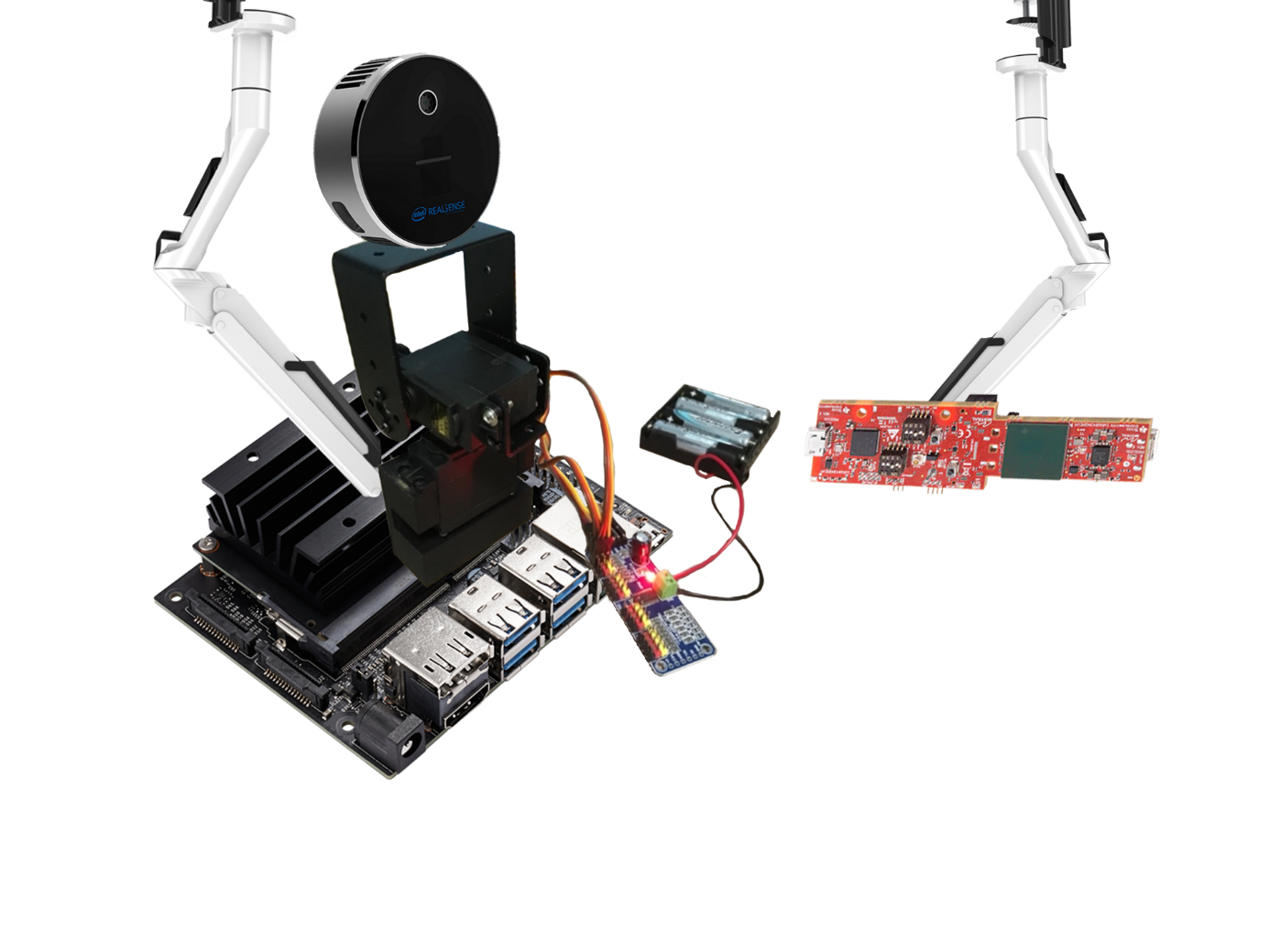
Logo, company name

Description automatically generated

­­­­­­­­­­­­­­­­

Group 4

Goh Kheng Xi, Jevan (A0199806L)

Ryan Tang Li Yen (A0200900A)

Lock Mei Lin (A0204751M)

Final Report

**C**LUSTER TRACKER

­­

Contents

[Acknowledgements 2](#_Toc87908554)

[1. Abstract 2](#_Toc87908555)

[1.1 Key Components and Set-up 2](#_Toc87908556)

[1.2 General Specifications 3](#_Toc87908557)

[1.3 System Overview 3](#_Toc87908558)

[1.4 Bill of Materials 4](#_Toc87908559)

[1.5 Project Timeline 5](#_Toc87908560)

[1.6 Contributions 5](#_Toc87908561)

[2. Devices and Sensors 5](#_Toc87908562)

[2.1 Jetson Nano, MG995 Servo Motors, and PCA9685 Servo Driver 6](#_Toc87908563)

[2.2 IWR6843 AOP EVM (mmWave RADAR) 6](#_Toc87908564)

[2.2.1 Specifications 7](#_Toc87908565)

[2.2.2 Signal specifications 7](#_Toc87908566)

[2.3 L515 8](#_Toc87908567)

[2.3.1 Specifications 8](#_Toc87908568)

[2.3.2 Depth data 8](#_Toc87908569)

[2.3.3 Distance between targets 9](#_Toc87908570)

[3. Algorithm 9](#_Toc87908571)

[3.1 IWR6843 on-chip algorithm 9](#_Toc87908572)

[3.2 Server Script: Serial output from IWR6843 10](#_Toc87908573)

[3.3 Server script 10](#_Toc87908574)

[3.4 Jetson Nano 11](#_Toc87908575)

[3.5 L515 11](#_Toc87908576)

[4. Configuration and Tunings 12](#_Toc87908577)

[4.1 Setup 12](#_Toc87908578)

[4.2 IWR6843 on-chip algorithm parameter tunings 13](#_Toc87908579)

[4.3 L515 Artificial Intelligence 14](#_Toc87908580)

[5. Evaluations and Key Considerations 14](#_Toc87908581)

[6. References 16](#_Toc87908582)

[ANNEX A 17](#_Toc87908583)

# Acknowledgements

We would like to extend our thanks to Professor Jeroen and Andrew for their continuous support and feedback throughout the ESP3902 project. Their constructive critics have given us serious considerations that steered us in the right direction especially during the initial phase of the project; and have given us great insights about the technology implemented in this project.

Special thanks to Huei Ming for his consistent guidance; our completion of the project could not have been accomplished without the help of Huei Ming. Thanks for constantly answering to our queries and trying to meet our requests in addition to the valuable advice.

Lastly, we would like to offer our sincere gratitude for the time and patience that the professors and Huei Ming have shown.

# 1. Abstract

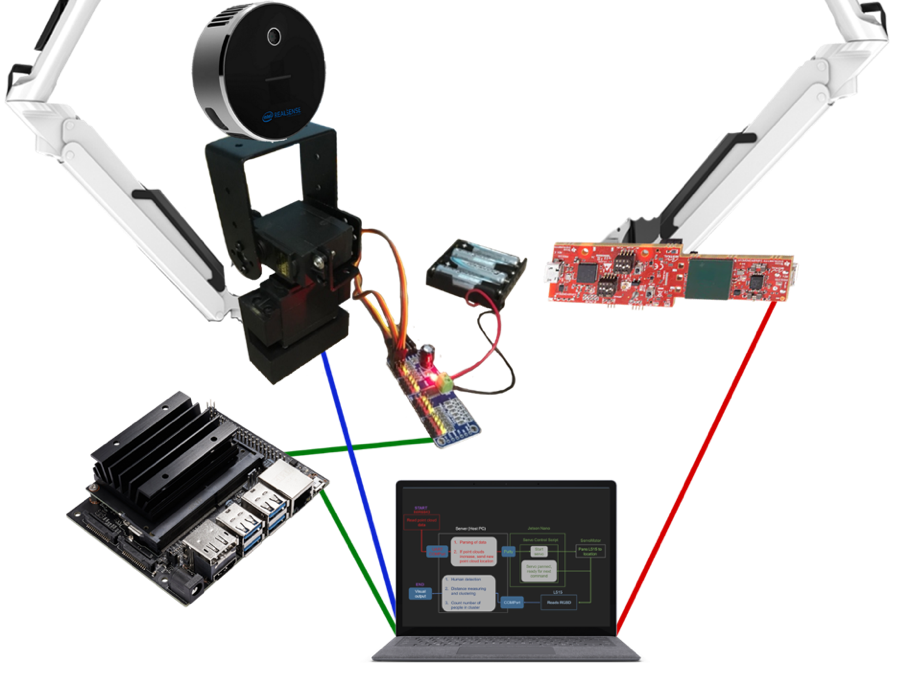
­­­­­­­The *Cluster Tracker* is a sensor system designed to track clusters of people through sensor fusion and artificial intelligence. It aims to alleviate the manpower required to enforce social-distancing rules; however, minor adjustments can be made for the system to perform other tasks. The following are the targets’ metadata obtainable by the sensors onboard the system.

1. Velocity of targets relative to the system
2. Acceleration of targets relative to the system
3. Range of targets relative to the system
4. Azimuth and elevation of targets relative to the system
5. Number of targets present in the Field-Of-View (FOV) of the system

## 1.1 Key Components and Set-up

There are two sensors in the system, the first is a millimetre wave (mmWave) RADAR and the second is a 3D LiDAR Camera. The former has a better tracking range whereas the latter is responsible for people detection and accurate distance tracking. Data processing is done on the Host PC.

Figure 1. Cluster Tracker Setup



1. Texas Instruments’ IWR6843 AOP EVM mmWave Sensor

2. Intel RealSense L515 LiDAR Camera

3. Host PC

4. NVIDIA Jetson Nano

6. PCA9685 PWM/Servo Driver

Ceiling Mounts

Batteries

133 mm

61 mm

60 mm

91 mm

23 mm

## 1.2 General Specifications

|  |  |
| --- | --- |
| FOV (horizontal) | 120°C |
| Range | 12 m |
| Maximum velocity of target detectable | 6 m/s |
| Maximum number of targets in a frame | 13 |
| Accuracy[[1]](#footnote-2) | 82 % |

Tests were conducted indoor (12 m x 8 m lab), under normal room lightings, and at about 26°C. The sensors were mounted on tripods at a height of 130 cm above the ground.

Table . System specifications

|  |  |  |
| --- | --- | --- |
| Device | Nominal Power | Source |
| PCA9685 Servo Driver | 5 V 2 A | 4 x AA alkaline cells |
| L515 and IWR6843 | 5 V 0.5 A | Host PC USB port |
| Jetson Nano | 5 V 2 A | AC power adaptor |

Table 2. Power Consumption

## 1.3 System Overview

A picture containing graphical user interface

Description automatically generatedWith its wide field of view, the IWR6843 monitors people in the room in the form of point clouds. When a new cluster of points (target) is detected, its location is sent to the Jetson Nano, which triggers the panning of the L515 toward the direction of the target. The L515 then checks for the presence and number of people in the cluster, whereupon the distance between each person and between the new and neighbouring clusters are measured. If any distance falls below the pre-set minimum, the L515 outputs it visually while a violation signal that was sent to the server script outputs its location.

Figure 2. System architecture and operational flow

## 1.4 Bill of Materials

Table 3. Bill of System Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source** | **Item** | **Description** | **Qty** | **SGD Cost (incl. GST)** | **Link** |
| Digi-key | Intel RealSense LiDAR Camera L515 | Resolution: 1920 x 1080  Frame rate: 60 fps  Shutter: Rolling Depth Accuracy: ~5 mm to ~14 mm thru 9 m2 Depth Field of View (FOV): 70° × 55° (±3°) Depth output resolution: Up to 1024 × 768 Depth frame rate: 30 fps | 1 | $532.89 | [L515](https://www.digikey.sg/products/en/sensors-transducers/camera-modules/1003?k=intel%20l515) |
| Texas Instruments | IWR6843 AOPEVM  mmWave sensor | Evaluation module (EVM) for integrated antenna-on-package (AoP)  Operating frequency: 60-GHz to 64-GHz Antennas and range: 4 receive 3 transmit antenna with 120° azimuth FOV and 120° elevation FOV | 1 | $191.15 | [IWR6843AOPEVM](https://www.ti.com/tool/IWR6843AOPEVM#order-start-development) |
| Huei Ming | NVIDIA Jetson Nano Developer Kit | Memory: 4GB Power supply: 5W Model: B01 | 1 | $169.00 | [Jetson Nano 4GB](https://www.lazada.sg/products/nvidia-jetson-nano-4gb-developer-kit-singapore-distributor-stocks-i353214996-s823532733.html?) |
| MicroSD Card | Storage: 64GB | 1 | $3.99 | [64GB microSDXC](https://www.amazon.sg/gp/product/B092PQ4MM7/ref=ox_sc_act_title_2?smid=A1TJHJ1Q27M030&psc=1) |
| (Note: Items above are not deducted from SGD500 budget) **SUBTOTAL** | | | | **$893.04** |  |
| Unicell International P/L | Power Adapter for Jetson Nano | Output voltage: 5V  Output current: 5A Output power: 25W Output terminal type: PC04 DC plug (5.5 x 2.5 mm) | 1 | $36.84 | NA |
| Power Cable | Length: 1.5 m | 1 | $3.30 | NA |
| blueidea (Lazada) | Servo Mount Bracket | Number of brackets: 2 Number of sets: 2 DOF: 2 (pan and tilt) | 1 | $6.80 | [Servo Mount Brackets](https://www.lazada.sg/products/3-sets-servo-mount-bracket-2-dof-for-mg995-mg996r-s3003-steering-gear-pan-and-tilt-mount-robot-car-boat-i1951759806-s10468234540.html?) |
| Shipping | From China |  | $1.49 |
| Open-source (Lazada) | PWM/Servo Driver | Frequency: 40-1000Hz Number of channels: 16 Resolution: 12 bit Voltage: DC 5 -10V Size: 60 x 25 mm2 | 1 | $5.54 | [PCA9685](https://www.lazada.sg/products/pca9685-16-channel-12bit-pwm-servo-motor-driver-board-controller-iic-interface-for-arduino-raspberry-pi-zerozero-wzero-robot-i1878289564-s10015048565.html?spm=a2o42.searchlist.list.1.438013166kb0mk&search=1) |
| Shipping | From China |  | $1.49 |
| luckcute.sg (Shopee) | MG995 Servo Motor (\*) | Dimension: 40mm x 19mm x 43mm Net weight: 69g Operating Voltage: 4.8 - 7.2 V Operating Speed: 0.17sec / 60 degrees (4.8 V no load) or 0.13sec / 60 degrees (6.0 V no load) | 2 | $3.90 | [MG995 Servo](https://shopee.sg/13KG-15KG-Servos-Digital-MG995-MG996-Servo-Metal-Gear-for-Futaba-JR-Car-RC-Model-Helicopter-Boat-For-Arduino-UNO-diy-i.161750523.7859759120) |
| Shipping | From China |  | $1.00 |
| Sun Light Eectronics  Pte Ltd | Jumper Cables | Configuration: Female-to-female Length: 300 mm Quantity: 30 | 1 | $7.50 | NA |
| MG995 Servo Motor | Same as (\*) | 1 | $15.00 |
| Battery Holder | 4 x AA Batteries | 1 | $1.20 |
| **SUBTOTAL** | | | | **$88.96** |  |
| **GRAND TOTAL** | | | | **$982.00** |
| **BALANCE of SGD 500 Budget** | | | | **$411.04** |

## 1.5 Project Timeline

Table 4. Tasks and Timeline

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Week | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | R | 7 | 8 | 9 | 10 | 11 | 12 |
| Ideation, tutorials, research on different sensors |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research on trained AI models or datasets |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Buy components |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Receive components |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research on L515 and IWR6843 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pan-tilt mechanism for L515 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Servo control with Jetson Nano |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Data output from L515 and IWR6843 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Calibrate L515 and IWR6843 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Tuning and testing with L515 and IWR6843 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Increase IWR6843 detected points and tracks |  |  |  |  |  |  |  |  |  |  |  |  |  |
| People detection |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Number of people |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Port software and code to Jetson Nano |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Interface host PC and Jetson Nano |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Interface sensors and servo motors |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Distance between people |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Full integration |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing of completed system |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 1.6 Contributions

For the project, Ryan was responsible for the Jetson Nano, L515 pan-tilt mount, and the Jetson Nano Python script. Jevan focused on the IWR6843AOPEVM, the server script, and the parametric tunings of the RADAR. Mei Lin worked on the L515 and Artificial Intelligence. Nonetheless, due to the complex and meshed interfacing of all the devices, everyone contributed equally to each component of the project. For the report, each team member contributed to their respective sections while keeping an eye out for each other’s parts.

# 2. Devices and Sensors

The three main devices as seen in Sec. 1.3 are the Jetson Nano, IWR6843, and the L515. This section briefly describes their capabilities and working principles, which provide us the basis for sensor fusion.

A picture containing text, indoor, ceiling

Description automatically generatedA picture containing text

Description automatically generatedThe IWR6843 has a wide field of vision (FOV) and a long range; but because its data is low in resolution, it is difficult to separate the entities in proximity to one another. On the other hand, the L515, whose range is much shorter and view more narrower, is able to complement it with its higher RGBD resolution and its higher accuracy in range-finding. A comparison between their outputs can be seen below.

Figure 3b. L515 RGB output

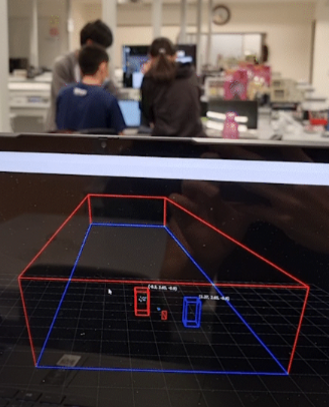


Figure 3a. IWR6843 output

Figure c. L515 Depth sensor output showing that a rectangular pillow is the nearest object in view.

## 2.1 Jetson Nano, MG995 Servo Motors, and PCA9685 Servo Driver

|  |  |
| --- | --- |
| Power to servo motor | 6 V from 4 x AA alkaline cells |
| Actuation range | 120° |
| Pulse width range | 1000 – 2000 μs (resolution of 4 μs at 60 Hz update rate) |
| Operation speed | 0.6 0.2 / 60° (with L515 mounted) |
| Torque | 0.83 N m (8.5 kgf cm) |

A picture containing indoor, wall, person, hand

Description automatically generatedNVIDIA's Jetson Nano is a popular small computer used for simple AI and robotics applications [1]. Here it serves as the connection point to L515's pan-tilt mount. By expanding its computing power with an external tensor processing unit (TPU), we could, in principle, enhance its edge computing capabilities to carry out object detection and tracking via L515's data, but we did not attempt this as we are already doing the computation on a more powerful PC. Two MG995 servo motors are used to enable pan and tilt motions of the L515. The key specifications of the servo are tabulated below.

Figure . Pan-tilt mount for L515

Table 5. MG995 specifications

In our system, the servo motor for panning is mounted a simple 3D-printed chassis, while the other is affixed onto a metal bracket on top of the first (cf. Fig. 4). The pan-tilt mechanism is mounted onto a tripod via the typical 1/4" - 20 UNC thread mounting point. The 12-bit PCA9685 pulse width modulation (PWM) driver is controlled with a 3.3 V logic from the Jetson Nano via I2C. The former comes with a built-in 25MHz clock and is capable of driving up to 16 servo motors or LEDs [2].

## 2.2 IWR6843 AOP EVM (mmWave RADAR)

The IWR6843 AOP EVM is a millimeter-wave RADAR that operates on gigahertz frequencies [3]. Due to the relatively higher frequency and thus, shorter wavelength, it has a higher accuracy but lower range compared to other RADARs [3].

### 2.2.1 Specifications

The specifications for the IWR6843 AOP EVM is based on tests conducted in indoor environments with normal room conditions: a 20m by 8m laboratory and an 8m by 5m bedroom.

Table 6. IWR6843 Specifications

|  |  |
| --- | --- |
| Frequency | 60 GHz |
| FOV (horizontal) | 120° |
| Range | 16 m |
| Maximum velocity | 6 m/s |
| Minimum velocity | 0.032 m/s |
| Max number of targets | 13 |
| Accuracy\* | ~ 80% (50% to 90%) |

\*Refers to the ratio of range readings of target by sensor to the actual range of target

The maximum number of targets tracked by the IWR6843 is low as the RADAR is not able to distinguish separate targets which are close to one another. Additionally, the sensor may lose track of the targets occasionally and cause abnormal drops in accuracy. The tunings and configurations performed will be described in Sec. 4.3.

### 2.2.2 Signal specifications

The IWR6843 operates using frequency-modulated continuous wave (FMCW) of approximately 60 – 64 GHz, which is unlike conventional radar systems that transmit short pulses at regular intervals. Due to this, the IWR6843 can obtain the following information from the transmitted and received signals:

1. Range of target
2. Velocity of target
3. Acceleration of target
4. Angle of Arrival (AoA) of signal which corresponds to azimuth and elevation of target

The range of the target, d, is the product of the speed of light, c, and the time between the transmitted and received signal, [3]. Target velocity is obtained from the phase difference between the signals received from two time-separated chirps, whereupon acceleration is estimated from the change in velocity through a state estimator [3].

A picture containing text, clock, watch

Description automatically generatedTarget velocity is obtained from the phase difference between the signals received from two time-separated chirps (Fig. 5a and Eq. 2), whereupon acceleration is estimated from the change in velocity through a state estimator [3]. The azimuth and elevation of the target is obtained from the phase difference in the signal received by two different antennas of the same signal (Fig. 5b and Eq. 3) [3].

Diagram

Description automatically generated

Figure 5b. Phase difference between space-separated chirps

Figure 5a. Phase difference between time-separated chirps

## 2.3 L515

The Intel RealSense L515 is a solid state time-of-flight depth sensing 3D camera [4] that uses infrared light with a wavelength of 860nm to collect depth data. The depth error average is around 14mm at the maximum depth range (9 m) of the L515 [5]. The L515 is 61mm in diameter and 26mm in height. It weighs about 100g, and the power required is 3.5W, which is much lower than the other TOF depth sensors. This is because the L515 uses a proprietary micro-electro-mechanical systems (MEMS) mirror scanning technology, enabling better laser power efficiency compared to other TOF technologies [5]. With an internal vision processor, the L515 reduces motion blur artifacts by an exposure time of less than 100 ns, making it suitable for recording fast moving objects. Additionally, the 4 ms photon-to-depth latency enables real-time applications such as our Cluster Tracker [6]. The L515 does not require an external computer for depth-processing, as there is an onboard vision processor that processes depth data at a rate of 23 million depth points per second.

We use the L515 to detect and track people standing close together using the L515's RGB sensor with an object detection algorithm. To determine whether they are standing too close together, we make use of its depth data.

### 2.3.1 Specifications

The specifications of L515 are based on the experiments conducted in the laboratory, as well as in a bedroom with low ambient lighting. Here we find that the depth sensor has a range much smaller than the RGB sensor. This means that even though humans can be detected from 16 m away, the distance of those humans away from the L515 will not be accurate, as the average range of the depth sensor, based on tests with several different coloured attires in the lab and bedroom, is 5.6 m. The L515 requires USB-C® 3.1 Gen 1 connection for the resolution of the RGB and depth sensors to be in 1960 x 1080 pixels and 1024 x 768 pixels, respectively. The tables below summarise these specifications.

Table 7. L515 specifications

|  |  |
| --- | --- |
| Depth sensor | |
| Range | 5.6 m |
| Horizontal FOV | 70° |
| Accuracy\* | ~ 93.87% |
| RGB sensor | |
| Range | 16 m |
| Horizontal FOV | 55.8° |
| Max number of targets | 21 |

\*Refers to the ratio of range readings of target by sensor to the actual range of target

### 2.3.2 Depth data

The distance of objects away from the camera is used to create a depth map. Each pixel in the depth image represents how far the corresponding object is from the sensor. However, the RGB and depth sensor have different FOVs. The RGB and depth streams need to be aligned before obtaining the pixel value from the RGB image, where the object detection takes place. Aligning the streams refers to reconstructing a depth image using the origin and dimensions of the RGB sensor. For each pixel in depth image, a geometric transformation based on the depth data is done. A new frame with the same size as the RGB stream is generated, which, instead of containing the RGB data, contains depth data calculated in the RGB sensor coordinate system. Then, the original RGB and the re-projected depth frames can be used to determine the depth value of pixels representing humans.

### 2.3.3 Distance between targets

To obtain the distance between detected targets, the deprojection function from the Intel RealSense Python library is used. Deprojection takes in intrinsic camera parameters, such as depth data, and the 2D pixel coordinate, and returns the corresponding 3D point coordinate. The stream of images has a 2D coordinate space specified in pixels values. The 2D pixel coordinate [0,0] refers to the centre of the pixel at the top left corner of the image, and from the perspective of the camera, the x-axis points to the right and the y-axis points downwards. Pixel coordinates are used to index images to find the content of the specified pixels. The stream of images also has a 3D coordinate space. The 3D point coordinate [0,0,0] refers to the centre of the image. From the perspective of the camera, the positive x-axis points to the right, the positive y-axis points downwards, and the positive z-axis points forward. The stream's 2D and 3D coordinate system are related to the intrinsic camera parameters for the L515. Using the corresponding 3D x and z coordinate of the detected humans, distance between humans can be calculated using the Pythagorean theorem.

# 3. Algorithm

This section describes the detailed chronological sequence of events that takes place during operation. The overall schematics of the algorithm in the cluster tracker is shown in Fig. 2 in Sec. 1.3.

## 3.1 IWR6843 on-chip algorithm

**Detection Layer**

The cluster tracker algorithm starts from the built-in chip of the IWR6843, where Fast-Fourier transform is performed to identify the frequency peaks. Next, static clutter is removed by identifying points of minimal velocity and the heatmap of range, azimuth, and the covariance matrix are obtained through a capon beamformer. CFAR-CASO (Constant False Alarm Rate) algorithm is used to identify and remove false detected points by specifying a threshold [7], following which the elevation and doppler are estimated for the legitimate data points.

**Tracking Layer**

After the data points are detected, they are now clustered into targets and are tracked. First, the points are tagged according to the boundary values configured corresponding to the dimensions of the room (currently set to 12m by 8m), and points that falls outside this boundary are ignored. Next, an extended Kalman Filter[[2]](#footnote-3), which is a state estimator [8], predicts the next state and covariance matrix of the targets where state vector = . (Initial iteration does not have the prediction step as no targets are formed yet)

In the association step, gates are formed around each target based on the gating parameters configured which should try to match to the dimensions and velocity of the target as closely as possible. DBSCAN[[3]](#footnote-4) (Density-Based Spatial Clustering for Applications with Noise) is performed for each data point where the Mahalanobis distance[[4]](#footnote-5) is calculated between each point and each target. For the points that falls outside the gating parameters, a bidding score system is used where they will be assigned to the target of the highest bidding score. Each point has a unique associated target ID and cannot be assigned to more than 1 target ID.

The remaining points that are not associated with any targets are then allocated to a new target (clustering). In order to qualify as a new target, several parametric tests are performed which include the minimum number of points and combined Signal-to Noise Ratio (SNR). The algorithm then checks for the presence of targets within the occupancy area and determines it based on occupancy thresholds. Lastly, the targets’ state (state here differs from the state mentioned earlier in the extended Kalman Filter during the prediction step) is determined and maintain based of these 4 states:

1. Detect: detecting phase to determine the legitimacy of a detected target
2. Active: target is confirmed and currently being tracked
3. Static: target is not moving but is still being tracked
4. Free: target is lost

## 3.2 Server Script: Serial output from IWR6843

The data output of the IWR6843 is dependent on the image flashed onto the hardware. For our Cluster Tracker, the IWR6843 outputs the serial data through UART (Universal Asynchronous Receiver Transmitter) which is read by the Host PC through a USB port installed with a USB-to-UART driver. The serial output can be read and parsed using the Pyserial library from Python. The serial outputs are of little endian byte order and are TLV(type/tag-length-value) encoded (cf. Annex A). The output consists of a list of targets with their metadata such as their unique ID, position, velocity, and acceleration in cartesian coordinates.

## 3.3 Server script

In the server script, two Python dictionaries (hash table) of checked targets and unchecked targets are maintained, where the keys are the IDs of the targets and the values are python objects containing their metadata. Dictionaries are used due to the fast lookup and modification of O(1) time complexity [11], as numerous operations are performed to check for the existence of targets in the two data containers. The unchecked targets dictionary acts a queue where new targets to be checked are appended to the dictionary while the current target is constantly fed to the Jetson Nano to be checked for violation. A Socket is opened to communicate via SSHV2 Protocol with a Python script in Jetson Nano using the Python’s Paramiko library. Additionally, a pipe is opened that allows the server script (parent) to receive data from the L515 script (child) that is running in parallel.

For each IWR6843 frame or each loop of the server algorithm, after the metadata of a target is parsed from the IWR6843 serial output (Sec. 3.2), the script checks for its existence within the checked and unchecked targets, and if it is not present in either, it is queued into the unchecked targets dictionary, and if they exist in either of the list, their metadata is updated instead. At the same time, the script checks if the targets in the checked targets are in the list of detected targets in the current frame of the IWR6843. If any of the checked targets are not present in the current frame, they are removed from the checked targets as these targets are freed and there is a possibility that the ID will be reused by the IWR6843 on-chip algorithm.

In each loop, the script also allocates 2 seconds of tracking for each target. If the time difference between the start tracking time of the current target and the current time is more than 2 seconds, the L515 will be freed from the current target and the target will be moved from the unchecked to the checked targets dictionary. Then, the first target in the unchecked list will be sent for checking, by sending its location to the Jetson Nano. The time is then recorded as the starting tracking time of the current target. If the time difference between the current loop the starting time of the current tracked targets is less than 2 seconds, the system continues to track the current target, and at the same time constantly send the updated target location to the Jetson Nano such that it controls to L515 to constantly follow the currently tracked target only.

Simultaneously, the script receives violation data from the L515 script via the multiprocessing library’s pipe. If there is a violation, the location of the violation obtained from the IWR6843 and the distance from the system obtained from the L515 are shown on terminal. The target directly associated with the violation can be seen from the viewfinder of the L515 script. The first 0.1 seconds of a new track will not be receiving input from the L515 in order to allow sufficient time to turn it.

In each frame, while the algorithms check for new unchecked targets to queue to the unchecked targets dictionary, the algorithm also checks for the distance between each target (represents cluster) detected by the IWR6843. Should any of the targets be less than 2m apart from any other targets, signifying that the cluster of people might be breaching social distancing rules, the target is inserted into the front of the queue and thus would be the next immediate target sent to the Jetson Nano for checking.

## 3.4 Jetson Nano

Ethernet access to the Jetson Nano is enabled through the micro-USB port of the Jetson Nano, which creates a virtual network between the Host PC and the Jetson Nano. At the initialisation phase of the server script, Jetson Nano is prompted to run a Python script locally. The script constantly receives input from the Host PC in the form of floating-point values corresponding to the angles of the targets and passes them to the PCA9685 servo driver, which will then activate the corresponding PWM signals to drive the servo motor, and hence L515, to the correct orientation.

## 3.5 L515

At the start of the server script, the server script starts the L515 script and it is ran as a parallel process alongside the server script. A pipe connecting these two scripts is formed where the server script is the parent process and the L515 script is the child process.

The RGB data from the sensors is propagated through the YOLOV5 object detection model where the image is first separated into grids of varying dimensions and the bounding box regressor in the model performs a forward pass on the image; then each cell predicts the coordinates of the bounding boxes and their corresponding confidence score [12]. Next, the class of the object in the bounding box is predicted by propagating it through a convolutional neural network [12]. As there may be overlapping bounding boxes, the Intersection Over Union (IOU) calculates the ratio of intersecting area to the union area for each bounding boxes and then discards those bounding boxes with IOU higher than a configured threshold, displaying only the one with the highest class probability [12].

DeepSORT comprises of deep learning and a Kalman filter to associate the detected targets in each frame to their respective tracks [13]. The state vector in the Kalman filter consists of the feature vector of the bounding box, containing the features of the bounding box like its coordinates [13]. The feature vector also contains features extracted from deep learning. The Kalman Filter then makes new predicted tracks based on the current state vectors and duplicated tracks are removed based on a threshold number of detections [13]. Then Mahalanobis distance between the predicted tracks and the newly detected objects are then associated using the efficient Hungarian algorithm[[5]](#footnote-6) [13]. The Mahalanobis distance here not only measures the distance between the bounding box coordinate, but also the distance (i.e. amount of difference) between features of the predicted and the actual detected tracks [13].

After the bounding box of the detected objects and their respective classes are generated, the L515 scripts filter out all the other classes except for humans. Now the depth profile from the L515 is mapped over the RGB sensors and the distance to the object is obtained. Following this the distance matrix containing all the distances between each target is computed and a violation occurs when any distance is less than a pre-configured threshold (currently set at 2 meters, cf. Fig. 6a and Fig. 6b). Next, a Boolean variable representing the presence of a violation along with the location of the target is stored and sent back to the main server script through the pipe.

# A picture containing person, floor, indoor, standing Description automatically generatedA picture containing person, indoor, floor, standing Description automatically generated4. Configuration and Tunings

Figure 6a. Blue boundary box indicates no violation

Figure 6b. Red boundary box indicates a violation

Due to the complexity of the system, the setup must be tuned and configured carefully. The height and angle of the sensors’ placements are calibrated and the parameters for each sensor are tuned.

## 4.1 Setup

A picture containing text, electronics

Description automatically generatedThe electronics assembly for the Cluster Tracker is straightforward and is summarised below.

Figure 7. The wiring of the set-up. The IWR6843 and L515 are directly connected to and powered by the host PC via USB-A.

As both sensors are connected to a single PC, we simply put them side-by-side at a height of around 130 cm above the ground. As mentioned in Sec. 2.1, it is also possible to run the L515 script on a separate computer; hence, the two sensors can be placed at different positions in the room, such that they are able to cover different regions and fit into the specific needs of the user. For on ground testing, we set the L515 to be vertically aligned and only enabled it to be panned, sweeping from -60° to 60°, where 0° is its home angle at which its view is exactly parallel with that of the IWR6843. Because the separation between the two sensors is less than 15 cm and given that the L515 already covers a 70°-field, we can approximate them to have the same line of sight.

## 4.2 IWR6843 on-chip algorithm parameter tunings

The on-chip algorithm of the IWR6843 has numerous parameters to be configured and tuned. Due to the limited time, not all of them were explored.

The velocity resolution was increased to detect slower and smaller motions such that sufficient data points can be captured, which involve static clusters that are otherwise removed. This is key to detecting people even when they are almost stationary. In order to achieve the higher velocity resolution, the idle time was increased from 30 to 60 and the ramp end time was increased from 59.10 to 70. Consequently, the chirp time was also increased, which, in turn, increased the frame periodicity, so as to keep the total chirping and processing time within the frame period. This drops the frame rate, but to a value that remained tolerable for our application.

Fine motion configuration is a new API introduced to the firmware that allows the running of two different chirp blocks in memory to utilise the current and previous frames’ chirp sub-blocks and thus, virtually increase the chirping window [3]. This allows the RADAR to detect finer motion due to the longer chirping window, albeit at the expense of more memory usage [3]. Here, the number of chirping loops within a frame is halved from 96 to 48. The drop in chirping loops decreases the velocity resolution in a single frame but is recovered through the data processing across multiple frames [3]. This mode will also decrease the SNR by about 3 dB, and can be recovered by enabling the BPM-MIMO (Binary-Phase-Modulation-Multiple-Input-Multiple-Output) mode across two transmission antennas. This allows simultaneous transmissions on them, while ensuring clear separation through binary spatial encodings where the phases of the signals are only either or [3]. However, the attempt to enable BPM-MIMO mode failed and thus, the current Cluster Tracker suffers from the 3dB drop in SNR. Nonetheless, it should not be operationally critical in this application as decreasing the SNR threshold leads to more false positives.

Spatial resolution was increased by increasing the angular search grid (search angle resolutions in azimuth and elevation) in the Capon beamformer. Even though this increased the number of detected points, it had no significant effect on the Cluster Tracker as it was increasing the detected points in the regions where presence of targets was already detected. If the IWR6843 is tested beyond the parameters of our environment, an increase in spatial resolution would improve the performance at longer ranges and, in a sense, increase the range of the IWR6843.

After configuring the velocity resolution, the number of detected points increased considerably; however, the number of tracked targets only increased slightly. Hence, the tracking parameters were tuned. Gating parameters were adjusted but eventually configured back to the original settings as no significant improvement was observed. The minimum number of data points to allocate to a new target was dropped from 20 to 8 to increase the number of targets detected. Combined SNR threshold was also decreased from 40 to 20. Keeping in mind that our application is stricter on the false negatives detection but more lenient on the false positive detection, the parameters were tuned such that the system is as aggressive as possible. The number of continuous frames missed to change the target’s state from *Detect* to *Free* was increased from 3 to 6 and the number of continuous frames missed to change the target’s state from static to free was increased from 20 to 300. Our environment contains many static targets, and hence, the number of frames allowable in the static state was increased from 1000 to 6000.

As a result, the number of tracked targets increased dramatically from a maximum of 4 targets to 13 targets. The number of targets were based on an environment with approximately 20 human targets and the 13 targets detected could be due to the inability of the RADAR to distinguish separate entities in a cluster which is expected and hence, the need for the L515.

## 4.3 L515 Artificial Intelligence

To locate the presence of human with a bounding box, the YOLOv5 object detection model is used to detect the human class. YOLO stands for You Only Look Once and it is an object detection framework using only a single convolutional network. YOLO is much faster than other object detection models because it scans the whole input image at once, as opposed to sweeping it pixel-by-pixel. YOLOv5 provides highly accurate, fast models with its weights and biases trained on the Common Objects in COntext (COCO) dataset which consists of 80 object categories, including humans.

However, the choice of YOLOV5 is due to its aggressive detection rate rather than its speed. It was compared with the faster Region-Based Convolutional Networks (R-CNN) model which we tested to have a relatively lower false positive rate but higher false negative rate and a slightly slower inference time. Moreover, the difference in decreased false positive rate of about 3% was too small to justify a change in our model.

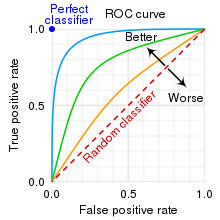


Figure 8. Example of AUC-ROC curve

The low confidence threshold was configured to be 0.4 to lower the false negative rates. For other applications of this AI model that is sensitive to both true and false positive detections (i.e. sensitivity), the confidence threshold hyperparameter can be tuned by plotting the Receiver Operating Characteristics (ROC) curve, where the true detections are plotted against the false detections for each confidence threshold and the optimal confidence is the one that corresponds to the curve with the highest Area Under Curve (AUC) (cf. Fig. 8).

While larger models like YOLOv5s will produce more accurate results, they have more parameters, which requires more CUDA memory to train, resulting in lower frame rates per second for the L515 [14]. Thus the model used is the nano model YOLOv5n, which maintains the YOLOv5s depth multiple of 0.33 but reduces the YOLOv5s width multiple from 0.50 to 0.25, leading to a reduction in the number of parameters from 7.5 million to 1.9 million [15].

# 5. Evaluations and Key Considerations

This system has a large redundancy in components used. For example, the powerful Jetson Nano is only being used as a microcontroller to control the servo motors and can be replaced with just a simple Arduino board. The reason for this is because the entire system was meant to be ran on the Jetson Nano. However, due to the different architectures and instruction sets of x86 (CISC) and ARM (RISC), the mmWave SDK could not be installed onto the Jetson Nano. Therefore, the host PC was introduced to the system, which adds another dimension to the complexity.

However, this allowed us to use more computationally intensive but better performing artificial intelligence models of YOLOv5 and DeepSORT. According to the initial plan of running the entire system on the Jetson Nano, less computationally intensive but poorer performing artificial intelligence algorithms were considered such as the MobileNet-SSD, Haar Cascade, and HOG (Histogram of gradients). As the accuracy of the object detection is not the limiting factor of the system’s accuracy, less emphasis was placed on the system’s AI. Thus, the weights and biases are trained on a general dataset rather than those specifically for humans. This is such that the system can easily be configured to perform other tasks such as projectile and balloon tracking. However, the re-tuning and recalibrating of the IWR6843 might be challenging.

For the Cluster Tracker, the accuracy of the system is undermined by the IWR6843’s inability to track a target consistently. Although data points are sufficient after the tunings, the targets are occasionally lost, which could be due to the incorrect settings for some of the parameters. For example, the gain in the gating parameter, which estimates the errors and uncertainties in the detection algorithm, might not be configured to match the actual errors and uncertainties.

Currently, the Cluster Tracker has only been tested indoors where the brightness and noise levels of the environment is well-controlled. In outdoor settings, noise and ambient light might affect the sensors’ performances. The IWR6843 was mostly reconfigured to increase the velocity resolution, but should the cluster tracker be deployed in a larger environment, increasing the spatial resolution should also be considered.

Additionally, the current system has only been tested in an environment with low volume of human traffic. If the traffic of the environment increases such that number of targets to be checked is added at a faster rate than the rate at which the Cluster Tracker takes to detect a target (currently 1 target per 2 seconds), the queue of unchecked targets will always be increasing and cause a latency in the system. At this point, it would be better to configure it so that unchecked target is directly fed to the Jetson Nano rather than appending it to the queue. However, this might cause the tracker to miss some of the targets.

The cluster tracker might not be the perfect showcase of the capabilities of this sensor fusion as the velocities and accelerations of the targets obtained from the IWR6843 were not utilised. However, if desired, it is possible to map the velocity and acceleration obtained from the IWR6843 to the point clouds obtained from L515, thereby allowing for more complex applications such as avoidance systems and even target interception.

# 6. References

[1] “Jetson Nano Developer Kit,” *NVIDIA Developer*, Mar. 06, 2019. https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed Nov. 15, 2021).

[2] A. Industries, “Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface.” https://www.adafruit.com/product/815 (accessed Nov. 15, 2021).

[3] C. Iovescu and S. Rao, “The fundamentals of millimeter wave radar sensors,” p. 9, 2020.

[4] “Optimizing the Intel RealSense LiDAR Camera L515 Range,” *Intel® RealSenseTM Depth and Tracking Cameras*, Aug. 06, 2020. https://www.intelrealsense.com/optimizing-the-lidar-camera-l515-range/ (accessed Nov. 15, 2021).

[5] “Intel® RealSenseTM LiDAR Camera L515,” *Intel® RealSenseTM Depth and Tracking Cameras*. https://www.intelrealsense.com/lidar-camera-l515/ (accessed Nov. 15, 2021).

[6] “LiDAR Depth Camera L515 - Intel | DigiKey.” https://www.digikey.sg/en/product-highlight/i/intel/lidar-depth-camera-l515 (accessed Nov. 15, 2021).

[7] M. E. Yanik, “Detection Layer Parameter Tuning Guide for the 3D People Counting Demo – Rev 2.0,” p. 55.

[8] “Estimate states of discrete-time nonlinear system using extended Kalman filter - Simulink - MathWorks 中国.” https://ww2.mathworks.cn/help/control/ref/ekf\_block.html (accessed Nov. 15, 2021).

[9] K. S. do Prado, “How DBSCAN works and why should we use it?,” *Medium*, Jun. 03, 2019. https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80 (accessed Nov. 15, 2021).

[10] Stephanie, “Mahalanobis Distance: Simple Definition, Examples,” *Statistics How To*, Nov. 21, 2017. https://www.statisticshowto.com/mahalanobis-distance/ (accessed Nov. 15, 2021).

[11] “Complexity Cheat Sheet for Python Operations,” *GeeksforGeeks*, Mar. 02, 2020. https://www.geeksforgeeks.org/complexity-cheat-sheet-for-python-operations/ (accessed Nov. 15, 2021).

[12] “Introduction to YOLO Algorithm for Object Detection,” *Engineering Education (EngEd) Program | Section*. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/ (accessed Nov. 15, 2021).

[13] “DeepSORT: Deep Learning to track custom objects in a video.” https://nanonets.com/blog/object-tracking-deepsort/ (accessed Nov. 15, 2021).

[14] “GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite.” https://github.com/ultralytics/yolov5#pretrained-checkpoints (accessed Nov. 15, 2021).

[15] “ultralytics/yolov5: v6.0 - YOLOv5n ‘Nano’ models, Roboflow integration, TensorFlow export, OpenCV DNN support | Zenodo.” https://zenodo.org/record/5563715#.YZJQD1MRX0p (accessed Nov. 15, 2021).

ANNEX A

TLV encodings for the IWR6843 serial data output are as follows:

**Frame Header (48 bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Synchronization Pattern | Uint64 | 8 | “02 01 04 03 06 05 08 07” |
| Version | Uint32 | 4 | Version of SDK |
| Total Packet Length | Uint32 | 4 | Inclusive of Header |
| Platform | Uint32 | 4 | 0xA6843 |
| Frame Number | Uint32 | 4 | - |
| Sub Frame Number | Uint32 | 4 | - |
| Chirp Processing Margin | Uint32 | 4 | - |
| Frame Processing Margin | Uint32 | 4 | - |
| Track Processing Time | Uint32 | 4 | - |
| UART Sent Time | Uint32 | 4 | - |
| Number of TLVs | Uint16 | 2 | - |
| Check Sum | Uint16 | 2 | Header check sum |

For each TLV payload, there is a header which is encoded as follows:

**TLV Header (8 bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Type | Uint32 | 4 | “06”, “07” or “08” |
| Length | Uint32 | 4 | Length of current TLV inclusive of TLV header |

There are 3 types of TLV payloads and they are:

|  |  |
| --- | --- |
| Type Number | Type |
| 06 | Point cloud |
| 07 | Target object list |
| 08 | Target index |

**Point Cloud TLV (Type 06)**

For point cloud TLVs, the first section of this payload is a structure consisting of the units to be multiplied to each of the point’s attributes. This is so that data can be compressed into lesser bytes. The point unit structure encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Elevation Unit | Float | 4 | - |
| Azimuth Unit | Float | 4 | - |
| Doppler Unit | Float | 4 | - |
| Range Unit | Float | 4 | - |
| SNR Unit | Float | 4 | - |

The rest of this TLV payload are an array of point structure of 8 bytes each and the encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Elevation | Int8\_t | 1 | - |
| Azimuth | Int8\_t | 1 | - |
| Doppler | Int16\_t | 2 | - |
| Range | Uint16\_t | 2 | - |
| SNR | Uint\_16 | 2 | - |

**Target List TLV (Type 07)**

Target Object List TLVs contains an array of targets and their respective metadata. Each target are encoded in 112 bytes and the encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| ID | Uint32 | 4 | ID of target |
| x-coordinate | Float | 4 | - |
| y-coordinate | Float | 4 | - |
| z-coordinate | Float | 4 | - |
| x-velocity | Float | 4 | - |
| y-velocity | Float | 4 | - |
| z-velocity | Float | 4 | - |
| x-acceleration | Float | 4 | - |
| y-acceleration | Float | 4 | - |
| z-acceleration | Float | 4 | - |
| Ec[16] | Float | 16 x 4 | 4 x 4 error covariance matrix (range, azimuth, elevation, doppler) |
| Gating function gain | Float | 4 | - |
| Confidence level | Float | 4 | - |

**Target Index TLV (Type 08)**

Target Index TLV consist of a list of target ID of 1 byte each and are encoded as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| ID | Uint32 | 4 | ID of target |

1. Refers to the average percentage number of violations detected by the system. If false positive detections are considered, the accuracy drops to 62%. [↑](#footnote-ref-2)
2. The non-linear version of the classical Kalman Filter [8]. [↑](#footnote-ref-3)
3. DBSCAN is a data clustering algorithm that unlike K-means clustering, is able to cluster non-spherical data [9]. [↑](#footnote-ref-4)
4. Euclidean distance between a data point and a distribution of data [10]. [↑](#footnote-ref-5)
5. Efficient combinatorial solver using bipartite graph data structure [↑](#footnote-ref-6)