ROS2 Object Detection and Tracking in field Autonomous Vehicles

การทำ Objective detection ด้วย ROS2 สำหรับการใช้งานรถยนต์อัตโนมัติ

Weerapat Supapornopas 63340500061

Problem Statement and Introduction

การใช้งานระบบการติดตามการเคลื่อนไหวของวัตถุเป็นสิ่งสำคัญในหลายแอปพลิเคชัน เช่น การติดตามการเคลื่อนไหวของรถยนต์ หรือการติดตามการเคลื่อนไหวของผู้คนในระบบขับขี่ไร้คนขับ โดยผ่านการใช้ Computer Vision ซึ่งสามารถช่วยในการระบุตำแหน่งและชนิดของวัตถุได้ อย่างไรก็ตามทางบริษัท gensurv co. ltd นั้นได้กำลังพัฒนาระบบรถขับเคลื่อนอัตโนมัติอยู่แต่ยังติดปัญหาในการตรวจจับรถจักรยานยนต์เนื่องจากตัวระบบนั้นไม่ได้รองรับในส่วนนี้

ดังนั้นเพื่อเพิ่ม ความสามารถของระบบการตรวจจับวัตถุในรถขับเคลื่อนอัตโนมัติ จึงเลือกหัวข้อการ ทำ implement Object detection กับ Ros2 ให้สามารถระบุตำแหน่งและชนิดของวัตถุใน Ros 2 เพื่อไปพัฒนาในระบบรถยนต์ไร้ขนคับต่อไป

Background Study

1. ROS2

ROS2 หรือ Robot Operating System 2 เป็น framework ที่ถูกออกแบบมาเพื่อใช้ในการพัฒนาระบบหุ่นยนต์และระบบอัตโนมัติ โดยมีความยืดหยุ่นในการทำงานเป็นจุดเด่นที่สำคัญของ ROS2 ซึ่งสามารถทำงานร่วมกับหลายฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในหุ่นยนต์ได้ นอกจากนี้ ROS2 ยังมีความสามารถในการสื่อสารและจัดการข้อมูลในระบบหุ่นยนต์ได้อย่างมีประสิทธิภาพ

ROS2 มีโครงสร้างแบ่งออกเป็น module ที่เรียกว่า nodes ซึ่งทำให้สามารถทำงานแบบ distributed ได้ นั่นคือการทำงานใน ROS2 จะเป็นการแบ่งงานออกเป็นหลายๆ ส่วน และทำงานในแต่ละส่วนนั้นสามารถทำงานบนคอมพิวเตอร์หรืออุปกรณ์ต่างๆ ที่เชื่อมต่อกันผ่านทางเครือข่ายได้ โดยมีการใช้โปรโตคอล ROS2 ในการแลกเปลี่ยนข้อมูลระหว่าง nodes

นอกจากนี้ ROS2 ยังมีความยืดหยุ่นในการทำงานภายใน node ด้วยการใช้ตัวกลางที่เรียกว่า ROS2 middleware ซึ่งช่วยในการจัดการการสื่อสารและข้อมูลในระบบ โดย ROS2 middleware จะช่วยในการเชื่อมต่อหรือเปลี่ยนแปลงฮาร์ดแวร์และซอฟต์แวร์ต่างๆ ที่ใช้ในหุ่นยนต์ได้อย่างมีประสิทธิภาพ นอกจากนี้ ROS2 ยังสามารถรองรับการใช้งานหลายภาษาและรองรับการทำงานบนหลายแพลตฟอร์ม ซึ่งทำให้ ROS2 เป็นเครื่องมือที่ใช้งานได้หลากหลายและมีความยืดหยุ่นสูงในการพัฒนาระบบหุ่นยนต์และระบบอัตโนมัติ

นอกจากนี้ ROS2 ยังมีการพัฒนาต่อยอดอยู่เรื่อยๆ โดยมีการเพิ่มฟีเจอร์ใหม่ๆ เพื่อตอบสนองความต้องการของผู้ใช้งานในสายงานหุ่นยนต์และระบบอัตโนมัติ โดย ROS2 ยังมีการใช้งานในโครงการหลายๆ

โครงการที่เกี่ยวข้องกับหุ่นยนต์และระบบอัตโนมัติ เช่น โครงการ ROS-Industrial ที่ใช้งานในการพัฒนาระบบการผลิตและจัดการโรงงาน นอกจากนี้ยังมีการใช้งาน ROS2 ในการพัฒนาหุ่นยนต์ที่ใช้ในงานวิจัยและการศึกษาด้วยเช่นกัน

2. Open-CV

OpenCV (Open Source Computer Vision Library) เป็นไลบรารี Open souce เพื่อการประมวลผลภาพและวิดีโอ รวมถึงฟังก์ชันการทำความเข้าใจภาพ เช่น การตรวจจับวัตถุ การจัดการกับภาพและวิดีโอ การแยกแยะวัตถุ การหาและวิเคราะห์ลักษณะของภาพ และอื่นๆ OpenCV

ถูกพัฒนาโดยใช้ภาษา C++ และสามารถรองรับการใช้งานบนหลายแพลตฟอร์ม เช่น Windows, Linux, macOS, iOS, Android และ Raspberry Pi เป็นต้น และยังสามารถเชื่อมต่อกับกล้องและอุปกรณ์อื่นๆ เพื่อดึงภาพมาใช้งานในการประมวลผลภาพได้อย่างง่าย

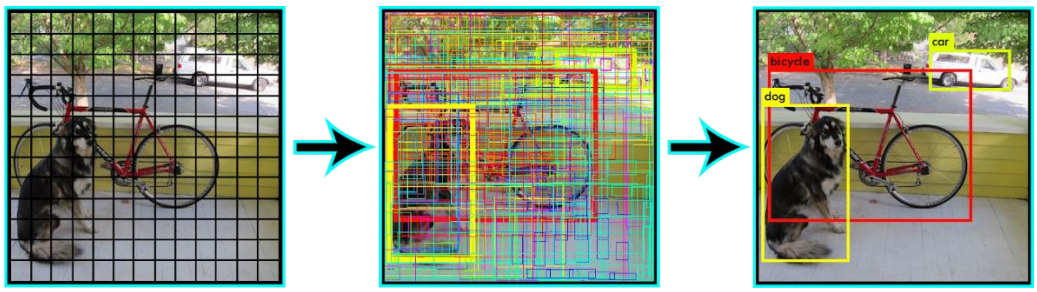
สำหรับฟังก์ชันการประมวลผลภาพที่ OpenCV สามารถทำได้ สามารถแบ่งได้เป็นหลายๆ ประเภท เช่น

- การจัดการกับภาพ : เช่น การปรับขนาดภาพ, การปรับความคมชัดภาพ, การตัดภาพ, การแปลงรูปแบบสี
- การตรวจจับวัตถุ : เช่น การตรวจจับใบหน้า, การตรวจจับตำแหน่งและขนาดของวัตถุ, การตรวจจับคน
- การวิเคราะห์ภาพ : เช่น การวิเคราะห์รูปแบบการเคลื่อนไหวของวัตถุ, การวิเคราะห์ภาพการแสดงอารมณ์, การวิเคราะห์ภาพสไตล์เสื้อผ้า

นอกจากนี้ OpenCV ยังสามารถเชื่อมต่อกับกล้องและอุปกรณ์อื่นๆ ได้อย่างสะดวกสบาย เพื่อดึงภาพมาใช้งานในการประมวลผลภาพได้อย่างง่ายดาย ทำให้สามารถนำไปประยุกต์ใช้งานในหลากหลาย

อุตสาหกรรมได้ เช่น หุ่นยนต์, การวิเคราะห์ภาพแพทย์, การตรวจสอบความถูกต้องของผลิตภัณฑ์, การตรวจสอบคุณภาพ, และอื่นๆ

3. YOLO



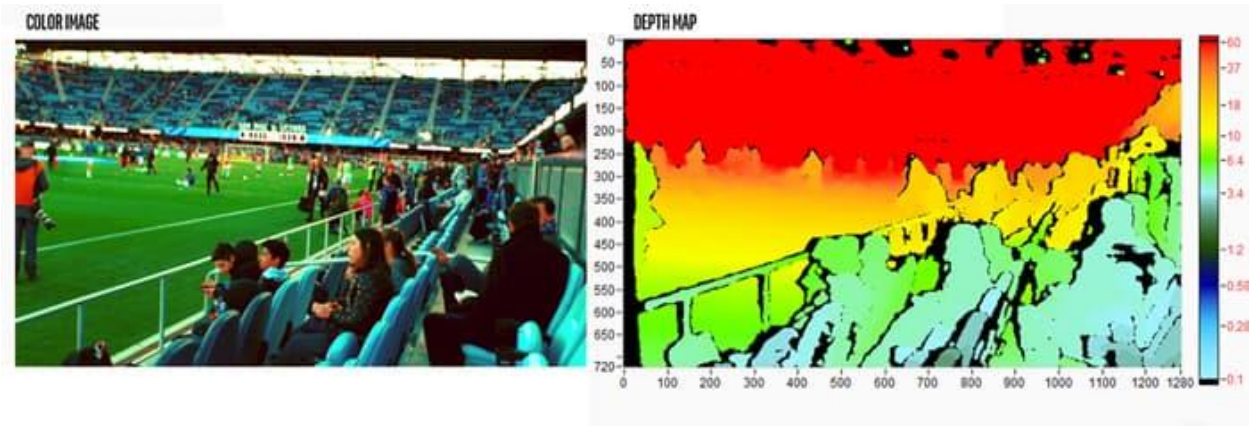
YOLO (You Only Look Once) เป็นโมเดลการตรวจจับวัตถุ (object detection) ที่ออกแบบมาให้มีประสิทธิภาพสูงและความเร็วสูงในการทำงาน เป็นโมเดลที่นิยมใช้ในงาน Computer Vision โดยเฉพาะในงานตรวจจับวัตถุในวิดีโอ (video object detection)

YOLO มีจุดเด่นคือการทำงานโดยการแบ่งภาพเป็นเส้นตารางหรือตาราง (grid cell) แล้วสร้าง bounding box โดยใช้โมเดล regression เพื่อหาพารามิเตอร์ของกล่องขอบเขตนั้น พร้อมทั้งกำหนดค่าความเชื่อมั่น (confidence score) ว่าวัตถุที่ตรวจจับเป็นวัตถุประเภทใด และตำแหน่งของวัตถุนั้นอยู่ที่ใดในภาพ

โดย YOLO แบ่งพื้นที่ของภาพเป็นส่วนเล็ก ๆ แต่มีจำนวนมาก แล้วทำการตรวจจับวัตถุบนแต่ละพื้นที่ดังกล่าว ดังนั้นการทำงานของ YOLO จึงมีความเร็วสูงและสามารถตรวจจับวัตถุได้หลายวัตถุในภาพเดียว อีกทั้ง YOLO ยังสามารถจำแนกวัตถุที่อยู่ใกล้เคียงกันได้ดีเนื่องจากใช้เทคนิค non-max suppression เพื่อลดจำนวน bounding box ที่ไม่จำเป็นออกจากภาพ นอกจากนี้ YOLO ยังสามารถใช้งานได้ง่ายและมีการตั้งค่าที่หลากหลายให้กับผู้ใช้งาน

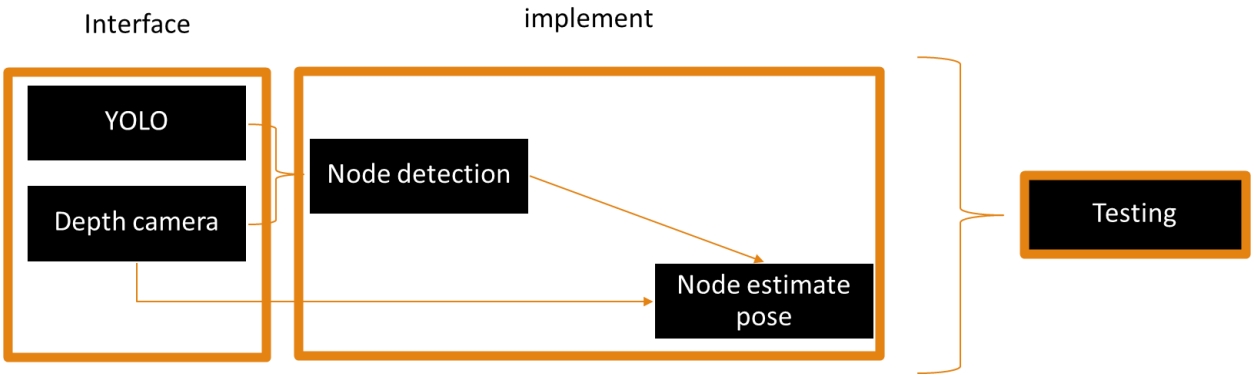
4. Depth map

เป็นข้อมูลที่ใช้บอกความลึกของ pixel ของรูปภาพโดยที่ผู้จัดทำเลือกใช้ Intel realsense Depth camera D455 ที่มี framework ในการคำนวณหาความลึกผ่าน Stereo camera 2 ตัว



Conceptual Design

ศึกษาตัว Model object detection ที่ใช้งานกันในปัจจุบันแล้วทำการไปใช้งานกับตัว Intel real sense camera เพื่อช่วยในการหาตำแหน่งของวัตถุโดยที่ต้องนำมา Implement ใน Ros2



Requirement

- ตัว **model** ต้องสามารถ **detect** หามอเตอร์ไซต์ได้ภายในไม่เกินวินาที
- ต้องสามารถทำงานได้แบบเรียลไทม์

Team, Task, Process

Mildstone	Task	Deliver Date
Scrum 2	- ศึกษาวิธีการใช้งานของแต่ละ เวอร์ชั้น YOLO - ทำการ Interface ตัว depth camera	**22/03/2023**
Scrum 3	- ปรับใช้ตัว model YOLO เค้ากับระบบของ Ros2 - เตรียม data set เองมาเทสตัว model - นำ depth frame มาใช้ในการ estimate position	**19/04/2023**
Scrum 4	- ศึกษา Ros bag มาใช้ในการบันทึกข้อมูล - บันทึกข้อมูล - ลองนำโค้ดเทสกับข้อมูลที่ได้บันทึกไว้	**10/05/2023**
Final presentation	- Optimize code - final report - presentation	**24/05/2023**

Learning Plan

1. Study and Interface
    - Yolo
    - Depth camera with ros2
2. Implement
    - Detection node
    - Estimate position mode
3. Testing
    - Record and save data.
    - Testing code
4. Optimize code

Methodology

1. Interface

1.1. Yolo

โดยจะเน้นศึกษาไปที่ตัว Yolov5 เนื่องจากเป็นที่นิยมและเป็นเวอร์ชั่นที่ออกมานานและเสถียรแล้ว

รูปภาพที่ 1

1.2. ทำการติดตั้ง SDK ควบคู่กับ ros2 realsense

รูปภาพที่ 2

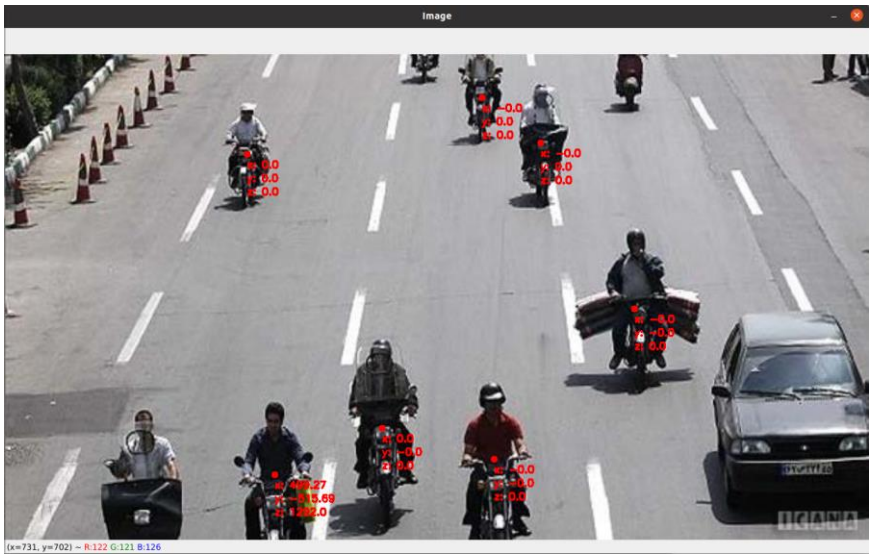
2. Implement

2.1. Qos

เป็นการตั้งค่าและจัดการคุณภาพของการสื่อสารระหว่างโหนด ในระบบ ROS ซึ่งช่วยให้สามารถควบคุมการส่งข้อมูลระหว่างโหนดได้อย่างมีประสิทธิภาพ ไว้สำหรับการจัดการเวลาส่ง รูปภาพที่ใช้ขนาดในการส่งข้อมูลที่ใหญ่

2.2. Node detection

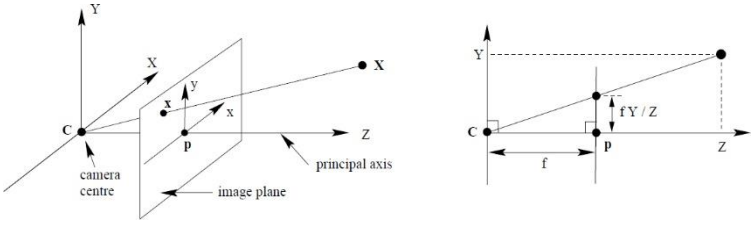
Subscriber รูปภาพจาก topic ของตัวกล้องและใช้โมเดล YOLO ในการตรวจจับพาหนะและส่งออกตำแหน่งที่เป็นพิกเซล



รูปภาพที่ 3

2.3. Node estimate pose

จะเป็นการรับตำแหน่งพาหนะจาก Node detection จากนั้นนำตำแหน่งที่ได้รวมกับ Topic Depth frame เพื่อระบุตำแหน่งใน ROS2 โดยใช้ pyrealsense2 ที่จะใช้ Pinhole camera ในการคำนวณเป็นการหาตำแหน่งของวัตถุใน frame world ผ่านการแปลงจากพิกแซลโดยที่จะอาศัยพิกเซลรูปภาพคู่กับ depth frame โดยที่จะมี f ระยะโฟกัส p เป็นตำแหน่งจุดกึ่งกลางของภาพ

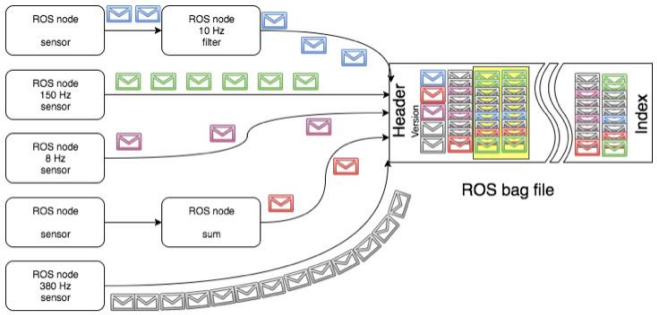


2.4. Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

3. Testing

3.1. Record

โดยเราจะทำการเก็บข้อมูลและบันทึก ผ่านตัว Ros bag เป็นโปรแกรมใน ROS 2 ที่ใช้ในการบันทึกข้อมูลและเล่นกลับ (replay) ข้อมูลที่สร้างขึ้นในระบบ ROS 2



รูปภาพที่ 4

3.2. ทดสอบ node ต่างๆกับ Ros bag file และทำการ ทดสอบ accuracy ของตัว estimate node

Results

1. Test model

ทดสอบจาก Data มอเตอร์ไซต์จำนวน 221 คัน จากรูปภาพ 111 ใบ

การทดสอบที่	Confidence	Image size	Accuracy	average Runtime/image
1	0.8	original size	0.719457014	0.140016427
2	0.5	original size	0.9095	0.140016427
3	0.5	original but format BGR	0.859728507	0.140016427
4	0.5	720x480	0.65158371	0.041946704
5	0.5	480x240	0.416289593	0.035787849

ตัวอย่างรูปภาพที่ไม่สามารถ Predict ได้

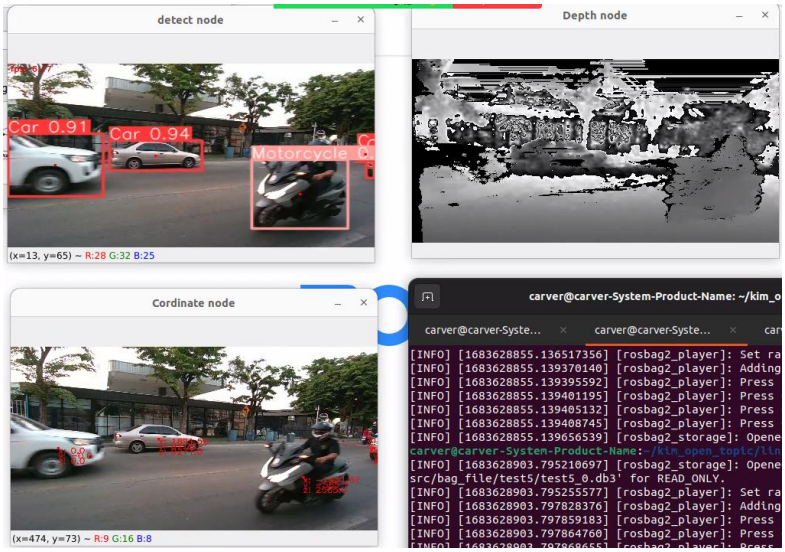
รูปภาพที่ 5

รูปภาพที่ 6

2. Test node

2.1 Test data from ros bag

ทำการทดสอบการทำงานแบบเรียลไทม์จากข้อมูลที่ได้ทำการบันทึกไว้ Link : File video



รูปภาพที่ 7 การ Demo 1

2.2 test performace code Optimize

ทำการปรับแก้โค้ดให้แยกตัว Task ที่ Interface รับและส่งข้อมูลรูปภาพออกจากการโมเดล Yolo เพื่อแยกภาระงานที่หนักออกมาอยู่ใน thread เดียว

```
main thread time:  0.09609556198120117
main thread time:  0.09662055969238281
main thread time:  0.09623408317565918
```

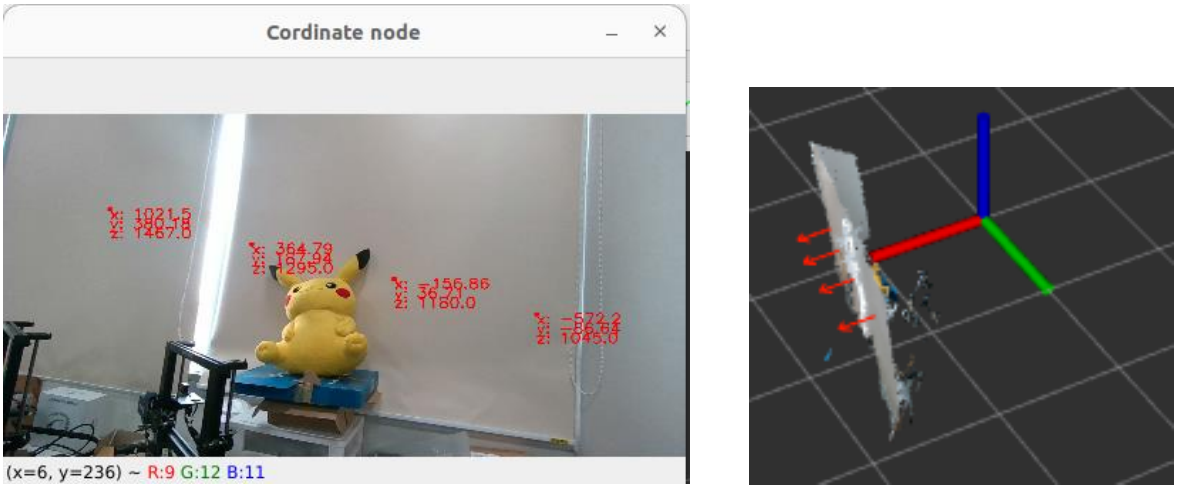
รูปภาพที่ 8 Single tread

```
anothor thread use time to process: 0.12292
main thread time:  0.007926225662231445
main thread time:  0.0073740482330322266
```

รูปภาพที่ 9 mutithread

2.3 test estimate node

ทำการเปรียบเทียบค่าที่ได้จาก Node estimate กับตำแหน่งใน Point Clound จากตัวกล้องโดยจะการส่งค่า Pose จาก node estimate ใน frame camera เพื่อดูความแม่นความใกล้เคียงของ ตำแหน่งที่ Estimate ได้



รูปภาพที่ 10 ตำแหน่งที่คำนวณเทียบกับตำแหน่ง point cloud

Discussion and Conclusion

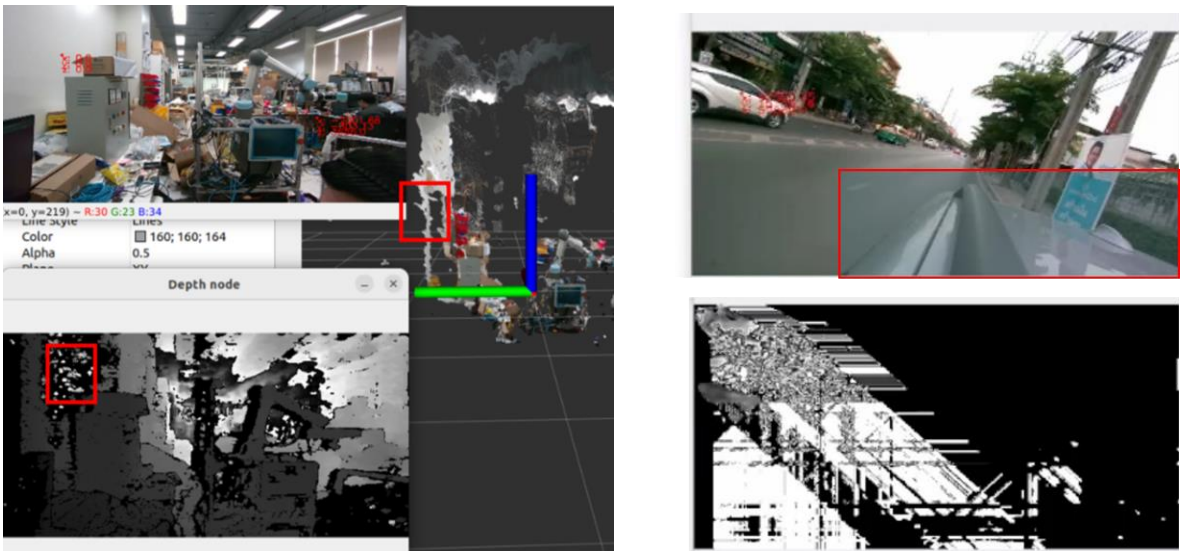
จากการทดสอบพบว่าเวลาที่รัน Node detection จะเกิดการ delay ของเฟรม 1 เฟรม โดยที่ Node detection นั้นใช้ทรัพยากร Gpu ที่สูงมากทำให้ต้องกำหนดความถี่ในการคำนวณไว้เพียงแค่ 5-10 Hz ทำให้เวลาเกิดการเฟรมล่าช้าจะเห็นผลได้ชัดเจนมากจากวิดีโอ โดยการแก้ไขนั้นสามารถทำได้โดยที่การย้าย task ในการประมวลผลภาพไปอยู่อีก Task หนึ่งเดียวๆเลย

ปัญหาที่พบเลยระหว่างการทำงาน

1. เรื่องสภาพแวดล้อมในการเก็บผลเนื่องจากตัวกล้องสามารถทำงานได้ในช่วงอุณหภูมิ 0 ถึง 35 องศาทำให้ช่วงเวลาในการเก็บผลจะเหลือเพียงแค่ช่วง 4 โมงของแต่ละวันซึ่งหากไปเก็บสายเกินไปจะเป็นช่วงเวลาเลิกงานซึ่งมีการจราจรที่หนาแน่นหรือหากเดินทางไปเร็วเกินด้วยสภาพอากาศที่ร้อนจะทำให้ตัวกล้องเกิด Error ขึ้น

รูปภาพที่ 12 รูปภาพที่เกิดการ Error จากอากาศที่ร้อน

2. การ Error ของDepth frame หากตัวรูปภาพมี pixel ที่มีสีใกล้เคียงกันหรือส่วนใหญ่มีพื้นที่ใกล้ๆกล้อง ตัว Depth frame จะไม่สามารถคำนวณในช่วงหนึ่งได้หรืออาจจะเกิด Error เลยทั้งภาพ



รูปภาพที่ 13 ข้อมูลที่ Depth frame เกิดการ Error จากการที่มีส่วนหนึ่งมี Pixel ที่คล้ายคลึ่งกัน

Reference

https://github.com/pound03/FRA361-ROBOTICS-STUDIO-IV-OPEN-TOPICS

[1] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

[2] IntelRealSense. realsense-ros, https://github.com/IntelRealSense/realsense-ros , 2022

[3] glenn-jocher, yoloV5, https://github.com/ultralytics/yolov5 , 2022

[4] MaryamBoneh, data set Vehicle, https://github.com/MaryamBoneh/Vehicle-Detection .2020