นยนต์ที่วางอยู่บนพื้นที่ทดสอบซึ่งเป็นทางตรงยาวสำหรับให้หุ่นยนต์ขับเคลื่อนไปข้างหน้า ซึ่งไม่มีสิ่งกีดขวางข้างหน้า เพื่อให้สามารถเก็บข้อมูลวัตถุได้ถูกต้องสำหรับทดสอบหุ่นยนต์อัตโนมัติ



รูปที่ 3.2 พื้นที่ทดสอบที่มีเครื่องหมายวัดระยะ

ภาพที่ 3.2 แสดงการทำเครื่องหมายบนพื้นเพื่อกำหนดขอบเขตในการเก็บข้อมูล LiDAR และใช้ทดสอบระยะการตรวจจับวัตถุจากอัลกอริทึม Pointpillar โดยแต่ละจุดห่างกัน 100 เซนติเมตรหรือ 1 เมตรการตั้งเครื่องหมายนี้เป็นส่วนสำคัญในการทดสอบอัลกอริทึมในสภาพแวดล้อมที่จำกัดและเข้าใจได้ง่ายและยังช่วยให้สามารถทำการทดสอบซ้ำได้โดยคงไว้ซึ่งความสม่ำเสมอของตำแหน่งและระยะทางในแต่ละการทดลอง เพื่อเก็บข้อมูลที่สามารถนำไปวิเคราะห์ได้อย่างถูกต้อง

รูปที่ 3.3 พื้นที่ทดสอบที่มีวัตถุวางอยู่ที่ใช้ในการตรวจจับ

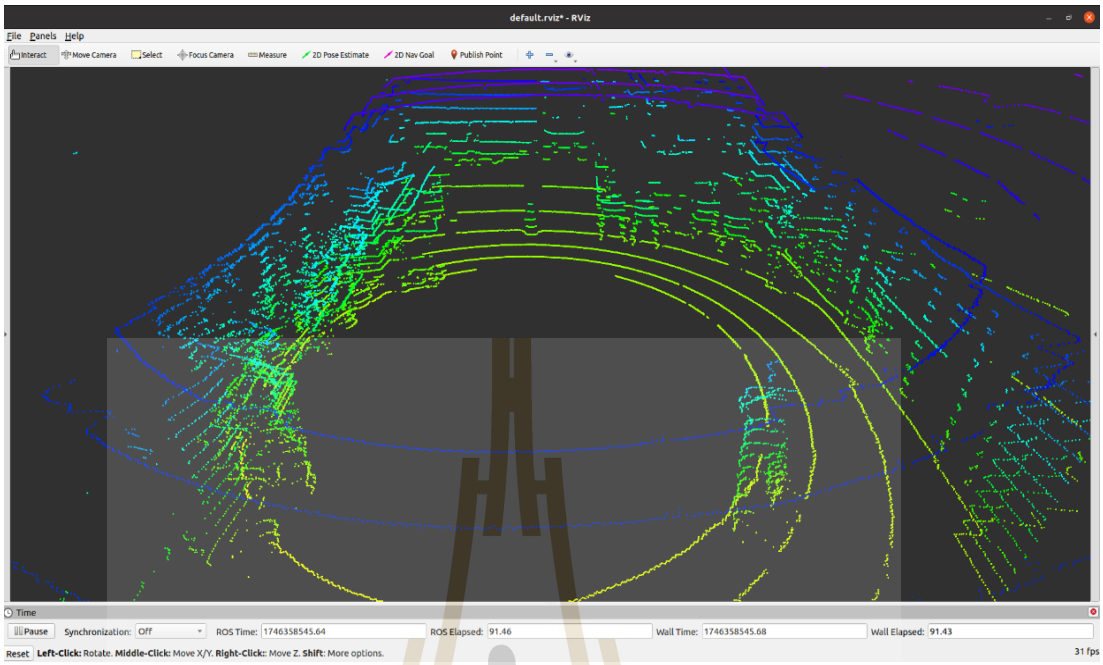
ในภาพที่ 3.3 มีพื้นที่ที่ใช้วางวัตถุสำหรับเป็นชุดข้อมูลที่จะใช้ในการทดสอบ ในภาพนี้คือกรวย ที่จะใช้ในการตรวจจับ 3 มิติของอัลกอริทึม Pointpillar อยู่บนพื้นที่ทดสอบ ซึ่งใช้เป็นตัวแทนในชุดข้อมูลสำหรับจำแนกของประเภทวัตถุ การจัดตั้งพื้นที่ดังกล่าวช่วยให้การ ทดลองสามารถควบคุมได้อย่างแม่นยำและเก็บข้อมูลที่จำเป็นได้อย่างสมบูรณ์ สำหรับการวิเคราะห์และการปรับปรุงชุดข้อมูล

3.1.3 ทดสอบการทำงานของ LiDAR

การทดสอบการทำงานของ LiDAR ด้วยไลบารี่ velodyne เป็นกระบวนการที่สำคัญในการพัฒนาระบบวิชัน คอมพิวเตอร์เพื่อการตรวจจับ 3 มิติ ภายในกลุ่มพอยคลาวด์ที่ได้จาก LiDAR การทดสอบนี้มุ่งเน้นไปที่การประเมินความสามารถของ LiDAR ในการจับกลุ่มพอยคลาวด์ที่ชัดเจน

รูปที่ 3.4 เซนเซอร์ LiDAR

จากภาพที่ 3.4 แสดงให้เห็นเซนเซอร์ LiDAR ที่ติดตั้งอยู่หุ่นยนต์อัตโนมัติเพื่อใช้ในการตรวจจับ 3 มิติ



รูปที่ 3.5 ภาพการทดสอบเปิด LiDAR โดยใช้ C++ และ ไลบรารี่ velodyne บนซอฟต์แวร์ rviz

ในภาพที่ 3.5 แสดงการทำงานของเซนเซอร์ LiDAR ที่ใช้เลเซอร์สแกนรอบทิศทางของตัวหุ่นยนต์ไปตกกระทบวัตถุต่างๆแล้วจะสะท้อนกลับมาที่เซนเซอร์ทำให้หุ่นยนต์สร้างกลุ่มข้อมูลพอยคลาวด์จากวัตถุที่อยู่รอบตัวหุ่นยนต์เพื่อนำข้อมูลไปใช้สำหรับการประมวลผลต่อไป

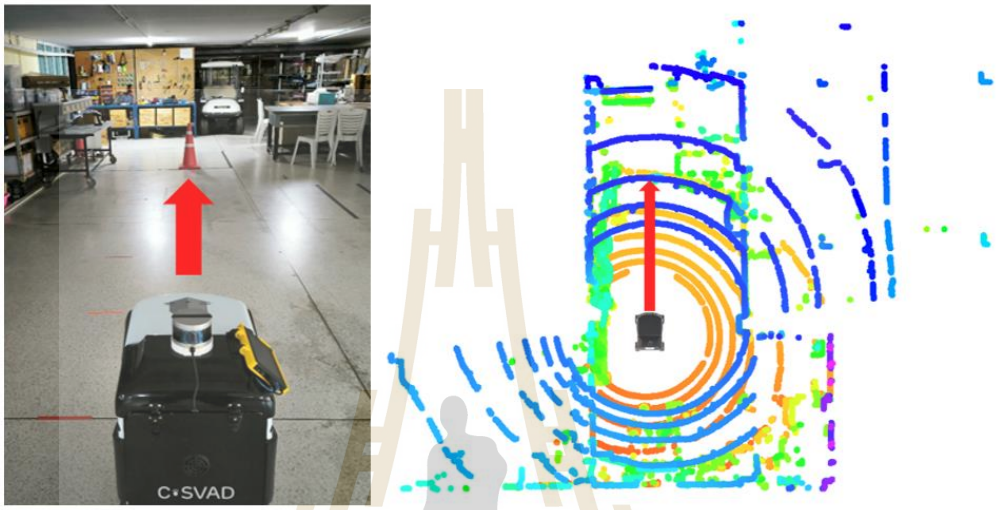
3.2 การปรับปรุงสถาปัตยกรรม PointPillars

การปรับปรุงสถาปัตยกรรม PointPillars ให้หุ่นยนต์อัตโนมัติที่มีตัวประมวลผลที่มีทรัพยากรการคำนวณที่จำกัดโดยใช้ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System) ซึ่งเป็นมาตรฐานในอุตสาหกรรมสำหรับการพัฒนาแอปพลิเคชันหุ่นยนต์ เทคโนโลยีนี้ช่วยให้การปรับปรุงสถาปัตยกรรม PointPillars และ ใช้งานเซ็นเซอร์ เป็นไปอย่างราบรื่น และสามารถสื่อสารกับระบบซับซ้อนได้อย่างมีประสิทธิภาพโดยแยกหัวข้อในการปรับปรุงสถาปัตยกรรม และทดสอบอัลกอริทึมออกเป็นหัวข้อย่อยต่อไปนี้

3.2.1 การเก็บข้อมูลพอยคลาวด์

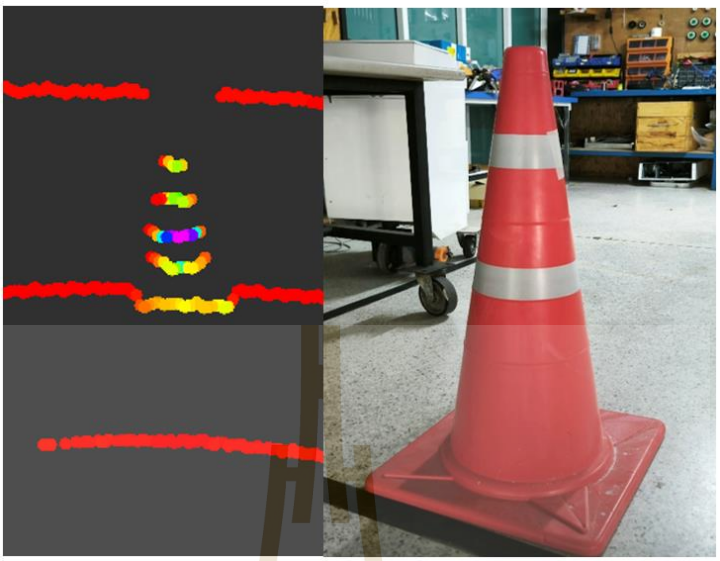
ก่อนจะเก็บข้อมูล จำเป็นต้องมีวางแผนการเก็บข้อมูล เนื่องจากปัญญาประดิษฐ์จะเรียนรู้การตรวจจับจากข้อมูลในสถานการณ์จริง ซึ่งในการวิจัยนี้กำหนดให้หุ่นยนต์วิ่งเข้าหาวัตถุที่อยู่นิ่ง หมายความว่าถ้าต้องการให้ตรวจจับวัตถุขณะหุ่นยนต์วิ่ง จำเป็นต้องเก็บข้อมูลตั้งแต่ที่ หุ่นยนต์เคลื่อนที่ตั้งแต่ 7 เมตรถึง 2 เมตรโดยเทียบระยะจากเครื่องหมายวัดระยะที่ติดไว้บนพื้นเพื่อให้โมเดล

สามารถเรียนรู้ข้อมูลจากสถานการณ์จริงแล้วทำนายได้ถูกต้อง นอกจากจะเก็บด้านหน้าของหุ่นยนต์แล้วจะให้วิ่งตรงแล้วเฉียงซ้ายและเฉียงขวา เพื่อครอบคลุมด้านหน้าของหุ่นยนต์ เผื่อหุ่นยนต์หันหัวไม่ตรงก็สามารถเห็นวัตถุได้



รูปที่ 3.6 วางแผนการเก็บข้อมูล

การเก็บข้อมูลพอยคลาวด์ เป็นกระบวนการที่จะเก็บจำนวนพอยคลาวด์โดยจะใช้จะเครื่องมือ rosbag ของ 1 ในเครื่องมือของ Robot Operating System (ROS) ที่ช่วยบันทึกและการเล่นซ้ำข้อมูลจากเซนเซอร์และการสื่อสารภายในระบบ ROS ด้วยวิธีการนี้จะช่วยเก็บชุดข้อมูลพอยคลาวด์ที่ผ่านการกรองข้อมูลแล้วมาใช้ในการสร้างโมเดลการเรียนรู้ โดยข้อมูลที่ผ่านการกรองข้อมูลจะเป็นชุดข้อมูลของแต่ละวัตถุที่จะใช้ในการจำแนกประเภทได้แก่กรวย คน คนขี่จักรยาน และรถ

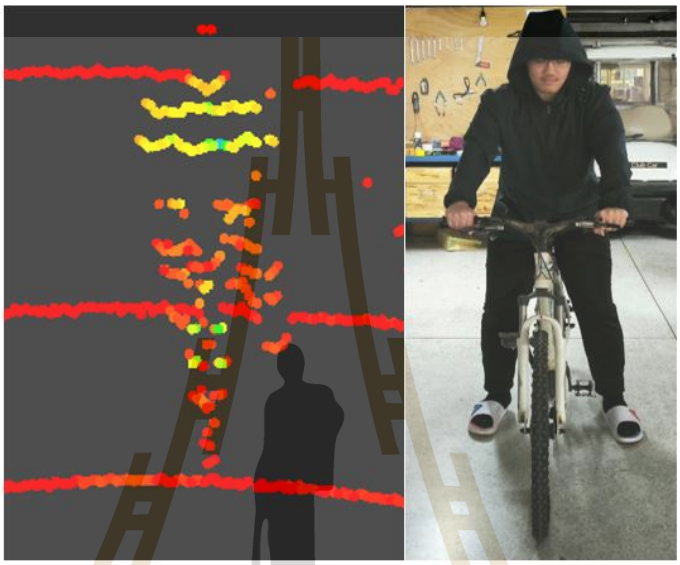


รูปที่ 3.7 ภาพกลุ่มพอยคลาวด์ของกรวยจากเซนเซอร์ LiDAR และภาพกรวยจากกล้อง

ในภาพที่ 3.7 จะแสดงภาพกลุ่มพอยคลาวด์ของกรวยจากเซนเซอร์จะเห็นว่ามีกลุ่มของจุดพอยคลาวด์ที่ถูกสแกนด้วยเลเซอร์กระจายเป็นกลุ่มของจุดเป็นเส้นแนวนอน แบ่งเป็นชั้นๆซึ่งบอกถึงรูปร่างลักษณะของกรวยที่ถูกสแกนมา และภาพกรวยจากกล้องที่ถ่ายมาเพื่อแสดงความแตกต่างของข้อมูลทั้งสองประเภท

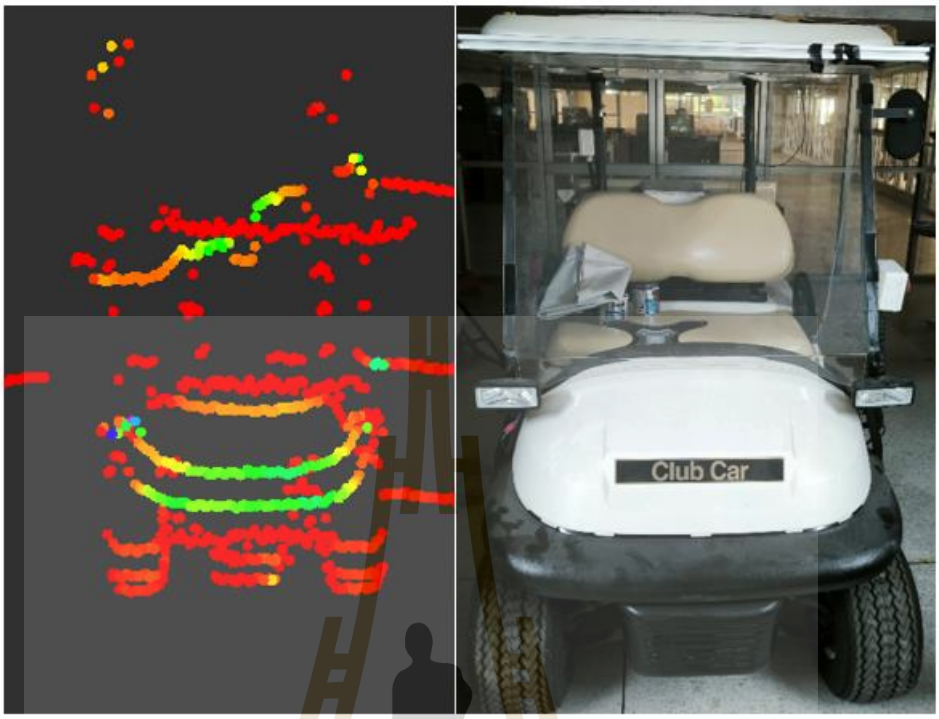
รูปที่ 3.8 ภาพกลุ่มพอยคลาวด์ของคนจากเซนเซอร์ LiDAR และภาพคนจากกล้อง

ในภาพที่ 3.8 จะแสดงภาพกลุ่มพอยคลาวด์ของคนจากเซนเซอร์จะเห็นว่ามีกลุ่มของจุดพอยคลาวด์ที่ถูกสแกนด้วยเลเซอร์กระจายเป็นกลุ่มของจุดเป็นเส้นแนวนอน แบ่งเป็นชั้นๆซึ่งบอกถึงรูปร่างลักษณะของคนที่ถูกสแกนมา และภาพคนจากกล้องที่ถ่ายมาเพื่อแสดงความแตกต่างของข้อมูลทั้งสองประเภท



รูปที่ 3.9 ภาพกลุ่มพอยคลาวด์ของคนขี่จักรยานจากเซนเซอร์ LiDAR และภาพคนขี่จักรยานจากกล้อง

ในภาพที่ 3.9 จะแสดงภาพกลุ่มพอยคลาวด์ของคนขี่จักรยานจากเซนเซอร์จะเห็นว่ามีกลุ่มของจุดพอยคลาวด์ที่ถูกสแกนด้วยเลเซอร์กระจายเป็นกลุ่มของจุดเป็นเส้นแนวนอน แบ่งเป็นชั้น ๆ ซึ่งบอกถึงรูปร่างลักษณะของคนขี่จักรยานที่ถูกสแกนมา และภาพคนขี่จักรยานจากกล้องที่ถ่ายมาเพื่อแสดงความแตกต่างของข้อมูลทั้งสองประเภท

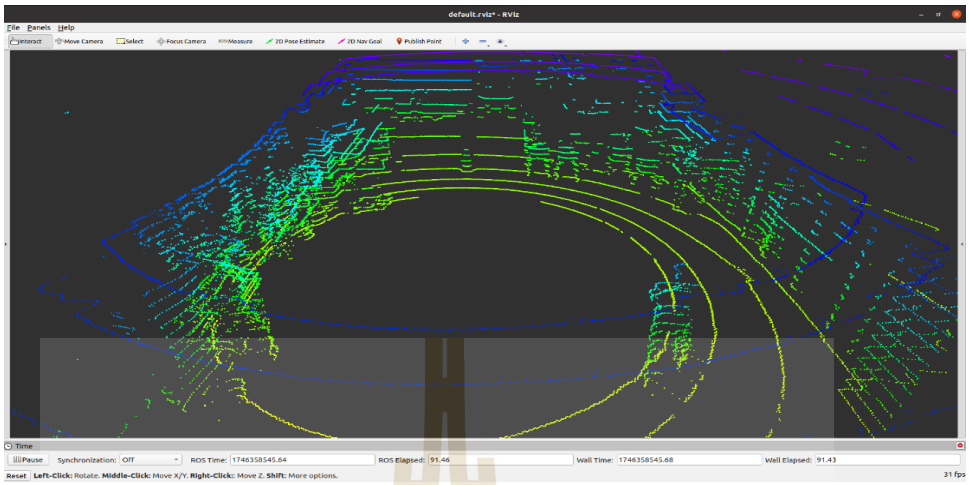


รูปที่ 3.10 ภาพกลุ่มพอยคลาวด์ของรถจากเซนเซอร์ LiDAR และภาพรถจากกล้อง

ในภาพที่ 3.10 จะแสดงภาพกลุ่มพอยคลาวด์ของรถจากเซนเซอร์จะเห็นว่ามีกลุ่มของจุดพอยคลาวด์ที่ถูกสแกนด้วยเลเซอร์กระจายเป็นกลุ่มของจุดเป็นเส้นแนวนอน แบ่งเป็นชั้น ๆ ซึ่งบอกถึงรูปร่างลักษณะของรถที่ถูกสแกนมา และภาพรถกล้องที่ถ่ายมาเพื่อแสดงความแตกต่างของข้อมูลทั้งสองประเภท

3.2.2 การกรองข้อมูล

การกรองข้อมูล เป็นกระบวนการที่ใช้ลดจำนวนพอยคลาวด์โดยจะใช้ Passthrough filter ซึ่งเป็น 1 ในเครื่องมือของ PCL หรือ Point Cloud Library ที่ช่วยจัดการข้อมูลพอยคลาวด์ โดยการกำหนดขอบเขตในแกน X, Y, หรือ Z และเฉพาะจุดข้อมูลที่ตกอยู่ในขอบเขตที่กำหนดเท่านั้นที่จะถูกเก็บไว้ ด้วยวิธีการนี้จะช่วยลดจุดข้อมูลที่อยู่นอกขอบเขตที่สนใจ ทำให้ชุดข้อมูลมีความเข้มข้นและเฉพาะเจาะจงมากขึ้น นอกจากนี้จะทำให้กลุ่มพอยคลาวด์ที่เข้าอัลกอลิทึม PointPillars ลดลง ทำให้ลดภาระในการคำนวณของตัวประมวลผล



รูปที่ 3.11 ภาพก่อนการกรองข้อมูล LiDAR บนซอฟต์แวร์ rviz

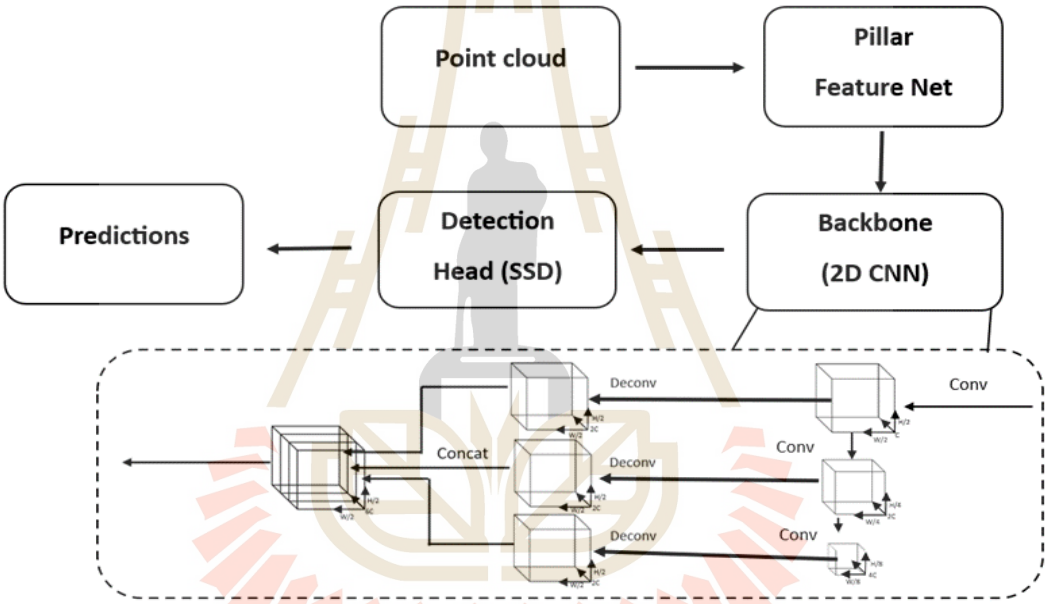
ในภาพที่ 3.11 จะแสดงภาพก่อนการกรองข้อมูลของเซนเซอร์ LiDAR จะเห็นว่ามีกลุ่มข้อมูลพอยคลาวด์ที่เกิดจากการเลเซอร์สแกนของเซนเซอร์ LiDAR กระจายอยู่ทุกทีในพื้นที่ทดสอบรอบตัวหุ่นยนต์

รูปที่ 3.12 ภาพหลังจากการกรองข้อมูล LiDAR บนซอฟต์แวร์ rviz

ในภาพที่ 3.12 จะแสดงภาพหลังการกรองข้อมูลของเซนเซอร์ LiDAR จะเห็นว่ามีกลุ่มข้อมูลพอยคลาวด์ที่เกิดจากการเลเซอร์สแกนของเซนเซอร์ LiDAR ในพื้นที่ทดสอบถูกลบออกให้เหลือแต่เฉพาะส่วนต้องการให้ตัวหุ่นยนต์เห็น เพื่อให้ในการเก็บข้อมูลต่อไป

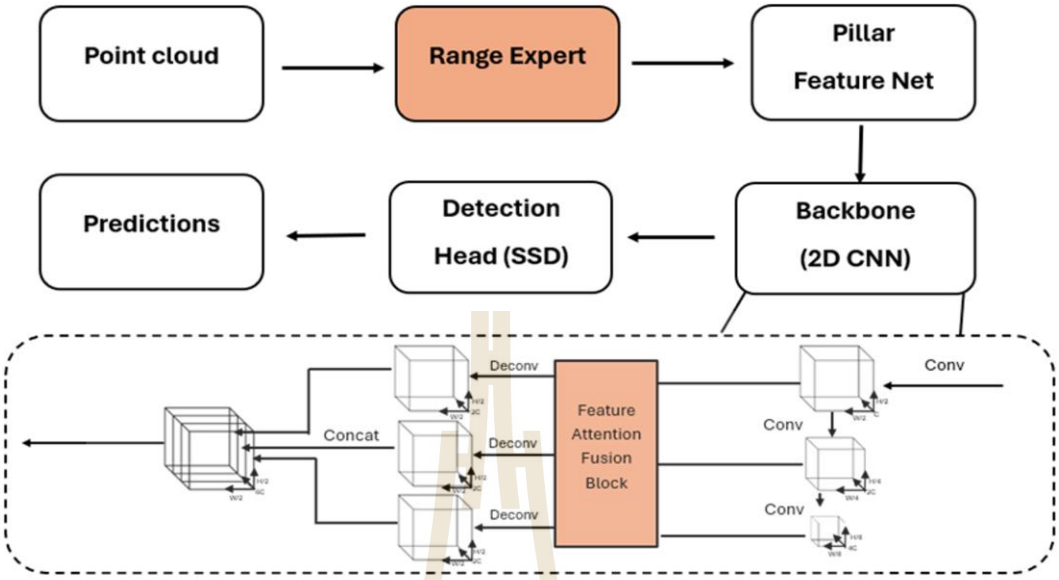
3.2.3 การปรับแต่งโครงสร้าง

โมเดล PointPillars เป็นหนึ่งในสถาปัตยกรรมการตรวจจับวัตถุ 3 มิติจากข้อมูล Point Cloud ที่ได้รับความนิยม เนื่องจากสามารถแปลงข้อมูลสามมิติให้เป็นภาพมุมมองจากด้านบน (Bird's Eye View) เพื่อนำเข้าโครงข่ายคอนโวลูชัน (2D CNN) อย่างมีประสิทธิภาพ อย่างไรก็ตาม สถาปัตยกรรมดั้งเดิมของ PointPillars ถูกออกแบบมาเพื่อทำงานบนระบบที่มีทรัพยากรประมวลผลสูง เช่น คอมพิวเตอร์ที่ใช้ GPU ขนาดใหญ่ ซึ่งไม่เหมาะสมกับการนำไปใช้ในอุปกรณ์ฝังตัว (Embedded Systems) หรือระบบหุ่นยนต์อัตโนมัติที่มีข้อจำกัดด้านพลังงานและการประมวลผล เช่น Jetson Xavier AGX

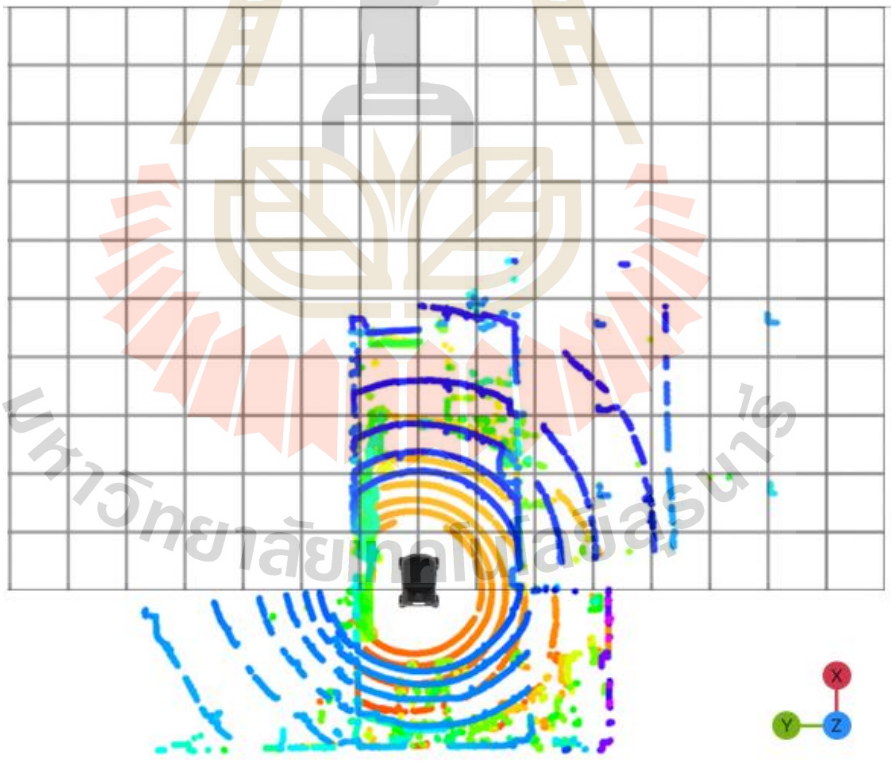


รูปที่ 3.13 กระบวนการของ Pointpillar ก่อนปรับปรุง

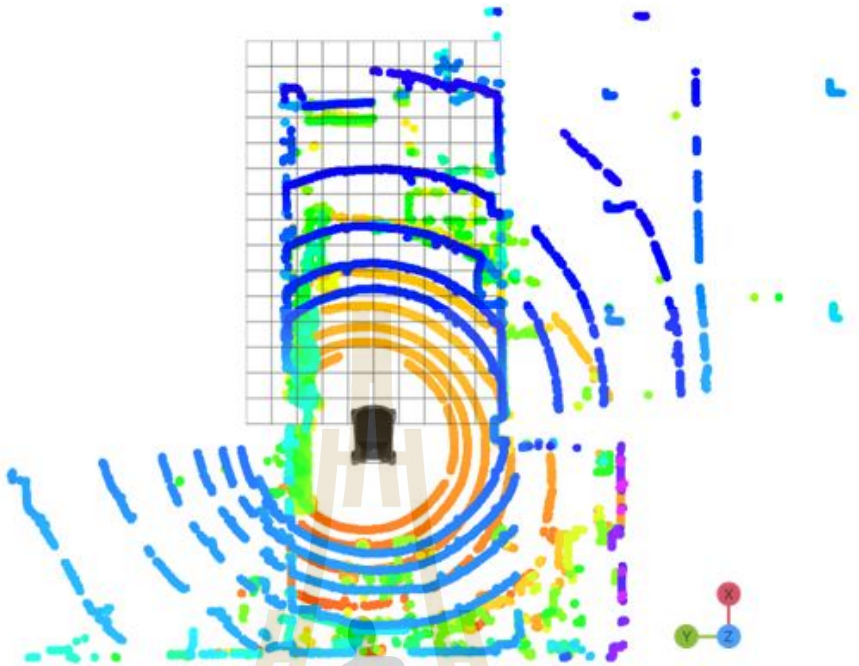
ดังนั้น ในการวิจัยนี้ได้มีการปรับแต่งโครงสร้างของ PointPillars เพื่อให้สามารถประมวลผลได้อย่างมีประสิทธิภาพบนแพลตฟอร์ม Edge AI โดยเฉพาะในบริบทของหุ่นยนต์เคลื่อนที่ โดยมีแนวทางการปรับแต่งสำคัญดังนี้



รูปที่ 3.14 กระบวนการของ Pointpillar ที่ปรับปรุงแล้ว



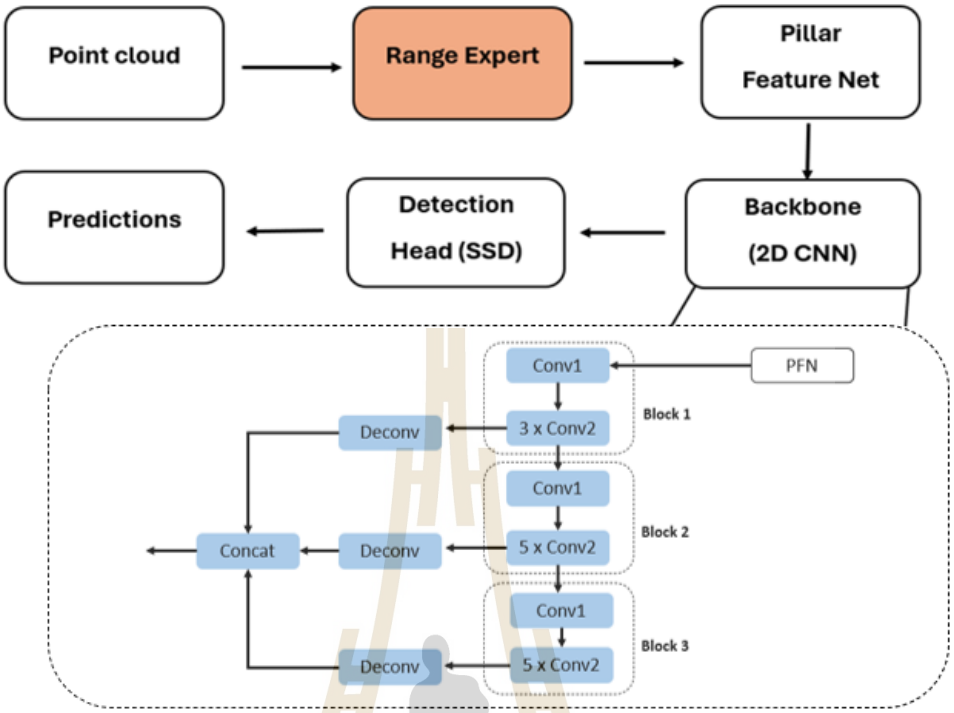
รูปที่ 3.15 พื้นที่การตรวจจับพอยคลาวด์และวอกเซลขนาดใหญ่



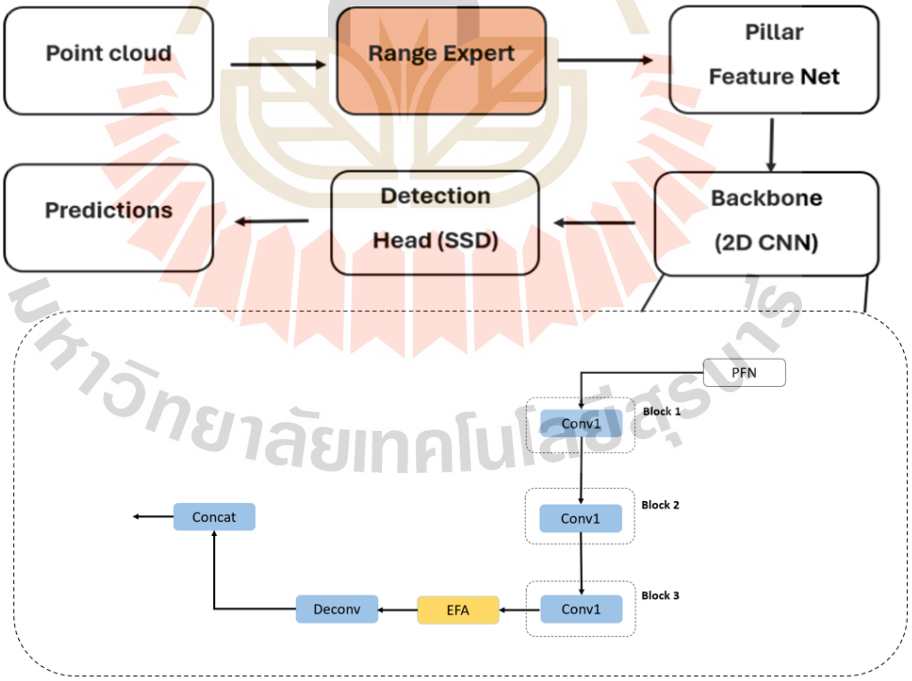
รูปที่ 3.16 พื้นที่การตรวจจับพอยคลาวด์และวอกเซลขนาดเล็ก

ในการปรับขนาดของ Voxel Grid: กำหนดขนาดของ voxel ที่เหมาะสมในรูปแบบ [x,y,z] ในรูปที่ 3.15 ใช้ขนาด [0.16, 0.16, 4.0] ซึ่งเป็น voxel ที่ใหญ่จะช่วยลดจำนวน voxel และลดเวลาในการประมวลผลเพื่อควบคุมจำนวน voxel ต่อเฟรมให้เหมาะสมกับทรัพยากรประมวลผล แต่จะสูญเสียความแม่นยำ ในขณะที่รูปที่ 3.16 ใช้ระยะที่ [0.1,0.1,4.0] เป็น voxel ที่เล็กจะให้รายละเอียดสูงแต่เพิ่มภาระการคำนวณแต่จะได้ความแม่นยำมากกว่า ซึ่งสามารถเลือกใช้ตามความเหมาะสมของงาน เช่น การตรวจจับระยะใกล้หรือไกล

การจำกัดขอบเขตของ Point Cloud (Point Cloud Range): กำหนดขอบเขตของพื้นที่ตรวจจับในรูปแบบ $[x_1,y_1,z_1,x_2,y_2,z_2]$ โดยจะกำหนดขอบเขตจากในแกน x คือ x_1 ถึง x_2 ในแกน y คือ y_1 ถึง y_2 ในแกน z คือ z_1 ถึง z_2 ตามลำดับ ในรูปที่ 3.15 ใช้ระยะที่ [0, -40, -3, 70, 40, 1] เป็นปรับขนาดพื้นที่ตรวจจับที่ใหญ่จะให้การตรวจจับที่กว้างและสามารถมองเห็นวัตถุไกลได้แต่เพิ่มภาระการคำนวณ แต่ขณะที่รูปที่ 3.16 ใช้ระยะที่ [0, -3, -1, 7, 3, 3] เมตร ที่ปรับขนาดพื้นที่ตรวจจับให้ลดลงจะให้การตรวจจับที่แคบและสามารถมองเห็นวัตถุใกล้ เพื่อลดข้อมูลจุดที่ไม่จำเป็นและมุ่งเน้นเฉพาะพื้นที่รอบหุ่นยนต์ ซึ่งช่วยลดจำนวน voxel ที่ต้องประมวลผลและเพิ่มความเร็วในการตรวจจับ โดยแนวทางนี้ถือเป็นการปรับใช้แนวคิดของ Range Expert

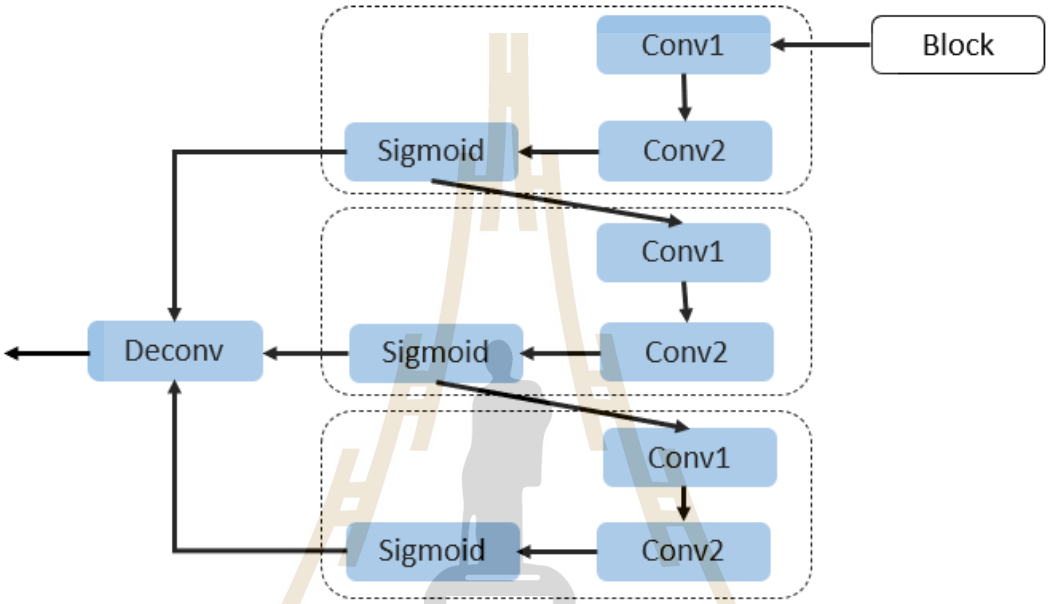


รูปที่ 3.17 กระบวนการของ Backbone ใน Pointpillar ก่อนปรับปรุง



รูปที่ 3.18 กระบวนการของ Backbone ใน Pointpillar ที่ปรับปรุงแล้ว

ในภาพที่ 3.18 คือ ภาพรวมของการปรับแต่งโมเดลจะประกอบด้วยวิธีการนำการปรับพื้นที่การตรวจจับพอยคลาวด์และวอกเซลของแนวคิดของ Range Expert เพื่อลดภาระการคำนวณ และการ Lightweight Attention Pyramid Network (LAPN) มาช่วยให้เครือข่ายสามารถโฟกัสกับลักษณะที่สำคัญและลดความฟุ้งของข้อมูลที่ไม่เกี่ยวข้องได้อย่างมีประสิทธิภาพ



รูปที่ 3.19 กระบวนการของ Attention Pyramid Network หรือ EFA ใน Pointpillar

Original PointPillars Model

Voxel Feature Encoder

- `Linear(10 → 64) → BatchNorm1d → ReLU`

BEV Backbone

- **Block 0**:
  - `ZeroPad2d`
  - `1 x Conv2d(64 → 32, stride=2)` with `BatchNorm2d + ReLU`
  - `3 x Conv2d(32 → 32, stride=1)` with `BatchNorm2d + ReLU`
- **Block 1**:
  - `ZeroPad2d`
  - `1 x Conv2d(32 → 64, stride=2)` with `BatchNorm2d + ReLU`
  - `5 x Conv2d(64 → 64, stride=1)` with `BatchNorm2d + ReLU`
- **Block 2**:
  - `ZeroPad2d`
  - `1 x Conv2d(64 → 128, stride=2)` with `BatchNorm2d + ReLU`
  - `5 x Conv2d(128 → 128, stride=1)` with `BatchNorm2d + ReLU`
- **Upsample / Deblocks**:
  - `ConvTranspose2d(32 → 128, stride=1)`
  - `ConvTranspose2d(64 → 128, stride=2)`
  - `ConvTranspose2d(128 → 128, stride=4)`
- **Concat outputs** → `128 × 3 = 384 channels`

Detection Head

- `Conv2d(384 → 18) → Classification`
- `Conv2d(384 → 42) → Bounding Box`
- `Conv2d(384 → 12) → Direction`
- **Loss**: Focal + SmoothL1 + CrossEntropy

รูปที่ 3.20 โครงสร้าง 2D Backcone ของ Pointpillar ก่อนปรับปรุง

Enhancing PointPillars Model

Voxel Feature Encoder

- `Linear(10 → 64) → BatchNorm1d → ReLU`

BEV Backbone (Lightweight + EFA)

- **Block 0**:
  - `1 x Conv2d(64 → 32, stride=2)`
  - `BatchNorm2d + ReLU`
- **Block 1**:
  - `1 x Conv2d(32 → 64, stride=2)`
  - `BatchNorm2d + ReLU`
- **Block 2**:
  - `1 x Conv2d(64 → 128, stride=2)`
  - `BatchNorm2d + ReLU`
  - `EfficientFeatureAttentionBlock (in: 128, mid: 8, out: 128)`
- **Upsample / Deblocks**:
  - `ConvTranspose2d(32 → 128, stride=1)`
  - `ConvTranspose2d(64 → 128, stride=2)`
  - `ConvTranspose2d(128 → 128, stride=4)`
- **Concat outputs** → `128 × 3 = 384 channels`

Detection Head

- `Conv2d(384 → 18) → Classification`
- `Conv2d(384 → 42) → Bounding Box`
- `Conv2d(384 → 12) → Direction`
- **Loss**: Focal + SmoothL1 + CrossEntropy

รูปที่ 3.21 โครงสร้าง 2D Backcone ของ Pointpillar ที่ปรับปรุงแล้ว

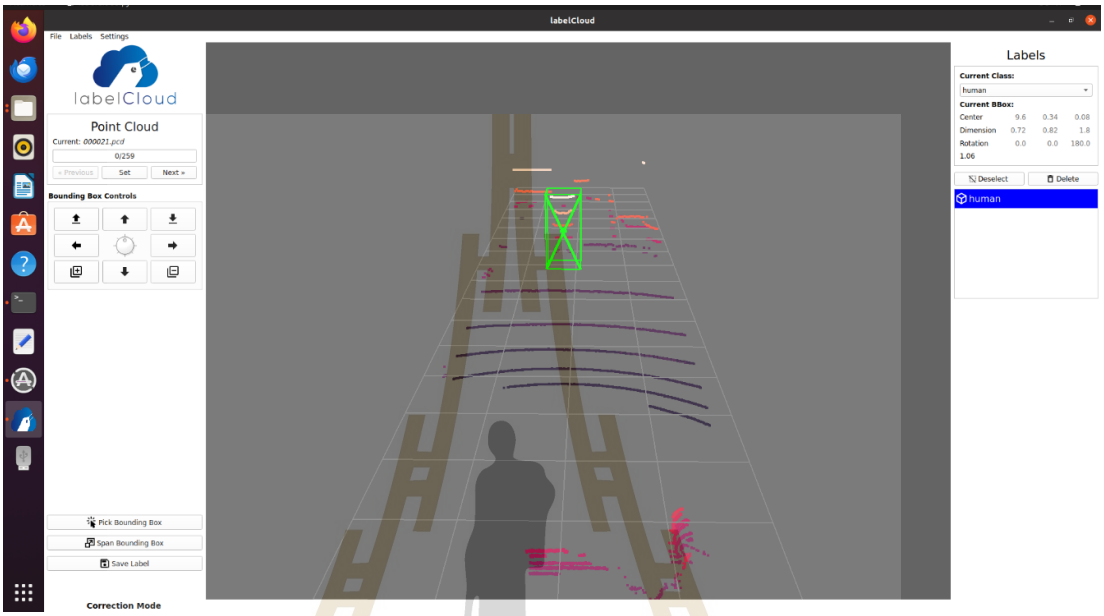
ในภาพที่ 3.21 การปรับแต่งโมเดล PointPillars ที่เสนอในงานวิจัยนี้ ได้มีการเปลี่ยนแปลงโครงสร้าง Backbone โดยแทนที่สถาปัตยกรรม 2D CNN ดั้งเดิมประกอบด้วยบล็อกคอนโวลูชันสามชั้นที่มีจำนวนช่องและขนาด kernel ที่เหมาะสม ได้แก่ Conv2D(64⟶32), Conv2D(32⟶64), และ Conv2D(64⟶128) ตามลำดับ พร้อมการใช้ Deconvolution เพื่อขยายขนาดฟีเจอร์แมปกลับมา และทำการรวมผลลัพธ์จากหลายระดับความละเอียดเพื่อเพิ่มการเรียนรู้ลักษณะที่มีความซับซ้อนในหลายสเกล

ด้วยการนำ Lightweight Attention Pyramid Network (LAPN) ซึ่งมีการฝังหน่วย Feature Attention Block ลงในช่วงการรวมฟีเจอร์เพื่อปรับน้ำหนักของข้อมูลที่สำคัญในแต่ละช่องสัญญาณ (feature channel) ช่วยให้เครือข่ายสามารถโฟกัสกับลักษณะที่สำคัญและลดความฟุ้งของข้อมูลที่ไม่เกี่ยวข้องได้อย่างมีประสิทธิภาพ สำหรับส่วน Detection Head ได้ใช้หัวตรวจจับแบบ Single Shot Detector (SSD) ที่ประกอบด้วยชั้นคอนโวลูชัน 3 ช่อง ได้แก่ Conv2D (384⟶18) สำหรับการจัดกลุ่ม (classification), Conv2D(384⟶42) สำหรับการประมาณกรอบ (bounding box regression) และ Conv2D (384⟶6) สำหรับการพิจารณาทิศทาง (direction classification)

3.2.4 การตรวจจับวัตถุ 3 มิติจากชุดข้อมูลพอยคลาวด์

การตรวจจับวัตถุ 3 มิติจากชุดข้อมูลพอยคลาวด์ (3D Object Detection) เป็นกระบวนการที่ใช้ในการระบุและแยกประเภทวัตถุที่ปรากฏในข้อมูลพอยคลาวด์ที่เกิดจากการสแกนเลเซอร์ของเซนเซอร์ LiDAR โดยทั่วไปแล้วจะใช้เทคโนโลยีปัญญาประดิษฐ์ตรวจจับ 3 มิติ เพื่อประมวลผลพอยคลาวด์และตรวจจับวัตถุที่อยู่ในพอยคลาวด์อย่างรวดเร็วและมีประสิทธิภาพ Pointpillar เป็นหนึ่งในโมเดลที่ถูกใช้อย่างแพร่หลายใน การตรวจจับวัตถุ 3 มิติ เนื่องจากสามารถประมวลผลได้อย่างรวดเร็วและแม่นยำ โดยโมเดลนี้จะสร้างพื้นที่สำหรับการตรวจจับออกเป็นตารางกริดที่ถูกกำหนดไว้ก่อนแล้วนำข้อมูลในกริดไปแปลงมุมพิจารณาข้อมูลเป็นมุมสูง ทำให้ลดข้อมูลในการประมวลผลของตัวประมวลและสามารถตรวจจับวัตถุในพอยคลาวด์ได้อย่างรวดเร็ว แล้วใช้เครือข่ายประสาทเทียมแบบคอนโวลูชัน (CNN) ดึงคุณลักษณะกลุ่มพอยคลาวด์ แล้วนำไปเข้าสู่หัวตรวจจับ (Detection Head) สำหรับกระบวนการจำแนกประเภท (Classification) และการระบุตำแหน่งเชิงสามมิติ (3D Bounding Box Regression) ของวัตถุที่ตรวจจับได้ ในการรวบรวมพอยคลาวด์นั้นจะทำการบันทึกพอยคลาวด์เป็น rosbag ซึ่งเป็นเครื่องมือบันทึกและการเล่นซ้ำข้อมูลจากเซนเซอร์และการสื่อสารภายในระบบ ROS เพื่อใช้ในการเก็บข้อมูลโดยที่ rosbag จะเป็นสถานที่ที่ต้องการใช้งานของโมเดลเพื่อเพิ่มความแม่นยำยิ่งขึ้นในการใช้งานโมเดล

การเลือกพอยคลาวด์ที่ใช้ในการติดป้ายกำกับ เนื่องจากการฝึกปัญญาประดิษฐ์ต้องการข้อมูลที่มีความหลากหลาย พอยคลาวด์ที่เลือกจะต้องครอบคลุมหลายสถานการณ์ทั้งในระยะใกล้และไกลของวัตถุ เพื่อให้โมเดลสามารถเรียนรู้และทำนายได้อย่างแม่นยำในสถานการณ์ต่าง ๆ ที่อาจเกิดขึ้นได้



รูปที่ 3.22 ขั้นตอนการติดป้ายกำกับโดยใช้เครื่องมือ LabelCloud

รูปที่ 3.23 ขั้นตอนการแบ่งชุดข้อมูลก่อนนำไปสร้างโมเดลการเรียนรู้

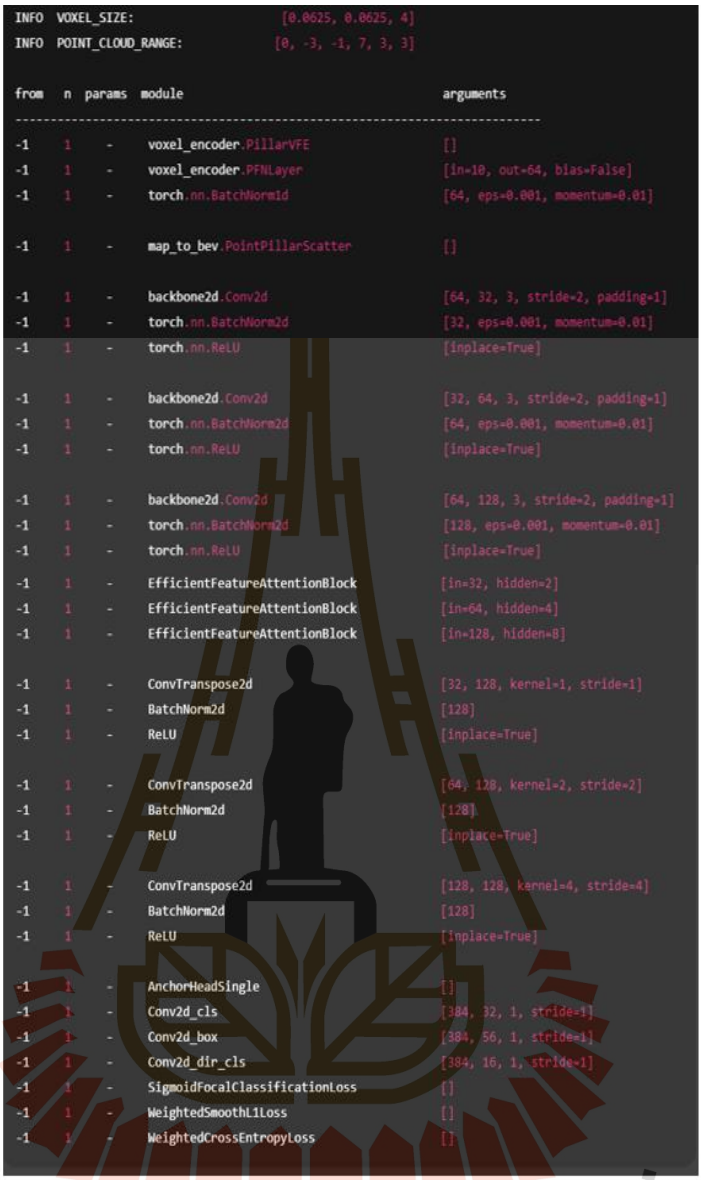
หลังจากที่รวบรวมชุดข้อมูลพอยคลาวด์ได้แล้ว ชุดข้อมูลเหล่านั้นจะถูกนำไปผ่านกระบวนการติดป้ายกำกับโดยใช้ LabelCloud ซึ่งเป็นเครื่องมือที่ช่วยในการระบุวัตถุในกลุ่มพอยคลาวด์โดยใช้ Bounding Box กำหนดขอบเขตวัตถุที่ต้องการตรวจจับ ข้อมูลที่ติดป้ายกำกับจะถูกแบ่งออกเป็นสามชุด คือ ชุดข้อมูลสำหรับการฝึกฝนโมเดล ชุดข้อมูลสำหรับการตรวจสอบความแม่นยำ และ ชุดข้อมูลสำหรับการทดสอบโมเดล โดยข้อมูล ถูกแบ่งออกเป็น 80% สำหรับการฝึกฝน 10% สำหรับการทดสอบ และ 10% สำหรับการตรวจสอบ

3.2.5 การสร้างโมเดลการเรียนรู้

การสร้างโมเดลการเรียนรู้เริ่มจากการเตรียมข้อมูลที่ติดป้ายกำกับ (Labeling) ผ่านซอฟต์แวร์ labelCloud ข้อมูลเหล่านี้ถูกแบ่งออกเป็นชุดฝึกสอน (Training Set), ชุดตรวจสอบโมเดล (Validation Set), และชุดทดสอบ (Test Set) เพื่อให้โมเดลสามารถเรียนรู้และทดสอบได้อย่างแม่นยำ การใช้ Pointpillar การฝึก โมเดลนี้มีจึงความสำคัญ เนื่องจาก Pointpillar ถูกออกแบบมาให้สามารถตรวจจับวัตถุได้รวดเร็วและแม่นยำ โมเดลนี้สามารถเรียนรู้ข้อมูลจากกลุ่มพอยคลาวด์ในการตรวจจับ 3 มิติ

รูปที่ 3.24 คำสั่งที่ใช้ในการฝึกสอนโมเดล 9000 รอบ (Epochs) และปรับขนาดแบตช์เป็น 7

ขั้นตอนการฝึกโมเดลเริ่มต้นจากการนำข้อมูลที่เตรียมไว้เข้าสู่ตารางกริดที่ถูกกำหนดไว้ ไปสู่ Pillar Feature net ที่จะแปลงมุมพิจารณาข้อมูลเป็นมุมสูง แล้วใช้เครือข่ายประสาทเทียมแบบคอนโวลูชัน (CNN) ดึงคุณลักษณะกลุ่มพอยคลาวด์ แล้วนำไปเข้าสู่หัวตรวจจับ (Detection Head) สำหรับกระบวนการจำแนกประเภท (Classification) และการระบุตำแหน่งเชิงสามมิติ (3D Bounding Box Regression) ของวัตถุที่ตรวจจับได้ ซึ่ง Pointpillar สามารถตรวจจับวัตถุได้อย่างมีประสิทธิภาพในกลุ่มพอยคลาวด์ของการสแกนแต่ละครั้ง กระบวนการฝึกฝนโมเดลจะถูกตั้งค่าให้ทำการฝึกเป็นจำนวน 9000 รอบ (Epochs) ซึ่งช่วยให้โมเดลมีโอกาสเรียนรู้จากข้อมูลที่ป้อนเข้ามามากขึ้น โดยมีการปรับขนาดแบตช์เป็น 7 ทำให้การฝึกเร็วขึ้น ขนาดนี้เหมาะสมต่อการฝึกโมเดลโดยไม่ลดทอนประสิทธิภาพหรือเพิ่มภาระการประมวลผลที่สูงเกินไป



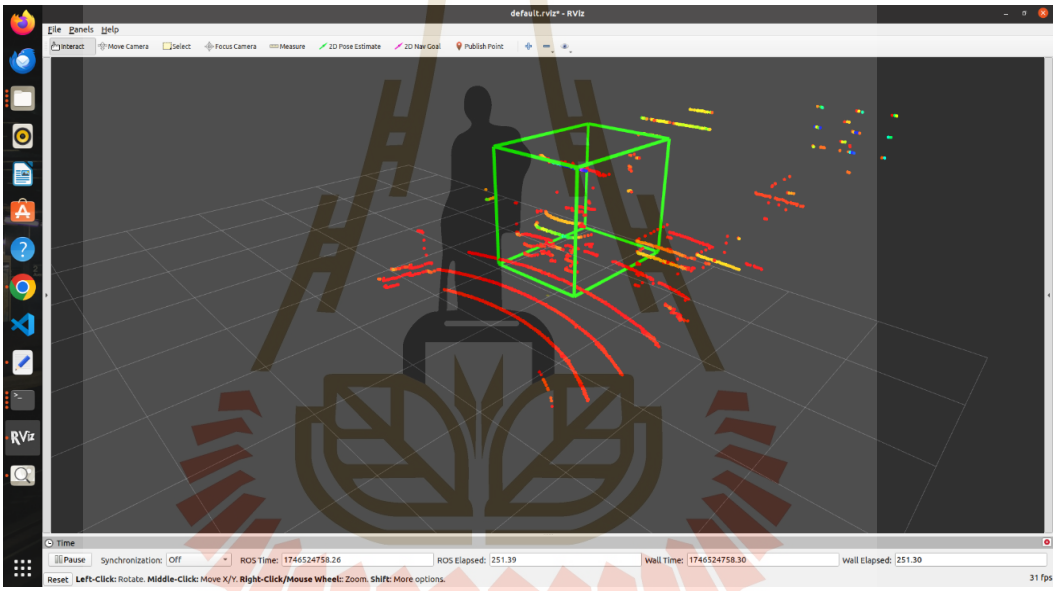
```
INFO  VOXEL_SIZE:                [0.0625, 0.0625, 4]
INFO  POINT_CLOUD_RANGE:         [0, -3, -1, 7, 3, 3]

from   n  params  module                                  arguments
-------------------------------------------------------------------------
-1     1     -    voxel_encoder.PillarVFE                 []
-1     1     -    voxel_encoder.PFNLayer                  [in=10, out=64, bias=False]
-1     1     -    torch.nn.BatchNorm1d                    [64, eps=0.001, momentum=0.01]

-1     1     -    map_to_bev.PointPillarScatter           []

-1     1     -    backbone2d.Conv2d                       [64, 32, 3, stride=2, padding=1]
-1     1     -    torch.nn.BatchNorm2d                    [32, eps=0.001, momentum=0.01]
-1     1     -    torch.nn.ReLU                           [inplace=True]

-1     1     -    backbone2d.Conv2d                       [32, 64, 3, stride=2, padding=1]
-1     1     -    torch.nn.BatchNorm2d                    [64, eps=0.001, momentum=0.01]
-1     1     -    torch.nn.ReLU                           [inplace=True]

-1     1     -    backbone2d.Conv2d                       [64, 128, 3, stride=2, padding=1]
-1     1     -    torch.nn.BatchNorm2d                    [128, eps=0.001, momentum=0.01]
-1     1     -    torch.nn.ReLU                           [inplace=True]
-1     1     -    EfficientFeatureAttentionBlock          [in=32, hidden=2]
-1     1     -    EfficientFeatureAttentionBlock          [in=64, hidden=4]
-1     1     -    EfficientFeatureAttentionBlock          [in=128, hidden=8]

-1     1     -    ConvTranspose2d                         [32, 128, kernel=1, stride=1]
-1     1     -    BatchNorm2d                             [128]
-1     1     -    ReLU                                    [inplace=True]

-1     1     -    ConvTranspose2d                         [64, 128, kernel=2, stride=2]
-1     1     -    BatchNorm2d                             [128]
-1     1     -    ReLU                                    [inplace=True]

-1     1     -    ConvTranspose2d                         [128, 128, kernel=4, stride=4]
-1     1     -    BatchNorm2d                             [128]
-1     1     -    ReLU                                    [inplace=True]

-1     1     -    AnchorHeadSingle                        []
-1     1     -    Conv2d_cls                              [384, 32, 1, stride=1]
-1     1     -    Conv2d_box                              [384, 56, 1, stride=1]
-1     1     -    Conv2d_dir_cls                          [384, 16, 1, stride=1]
-1     1     -    SigmoidFocalClassificationLoss          []
-1     1     -    WeightedSmoothL1Loss                    []
-1     1     -    WeightedCrossEntropyLoss                []
```

รูปที่ 3.25 โครงสร้างของ Pointpillar ที่ปรับปรุงแล้ว

การฝึกโมเดลจะใช้ฟังก์ชันการสูญเสีย (Loss Function) ซึ่งทำหน้าที่ประเมินความแตกต่างระหว่างผลลัพธ์ที่โมเดลทำนายและค่าจริงที่กำหนดไว้ หากโมเดลทำการทำนายผิดพลาด ฟังก์ชันการสูญเสียจะช่วยปรับปรุงค่าพารามิเตอร์ในโมเดลให้แม่นยำมากขึ้นในรอบถัดไป การปรับปรุงค่าพารามิเตอร์นี้จะทำให้โมเดลสามารถทำนายวัตถุได้อย่างแม่นยำมากยิ่งขึ้นในแต่ละรอบ

หลังจากที่โมเดลทำการฝึกฝนแล้ว จะนำโมเดลไปทดสอบด้วยชุดข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน เพื่อประเมินความสามารถในการตรวจจับวัตถุ การทดสอบโมเดลนี้ใช้เพื่อตรวจสอบว่าโมเดลสามารถนำไปใช้ในสถานการณ์จริงแม่นยำแค่ไหน โดยข้อมูลที่ใช้ในการทดสอบจะเป็นพอย

คลาด์ที่ถูกเก็บไว้แยกเพื่อใช้ประเมินประสิทธิภาพของโมเดล ซึ่งจะมีการเปรียบเทียบผลลัพธ์ที่ได้กับคำที่แท้จริงในระหว่างการฝึกโมเดลจะมีการแสดงค่า Confusion Matrix เพื่อให้สามารถวิเคราะห์ผลการตรวจจับของโมเดลได้อย่างละเอียด เครื่องมือนี้จะช่วยให้ทราบถึงความถูกต้องของการตรวจจับในแต่ละหมวดหมู่วัตถุที่สนใจ และปรับปรุงโมเดลให้มีประสิทธิภาพดียิ่งขึ้นในการทำนายตำแหน่งและขนาดของวัตถุ โมเดลจะถูกนำไปใช้งานในการทำนายผลจากชุดข้อมูลที่ยังไม่เคยมีการฝึกฝนมาก่อน (Inference) การทำนายผลนี้ช่วยให้สามารถตรวจสอบได้ว่าโมเดลที่สร้างขึ้นมีความสามารถในการตรวจจับวัตถุได้จริงในสถานการณ์ที่หลากหลายหรือไม่ การแสดงผลของโมเดลจะถูกแสดงผ่าน Bounding Box รอบวัตถุที่ถูกตรวจจับ ทำให้สามารถตรวจสอบได้อย่างชัดเจนว่าผลลัพธ์ตรงกับสิ่งที่ต้องการหรือไม่



รูปที่ 3.26 ทดสอบการใช้โมเดลตรวจจับวัตถุ 3 มิติที่ได้จากการฝึกฝน

3.2.6 การทดสอบภาระการคำนวณบนหุ่นยนต์

Jetson stats เป็นซอฟต์แวร์ที่จะใช้เพื่อตรวจสอบความเหมาะสมของสถาปัตยกรรม PointPillars ที่ได้รับการปรับแต่งสำหรับการทำงานบนอุปกรณ์ฝังตัวที่มีทรัพยากรจำกัด เช่น Jetson Xavier AGX การทดสอบภาระการคำนวณจึงถูกออกแบบขึ้นเพื่อประเมินประสิทธิภาพของระบบในบริบทการใช้งานจริง โดยมุ่งเน้นที่การวัดค่าการใช้ทรัพยากรหลักของระบบ ได้แก่ หน่วยประมวลผลกลาง (CPU), หน่วยประมวลผลกราฟิก (GPU และอัตราเฟรมต่อวินาที (Frame Per Second: FPS) ซึ่งเป็นตัวชี้วัดความสามารถในการประมวลผลแบบเรียลไทม์ของโมเดล

เครื่องมือที่ใช้ในการทดสอบ คือ Jetson Stats (jtop) ซึ่งเป็นซอฟต์แวร์โอเพ่นซอร์สที่พัฒนาให้ใช้เฉพาะสำหรับแพลตฟอร์ม Jetson โดยสามารถแสดงผลการใช้งานทรัพยากรในระบบแบบเรียลไทม์ รวมถึงรองรับการส่งออกข้อมูลในรูปแบบที่สามารถนำไปวิเคราะห์ได้เพิ่มเติม นอกจากนี้ยังมีการใช้เทคนิคการจับเวลาภายในโค้ดเพื่อวัดระยะเวลาที่ใช้ในการประมวลผลต่อเฟรม ซึ่งสามารถคำนวณเป็นค่า

- **CPU และ GPU Usage (%)**: แสดงสัดส่วนของการใช้งานทรัพยากรประมวลผล เพื่อระบุระดับภาระของระบบขณะทำงาน

กระบวนการทดสอบดำเนินการโดยการนำโมเดลที่ได้รับการปรับแต่งไปประมวลผลข้อมูล Point Cloud แบบต่อเนื่องภายใน ROS Node และบันทึกค่าการใช้งานทรัพยากรของระบบในระหว่างการรัน การประเมินผลจะพิจารณาทั้งค่าเฉลี่ยและค่าสูงสุดของแต่ละตัวชี้วัด จากนั้นเปรียบเทียบกับโมเดล PointPillars ต้นฉบับ เพื่อแสดงให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้นทั้งในด้านความเร็วและการลดการใช้ทรัพยากร

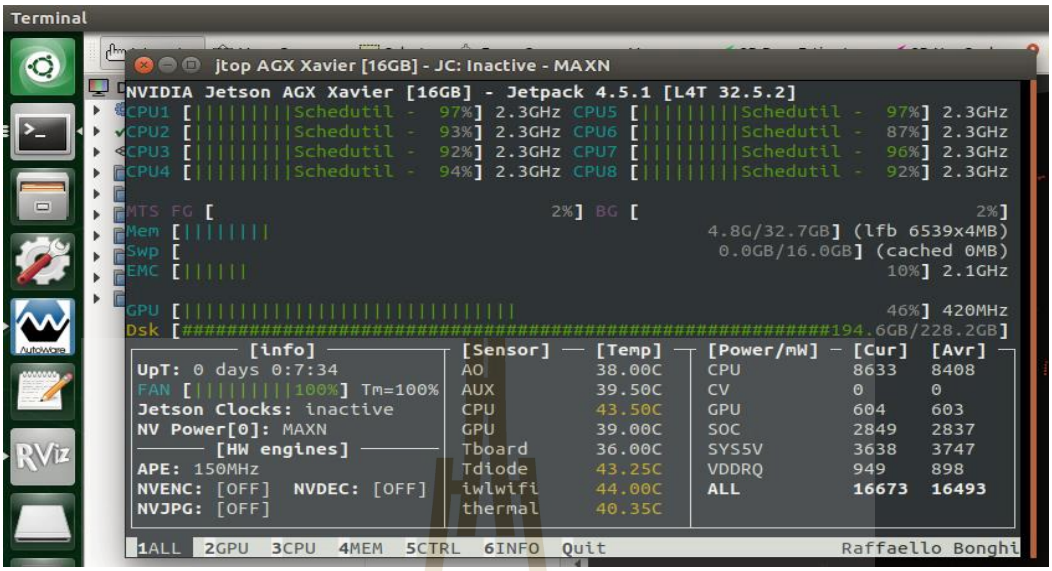
ผลลัพธ์จากการทดสอบดังกล่าวมีความสำคัญต่อการนำโมเดลไปใช้งานจริงในหุ่นยนต์อัตโนมัติที่ต้องการการประมวลผลแบบทันทีภายใต้ข้อจำกัดของฮาร์ดแวร์ โดยเฉพาะในสภาพแวดล้อมที่มีการเปลี่ยนแปลงตลอดเวลา ซึ่งระบบจำเป็นต้องตอบสนองอย่างรวดเร็วเพื่อความปลอดภัยและความเสถียรในการทำงาน เพื่อที่จะสามารถเปรียบเทียบการวัดผลภาระการคำนวณบนหุ่นยนต์ได้นั้นจะแบ่งสถานการณ์ที่ทดสอบ 4 กรณี ดังนี้

กรณี 1: เปิดเครื่องฮาร์ดแวร์ ไม่รันโปรแกรมใด ๆ

กรณี 2: เปิดระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดล

กรณี 3: รันเฉพาะโมเดล PointPillars ทั้งบน Jetson Xavier และ Onboard Computer

กรณี 4: รันโมเดล PointPillars พร้อมระบบขับเคลื่อนอัตโนมัติเต็มระบบ



รูปที่ 3.27 แสดงหน้าต่าง Jetson stats ที่ใช้วัดภาระการคำนวณของค่า CPU และ GPU

3.2.7 การทดสอบการวิเคราะห์เพื่อหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้

การทดสอบในหัวข้อนี้มุ่งเน้นการนำผลลัพธ์จากการตรวจจับวัตถุด้วยโมเดล PointPillars มาวิเคราะห์เพื่อประเมินระยะห่างระหว่างหุ่นยนต์อัตโนมัติและวัตถุในสภาพแวดล้อม โดยระยะห่างนี้มีความสำคัญอย่างยิ่งต่อระบบการตัดสินใจหลีกเลี่ยงการชน (Collision Avoidance System) โดยเฉพาะในบริบทของหุ่นยนต์บริการหรือหุ่นยนต์ขับเคลื่อนอัตโนมัติที่ทำงานในพื้นที่จำกัด

การคำนวณระยะห่างดำเนินการภายใต้ระบบพิกัดคาร์ทีเซียน (Cartesian Coordinate System) โดยใช้สมการระยะทางระหว่างจุดสองจุดในระนาบสองมิติ ดังแสดงใน รูปที่ 3.28:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

โดยที่

- (x_1,y_1) คือ ตำแหน่งเริ่มต้นหรือจุดอ้างอิงของเซนเซอร์ LiDAR ซึ่งถือเป็นจุดอ้างอิง (0,0) บนหุ่นยนต์
- (x_2,y_2) คือ ตำแหน่งของจุดศูนย์กลางของ Bounding box ที่ถูกสร้างจากวัตถุที่ตรวจจับได้โดยโมเดล

รูปที่ 3.28 สมการหาระยะห่างระหว่างจุดสองจุดในระบบพิกัดคาร์ทีเซียน

การวิเคราะห์นี้สามารถนำไปใช้ในระบบการตัดสินใจแบบอัตโนมัติ เพื่อกำหนดว่าจะหยุดหุ่นยนต์ เมื่อวัตถุที่อยู่ด้านหน้ามีแนวโน้มที่เสี่ยงต่อการชน ระบบจะทำการคำนวณระยะห่าง d ของวัตถุแต่ละประเภทที่อยู่ในภาพรวม แล้วเปรียบเทียบกับเกณฑ์ระยะปลอดภัยที่กำหนดไว้ล่วงหน้า เช่น หากระยะที่คำนวณได้ต่ำกว่า 2.0 เมตร หุ่นยนต์จะถูกสั่งให้หยุดเพื่อป้องกันการชน หากอยู่ระหว่าง 1.0–2.5 เมตร

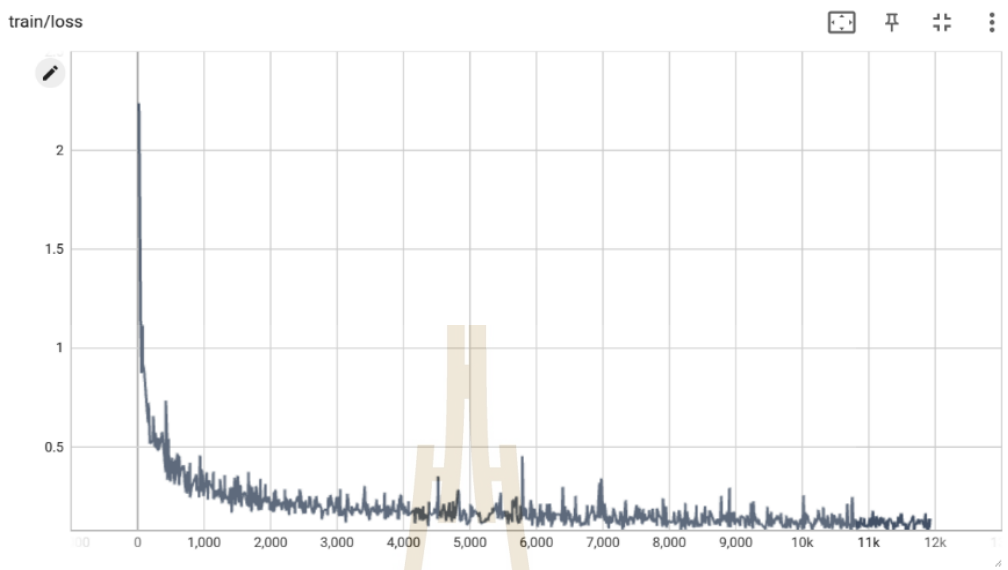
การประยุกต์ใช้สมการนี้ร่วมกับผลลัพธ์ของโมเดลตรวจจับวัตถุ 3 มิติ จึงเป็นกลไกสำคัญที่ช่วยให้หุ่นยนต์สามารถรับรู้สภาพแวดล้อมและตัดสินใจได้อย่างมีประสิทธิภาพ โดยเฉพาะในกรณีของหุ่นยนต์ที่ทำงานในพื้นที่ที่มีคนเดินผ่าน เช่น หุ่นยนต์ส่งของอัตโนมัติที่ต้องหยุดเมื่อมีบุคคลเดินเข้ามาขวางทางเดินในระยะที่กำหนด

บทที่ 4
ผลการวิจัยและอภิปรายผล

4.1 ผลการทดสอบระบบตรวจจับวัตถุ 3 มิติ

ในการประเมินประสิทธิภาพของระบบตรวจจับวัตถุสามมิติในงานวิจัยนี้ ได้ดำเนินการฝึกสอนโมเดลด้วยสถาปัตยกรรม PointPillars ที่ได้รับการปรับปรุง โดยมีการปรับค่าพารามิเตอร์ของข้อมูลพอยต์คลาวด์ให้เหมาะสมกับสภาพแวดล้อมการทำงานจริงของหุ่นยนต์อัตโนมัติ โดยกำหนดขอบเขตของข้อมูลพอยต์คลาวด์ (Point Cloud Range) เป็นช่วง [0, -3, -1] ถึง [7, 3, 3] เมตร ซึ่งเน้นพื้นที่ตรวจจับเฉพาะบริเวณด้านหน้าของหุ่นยนต์ และปรับขนาดของวอกเซล (Voxel Size) ให้ละเอียดเป็น [0.0625, 0.0625, 4] เมตร เพื่อเพิ่มความสามารถในการตรวจจับวัตถุขนาดเล็กในระยะใกล้ได้อย่างแม่นยำยิ่งขึ้น

นอกจากนี้ ในส่วนของโครงสร้าง Backbone ได้มีการแทนที่โครงข่ายคอนโวลูชันแบบเดิมด้วย Lightweight Attention Pyramid Network (LAPN) ซึ่งมีความสามารถในการสกัดคุณลักษณะเชิงลึกจากหลายระดับความละเอียด (Multi-scale Feature Extraction) ร่วมกับการฝังหน่วยความสนใจ (Attention Mechanism) ที่ช่วยเสริมการเรียนรู้ในเชิงบริบท โดยเน้นเฉพาะช่องสัญญาณ (Feature Channels) ที่มีความสำคัญต่อการตรวจจับวัตถุ โดยการประเมินผลได้ใช้เกณฑ์หลายประการเพื่อวัดความแม่นยำและประสิทธิภาพของโมเดลที่สร้างขึ้น ดังนั้นในการอภิปรายผลนี้จะเน้นไปที่การวิเคราะห์ตัวชี้วัด หลักต่าง ๆ ที่สะท้อนถึงประสิทธิภาพของโมเดลและการเรียนรู้ของโมเดลในการฝึกตรวจจับวัตถุในภาพ



รูปที่ 4.1 กราฟจากผลลัพธ์การฝึกของโมเดล

จากรูปที่ 4.1 จะเห็นว่าในช่วงเริ่มต้นการฝึก โมเดลยังไม่ได้รับการปรับแต่งและมีค่า Loss (train/loss) สูงขึ้น ซึ่งหมายถึงโมเดลยังไม่สามารถตรวจจับ Bounding Box ของวัตถุได้อย่างแม่นยำ เมื่อจำนวน epoch เพิ่มขึ้น ความการสูญเสีย (Loss) ของโมเดลลดลงอย่างสม่ำเสมอ แสดงถึงการที่โมเดลเรียนรู้จากข้อมูล การฝึกอย่างมีประสิทธิภาพและสามารถปรับปรุงการทำนายตำแหน่งของวัตถุได้มากขึ้น ในขั้นตอนการฝึก ช่วงท้าย ๆ ค่า Loss ลดลงจนเข้าสู่ระดับที่ต่ำมาก ซึ่งเป็นสัญญาณว่าความแม่นยำของการตรวจจับ Bounding Box มีแนวโน้มที่ดีขึ้นเรื่อย ๆ

แบ่งตัวชี้วัดที่ใช้ในการวัดประสิทธิภาพการตรวจจับได้แก่ confusion matrix, mAP และ prediction time

ตารางที่ 4.1 แสดงค่า loss และ parameter จากการฝึกสอนโมเดล

Model	Train loss	Parameter
Default model	0.01647	1402984
Enhancing range	0.02472	1211688
Enhancing range and attention	0.06286	453672

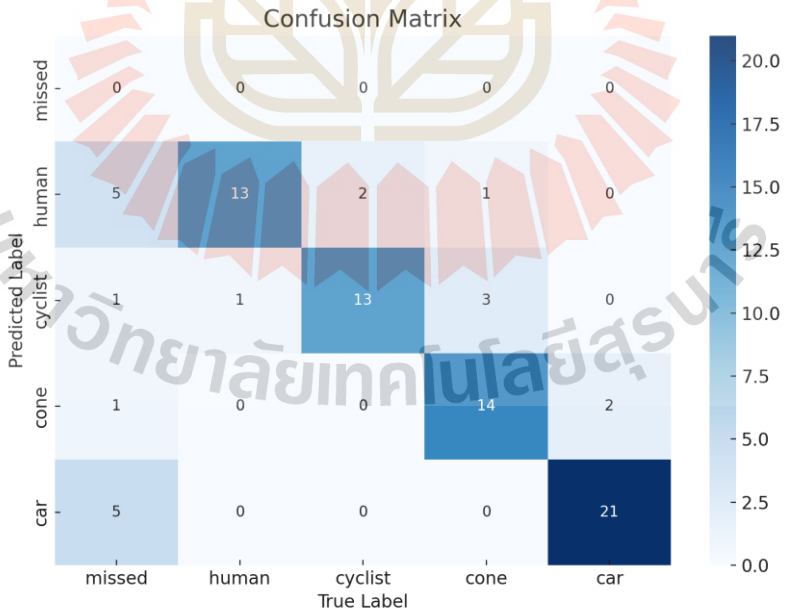
จากในตารางที่ 4.1 แสดงค่า train loss และจำนวน parameter ของโมเดลทั้งสามรูปแบบ ได้แก่ Default model, Enhancing range และ Enhancing range and attention ผลการทดลอง

แสดงให้เห็นว่าโมเดลดั้งเดิมมีค่า train loss ต่ำที่สุดที่ 0.01647 และมีจำนวนพารามิเตอร์มากที่สุดที่ 1,402,984 พารามิเตอร์ แสดงถึงความสามารถในการเรียนรู้ที่ดีแต่ต้องใช้ทรัพยากรในการประมวลผลสูง

สำหรับโมเดลที่ปรับช่วงการประมวลผล (Enhancing range) พบว่าจำนวนพารามิเตอร์ลดลงเหลือ 1,211,688 หรือคิดเป็นประมาณ 13.6% เมื่อเทียบกับโมเดลดั้งเดิม ในขณะที่ค่า train loss เพิ่มขึ้นเล็กน้อยเป็น 0.02472 แสดงถึงการแลกเปลี่ยนระหว่างการลดความซับซ้อนของโมเดลและความแม่นยำในการเรียนรู้

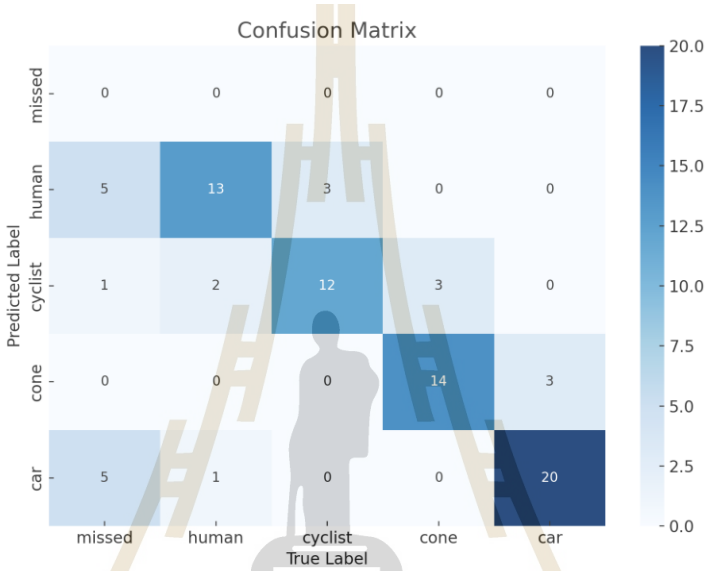
ส่วนโมเดลที่ปรับทั้งช่วงการประมวลผลและเพิ่มโครงสร้าง *attention network* มีจำนวนพารามิเตอร์ลดลงอย่างมีนัยสำคัญเหลือเพียง 453,672 หรือคิดเป็นการลดลงประมาณ 67% เมื่อเทียบกับโมเดลดั้งเดิม แต่ค่า train loss สูงขึ้นเป็น 0.06286 ซึ่งสะท้อนให้เห็นว่าโมเดลนี้มีความซับซ้อนลดลงมากและใช้ทรัพยากรน้อยลง เหมาะสมต่อการนำไปใช้งานจริงบนอุปกรณ์ที่มีข้อจำกัดด้านการประมวลผล เช่น Jetson Xavier AGX แม้ว่าจะต้องแลกมากับประสิทธิภาพในการเรียนรู้ที่ลดลง

Confusion Matrix ใช้เพื่อวิเคราะห์ความถูกต้องในการจำแนกประเภทของโมเดล โดยแสดงจำนวนผลการทำนายที่ถูกต้องและผิดพลาดในแต่ละคลาส ซึ่งช่วยให้เห็นข้อดีและข้อจำกัดของโมเดลได้ชัดเจน



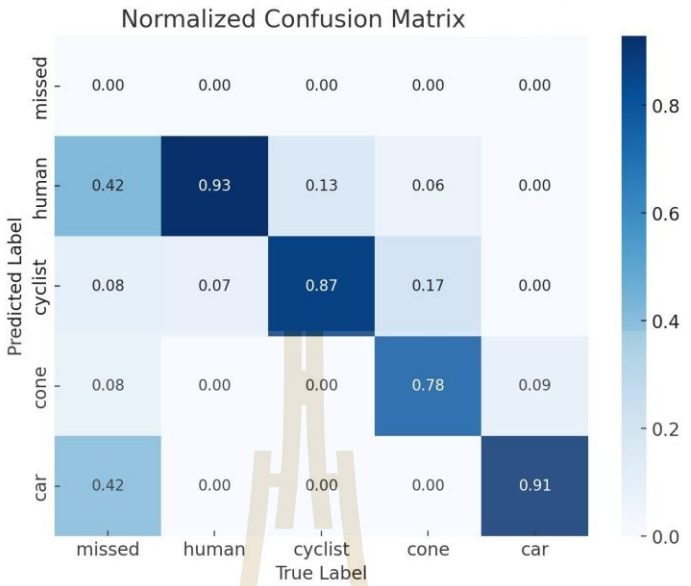
รูปที่ 4.2 Confusion Matrix Unnormalized แสดงผลการทดสอบของโมเดลดั้งเดิม

จากการทดสอบโมเดลดั้งเดิม แสดงผลลัพธ์ใน รูปที่ 4.2 Confusion Matrix Unnormalized พบว่าโมเดลสามารถจำแนกวัตถุในคลาสต่าง ๆ ได้ในระดับที่น่าพอใจ โดยเฉพาะคลาส car ที่ถูกจำแนกถูกต้องจำนวน 21 ตัวอย่าง และคลาส cone ถูกจำแนกถูกต้อง 14 ตัวอย่าง อย่างไรก็ตามยังคงมีการจำแนกผิดอยู่บ้าง เช่น คลาส human ถูกทำนายผิดเป็น missed หรือไม่พบวัตถุ จำนวน 5 ตัวอย่าง และ cyclist ถูกจำแนกสับสนกับคลาสอื่น 4 ตัวอย่าง

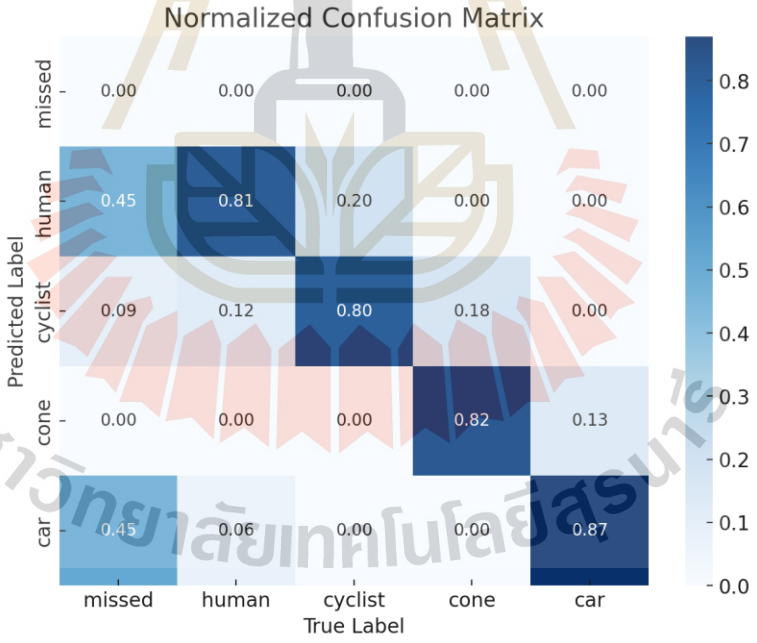


รูปที่ 4.3 Confusion Matrix Unnormalized แสดงผลการทดสอบของโมเดลที่ปรับปรุงแล้ว

เมื่อปรับปรุงโมเดลแล้ว ผลลัพธ์แสดงใน รูปที่ 4.3 Confusion Matrix Unnormalized พบว่าความสามารถในการจำแนกวัตถุบางคลาสมีการเปลี่ยนแปลง โดยคลาส car ยังคงถูกจำแนกถูกต้องใกล้เคียงกับโมเดลดั้งเดิม แต่มีการลดลงเล็กน้อยจาก 21 ตัวอย่าง เหลือ 20 ตัวอย่าง ในขณะที่คลาส cone ยังคงมีการจำแนกถูกต้องเท่ากับ 14 ตัวอย่าง ส่วนคลาส human และ cyclist ยังมีการสับสนกับคลาสใกล้เคียงอยู่



รูปที่ 4.4 Confusion Matrix Normalized แสดงผลการทดสอบของโมเดลดั้งเดิม

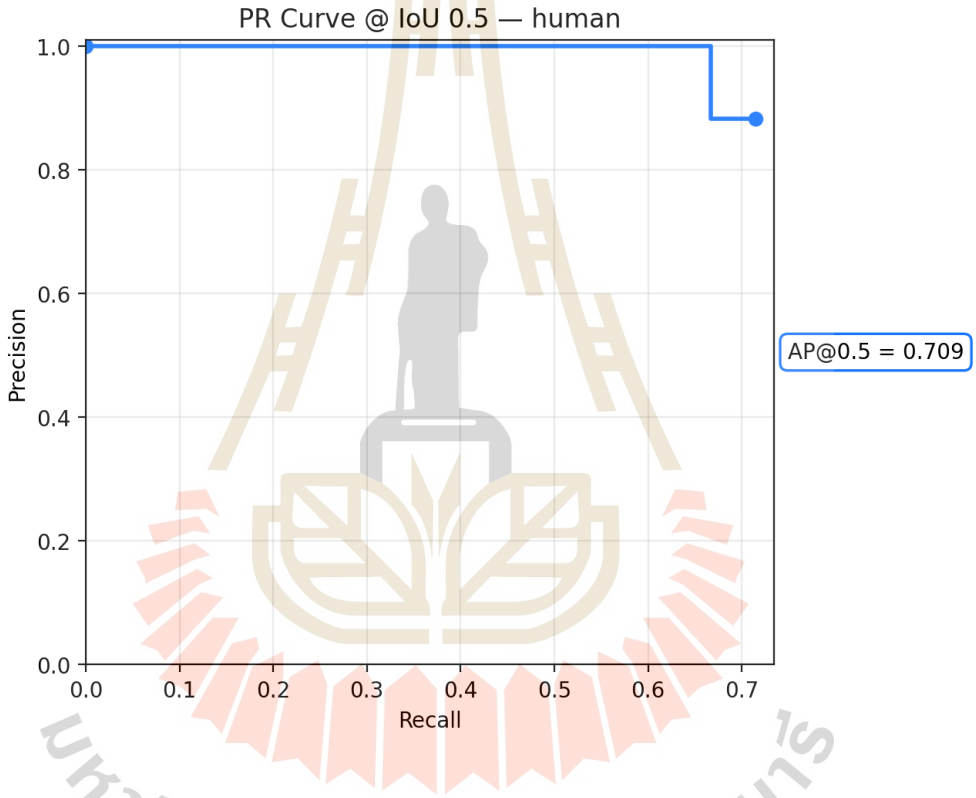


รูปที่ 4.5 Confusion Matrix Normalized แสดงผลการทดสอบโมเดลที่ปรับระยะการตรวจจับแล้ว

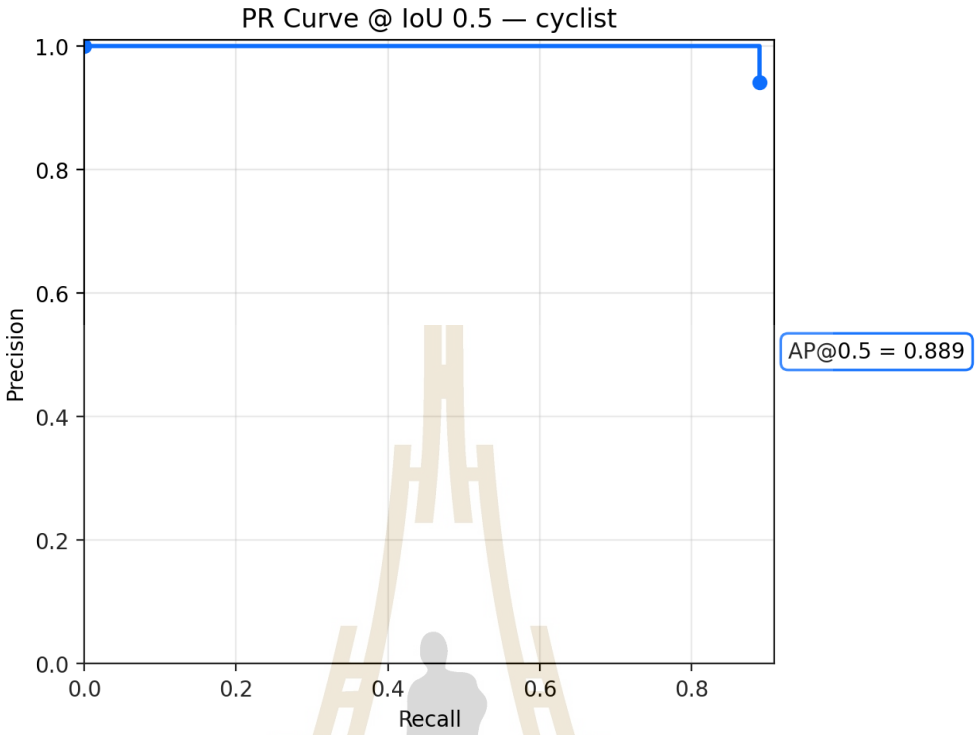
หากพิจารณาในรูปแบบ Normalized Confusion Matrix ดังแสดงใน รูปที่ 4.4 และ รูปที่ 4.5 จะเห็นได้ชัดเจนยิ่งขึ้นว่า โมเดลดั้งเดิมมีความแม่นยำในการจำแนกคลาส human สูง

ถึง 93% และ car อยู่ที่ 91% แต่คลาส cone ยังมีการทำนายผิดประมาณ 22% ขณะที่ผลการปรับปรุงโมเดล (รูปที่ 4.5) พบว่าความแม่นยำของคลาส human ลดลงจาก 93% เหลือ 81% และคลาส car ลดลงเล็กน้อยเหลือ 87% แต่คลาส cone มีความแม่นยำเพิ่มขึ้นเป็น 82% และ cyclist ยังคงรักษาระดับได้ใกล้เคียงเดิม

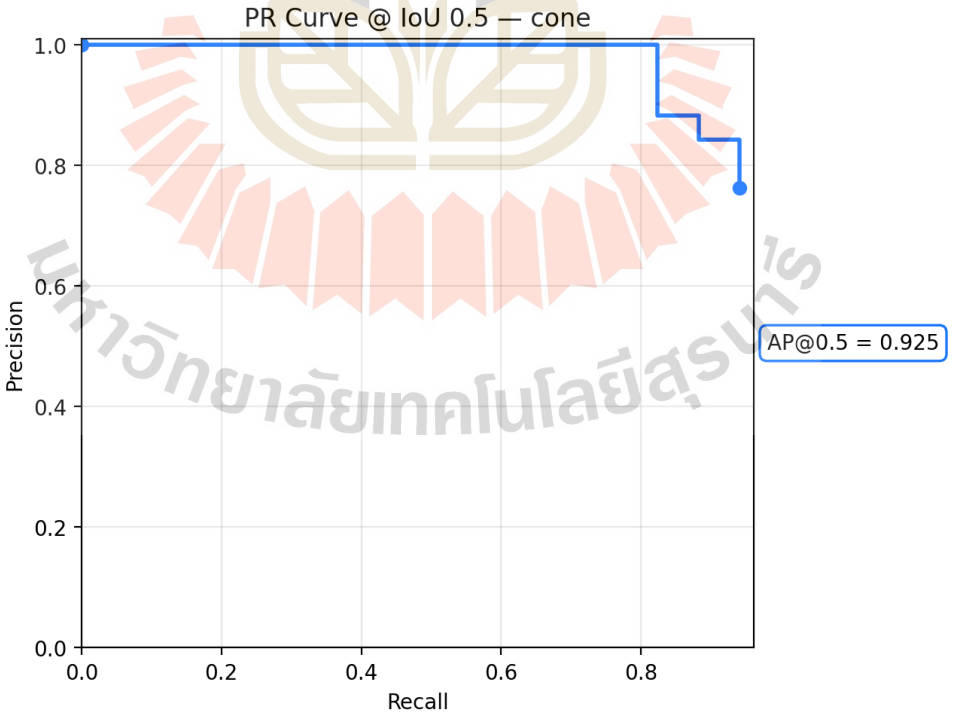
โดยสรุป โมเดลที่ปรับปรุงแล้วสามารถลดภาระการประมวลผลและทำงานได้เร็วขึ้น แต่ความแม่นยำในการจำแนกวัตถุบางคลาสมีการเปลี่ยนแปลงไปในลักษณะ trade-off กล่าวคือบางคลาสมีประสิทธิภาพลดลง ขณะที่บางคลาสมีประสิทธิภาพดี



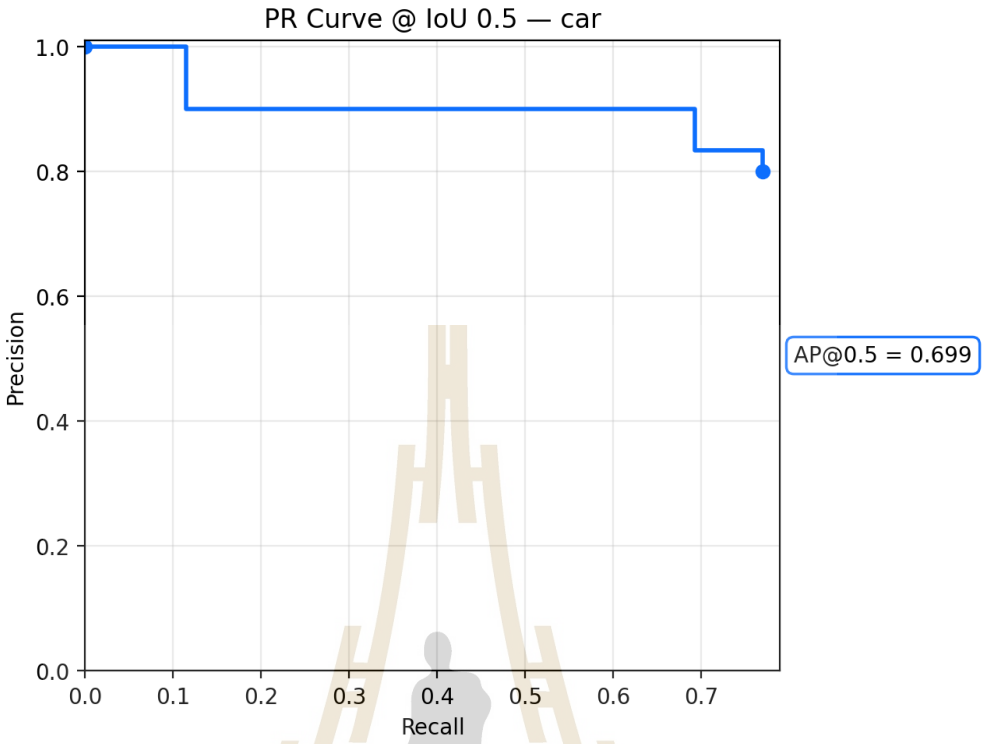
รูปที่ 4.6 กราฟ Precision-Recall Curve ของคลาสคนจากโมเดลดั้งเดิม



รูปที่ 4.7 กราฟ Precision-Recall Curve ของคลาสคนขี่จักรยานจากโมเดลดั้งเดิม

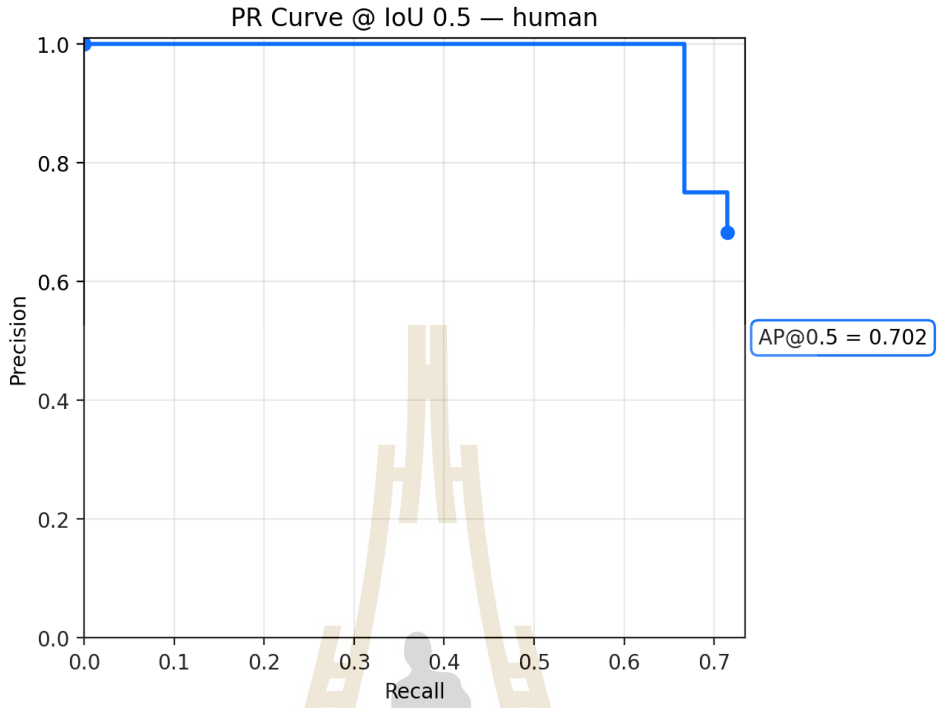


รูปที่ 4.8 กราฟ Precision-Recall Curve ของคลาสกรวยจากโมเดลดั้งเดิม

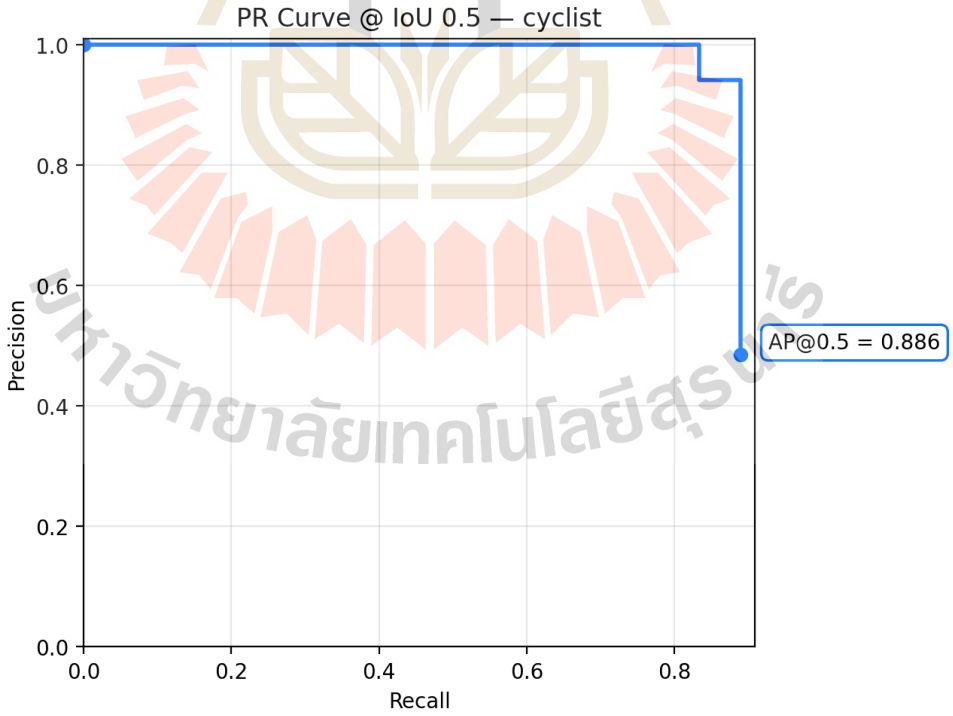


รูปที่ 4.9 กราฟ Precision-Recall Curve ของคลาสรถยนต์จากโมเดลดั้งเดิม

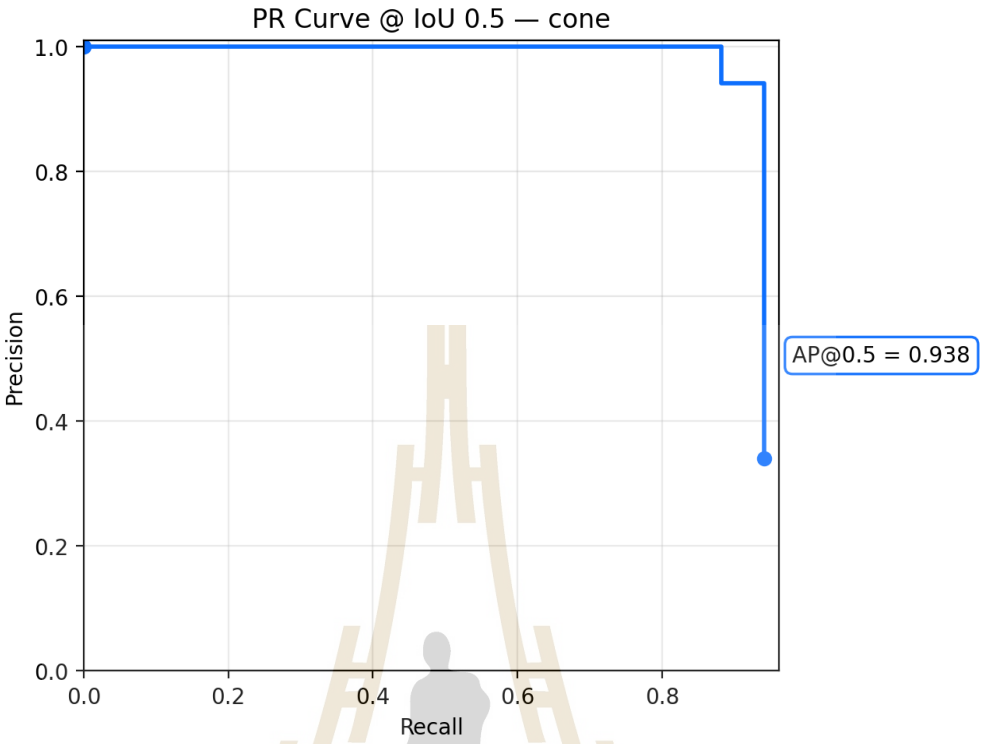
รูปที่ 4.6–4.9 แสดงกราฟ Precision–Recall Curve (PR Curve) ของโมเดลดั้งเดิมในแต่ละคลาส ซึ่งใช้ในการประเมินสมรรถนะการตรวจจับวัตถุ โดยจะเห็นได้ว่าคลาสคน (human) มีค่า AP@0.5 เท่ากับ 0.709 แสดงถึงความสามารถในการตรวจจับที่ค่อนข้างดี ส่วนคลาสคนขี่จักรยาน (cyclist) มีค่า AP@0.5 สูงที่สุดคือ 0.889 แสดงให้เห็นว่าโมเดลสามารถจำแนกและตรวจจับวัตถุประเภทนี้ได้อย่างแม่นยำมาก ในขณะที่คลาสกรวย (cone) ก็มีผลลัพธ์ที่ดีเช่นกัน โดยมีค่า AP@0.5 เท่ากับ 0.925 ซึ่งเป็นค่าที่สูงที่สุด แสดงถึงความแม่นยำและความสมดุลของ precision และ recall ที่ดี ส่วนคลาสรถยนต์ (car) มีค่า AP@0.5 เท่ากับ 0.699 ซึ่งแม้จะอยู่ในระดับที่ยอมรับได้ แต่ยังต่ำกว่าคลาสอื่น แสดงให้เห็นถึงข้อจำกัดของโมเดลในการตรวจจับรถยนต์ สรุปได้ว่า โมเดลดั้งเดิมมีประสิทธิภาพที่แตกต่างกันไปในแต่ละคลาส โดยทำงานได้ดีมากกับคลาส cone และ cyclist ในขณะที่การตรวจจับคลาส human และ car ยังสามารถปรับปรุงได้เพิ่มเติม



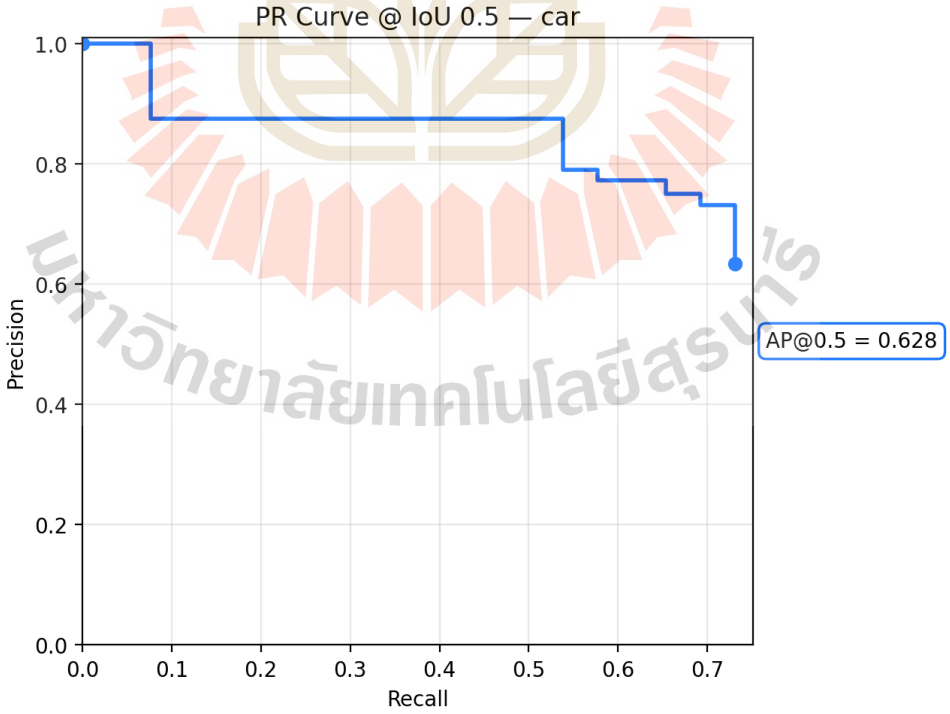
รูปที่ 4.10 กราฟ Precision-Recall Curve ของคลาสคนจากโมเดลที่ปรับปรุงแล้ว



รูปที่ 4.11 กราฟ Precision-Recall Curve ของคลาสคนขี่จักรยานจากโมเดลที่ปรับปรุงแล้ว



รูปที่ 4.12 กราฟ Precision-Recall Curve ของคลาสกรวยจากโมเดลที่ปรับปรุงแล้ว



รูปที่ 4.13 กราฟ Precision-Recall Curve ของคลาสรถยนต์จากโมเดลที่ปรับปรุงแล้ว

จากรูปที่ 4.10–4.13 แสดงกราฟ Precision-Recall Curve ของโมเดลที่ได้รับการปรับปรุง พบว่าโมเดลสามารถรักษาความแม่นยำได้ค่อนข้างสูงในคลาส cyclist (AP@0.5 = 0.886) และ cone (AP@0.938) ซึ่งมีค่าใกล้เคียงหรือสูงกว่าโมเดลดั้งเดิมเล็กน้อย ส่วนคลาส human (AP@0.702) และ car (AP@0.628) มีค่าลดลงเล็กน้อยเมื่อเทียบกับโมเดลดั้งเดิม อย่างไรก็ตาม กราฟ PR Curve ของทั้งสี่คลาสยังคงมีลักษณะใกล้เส้นขอบด้านบน แสดงให้เห็นว่าโมเดลมีความสามารถในการรักษาค่าความแม่นยำที่สูงแม้เมื่อเพิ่มค่า recall ซึ่งสะท้อนถึงศักยภาพของโมเดลที่ปรับปรุงแล้วในการนำไปใช้งานจริง โดยมีการแลกเปลี่ยนระหว่างความแม่นยำที่ลดลงในบางคลาสกับประสิทธิภาพการประมวลผลที่รวดเร็วขึ้นตามเป้าหมายของการปรับปรุงสถาปัตยกรรม

เมื่อเปรียบเทียบผลลัพธ์ของโมเดลดั้งเดิม (รูปที่ 4.6–4.9) กับโมเดลที่ปรับปรุงแล้ว (รูปที่ 4.10–4.13) พบว่าโมเดลที่ปรับปรุงสามารถรักษาประสิทธิภาพการตรวจจับได้ใกล้เคียงในหลายคลาส โดยเฉพาะคลาส cyclist และ cone ที่มีค่า AP@0.5 สูงใกล้เคียงหรือดีกว่าโมเดลดั้งเดิม ขณะที่คลาส human และ car มีค่าความแม่นยำลดลงเล็กน้อย สะท้อนให้เห็นถึงการแลกเปลี่ยน (trade-off) ระหว่างความแม่นยำกับความเร็วในการประมวลผล ซึ่งสอดคล้องกับวัตถุประสงค์ของการวิจัยที่ต้องการลดภาระการคำนวณของโมเดลเพื่อนำไปใช้งานจริงบน Jetson Xavier การเปรียบเทียบนี้ชี้ให้เห็นว่าแม้ความแม่นยำบางส่วนจะลดลง แต่โมเดลที่ปรับปรุงแล้วก็ยังคงมีศักยภาพเพียงพอสำหรับการใช้งานในระบบหุ่นยนต์เคลื่อนที่อัตโนมัติที่ต้องการการตอบสนองแบบเรียลไทม์

ตารางที่ 4.2 แสดงค่า AP ของโมเดลตรวจจับวัตถุของโมเดลดั้งเดิม

Class name	AP (%) at 0.5 IOU
human	70.87
cyclist	88.89
cone	92.50
car	69.87
All class mean AP	80.53

ตารางที่ 4.3 แสดงค่า AP ของโมเดลตรวจจับที่ปรับปรุงแล้ว

Class name	AP (%) at 0.5 IOU
human	70.24
cyclist	88.56
cone	93.77
car	62.75
All class mean AP	78.83

จากกราฟ Precision–Recall Curve ของแต่ละคลาส (รูปที่ 4.6–4.13) ได้ทำการคำนวณค่า Average Precision (AP) ที่ค่า IoU = 0.5 เพื่อสรุปประสิทธิภาพการตรวจจับของโมเดล โดยแสดงผลในตารางที่ 4.2 และ 4.3 สำหรับโมเดลดั้งเดิมและโมเดลที่ปรับปรุงแล้วตามลำดับ ผลการประเมินพบว่าโมเดลดั้งเดิมมีค่าเฉลี่ย mAP เท่ากับ 80.53% ขณะที่โมเดลที่ปรับปรุงแล้วมีค่าเฉลี่ย mAP เท่ากับ 78.83% โดยยังคงมีค่า AP ของแต่ละคลาสอยู่ในระดับสูงกว่า 60% ทั้งหมด ซึ่งสะท้อนถึงความสามารถในการตรวจจับที่น่าเชื่อถือ เมื่อพิจารณารายคลาส พบว่าค่า AP ของ cone และ cyclist ยังคงสูงเกิน 88% ในทั้งสองโมเดล ในขณะที่ human และ car มีค่าลดลงเล็กน้อยในโมเดลที่ปรับปรุงแล้ว การเปรียบเทียบนี้แสดงให้เห็นว่าแม้การปรับโมเดลจะมีผลให้ค่าเฉลี่ย mAP ลดลงเล็กน้อย แต่ก็ยังคงอยู่ในระดับที่เหมาะสมสำหรับการใช้งานจริง โดยแลกมากับความเร็วในการประมวลผลที่สูงขึ้นตามวัตถุประสงค์ของการวิจัย

ตารางที่ 4.4 แสดงค่า Precison ,Recall และ F1score ของโมเดลตรวจจับวัตถุของโมเดลดั้งเดิม

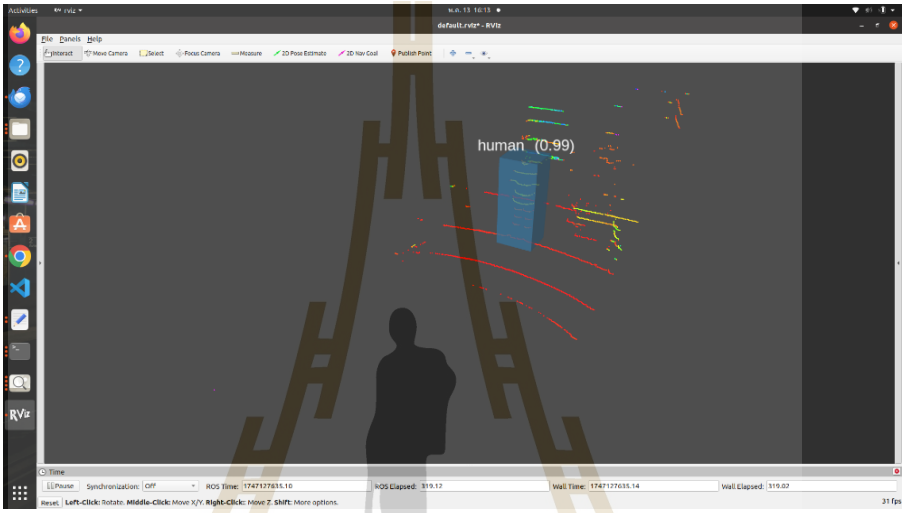
Class name	Precision	Recall	F1score
human	92.86	61.90	74.29
cyclist	86.67	72.22	78.79
cone	77.78	82.35	80.00
car	91.30	80.77	85.71

ตารางที่ 4.5 แสดงค่า Precison ,Recall และ F1score ของโมเดลตรวจจับที่ปรับปรุงแล้ว

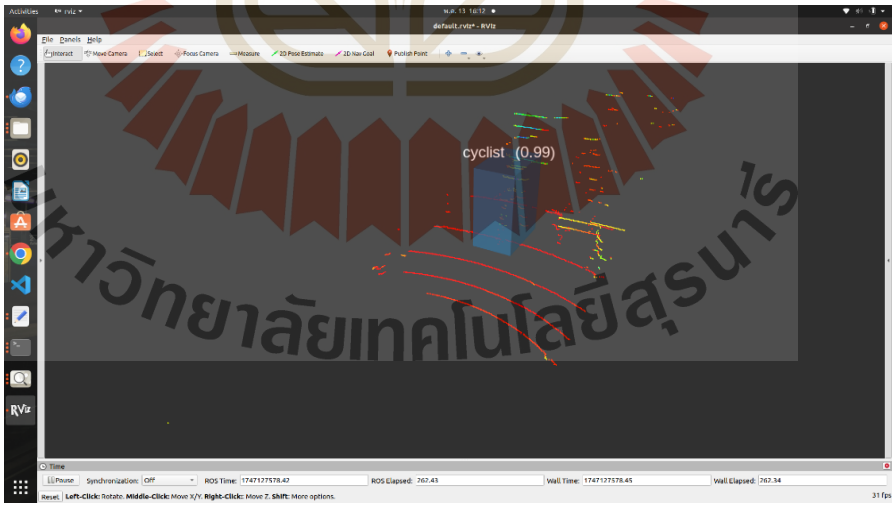
Class name	Precision (%)	Recall (%)	F1-score (%)
human	81.25	61.90	70.27
cyclist	80.00	66.67	72.73
cone	82.35	82.35	82.35
car	86.96	76.92	81.63

จากตารางที่ 4.4 และ 4.5 แสดงค่า Precision, Recall และ F1-score ของโมเดลตรวจจับวัตถุทั้งดั้งเดิมและที่ปรับปรุงแล้ว พบว่าโมเดลดั้งเดิม (ตารางที่ 4.4) มีค่า Precision ค่อนข้างสูง โดยเฉพาะคลาส human และ car ที่มีค่าเกิน 90% แต่ค่าการครอบคลุม (Recall) ของคลาส human ยังค่อนข้างต่ำ ส่งผลให้ค่า F1-score อยู่เพียง 74.29% ในขณะที่คลาส cone และ car มี F1-score สูงกว่า 80% แสดงถึงความสมดุลที่ดีของการตรวจจับ สำหรับโมเดลที่ปรับปรุงแล้ว (ตารางที่ 4.5) พบว่าค่า Precision ลดลงในทุกคลาสเมื่อเทียบกับโมเดลดั้งเดิม แต่ Recall ยังคงรักษาระดับ

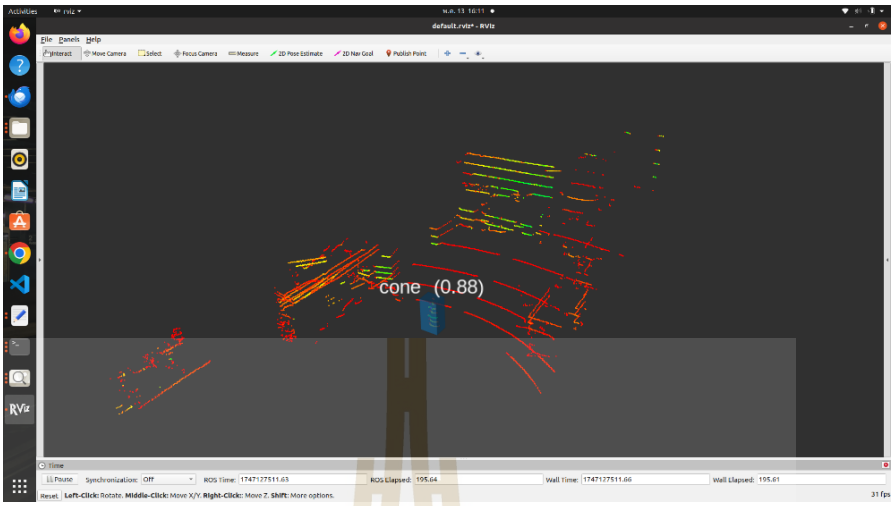
ใกล้เคียงหรือดีกว่าในบางคลาส โดยเฉพาะ cone ที่ยังคงมี Recall สูงสุดที่ 82.35% ส่งผลให้ค่า F1-score มีค่ามากกว่า 80% เช่นเดียวกัน อย่างไรก็ตาม F1-score ของคลาส human และ cyclist ลดลงจากโมเดลดั้งเดิมเล็กน้อย สะท้อนให้เห็นว่าโมเดลที่ปรับปรุงแล้วมีความสมดุลระหว่าง Precision และ Recall มากขึ้น แม้ค่าเฉลี่ยโดยรวมจะลดลงบ้าง แต่ยังคงเพียงพอต่อการนำไปใช้งานในสภาพแวดล้อมจริงได้



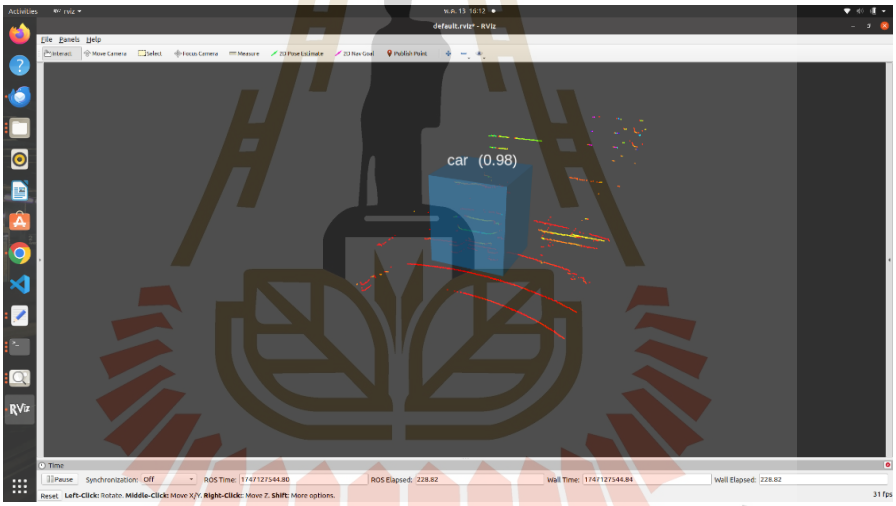
รูปที่ 4.14 ผลการตรวจจับวัตถุจากพอยคลาวด์(1)



รูปที่ 4.15 ผลการตรวจจับวัตถุจากพอยคลาวด์(2)



รูปที่ 4.16 ผลการตรวจจับวัตถุจากพอยคลาวด์(3)



รูปที่ 4.17 ผลการตรวจจับวัตถุจากพอยคลาวด์(4)

ขั้นตอนสุดท้ายได้นำระบบตรวจจับวัตถุจากพอยคลาวด์ Pointpillar ที่ปรับปรุงแล้วที่ผ่านกระบวนการเรียนรู้และ ทดสอบเรียบร้อยแล้ว มาทดลองใช้งานกับสภาพแวดล้อมจริง ซึ่งสามารถตรวจจับวัตถุต่าง ๆ ที่อยู่ในพื้นที่ทดสอบได้ดี ดังตัวอย่างของผลการตรวจจับในภาพที่ 4.14, 4.15, 4.16 และ 4.17

ตารางที่ 4.6 สรุปคุณลักษณะหลักของ Jetson Xavier AGX และคอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ

รายการ	**Jetson Xavier AGX**	**คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (Onboard Computer)**
หน่วยประมวลผลกลาง (CPU)	8-Core ARM v8.2 64-bit	Intel® Core™ i7-11700K (8 คอร์ 16 เธรด)
หน่วยประมวลผลกราฟิก (GPU)	NVIDIA Volta GPU พร้อม Tensor Cores	NVIDIA GeForce RTX 3080 Ti (Ampere Architecture)
หน่วยความจำ (Memory)	32 GB LPDDR4x	32 GB DDR4

หลังจากทำการเปรียบเทียบคุณลักษณะทางเทคนิคของหน่วยประมวลผลทั้งสองระบบดังแสดงในตารางที่ 4.6 จะเห็นได้ว่า Jetson Xavier AGX ถูกออกแบบมาเพื่อการประมวลผลแบบ Edge AI ที่ให้ประสิทธิภาพสูงภายใต้การใช้พลังงานต่ำ เหมาะสำหรับการติดตั้งบนอุปกรณ์เคลื่อนที่ เช่น หุ่นยนต์หรือยานยนต์อัตโนมัติ ในขณะที่ คอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (วงศธร ,2565) มีประสิทธิภาพการประมวลผลสูงกว่า เนื่องจากใช้หน่วยประมวลผลกลาง (CPU) แบบ Intel Core i7 และหน่วยประมวลผลกราฟิก (GPU) ระดับสูงอย่าง NVIDIA GeForce RTX 3080 Ti ซึ่งเหมาะสำหรับงานประมวลผลเชิงลึกและโมเดลปัญญาประดิษฐ์ขนาดใหญ่

เพื่อประเมินผลกระทบของสมรรถนะฮาร์ดแวร์ต่อประสิทธิภาพของโมเดลปัญญาประดิษฐ์ จึงได้ทำการทดลองรันโมเดลเดียวกันบนทั้งสองแพลตฟอร์ม โดยวัดค่า อัตราเฟรมต่อวินาที (Frames Per Second: FPS) และ เวลาในการทำนาย (Prediction Time) ขณะรันโมเดลเพียงอย่างเดียว ผลการทดสอบแสดงในตารางที่ 4.7

ตารางที่ 4.7 แสดงค่า FPS และเวลาที่โมเดลใช้ในการทำนาย ขณะรันโมเดลอย่างเดียว

Precessor Unit	Model	FPS	Prediction time (second)	Reduce Prediction time(%)
Jetson Xavier	Default Model	12.05	0.083	-
	Enhancing range	30.17	0.033	60.24
	Enhancing range and attention network	38.72	0.026	68.67
Onboard Computer	Default Model	87.11	0.011	-
	Enhancing range	96.99	0.01	9.09
	Enhancing range and attention network	116.89	0.009	18.18

จากตารางที่ 4.7 แสดงผลการประเมินความเร็วในการทำงานของโมเดลในขั้นตอนการทำนาย โดยพิจารณาจากค่า FPS และเวลาเฉลี่ยในการทำนายต่อเฟรม ภายใต้การทดสอบเฉพาะโมเดลโดยไม่รวมระบบขับเคลื่อนอัตโนมัติ พบว่า โมเดลดั้งเดิม (Default Model) ที่ทำงานบนหน่วยประมวลผล Jetson Xavier มีค่า FPS เท่ากับ 12.05 และใช้เวลาในการทำนายเฉลี่ย 0.083 วินาที ซึ่งเป็นค่าที่ต่ำเมื่อเทียบกับโมเดลที่ได้รับการปรับปรุง

สำหรับ โมเดล Enhancing range ที่เพิ่มขอบเขตการรับข้อมูล พบว่าค่า FPS เพิ่มขึ้นเป็น 30.17 และใช้เวลาในการทำนายลดลงเหลือ 0.033 วินาที ลดเวลาได้ถึง 60.24% เมื่อเทียบกับโมเดลดั้งเดิม ขณะที่ โมเดล Enhancing range and attention network ให้ประสิทธิภาพสูงสุด โดยมีค่า FPS 38.72 และใช้เวลาในการทำนายเฉลี่ยเพียง 0.026 วินาที ลดเวลาได้ถึง 68.67% แสดงให้เห็นถึงประสิทธิภาพของกลไก attention ที่ช่วยให้โมเดลมุ่งเน้นการประมวลผลเฉพาะบริเวณที่สำคัญได้ดียิ่งขึ้น

ในส่วนของ คอมพิวเตอร์บนรถกอล์ฟ (Onboard Computer) (วงศธร,2565) ซึ่งมีประสิทธิภาพด้านการประมวลผลสูงกว่า พบว่าค่า FPS สูงกว่า Jetson Xavier อย่างชัดเจน โดยโมเดลดั้งเดิมมีค่า FPS 87.11 และใช้เวลาในการทำนายเพียง 0.011 วินาที สำหรับโมเดล Enhancing range มีค่า FPS เพิ่มขึ้นเป็น 96.99 และลดเวลาในการทำนายเหลือ 0.010 วินาที (ลดลง 9.09%) ส่วนโมเดล Enhancing range and attention network มีค่า FPS สูงสุดที่ 116.89 และใช้เวลาในการทำนายเพียง 0.009 วินาที (ลดลง 18.18%)

จากผลการทดลองสามารถสรุปได้ว่า การปรับปรุงโครงสร้างของโมเดลด้วยการเพิ่มช่วงการรับข้อมูลและการใช้ attention network ช่วยเพิ่มประสิทธิภาพเชิงเวลาได้อย่างมีนัยสำคัญ ทั้งในแพลตฟอร์ม Jetson Xavier และ Onboard Computer โดยเฉพาะอย่างยิ่งเมื่อใช้งานบนคอมพิวเตอร์ที่มีสมรรถนะสูง ผลลัพธ์ดังกล่าวสะท้อนให้เห็นถึงศักยภาพของโมเดลที่ได้รับการปรับปรุงในการทำงานแบบเรียลไทม์ได้ดียิ่งขึ้น

ตารางที่ 4.8 แสดงค่า FPS และเวลาที่โมเดลใช้ในการทำนาย พร้อมระบบขับเคลื่อนอัตโนมัติเต็มระบบ

Precessor Unit	Model	FPS	Prediction time (second)	Reduce Prediction time(%)
Jetson Xavier	Default Model	9.05	0.111	-
	Enhancing range	21.95	0.046	58.56
	Enhancing range and attention network	32.35	0.031	72.07

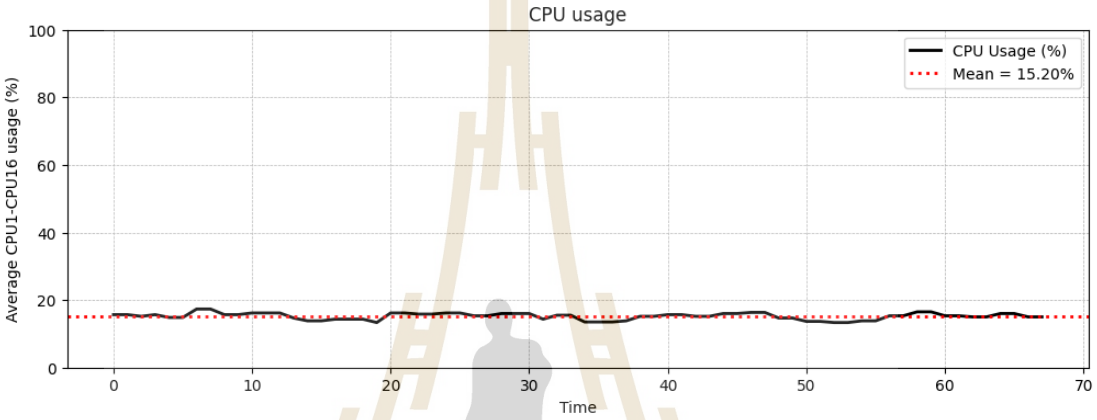
จากตารางที่ 4.8 แสดงผลการประเมินความเร็วในการทำงานของโมเดลในขั้นตอนการทำนาย โดยพิจารณาจากค่า FPS และเวลาเฉลี่ยในการทำนายต่อเฟรมภายใต้การทำงานร่วมกับระบบอัตโนมัติเต็มรูปแบบ พบว่าโมเดลดั้งเดิมมีค่า FPS เพียง 9.05 และใช้เวลาในการทำนาย 0.111 วินาที ซึ่งเป็นค่าที่ต่ำเมื่อเทียบกับโมเดลที่ได้รับการปรับปรุง สำหรับโมเดล Enhancing range มีค่า FPS เพิ่มขึ้นเป็น 21.95 และลดเวลาในการทำนายลงเหลือ 0.046 วินาที สามารถลดเวลาได้ 58.56% เมื่อเทียบกับโมเดลดั้งเดิม ขณะที่โมเดล Enhancing range and attention network ให้ผลลัพธ์ดีที่สุด โดยมีค่า FPS สูงถึง 32.35 และใช้เวลาในการทำนายเพียง 0.031 วินาที ลดเวลาได้มากถึง 72.07% ผลการประเมินนี้ชี้ให้เห็นว่าการปรับปรุงโครงสร้างโมเดลช่วยยกระดับประสิทธิภาพเชิงเวลาอย่างชัดเจน ทำให้เหมาะสมต่อการประยุกต์ใช้งานในสภาพแวดล้อมจริงที่ต้องการการทำงานแบบ real-time

4.2 ผลการทดสอบภาระการคำนวณบนหุ่นยนต์

เพื่อประเมินความเหมาะสมของโมเดลตรวจจับวัตถุสามมิติที่ถูกพัฒนาขึ้นสำหรับการนำไปใช้งานจริงบนหุ่นยนต์อัตโนมัติ จึงได้ดำเนินการทดสอบภาระการประมวลผลบนฮาร์ดแวร์ Jetson Xavier AGX ซึ่งเป็นอุปกรณ์ประมวลผลบนหุ่นยนต์ โดยใช้เครื่องมือ Jetson Stats (jtop) เป็น

ซอฟต์แวร์ที่ใช้สำหรับติดตามการใช้งาน CPU, GPU แบบเรียลไทม์ และทดสอบภาระการประมวลผลบน Onboard Computer เพื่อเปรียบเทียบการทำงานของฮาร์ดแวร์ที่แตกต่างกัน โดยวัดผลภาระการคำนวณบนหุ่นยนต์จะแบ่งสถานการณ์ที่ทดสอบ 4 กรณี

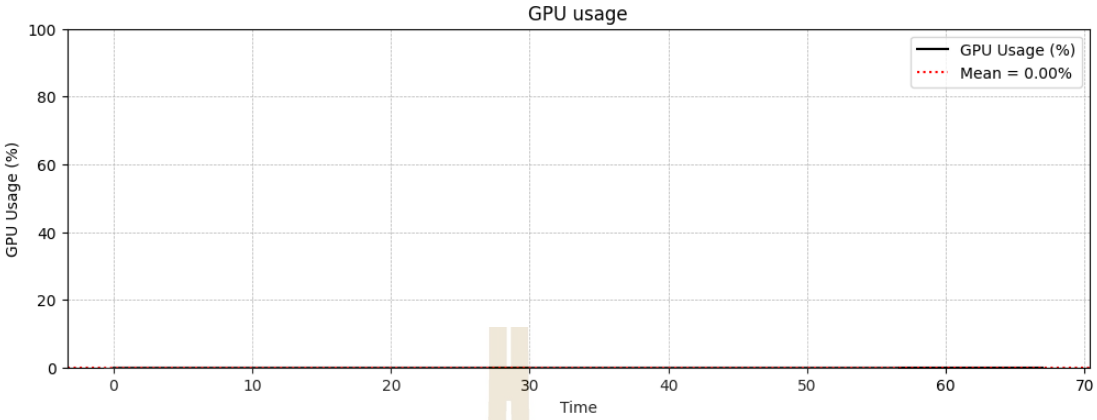
กรณี 1: เปิดเครื่องฮาร์ดแวร์ ไม่รันโปรแกรมใด ๆ



รูปที่ 4.18 การใช้งาน CPU ในสถานะฮาร์ดแวร์ ไม่รันโปรแกรมใด ๆ

จากภาพที่ 4.18 ได้ทำการทดสอบภาระการประมวลผลในสถานะที่ยังไม่รันโปรแกรมใด ๆ บน Jetson Xavier AGX เพื่อใช้เป็นค่าพื้นฐานเปรียบเทียบกับสถานะอื่น โดยใช้เครื่องมือ Jetson Stats (jtop) ในการบันทึกค่าการใช้งานของ CPU แบบเรียลไทม์ ผลลัพธ์จาก รูปที่ 4.8 แสดงให้เห็นว่าอัตราการใช้งาน CPU ของทั้ง 6 คอร์ในช่วงเวลาการทำงานพื้นฐาน มีค่าเฉลี่ยอยู่ที่ประมาณ 15–18% ตลอดช่วงเวลา 60 วินาที ซึ่งถือว่าอยู่ในระดับต่ำและแสดงให้เห็นว่าระบบไม่ได้ถูกโหลดจากกระบวนการพื้นหลังมากนัก

กราฟแสดงแนวโน้มคงที่ของการใช้งาน CPU โดยไม่มีการกระชากหรือสวิงของค่า ซึ่งสะท้อนว่าสภาวะแวดล้อมของฮาร์ดแวร์มีเสถียรภาพ และเหมาะสมที่จะใช้เป็นเกณฑ์เปรียบเทียบกับกรณีที่มีการรันโมเดลหรือระบบควบคุมหุ่นยนต์แบบเต็มระบบในขั้นตอนถัดไป

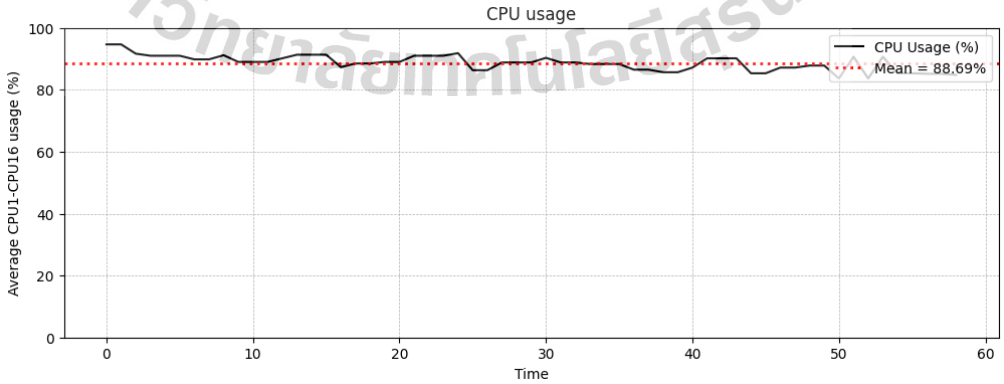


รูปที่ 4.19 การใช้งาน GPU ในสถานะฮาร์ดแวร์ ไม่รันโปรแกรมใด ๆ

จากภาพที่ 4.19 ได้ทำการทดสอบภาระการประมวลผลในสถานะที่ยังไม่รันโปรแกรมใด ๆ บน Jetson Xavier AGX เพื่อใช้เป็นค่าพื้นฐานเปรียบเทียบกับสถานะอื่น โดยใช้เครื่องมือ Jetson Stats (jtop) ในการบันทึกค่าการใช้งานของ GPU แบบเรียลไทม์ ผลลัพธ์จาก รูปที่ 4.19 แสดงให้เห็นว่าอัตราการใช้งาน GPU ของทั้ง 6 คอร์ในช่วงเวลาการทำงานพื้นฐาน มีค่าเฉลี่ยอยู่ที่ประมาณ 0% ตลอดช่วงเวลา 60 วินาที ซึ่งถือว่าอยู่ในระดับต่ำและแสดงให้เห็นว่าระบบไม่ได้ถูกโหลดจากกระบวนการพื้นหลังมากนัก

กราฟแสดงแนวโน้มคงที่ของการใช้งาน GPU โดยไม่มีการเปลี่ยนแปลงค่าใด ๆ ซึ่งสะท้อนว่าสภาวะแวดล้อมของฮาร์ดแวร์มีเสถียรภาพ และเหมาะสมที่จะใช้เป็นเกณฑ์เปรียบเทียบกับกรณีที่มีการรันโมเดลหรือระบบควบคุมหุ่นยนต์แบบเต็มระบบในขั้นตอนถัดไป

กรณี 2: เปิดระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดลตรวจจับ

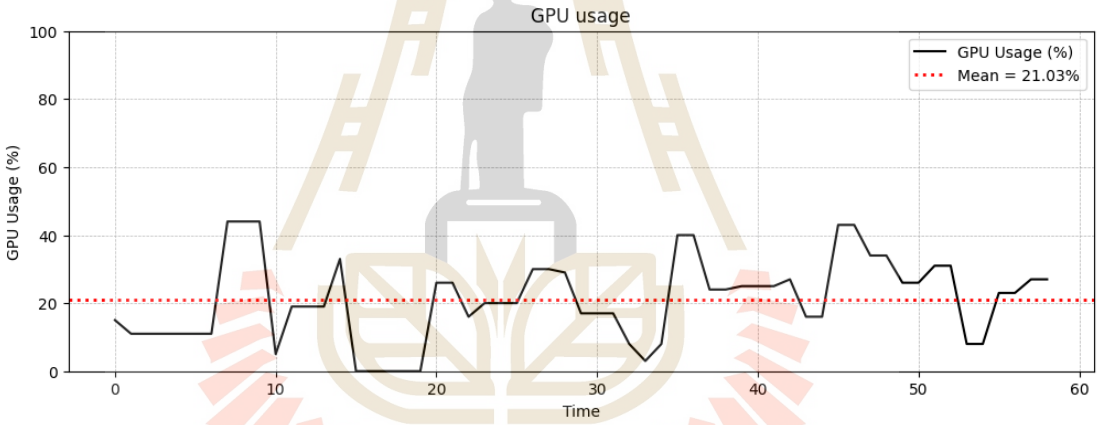


รูปที่ 4.20 การใช้งาน CPU ในสถานะระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดลตรวจจับ

จากภาพที่ 4.20 ได้ทำการทดสอบการใช้งานทรัพยากรเมื่อเปิดใช้งานระบบควบคุมอัตโนมัติแบบเต็มรูปแบบ (Full Autonomy) ซึ่งรวมถึงระบบนำทาง การรับข้อมูลจากเซนเซอร์ต่าง ๆ และการวางแผนเส้นทาง โดยยังไม่มีการรันโมเดลตรวจจับวัตถุสามมิติ เพื่อวัดผลกระทบของระบบสนับสนุนต่อภาระการคำนวณของหน่วยประมวลผล

จากผลการทดสอบที่แสดงใน รูปที่ 4.20 พบว่าอัตราการใช้งาน CPU เพิ่มสูงขึ้นอย่างมีนัยสำคัญ โดยมีค่าเฉลี่ยตลอดช่วงการวัดอยู่ที่ประมาณ 88–95% และยังคงที่ในระดับสูงอย่างต่อเนื่องตลอดช่วงเวลาการทดสอบ

พฤติกรรมดังกล่าวสะท้อนว่า แม้จะยังไม่รันโมเดลตรวจจับวัตถุ แต่ระบบควบคุมอัตโนมัติแบบเรียลไทม์มีภาระการประมวลผลที่สูงมาก ซึ่งอาจกระทบต่อความสามารถในการรองรับโมเดลประมวลผลเชิงลึกเพิ่มเติมหากไม่มีการเพิ่มประสิทธิภาพการจัดสรรทรัพยากรหรือลดการประมวลผลของโมเดล



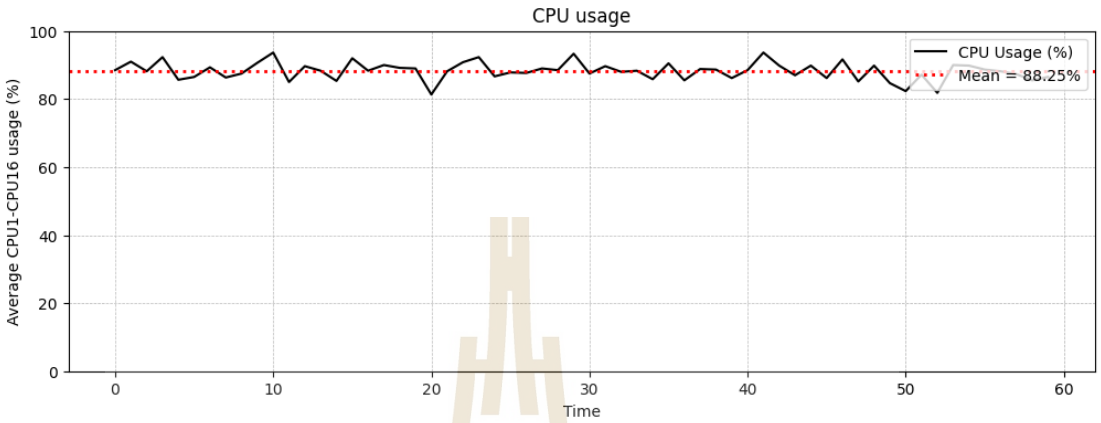
รูปที่ 4.21 การใช้งาน GPU ในสถานะระบบอัตโนมัติเต็มระบบโดยไม่รันโมเดลตรวจจับ

จากภาพที่ 4.21 นอกจากการใช้งานของ CPU แล้ว ยังได้ทำการตรวจสอบภาระการใช้งานของหน่วยประมวลผลกราฟิก (GPU) ในสถานะที่ระบบอัตโนมัติทำงานเต็มระบบ โดยยังไม่รันโมเดลตรวจจับ เพื่อแยกผลกระทบที่เกิดจากโมดูลควบคุมอัตโนมัติและระบบสนับสนุนอื่น ๆ

จากผลลัพธ์ที่แสดงใน รูปที่ 4.21 พบว่า GPU มีการใช้งานในระดับปานกลาง โดยมีความผันผวนของค่าอยู่ในช่วงประมาณ 10–55% ตลอดช่วงเวลาทดสอบ ซึ่งเกิดจากการใช้งานของระบบย่อยบางส่วนที่อาจพึ่งพา GPU สำหรับการเรนเดอร์แผนที่แบบภาพกราฟิก (Visualization)

อย่างไรก็ตาม ค่าการใช้งาน GPU โดยเฉลี่ยยังอยู่ในระดับที่ต่ำเพียงพอ และ ยังมีทรัพยากรเหลือมากเพียงพอ สำหรับการรันโมเดลการเรียนรู้เชิงลึก เช่น PointPillars ได้อย่างมีประสิทธิภาพในลำดับขั้นถัดไป

กรณี 3.1: รันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Jetson Xavier

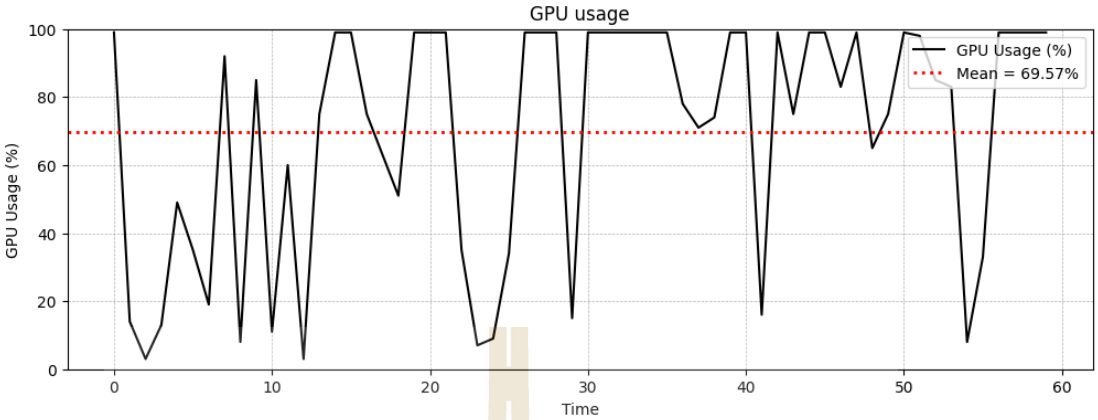


รูปที่ 4.22 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุง

จากผลการทดสอบในรูปที่ 4.22 แสดงข้อมูลการใช้งานของหน่วยประมวลผลกลาง (CPU) ขณะที่ทำการรันโมเดล PointPillars เวอร์ชันดั้งเดิมบนฮาร์ดแวร์ Jetson Xavier AGX โดยไม่มีการเปิดระบบนำทางหรือโมดูลอัตโนมัติอื่น การทดสอบนี้มุ่งเน้นเพื่อวิเคราะห์ภาระการประมวลผลที่เกิดขึ้นจากโมเดลต้นฉบับก่อนการปรับแต่ง

ผลลัพธ์แสดงให้เห็นว่า อัตราการใช้งาน CPU มีความผันผวนสูงในช่วงเวลาเริ่มต้น และโดยเฉลี่ยอยู่ในช่วงประมาณ 30–70% ตลอดระยะเวลาการทดสอบ โดยมีหลายจุดที่ค่าการใช้งานพุ่งสูงขึ้นอย่างเฉียบพลันเกิน 80% ซึ่งอาจเกิดจากกระบวนการโหลดโมเดล, การจัดสรรหน่วยความจำ และการประมวลผลข้อมูล Point Cloud แบบต่อเนื่อง ความผันผวนเหล่านี้สะท้อนถึงลักษณะการใช้งานทรัพยากรที่ไม่มีเสถียรภาพ และแสดงให้เห็นว่าโมเดลเวอร์ชันดั้งเดิมมีการใช้ทรัพยากรในระดับสูง

ลักษณะการใช้งานที่ไม่สม่ำเสมอและระดับการใช้ CPU ที่สูงเช่นนี้อาจส่งผลกระทบต่อการใช้งานร่วมกับระบบอัตโนมัติอื่นบนแพลตฟอร์มเดียวกัน โดยเฉพาะเมื่อจำเป็นต้องมีการประมวลผลแบบเรียลไทม์ร่วมกันหลายโมดูล



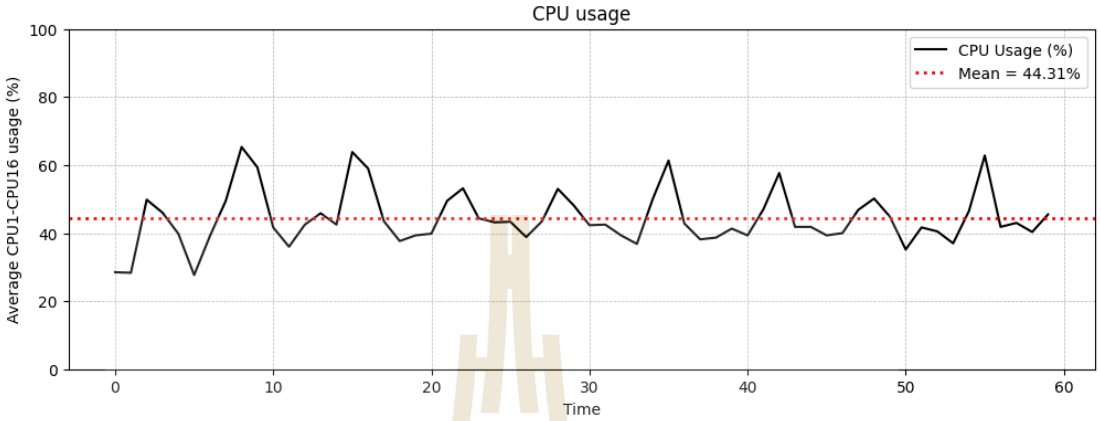
รูปที่ 4.23 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุง

จากรูปที่ 4.23 แสดงการใช้งานหน่วยประมวลผลกราฟิก (GPU) ขณะทำการรันเฉพาะโมเดล PointPillars เวอร์ชันดั้งเดิมบนแพลตฟอร์ม Jetson Xavier AGX โดยไม่มีการใช้งานโมดูลระบบนำทางหรือควบคุมอัตโนมัติอื่น การทดสอบนี้มุ่งเน้นเพื่อประเมินภาระการประมวลผลของโมเดลเดิมก่อนปรับปรุงเมื่อประมวลผลข้อมูล Point Cloud ในลักษณะเรียลไทม์

ผลการทดสอบแสดงให้เห็นถึงลักษณะการใช้งาน GPU ที่ผันผวนรุนแรง โดยช่วงต้นมีการใช้งานอยู่ในระดับต่ำมาก (ต่ำกว่า 20%) และในช่วงกลางถึงปลายของกราฟพบการพุ่งขึ้นถึง 100% อย่างต่อเนื่องหลายครั้ง สลับกับช่วงที่ค่าใช้งานลดลงต่ำสุดอย่างรวดเร็ว ซึ่งสะท้อนลักษณะการประมวลผลแบบที่ไม่ต่อเนื่องและเกิดเป็นช่วงๆที่ไม่สม่ำเสมอ

พฤติกรรมดังกล่าวบ่งชี้ถึงความไม่มีเสถียรภาพในการใช้ทรัพยากร GPU ของโมเดลเดิม โดยเฉพาะอย่างยิ่งในช่วงหลังที่เกิดอาการสภาวะการใช้งานเต็มขีดความสามารถของหน่วยประมวลผลอย่างต่อเนื่อง อาจส่งผลให้เกิดคอขวด (bottleneck) ในระบบ หากมีการรันร่วมกับโมดูลอื่นในสภาพแวดล้อมจริง ทั้งนี้อัตราการใช้ GPU ที่พุ่งถึง 100% ซ้ำซ้อนในช่วงเวลาสั้น ๆ อาจบ่งชี้ถึงการจัดการทรัพยากร ที่ไม่เหมาะสมหรือเกิดจากการใช้ทรัพยากร CUDA อย่างไม่มีประสิทธิภาพ

กรณี 3.2: รันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้วบน Jetson Xavier

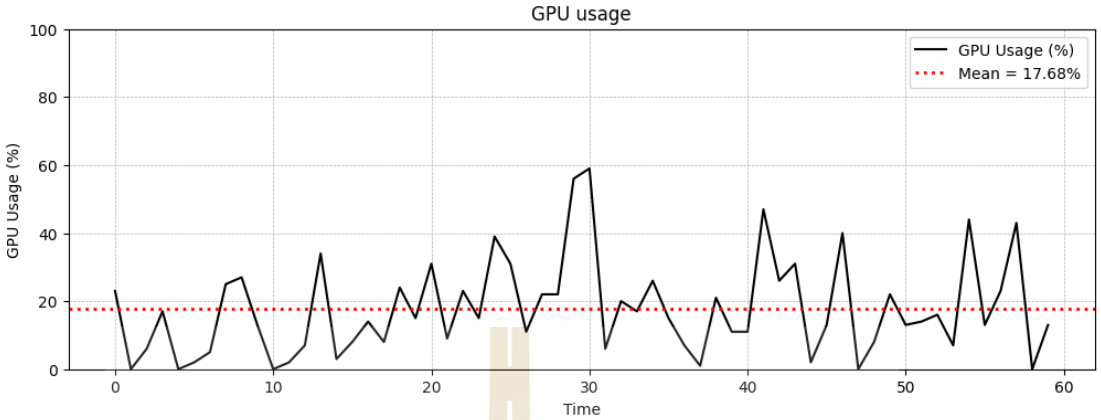


รูปที่ 4.24 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว

จากภาพที่ 4.24 ได้ทำการทดสอบภาระการประมวลผลของหน่วยประมวลผลกลาง (CPU) ขณะทำการรันเฉพาะโมเดลตรวจจับวัตถุสามมิติ PointPillars โดยไม่เปิดใช้ระบบควบคุมอัตโนมัติอื่น ๆ เพื่อให้เห็นผลกระทบของโมเดลโดยตรงต่อการใช้ทรัพยากรของระบบ

จากผลการวัดใน รูปที่ 4.24 พบว่าอัตราการใช้งานของ CPU มีค่าเฉลี่ยอยู่ที่ประมาณ 35–40% ตลอดช่วงเวลาทดสอบ โดยมีความผันผวนเล็กน้อยในช่วงต้นซึ่งเกิดจากกระบวนการเริ่มต้นการโหลดโมเดลและการจัดสรรหน่วยความจำ ก่อนที่ค่าจะคงที่ในระดับปานกลาง

ค่าดังกล่าวแสดงให้เห็นว่าโมเดล PointPillars ที่ได้รับการปรับแต่งสามารถประมวลผลได้อย่างมีประสิทธิภาพ โดยไม่ก่อให้เกิดภาระเกินความสามารถของหน่วยประมวลผลหลักของ Jetson Xavier AGX ซึ่งสอดคล้องกับเป้าหมายของงานวิจัยที่มุ่งเน้นการใช้งานโมเดลบนอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร (Edge AI)



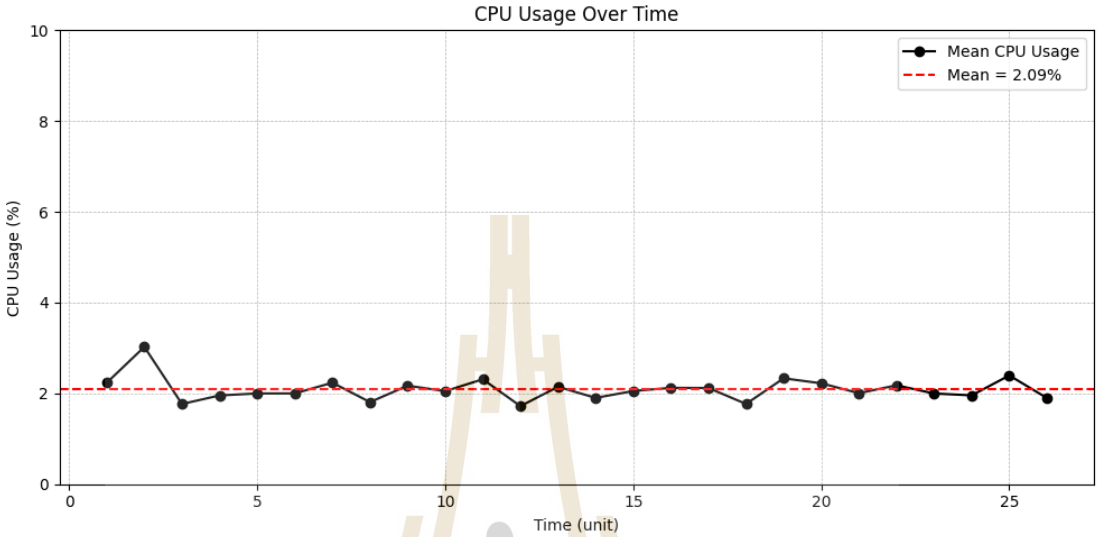
รูปที่ 4.25 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว

จากภาพที่ 4.25 ซึ่งเป็นการรันเฉพาะโมเดล PointPillars โดยไม่เปิดระบบควบคุมอัตโนมัติอื่น ได้ทำการตรวจสอบภาระการใช้งานของหน่วยประมวลผลกราฟิก (GPU) เพื่อประเมินการใช้ทรัพยากรในการประมวลผลเชิงลึกแบบเรียลไทม์

ผลลัพธ์จาก รูปที่ 4.25 แสดงให้เห็นว่าการใช้งาน GPU มีความผันผวนค่อนข้างมาก โดยมีค่าขั้นต่ำอยู่ที่ประมาณ 10% และพุ่งสูงเป็นระยะ ๆ ถึงระดับ 80–90% ในช่วงเวลาที่โมเดลทำการ inference กับข้อมูลพอยต์คลาวด์แต่ละชุด

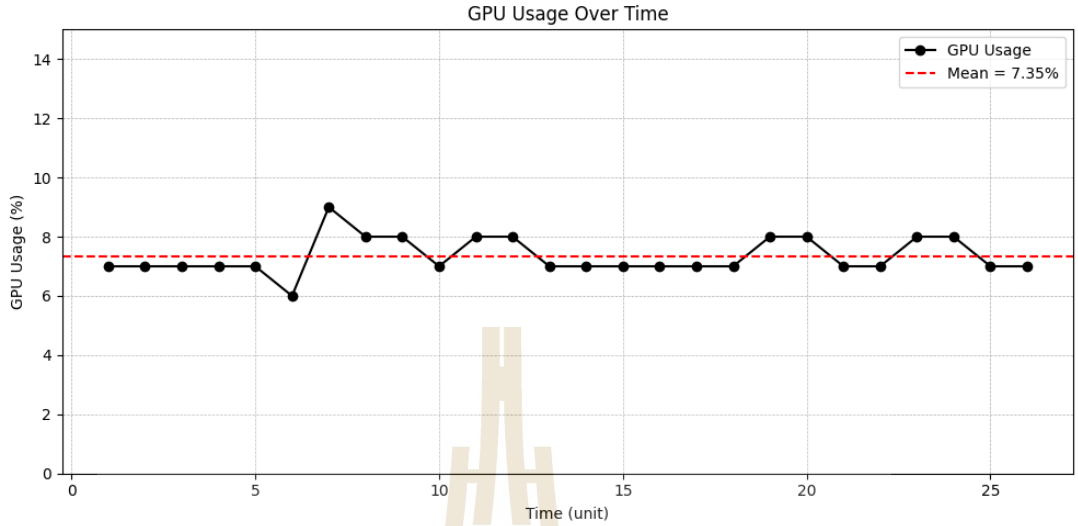
อย่างไรก็ตาม โดยรวมแล้ว GPU ยังสามารถรองรับภาระงานได้โดยไม่เกิดอาการค้างหรือทรัพยากรเต็ม และมีช่วงเวลาที่ GPU กลับมาอยู่ในระดับต่ำสลับเป็นช่วง ๆ ซึ่งแสดงถึงความสามารถในการประมวลผลแบบ dynamic ตามภาระงานจริง ทั้งนี้ลักษณะการทำงานแบบสวิงของค่า ดังกล่าวเป็นเรื่องปกติของโมเดล deep learning บนอุปกรณ์ที่มีตัวประมวลผลที่จำกัด

กรณี 3.3: รันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Onboard Computer



รูปที่ 4.26 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Onboard Computer

จากผลการทดสอบในรูปที่ 4.26 แสดงข้อมูลการใช้งานของหน่วยประมวลผลกลาง (CPU) เมื่อทำการรันโมเดล PointPillars เวอร์ชันดั้งเดิมบนคอมพิวเตอร์บนรถกอล์ฟอัตโนมัติ (Onboard Computer) (วงศธร,2565) โดยไม่มีการเปิดใช้งานโมดูลอื่นร่วม ผลลัพธ์จากกราฟแสดงให้เห็นว่าค่าเฉลี่ยของการใช้งาน CPU อยู่ที่ประมาณ 2.09% ซึ่งอยู่ในระดับต่ำมากและมีความคงที่ตลอดช่วงเวลาการทดสอบ โดยค่าการใช้งานสูงสุดไม่เกิน 4% การใช้งานที่อยู่ในระดับต่ำและมีเสถียรภาพสูงนี้สะท้อนให้เห็นว่า Onboard Computer สามารถจัดการภาระการประมวลผลของโมเดล PointPillars เวอร์ชันดั้งเดิมได้อย่างมีประสิทธิภาพโดยไม่เกิดการใช้งานทรัพยากรเกินจำเป็น

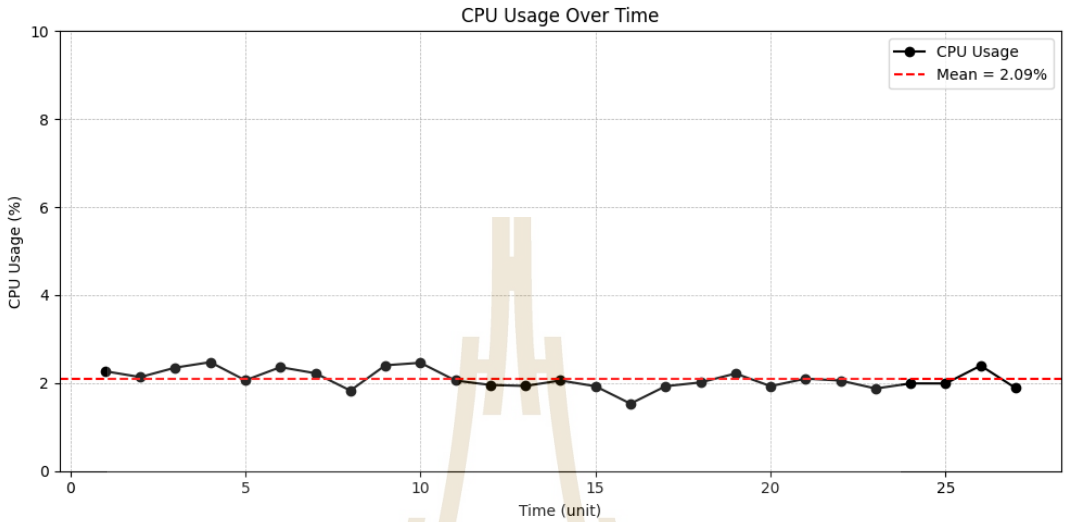


รูปที่ 4.27 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ก่อนปรับปรุงบน Onboard Computer

จากผลการทดสอบในรูปที่ 4.27 แสดงข้อมูลการใช้งานของหน่วยประมวลผลกราฟิก (GPU) ขณะทำการรันโมเดล PointPillars เวอร์ชันดั้งเดิมบน Onboard Computer โดยไม่มีการเปิดใช้งานโมดูลอื่นร่วม การทดสอบนี้มีวัตถุประสงค์เพื่อประเมินภาระของหน่วยประมวลผลด้านกราฟิกที่เกิดขึ้นจากการประมวลผลโมเดลในสภาวะพื้นฐานก่อนการปรับปรุง

จากกราฟในรูปที่ 4.27 พบว่าอัตราการใช้งาน GPU เฉลี่ยอยู่ที่ประมาณ **7.35%** ซึ่งถือว่าค่อนข้างต่ำเมื่อเทียบกับศักยภาพของการ์ดประมวลผลที่ติดตั้งใน Onboard Computer การใช้งานมีความคงที่ส่วนใหญ่ระหว่าง 6% ถึง 9% และไม่มีการเพิ่มขึ้นอย่างมีนัยสำคัญ แสดงให้เห็นว่าการประมวลผลของโมเดล PointPillars ดั้งเดิมไม่ใช้ทรัพยากร GPU ในระดับสูงมากนัก ซึ่งอาจเป็นผลมาจากโครงสร้างโมเดลที่มีการใช้การคำนวณแบบเบาบาง (sparse computation) และมีการพึ่งพา CPU ในขั้นตอนการเตรียมข้อมูลก่อนเข้าสู่เครือข่ายมากกว่า

กรณี 3.4: รันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้วบน Onboard Computer



รูปที่ 4.28 การใช้งาน CPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว Onboard Computer

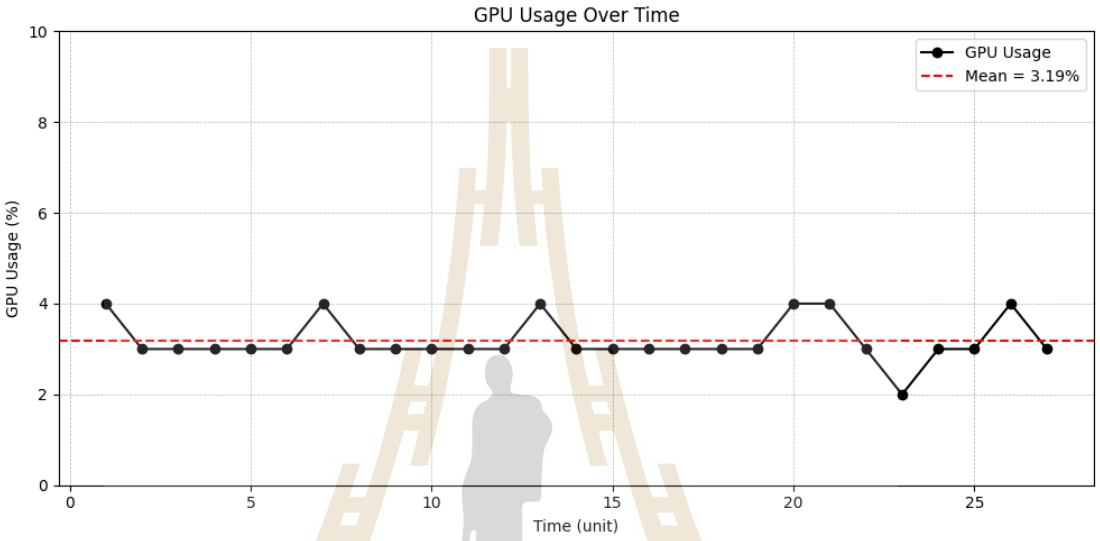
จากผลการทดสอบในรูปที่ 4.28 แสดงให้เห็นว่าหลังจากการปรับปรุงโครงสร้างของโมเดล PointPillars แล้ว การใช้งาน CPU ยังคงอยู่ในระดับต่ำอย่างมาก โดยมีค่าเฉลี่ยอยู่ที่ประมาณ 2.09% เช่นเดียวกับก่อนการปรับปรุง กราฟแสดงแนวโน้มที่มีความคงที่สูงโดยไม่มีความผันผวนอย่างมีนัยสำคัญ แสดงให้เห็นว่าโมเดลที่ได้รับการปรับปรุงยังคงรักษาประสิทธิภาพด้านการใช้ทรัพยากร CPU ได้ดี และมีการกระจายภาระการคำนวณระหว่าง CPU และ GPU อย่างเหมาะสม

ผลลัพธ์นี้ชี้ให้เห็นว่าการปรับปรุงโครงสร้างโมเดล เช่น การลดจำนวนฟีเจอร์ที่ไม่จำเป็น และการเพิ่มกลไก Attention Network ไม่ได้เพิ่มภาระให้กับ CPU ของระบบ Onboard Computer ซึ่งบ่งบอกถึงความเหมาะสมของการออกแบบโมเดลสำหรับการประมวลผลบนฮาร์ดแวร์ประสิทธิภาพสูงในสภาพแวดล้อมจริง

จากการวิเคราะห์ผลการทดสอบใน รูปที่ 4.26 และ รูปที่ 4.28 ซึ่งแสดงการใช้งานของหน่วยประมวลผลกลาง (CPU) ของ Onboard Computer ทั้งก่อนและหลังการปรับปรุงโมเดล PointPillars พบว่า ค่าเฉลี่ยของการใช้งาน CPU อยู่ในระดับต่ำใกล้เคียงกัน โดยมีแนวโน้มความคงที่สูงและไม่มีการผันผวนของค่าการใช้งานอย่างมีนัยสำคัญ

ผลลัพธ์ดังกล่าวสะท้อนให้เห็นว่า Onboard Computer ที่ใช้ในการทดสอบมีประสิทธิภาพการประมวลผลสูง โดยเฉพาะจากการที่มีหน่วยประมวลผลแบบ 16 คอร์ (16-Core CPU) ทำให้สามารถกระจายภาระการคำนวณของโมเดลได้อย่างมีประสิทธิภาพ จึงส่งผลให้การใช้งาน CPU

โดยรวมอยู่ในระดับต่ำทั้งในเวอร์ชันก่อนและหลังการปรับปรุงโมเดล ทั้งนี้แสดงให้เห็นว่า ความแตกต่างด้านประสิทธิภาพของ CPU ระหว่างสองเวอร์ชันไม่ได้มีนัยสำคัญมากนัก เนื่องจากข้อได้เปรียบทางฮาร์ดแวร์ของ Onboard Computer เองที่สามารถรองรับการประมวลผลได้อย่างราบรื่น

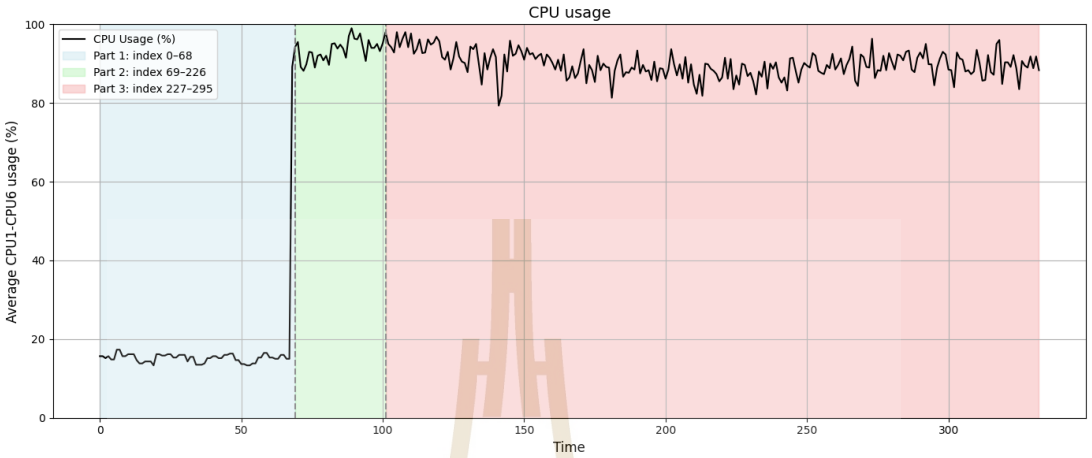


รูปที่ 4.29 การใช้งาน GPU ในสถานะรันเฉพาะโมเดล PointPillars ที่ปรับปรุงแล้ว Onboard Computer

จากรูปที่ 4.29 แสดงการใช้งาน GPU ของ Onboard Computer ในขณะรันเฉพาะโมเดล PointPillars ที่ได้รับการปรับปรุง พบว่าการใช้งาน GPU มีค่าเฉลี่ยอยู่ที่ **3.19%** โดยกราฟมีความคงที่และมีการเปลี่ยนแปลงเพียงเล็กน้อยในช่วงการประมวลผล ซึ่งสะท้อนให้เห็นว่าโมเดลที่ปรับปรุงแล้วสามารถทำงานได้อย่างมีประสิทธิภาพและใช้ทรัพยากร GPU ในระดับต่ำมาก การใช้ GPU ที่น้อยนี้เกิดจากการลดภาระการคำนวณในส่วนของ Backbone และการประมวลผล Feature Map ที่ได้รับการปรับให้เหมาะสม ส่งผลให้การทำงานของโมเดลไม่ก่อให้เกิดภาระเพิ่มเติมต่อระบบประมวลผลกราฟิกของ Onboard Computer

เมื่อเปรียบเทียบระหว่างรูปที่ 4.27 และ 4.29 จะเห็นได้ชัดว่าการใช้งาน GPU ของโมเดลที่ปรับปรุงแล้วมีแนวโน้มลดลงอย่างมีนัยสำคัญจาก **7.35%** เหลือเพียง 3.19% ซึ่งสะท้อนให้เห็นถึงประสิทธิภาพในการปรับปรุงสถาปัตยกรรมโมเดลให้เหมาะสมกับการประมวลผลบน Onboard Computer ได้ดียิ่งขึ้น โดยยังคงรักษาการทำงานแบบเรียลไทม์และใช้พลังงานคำนวณอย่างมีประสิทธิภาพมากกว่าเดิม

กรณี 4.1: รันโมเดล PointPillars ก่อนปรับปรุงพร้อมระบบขับเคลื่อนอัตโนมัติเต็มระบบ

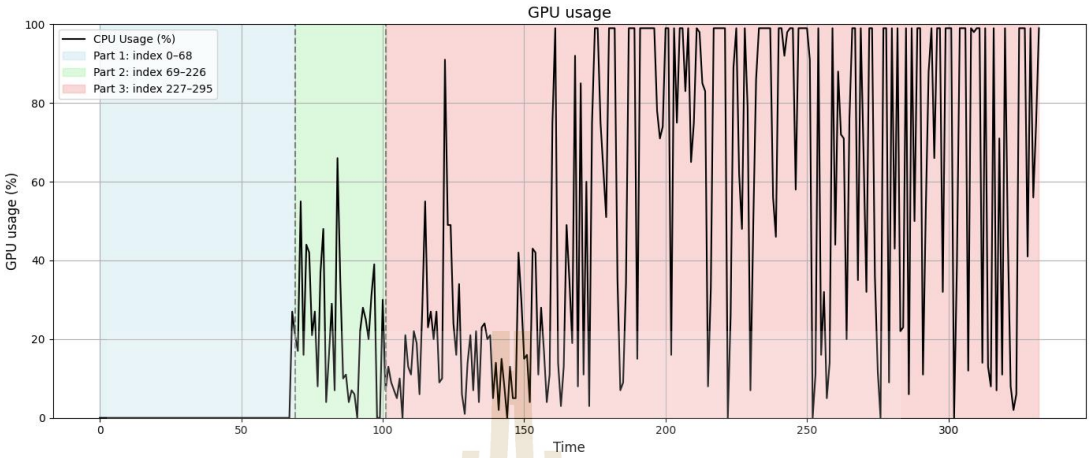


รูปที่ 4.30 การใช้งาน CPU ในสถานะรันโมเดล PointPillars ก่อนปรับปรุงพร้อมระบบอัตโนมัติ

จากรูปที่ 4.30 สามารถแบ่งลำดับเวลาออกเป็น 3 ช่วงเพื่อวิเคราะห์พฤติกรรมการใช้ทรัพยากรของหน่วยประมวลผลกลาง (CPU) ได้แก่ ช่วงไม่มีกระบวนการทำงาน (Index 0–68; แรเงาสีน้ำเงิน), ช่วงที่รันเฉพาะระบบอัตโนมัติ (Index 69–226; แรเงาสีเขียว), และช่วงที่รันทั้งระบบอัตโนมัติร่วมกับโมเดล PointPillars เวอร์ชันดั้งเดิม (Index 227–295; แรเงาสีแดง)

ในช่วงเริ่มต้นที่ยังไม่มีกระบวนการทำงานใด ๆ พบว่าค่าเฉลี่ยการใช้งาน CPU อยู่ในระดับต่ำที่ประมาณ 15–18% ซึ่งใช้เป็นฐานเปรียบเทียบเพื่อประเมินผลกระทบจากการเปิดใช้งานระบบต่าง ๆ ต่อเนื่องกัน เมื่อเข้าสู่ช่วงที่เปิดใช้งานระบบอัตโนมัติเต็มรูปแบบ (ช่วงสีเขียว) ค่า CPU เพิ่มขึ้นอย่างรวดเร็ว โดยมีค่าเฉลี่ยอยู่ที่ประมาณ 90–95% และในช่วงสุดท้ายที่มีการรันระบบอัตโนมัติควบคู่กับโมเดล PointPillars แบบดั้งเดิมพร้อมกัน (ช่วงสีแดง) พบว่าการใช้งาน CPU เพิ่มสูงขึ้นต่อเนื่องโดยมีค่าเฉลี่ยประมาณ 96.4% และค่าสูงสุดแตะระดับ 99.7% ตลอดช่วงเวลา

พฤติกรรมดังกล่าวสะท้อนให้เห็นว่าโมเดล PointPillars แบบดั้งเดิมยังคงมีภาระการประมวลผลที่สูงเมื่อทำงานร่วมกับระบบนำทางอัตโนมัติ ซึ่งส่งผลให้หน่วยประมวลผลกลางทำงานในระดับเกือบเต็มศักยภาพอย่างต่อเนื่อง

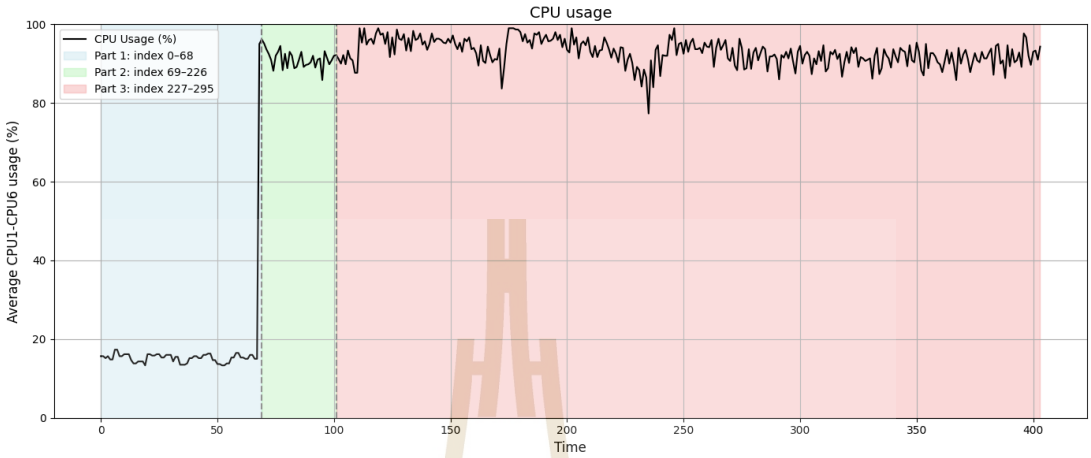


รูปที่ 4.31 การใช้งาน GPU ในสถานะรันโมเดล PointPillars ก่อนปรับปรุงพร้อมระบบอัตโนมัติ

จากรูปที่ 4.31 แสดงให้เห็นถึงพฤติกรรมของการใช้งานหน่วยประมวลผลกราฟิก (GPU) ขณะรันโมเดล PointPillars เวอร์ชันดั้งเดิมร่วมกับระบบควบคุมอัตโนมัติเต็มรูปแบบบนแพลตฟอร์ม Jetson Xavier AGX โดยช่วงเวลาทดสอบแบ่งออกเป็น 3 ช่วง ได้แก่ ช่วงไม่มีกระบวนการทำงาน (แรเงาสีน้ำเงิน), ช่วงเปิดระบบอัตโนมัติแต่ไม่รันโมเดล (แรเงาสีเขียว), และช่วงรันทั้งระบบอัตโนมัติและโมเดล PointPillars (แรเงาสีแดง)

ในช่วงแรเงาสีแดง ซึ่งเป็นช่วงที่มีการทำงานของโมเดล PointPillars ร่วมกับระบบควบคุมอัตโนมัติ พบว่า GPU มีการใช้งานที่ผันผวนสูง และพุ่งขึ้นถึงค่าสูงสุด 100% อย่างต่อเนื่อง โดยเฉพาะในช่วงเวลา 180–320 ซึ่งเป็นช่วงที่ระบบกำลังประมวลผลข้อมูล Point Cloud อย่างเข้มข้น ค่าเฉลี่ยของการใช้งาน GPU ในช่วงนี้อยู่ที่ 73.4% โดยมีค่าสูงสุดที่ 100% และต่ำสุดที่ 9.7% สะท้อนให้เห็นว่าการประมวลผลเกิดขึ้นแบบไม่สม่ำเสมอ หรือเกิดเป็นช่วง ๆ ซึ่งอาจทำให้เวลาการตอบสนองของระบบไม่แน่นอน และนำไปสู่ปัญหาความล่าช้า ซึ่งส่งผลต่อความเสถียรของระบบในสถานการณ์ที่ต้องตัดสินใจหรือควบคุมแบบเรียลไทม์ เช่น การหลีกเลี่ยงสิ่งกีดขวาง

กรณี 4.2: รันโมเดล PointPillars ที่ปรับปรุงแล้วพร้อมระบบขับเคลื่อนอัตโนมัติเต็มระบบ

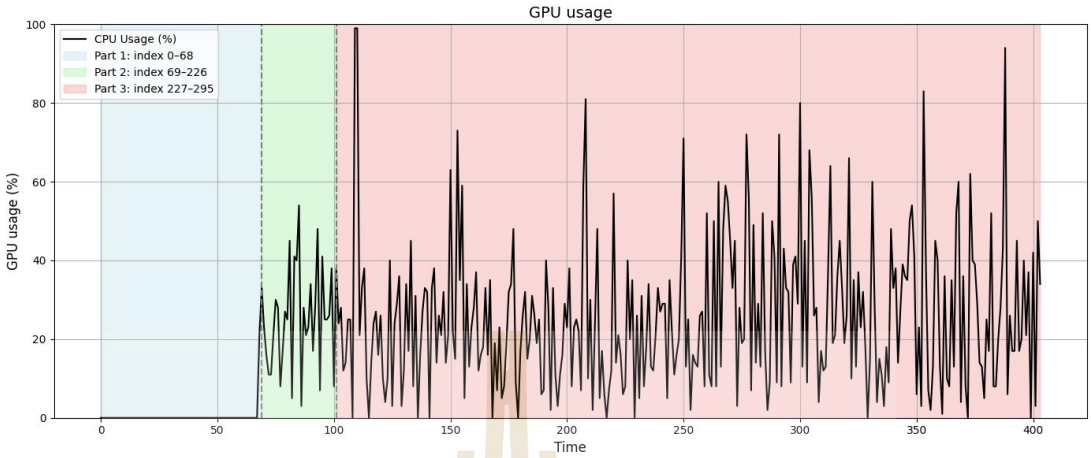


รูปที่ 4.32 การใช้งาน CPU ในสถานะรันโมเดล PointPillars ที่ปรับปรุงแล้วพร้อมระบบอัตโนมัติ

ในรูปที่ 4.32 แสดงผลการใช้งานหน่วยประมวลผลกลาง (CPU) ของระบบหุ่นยนต์อัตโนมัติ ขณะรันโมเดล PointPillars ที่ได้รับการปรับปรุงร่วมกับระบบควบคุมอัตโนมัติเต็มระบบบน Jetson Xavier AGX โดยแบ่งช่วงเวลาออกเป็น 3 ช่วง ได้แก่ ช่วงไม่รันโปรแกรมใด ๆ (แรเงาสีน้ำเงิน), ช่วงเปิดระบบอัตโนมัติ (แรเงาสีเขียว), และช่วงที่ระบบทำงานร่วมกับโมเดลตรวจจับที่ปรับปรุงแล้ว (แรเงาสีแดง)

ผลการทดสอบแสดงให้เห็นว่า ช่วงแรเงาสีแดงซึ่งเป็นช่วงที่ระบบรันทั้งโมเดล PointPillars และระบบควบคุมอัตโนมัติ ค่าเฉลี่ยของการใช้งาน CPU อยู่ที่ประมาณ 91.7% โดยมีแนวโน้มที่คงที่และไม่กระชากสูงเท่ากับเวอร์ชันดั้งเดิมที่ยังไม่ปรับปรุง (ซึ่งมีค่าเฉลี่ยสูงถึง 96.8%) ความผันผวนของค่าถูกลดลงอย่างชัดเจน แสดงให้เห็นถึง ประสิทธิภาพของการปรับแต่งโมเดล ทั้งในด้านโครงสร้างเบากว่า (Lightweight) และการใช้ Attention Mechanism เพื่อเน้นเฉพาะข้อมูลที่จำเป็น ทำให้ลดภาระของ CPU ลงได้

ลักษณะการใช้งาน CPU ที่เสถียรมากขึ้นนี้ชี้ให้เห็นว่าโมเดลที่ปรับปรุงแล้วสามารถ ทำงานแบบเรียลไทม์ร่วมกับระบบนำทางอัตโนมัติ ได้โดยไม่ก่อให้เกิดความล่าช้าหรือการใช้ทรัพยากรเกินขีดจำกัดของแพลตฟอร์ม Edge Computing อย่าง Jetson Xavier AGX จึงถือเป็นหลักฐานสนับสนุนว่าการปรับแต่งสถาปัตยกรรมของโมเดลสามารถยกระดับการนำไปใช้งานจริงได้อย่างมีประสิทธิภาพและเสถียรภาพ



รูปที่ 4.33 การใช้งาน GPU ในสถานะรันโมเดล PointPillars ที่ปรับปรุงแล้วพร้อมระบบอัตโนมัติ

จากรูปที่ 4.33 ซึ่งแสดงพฤติกรรมการใช้งานหน่วยประมวลผลกราฟิก (GPU) ในกรณีที่มีการรันโมเดล PointPillars ที่ได้รับการปรับปรุงร่วมกับระบบขับเคลื่อนอัตโนมัติเต็มระบบบน Jetson Xavier AGX สามารถสรุปอธิบายได้ในลักษณะเชิงวิชาการดังนี้:

ในช่วงแรเงาสีแดง (Index 227–295) ซึ่งเป็นช่วงที่ระบบทำงานพร้อมกันทั้งโมเดล 3D Detection และระบบนำทาง พบว่าการใช้งาน GPU มีลักษณะการผันผวนที่ลดลงจากกรณีก่อนการปรับปรุง โดยมีค่าเฉลี่ยการใช้งาน GPU อยู่ที่ประมาณ 43.8% ค่าสูงสุดอยู่ที่ 91.3% และค่าต่ำสุดที่ 4.7% ซึ่งแสดงถึงพฤติกรรมการใช้งานที่สมดุลและต่อเนื่องมากขึ้นเมื่อเทียบกับกราฟก่อนปรับปรุงที่มีค่าสูงสุดแตะ 100% อย่างต่อเนื่อง และแสดงลักษณะการทำงานที่มีภาระสูงแบบเป็นระยะๆอย่างชัดเจน

การลดความถี่ของการ spike และการกระจายโหลดที่สม่ำเสมอกว่าเดิมในช่วงเวลาการทำงานจริง แสดงให้เห็นว่าโครงสร้างโมเดล PointPillars ที่ได้รับการปรับปรุง เช่น การลด Point Cloud Range, ปรับ Voxel Size และนำ Lightweight Attention Pyramid Network (LAPN) มาใช้ มีผลในการลดภาระการประมวลผลบน GPU อย่างมีนัยสำคัญ

ดังนั้น พฤติกรรมการใช้งาน GPU ที่แสดงในรูปนี้สะท้อนให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้นของโมเดลที่ได้รับการปรับปรุง ซึ่งมีความเหมาะสมต่อการประยุกต์ใช้ในระบบหุ่นยนต์อัตโนมัติที่ทำงานในสภาพแวดล้อมแบบเรียลไทม์ โดยยังคงสามารถรักษาความแม่นยำในการตรวจจับได้ ขณะเดียวกันก็ลดโอกาสเกิดความไม่เสถียรของเวลาประมวลผล และปัญหาด้านการจัดสรรทรัพยากรบนแพลตฟอร์ม edge ที่มีข้อจำกัดด้านการประมวลผล

4.3 ผลการทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้

การทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้และการหยุดเมื่อเสี่ยงต่อการชน ถือเป็นขั้นตอนสำคัญในกระบวนการพัฒนารถอัตโนมัติหรือระบบหุ่นยนต์ที่ต้องการความปลอดภัยสูง หุ่นยนต์จำเป็นต้องสามารถตรวจจับวัตถุที่อยู่ด้านหน้าได้อย่างแม่นยำ และประมวลผลการตัดสินใจเพื่อชะลอความเร็วหรือหยุดการเคลื่อนที่ได้ทันเวลา ข้อมูลที่ใช้ในการตรวจจับได้มาจากเซ็นเซอร์ LiDAR ซึ่งช่วยให้สามารถระบุตำแหน่งและรูปร่างของวัตถุได้อย่างถูกต้อง เมื่อวัตถุอยู่ในระยะที่อาจก่อให้เกิดอันตราย ระบบจะทำการประมวลผลตำแหน่งและส่งคำสั่งหยุดหุ่นยนต์โดยอัตโนมัติ ผลการทดสอบดังกล่าวได้นำเสนอไว้ในตารางที่ 4.9 และ 4.10 เพื่อตรวจสอบทั้งความสามารถในการตรวจจับและการตอบสนองของระบบต่อสถานการณ์จริง

ตารางที่ 4.9 ทดสอบการตรวจจับ 3 มิติ

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะที่ต้องการให้ตรวจจับและหยุด (เมตร)	ระยะที่โมเดลตรวจจับได้ (เมตร)	ความเร็วหุ่นยนต์ ขณะทดสอบ (km/hr)
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.1775	1.31
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.275	1.34
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.215	1.28
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.1825	1.29
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.2275	1.31
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.0325	1.28
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.24	1.27
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.26	1.29

ตารางที่ 4.9 ทดสอบการตรวจจับ 3 มิติ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะที่ต้องการให้ตรวจจับและหยุด (เมตร)	ระยะที่โมเดลตรวจจับได้ (เมตร)	ความเร็วหุ่นยนต์ ขณะทดสอบ (km/hr)
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.2125	1.27
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.2225	1.26
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.2025	1.28
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	5.64	1.28
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.15	1.82225
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.195	1.83795
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.1575	1.86162
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.1475	1.8182
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.2	1.79854
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.1375	1.8419
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.13	1.86714
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	3.9925	1.81934
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	5.5375	1.93599

ตารางที่ 4.9 ทดสอบการตรวจจับ 3 มิติ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะที่ต้องการให้ตรวจจับและหยุด (เมตร)	ระยะที่โมเดลตรวจจับได้ (เมตร)	ความเร็วหุ่นยนต์ ขณะทดสอบ (km/hr)
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.105	1.82049
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.1525	1.85062
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.0175	1.7855
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.04	2.13594
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.08	2.1721
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.03	2.18434
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.0425	2.04205
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.0475	2.22194
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	6.0475	2.18434
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.0025	2.19577
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	4.015	2.18434
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	5.2975	2.14709

ตารางที่ 4.9 ทดสอบการตรวจจับ 3 มิติ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะที่ต้องการให้ตรวจจับและหยุด (เมตร)	ระยะที่โมเดลตรวจจับได้ (เมตร)	ความเร็วหุ่นยนต์ ขณะทดสอบ (km/hr)
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	2	2.015	2.16849
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	4	3.98	2.20685
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	6	5.685	2.26895

ในการทดสอบตามตารางที่ 4.9 ได้ทำการประเมินความสามารถของโมเดลในการตรวจจับวัตถุแบบสามมิติ โดยแบ่งวัตถุที่ใช้ทดสอบออกเป็น 4 ประเภท ได้แก่ กรวย คน คนขี่จักรยาน และรถ ซึ่งครอบคลุมทั้งวัตถุขนาดเล็ก ขนาดกลาง และขนาดใหญ่ เพื่อสะท้อนการใช้งานจริงในสภาพแวดล้อมที่ซับซ้อน หุ่นยนต์ถูกตั้งค่าให้เคลื่อนที่เข้าหาวัตถุที่วางไว้ด้านหน้าในระยะ 2, 4 และ 6 เมตร ผลลัพธ์จากการตรวจจับแสดงให้เห็นว่าระบบสามารถตรวจจับวัตถุได้อย่างถูกต้องและใกล้เคียงกับระยะจริงที่กำหนด เช่น การตรวจจับกรวยที่ระยะ 2 เมตรตรวจจับได้จริงที่ 2.1775 เมตร และการตรวจจับคนที่ระยะ 4 เมตรตรวจจับได้จริงที่ 4.2275 เมตร แสดงให้เห็นว่าความคลาดเคลื่อนจากค่าจริงอยู่ในช่วงเพียงไม่กี่เซนติเมตร ซึ่งอยู่ในเกณฑ์ความแม่นยำของเซ็นเซอร์ LiDAR

เมื่อพิจารณารายละเอียดเชิงเปรียบเทียบ พบว่าวัตถุขนาดเล็ก เช่น กรวย สามารถตรวจจับได้เร็วและใกล้เคียงกับระยะจริงมากกว่าวัตถุขนาดใหญ่ เนื่องจากพอยต์คลาวด์ของวัตถุขนาดเล็กมีความหนาแน่นและชัดเจนในพื้นที่จำกัด ทำให้โมเดลสามารถสร้าง Bounding Box ได้แม่นยำแม้ในระยะที่ไม่ใกล้มากนัก ขณะที่วัตถุขนาดใหญ่ เช่น รถ หรือคนขี่จักรยาน ต้องอาศัยการเข้าใกล้มากขึ้นเพื่อให้ข้อมูลพอยต์คลาวด์มีความสมบูรณ์และตรงกับรูปแบบที่โมเดลได้เรียนรู้ ส่งผลให้ค่าระยะตรวจจับจริงอาจเบี่ยงเบนจากค่าที่กำหนดมากกว่าเล็กน้อย อย่างไรก็ตาม ความคลาดเคลื่อนเหล่านี้ยังคงอยู่ในเกณฑ์ที่ยอมรับได้ และไม่ส่งผลต่อความสามารถโดยรวมของระบบในการตรวจจับวัตถุ

นอกจากนี้ การทดสอบยังควบคุมความเร็วเฉลี่ยของหุ่นยนต์ให้อยู่ในช่วง 1.26–1.34 กิโลเมตรต่อชั่วโมง ซึ่งเป็นค่าที่เหมาะสมสำหรับการประเมินความแม่นยำของการตรวจจับในสภาวะที่ปลอดภัย ความเร็วในระดับนี้ช่วยลดผลกระทบจากการเคลื่อนไหวต่อการรับข้อมูลพอยต์คลาวด์และเพิ่มความน่าเชื่อถือของผลการทดลอง โดยสรุป ผลการทดสอบในตารางที่ 4.9 ยืนยันว่าโมเดล

สามารถตรวจจับและประเมินตำแหน่งของวัตถุได้อย่างถูกต้องในทุกประเภทและทุกระยะที่กำหนด อย่างไรก็ตาม การตรวจจับเพียงอย่างเดียวยังไม่เพียงพอต่อการรับรองความปลอดภัยของระบบ จึงจำเป็นต้องทดสอบขั้นต่อไป คือการตอบสนองของหุ่นยนต์ในการหยุดเมื่อเข้าใกล้วัตถุ ซึ่งได้นำเสนอไว้ในตารางที่ 4.10

ตารางที่ 4.10 ทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะห่างจากวัตถุ (เมตร)	เวลาตอบสนองของหุ่นยนต์ (วินาที)	ระยะเบรก (เมตร)	สถานะการหยุดของหุ่นยนต์	ความเร็วหุ่นยนต์ขณะทดสอบ (km/hr)
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.94	0.000005	0.06	สำเร็จ	1.31
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.915	0.000010	0.085	สำเร็จ	1.34
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.9875	0.000005	0.0125	สำเร็จ	1.28
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.8625	0.000009	0.1375	สำเร็จ	1.29
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.91	0.000005	0.09	สำเร็จ	1.31
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.7325	0.000005	0.2675	สำเร็จ	1.28
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.7425	0.000005	0.2575	สำเร็จ	1.27
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.77	0.000006	0.23	สำเร็จ	1.29
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.7775	0.000008	0.2225	สำเร็จ	1.27
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.8075	0.000004	0.1925	สำเร็จ	1.26

ตารางที่ 4.10 ทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะห่างจากวัตถุ (เมตร)	เวลาตอบสนองของหุ่นยนต์ (วินาที)	ระยะเบรก (เมตร)	สถานะการหยุดของหุ่นยนต์	ความเร็วหุ่นยนต์ขณะทดสอบ (km/hr)
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.8275	0.000004	0.1725	สำเร็จ	1.28
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.19	0.000005	0.81	สำเร็จ	1.28
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.86625	0.000004	0.13375	สำเร็จ	1.82225
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.8025	0.000006	0.1975	สำเร็จ	1.83795
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.9775	0.000004	0.0225	สำเร็จ	1.86162
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.67	0.000006	0.33	สำเร็จ	1.8182
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.67625	0.000005	0.32375	สำเร็จ	1.79854
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.675	0.000005	0.325	สำเร็จ	1.8419
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.8775	0.000006	0.1225	สำเร็จ	1.86714
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.7825	0.000005	0.2175	สำเร็จ	1.81934
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.2625	0.000005	0.7375	สำเร็จ	1.93599
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.84125	0.000005	0.15875	สำเร็จ	1.82049

ตารางที่ 4.10 ทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะห่างจากวัตถุ (เมตร)	เวลาตอบสนองของหุ่นยนต์ (วินาที)	ระยะเบรก (เมตร)	สถานะการหยุดของหุ่นยนต์	ความเร็วหุ่นยนต์ขณะทดสอบ (km/hr)
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.7625	0.000003	0.2375	สำเร็จ	1.85062
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.79625	0.000004	0.20375	สำเร็จ	1.7855
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.75875	0.000008	0.24125	สำเร็จ	2.13594
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.73875	0.000007	0.26125	สำเร็จ	2.1721
กรวย	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.78625	0.000005	0.21375	สำเร็จ	2.18434
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.56	0.000005	0.44	สำเร็จ	2.04205
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.55125	0.000004	0.44875	สำเร็จ	2.22194
คน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.59	0.000006	0.41	สำเร็จ	2.18434
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.74875	0.000010	0.25125	สำเร็จ	2.19577
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.785	0.000005	0.215	สำเร็จ	2.18434
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.07	0.000007	0.93	สำเร็จ	2.14709
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	1.6475	0.000004	0.3525	สำเร็จ	2.16849

ตารางที่ 4.10 ทดสอบการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ (ต่อ)

ประเภทของวัตถุที่ตรวจจับ	สถานการณ์ทดสอบ	ระยะห่างจากวัตถุ (เมตร)	เวลาตอบสนองของหุ่นยนต์ (วินาที)	ระยะเบรก (เมตร)	สถานะการหยุดของหุ่นยนต์	ความเร็วหุ่นยนต์ขณะทดสอบ (km/hr)
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.54875	0.000004	0.45125	สำเร็จ	2.20685
รถ	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	5.43625	0.000006	0.56375	สำเร็จ	2.26895
คนขี่จักรยาน	วิ่งเข้าวัตถุที่อยู่นิ่งด้านหน้า	3.785	0.000005	0.215	สำเร็จ	2.18434

ในการทดสอบตามตารางที่ 4.10 ได้ทำการประเมินประสิทธิภาพของระบบในการหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้ โดยหุ่นยนต์อัตโนมัติถูกตั้งค่าให้เคลื่อนที่ไปข้างหน้าด้วยความเร็วคงที่และหยุดเมื่อมีการตรวจจับวัตถุในระยะที่กำหนด ผลการทดสอบพบว่าระบบสามารถตอบสนองได้ทันเวลาในทุกกรณี โดยวัตถุที่ใช้ทดสอบประกอบด้วยกรวย คน คนขี่จักรยาน และรถ ซึ่งเป็นการครอบคลุมทั้งวัตถุขนาดเล็ก ขนาดกลาง และขนาดใหญ่ เพื่อให้เห็นประสิทธิภาพของระบบอย่างรอบด้าน การหยุดของหุ่นยนต์สอดคล้องกับการตรวจจับจากเซ็นเซอร์ LiDAR โดยสามารถชะลอและหยุดได้ก่อนที่จะเกิดการชนจริง

เมื่อพิจารณารายละเอียดพบว่า วัตถุขนาดเล็ก เช่น กรวย สามารถตรวจจับและหยุดได้ใกล้เคียงกับระยะที่กำหนดมากที่สุด เนื่องจากมีลักษณะพอยต์คลาวด์ที่ชัดเจนตั้งแต่ระยะไกล ขณะที่วัตถุขนาดใหญ่ เช่น รถ หรือคนขี่จักรยาน พบว่ามีความคลาดเคลื่อนเพิ่มขึ้นเล็กน้อย เนื่องจากการสร้าง Bounding Box มีขนาดใหญ่ ทำให้การจับรูปร่างที่สมบูรณ์ของวัตถุต้องใช้ระยะเข้าใกล้มากกว่า อย่างไรก็ตาม ความคลาดเคลื่อนที่เกิดขึ้นยังคงอยู่ในระดับที่ยอมรับได้ และไม่ส่งผลต่อความสามารถของระบบในการหลีกเลี่ยงการชน โดยระบบสามารถหยุดได้ก่อนถึงวัตถุในทุกกรณี ซึ่งเป็นสิ่งสำคัญในการใช้งานจริง

โดยสรุป ผลการทดสอบตารางที่ 4.10 แสดงให้เห็นว่าระบบตรวจจับและหยุดหุ่นยนต์มีความน่าเชื่อถือสูง สามารถตอบสนองต่อวัตถุหลากหลายประเภทได้อย่างถูกต้องและปลอดภัย แม้จะมีความคลาดเคลื่อนเล็กน้อยในกรณีวัตถุขนาดใหญ่ แต่ผลการทำงานโดยรวมยังอยู่ในเกณฑ์ที่เหมาะสมต่อการใช้งานจริงในสภาพแวดล้อมที่ซับซ้อน ทั้งนี้ยังสะท้อนให้เห็นถึงความสอดคล้องของโมเดลที่

พัฒนาขึ้นกับเซ็นเซอร์ LiDAR ที่ใช้ และความพร้อมที่จะประยุกต์ในระบบหุ่นยนต์อัตโนมัติสำหรับการนำทางและการหลีกเลี่ยงการชน

บทที่ 5
สรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้มุ่งเน้นการพัฒนาโมเดลตรวจจับวัตถุสามมิติจากข้อมูล LiDAR ที่มีประสิทธิภาพสูงและสามารถทำงานได้บนอุปกรณ์ประมวลผลที่มีข้อจำกัดด้านทรัพยากร เช่น หุ่นยนต์อัตโนมัติขนาดเล็ก โดยมีการปรับปรุงสถาปัตยกรรมของโมเดล PointPillars ทั้งในด้านโครงสร้างเครือข่าย, การจัดการข้อมูลพอยต์คลาวด์ และการเลือกใช้เทคนิคด้าน Attention เพื่อเพิ่มความสามารถในการตรวจจับและลดภาระการคำนวณ ในหัวข้อนี้จะนำเสนอผลการทดสอบที่ได้จากการประเมินระบบใน 3 ด้านหลัก ได้แก่ ประสิทธิภาพของระบบตรวจจับวัตถุสามมิติ ภาระการประมวลผลบนหุ่นยนต์ และความสามารถในการหลีกเลี่ยงการชน รวมถึงข้อเสนอแนะเพื่อการต่อยอดในอนาคต

5.1.1 ผลการทดสอบระบบตรวจจับวัตถุ 3 มิติ

ผลการประเมินประสิทธิภาพของโมเดลตรวจจับวัตถุสามมิติที่ได้รับการปรับปรุงแสดงให้เห็นถึงความก้าวหน้าอย่างชัดเจนทั้งในด้านความแม่นยำและประสิทธิภาพเชิงเวลา โดยมีการปรับขอบเขตของพอยต์คลาวด์ (Point Cloud Range) ให้สอดคล้องกับระยะตรวจจับด้านหน้าของหุ่นยนต์ และการปรับขนาดของวอกเซล (Voxel Size) ให้มีความละเอียดมากขึ้น เพื่อเพิ่มความสามารถในการตรวจจับวัตถุขนาดเล็กในระยะใกล้ได้อย่างแม่นยำ

นอกจากนี้ งานวิจัยยังได้ปรับเปลี่ยนโครงสร้าง Backbone ของ PointPillars โดยแทนที่โครงข่ายคอนโวลูชันแบบเดิมด้วย Lightweight Attention Pyramid Network (LAPN) ซึ่งออกแบบมาเพื่อเพิ่มประสิทธิภาพการสกัดคุณลักษณะแบบหลายสเกล (Multi-scale Feature Extraction) ในขณะที่ยังคงความเบาในการคำนวณ LAPN ใช้กลไกความสนใจ (Attention Mechanism) ในการเน้นเฉพาะส่วนสำคัญของฟีเจอร์ที่เกี่ยวข้องกับการตรวจจับวัตถุโดยตรง ส่งผลให้โมเดลสามารถมุ่งเน้นไปยังบริเวณสำคัญและลดผลกระทบจากข้อมูลรบกวน (irrelevant features) ได้อย่างมีประสิทธิภาพ ซึ่งส่งผลต่อความแม่นยำที่เพิ่มขึ้น

จากผลการฝึกสอน พบว่าโมเดลสามารถลดค่าความสูญเสีย (Loss) ได้อย่างต่อเนื่องในแต่ละ epoch จนกระทั่งเข้าสู่ค่าที่ต่ำมากในช่วงท้ายของการฝึก ซึ่งสะท้อนถึงเสถียรภาพของกระบวนการเรียนรู้และความสามารถในการจำแนกลักษณะของวัตถุจากข้อมูลพอยต์คลาวด์ได้อย่างแม่นยำ

การประเมินค่า Mean Average Precision (mAP) ที่ค่า IoU 0.5 พบว่าโมเดลที่ได้รับการปรับปรุงมีค่า mAP รวม 0.7883 โดยเฉพาะคลาส “cyclist” และ “cone” ที่มีค่า AP สูงถึง 0.8856 และ 0.9377 ตามลำดับ ส่วนคลาส “human” และ “car” มีค่า AP เท่ากับ 0.7024 และ 0.6275 ตามลำดับ ซึ่งอยู่ในระดับที่เพียงพอต่อการใช้งานจริง จุดเด่นของสถาปัตยกรรมที่ปรับปรุงด้วย LAPN ไม่ได้ทำให้ความแม่นยำเพิ่มขึ้นอย่างมีนัยสำคัญ แต่ช่วยให้โมเดลสามารถคงความแม่นยำได้ในระดับสูง แม้จะลดจำนวนการคำนวณและความซับซ้อนของโครงสร้างลง ซึ่งถือว่าเป็นประโยชน์อย่างยิ่งในการประยุกต์ใช้กับอุปกรณ์ฝังตัวที่มีข้อจำกัดด้านทรัพยากร

ในด้านการวิเคราะห์จาก Confusion Matrix พบว่าระบบสามารถจำแนกวัตถุได้อย่างแม่นยำ โดยมีค่า Precision โดยรวม 0.8301 และค่า Recall 0.7195 ขณะที่ค่า F1 Score อยู่ที่ 0.7692 ซึ่งถือว่ามีความสมดุลระหว่างการตรวจพบและความถูกต้องของการทำนาย อีกทั้งยังพบว่าอัตราการตรวจไม่พบวัตถุ (False Negative) อยู่ในระดับต่ำและยอมรับได้ แม้ในกรณีที่ข้อมูลพอยต์คลาวด์มีความเบาบาง โมเดลยังคงสามารถดำเนินการทำนายได้อย่างน่าเชื่อถือ สะท้อนถึงความสามารถในการใช้งานจริงในสภาพแวดล้อมที่มีความไม่แน่นอนสูง

นอกจากนี้ ยังได้มีการประเมินประสิทธิภาพด้าน เวลาในการทำนาย (Inference Time) ในภายใต้สถานการณ์ที่การรันเฉพาะโมเดลโดยไม่รวมระบบขับเคลื่อนอัตโนมัติโดยเปรียบเทียบระหว่างโมเดลดั้งเดิม (Default Model) และโมเดลที่ได้รับการปรับปรุง (Enhancing Model) ดังแสดง ในตารางที่ 4.7 แสดงผลการประเมินประสิทธิภาพของโมเดลในขั้นตอนการทำนายภายใต้การรันเฉพาะโมเดลโดยไม่รวมระบบขับเคลื่อนอัตโนมัติ พบว่าโมเดลที่ได้รับการปรับปรุงให้สามารถเพิ่มช่วงการรับข้อมูล (Enhancing range) และเพิ่มกลไกการให้ความสนใจ (Attention network) มีประสิทธิภาพสูงกว่าโมเดลดั้งเดิม (Default Model) อย่างชัดเจน โดยหน่วยประมวลผล Jetson Xavier มีค่า FPS เพิ่มจาก 12.05 ในโมเดลดั้งเดิมเป็น 38.72 ในโมเดลที่ได้รับการปรับปรุงและลดเวลาในการทำนายจาก 0.083 วินาที เหลือเพียง 0.026 วินาทีต่อเฟรม คิดเป็นการลดเวลาได้ถึง 68.67%

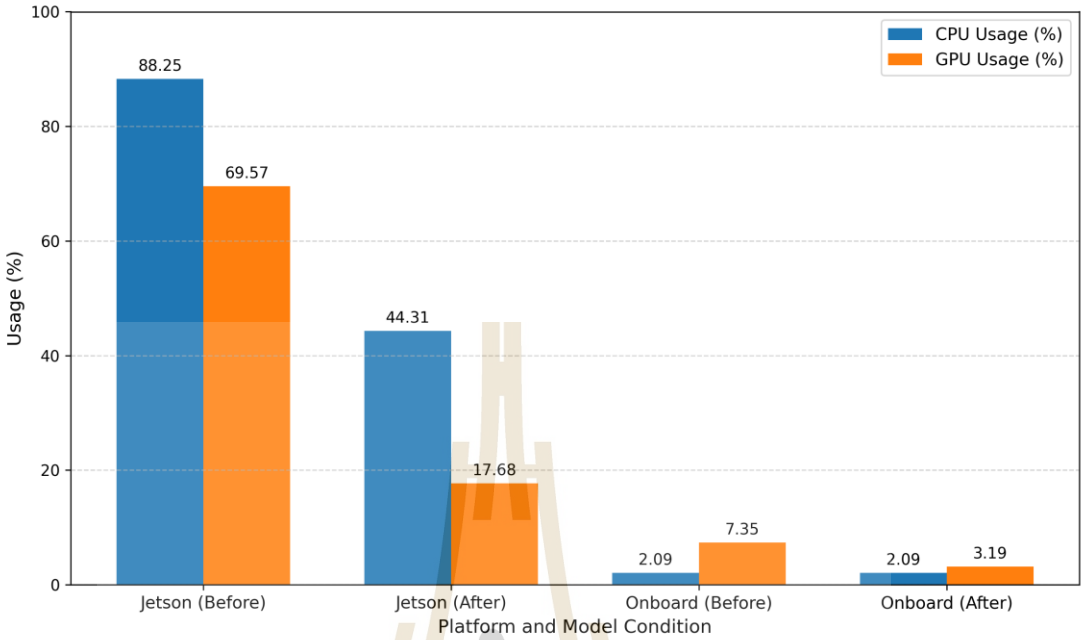
ในส่วนของคอมพิวเตอร์บนรถกอล์ฟ (วงศธร, 2565) ซึ่งมีประสิทธิภาพด้านการประมวลผลสูงกว่า พบว่าค่า FPS สูงกว่า Jetson Xavier อย่างชัดเจน โดยโมเดลดั้งเดิมมีค่า FPS เท่ากับ 87.11 และลดเวลาในการทำนายลงเหลือ 0.011 วินาที ขณะที่โมเดล Enhancing range และ Enhancing range with attention network ให้ค่า FPS สูงสุดที่ 116.89 และลดเวลาในการทำนายได้ถึง 18.18% ซึ่งสะท้อนให้เห็นว่าการปรับปรุงโครงสร้างของโมเดลช่วยเพิ่มประสิทธิภาพเชิงเวลาได้ทั้งในอุปกรณ์แบบฝังตัวและคอมพิวเตอร์ประสิทธิภาพสูง

ในตารางที่ 4.8 พบว่าในสถานการณ์โมเดลที่ปรับปรุง ทำงานร่วมกับระบบอัตโนมัติแบบเต็มรูปแบบ สามารถลดเวลาในการประมวลผลได้อย่างมีนัยสำคัญ จากค่าเฉลี่ย FPS ที่เพิ่มขึ้นจาก 9.05 เป็น 32.35 และเวลาในการทำนายที่ลดลงจาก 0.111 วินาที เหลือเพียง 0.031 วินาทีต่อเฟรม หรือคิดเป็นการลดเวลาถึง 72.07% ซึ่งส่งผลโดยตรงต่อการตอบสนองของระบบแบบเรียลไทม์และช่วยลดภาระการประมวลผลบนอุปกรณ์ฝังตัว (Edge Device) ได้อย่างชัดเจน

โดยสรุป ผลการทดสอบทั้งหมดแสดงให้เห็นว่าโมเดลตรวจจับที่ได้รับการปรับปรุงมีประสิทธิภาพสูงขึ้นทั้งในด้านความแม่นยำ ความเร็วในการทำงาน และความสามารถในการจัดการทรัพยากรของฮาร์ดแวร์อย่างมีประสิทธิภาพ ทำให้เหมาะสมอย่างยิ่งสำหรับการนำไปประยุกต์ใช้ในหุ่นยนต์อัตโนมัติที่ต้องการการตรวจจับแบบเรียลไทม์ภายใต้ข้อจำกัดของทรัพยากร

5.1.2 ผลการทดสอบภาระการคำนวณบนหุ่นยนต์

ผลการประเมินภาระการประมวลผลของโมเดล PointPillars บนแพลตฟอร์ม Jetson Xavier AGX และ คอมพิวเตอร์ประมวลผลบนรถกอล์ฟอัตโนมัติ (Onboard Computer) พบว่า โมเดลเวอร์ชันดั้งเดิมไม่สามารถทำงานร่วมกับระบบนำทางอัตโนมัติเต็มรูปแบบได้อย่างมีเสถียรภาพ โดยเฉพาะเมื่อรันบน Edge Device ซึ่งมีข้อจำกัดด้านทรัพยากร ทำให้เกิดภาระการใช้งาน CPU สูงจนเกือบเต็มขีดความสามารถของฮาร์ดแวร์ และมีแนวโน้มเกิดความล่าช้าในการตอบสนองแบบเรียลไทม์ อย่างไรก็ตาม หลังจากมีการปรับปรุงโครงสร้างของโมเดลให้เหมาะสมกับสภาพแวดล้อมการทำงานบนอุปกรณ์ประมวลผลประสิทธิภาพต่ำ โดยดำเนินการ ลดขอบเขตของข้อมูลพอยต์คลาวด์ (Point Cloud Range), ปรับขนาดเวกเซล (Voxel Size) ให้เหมาะสม และ เสริมด้วยโครงข่าย Lightweight Attention Pyramid Network (LAPN) เพื่อเพิ่มประสิทธิภาพในการสกัดคุณลักษณะและลดภาระคำนวณ พบว่าโมเดลที่ได้รับการปรับปรุงสามารถทำงานได้อย่างต่อเนื่องและมีเสถียรภาพมากขึ้น ทั้งบน Edge Device และ Onboard Computer การวิเคราะห์จากเครื่องมือ Jetson Stats (jtop) แสดงให้เห็นว่าการใช้งาน CPU และ GPU ของแต่ละแพลตฟอร์มมีความแตกต่างกันอย่างชัดเจน ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 แสดงการเปรียบเทียบค่าเฉลี่ยการใช้งานของ CPU และ GPU ระหว่างโมเดล PointPillars ก่อนและหลังการปรับปรุงบนแพลตฟอร์ม Jetson Xavier และ Onboard Computer

ตารางที่ 5.1 สรุปเปอร์เซ็นต์การลดลงของการใช้งาน CPU และ GPU หลังการปรับปรุงโมเดล PointPillars

แพลตฟอร์ม (Platform)	ประเภทการใช้งาน (Usage Type)	ก่อนปรับปรุง (%)	หลังปรับปรุง (%)	ลดลงจริง (หน่วย %)	ลดลงเมื่อเทียบกับเดิม (%)
Jetson Xavier	CPU	88.25	44.31	43.94	49.79
	GPU	69.57	17.68	51.89	74.58
Onboard Computer	CPU	2.09	2.09	0.00	0.00
	GPU	7.35	3.19	4.16	56.60

จากรูปที่ 5.1 และตารางที่ 5.1 แสดงให้เห็นว่า หลังการปรับปรุงโมเดล PointPillars มีการเปลี่ยนแปลงอย่างชัดเจนในด้านการใช้งานทรัพยากรของหน่วยประมวลผล โดยในกรณีของ Jetson Xavier พบว่าค่าเฉลี่ยการใช้งาน CPU ลดลงจาก 88.25% เหลือ 44.31% และ GPU ลดลงจาก 69.57% เหลือ 17.68% ซึ่งคิดเป็นการลดลงของภาระการประมวลผล CPU ประมาณ 49.79%

และ GPU ประมาณ 74.58% เมื่อเทียบกับเวอร์ชันก่อนการปรับปรุง ผลดังกล่าวสะท้อนให้เห็นว่าโมเดลที่ได้รับการปรับปรุงมีประสิทธิภาพมากขึ้น สามารถจัดสรรทรัพยากรการประมวลผลได้เหมาะสมและมีเสถียรภาพสูงขึ้น

สำหรับ Onboard Computer ของรถกอล์ฟอัตโนมัติ พบว่าก่อนการปรับปรุงโมเดล มีค่าเฉลี่ยการใช้งาน CPU อยู่ที่ 2.09% และ GPU อยู่ที่ 7.35% หลังจากการปรับปรุง ค่าเฉลี่ยการใช้งาน CPU ยังคงเท่าเดิมที่ 2.09% ขณะที่ GPU ลดลงเหลือเพียง 3.19% หรือคิดเป็นการลดลงประมาณ 56.60% แสดงให้เห็นว่าโมเดลหลังการปรับปรุงสามารถทำงานได้อย่างมีเสถียรภาพมากขึ้น ใช้พลังงานและทรัพยากรลดลง แต่ยังคงมีประสิทธิภาพเพียงพอต่อการประมวลผลแบบเรียลไทม์

โดยสรุป การปรับปรุงโมเดล PointPillars ส่งผลให้ทั้งแพลตฟอร์ม Jetson Xavier และ Onboard Computer มีการจัดสรรทรัพยากรการประมวลผลที่มีประสิทธิภาพมากขึ้น โดยเฉพาะในส่วนของ GPU ซึ่งลดภาระลงได้อย่างชัดเจน ช่วยให้ระบบสามารถทำงานได้ต่อเนื่องโดยไม่เกิดความหน่วงหรือการค้างของกระบวนการประมวลผล

จากผลการทดสอบทั้งหมดสามารถสรุปได้ว่า การปรับปรุงโมเดล PointPillars ช่วยเพิ่มประสิทธิภาพการประมวลผลทั้งบน Edge Device (Jetson Xavier AGX) และ Onboard Computer ของรถกอล์ฟอัตโนมัติ ได้อย่างชัดเจน โดยช่วยลดภาระการใช้งานทรัพยากร GPU ได้อย่างมีนัยสำคัญ ขณะเดียวกันยังคงรักษาความเสถียรของระบบและความต่อเนื่องในการทำงาน ผลลัพธ์นี้ยืนยันว่าโมเดลที่ได้รับการปรับแต่งมีความเหมาะสมต่อการนำไปใช้งานจริงในระบบนำทางอัตโนมัติหรือหุ่นยนต์ขนาดเล็กที่มีข้อจำกัดด้านทรัพยากร และเป็นตัวอย่างที่ชัดเจนของประสิทธิภาพแนวคิด Edge Computing ที่ช่วยให้การประมวลผลแบบเรียลไทม์สามารถทำงานได้โดยไม่ต้องพึ่งพาฮาร์ดแวร์ประสิทธิภาพสูงเพิ่มเติม

5.1.3 ผลการทดสอบการวิเคราะห์เพื่อหลีกเลี่ยงการชนกับวัตถุที่ตรวจจับได้

ผลการทดสอบระบบหลีกเลี่ยงการชนในหุ่นยนต์อัตโนมัติแสดงให้เห็นว่าโมเดลตรวจจับวัตถุสามมิติที่ได้รับการปรับปรุงสามารถประมวลผลข้อมูลจาก LiDAR ได้อย่างแม่นยำ และตอบสนองต่อสถานการณ์ที่เสี่ยงต่อการชนได้อย่างมีประสิทธิภาพ โดยหุ่นยนต์สามารถตรวจจับวัตถุในระยะที่ปลอดภัยและหยุดเคลื่อนที่ได้ก่อนถึงวัตถุในทุกกรณีที่ทดสอบ ไม่ว่าจะเป็นวัตถุขนาดเล็ก เช่น กรวยจราจร หรือวัตถุขนาดใหญ่ เช่น รถ

จากการเปรียบเทียบระยะที่โมเดลตรวจจับได้ (จากข้อมูล Point Cloud ของ LiDAR) กับระยะที่วัดจริงจากพื้นทดสอบหลังหุ่นยนต์หยุด พบว่าความคลาดเคลื่อนอยู่ในช่วงที่ต่ำมาก (ประมาณ 2–4 เซนติเมตร) โดยเฉพาะในกรณีวัตถุขนาดเล็กและขนาดกลาง เช่น กรวยและคน ซึ่งสามารถหยุดได้ในตำแหน่งที่ใกล้เคียงกับค่าที่กำหนดไว้ล่วงหน้า ส่วนในกรณีวัตถุขนาดใหญ่ เช่น รถ

และจักรยาน ความคลาดเคลื่อนมีแนวโน้มเพิ่มขึ้นเล็กน้อย อันเนื่องมาจากลักษณะของพอยต์คลาวด์ที่เบาบางในระยะไกล และรูปทรง Bounding Box ที่ใหญ่ขึ้นซึ่งส่งผลต่อระยะตรวจจับ

การทดลองนี้แสดงให้เห็นว่าระบบสามารถตัดสินใจหยุดได้อย่างมีประสิทธิภาพ โดยใช้เพียงข้อมูลจาก LiDAR โดยไม่ต้องพึ่งกล้องหรือเซ็นเซอร์อื่นเพิ่มเติม อีกทั้งยังยืนยันความแม่นยำของการตรวจจับตำแหน่งวัตถุในพื้นที่ทดสอบจริง ความสามารถนี้จึงมีความสำคัญอย่างยิ่งต่อการใช้งานในระบบหุ่นยนต์อัตโนมัติที่ต้องการความปลอดภัยสูง เช่น หุ่นยนต์ส่งของ หรือยานพาหนะไร้คนขับในพื้นที่จำกัด

5.1.4 การประยุกต์ใช้ในระบบอัตโนมัติ

จากผลการทดสอบในหลายด้าน ไม่ว่าจะเป็นประสิทธิภาพของการตรวจจับวัตถุสามมิติ ความสามารถในการทำงานร่วมกับระบบควบคุมอัตโนมัติ และความแม่นยำในการหลีกเลี่ยงการชน แสดงให้เห็นว่าโมเดล PointPillars ที่ได้รับการปรับปรุงสามารถนำไปใช้งานจริงในระบบหุ่นยนต์อัตโนมัติได้อย่างมีประสิทธิภาพ แม้จะประมวลผลบนฮาร์ดแวร์ที่มีข้อจำกัดด้านทรัพยากร เช่น Jetson Xavier AGX

โมเดลที่ปรับปรุงแล้วสามารถตรวจจับวัตถุในระยะใกล้ได้อย่างแม่นยำ โดยเฉพาะในบริบทของหุ่นยนต์ที่เคลื่อนที่ในพื้นที่จำกัด เช่น ภายในอาคาร โดยสามารถหยุดการเคลื่อนที่ได้อย่างปลอดภัยเมื่อมีวัตถุอยู่ด้านหน้า อีกทั้งยังสามารถรันพร้อมกับระบบนำทางอัตโนมัติแบบเรียลไทม์โดยไม่เกิดปัญหาด้านประสิทธิภาพ

ดังนั้น งานวิจัยนี้แสดงให้เห็นถึงความเป็นไปได้และความเหมาะสมของการนำสถาปัตยกรรม PointPillars ที่ผ่านการปรับแต่ง ไปประยุกต์ใช้งานในระบบหุ่นยนต์อัตโนมัติ โดยเฉพาะในงานที่ต้องการระบบตรวจจับวัตถุที่แม่นยำ ปลอดภัย และสามารถดำเนินการได้บนแพลตฟอร์มประมวลผลขนาดเล็กอย่างมีประสิทธิภาพ

5.2 ข้อเสนอแนะ

จากผลการวิจัยที่ได้ดำเนินการในครั้งนี้ พบว่าแม้โมเดล PointPillars ที่ได้รับการปรับปรุงสามารถตรวจจับวัตถุสามมิติได้อย่างมีประสิทธิภาพบนอุปกรณ์ที่มีข้อจำกัดด้านการประมวลผล แต่ยังมีประเด็นบางประการที่ควรได้รับการพิจารณาเพิ่มเติมในงานวิจัยต่อไป เพื่อพัฒนาและขยายขอบเขตการใช้งานให้กว้างขึ้นและตอบสนองความต้องการในสภาพแวดล้อมที่ซับซ้อนมากยิ่งขึ้น ดังนี้

5.2.1 การปรับขยายขอบเขตการตรวจจับ

แม้การลดระยะของ Point Cloud Range จะช่วยลดภาระการคำนวณ แต่การจำกัดขอบเขตมากเกินไปอาจทำให้สูญเสียข้อมูลสำคัญในบางสถานการณ์ การวิจัยในอนาคตควรพิจารณาแนวทางที่ยืดหยุ่น เช่น การปรับช่วงตรวจจับแบบไดนามิกตามบริบทหรือพฤติกรรมของวัตถุ

5.2.2 การทดลองกับเซ็นเซอร์หลากหลายประเภท

งานวิจัยนี้ใช้ข้อมูลจาก LiDAR เพียงอย่างเดียว การผสานข้อมูลจากเซ็นเซอร์อื่น เช่น กล้อง RGB หรือกล้องความลึก (Depth Camera) อาจช่วยเพิ่มความแม่นยำในการตรวจจับ โดยเฉพาะในพื้นที่ที่มีโครงสร้างซับซ้อน

5.2.3 การปรับใช้เทคนิคลดขนาดโมเดลเพิ่มเติม

แม้จะใช้ Lightweight Attention Pyramid Network แล้ว แต่ยังสามารถลดภาระการประมวลผลเพิ่มเติมได้อีก เช่น การใช้การควอนไทเซชัน (Quantization), การปรับโมเดลให้เป็นแบบ INT8 หรือการใช้โมเดลขนาดเล็กอื่น เช่น PointNet++, PV-RCNN Lite

5.2.4 การทดลองในสภาพแวดล้อมที่มีความซับซ้อนมากขึ้น

การทดสอบในงานวิจัยนี้ดำเนินการในสถานการณ์ที่ควบคุมได้ หากนำไปประยุกต์ใช้ในสภาพแวดล้อมจริง เช่น พื้นที่กลางแจ้งที่มีคนจำนวนมากหรือสิ่งกีดขวางไม่แน่นอน ควรมีการทดสอบซ้ำในเงื่อนไขที่หลากหลายมากขึ้น เพื่อยืนยันความเสถียรของระบบ

5.2.5 การจัดสรรทรัพยากรของระบบให้เหมาะสม

ควรมีการศึกษาวิธีการจัดลำดับความสำคัญของการประมวลผล เช่น การลดความถี่ของการตรวจจับในช่วงที่ไม่มีสิ่งกีดขวาง หรือการใช้หน่วยประมวลผลเฉพาะทาง (เช่น DLA หรือ TensorRT) เพื่อเพิ่มประสิทธิภาพโดยรวมของระบบ

รายการอ้างอิง

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. **(2019). Pointpillars: Fast encoders for object detection from point clouds.** In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition **(pp. 12697-12705).**

ปริญญา สงวนสัตย์. (2562). **Artificial Intelligence with Machine Learning: AI สร้างได้ด้วยแมชชีนเลิร์นนิ่ง.** กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). **Pointnet: Deep learning on point sets for 3d classification and segmentation.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).

Peri, N., Li, M., Wilson, B., Wang, Y. X., Hays, J., & Ramanan, D. (2023). **An empirical analysis of range for 3d object detection.** In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4074-4083).

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). **Feature pyramid networks for object detection.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

Zhang, J., Yan, Y., Cheng, Z., & Wang, W. (2020). **Lightweight attention pyramid network for object detection and instance segmentation.** Applied Sciences, 10(3), 883.

Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018, December). **Convolutional neural network (CNN) for image detection and recognition**. In 2018 first international conference on secure cyber computing and communication (ICSCCC) (pp. 278-282). IEEE.

Sager, C., Zschech, P., & Kühl, N. (2022). ***labelCloud: A Lightweight Labeling Tool for Domain-Agnostic 3D Object Detection in Point Clouds***. Computer-Aided Design and Applications, 19(6), 1191-1206. DOI: 10.14733/cadaps.2022.1191-1206

Rusu, R. B., & Cousins, S. (2011). 3D is here: **Point Cloud Library (PCL).** In 2011 IEEE International Conference on Robotics and Automation (pp. 1–4). IEEE.

Zhou, Q. Y., Park, J., & Koltun, V. (2018). **Open3D: A modern library for 3D data processing.** arXiv preprint arXiv:1801.09847.

Zhang, J., Singh, S. (2017). ***LOAM: Lidar Odometry and Mapping in Real-time.*** Robotics: Science and Systems Conference (RSS).

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). **PointCNN: Convolution On X-Transformed Points.** Advances in Neural Information Processing Systems (NeurIPS), 31.

Shi, S., Wang, Z., & Li, H. (2020). **PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection.** Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10529–10538.

Geiger, A., Lenz, P., & Urtasun, R. (2012). **Are we ready for autonomous driving? The KITTI vision benchmark suite.** In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361.

KITTI Vision Benchmark Suite. (2023). ***3D Object Detection Evaluation.*** [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_object.php

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. **SSD: Single shot multibox detector.** In ECCV, 2016.

Ren et al., **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**, NeurIPS 2015.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). **ROS: an open-source Robot Operating System.** In ICRA Workshop on Open Source Software (Vol. 3, No. 3.2, p. 5).

ROS Wiki. (2020a). **Melodic Morenia (May 2018 - May 2023).** Retrieved from https://wiki.ros.org/melodic

ROS Wiki. (2020b). **Noetic Ninjemys (May 2020 - May 2025).** Retrieved from https://wiki.ros.org/noetic

Velodyne LiDAR Inc. (2020). **VLP-16 User Manual and Specifications.** [Online]. Available: https://velodynelidar.com/

Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). **Multi-View 3D Object Detection Network for Autonomous Driving.** In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6526–6534.

Bonghi, R. (2020). **Jetson Stats - Monitor and control your NVIDIA Jetson.** GitHub Repository. Retrieved from https://github.com/rbonghi/jetson_stats

NVIDIA. (2020). **NVIDIA Jetson AGX Xavier Developer Kit: Product Brief.** Retrieved from https://developer.nvidia.com/embedded/jetson-agx-xavier

Powers, D. M. W. (2011). **Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation.** *Journal of Machine Learning Technologies*, 2(1), 37–63.

Sasaki, Y. (2007). **The truth of the F-measure.** Teach Tutor Mater, 1(5), 1–5.

วงศธร อ่างเข็ม. (**2565**). **การพัฒนารถกอล์ฟไฟฟ้าอัตโนมัติสำหรับขนส่งผู้โดยสารระหว่างอาคาร.** วิทยานิพนธ์ปริญญามหาบัณฑิต, สาขาวิศวกรรมเครื่องกลและกระบวนการ, มหาวิทยาลัยเทคโนโลยีสุรนารี.

Shi, G., Li, R., & Ma, C. (2022). **PillarNet: Real-time and high-performance pillar-based 3D object detection.** In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 35–52). Cham: Springer Nature Switzerland.

ภาคผนวก ก

ผลงานวิจัยที่ได้รับการเผยแพร่ระหว่างการศึกษา

รายชื่อบทความที่ได้รับการเผยแพร่ระหว่างการศึกษา

Phuangmalee, N., & Tantrairatn, S. (2024, November). **Enhancing 3D Object Detection for Edge Devices: A PointPillars Approach with Attention Networks and Range Adjustment**. In 2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 111-114). IEEE.

2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA)

Enhancing 3D Object Detection for Edge Devices: A PointPillars Approach with Attention Networks and Range Adjustment

Nattapong Phuangmalee
School of Mechatronics Engineering
Institute of Engineering
Suranaree University of Technology
Nakhon Ratchasima, Thailand
M6501211@g.sut.ac.th

Suradet Tantrairatn
School of Mechanical Engineering
Institute of Engineering
Suranaree University of Technology
Nakhon Ratchasima, Thailand
suradetj@sut.ac.th

2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA) | 979-8-3503-7725-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAICA63239.2024.10823075

***Abstract*—The advancement of 3D object detection on edge devices poses significant challenges, especially for autonomous robots and vehicles requiring computational efficiency within constrained hardware environments. This study addresses these challenges by proposing an optimized 3D detection framework that integrates the Jetson Xavier AGX platform with LiDAR sensors, enhancing the PointPillars model to balance accuracy with efficiency. The approach employs a Range Expert mechanism to dynamically adjust detection range and voxel size, reducing processing load while sustaining high precision. A Lightweight Attention Pyramid Network further improves feature extraction, enhancing the model's ability to detect objects in diverse scales and conditions. Evaluation experiments have demonstrated that the system can achieve a 3D object detection task within 10 meters with a frame rate of 23.39 FPS, a mean Average Precision (mAP) value of 0.969, and a recall value of 0.875. The results underscore advantages of the model in functionally relevant tasks such as navigation, collision avoidance, and path planning scaling a real-world 3D detection system on autonomous assets with limited resources.**

Keywords*—*PointPillars, 3D Detection, LiDAR

I. Introduction

The task of 3D object detection within the field of computer vision entails the identification and localization of objects within a three-dimensional environment, predicated upon their shape, spatial coordinates, and orientation. Diverging from 2D detection, which only provides object class without the positional or distance metrics. The 3D object detection is being used in autonomous vehicles or autonomous robot that requires the exact location of an object to perform safety navigation tasks.

Several techniques have been researched toward 3D object detection in this research domain. Li et al.[1] made use of a stereo camera configuration, where depth information contributed by two cameras was combined with 2D bounding box detection results toward 3D localization of an object. Wang et al. [2] also incorporated a depth camera together with an infrared light sensor to capture depth information and 2D object detection, resulting in full 3D object recognition. Liu et al. [3], on the other hand, used a combined strategy using LiDAR and camera images and therefore used a depth range of LiDAR with 2D detection from a camera and therefore achieved 3D object detection. Still, practical factors like poor performance in scenarios with little or no light and high computational requirements have set a narrow focus toward the utilization of LiDAR only sensors that are more flexible and efficient.

This research endeavors to propose the novel method of 3D object detection under varied environmental conditions by employing LiDAR sensors exclusively. Capitalizing on the efficacy of the improved PointPillars [10] method. Notably, Peri et al. [4] introduced techniques for adjusting the detection range and voxel size to reduce the volume of point cloud data, thereby decreasing the computational time required for model processing. Moreover, Zhang et al., [7] utilized an attention layer inside a CNN in order to reduce the number of features, in the proposed research, machine learning processes are utilized to identify salient features and make predictions on the attributes of the objects. However, there are shortcomings still such as the computation burden posed by the Jetson Xavier AGX device which brings in the task of balancing the processing power requirements with the confidence accuracy metrics targets in 3D object detection.

Therefore, the current research looks into 3D objects detection, the use of machine learning optimization, and computational limitations in real time edge computing scenarios. Utilizing the Jetson Xavier AGX hardware for 3D point cloud data sensing, the study successfully combines PointPillars architecture with voxel scaling and distance modulation and a Lightweight Attention Pyramid Network, as illustrated in Figure 1. These approaches are used to enhance the real world implementations of such systems, improving the detection performance cost relationship in real autonomous systems.

II. Proposed Method

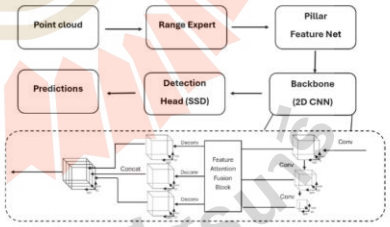


Fig. 1: Proposed method

A. PointPillars

The PointPillars method is essential in effective 3D detection because it allows the LiDAR point cloud to be represented as a 2D BEV image. Since this process minimizes the amount of point cloud data and allows for quicker computation, accuracy is not lost. After the point cloud transformation into 2D, Convolutional Neural Networks (CNNs) [9] are used for the important features extraction and additional Artificial Neural Networks (ANNs) are employed to classification and localization of the detected object.

The Pointpillar method comprises three primary processes:

- PointPillars Feature Net: This is a module that renders the point cloud data into a birds eye view and makes voxels which span the entire detection area. Features of each voxel are then collected using the PointNet[8] network guaranteeing that enough quality features are gathered for things to come forth.
- Backbone Network (2D CNN): A Feature Pyramid Network [6] with different layers of convolution neural networks works on the bird's eye transformed map and hence enables high quality feature detection around the object over many scales.
- Detection Head: The very last stage makes predictions about the 3–D bounding boxes and probable classes of objects using the extracted features.

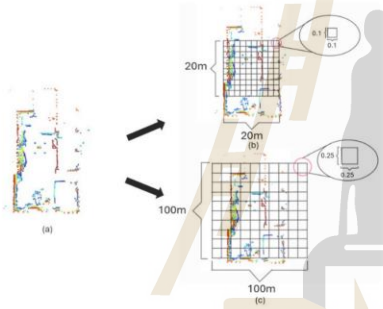


Fig. 2: Range expert

B. Range expert

Range expert [4] is optimize the model by adjusting both the detection range and voxel resolution during the configuration phase. These modifications allow for tailoring the model to meet specific application requirements, such as close-range collision avoidance or long-range situational awareness. By fine-tuning these parameters, the model's performance can be aligned with the operational needs of the system, improving both accuracy and computational efficiency

Figure 2(a) presents the point cloud data collected by the LiDAR sensor over the test area. For scenarios requiring detecting and avoiding an obstacle at close range, the voxel size should be no larger than 0.1 m as is illustrated in Figure 2(b). Because of the smaller scale limitation, it is possible to represent smaller parts of objects and their movement or position in a better way. Also, the range of interest is set to 20 m as shown in Figure 2(b) in order to make sure that most of the computation resources are utilized for the detection of the surroundings making the irrelevant areas to be non-essential.

In the case of far-field sensing task focused on large areas in the monitoring of the environment, voxel size of 0.25 is used as demonstrated in Figure 2(c) in order to maintain a lower spatial resolution which reduces computational load, especially in the near-field zones where accuracy is not necessary. Furthermore, the range of sensing has been increased to 100 meters, as seen in Figure 2(c), which enables the effective detection of objects from a distance and therefore increases the overall coverage area. The method achieves an ideal combination of spatial resolution and range which optimally enhances real time processing and wide area detection in resource constrained situations.

C. Lightweight attention pyramid network

The APN (attention pyramid network) is a specific structure of a neural network model created for tasks like instance segmentation and object detection. As an enhancement of classical pyramid networks, it contains attention mechanisms which allow for more effective feature exploitations across scale variations, as shown in Figure 3. The approach helps achieve better accuracies especially for objects of different scales by employing two components.

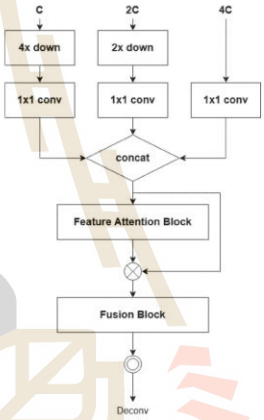


Fig. 3: Feature Attention Fusion Block.

- Feature Attention Block: This module applies attention mechanisms to highlight important feature channels and suppress irrelevant ones. The APN improves the effectiveness and efficiency of the network by bringing the most informative channels to the fore as required. As a result, the network is better able to accurately represent features, enabling it to perform better in detecting objects even in crowded environments.
- Feature Fusion: While the APN combines high level features that are more general semantic details with low level features that are more specific in their

spatial orientation. This method enables more effective detection by fostering retention of both low resolution and high resolution information. All these multiscale features work together to enhance the robustness of the model in object recognition across different shapes and sizes.

III. Methodology

A. Data acquisition and Preparation

A simulation environment was established to represent a single-lane road designated for autonomous robot movement. A zebra crossing was incorporated to simulate pedestrian crossings, with detection restricted to pedestrians within the robot's lane, defined as a 2-meter-wide path marked by cones, as shown in Figure 4. Detection was intentionally excluded for individuals crossing beyond these lane markers to focus solely on potential collision threats within the robot's pathway.

Point cloud data was obtained from a constant 0.6-meter surface mounting of a VLP-16 LiDAR sensor and its configuration parameters were set to figure out the approximate height of pedestrians ranging from 155 to 180 centimeters, a standard human height that assists in object filtering. Additionally, in order to increase the accuracy of the detection of the unmanned device, there were changes in the point cloud data to the bird eye view perspective along with other suitable trimmed point clouds before and after the floor and ceiling, as shown in Figure 5. This method has reduced the noise and hence, raised the efficiency of the processing and also preserved the simple data pipeline.

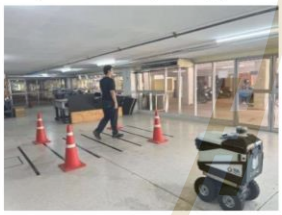


Fig. 4: Testing ground

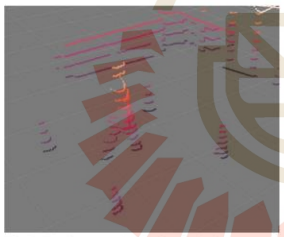


Fig. 5: Testing ground on Pointcloud data

B. PointPillars Setup

The input features that were included in the original PointPillars model include x, y and z coordinates, and intensity values. For this approach to be used in near-field detection, the range of detection has been set to -50 to 50, with the voxel property set to 0.25 meters in all dimensions (x=0.25,y=0.25,z=4). This smaller voxel size is essential in having a finer resolution in close object detection. A point cloud data is repeated segmentation into voxel grids within the detection range facilitates the detection of smaller objects as well as provides a detailed view of those objects.

C. Range Expert for Near-Field Detection

In line with the Range Expert method [4], the object detection range and the voxel sizes are optimized for the specific pirate object detection task within the range of radius 2 to 5 meters. The detection range is set to the values of 0 to 10 meters, with the voxel size unit as 0.1. This fine voxel size enables the voxels to capture details even in the near field, thus making certain that minute and subtle motions are detected accurately.

D. APN Implementation

When the model receives point cloud data from the VLP-16 sensor, the data is passed through the near-field proximity grid defined by the Range Expert. This grid is optimized for detecting objects within 10 meters, using fine voxel resolution for near-field proximity detection.

The Pillar Feature Net processes the point cloud data by converting it into pillars, resulting in a 2D pseudo-image representation of the 3D point cloud. This pseudo-image is passed through the backbone CNN, which extracts features from each pillar. After an initial convolutional layer extracts a feature map, a max-pooling operation reduces the image's size, retaining the most important features.

Next, the features are processed by the Feature Attention Fusion Block of the Lightweight APN [7]. In this block:

- Feature Attention identifies which features are the most salient across the various areas.
- Feature Fusion supplements low-level features with high-level features for a more complete feature representation.

The proposed multi-scale fusion allows retaining both fine details and contextual information which enhances detection of objects of different sizes. Finally, the upsampled features are passed through the Detection Head [5] (SSD Single Shot Detector) that infers the target's class and 3D bounding boxes.

E. Hardware and Training

The model was realized on the Jetson Xavier AGX platform with the intention of employing its resources for real-time 3D object recognition. The model was trained on a number of key parameters such as the voxel size and the detection area that would be required to increase the performance of the model. The training process lasted for 9000 episodes and trained this model so well such that the training loss was below 0.1. A dataset was used that contained information on the annotation of the people, and several data augmentation processes, namely, random rotation and translations, were performed to make the model stronger and more accurate under different conditions. These modifications positively affected the model's ability to correctly detect people, which can be used in autonomous navigation and collision avoidance systems.

IV. Results and Discussion

A performance evaluation of the model PointPillars and the devised method is given and based on the test data set. The models are also compared based on Frames Per Second (FPS), mean Average Precision (mAP), and Recall values . This comparison pursues the goal of measuring effectiveness and efficiency of each of the models while in similar testing conditions, with the results of the proposed model presented in Figure 6.

TABLE I. Performance evaluation of model

Evaluation matrix	FPS	mAP(0.5 IOU)	Recall
PointPillars	14.88	0.9687	0.75
Proposed model	23.99	0.969	0.8750

- According to Table I, the FPS value in the case of the proposed method is more than that of the PointPillars model, therefore, its practical implementation can be optimized even further. This improvement is made possible by reducing the detection range from 100 meters to 20 meters so as to lessen the burden on the processor or Jetson Xavier allowing some effectiveness in operating the models.
- According to Table I again, the mAP value in the case of the proposed method appears to be within the range of the PointPillars model which shows that the proposed method was also quite effective through the test statistics. This means that the methods planned to be deployed in the models are able to achieve accuracy close to that of the PointPillars model with possibilities of better efficiency.
- Utilizing the information presented in Table I, it can be understood that the proposed method has a higher Recall value than the PointPillar model meaning that object detection is enhanced when this method is used. This is achieved by increasing the density of point cloud points which is done by shrinking the voxel size from 0.25 to 0.1 and activating an Attention block that enhances finer feature extraction more than the original CNN-based structure features. Overall, it can be said that these changes lower prediction error overall where the application instated for safety critical areas is very important for accurate object search.

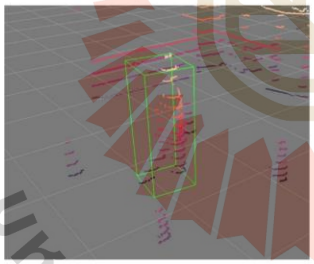


Fig. 6. Prediction of proposed method

V. Conclusion

A robust autonomous edge 3D object detection system was successfully created and implemented on the Jetson Xavier AGX platform. Substantial improvements in computation speed and detection accuracy were achieved by optimizing the PointPillars model through careful modification of voxel size, detection range, and the adoption of a Lightweight Attention Pyramid Network. The improved technique demonstrated strong mAP results and a higher Recall compared to the original PointPillars model, highlighting the feasibility of real-time 3D detection in low-dynamic environments. This advancement enables applications requiring high accuracy and fast computations, such as collision avoidance and path planning for autonomous navigation.

References

[1] Li, P., Chen, X., & Shen, S. (2019). Stereo r-cnn based 3d object detection for autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 7644-7652).

[2] Wang, Y., Xu, S., & Zell, A. (2020). Real-time 3D Object Detection from Point Clouds using an RGB-D Camera. In ICPRAM, (pp. 407-414).

[3] Liu, L., He, J., Ren, K., Xiao, Z., & Hou, Y. (2022). A LiDAR–camera fusion 3D object detection algorithm. Information, 13(4), 169.

[4] Peri, N., Li, M., Wilson, B., Wang, Y. X., Hays, J., & Ramanan, D. (2023). An empirical analysis of range for 3d object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, (pp. 4074-4083).

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In ECCV, 2016.

[6] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, (pp. 2117-2125).

[7] Zhang, J., Yan, Y., Cheng, Z., & Wang, W. (2020). Lightweight attention pyramid network for object detection and instance segmentation. Applied Sciences, 10(3), 883.

[8] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, (pp. 652-660).

[9] Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018, December). Convolutional neural network (CNN) for image detection and recognition. In 2018 first international conference on secure cyber computing and communication (ICSCCC), (pp. 278-282). IEEE.

[10] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, (pp. 12697-12705).

ประวัติผู้เขียน

นายนัฐพงษ์ พวงมาลี เกิดเมื่อวันที่ 11 กุมภาพันธ์ พ.ศ.2543 ที่โรงพยาบาลค่ายสุรนารี อ.เมืองนครราชสีมา จ.นครราชสีมา สำเร็จการศึกษาชั้นมัธยมต้นและมัธยมปลายที่โรงเรียนสุรธรรมพิทักษ์ จังหวัดนครราชสีมา ปี พ.ศ. 2561 เข้ารับการศึกษาระดับอุดมศึกษา ณ มหาวิทยาลัยเทคโนโลยี สุรนารี สาขาวิชาวิศวกรรมเครื่องกล สำเร็จการศึกษาปี พ.ศ. 2564

ปีพ.ศ. 2565 เข้ารับการศึกต่อในระดับปริญญาโท ณ มหาวิทยาลัยเทคโนโลยีสุรนารีหลักสูตรวิศวกรรม ศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมเมคคาทรอนิกส์ หัวข้อวิจัยเกี่ยวกับการปรับปรุงสถาปัตยกรรม PointPillars ด้วยการประมวลผลข้อมูล LiDAR เพื่อลดภาระการคำนวณของระบบตรวจจับ 3 มิติได้รับการสนับสนุนทุน OROG เป็นระยะเวลา 2 ปี ในระหว่างการศึกษาได้รับมอบหมายให้ เป็นผู้ช่วยสอนในรายวิชา ระบบยานพาหนะภาคพื้นดินที่ขับขี่ด้วยตัวเอง มีโอกาส ได้ร่วมทำวิจัยกับคณาจารย์ผู้ทรงคุณวุฒิ จนได้รับการเผยแพร่ผลงานจำนวนทั้งสิ้น 1 งาน คือ ผลงานประชุมวิชาการระดับนานาชาติ 1 ฉบับ

1) Phuangmalee, N., & Tantrairatn, S. (2024, November). Enhancing 3D Object Detection for Edge Devices: A PointPillars Approach with Attention Networks and Range Adjustment. In 2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 111-114). IEEE.