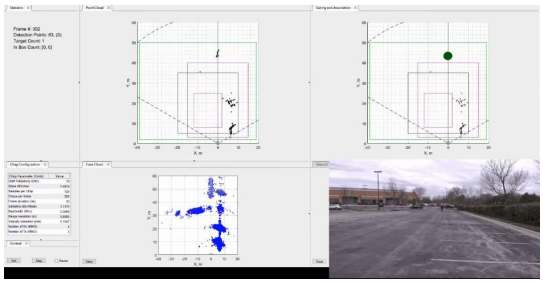
Overview

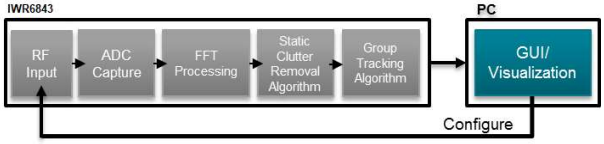
This lab demonstrates the use of TI mmWave sensors to count and track multiple people simultaneously up to 50m away. Detection and tracking algorithms run onboard the IWR6843 mmWave sensor and are used to localize people and track their movement with a high degree of accuracy. mmWave sensors can reduce false detections from challenging environments such as direct sunlight, no-light, fog, or smoke, and are particularly suited for privacy-conscious applications. In this demonstration, localization and tracking is performed upon any moving object in the scene; static objects such as chairs, tables, and walls are ignored. The IWR6843 device outputs a data stream consisting of point cloud information and a list of tracked objects which can be visualized using the software included in this lab.

Detection Range and FOV have improved. Please see results in section 4.2.2

ES2.0 Devices Only

This lab only runs on ES2.0 devices.





Quickstart

1. Hardware and Software Requirements

Hardware

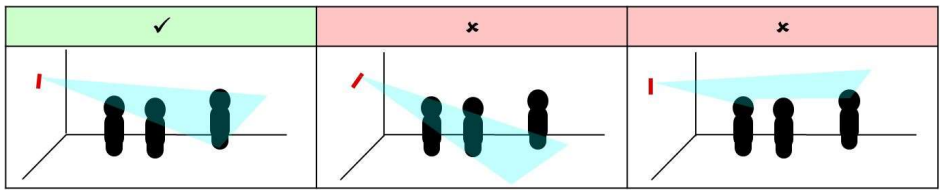
Item	Details
Device	Industrial mmWave Carrier Board (http://www.ti.com/tool/MMWAVEICBOOST) and IWR6843 Long Range Antenna Board (http://www.ti.com/tool/IWR6843ISK).
Mounting Hardware	The EVM needs to be mounted at a height of ~2.0-2.5m with a slight downtilt. An adjustable clamp style smartphone adapter mount for tripods (https://www.amazon.com/Vastar-Universal-Smartphone-Horizontal-Adjustable/dp/B01L3B5PBI/) and a 60-75" tripod (https://www.amazon.com/Neewer-Portable-centimeters-Camcorder-kilograms/dp/B01N6JCW8F/) can be used to clamp and elevate the EVM. This is only an example solution for mounting; other methods can be used so far as setup specifications are met.
Computer	PC with Windows 7 or 10. If a laptop is used, please use the 'High Performance' power plan in Windows.
Micro USB Cable	Due to the high mounting height of the EVM, an 8ft+ cable or USB extension cable is recommended.
Power Supply	5V, 3A with 2.1-mm barrel jack (center positive). The power supply can be wall adapter style or a battery pack with a USB to barrel jack cable.
Tape Measure	

Software

Tool	Version	Required For	Download Link
mmWave Industrial Toolbox	Latest	Contains all lab material.	mmWave Industrial Toolbox (http://dev.ti.com/tirex/explore/node?node=AJoMGA2ID9pCPWEKPi16wg__VLyFKFf__LATEST)
MATLAB Runtime	2017a (9.2)	Quickstart Visualizer	To run the quickstart visualizer the runtime (https://www.mathworks.com/products/compiler/matlab-runtime.html) is sufficient.
Uniflash	Latest	Quickstart Firmware	Download offline tool (http://www.ti.com/tool/UNIFLASH) or use cloud version (https://dev.ti.com/uniflash/#!/)

2. Physical Setup

1. Follow the instructions for Hardware Setup of ICB for Functional Mode (../../../common/docs/hardware_setup/hw_setup_antenna_module_and_carrier_for_functional.html)
2. For best results, the EVM should be positioned high enough to be above the top of tracked objects and with a slight down tilt. The aim is to position the EVM so that the antenna beam can encompass the area of interest. If the down tilt is too severe, noise from ground clutter would increase and the effective sensing area would decrease. If threre is no down tilt, counting performance would be worse for cases in which one person is in line with and shielded by another person. Given the antenna radiation pattern of the EVM, consideration should be taken to not mount the EVM too close or oriented with beam directed to the ceiling as this can increase the noise floor and result in less optimal performance.



Setup Requirements:

- Elevate EVM: 2.0-2.5m high
- Down tilt: ~2-3 degree

Setup using suggested tripod and smartphone clamp mount:

1. Screw on clamp mount to tripod
2. Clamp EVM across its width below power barrel jack to attach EVM
3. Adjust tripod head for ~2-3 degree down tilt (Tip: Bubble or level smartphone apps can be used to measure down tilt)
4. Plug in micro-usb and power supply to EVM
5. Extend tripod so that the EVM is elevated 2.0-2.5m from the ground
6. Position EVM and tripod assembly in desired location of room. The EVM should be positioned so that the 120 degree FOV of the EVM antenna encompasses the area of interest and points to the region in which people are expected to enter the space.

3. Flash the EVM

- Follow the instructions for Hardware Setup of ICB for Flashing Mode (../../../common/docs/hardware_setup/hw_setup_antenna_module_and_carrier_for_flashing.html)
- Follow the instruction to Flash the mmWave Device (../../../common/docs/software_setup/using_uniflash_with_mmwave.html)

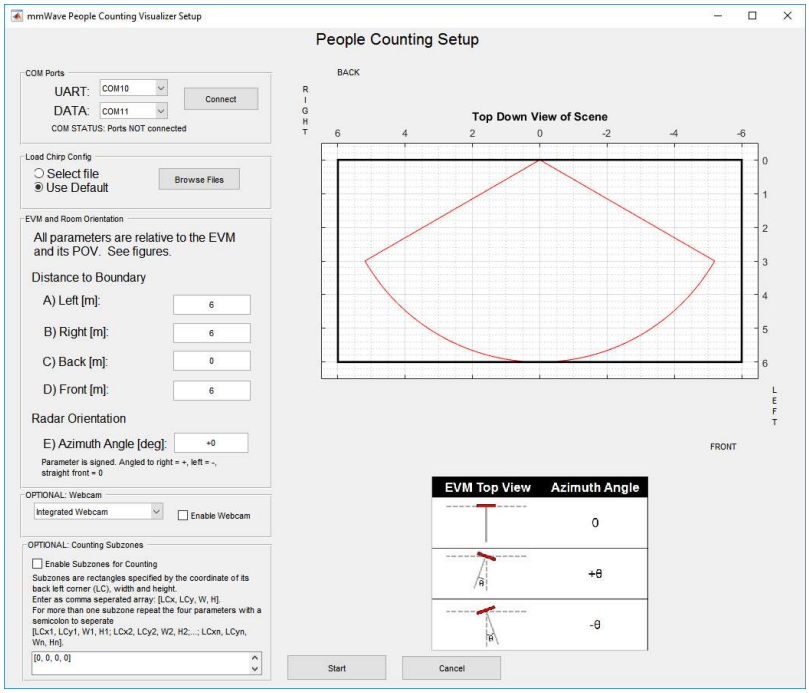
Image	Location
Meta Image 1/RadarSS	`C:\ti\<mmwave_industrial_toolbox_install_dir>\labs\long_range_people_detection\68xx_long_range_people_det\prebuilt_binaries\long_range`

4. Run the Lab

To run the lab, launch and configure the visualizer which displays the detection and tracked object data received via UART.

1. Launch the visualizer:

- Navigate to `C:\ti\<mmwave_industrial_toolbox_install_dir>\labs\long_range_people_detection\68xx_long_range_people_det\gui\long_range_people_detection_gui.`
- Run `pplcount_gui.exe`
- A black console log window will appear.
- After 30-60sec, the **mmWave People Counting Visualizer Setup** window should appear



2. Configure Visualizer

On the left side of the visualizer setup window are options and parameters for running the demo. On the right side are graphics to aid in understanding the configuration options. The plot titled **Top View of Scene** updates if **Chirp Configuration** or **EVM and Room Orientation** are changed. This plot illustrates how the visualizer thinks the radar is setup in a scene. The approximate field of view and range of the radar is depicted by the red outline. The thicker black rectangle shape represents the defined visualization area. Typically, if the demo is setup in an indoor environment, the visualization area should be defined to match the dimensions of the room.

The following sections will step through the setup requirements to run the people counting demo:

1. Select COM Ports

- Specify **UART** and **DATA** COM ports using the text boxes. Click **Connect** to open and connect to ports.

> **COM Status**
>
> Message should update to show that the COM ports have been connected before continuing.

2. Chirp Configuration

- A custom chirp configuration can be loaded or leave the default chirp developed for people counting selected.
  - To load a custom config: select **Select file** option and then click **Browse Files** button and choose desired '.cfg' file. The **Top Down View of Scene** plot will update the depiction of the radar range depending on the chirp loaded.

- The default chirp was developed for people counting in indoor environments with a max range of approximately 55m.
- The default chirp returns 2D point cloud data. There is also a chirp available that returns 3D data. The 2D chirp was chosen as default because it gives the best detection range. However, for some applications, 3D data may be valuable.
- Please see the below table for a performance comparison of the detection range at different angles of the 2D and 3D chirps

Angle of Arrival (degrees)	2D Approaching (m)	3D Approaching (m)	2D Departing (m)	3D Departing (m)
0	49	46	53	55
15	49	47	50	56
30	55	44	57	50
45	56	42	56	42
60	34	28	45	34

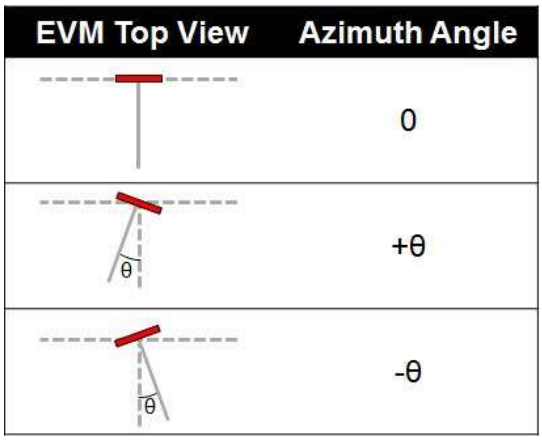
Expand for details of included chirps:

Default Chirp Parameters 2D and 3D

Chirp Parameter (Units)	Value 2D	Value 3D
Start Frequency (GHz)	60.0	60.0
Slope (MHz/us)	8.241	8.241
Samples per chirp	125	125
Chirps per frame	256	288
Frame duration (ms)	100	100
Sampling rate (Msps)	3.4330	3.4330
Bandwidth (GHz)	0.300	0.300
Range resolution (m)	0.49	0.49
Max Unambiguous Range (m)	60	60
Max Radial Velocity (m/s)	7.8806	7.8806
Velocity resolution (m/s)	0.1250	0.1250
Azimuth resolution (deg)	14.5	14.5
Number of Rx	4	4
Number of Tx	2	3

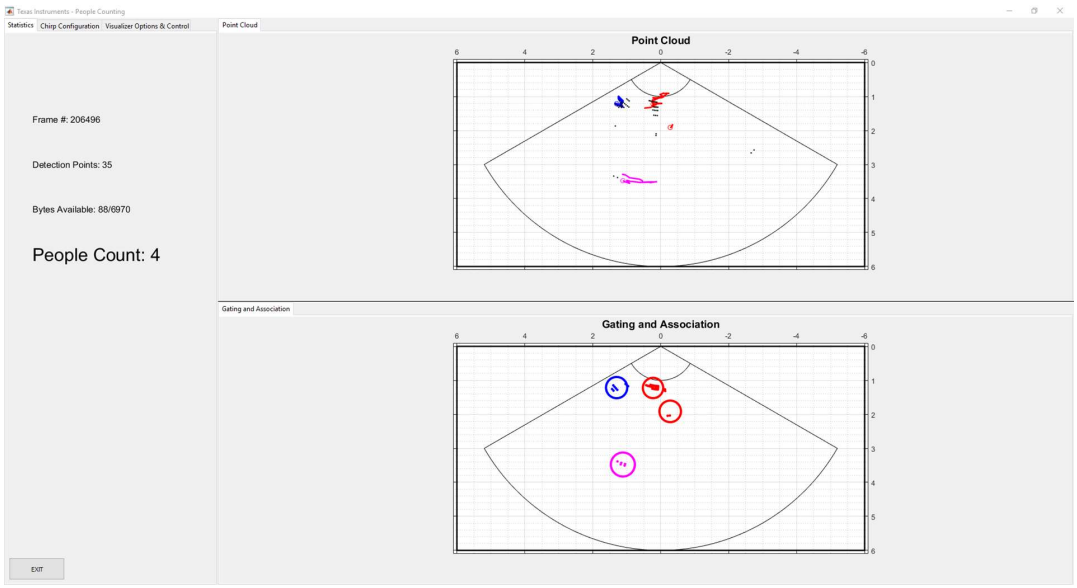
3. EVM and Room Orientation

- **Boundary Parameters A-D**: relate to the defined visualization area. They are measured from the perspective of and relative to the EVM. The EVM is always the origin.
    - Stated another way, changing A-D effectively zooms the plot in and out about the EVM origin. Changes to A-D are updated in **Top Down View of Scene** plot.
    - The default, pre-populated parameters correlate to plotting the radar's max FOV and the max range with the default chirp. They can be left as is if observing the maximum area of detection is desired.
    - If it is instead desired to define the visualization area to match the dimensions of the room then the parameters need to be modified.
    - To get a sense of the orientation directions, picture standing at the location the EVM is mounted and face straight ahead to the wall in front of you. The wall to your right is the right boundary. The distance from you (the EVM) to the right wall in meters is the value for parameter A). The same follows for the left, back, and front boundaries.
- **Azimuth Angle Parameter E**: azimuth tilt angle. If the EVM is not positioned facing straight ahead to the front boundary then it has an azimuth tilt angle. Specify the angle in degrees and view the change in the **Top Down View of Scene** plot. The red outline defining the sensor area will tilt either towards the left or right boundary.
    - This parameter is signed. If the EVM is tilted towards the right wall, include the + sign. For the left wall include the - sign. If EVM is oriented straight ahead the angle is 0 and the sign can be either. The figure below details the sign convention.



- OPTIONAL: Define Subzones for Counting
- Rectangular areas can be defined to specify specific regions for which to keep another count of the number of people present.
- For example, if two boxes are defined then the visualizer will report a total people count for all the people in the scene it detects, a Box 1 count, and a Box 2 count.
- **Launch Visualizer**
    - Click **Start** to launch visualizer with configurations specified.

6. Understanding the Output



The visualizer consists of:

- A top panel with a **Point Cloud** plot.
    - The black points represent the point cloud returned by the detection layer of the device.
        - Each new tracked object is assigned one of five possible colors (blue, red, green, cyan, and magenta).
        - The small colored ring represents the computed centroid of the point cloud for the tracked object.
        - The "snail trail" trace represent 100 frames of history of the tracked object's centroid.
- A bottom panel with a **Gating and Association** plot.
    - This plot visualizes the result of the tracking algorithm.
        - The colored points represent the detection points (black points in previous frame's Point Cloud plot) which are associated to a specific track. Unassociated points that do not belong to a track are not plotted.
        - The colored circular ring is centered over the centroid of of the tracked object. The diameter of the ring is related to the variance in location of the tracked object's detection points.
- A side panel with three tabs: Statistics, Chirp Configuration, and Visualizer Options
    - TIP: If lag is an issue, check the **Consolidate plotting** option in **Visualizer Options**. This will only display one of the two plots.

Quitting the Visualizer : To exit the visualizer use the exit button at the bottom left of the window. This will delete the open serial ports and save an output file of the session in fhist.mat.

Developer's Guide

Build the Firmware from Source Code

1. Software Requirements

Tool	Version	Download Link
mmWave Industrial Toolbox	Latest	mmWave Industrial Toolbox (http://dev.ti.com/tirex/explore/node?node=AJoMGA2ID9pCPWEKPi16wg__VLyFKFf__LATEST)
TI mmWave SDK	Latest	TI mmWave SDK (http://software-dl.ti.com/ra-processors/esd/MMWAVE-SDK/latest/index_FDS.html) and all the related tools are required to be installed as specified in the mmWave SDK release notes
Code Composer Studio	8.1.0	Code Composer Studio v8 (http://processors.wiki.ti.com/index.php/Download_CCS#Code_Composer_Studio_Version_8_Downloads)
TI SYS/BIOS	6.73.01.01	Included in mmWave SDK installer
TI ARM Compiler	16.9.6.LTS	Included in mmWave SDK installer
TI CGT Compiler	7.4.16	Version 7.4.16 must be downloaded and installed. Download link (https://www.ti.com/licreg/docs/swlicexportcontrol.tsp?form_type=2&prod_no=ti_cgt_c6000_7.4.16_windows_installer.exe&ref_url=http://software-dl.ti.com/codegen/esd/cgt_registered_sw/C6000/7.4.16)
XDC	3.50.08.24	Included in mmWave SDK installer
C64x+ DSPLIB	3.4.0.0	Included in mmWave SDK installer
C674x DSPLIB	3.4.0.0	Included in mmWave SDK installer
C674x MATHLIB (little-endian, elf/coff format)	3.1.2.1	Included in mmWave SDK installer
mmWave Radar Device Support Package	1.6.1 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
TI Emulators Package	7.0.188.0 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
Uniflash	Latest	Uniflash tool is used for flashing TI mmWave Radar devices. Download offline tool (http://www.ti.com/tool/UNIFLASH) or use the Cloud version (https://dev.ti.com/uniflash/#!/)

2. Import Lab Project

For the People Counting lab, there are two projects, the DSS for the C674x DSP core and the MSS project for the R4F core, that need to be imported to CCS and compiled to generate firmware for the xWR6843.

> **Project Workspace**
>
> When importing projects to a workspace, a copy is created in the workspace. All modifications will only be implemented for the workspace copy. The original project downloaded in mmWave Industrial Toolbox is not touched.

1. Start CCS and setup workspace as desired.
2. Import the project(s) specified below to CCS. See instructions for importing here (../../../../docs/readme.html#import-ccs-projects-from-the-mmwave-industrial-toolbox-into-code-composer-studio).
    - **long_range_people_det_68xx_mss**
    - **long_range_people_det_68xx_dss**
3. Verify that the import occurred without error: in CCS Project Explorer, both **long_range_people_det_68xx_mss** and **long_range_people_det_68xx_dss** should appear.

3. Build the Lab

The DSS project must be built before the MSS project.

1. Select the **long_range_people_det_dss** so it is highlighted. Right click on the project and select **Rebuild Project**. The DSS project will build.
2. Select the **long_range_people_det_mss** so it is highlighted. Right click on the project and select **Rebuild Project**. The MSS project will build, the the lab binary will be constructed automatically.
3. On successful build, the following should appear:
    - In long_range_people_det_dss → Debug, **long_range_people_det_dss.xe674** (this is the C67x binary used for CCS debug mode)
        - In long_range_people_det_mss → Debug, **long_range_people_det_mss.xer4f** (this is the Cortex R4F binary used for CCS debug mode) and **long_range_people_det_lab.bin** (this is the flashable binary used for deployment mode)

Selecting Rebuild instead of Build ensures that the project is always re-compiled. This is especially important in case the previous build failed with errors.

Build Fails with Errors

If the build fails with errors, please ensure that all the software requirements are installed as listed above and in the mmWave SDK release notes.

Note

As mentioned in the Quickstart section, pre-built binary files, both debug and deployment binaries are provided in the pre-compiled directory of the lab.

4. Execute the Lab

There are two ways to execute the compiled code on the EVM:

- Deployment mode: the EVM boots autonomously from flash and starts running the bin image
    - Using Uniflash, flash the **long_range_people_det_68xx_demo.bin** found at
      `<PROJECT_WORKSPACE_DIR>\long_range_people_det_68xx_mss\Debug\long_range_people_det_68xx_demo.bin`
        - The same procedure for flashing can be use as detailed in the Quickstart Flash the Device section.
- Debug mode: Follow the instructions for Using CCS Debug for Development (../../../common/docs/software_setup/using_ccs_debug.html)

After executing the lab using either method, the lab can be visualized using the Quick Start GUI or continue to working with the GUI Source Code

Visualizer Source Code

Working with and running the Visualizer source files requires a MATLAB License not just the MATLAB Runtime Engine

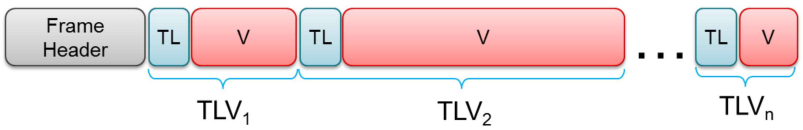
The detection processing chain and group tracking algorithm are implemented in the firmware. The visualizer serves to read the UART stream from the device and then plot the detected points and tracked objects.

Source files are located at `C:\ti\mmwave_industrial_toolbox_<VER>\labs\long_range_people_detection\68xx_long_range_people_det\gui` .

- **main_pplcount_viz.m**: the main program which reads and parses the UART data for visualization
- **setup.m**, **setup.fig**: creates the visualizer configuration window used in GUI setup mode where user can input setup parameters

Data Formats

A TLV(type-length-value) encoding scheme is used with little endian byte order. For every frame, a packet is sent consisting of a fixed sized **Frame Header** and then a variable number of TLVs depending on what was detected in that scene. The TLVs can be of types representing the point cloud, target list object, and associated points.



Frame Header

Size: 52 bytes

```
frameHeaderStructType = struct(...
    'magicWord',               {'uint64', 8}, ... % syncPattern in hex is: '02 01 04 03 06 05 08 07'
    'version',                 {'uint32', 4}, ... % Software Version
    'platform',                {'uint32', 4}, ... % A6843
    'timeStamp',               {'uint32', 4}, ... % Message create time in cycles
    'totalPacketLen',          {'uint32', 4}, ... % In bytes, including header
    'frameNumber',             {'uint32', 4}, ... % Frame Number
    'subFrameNumber',          {'uint32', 4}, ... % Sub-Frame number
    'chirpProcessingMargin',   {'uint32', 4}, ... % time left after chirp processing in cycles
    'frameProcessingMargin',   {'uint32', 4}, ... % time left after frame processing in cycles
    'trackingProcessingTime',  {'uint32', 4}, ... % time to run tracker
    'uartSendingTime',         {'uint32', 4}, ... % time to send uart message
    'numTLVs' ,                {'uint16', 2}, ... % Number of TLVs in this frame
    'checksum',                {'uint16', 2});    % Subframe number. In this lab, always set to 0.
```

Frame Header Structure in MATLAB syntax for name, type, length

TLVs

The TLVs can be of type **DPIF_PointCloudSpherical**, **DPIF_PointCloudSideInfo**, **TARGET_LIST_3D**, or **TARGET_INDEX**.

TLV Header

Size: 8 bytes

```
% TLV Type: 06 = DPIF Point cloud spherical, 07 = Target object list, 08 = Target index, 09 = DPIF Point Cloud Side I
nfo
tlvHeaderStruct = struct(...
    'type',            {'uint32', 4}, ... % TLV object
    'length',          {'uint32', 4});    % TLV object Length, in bytes, including TLV header
```

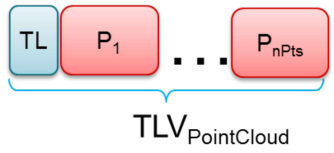
TLV header in MATLAB syntax

Following the header, is the the TLV-type specific payload

Point Cloud TLV

Type: DPIF_POINT_CLOUD_SPHERICAL

Size: sizeof (DPIF_PointCloudSpherical) x numberOfPoints



Each Point Cloud TLV consists of an array of points. Each point is defined in 16 bytes.

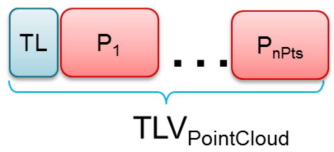
```
DPIF_PointCloudSpherical = struct(...
    'range',           {'float', 4}, ... % Range, in m
    'azimuth',         {'float', 4}, ... % Azimuth angle, in rad
    'elevation',       {'float', 4}, ... % Elevation angle, in rad
    'doppler'          {'float', 4}, ... % Doppler, in m/s
```

Point Structure in MATLAB syntax

Point Cloud Side Info TLV

Type: DPIF_POINT_CLOUD_SIDE_INFO

Size: sizeof(DPIF_PointCloudSideInfo) x numberOfPoints



Each Point Cloud Side Info TLV consists of an array of point side info data. Each is 8 bytes.

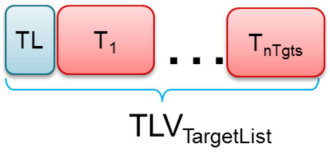
```
DPIF_PointCloudSideInfo = struct(...
    'snr',              {'int16_t', 2}, ... % SNR, ratio
    'noise'             {'int16_t', 2}, ... % Noise
```

Point Side Info Structure in MATLAB syntax

Target Object TLV

Type: TARGET_LIST_3D

Size: sizeof (targetStruct3D) x numberOfTargets



Each Target List TLV consists of an array of targets. Each target is defined in 68 bytes.

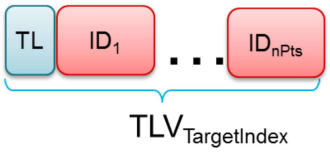
```
targetStruct3D = struct(...
    'tid',              {'uint32', 4}, ... % Track ID
    'posX',             {'float', 4}, ... % Target position in X dimension, m
    'posY',             {'float', 4}, ... % Target position in Y dimension, m
    'velX',             {'float', 4}, ... % Target velocity in X dimension, m/s
    'velY',             {'float', 4}, ... % Target velocity in Y dimension, m/s
    'accX',             {'float', 4}, ... % Target acceleration in X dimension, m/s2
    'accY',             {'float', 4}, ... % Target acceleration in Y dimension, m/s
    'posZ',             {'float', 4}, ... % Target velocity in Y dimension, m/s
    'velZ',             {'float', 4}, ... % Target acceleration in X dimension, m/s2
    'accZ',             {'float', 4}, ... % Target acceleration in Y dimension, m/s
```

Target Structure in MATLAB syntax

Target Index TLV

Type: TARGET_INDEX

Size: numberOfPoints



Each Target List TLV consists of an array of target IDs. A targetID at index ***i*** is the target to which point ***i*** of the frame's point cloud was associated. Valid IDs range from 0-249.

```
targetIndex = struct(...
'targetID',         {'uint8', 1});    % Track ID
```

Target ID Structure in MATLAB syntax

Other Target ID values:

Value	Meaning
253	Point not associated, SNR too weak
254	Point not associated, located outside boundary of interest
255	Point not associated, considered as noise

Customization

- Please refer to the **People Counting Demo Customization Guide** which can be found at `C:\ti\<mmwave_industrial_toolbox_install_dir>\labs\long_range_people_detection\68xx_long_range_people_det\docs\pplcount_customization_guide.pc`

Need More Help?

- Find answers to common questions on mmWave E2E FAQ (https://e2e.ti.com/support/sensor/mmwave_sensors/w/wiki)
- Search for your issue or post a new question on the mmWave E2E forum (https://e2e.ti.com/support/sensor/mmwave_sensors/f/1023)