

CHAPTER 2 - LITERATURE REVIEW

In this chapter, information on hardware, software, and theories adapted and used in this thesis project will be discussed and explained thoroughly. Firstly, SDC (self-driving car) and software used in this project, which is ROS 2.0 will be discussed. Next, the sensors used, theories, and systems will be covered. Lastly, several existing research papers and preceding related theses projects will also be discussed.

2.1 Theoretical Perspectives

2.1.1 Self-Driving Car

SDC (Self-Driving Car) or Autonomous Vehicle (AV) is a term for vehicles that are intelligent which have capabilities similar to a wheeled robot. SDC has a goal to arrive at a destination based on information obtained from sensors embedded in the car which includes perceptions of its environment, targeted route path as well as actuation of the car's movement. SDC has one main goal which is transporting people or objects to a predetermined target without any external intervention. (Zhao, Liang and Chen, 2018)

SAE International (Society of Automotive Engineers International), an association that develops standards for automotive, publishes a classification system for SDC in 2014 which consists of six levels from 0 to 5 based on the amount of driver intervention on the vehicle. It is also usually called "driving mode" which means the type of driving scenario required by the driver as seen in Figure 2-1. Level 0 (No Automation) requires the driver to fully control and put all attention on the vehicle. Level 1 (Hands-On) has the system assist with steering such as cruise or lane assistance. Level 2 (Pay-Attention) can provide steering and speed features in the system such as maneuvering in traffic and maintaining speed in traffic, but the driver still needs to put their hands on the steering wheel. Level 3 (Eyes-Off) allows the driver to take their hands off the steering wheel and put their eyes on other activities as the system can actuate the car automatically but will generate an alarm for the driver to take over. These levels are further distinguished by manufacturers as two groups, the driver is responsible (level

0-2) and not responsible (level 4 and 5). Level 3 is considered controversial as the driver needs to be responsible meanwhile the system is doing most of the control. (*SAE Self-Driving Levels 0 to 5 for Automation - What They Mean - AutoPilot Review*, no date)

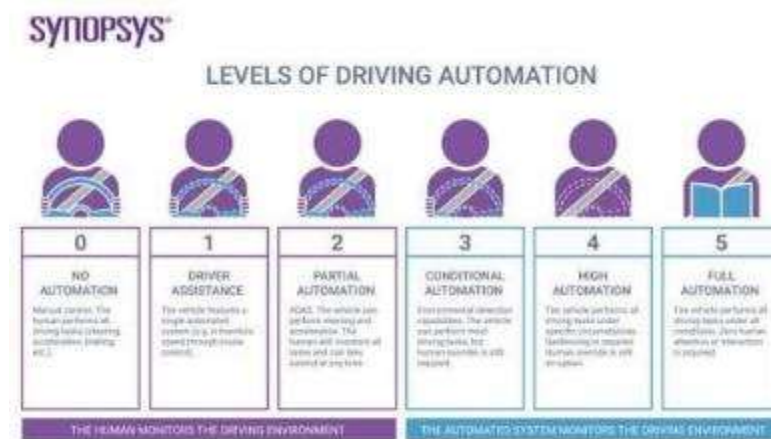


Figure 2-1 Levels of Driving Automation Based on SAE International

An SDC can be called a real SDC is when they are considered in level 4 or 5 as they do not require a driver to sit on the driver's seat. Level 4 (Mind Off) system provides all driving functions and safety requirements of the vehicle without needing human interventions. Level 4 vehicles are limited in some conditions such as bad weather in which humans can directly take control of the car or automatically park the car itself. Level 5 (Chauffeured) is the ultimate goal of SDC as human interventions will never be needed as they can overcome challenges in any kind of condition. (*SAE Self-Driving Levels 0 to 5 for Automation - What They Mean - AutoPilot Review*, no date)

On a single SDC, tons of the most sophisticated technology is contained inside it, such as artificial intelligence, computer vision, pattern recognition, and many others. As the goal of SDC is to replace humans and create a system that can act like a human, SDC is required to see the environment, process the information, make decisions, and actuate through the environment. The main technology of SDC is classified into four key parts, which are car navigation system, path planning, environment perception, and car control. This is further simplified into three processes which are called the AD (Autonomous Driving) Stack, consisting of sensing, computing, and actuation as seen in Figure 2-2. In the end, level 5 SDC is only achievable if the massive amount of data required are obtained. (Zhao, Liang and Chen, 2018)

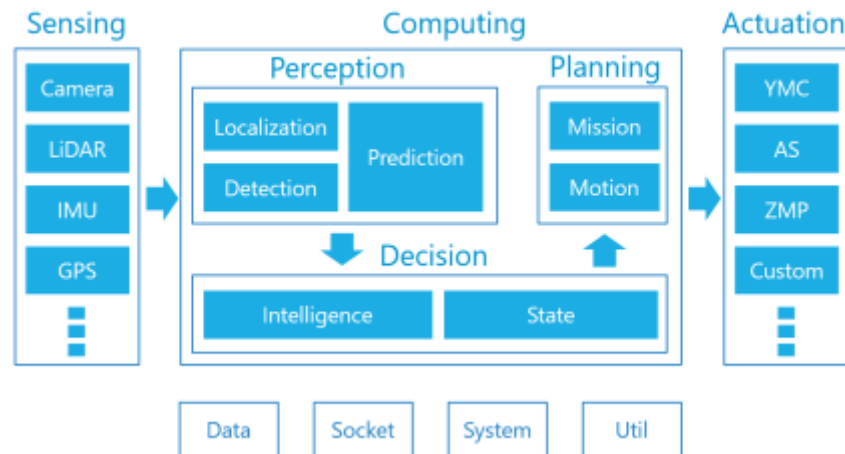


Figure 2-2 Autonomous Driving Stacks

A) Tesla versus Waymo SDC

In the world of SDC, the two companies, Tesla and Waymo, are considered the biggest contenders to make level 5 SDC a reality in the future as they currently have the most data required to achieve it despite their different approaches.

Tesla is a big company in the world of electric vehicles and is spread all over the world. With hundreds of thousands of customers, Tesla used these customer-owned cars to gather data to develop their SDC, without breaking its privacy policy as it collects the information on how the feature Tesla's cars provide. Elon Musk, Tesla's owner, stated that Tesla can collect 3 million miles of data per day and will increase as Tesla sells more cars. (O'Kane, 2018)

Waymo is started as Google's SDC project and uses supercomputer simulations to collect their data. From these simulations, it is said that they have collected up to 5 billion miles of driving data as well as 5 million miles on public roads obtained from their own self-driving Pacifica minivans. Recently, Waymo also announced a partnership with Jaguar Land Rover to develop an SDC version of their electric SUV. (O'Kane, 2018)

With these two different ways of data collection, Tesla is considerably leading in these data collection competitions as Waymo's data are mostly based on

simulations, these simulations would not be able to come up with all kinds of strange real-world scenarios. Despite having more advanced technologies for simulations, Tesla is not left behind in the SDC industry. (O’Kane, 2018)

B) Different Sensors Implementation on SDC

In an SDC, there are different kinds of data taking that are required to accomplish its data collection, such as localization and object detection. Localization is the act to obtain a robot’s position on a certain map, typically used sensors are GNSS, LiDAR, and robot odometry. Object detection is the act to detect obstacles on the route and used the data to avoid hitting the obstacle, typically used sensors are camera, LiDAR, sonar, and radar.

With different kinds of sensors for SDC applications, different companies have their own choices of sensors they utilize. Tesla uses various kinds of cameras, ultrasonic sensors, GNSS, and radar for localization and object detection. Meanwhile, Waymo utilizes cameras, radar, GNSS, and LiDAR. Autoware Foundation, a non-profit organization from Japan, released an SDC project called AutowareAuto which focuses on Level 4 SDC for Autonomous Valet Parking and Autonomous Depot Maneuvering. The sensors that are used in this project are cameras, LiDAR, IMU, GNSS, and radar.

2.1.2 ROS (Robot Operating System)

ROS is an open-source, meta-operating system used to create software for a robot. Although it has the name OS (Operating System) in it, it is not an actual operating system but a collection of frameworks that provides functions as an operating system on a heterogeneous computer cluster, set of computers connected to perform tasks with individual independent processors. ROS as a robotics middleware consists of tools and libraries that ease and reduce the complexity in creating robot software despite requiring to create a robust robot behavior. (Ademovic, no date)

ROS is developed because creating truly robust, general-purpose robot software is difficult. When creating robots, they are designed to imitate human activities. Problems

that seem easy for humans requires various kinds of actions of the robot to accomplish what human do easily. These variations remain a challenge when different kinds of robots needed to be made by different individuals or institutions. (*ROS.org | About ROS*, no date)

ROS was built with the goal of collaborative robotics software development. In which, reuse of codes in robotics research and development is the key goal of ROS. Different experts creating different kinds of codes to use together in one single platform, which is ROS as seen in Figure 2-3. Despite its primary goal of sharing and collaboration from robot software developers, there are other goals of the ROS framework, which are:

- 1) ROS is thin, in which ROS code can be used with other robot software frameworks such as OpenRAVE, Orocos, and Player.
- 2) ROS has agnostic libraries in which it is agnostic or independent of any other frameworks with clean functional interfaces.
- 3) ROS is language-independent as ROS frameworks are easy to implement with any modern programming language such as C++ and Python.
- 4) ROS is easy to test and simulate as there is a built-in test framework such as *rostopic*.
- 5) ROS is suitable for large runtime systems and large development processes.

(*ROS/Introduction - ROS Wiki*, no date)



Figure 2-3 ROS Development Architecture

ROS has thousands of packages with each package providing specialize functionality which makes ROS not only limited to robots but also other peripheral hardware. ROS provides functionality for hardware abstraction, device drivers, communication between processes over multiple machines, tools for testing and visualization, and so many others.

ROS has a unique feature in its software in how the software is run and how it communicates which allows multiple programs to run at once, which is using nodes.

Node is an individual process in form of executables that can receive or send data to other nodes through the topic. When two or more nodes use the same topic, they can communicate to each other using that particular topic, either publishing or subscribing. Topic despite having unique names also has a message type which determines the types of message that is capable to be transmitted. A topic can be used for many-to-many communication as many publishers can send messages to the same topic and many subscribers can receive them as well as seen in Figure 2-4. Even without a subscriber, a topic could still be published to and that is one of the reasons why ROS ease users in creating robotic software as these topics could be easily be debugged. For this research, ROS Foxy Fitzroy will be used as ROS2.0 is a new ROS API that supports real-time codes. (*Exchange Data with ROS Publishers and Subscribers - MATLAB & Simulink*, no date)

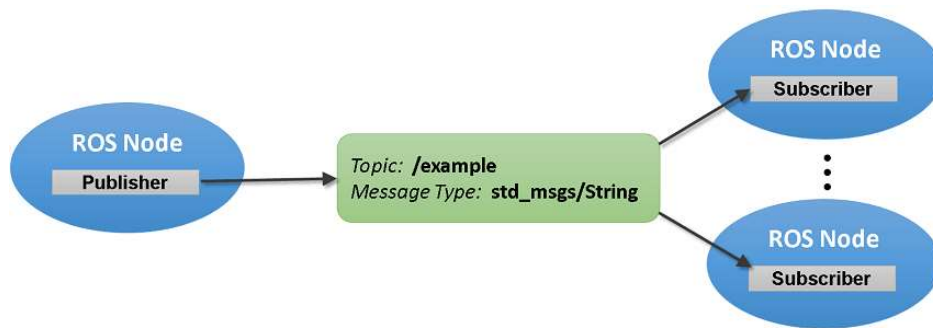


Figure 2-4 ROS Nodes and Topics Communications

ROS usage has expanded so widely into many different fields such as industrial arms, self-driving cars, aerial vehicles, and more. Here as seen in Figure 2-5 are some robots used and available to work with on ROS as well as in Figure 2-6 all the features of ROS.



Figure 2-5 Robots Created with ROS

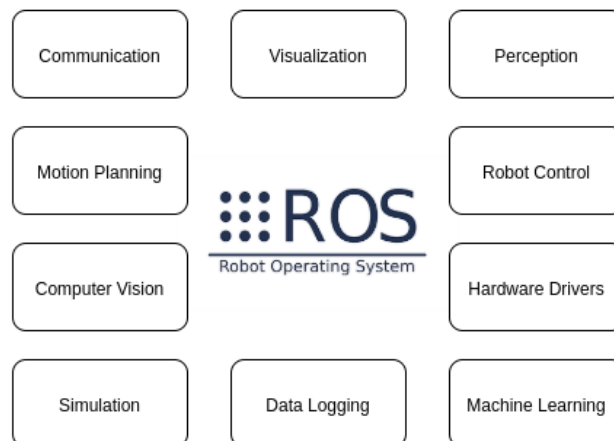


Figure 2-6 ROS Features

A) ROS2

ROS2.0 or ROS2 in short is created to adapt the changes from ROS, adapting what is great about ROS and improving what is not. ROS as a middleware has several problems such as capabilities of runtime adaptation to applications, communications between heterogeneous systems in different middleware levels, runtime safety assurance during failure, and others. In specific, here are ROS limitations and what ROS unable to do:

- 1) Multi-robot system: Being able to build more than one robot for one task or multi-robot system is key in industries. It could solve problems of insufficient performance and unavailability of a single robot.

- 2) Multi-platform: ROS is based on Linux OS and currently only has limited functions on other OS such as Windows, macOS, RTOS, and other systems.
- 3) Real-time: Many robot applications have high requirements for real-time performance, especially in the industry. ROS lacks real-time design and currently relies on other applications.
- 4) Network connection: ROS requires a good network environment to ensure data integrity. The network currently has no data encryption and security protection which causes any host in the network could obtain message data from the node.
- 5) Production: ROS has a hard time getting into the market as the stability of ROS is poor. Master and node communication can seldom be down in several cases, which keeps ROS robots mostly in the research phase.

(Introduction of Robot Operating Systems 2: ROS2 – SAFER AUTONOMOUS SYSTEMS, no date)

ROS2 changes and upgrades its platforms and dependencies compare to ROS. ROS2 is currently CI (continuously integrated) in Ubuntu Xenial, OS X El Capitan, and Windows 10 instead of only Ubuntu. For languages, ROS2 uses a wider variety of C++ up to C++11 to 17 and for Python, the requirement has increased to version 3.5. ROS2 is currently using an abstract middleware interface that is based on DDS (Data Distribution Service) standard. For package building, ROS2 allows not only C++ Catkin but also Python. ROS and ROS2 architecture comparison can be seen in Figure 2-7. (Thomas, no date)

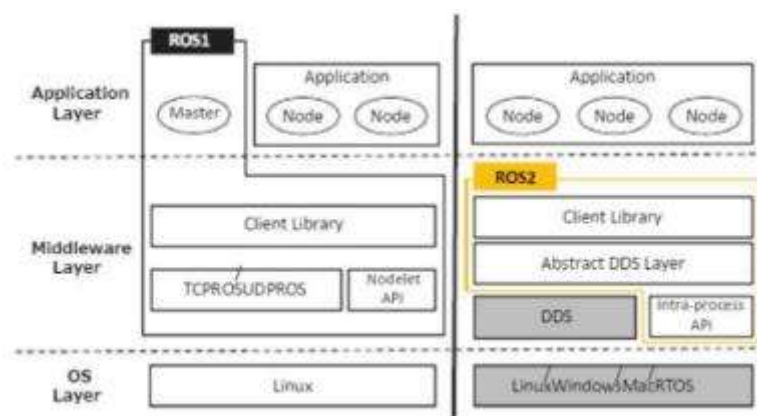


Figure 2-7 ROS and ROS2 Architecture Overview

ROS2 has almost all features and functionality of ROS such as publish/subscribe over topics, clients, and services, set/retrieve parameters, launch system for coordinating multiple nodes, namespaces support for nodes and topics, static remapping of ROS names, and many others. The new key functions that ROS2 has are transport and serialization over DDS, support for multiple DDS implementations, bridge communication between ROS, support for real-time code, and high-profile packages using top-notch tools such as gazebo_ros_pkgs, image_transport, navigation2, rosbag2, RQt, and RViz. ROS2 API and DDS communication can be seen in Figure 2-8. (*Features Status*, no date)

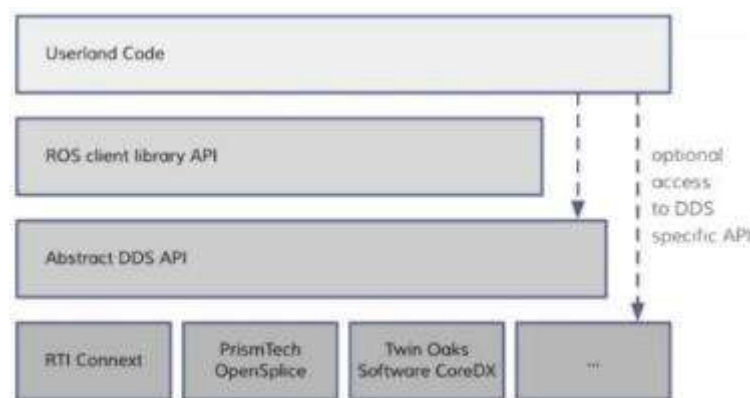


Figure 2-8 ROS2 API and DDS Communication

Currently, ROS2 has been built to meet industrial-grade safety and real-time requirements. People from different countries from different industries have started upgrading from using ROS into ROS2. These five things are the reason that supports the usage of ROS2, which are:

- 1) Safety and reliability, ROS2 has been designed to meet industrial-grade requirements for safety and reliability.
- 2) Many major companies contributing to ROS2 such as Bosch (IMU), Velodyne (LiDAR), Slamtech (LiDAR), SICK (LiDAR), Roboteq (Motor controller), Adafruit (sensor drivers and writing tutorial), and many others.
- 3) Lots of things are available for free and ready to use, there are simulators, visualizers, hardware drivers, and other components needed in robot application development.
- 4) Many companies from different fields are already using ROS2, even autonomous machines or software where companies keep developing new

products on top of ROS2 and old onboard software platforms are moving to ROS2.

- 5) ROS2 has wide support with the final release of ROS in May 2020 and all future releases will be in ROS2.

(Robot Operating System 2 (ROS 2) is Here – Why Should You Care?, no date)

2.1.3 Microcontroller

A microcontroller is a compact integrated circuit designed to do specific operations in an embedded system. A microcontroller is the upgraded version of microprocessors where microprocessors only consist of a central processing unit while microcontrollers emphasize more on interactions with hardware modules and controlling the system. It is created for tasks and applications that require both processing functionality and agile, as well as responsive interaction with digital, analog, or electromechanical components.

Microcontrollers consist of processors, memory, and peripherals. Processors or CPU (Central Processing Unit) performs arithmetic operations, manage data flows as well as generating signals issued by the sequence of instructions created by the programmer. Memory is divided into two which are non-volatile and volatile. Non-volatile or flash is used to store the program which is the machine-language instructions in form of binaries. Peripherals are the hardware that helps a microcontroller interact with an external system. There are several examples such as data converters, clock generation, timing, analog signal processing, input/output, and serial communication. Sometimes there is support circuitry which does not involve device control, which is debug circuitry, interrupts, and power-supply circuitry. Here as seen in Figure 2-9 is the architecture of a microcontroller. (Keim, 2019)

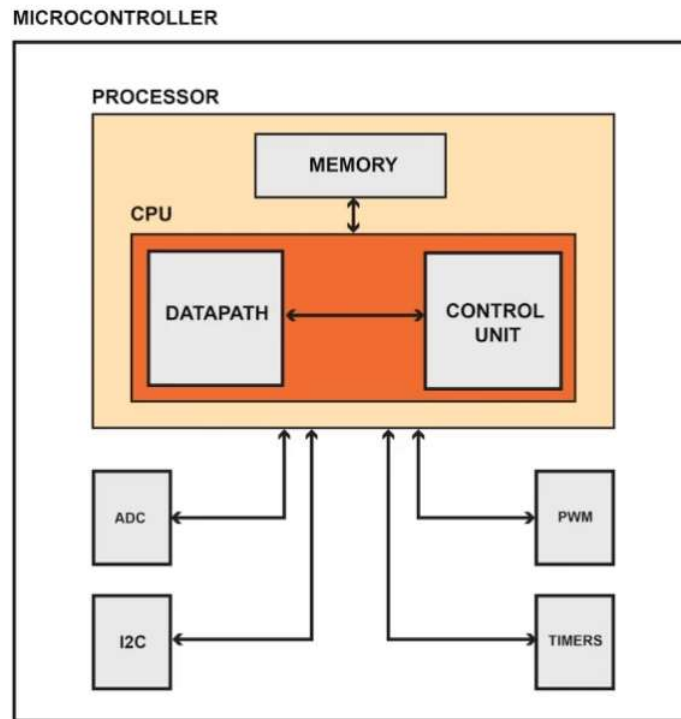


Figure 2-9 Microcontroller Architecture

For this thesis, there are three components required to connect as a microcontroller which are, Teensy, MicroROS, and Arduino.

A) Teensy

Teensy is a complete USB-based microcontroller that is capable of implementation for various kinds of projects. Teensy is compatible with Arduino software and libraries while programming it using Teensy Loader application ease the usage of Teensy. The Teensy board itself has pins that can be used on a solderless breadboard as seen in Figure 2-10. (*Teensy USB Development Board*, no date)



Figure 2-10 Teensy Usage on Solderless Breadboard

Teensy 4.1 is one of the latest Teensy microcontroller series with ARM Cortex-M7 processor at 600 MHz which performance is many times faster than typical 32-bit microcontrollers as seen in Figure 2-11. With a tightly coupled memory of 512kb, Teensy 4.1 can access memory with a fast single cycle as its 64 bit wide buses are a pair of 32-bit paths. Teensy 4.1 has a total of 55 input/output pins as seen in Figure 2-12, with 42 of them are accessible on a solderless breadboard. The pins consist of digital input-output, pullup, pulldown, keeper resistor, pin change interrupt, PWM (Pulse Width Modulation), slew rate limiting, variable drive, adjustable output bandwidth, LED pin, analog input, analog range, and analog comparators. Teensy 4.1 main communication is its USB port which operates at 490 Mbit/s. Several USB types can be used on this USB port as seen in Figure 2-13 such as Serial, MIDI, Keyboard, Mouse, Joystick, Touchscreen, and many others. Teensy 4.1 also allows a second USB port to operate in host mode which operates independently of the main port, allowing Teensy to communicate with two devices at the same time as seen in Figure 2-14. The most unique feature of Teensy 4.1 is its capabilities to connect to the Ethernet. Using an Ethernet controller and Ethernet PHY chip, to connect to an Ethernet cable only requires the RJ45 mag jack kit as seen in Figure 2-15. (*Teensy 4.1*, no date)

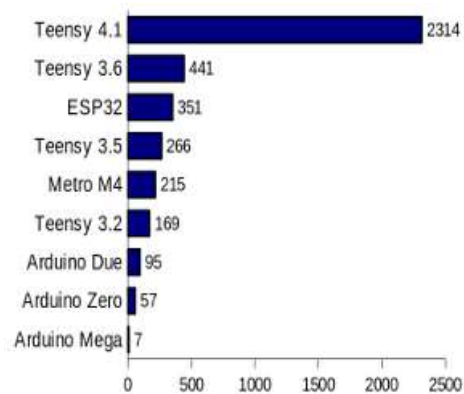


Figure 2-11 Teensy 4.1's ARM Cortex-M7 Speed Comparison

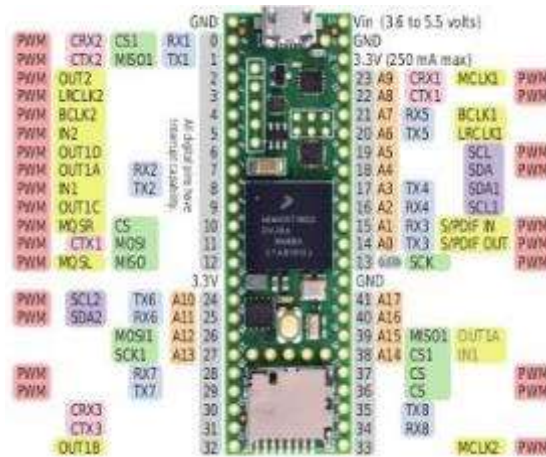


Figure 2-12 Teensy 4.1 55 Pins Configuration

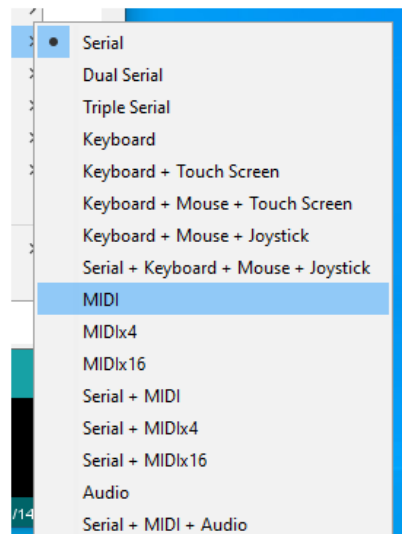


Figure 2-13 Teensy 4.1 USB Type Communication Choices



Figure 2-14 USB Host port on Teensy 4.1



Figure 2-15 Ethernet Kit on Teensy 4.1

B) Micro-ROS

Micro-ROS is a robotic software solution for robotic applications with resource-constrained systems such as a microcontroller. Devices that have minimum real-time or no operating system, battery-powered, wireless low bandwidth connections, and slow operation with sleep periods. (*micro-ROS*, no date) Micro-ROS has its architecture built inspired by ROS2 as seen in Figure 2-16 which will help the operations of robot in sensors and actuators. Micro-ROS can run on different hardware and software as it has a versatile build system which makes Micro-ROS optimal for RTOS (Real-Time Operating System) enabling it for any time-critical requirements which involve tasks that demand time-deadlines. (Finnocchiario, 2020)

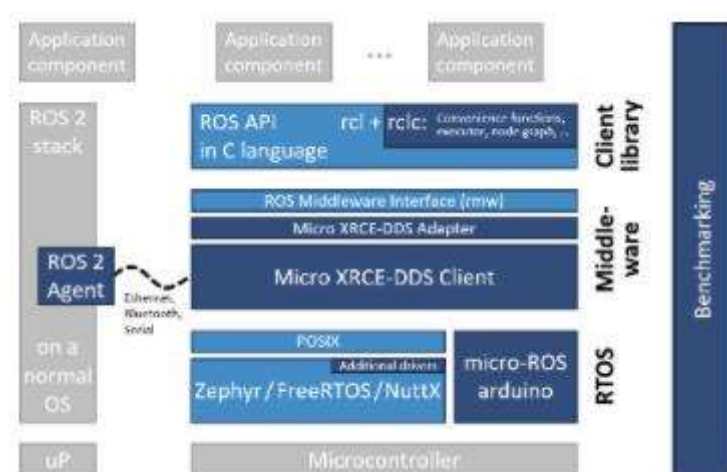


Figure 2-16 Micro-ROS Architecture

C) Arduino

Arduino, an open-source platform consisting of both a microcontroller and IDE (Integrated Development Environment) software, is used for building electronics-related projects. Arduino allows ease when uploading the code unto the board from your software which is only using a USB cable. The Arduino IDE itself is a simplified version of C++ which is beginner-friendly and more than thousands of packages and libraries can be downloaded in Arduino IDE for usage. (*What is an Arduino? - learn.sparkfun.com*, no date)

Arduino boards use various kinds of microprocessors and controllers varying from Atmel AVR (8-bit), ARM Cortex-M (32-bit), and Intel (32-bit). The boards are equipped with sets of digital and analog input/output pins which vary the amount for different types of board. Most Arduino boards have a 5V regulator, which limits the voltage coming in or out at 5V as well as a crystal oscillator or ceramic resonator used to oscillate and make timings for the Arduino program. They have their own boot loader inside it which simplifies uploading program into the flash memory of the Arduino. Most Arduino hardware, as seen in Figure 2-17, currently are produced by SparkFun Electronics and Adafruit Industries. (*Arduino - Products*, no date)



Figure 2-17 Varieties of Arduino Board Types

As Arduino is simple and easily accessible, it is becoming a trend in electronics-related projects and has been used for thousands of different projects and applications. Guides and tutorials are easily accessible not only from the Arduino website but also from videos on Youtube. In detail, here are the reasons why Arduino is preferred over other microcontrollers:

- 1) Inexpensive, Arduino is considered one of the cheapest microcontrollers costing around Rp. 600.000 each.
- 2) Cross-platform, Arduino IDE can run on Windows, Macintosh OS, and Linux, unlike most microcontrollers which are limited to Windows.
- 3) Simple, clear programming environment, with Arduino IDE, both beginners and professionals can bring out the maximum of Arduino as it has many packages, and libraries and easy to use as well.
- 4) Open-source and extensible hardware, Arduino allows experienced circuit designers as well as inexperienced users to make their own custom version of the module on the board directly or even onto a breadboard.

(*What is Arduino?* | *Arduino*, no date)

2.1.4 Robot Localization

Robot localization or localization refers to how a robot determines its position in an environment. Localization is one of the most fundamental tasks that a robot should have as the knowledge of the robot's location is essential to decide future actions of the robot. For robot localization, a map of the environment that can be represented and the robot equipped with sensors are both required to accomplish it. The position of the robot should be able to visualize on that represented map to continue for further actions. (Webster, Huang and Dissanayake, 2016)

Localization problem lies in estimating the robot's position and orientation within the map using the sensors equipped in the robot. Usually, sensors receive noises in their data taking which might cause inaccuracy in the localization process. Several filters can be used such as EKF (Extended Kalman Filter) for Landmark-based map as seen in Figure 2-18 and Particle Filter for Grid map as seen in Figure 2-19. Another problem in localization is when a represented map of the environment is not available. This requires the robot to do mapping of its environment and localization all on its own in

that unknown environment. SLAM (Simultaneous Localization and Mapping) is a solution for localization on an unknown map that requires vision-based sensors such as LiDAR or cameras to get the environment details. With the help of EKF and least-squares optimization, applying SLAM for localization using cameras or LiDAR is reliable for both indoor and outdoor localization. (Webster, Huang and Dissanayake, 2016)

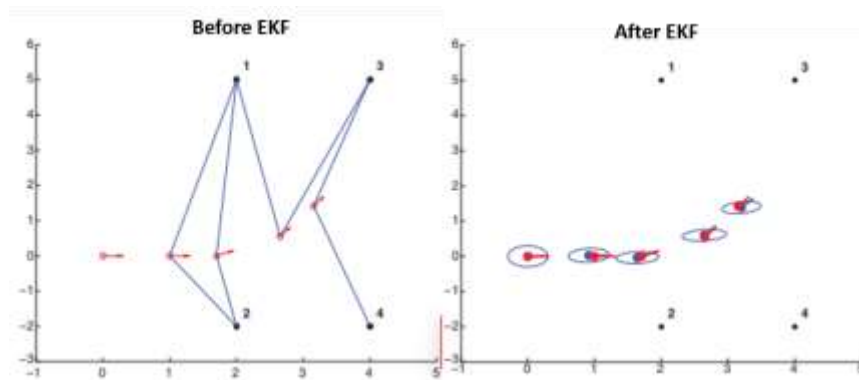


Figure 2-18 Extended Kalman Filter Localization

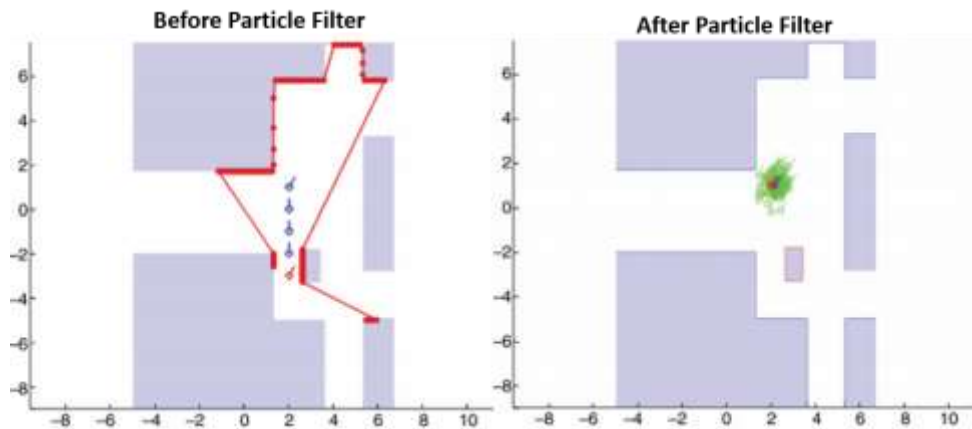


Figure 2-19 Particle Filter Localization

2.1.5 Object Recognition

CV (Computer Vision), one of the biggest branches in AI (Artificial Intelligence), is a technology that develops techniques that help computers “see” by gaining high-level understanding from digital images. The goal is so that the computer can replicate human vision and use the information to perform human activities. (Ballard and Brown, 1982)

Object recognition is the task of identifying objects on an image. Object recognition methods could be categorized into two main types which are the one-stage method that prioritizes speed and the two-stage method that prioritizes detection accuracy. (Ganesh, 2019) In object recognition, there are several different kinds of tasks as seen in Figure 2-20 which are image classification that is used to label a class of an object, and object localization that is used to draw bounding boxes around the objects. Object detection is the combination of both tasks, image classification, and object localization, in which it locates the presence of objects with bounding boxes then classifies the located objects. Which then object detection can be derived into object segmentation where detected objects could be highlighted in specific pixels instead of making bounding boxes. In the end, only two object recognition tasks are often used which are single-object localization, where position and scale of one instance of each object class as seen in Figure 2-21, and object detection, where position and scale of every instance of each object class as seen in Figure 2-22. (Brownlee, 2019)

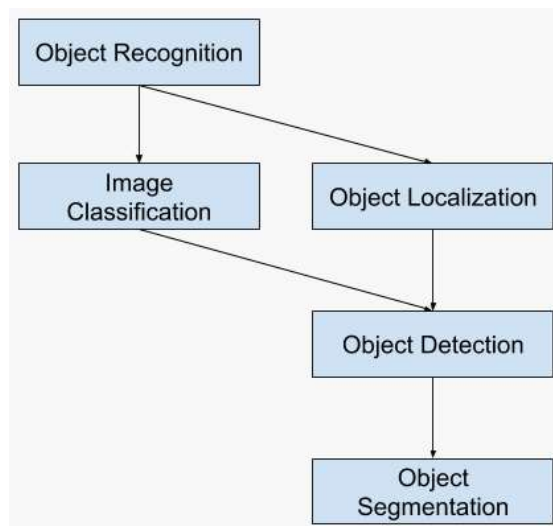


Figure 2-20 Object Recognition Tasks

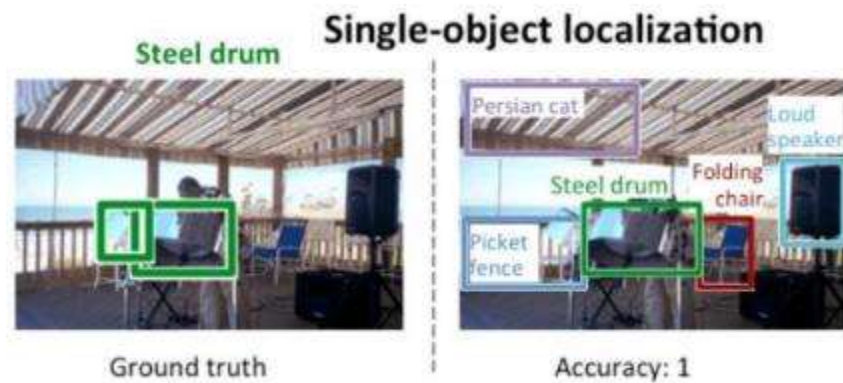


Figure 2-21 Single-Object Localization Output

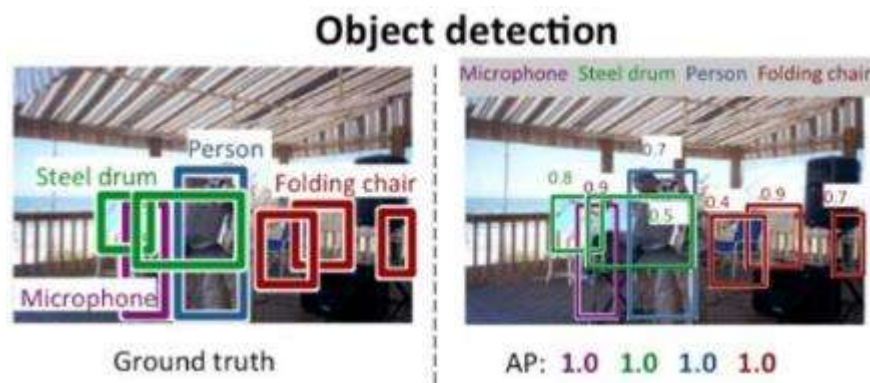


Figure 2-22 Object Detection Output

YOLO (You Only Look Once) is a popular method of object recognition developed by Joseph Redmon. YOLO despite being less accurate than R-CNN (Region-Based Convolutional Neural Network), one of the most accurate object recognition methods, is faster than R-CNN with a speed of 45 up to 155 frames per second. (Brownlee, 2019) YOLO has up to four versions which are YOLO, YOLOv2, YOLOv3, and YOLOv4. Here as seen in Figure 2-23 is the speed comparison of YOLOv3 with other detectors. (Redmon, no date)

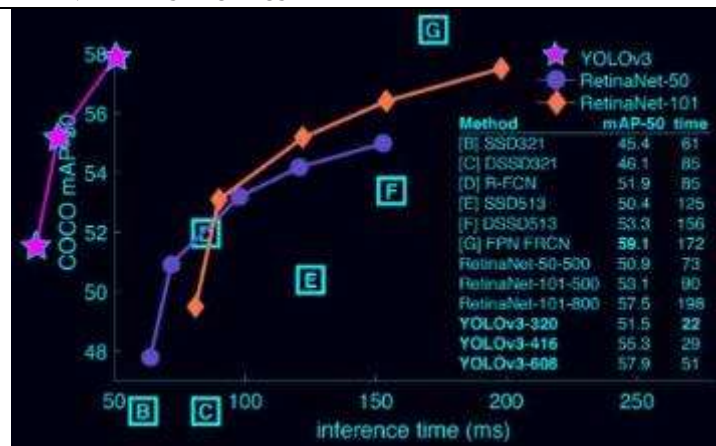


Figure 2-23 YOLO Speed Comparison

YOLO algorithm utilizes single convolutional network for object detection which instead of taking the image bit by bit to be processed, YOLO takes the whole image, split it into grids, pass it into a neural network to create bounding boxes and classify object with its pre-trained datasets as seen in Figure 2-24. YOLO implements IoU (Intersect over Union) and NMS (Non-Maximum Suppression) to calculate bounding boxes. IoU is the accuracy between the machine's predicting bounding box compare to the actual object's bounding box as seen in Figure 2-25. NMS is used to localize the concept of focus of an object detection where object detection might obtain several detections near the actual location. NMS filters the candidates of the detection by choosing the highest probability of the boxes that determine the same object as seen in Figure 2-26. (Velasco, 2019)

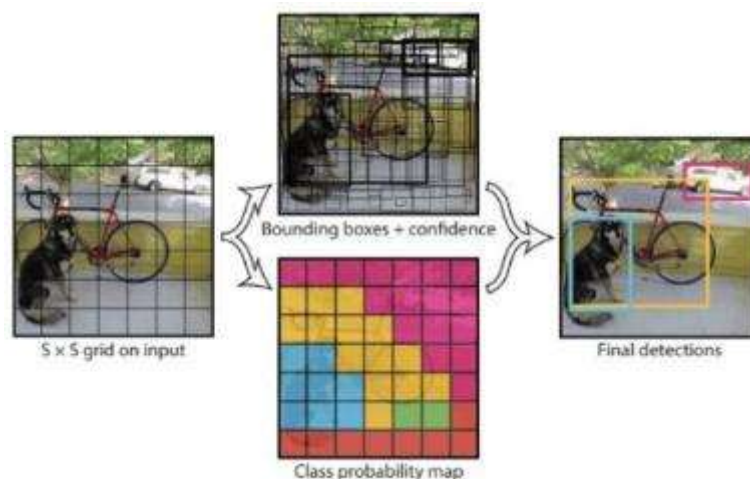


Figure 2-24 YOLO Algorithm Process

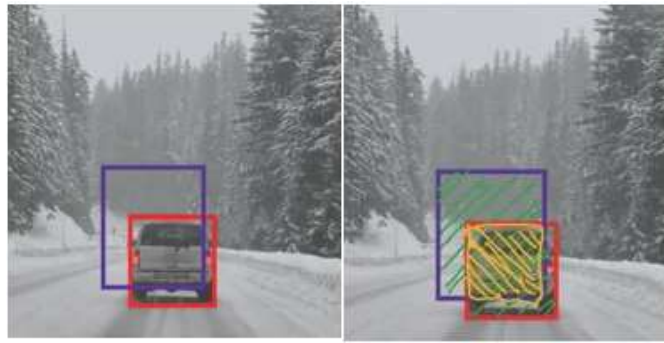


Figure 2-25 IoU on a Car (Purple: Machine; Red: Actual)

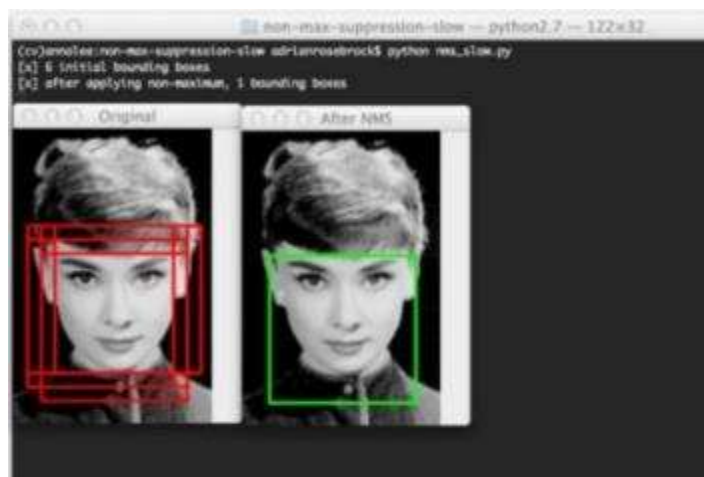


Figure 2-26 NMS on Human Face

By utilizing both IoU and NMS, predictions on various objects in an image can be done fast as it sees the entire image during training and tests its information based on datasets, it shows the classes in the images it sees. Another good example to apply YOLO is in self-driving cars as it requires the car to detect other cars and cyclists quickly as seen in Figure 2-27. The faster the object is detected, the faster can the self-driving cars react to avoid a collision. The negative side of YOLO is it is unable to detect too close or too small as seen in Figure 2-28. (Velasco, 2019)

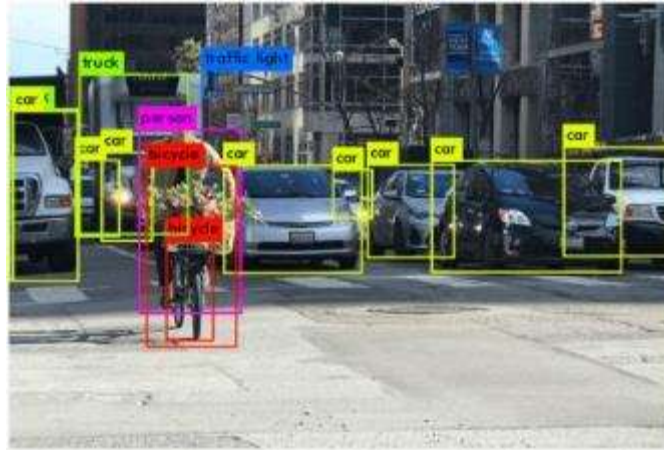


Figure 2-27 YOLO for Self-Driving Cars



Figure 2-28 Incompleteness of YOLO to Identify Small Objects

2.1.6 Sensors for Localization and Object Detection

Several sensors can be used for localization depending on how the localization is done. Most localizations are done while observing its environment using LiDAR and its own movement using IMU and rotary encoder. Other alternatives for localization are by extracting information from an external source such as GNSS (Global Navigation Satellite System). (Webster, Huang and Dissanayake, 2016) For object recognition, a simple camera can be used for any CV tasks. In some cases, 3D cameras which can produce depth images might be more reliable than normal 2D cameras.

A) RTK-GNSS

Satnav (satellite navigation) system is a system of satellites that can give autonomous global positioning around the globe and allow the receivers to obtain its location in longitude, latitude, and altitude using time signals transmitted using radio waves from satellites. GNSS is the constellation of satellites that provide

autonomous geo-spatial positioning with global coverage, thus providing satnav system. GNSS is the standardized term for all satellites that provide satnav system from different countries. (Venezia, no date) Here is the list of GNSS:

- 1) GPS (Global Positioning System) by United States of America (32 satellites)
 - 2) BDS (BeiDou Navigation Satellite System) by China (35 satellites)
 - 3) Galileo by European Union (24+ satellites)
 - 4) GLONASS (*Globalnaya Navigazionnaya Sputnikovaya Sistema*) by Russia (24+ satellites)
 - 5) IRNSS (Indian Regional Navigation Satellite System) by India (7 satellites)
 - 6) QZSS (Quasi-Zenith Satellite System) by Japan (4 satellites)
- (GPS.gov: Other Global Navigation Satellite Systems (GNSS), no date)

The performance of GNSS can be assessed by four criteria, which are:

- 1) Accuracy: the difference between a receiver's measured and real position
- 2) Integrity: the capability of the system to provide stable data during anomaly condition
- 3) Continuity: the capability of the system to function continuously without interference
- 4) Availability: the percentage of time when all accuracy, integrity, and continuity are fulfilled.

(What is GNSS? | European Global Navigation Satellite Systems Agency, no date)

RTK-GNSS (Real-Time Kinematic Global Navigation Satellite System) is a technique that elevates the usage of GNSS which improves the accuracy of the satnav system to acquire geospatial positions. RTK-GNSS uses the concept of reducing and removing errors by applying base station and rover pair as seen in Figure 2-29. RTK-GNSS has an accuracy of centimeter-level positioning, from 1 cm to 25 cm. (*Real-Time Kinematic (RTK)* | NovAtel, no date)

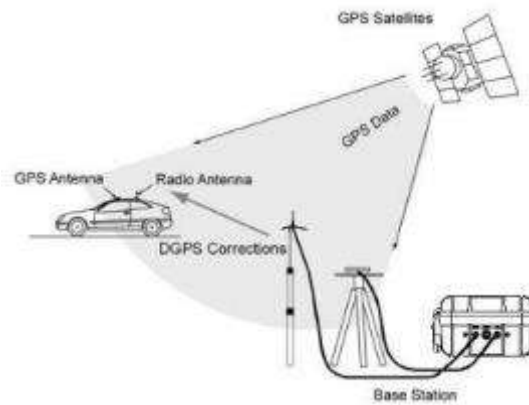


Figure 2-29 RTK-GNSS Working Concept

RTK-GNSS increases the accuracy by using a fixed base station that sends out corrections to a moving receiver wirelessly. The technique measures the carrier phase of the satellite signal, the number of carrier cycles between the satellite and the rover station, which is then subjected to statistical methods to align the phase of the signals. By applying this technique, it will eliminate the majority of ordinary GNSS errors as seen in Figure 2-30. This alignment algorithm is called the ambiguity resolution which consists of acquisition, ambiguity ‘float’, and ambiguity ‘fixed’. The accuracy can reach 20 – 75 cm in ‘float’ mode and 1 – 2 cm in ‘fixed’ mode. Corrections are sent every interval of one second. (VBOX Automotive - How does it work? - RTK, no date) The rover station uses this ambiguity resolution algorithm to get the position-dependent on the base station and the differential corrections. Not only does the rover station need to accept clear satnav signals but also the base station to achieve high accuracy. (Real-Time Kinematic (RTK) | NovAtel, no date)

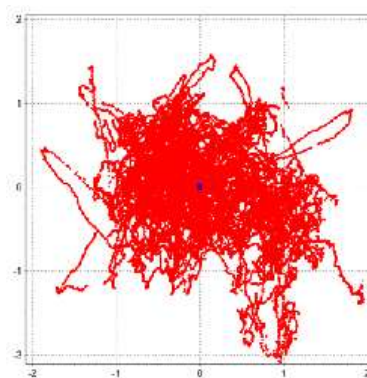


Figure 2-30 RTK-GNSS and Ordinary GNSS Data Comparison

B) LiDAR

LiDAR (Light Detection and Ranging) is an active remote sensing system that utilizes laser light by calculating the distances by illuminating the target with a laser and measure the reflection in a 360-degree rotation like a radar. As an active system, the system itself generates energy which is light, emitted from rapidly flying laser. LiDAR measures the time it takes the emitted light to travel to the surface and back using an optical detector (photodetector) with a high-frequency rate. The time is then converted into distance traveled which may get results such as in Figure 2-31 where LiDAR is used to record 3D maps of a forest. LiDARs data could be collected in two ways, which are discrete return and full waveform. Discrete return records individual (discrete) points of peaks in the waveform curves up to 1-4 points from each laser pulse. Full waveform records the whole return of the light energy, which captures more information thus requires a more complex process but providing a more concrete result. (Wasser, 2020)

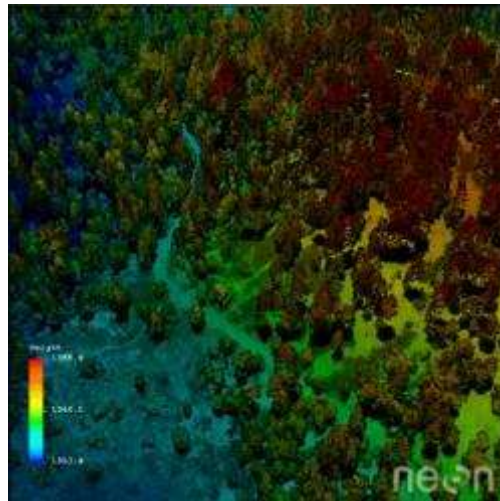


Figure 2-31 LiDAR 3D Map Output on Forest

C) 3D Camera as Vision Sensor

A camera is hardware used to capture an image, either conventionally or digitally. A conventional camera capture images by capturing the shape of objects into a film made of light-sensitive materials by capturing how much light is coming from the object. A digital camera captures an image using the same concept as a conventional camera but instead of using films, it uses some sort of solar panel consisting of

sensors capable to detect colors for each pixel. (*How Does A Camera Work?*, no date)

Vision sensor uses to determine the presence, orientation, and class of objects. Camera as vision sensor is expected to operate as how human sees objects. Cameras use visual data returned from the optics in the lens directly which ease the software to do analysis. With neural networks and object recognition algorithms, cameras can capture images quickly and provide information from the object recognition algorithm continuously as seen in Figure 2-32 is the example of Tesla using cameras as vision sensors. Cameras do not have a long detecting range and unable to accurately measure distance, especially in bad weather. (Tabora, 2020)



Figure 2-32 Camera as Vision Sensor on Tesla

A 3D stereo camera is a camera that can capture an image with its detailed features such as edges, corners, and depths of an image. These depth images can be useful in navigation or object detection applications which are collected as point cloud data, which is the same input given from LiDAR. (Chen *et al.*, no date)

D) Ultrasonic Sensor

An ultrasonic sensor is a device that measures the distance of an object by emitting ultrasonic sound waves and capturing the reflected sound waves. It consists of two main components, they are the transmitter which emits the sound waves using piezoelectric crystals, and the receiver which receives the sound waves that are reflected by the object. To measure the distance of objects using ultrasonic sensor, it measures the time it takes for the sound waves to travel from the transmitter until

it is received by the receiver as seen in Figure 2-33 by using the formula $D = \frac{1}{2} * T * C$ (D = distance, T = Time traveled, C = Speed of Sound).

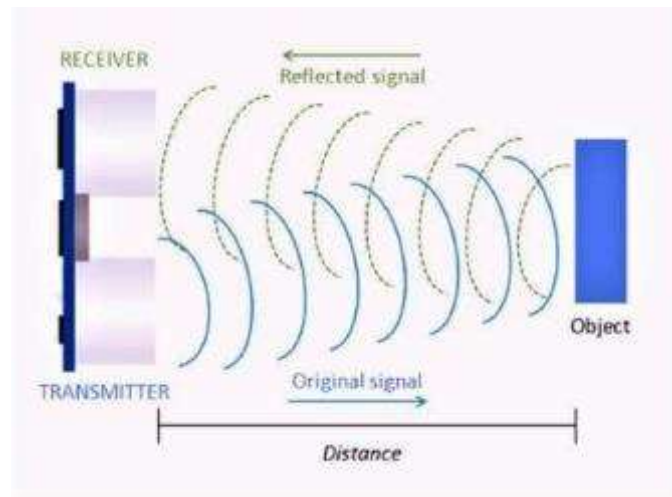


Figure 2-33 Ultrasonic Sensor Working Principle

In SDC applications, ultrasonic sensors are mostly used as proximity sensors which can be found in automobile self-parking or anti-collision safety systems. For AMR, it can be used for obstacle detection systems which supplements the usage of LiDAR or 3D camera's point clouds for obstacle avoidance.

2.1.7 Odometry, IMU and Rotary Encoder

Odometry is the method of using data from motion sensors to estimate the change in position and its distance traveled. In a robot, odometry computes the speed relative to the time to determine its position relative to the environment the robot is in. Odometry is the lowest level localization as it uses its own robot to measure its position in a map without checking on external sensors. The robot kinematic will become the mathematical model and calculation to determine the odometry as seen in Figure 2-34. For odometry, there are two crucial sensors required which are wheel encoder which is rotary incremental encoders, and IMU (Inertial Measurement Unit). (Ben-Ari *et al.*, 2018)

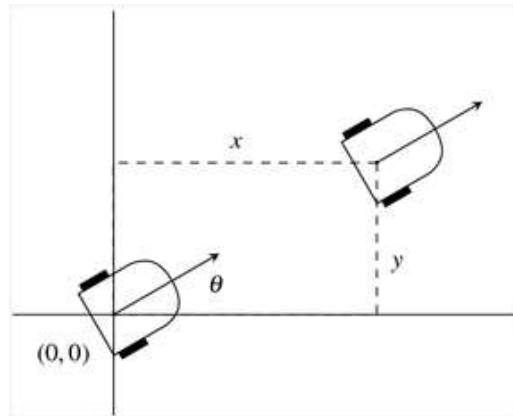


Figure 2-34 Robot Kinematic for Odometry Calculation

A rotary incremental encoder is a device that converts angular motions of a shaft into an analog or digital code so that the system can identify the position or velocity of the robot. As an incremental encoder does not record a base, it can only detect changes from one point to another point, which makes incremental encoder requires a reference device to calculate its motion. An incremental encoder is built to issue two pulses both with a phase difference of 90 degrees, which are pulse “A” and “B”. The phase difference could determine whether the device is moving clockwise or counterclockwise. An incremental encoder uses a rotating disk that has two segments for channels “A” and “B” with each segment has transparent or opaque parts, and a photodetector equipped with a laser. The laser will be shot into the segments and the photodetector will capture the light of the laser as seen in Figure 2-35. If the laser is shot into the transparent part of the segment, the light will be captured by the photodetector, resulting in a high input in the pulse. If the laser is shot into the opaque part of the segment, the light will not be captured by the photodetector, resulting in a low input. These combinations of high and low input will give pulse determining the rotary movement of the encoder which can be translated to acquire the distance traveled. (Craig, no date)

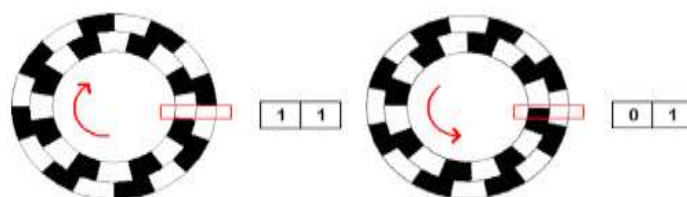


Figure 2-35 Rotary Incremental Encoder Working Principle

An IMU is a device where three types of sensors are combined to be able to obtain the absolute position and orientation of an object. IMU uses the concept of 9-DOF (nine degrees of freedom) using an accelerometer, magnetometer, and gyroscope. An accelerometer measures velocity and acceleration which is used to get the position, a magnetometer measures cardinal direction respective to the Earth's magnetic field which is used to get where the object is pointing to, and a gyroscope measures rotation and the rotational rate which can determine position and orientation. (*What is IMU? Inertial Measurement Unit Working | Arrow.com | Arrow.com, 2018*)

2.1.8 Mapviz Visualization

Mapviz is a visualization tool created by SWRI-Robotics (Southwest Research Institute Robotics) based on ROS that focuses on visualizing 2D data. Mapviz consists of many plug-ins that aid outdoor map robot applications such as tilemaps on Earth, GNSS localization, LiDAR laser scans, camera image captures, and even odometry as seen in Figure 2-36. As Mapviz is based on ROS, the communications between the topics for GNSS, LiDAR, and other sensors could be done through subscribing from ROS topics using nodes.



Figure 2-36 Mapviz Visual User Interface

2.1.9 Linorobot Software

Linorobot is an open-source library for several kinds of robot which is emphasized to provide students, developers, and researchers to experience different applications of robots on ROS. It supports different kinds of robot bases such as 2WD, 4WD, Mecanum Drive, Skid Steering, and Ackermann Steering. It uses commercial-friendly hardware

such as IMU, rotary encoder, motor H-Bridge, and many others. Linorobot provides flexibility for its hardware so any different sizes of robot can still use the software without rewriting it or at the very least, minimum changes.

Linorobot is powered by robotics middleware ROS as it has ROS's most robust and solid software development framework to create a robot. It has its completed set of libraries for microcontroller level which consists of IMU readings, rotary encoder readings as well as motor actuation. On the ROS side, it will read the required velocity then send it to Teensy to process the required input. It can also calculate the robot odometry from the IMU headings and rotary encoder readings. For indoor applications, Linorobot provides 2D maps using SLAM (Simultaneous Localization and Mapping). It also provides LiDAR laser scan libraries to use in addition with SLAM. After obtaining robot odometry as seen in Figure 2-37 and a map of the environment as seen in Figure 2-38, Linorobot has AMCL (Adaptive Monte Carlo Localization) to locate the position of the robot of the map given from SLAM. Thus, the robot can be given target positions for it to navigate automatically as seen in Figure 2-39 with an addition of LiDAR and 3D camera for object detection in which both libraries also provided in Linorobot.

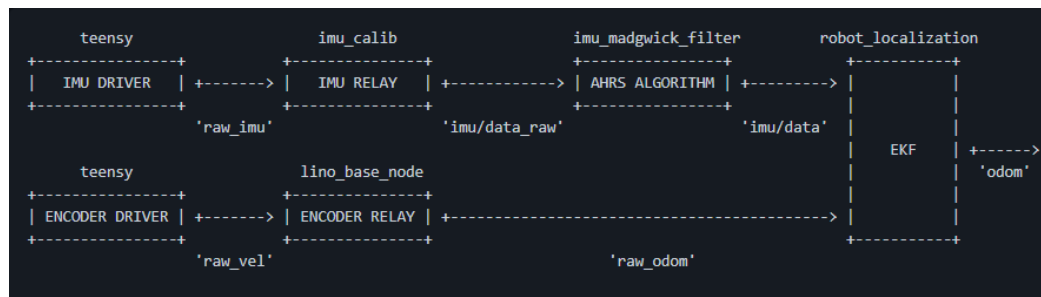


Figure 2-37 Linorobot Odometry from IMU and Rotary Encoder

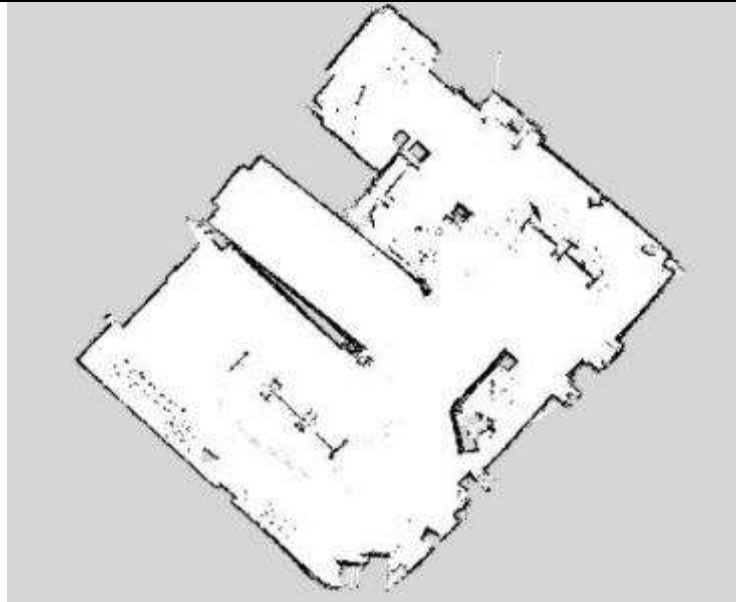


Figure 2-38 Linorobot SLAM Map Output

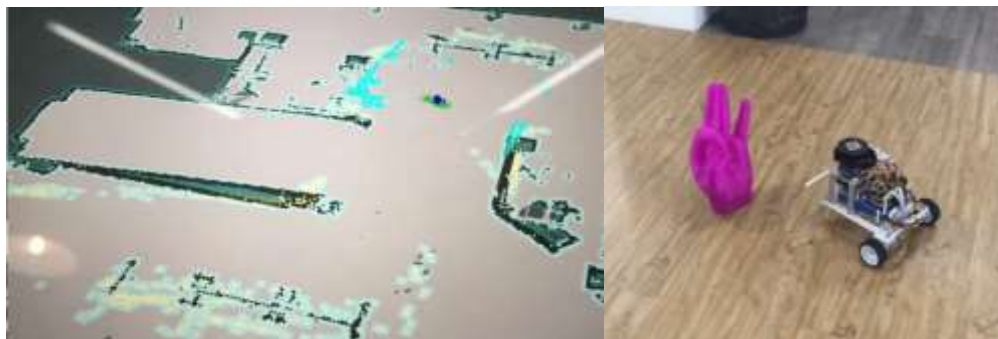


Figure 2-39 Linorobot Navigation Visualized on RViz

2.2 Existing Research and Previous Studies

2.2.1 Self-Driving and Driver Relaxing Vehicle

This research in which GNSS is used for localization with Google Maps as its base map on an AMR based on idea implementing in SDC. The goal of the research is to create a relaxing driver vehicle and an automated vehicle that is capable of changing its destination, unlike Google's SDC, Waymo, which has a static and fixed destination. The methodology of the research as seen in Figure 2-40 involves using two mobile robots which act like a car, one car (front) knows the route and the other car (follower) does not know the route. The follower car will need to follow the front car automatically. Arduino is used as the microcontroller for the research. The front car uses GPS (Global Positioning System) and GSM, a module required to connect to the

internet. The follower car uses GPS, GSM, ultrasonic sensor, motor, and compass. The scenario can be seen in Figure 2-41. This research concluded successfully as seen in Figure 2-42 as the front car could smoothly navigate to its destination while the following car using GPS, could follow the track and move towards the front car. (Memon *et al.*, 2016)



Figure 2-40 Test Robot Block Diagram (Existing Research 1)



Figure 2-41 Testing Scenario (Existing Research 1)



Figure 2-42 Research Results (Existing Research 1)

2.2.2 Adaptive Tuning Steering Control for Self-Driving Car

This thesis was written by Felicia Kristy, a batch 2015 Mechatronics student from Swiss German University. The objective of the thesis is to build an attachable module for the

car steering wheel that will enable the car to navigate autonomously as a Level 1 autonomous vehicle with its navigation based on RTK-GNSS localization without including obstacle avoidance. (Kristy, 2019)

The system of the car steering wheel module consists of three modes, which are monitor, record, and navigation modes. In monitor mode, the system will be fed data from the RTK-GNSS required for the adaptive tuning. In record mode, the system will record the data from RTK-GNSS into a “.txt” file. In navigation mode, the system will choose the recorded file and use the coordinates to navigate the car to reach the target. A safety system is added when the navigation does not work as expected, the driver can override and take over the steering wheel, which makes the system exit navigation mode. The system flowchart is as seen in Figure 2-43. (Kristy, 2019)

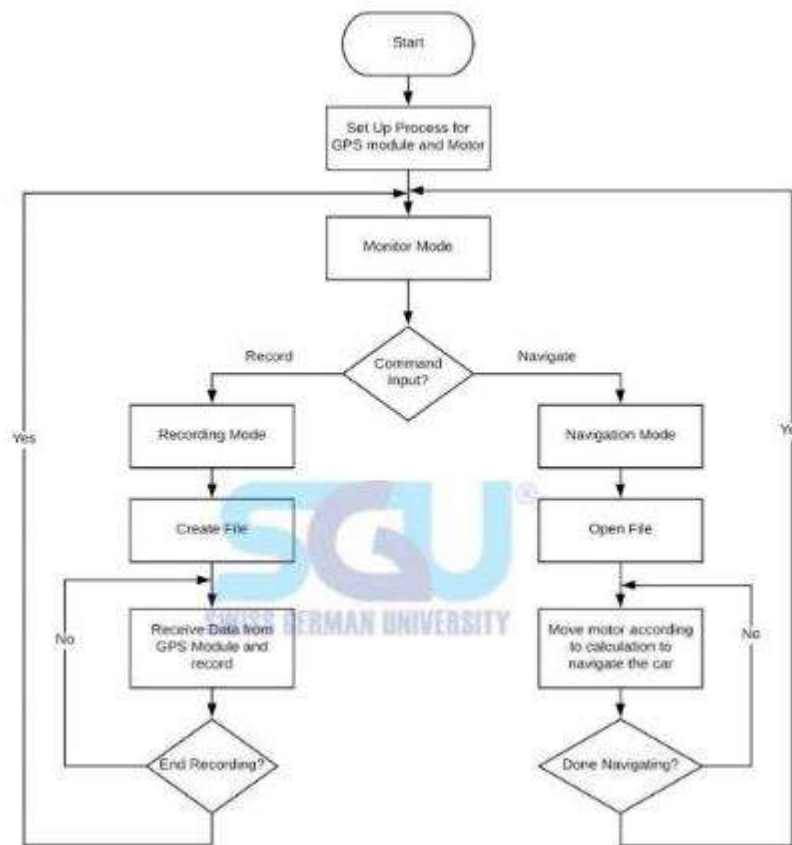


**Figure 2-43 Steering Wheel Module Program Algorithm
Flowchart (Existing Research 2)**

The thesis is completed successfully by testing twice on the same route to ensure the system reliability and repeatability, which the result plotted together as seen in Figure

2-44. The data of the longitude and latitude from the RTK-GNSS is recorded and compared. From this thesis, it can be concluded that the RTK-GNSS module is reliable with a deviation of only 276.2mm. The adaptive tuning system works well with an absolute error of 0.00354 radians from the mathematical model and 0.0004 radians from the car. The navigation of the system has good repeatability with the standard deviation compared to the target route is less than 58cm with the difference between the first test and the second test having the standard deviation of 7.2cm with the lowest of 1.0cm. (Kristy, 2019)

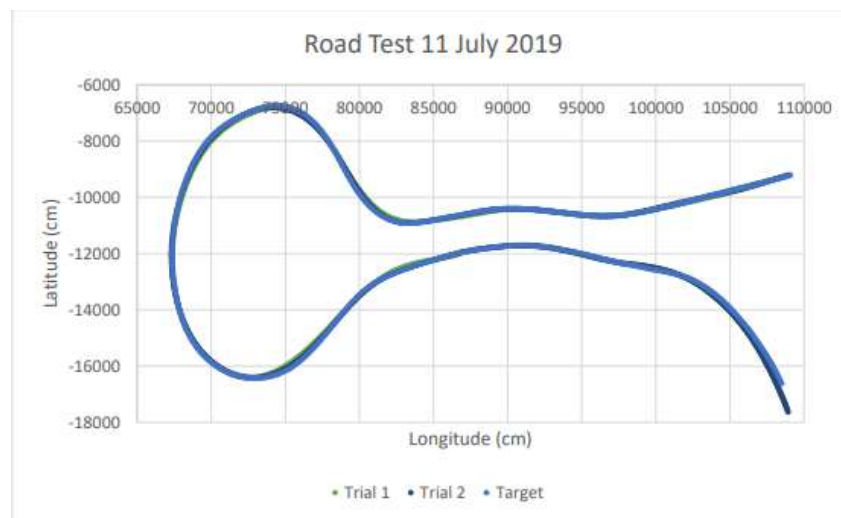


Figure 2-44 Road Test Navigation Comparison (Existing Research 2)

2.2.3 Prototype Development of Autonomous Mobile Robot with Indoor Navigation to Deliver Goods

This thesis was written by Wilvan Gunawan Wirjono, a batch 2016 Mechatronics student from Swiss German University. The objective of the thesis is to develop Autonomous Mobile Robot based on ROS using IMU and encoder as odometry, and LiDAR for navigation with a 3D stereo camera for obstacle avoidance equipped with an ultrasonic sensor for safety. (Wirjono, 2020)

In this thesis, the robot has two processes which are mapping and navigation as seen in Figure 2-45. When the robot is started, it will check whether the system has a map of the environment. If the map does not exist, the robot will apply SLAM (Simultaneous Localization and Mapping) to acquire the map. Then, the robot will do the navigation to reach the goal. Localization in this thesis is done by LiDAR, 3D camera, and

odometry. Data from the 3D camera is converted into laser scans similar to LiDAR which is used to detect the location of obstacles. (Wirjono, 2020)

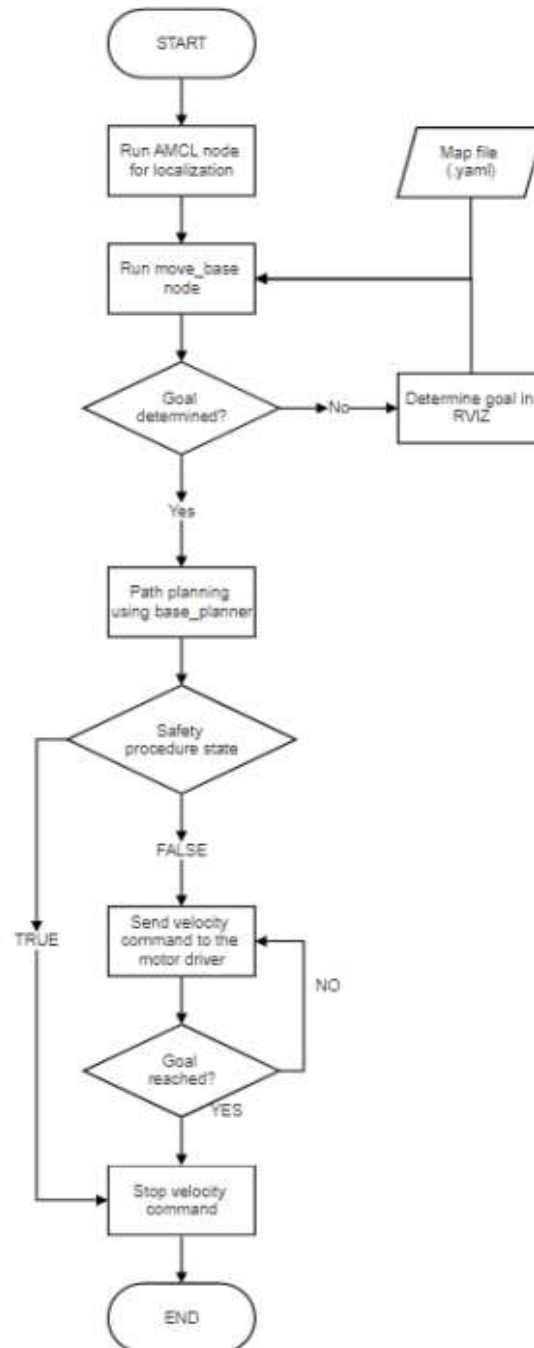


**Figure 2-45 Mapping and Navigation System Process
Flowchart (Existing Research 3)**

The thesis is conducted successfully as the robot can successfully plot the map of the environment using laser scans and navigate to reach the target goal. From the thesis,

some conclusions are made such as LiDAR could not detect opaques or non-reflective objects, the 3D camera used, and XBOX Kinect has limitations with maximum detection range in X-axis at 0.43m to 1.22m and Y-axis at 0.4m. (Wirjono, 2020)

2.2.4 Outdoor AMR using Robot Operating System (ROS) and GNSS

This thesis was written by Edrick, a batch 2016 Mechatronics student from Swiss German University. The objective of the thesis is to build an autonomous mobile robot that can transport objects between warehouses by using GPS, odometry, and a camera for navigation. (Edrick, 2020)

The thesis uses GNSS for its localization and uses Mapviz visualization to visualize the robot's position and its environment. This thesis also uses a 3D camera to obtain depth images to convert into laser scans which are used for object detection. By integrating both odometry and GNSS for localization, more accurate positioning can be obtained relevant to the map as seen in Figure 2-46. The data that are taken measures the latitude and longitude data from the GNSS for each waypoint in actuation towards the final goal. (Edrick, 2020)



**Figure 2-46 GNSS and Odometry Localization on Mapviz
Visualizer (Existing Research 4)**

This thesis is concluded successfully as the data taken from the GNSS which are the latitude and longitude under fixed condition is reliable with one meter accuracy and will achieve higher accuracy with the help of odometry. For digital signal processing, the usage of EKF (Extended Kalman Filter) is proven functional for outdoor navigation as it is used to filter noise from odometry and GNSS in this thesis. (Edrick, 2020)

2.2.5 A MEMS-based Smart Sensor System for Estimation of Camera Pose for Computer Vision Applications

This research was written by Dominik Aufderheide and Werner Krybus from South Westphalia University of Applied Sciences Soest as well as Dennis Dodds from the University of Bolton, School of the Built Environment and Engineering. The objective of this research is to create a smart sensor system based on a micro-electromechanical inertial sensor to aid computer vision applications to get accurate camera position.

The research uses an IMU which contains accelerometers, gyroscopes, and magnetometers with each has three degrees of freedom. The data from the sensor is sent through a microcontroller (as Analog-Digital Converter) into a PC as the architecture as well as the IMU configuration seen in Figure 2-47. As all electrical sensors will have different error sources, calibration and signal conditioning are required to reduce errors such as misalignment of sensitivity axes, absolute zero biases, measurement noises, and internal scaling factors of the data. The result, as seen in Figure 2-48 is the modeling of the sensor architecture. Usage of filters, specifically, Kalman filters is required to remove and reduce these errors as much as possible. Discrete Kalman Filter (DKF) and Extended Kalman Filter (EKF) are used in this research in which the sensor data from the IMU are filtered with DKF and EKF before combined to obtain the position of the camera as seen in Figure 2-49.

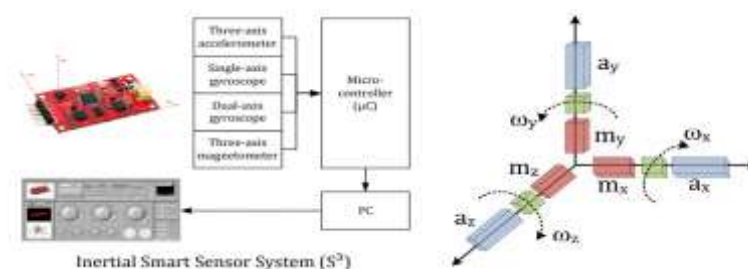


Figure 2-47 Smart Sensor System and IMU Configuration
(Existing Research 5)

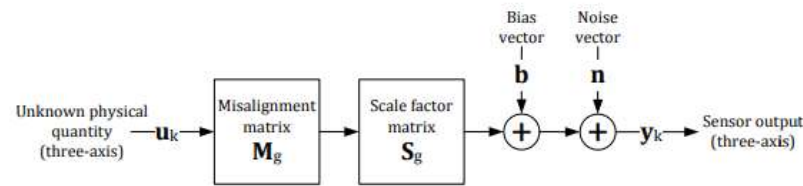


Figure 2-48 General Sensor Modelling Architecture
(Existing Research 5)

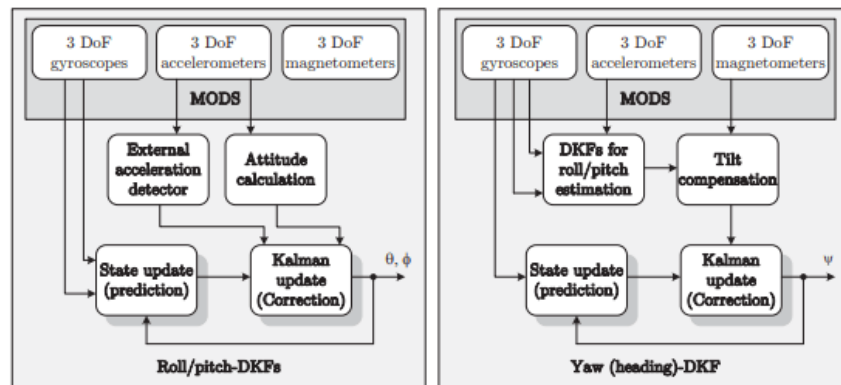


Figure 2-49 DKF Applications on IMU Sensors (Existing Research 5)

This research was concluded successfully by taking different kinds of tests, which are gyroscopes only, complimentary filtering (CF) which uses gyroscopes, an accelerometer with the signal source filtered, and weighting filter (EST) which uses the combination of gyroscope and accelerometer measurements by using fixed weights. The test is also taken for three different motion patterns, rotation around a single axis, consecutive rotation around two axes, and simultaneous rotation around two axes with all results as seen in Figure 2-50. (Aufderheide, Krybus and Dodds, 2011)

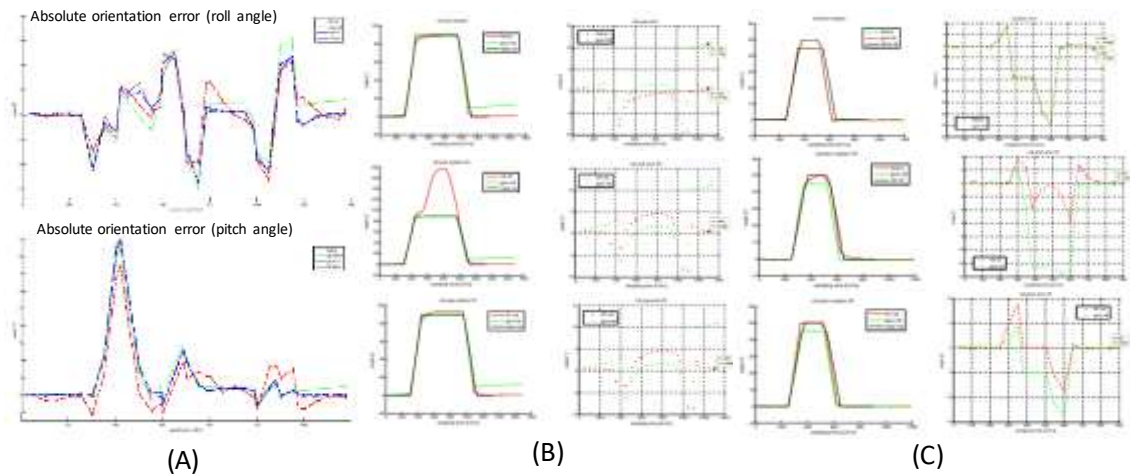


Figure 2-50 Research Results (A. Rotation on Single Axis, B. Consecutive Rotation around Two Axis, C. Simultaneous Rotation around Two Axis)
(Existing Research 5)

2.3 Thesis Objectives

In this thesis project, there are several objectives to achieve:

1. To develop a scaled-down SDC as an AMR
2. To integrate sensors such as IMU and encoder for robot odometry, RTK-GNSS, and LiDAR to detect surroundings to perform localization and navigation
3. To perform object detection using LiDAR and ultrasonic sensor for obstacle avoidance