User Manual for Radar-Camera Sensor Fusion

1. Introduction

This manual guides the Radar-Camera Sensor Fusion system's setup, operation, and resources, which are designed to enhance vehicle safety through advanced collision warning features.

2. System Components

2.1 Hardware

The hardware components of the Radar-Camera Sensor Fusion system are essential for accurate data collection and processing. Each component plays a crucial role in ensuring reliable performance and safety. This section outlines the essential hardware elements, including the radar sensor, camera, and processing unit, highlighting their specifications and functions in the overall system. Proper integration and configuration of these components are critical for achieving optimal sensor fusion and effective collision warning capabilities.

- **Hardware Requirements**
    - ➔ Orin
    - ➔ Radar
    - ➔ Basler Camera
    - ➔ GNSS Receiver
    - ➔ RTK
    - ➔ UPS Battery
    - ➔ Micro Auto Box

2.1.1 Orin

The NVIDIA® Jetson AGX Orin™ Developer Kit enables the development of full-featured AI applications for products based on Jetson Orin modules. It includes a high-performance, power-efficient Jetson AGX Orin module and can emulate the other Jetson Orin modules.

Requirements

- NVIDIA Jetson AGX Orin module (included in the box)
- Reference carrier board (included in the box)
- Wi-fi module (included in the box)
- USB Type-C power supply (included in the box)
- USB Type-C to USB Type-A cable (included in the box)
- Internet connection
- A PC monitor with a DisplayPort cable
- USB keyboard and mouse

System Monitor
A system monitor is a hardware or software component that monitors system resources and performance in a computer system.

2.1. 2 Radar

The Radar-Camera Sensor Fusion system integrates advanced sensing technologies to enhance vehicle safety and situational awareness. In our project, we utilize two types of Continental TI radar systems: the ARS430DI (Long Range) and the SRR520DI (Short Range), both of which operate at a frequency of 77 GHz.

Radar Specifications

1. **ARS430DI (Long Range Radar)**:
    - **Range**: Up to 200 meters, allowing for the detection of distant objects and potential hazards.
    - **Field of View (FoV)**: 120 degrees, providing a broad perspective of the surroundings.

    - **Functionality**: Operates as a 3D radar, capturing essential range and azimuth values of detected objects, although it does not provide elevation data.
2. **SRR520DI (Short Range Radar)**:
    - **Range**: Effective up to 100 meters, suitable for detecting nearby obstacles and vehicles.
    - **Field of View (FoV)**: 150 degrees, enabling comprehensive coverage of the immediate environment.
    - **Functionality**: Like the long-range counterpart, it operates as a 3D radar, focusing on range and azimuth without elevation information.

The combined capabilities of the ARS430DI and SRR520DI radars ensure thorough detection of objects within the vehicle's vicinity, making them indispensable for collision avoidance systems. The radar data is formatted in a point cloud structure, which allows for detailed processing and analysis, enabling the fusion of spatial data with visual inputs.

2.1.3 Basler ace 2 (acA1920-40uc) Camera

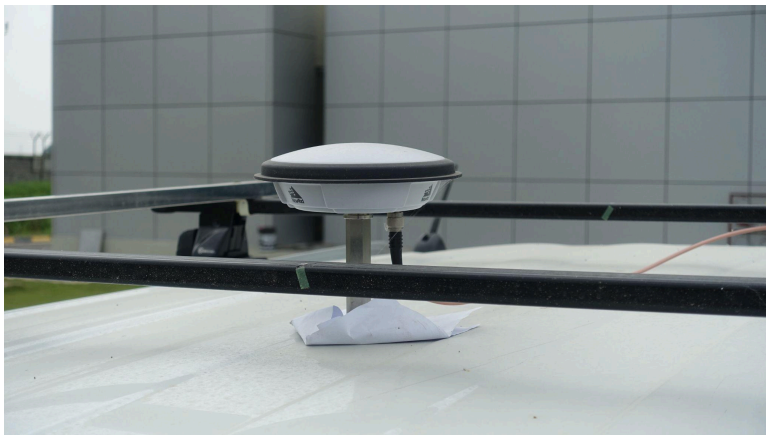
The ace 2 is as flexible as applications are diverse: in a compact housing, it has sensors from Sony and Gpixel with resolutions from VGA to 24 MP. The ace 2 offers excellent image quality with the latest sensor technology (including sensors for different wavelength ranges) and is available with a CoaXPress 2.0, USB 3.0, GigE, or 5GigE interface. It is tailored to fit different vision requirements. Proven Basler reliability and a comprehensive computer vision feature set for standard machine vision applications. These visSWIR cameras are equipped with Sony's SenSWIR sensors with up to 5.3 MP resolutions. These ace 2 cameras are available with sensors from Gpixel or Sony, including the potent 4th generation Sony Pregius S sensors and Sony SenSWIR sensors.
The camera features a resolution of 1936 x 1216 pixels at full settings and 1920 x 1200 pixels at default. It has a Sony IMX392LLR-C progressive scan CMOS sensor with a global shutter, formatted at 1/2.3" and an effective diagonal of 7.9 mm. The pixel size measures 3.45 x 3.45 µm, supporting a frame rate of 164 fps under default conditions. Classified under the ace 2 R product family, this mono camera utilizes a USB 3.0 interface with a nominal maximum bandwidth of 5 Gbit/s. Synchronization options include hardware or software triggers and free running modes. Exposure control can be managed via hardware triggers or programmed through the camera API. Its typical power consumption is approximately 2.8 W at 5 VDC, and it features one opto-coupled input line and two general-purpose I/O lines.

Fig: Basler ace2 Camera

2.1.4 GNSS Receiver

The PwrPak7 is a compact enclosure that delivers a scalable Global Navigation Satellite System (GNSS) with internal storage and INS options.
The PwrPak7 can track all present and upcoming GNSS constellations and satellite signals. It also offers optional integrated INS support for continuous position, velocity, and attitude through short periods of GNSS outage.

The VEXXIS GNSS-800 series of antennas features a patented multi-point feeding network and advanced radiation pattern optimization technology. The multi-point feeding network provides symmetric radiation patterns across all frequencies, delivering excellent multipath rejection and small phase center variation and offset. The result is better carrier phase measurements and a more accurate RTK solution.



Our radiation pattern optimization technology enables low-elevation satellite tracking without sacrificing gain for higher-elevation satellites. With more satellite observations and the low profile of GNSS-800 antennas, users can expect excellent performance even in challenging environments where parts of the sky can be obstructed.

2.1.5 RTK

Real-Time Kinematic (RTK) is an advanced satellite navigation technique that enhances the precision of position data derived from Global Navigation Satellite Systems (GNSS) such as GPS. By utilizing a fixed base station and one or more roving receivers, RTK achieves centimeter-level accuracy in real-time. The base station calculates the difference between its known position and the GNSS data it receives, transmitting correction information to the rover. This process minimizes the errors caused by atmospheric disturbances, multipath effects, and satellite orbit inaccuracies. RTK is widely used in various applications, including surveying, agriculture, autonomous vehicles, and construction, where high precision is crucial.

The SATEL Compact-Proof radio modem is highly beneficial in RTK networks due to its reliable and robust wireless data transfer capabilities. Operating in the license-free frequency ranges of 869 MHz (pan-European) or 865 MHz (Indian), it facilitates the transmission of real-time

correction signals from a base station to roving GNSS receivers. Its IP67 classification ensures durability and protection against dust and moisture, making it ideal for outdoor applications like land surveying, where conditions can vary significantly.

Equipped with a Liquid Crystal Display (LCD), the SATEL Compact-Proof allows users to easily monitor operational parameters such as frequency, channel number, and field strength, enhancing usability in the field. The long-lasting lithium-ion battery supports extended operating hours, which is crucial for continuous RTK operations, particularly in remote areas. By enabling efficient communication within RTK systems, the SATEL Compact-Proof helps achieve high-precision positioning, which is essential for tasks requiring centimeter-level accuracy.

2.1.6 UPS Battery

An uninterruptible power supply (UPS) offers guaranteed power protection for connected electronics. When power is interrupted or fluctuates outside safe levels, a UPS will instantly provide clean battery backup power and surge protection for plugged-in, sensitive equipment.

2.1.7 Micro Auto Box

The dSPACE Micro AutoBox is a compact, powerful platform for real-time testing and development of Advanced Driver Assistance Systems (ADAS). It features high-performance computing capabilities with a multi-core processor, extensive I/O options, and various sensors and communication protocols support. Key specifications include robust connectivity options like Ethernet, CAN, and LIN, allowing seamless integration with vehicle networks and external devices. The Micro AutoBox supports rapid prototyping and model-based design, enabling developers to efficiently implement and test complex algorithms. It is also designed for accessible deployment in-vehicle environments. It is ideal for hardware-in-the-loop (HIL) simulations and on-road testing of ADAS functionalities like adaptive cruise control, lane-keeping assistance, and automated emergency braking. Its compact form factor and high reliability make it an

essential tool for accelerating the development and validation of cutting-edge automotive technologies.

2.2 Software

The software components of the Radar-Camera Sensor Fusion system are vital for data acquisition, processing, and real-time analysis. This section details the software packages and tools required to integrate radar and camera data seamlessly. These elements include the radar software package for data handling, the fusion algorithms for intelligent decision-making, and the user interface for system monitoring and configuration. Ensuring that all software components are correctly installed and configured is essential for achieving the system's intended performance and functionality.

2.2.1 Virtual Environment

A virtual environment is a tool that helps keep dependencies required by different projects separate by creating isolated Python virtual environments. This is one of the most essential tools that most Python developers use.

A Python Virtual Environment is an isolated space where you can work on your Python projects separately from your system-installed Python.
You can set up your libraries and dependencies without affecting the Python system.
We will use 'virtualenv' to create a virtual environment in Python.

2.2.2 YOLO

YOLO (You Only Look Once) is a high-speed multi-object detection algorithm that uses a convolutional neural network (CNN) to detect and identify objects. The two Yolo versions that are used are represented below.

2.2.3 ROS

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

ROS Noetic Ninjemys is primarily targeted at the Ubuntu 20.04 (Focal) release, though other systems are supported to varying degrees.

2.2.4 Torch and TorchVision

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

To extend PyTorch, you can reuse your favorite Python packages, such as NumPy, SciPy, and Cython.

pip install torch

The torchvision package comprises popular datasets, model architectures, and standard image transformations for computer vision.

pip install torchvision

3. Setup Hardware and Software

Setting up the Radar-Camera Sensor Fusion system involves a series of steps to ensure that hardware and software components are correctly installed and configured. This section provides detailed instructions for installing the radar software package and setting up the environment on the NVIDIA Orin platform. Proper setup is crucial for achieving optimal functionality, allowing for effective data fusion and deploying advanced collision warning systems. Follow the guidelines carefully to establish a robust operational environment for your sensor fusion applications.

3.1 Hardware Setup

Effective sensor mounting is essential for the optimal performance of the Radar-Camera Sensor Fusion system. Proper placement and alignment of radar and camera sensors ensure accurate data collection, which is critical for timely collision detection and warning. This section outlines best practices for mounting sensors, including location, height, angle, and secure attachment considerations. By following these guidelines, you can maximize the effectiveness of your sensor fusion system and enhance overall vehicle safety.

CONFIGURATION

- ➔ Connect a display port cable to the Orin and to the PC monitor. You can use a DisplayPort to HDMI adapter if the monitor does not support the DisplayPort connection.
- ➔ Connect a mouse and a keyboard to the USB ports.
- ➔ Optionally connect an ethernet cable if a WiFi network is not available.

➔ Plug the USB Type-C™ power adapter to the Type-C™ port above the round power jack.

The sensor setup diagram illustrates a comprehensive system for integrating and managing various sensors and devices, ensuring seamless data collection and processing. At the core of the setup is the Extension Box, which serves as a central hub for connecting all components. The system is powered by a battery UPS, which ensures an uninterrupted power supply and includes a GNSS battery to provide backup power specifically for the GNSS receiver. The GNSS Receiver processes GNSS signals, which the RTK module enhances for high-precision GPS data.

The system includes a Basler camera for visual data capture, which connects to the computing platform via USB cables. The computing platform, often an Orin, runs the necessary computations and integrates data from all connected sensors. A Micro-Auto Box is also a robust and compact computing unit designed specifically for automotive applications.

The setup supports wireless communication through a Bluetooth dongle, facilitating remote control and data transfer. A monitor is connected to the computing platform via a DP (DisplayPort) cable to display real-time data and system status. A dedicated monitor power cable powers the monitor.

The entire system is supported by a power supply, which distributes power to each component through various cables. Ethernet cables provide network connectivity between devices, ensuring efficient data transfer. The GNSS Cable connects the GNSS receiver to both the battery and the computing platform, while the Orin Power Cable ensures the Orin platform is adequately powered. Finally, a DB-9 cable facilitates serial communication between different system components.

Overall, this setup ensures that data from multiple sensors is collected, processed, and displayed efficiently, with robust power management and connectivity solutions supporting the entire process.

3.1.1 Radar Sensor

3.1.2 Basler camera

Complementing the radar systems, we incorporate a Basler camera to capture real-time visual data. Operating at a frame rate of 20 frames per second (fps), the Basler camera is critical in enhancing object recognition tasks. Its high-resolution imagery allows for identifying and classifying objects, such as pedestrians, vehicles, and obstacles, providing vital contextual

information that complements the radar data. This synergy between radar and camera technologies fosters a more robust and reliable sensor fusion system, significantly improving overall safety and responsiveness in various driving conditions.

By leveraging the strengths of both radar and camera systems, this setup enhances detection capabilities and supports advanced driver-assistance systems (ADAS), contributing to safer driving experiences.

3.2 GNSS Receiver Mounting

Install required ros software on host pc

- Here i'm used ros noetic for ubuntu 20.04
- https://www.stereolabs.com/blog/ros-and-nvidia-jetson-xavier-nx

ROS Installation

Open a new terminal by pressing Ctrl + Alt + T or execute the "Terminal" application using the Ubuntu 18 launch system.

Set the Jetson Xavier NX to accept software from *packages.ros.org*:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Add a new apt key:

```
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Update the Debian packages index:

```
$ sudo apt update
```

Install the ROS Desktop package, including support for `rqt`, `rviz`and other useful robotics packages:

```
$ sudo apt install ros-melodic-desktop
```

Note: “ROS Desktop Full” is a more complete package. However, it is not recommended for embedded platforms. 2D/3D simulators will be installed, requiring increased storage space and computing power.

It is recommended that the ROS environment variables be loaded automatically when you execute a new shell session. Update your `.bashrc` script:

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
$ source ~/.bashrc
```

Install and initialize ‘rosdep’. The ‘rosdep’ allows you to easily install system dependencies for the source code you want to compile and is required to run some core components in ROS:

```
$ sudo apt install python-rosdep python-rosinstall python-rosinstall-generator
python-wstool build-essential
$ sudo rosdep init
```

```
$ rosdep update
```

To get the GNSS values, install the following packages
The following are required for installation

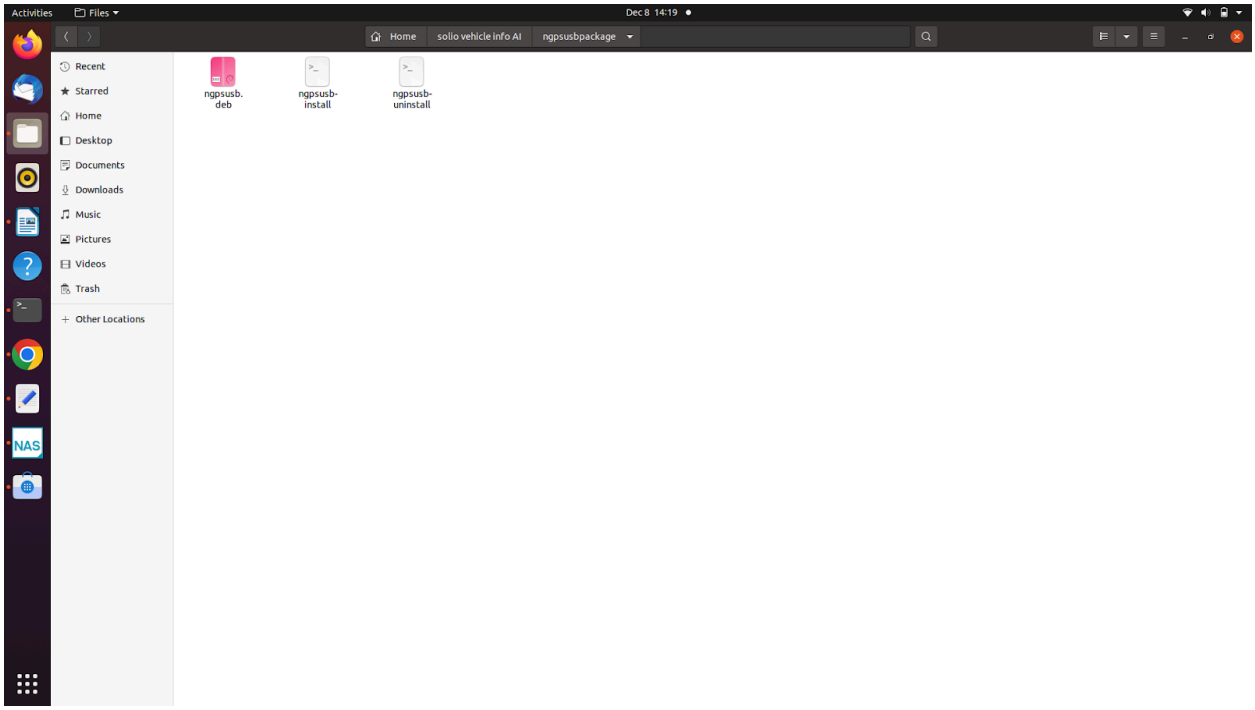
- NovAtel Application Suite:
- NovAtel USB Drivers

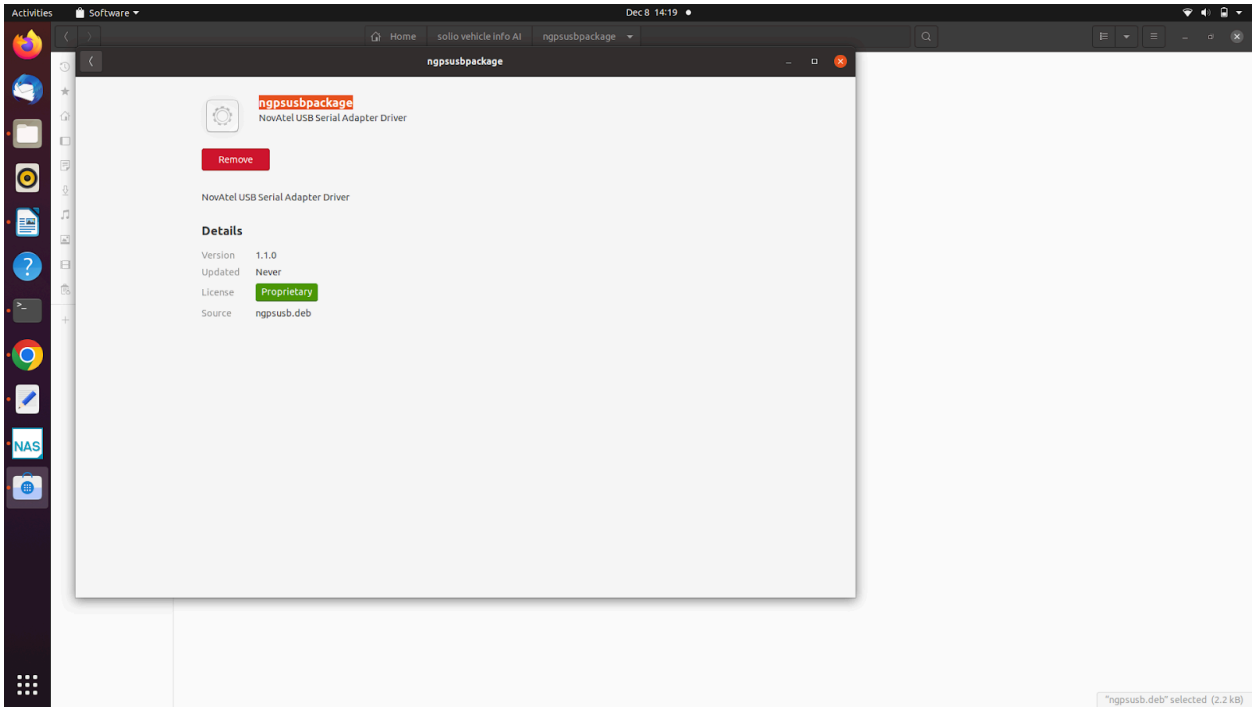
Link to download the NovAtel Application Suite and NovAtel USB Drivers

https://novatel.com/support/support-materials/software-downloads

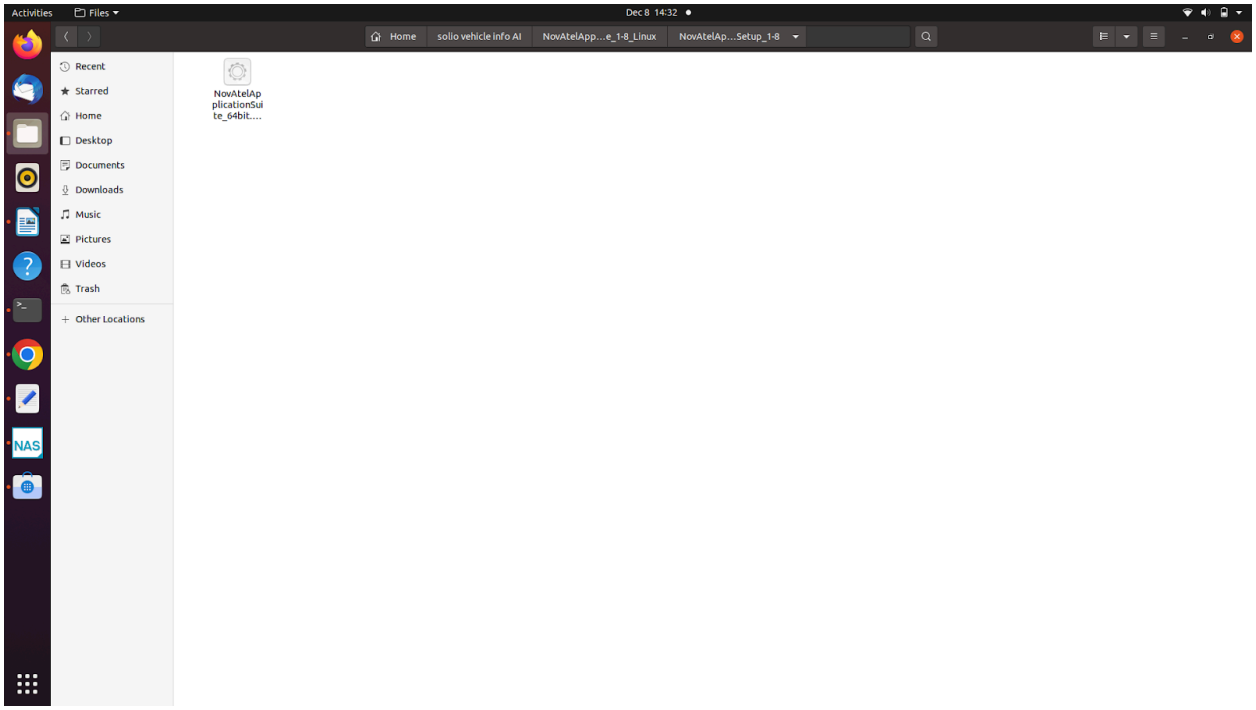
Step1: first, install Novatel USB drivers

First, download and extract the NovAtel USB Drivers file, installing for Ubuntu
Method 1: Double-click on the .deb file, and it will be installed.
Method 2: sudo apt install ./deb_file (deb_file means a path of the deb file)



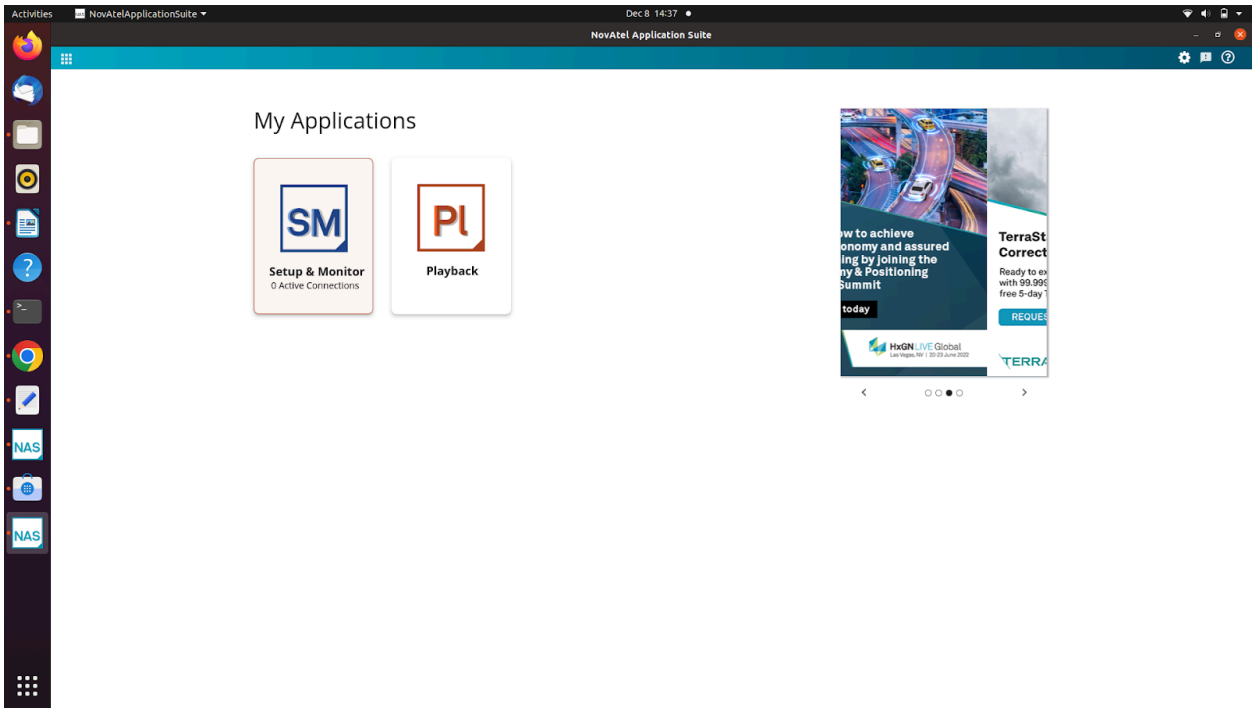


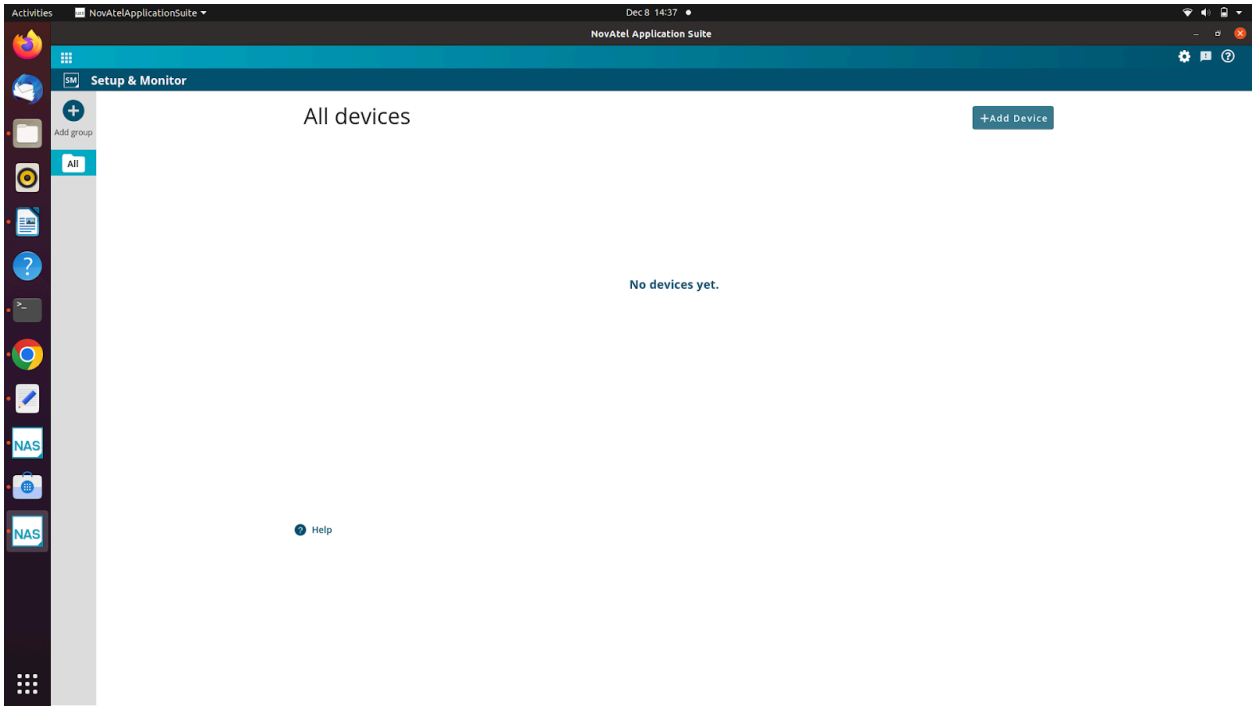
Step2: Download and extract the NovAtel Application Suite

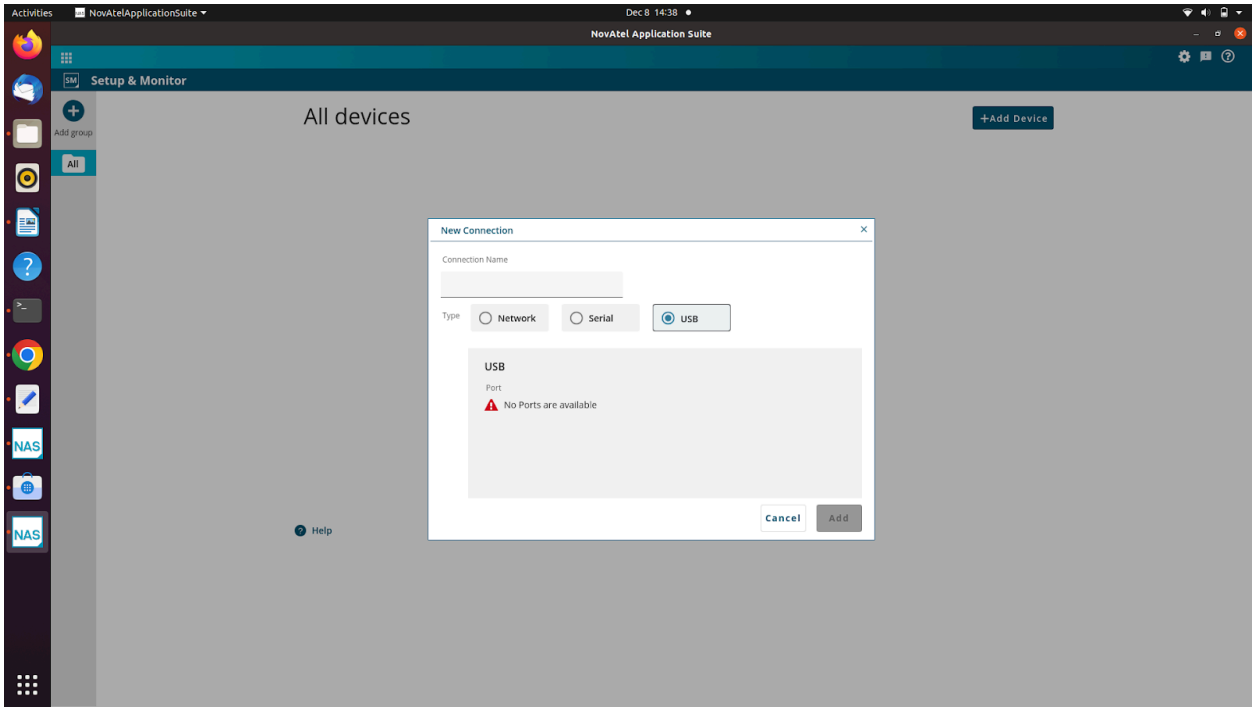


Right-click on the NovAtelApplicationSuite_64bit.AppImage, and you should click on the RUN command.

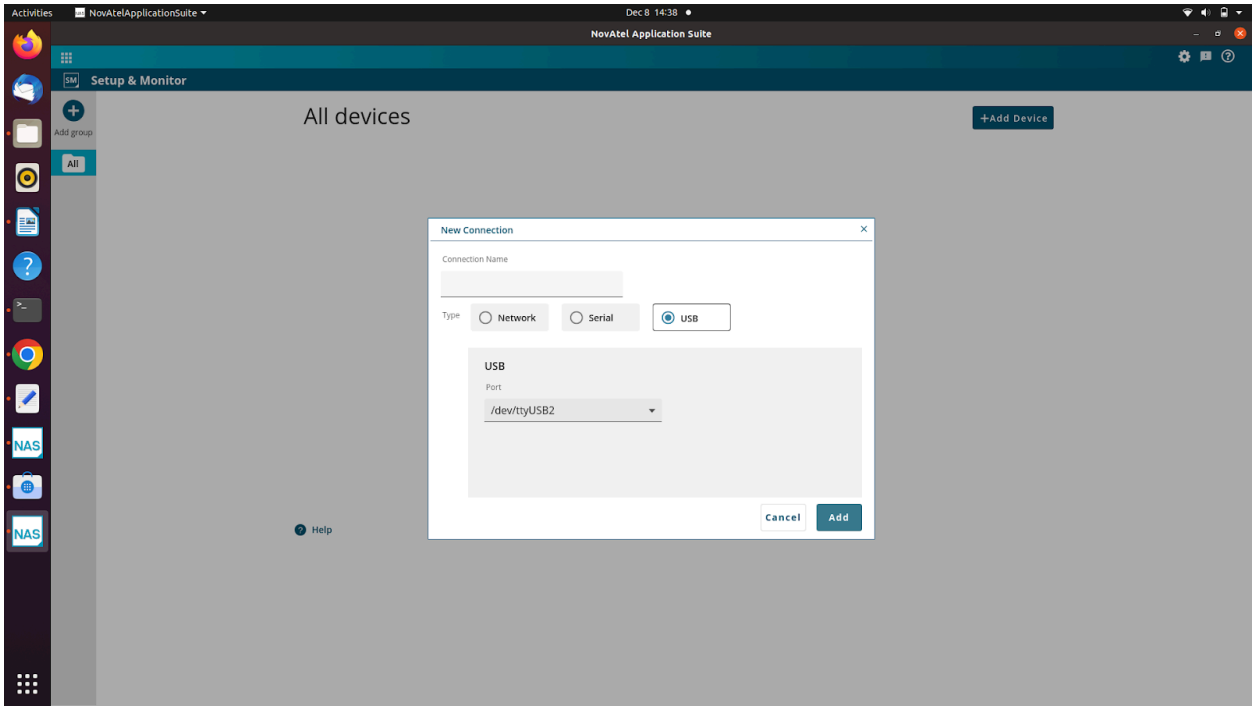
Then, it will open the interface





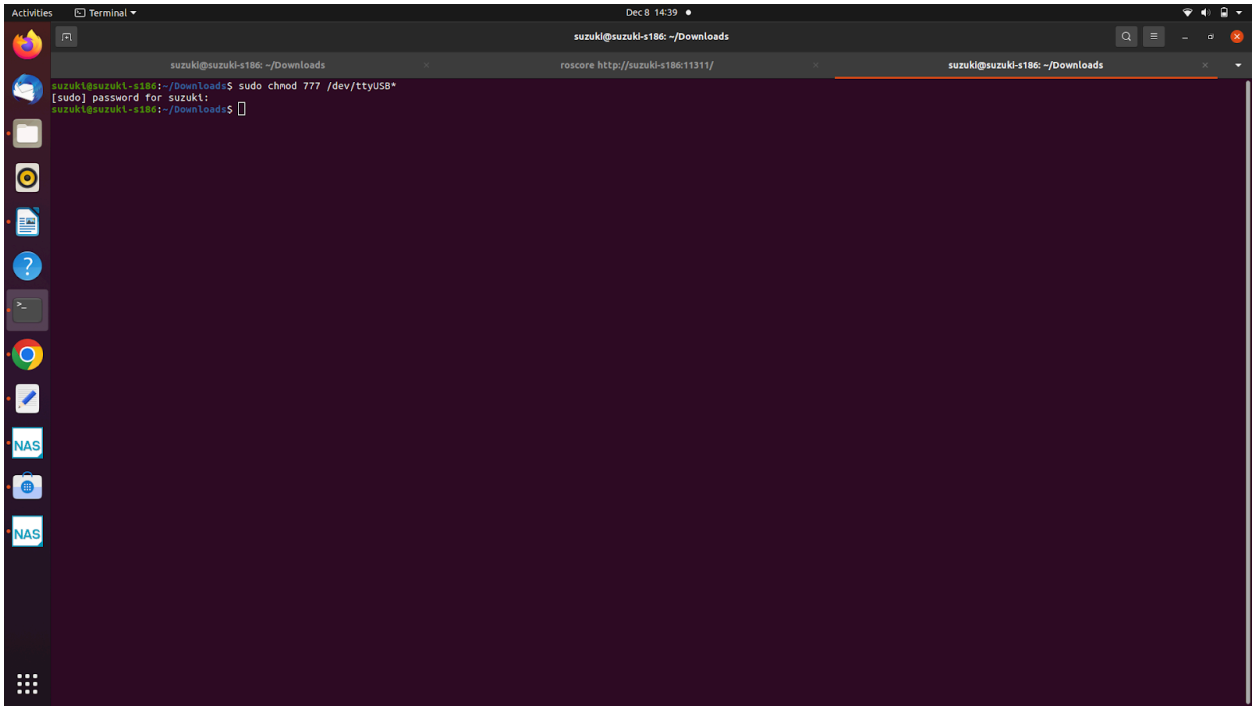


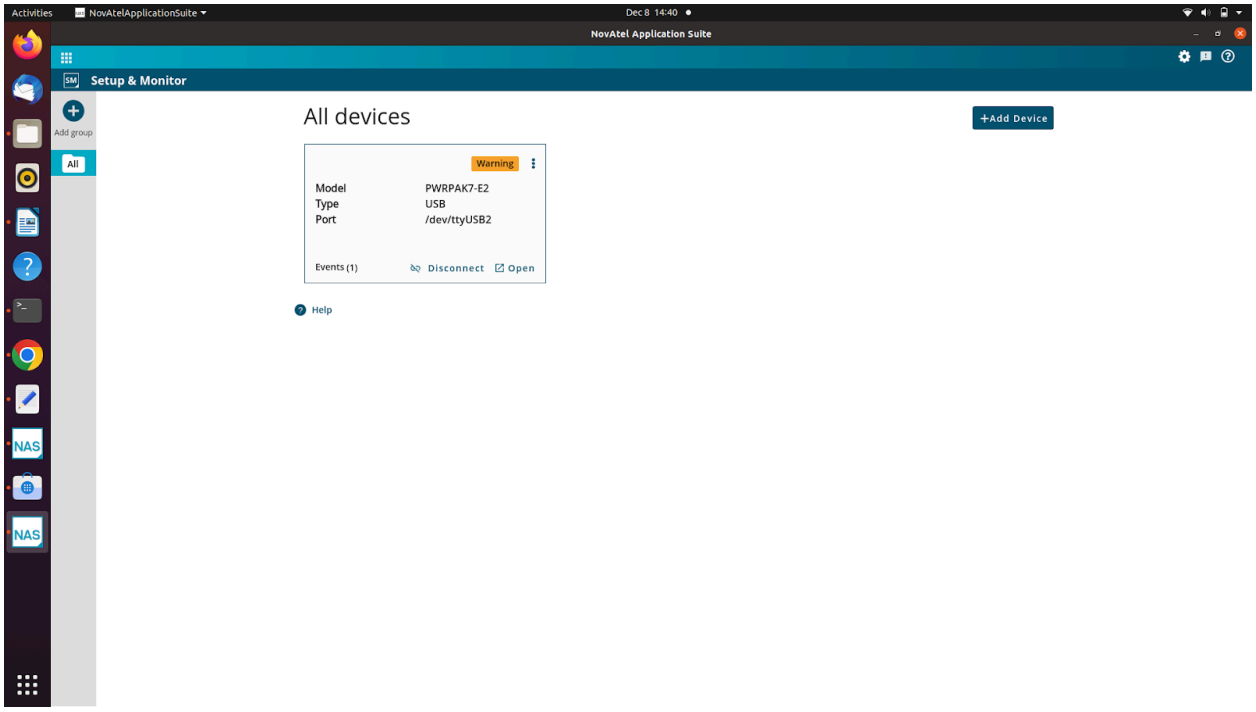
Select any listed USB option

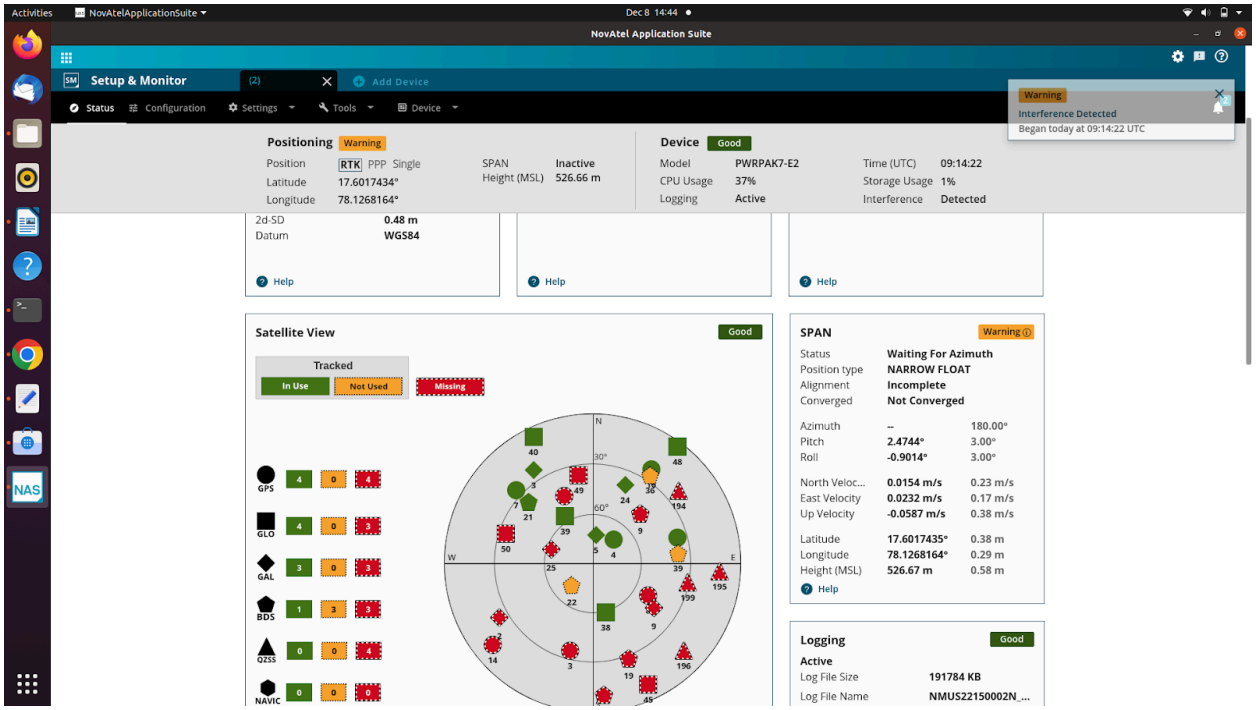


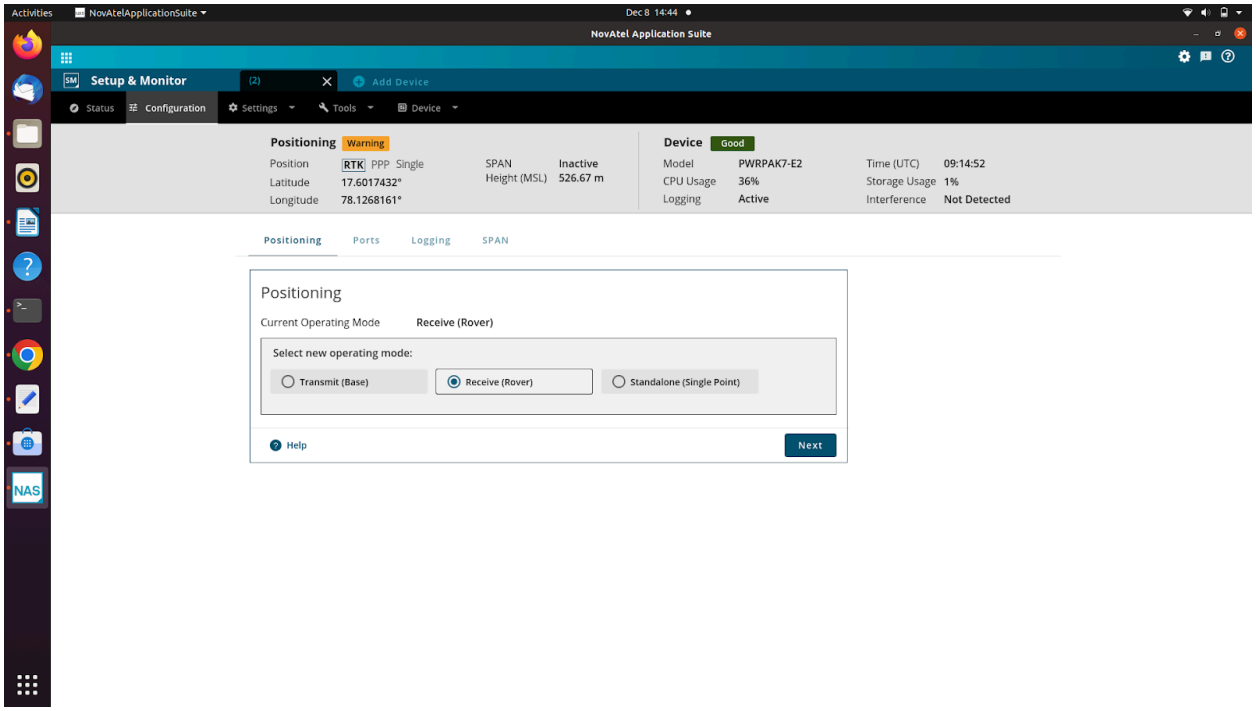
For giving USB permission

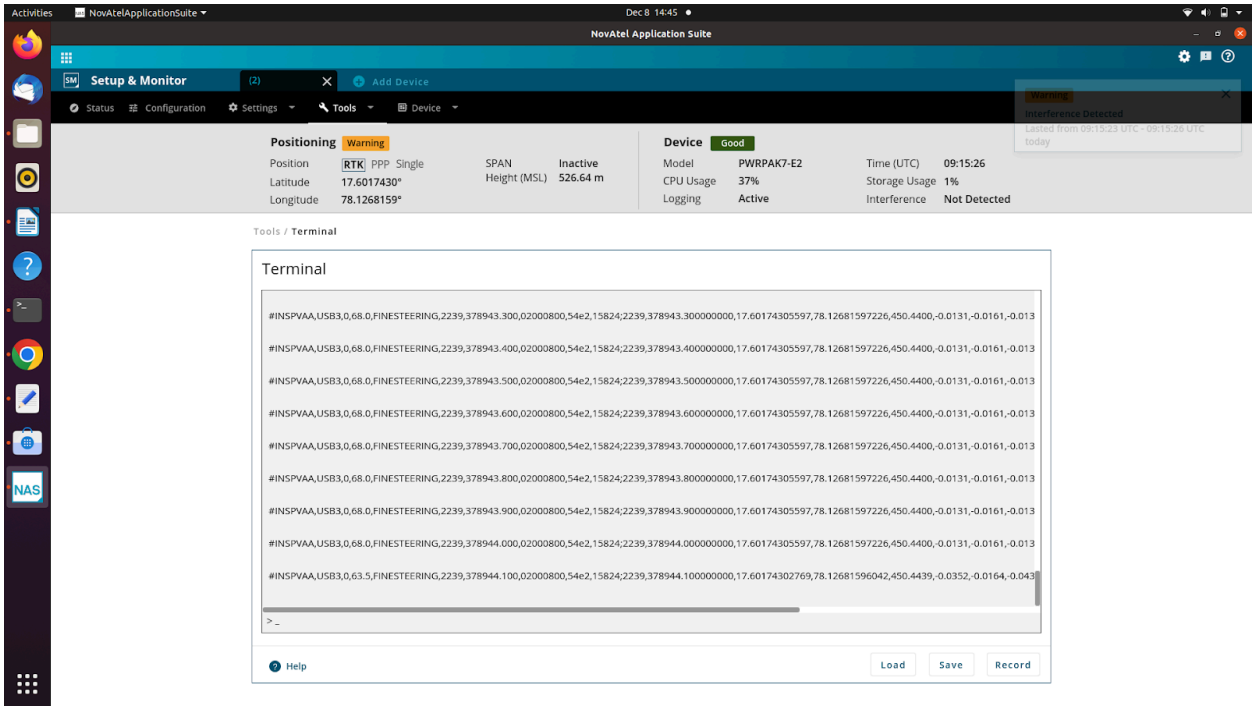
- sudo chmod 777 /dev/ttyUSB*


Activities
Terminal
Dec 8 14:39
suzuki@suzuki-s186: ~/Downloads
suzuki@suzuki-s186: ~/Downloads
roscore http://suzuki-s186:11311/
suzuki@suzuki-s186: ~/Downloads
suzuki@suzuki-s186:~/Downloads$ sudo chmod 777 /dev/ttyUSB*
[sudo] password for suzuki:
suzuki@suzuki-s186:~/Downloads$

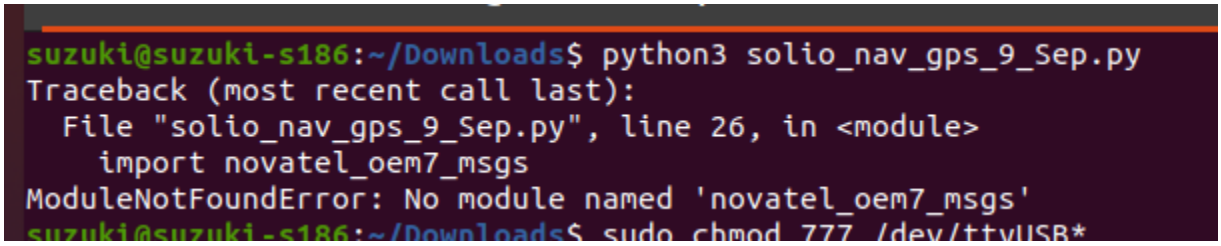

Activities
NovAtelApplicationSuite
Dec 8 14:40
NovAtel Application Suite
Setup & Monitor
Add group
All
All devices
+Add Device
Warning
Model PWRPAK7-E2
Type USB
Port /dev/ttyUSB2
Events (1)
Disconnect
Open
Help


Activities
NovAtelApplicationSuite
Dec 8 14:44
NovAtel Application Suite
Setup & Monitor
(2)
Add Device
Status
Configuration
Settings
Tools
Device
Warning
Interference Detected
Began today at 09:14:22 UTC
Positioning
Warning
Position
RTK PPP Single
Latitude
17.6017434°
Longitude
78.1268164°
SPAN
Inactive
Height (MSL)
526.66 m
Device
Good
Model
PWRPAK7-E2
CPU Usage
37%
Logging
Active
Time (UTC)
09:14:22
Storage Usage
1%
Interference
Detected
2d-SD
0.48 m
Datum
WGS84
Help
Satellite View
Good
Tracked
In Use
Not Used
Missing
GPS
GLO
GAL
BDS
QZSS
NAVIC
SPAN
Warning
Status
Waiting For Azimuth
Position type
NARROW FLOAT
Alignment
Incomplete
Converged
Not Converged
Azimuth
--
180.00°
Pitch
2.4744°
3.00°
Roll
-0.9014°
3.00°
North Veloc...
0.0154 m/s
0.23 m/s
East Velocity
0.0232 m/s
0.17 m/s
Up Velocity
-0.0587 m/s
0.38 m/s
Latitude
17.6017435°
0.38 m
Longitude
78.1268164°
0.29 m
Height (MSL)
526.67 m
0.58 m
Logging
Good
Active
Log File Size
191784 KB
Log File Name
NMUS22150002N_...


Activities
NovAtelApplicationSuite
Dec 8 14:44
NovAtel Application Suite
Setup & Monitor
(2)
Add Device
Status
Configuration
Settings
Tools
Device
Positioning
Warning
Position
RTK PPP Single
Latitude
17.6017432°
Longitude
78.1268161°
SPAN
Inactive
Height (MSL)
526.67 m
Device
Good
Model
PWRPAK7-E2
CPU Usage
36%
Logging
Active
Time (UTC)
09:14:52
Storage Usage
1%
Interference
Not Detected
Positioning
Ports
Logging
SPAN
Positioning
Current Operating Mode
Receive (Rover)
Select new operating mode:
Transmit (Base)
Receive (Rover)
Standalone (Single Point)
Help
Next

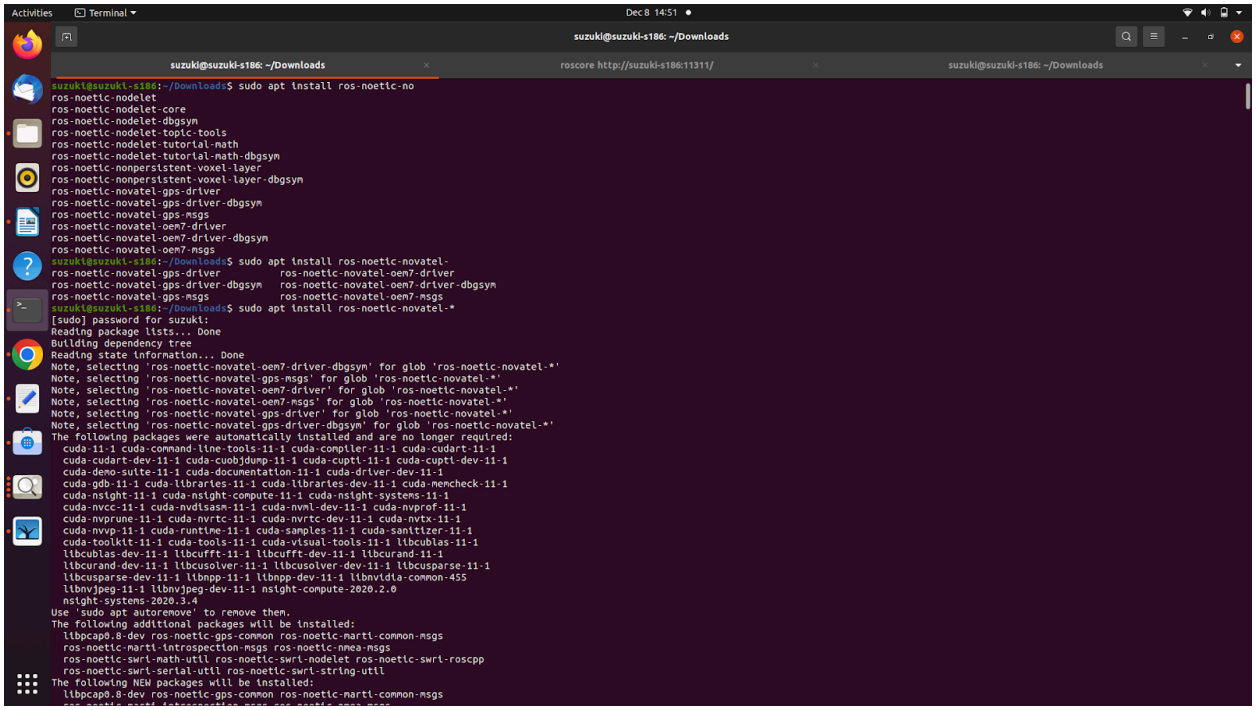


Close everything and install ‘ros novAtel’ drivers otherwise, you will get this error

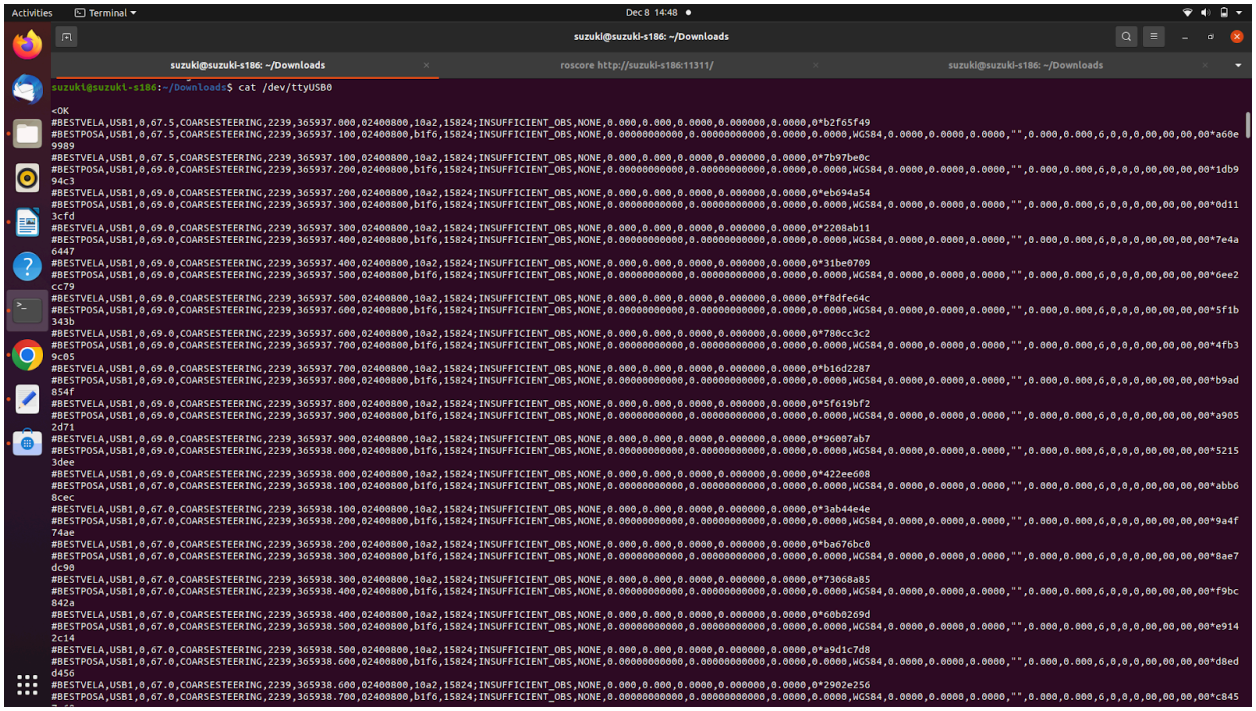


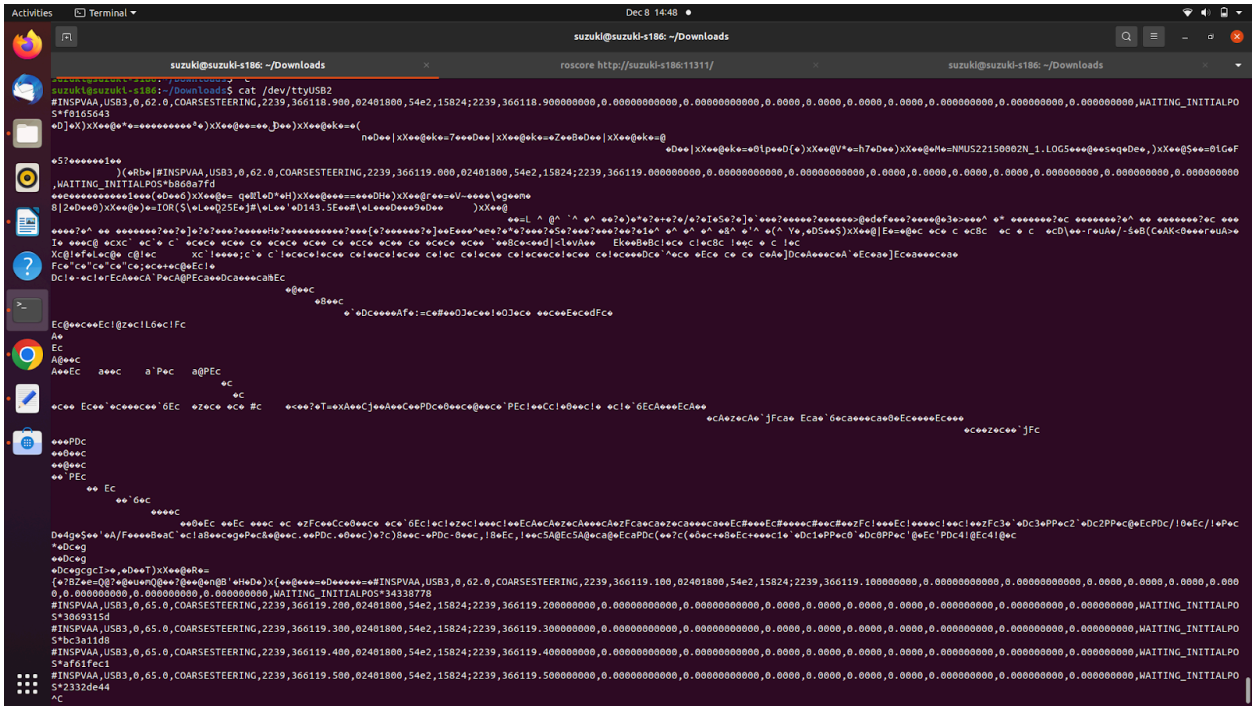
Solving this error: Run

- sudo apt install ros-noetic-no
- sudo apt install ros-noetic-novatel-
- sudo apt install ros-noetic-novatel-*



Check the GNSS receiver messages cat /dev/ttyUSB0 or cat /dev/ttyUSB2





Use this command to receive messages from GNSS

```
roslaunch  novatel_oem7_driver oem7_tty.launch oem7_tty_name:=/dev/ttyUSB0
```

3.3 Installation of radar package

3.4 Environment setup in Orin

- SETUP THE VIRTUAL ENVIRONMENT
  - First, install the virtualenv package and create a new Python 3 virtual environment:
    - sudo apt-get install virtualenv
    - python3 -m virtualenv -p python3 <chosen_venv_name>
- ACTIVATE THE VIRTUAL ENVIRONMENT
  - Next, activate the virtual environment:
    - source <chosen_venv_name>/bin/activate
    - Deactivate (for deactivating virtual environment)
- INSTALL DESIRED VERSION OF **Pytorch** AND **Torchvision**

  PyTorch v1.12.0

  - JetPack 5.0 (L4T R34.1.0) / JetPack 5.0.1 (L4T R34.1.1) / JetPack 5.0.2 (L4T R35.1.0)
  - Python 3.8 - torch-1.12.0a0+2c916ef.nv22.3-cp38-cp38-linux_aarch64.whl

- ➔ wget https://nvidia.box.com/shared/static/p57jwntv436lfrd78inwl7iml6p13fzh.whl -O torch-1.12.0a0+2c916ef.nv22.3-cp38-cp38-linux_aarch64.whl
- ➔ sudo apt-**get** install python3-pip libopenblas-**base** libopenmpi-dev libomp-dev
- ➔ pip3 install Cython
- ➔ pip3 install numpy torch-1.12.0a0+2c916ef.nv22.3-cp38-cp38-linux_aarch64.whl

Torchvision v0.16.1

- ➔ git clone --branch v0.16.1 https://github.com/pytorch/vision torchvision
- ➔ cd torchvision/
- ➔ export BUILD_VERSION=0.16.1
- ➔ export CUDA_HOME=/usr/local/cuda-11.4
- ➔ python3 setup.py install

- ★ Reference https://forums.developer.nvidia.com/t/pytorch-for-jetson/72048

❖ CONFIGURATION OF Basler WITH YOLO v5

4. Spatial and Temporal Synchronization

4.1 Data Representation and Sensor Integration

The radar and camera systems in the Radar-Camera Sensor Fusion framework work together to understand the vehicle's surroundings comprehensively. The radar data consists of a series of points in a 2D space, each representing detected objects and their range and azimuth coordinates. This format allows for effective spatial representation of nearby objects, which is essential for collision detection.

In parallel, the camera system captures sequential frames that visually represent the environment from the vehicle’s perspective. These frames contain detailed imagery that aids in identifying and classifying objects, enhancing situational awareness.

Both the radar and camera are securely mounted on the vehicle, as illustrated in **Fig. 3.** This strategic positioning optimizes data acquisition, minimizes interference, and ensures stability during vehicle operation. The complete field of view provided by the radar sensors is depicted in **Fig. 2**, illustrating their extensive coverage.

The front-mounted radar and camera systems are pivotal in delivering Forward Collision Warning (FCW) alerts, while the rear-mounted counterparts focus on providing Rear Collision Warning (RCW) notifications. This dual approach enhances the vehicle's safety by addressing potential hazards from both directions.

A critical aspect of effective sensor fusion is achieving time and spatial synchronization. This involves addressing discrepancies in data acquisition rates and differing coordinate systems between the radar and camera systems. The framework integrates information from multiple sensors by synchronizing the data streams, producing a coherent representation of the environment. This integrated data allows for advanced processing and decision-making capabilities, significantly improving the vehicle's response to potential collision scenarios.

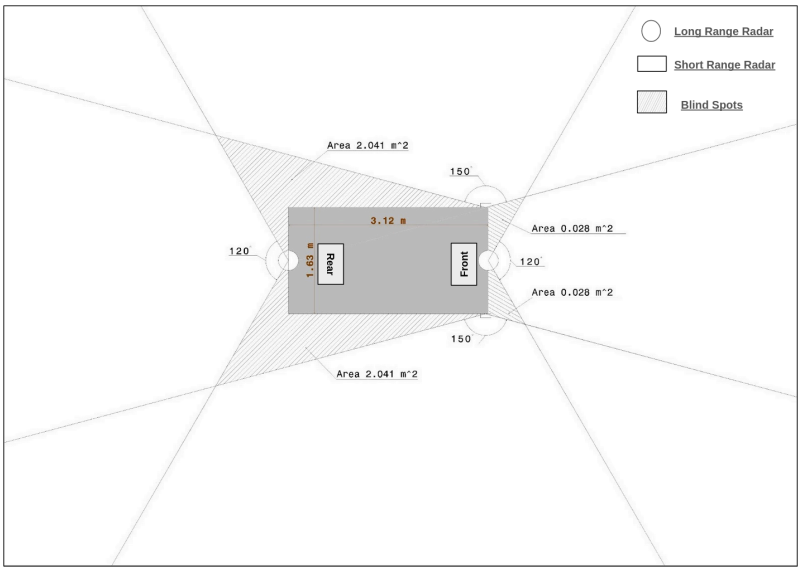


Fig. 2: FoV of Long Range Radar and Short Range Radar mounted on the vehicle with total Blind Spot area calculated around the vehicle.

Fig. 3: Placement of Radars and Cameras on the Vehicle for Enhanced Collision Detection. The figure on the left is the front view of the car, while on the left is the rear view

4.2 Coordinated Data Synchronization

In our Radar-Camera Sensor Fusion system, radar data is acquired in the Bird's Eye View (BEV) frame, which presents a different spatial representation compared to the camera's frame, as illustrated in Fig. 4. To address this disparity, we implement a Front View (FV) projection technique that effectively unifies these two datasets into a common spatial context. This process involves projecting radar-detected objects onto the camera frame, enabling accurate integration and interpretation of the data.

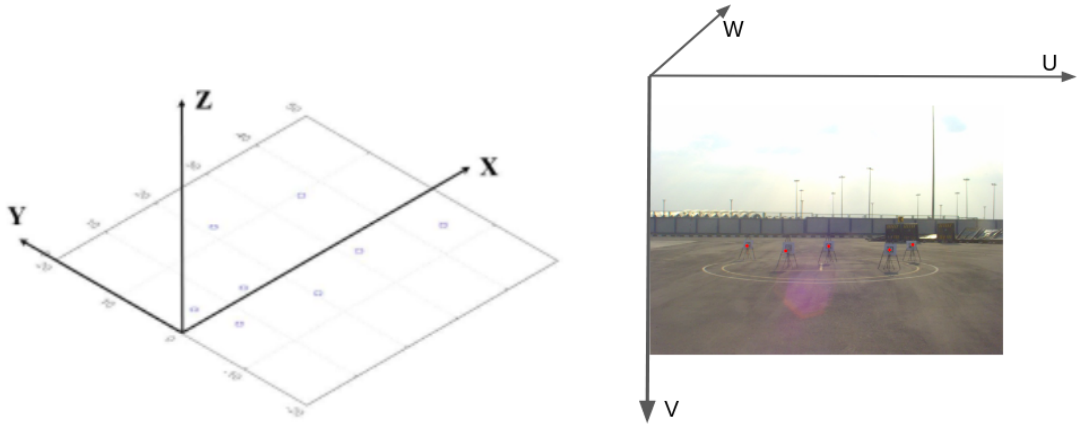
For spatial synchronization, we employ a geometric calibration method, as detailed in previous studies. This method utilizes Normalized Direct Linear Transformation (NDLT) to align the radar and camera frames, significantly reducing error in correspondence. As demonstrated in [16], normalizing NDLT followed by optimization helps achieve precise spatial alignment. The transformation matrix used in the NDLT method is represented in Equation (1):

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} u_{iw} \\ v_{iw} \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

In this equation, $[X_i, Y_i, Z_i]^T$ represents the radar coordinates, while $[u_i, v_i]^T$ denotes the corresponding camera coordinates. The variable ω acts as a scaling factor, ensuring proper dimensional consistency.

Given the inherent differences in data acquisition rates between our radar and camera systems—where the radar operates at a higher frequency than the camera's frame rate—temporal synchronization becomes crucial. To accurately associate corresponding data points, we align the timestamps from both sensors. The synchronization algorithm discussed in [16] facilitates this temporal alignment, ensuring that data from the radar and camera frames can be integrated with minimal latency and error. The ROS-based approach highlighted in [16] optimizes this process, enhancing the reliability of our experimental results.

By achieving both spatial and temporal synchronization, the Radar-Camera Sensor Fusion system effectively combines radar and camera data, enabling enhanced object detection and collision avoidance capabilities.



(a) Radar Coordinates (b) Camera Coordinates

Fig. 4: Distinct coordinate frames of sensors

4.3 Sensor Fusion Framework

The sensor fusion process begins with acquiring raw radar data using the Perception Development Kit (PDK), a specialized software module tailored for real-time data collection from the ARS430DI and SRR520DI sensors. The PDK is the backbone of radar data processing, enabling essential functionalities for a Sensor Fusion-Based Collision Warning System, including data extraction and analysis.

Concurrently, radar and camera data streams are obtained with synchronized timestamps through the Robot Operating System (ROS). This framework ensures smooth communication between the various components of the sensor fusion system, facilitating efficient data handling.

The live video feed from the Basler camera is processed using the YOLOv8 (You Only Look Once version 8) algorithm, which excels in real-time object detection. YOLOv8 is recognized for its state-of-the-art deep learning capabilities, allowing for efficient and accurate identification of objects within the camera frames. The algorithm produces bounding boxes around detected objects, offering critical spatial information for subsequent processing.

Following object detection, time synchronization is applied to ensure that the radar and camera data streams are aligned. As previously discussed, spatial synchronization guarantees coherence between the detected objects in the camera images and their corresponding radar features, such as distance and velocity. The fused results, illustrated in Fig. 5, showcase radar-derived distance and velocity data integration with visual object detection.

We achieve a cohesive representation of the environment with synchronized data streams and spatially aligned information. By harnessing the complementary strengths of radar and camera

sensors, the system significantly enhances its ability to detect and mitigate potential collisions in real-time, thus improving overall safety and situational awareness.

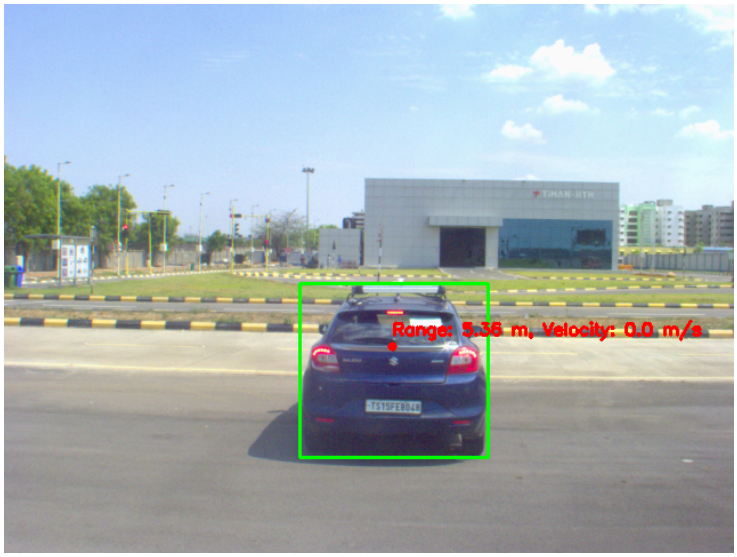
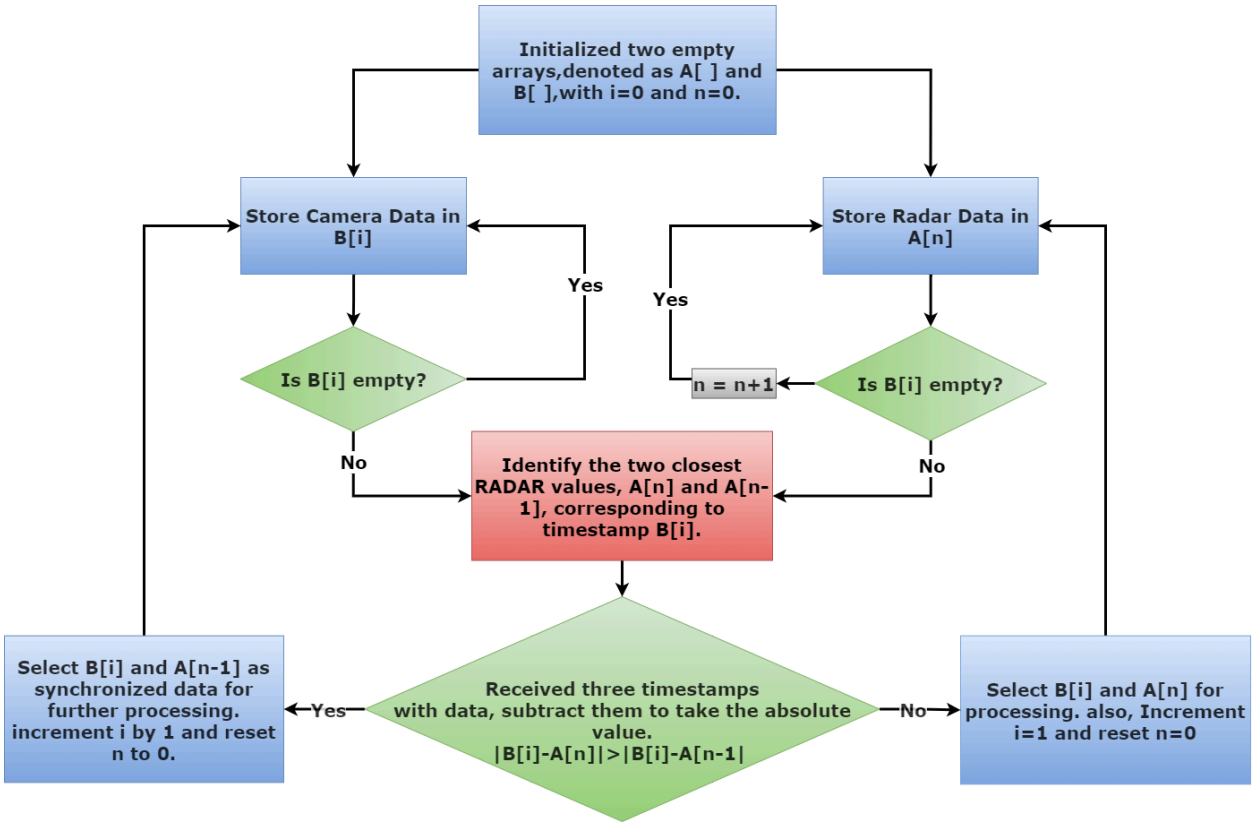


Fig. 5: Projection of radar detection onto the camera frame

5. Methodology



The flowchart outlines a process for synchronizing data from two sources, likely cameras and radar, using two empty arrays, `A[]` for Camera Data and `B[]` for Radar Data. Initially, indices `i` and `n` are set to 0. Data is stored in the respective arrays, and synchronization occurs by checking if `B[i]` is empty. If it contains data, the algorithm identifies the closest timestamps between `A[n]` and `B[i], selecting these synchronized datasets for further processing. After processing, the index `i` is incremented, and `n` may be reset to 0 as needed. The algorithm calculates the absolute difference between the relevant data points to determine which data set to prioritize. This synchronization process is crucial in contexts like autonomous vehicles or surveillance systems, where precise timing of data is essential for accurate analysis.

The flowchart outlines a safety mechanism or alert system likely designed for vehicles. It begins with sensors detecting the motion of the subject vehicle, monitoring aspects like acceleration and sudden changes in movement. Another set of sensors measures the distance to forward objects using radar or lidar, which helps assess potential collision risks. The system then processes data from both motion and distance sensors to determine if a warning is necessary. Suppose a critical situation is detected, such as rapid deceleration or proximity to an obstacle. In that case, it triggers a visual warning, which could manifest as a dashboard alert or heads-up display for the driver. This safety system combines motion and distance measurements to provide timely warnings, contributing to safer driving and accident prevention.

6. Results and discussions