Design Guide: TIDEP-0090

Traffic Monitoring Object Detection and Tracking Reference Design Using Single-Chip mmWave Radar Sensor

Description

The TIDEP-0090 demonstrates how TI's single-chip millimeter-wave (mmWave) technology can be used for robust, long-range sensing in traffic monitoring and other applications. The reference design uses the IWR1642BOOST evaluation module (EVM) and integrates a complete radar processing chain onto the IWR1642 device.Additionally, the IWR1843 and IWR6843 devices are also supported, giving the option for 3D object detection. Enabling 3D detection can improve robustness and accuracy of tracking with the ability to filter detected points based on elevation information. The processing chain includes the analog radar configuration, analog-to-digital converter (ADC) capture, low-level FFT processing, and high-level clustering and tracking algorithms. This reference design is intended to be built on top of the TI mmWave SDK for a cohesive software experience including APIs, libraries, and tools for evaluation, development, and data visualization.

Resources

TIDEP-0090	Design Folder
IWR1642	Product Folder
IWR1642BOOST	Tool Folder
IWR6843ISK	Tool Folder
IWR1843BOOST	Tool Folder
mmWave SDK	Tool Folder

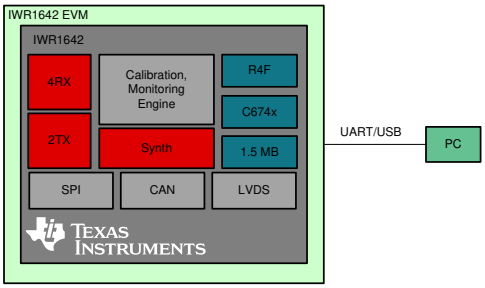
ASK Our E2E Experts

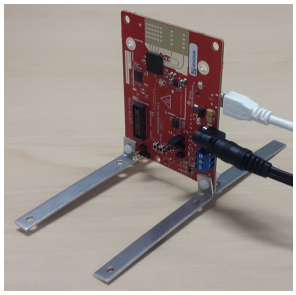
Features

- Demonstration of environmentally robust object detection, clustering, and tracking using TI single-chip mmWave sensor
- mmWave sensor to pinpoint location of objects in 100° field of view over a tange of 5 m to 180 m
- Measurement bandwidth of up to 4 GHz
- Onboard digital signal processor (C674x) to detect objects and ARM (R4F) to track vehicle's range and velocity over time
  - Device output is object data with range, velocity, angle, and track information
- Based on proven EVM hardware designs enabling quick time to market and out-of-the-box demonstration

Applications

- Traffic monitoring
- Road-railway sensors
- Intelligent lighting
- Building security – occupancy detection
- Surveillance – IP network camera





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

The TIDEP-0090 provides a reference for creating a traffic monitoring application using TI's mmWave radio-frequency complementary metal-oxide semiconductor (RF-CMOS) technology. The reference design provides test results and examples using the IWR1642BOOST EVM. The design has been further extended to enable traffic monitoring applications on IWR6843 and IWR1843 devices with the additional capability of 3D detection and tracking.

mmWave sensing technology detects vehicles, such as cars, motorcycles, and bicycles, at extended ranges regardless of environmental conditions, such as rain, fog, or dust. TI's mmWave sensing devices integrate a 4 GHz bandwidth mmWave radar front end with ARM® microcontroller (MCU) and TI DSP cores for single-chip systems.

This traffic monitoring reference design has several design goals to demonstrate the suitability of the mmWave sensors for traffic monitoring applications. This design targets the implementation of a wide, azimuth field of view (±50°), intermediate range (70 m) sensor configuration, which can detect small cars across four lanes and track their position and velocity as they approach the intersection and the stop bar.

This design guide implements algorithms for radar signal processing, detection, and tracking for an IWR1642 device on a TI EVM module. The design provides a list of required hardware, schematics, and foundational software to quickly begin traffic monitoring product development.

This reference design describes the example use case as well as the design principle, implementation details, and engineering tradeoffs made in the development of this application. High-level instructions for replicating the design are provided.

2 System Overview

2.1 Block Diagram

2.1.1 Hardware Block Diagram

The TIDEP-0090 is implemented on IWR1642BOOST EVM as an example of traffic monitoring capabilities using mmWave sensor. For details on enabling traffic monitoring applications on either the IWR6843 or IWR1843 devices refer to: Section 5.2 . The design considerations and enabled features in this document apply across all EVM platforms. The EVM is connected to a host PC through universal asynchronous receiver-transmitter (UART) for visualization.

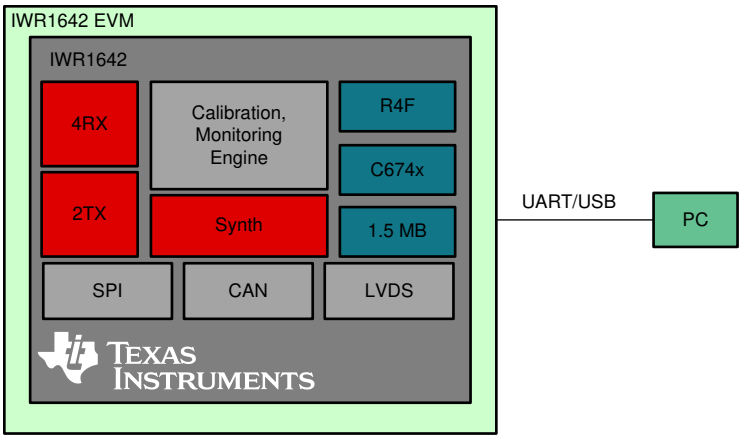


Figure 1. Hardware Block Diagram

2.1.2 Software Block Diagram

2.1.2.1 mmWave SDK Software Block Diagram

The mmWave software development kit (SDK) enables the development of mmWave sensor applications using IWR1443 SOC and EVM and IWR1642 SOC and EVM. The SDK provides foundational components that will facilitate end users to focus on their applications. In addition, the SDK provides several demonstration applications, which will serve as a guide for integrating the SDK into end-user mmWave applications. This design guide is a separate package installed on top of the SDK package. In the case of the traffic monitoring reference design, the yellow blocks of customer code are replaced by traffic monitoring application code.

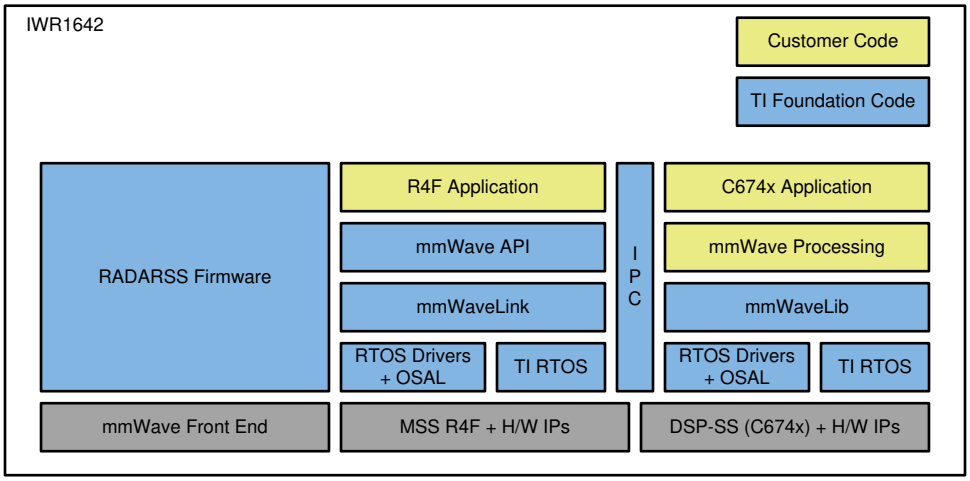


Figure 2. Generic mmWave SDK Software Block Diagram

2.1.2.2 Traffic Monitoring Application Software

As described in Figure 3, the implementation of the traffic-monitoring example consists of radar processing layers implemented on the IWR1642 target, with scene interpretation and visualization done at PC Host in MATLAB® environment.

There are three Radar processing layers, implemented at the target:

- Front End processing
- Low Level Processing
- High Level Processing

2.1.2.3 Front End Processing

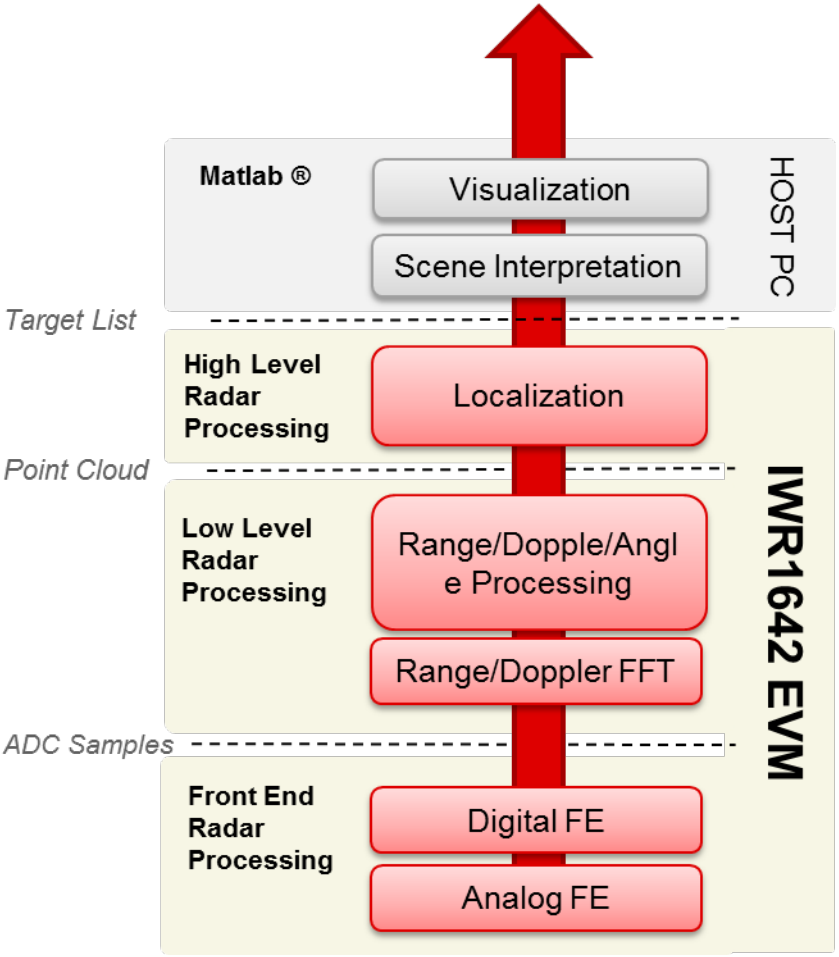


Figure 3. Traffic Monitoring Software Layers

Front end processing is implemented in HW and FW in RADAR subsystem (RADARSS). Front end is configured by an application using mmWave library, provided by mmWaveSDK platform SW.

2.1.2.4 Low Level Processing

As described in Figure 4, the implementation of the traffic-monitoring example in the signal-processing chain consists of the following blocks implemented as DSP code executing on the C674x core in the IWR1642:

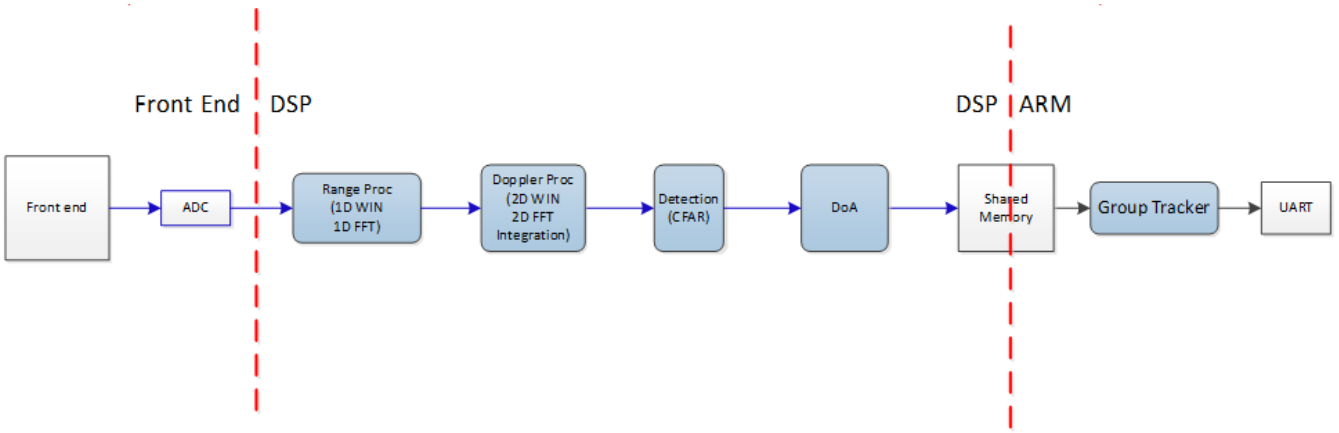


Figure 4. Low Level Radar Processing

- Range processing
  - For each antenna, 1D windowing, and 1D fast Fourier transform (FFT)
  - Range processing is interleaved with the active chirp time of the frame
- Doppler processing
  - For each antenna, 2D windowing, and 2D FFT
  - Then non-coherent combining of received power across antennas in floating-point precision
- Range-Doppler detection algorithm
  - Constant false-alarm rate, cell averaging smallest of (CASO-CFAR) detection in range domain, plus CFAR-cell averaging (CACFAR) in Doppler domain detection, run on the range-Doppler power mapping to find detection points in range and Doppler space.
- Angle estimation
  - For each detected point in range and Doppler space, reconstruct the 2D FFT output with Doppler compensation, then a beamforming algorithm returns one angle based on the angle correction for Vmax extension.

After the DSP finishes frame processing, the results consisting of [range, Doppler, angle, detection SNR] are formatted and written in shared memory (L3RAM) for R4F to perform high level processing. R4F then sends all the results to the host through a UART for visualization.

2.1.2.5 High Level Radar Processing

With advances in radar detection precision, the real world radar targets (cars, pedestrians, walls, road elements, etc.) are represented by a set of multiple reflection points with adjacent range, angular, and Doppler measurements. Due to fluctuating nature of those reflections, we found it rather impractical to track the individual points. Instead, we implemented a tracker, which models the behavior of the set of those points (a group), which is persistent from frame to frame, and represents target size and motion characteristics very well.

As described in Figure 5, high level processing is implemented in the ARM R4F core in the IWR1642. Input from the low-level processing layer (point cloud data) is copied from the shared memory and adapted to tracker interface.

Group Tracker is implemented with two sub-layers: one instance module managing multiple units. At the module layer, we first attempt to associate each point from the input cloud with a tracking unit. Non-associated points will undergo allocation procedure. At the unit level, each track uses Extended Kalman Filter (EKF) process to predict and estimate the properties of the group.

Transport block delivers point cloud, target list, and index array over serial interface. See section 3.5 UART Communications for interface definitions.

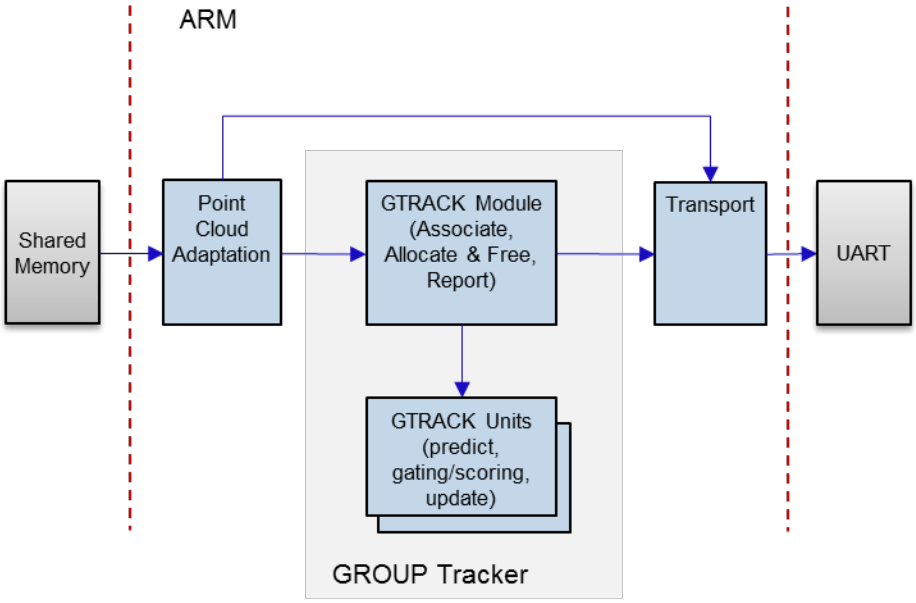


Figure 5. High Level Radar Processing

2.2 Highlighted Products

2.2.1 IWR1642

The IWR1642 is an integrated, single-chip, frequency modulated continuous wave (FMCW) sensor capable of operation in the 76- to 81-GHz band. The sensor is built with TI's low-power, 45-nm RFCMOS process and enables unprecedented levels of integration in an extremely small form factor. The IWR1642 is an ideal solution for low-power, self-monitored, ultra-accurate radar systems in the industrial space.

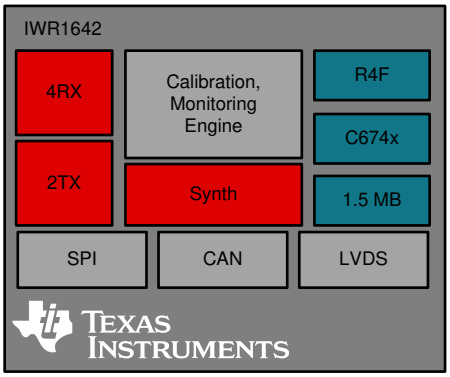


Figure 6. IWR1642 Block Diagram

IWR1642 has the following features:

- FMCW transceiver
  - Integrated PLL, transmitter, receiver, baseband, and A2D
  - 76- to 81-GHz coverage with 4-GHz available bandwidth
  - Four receive channels
  - Two transmit channels
  - Ultra-accurate chirp (timing) engine based on fractional-N PLL
  - TX power
    - 12 dBm
  - RX noise figure
    - 15 dB (76 to 77 GHz)
    - 16 dB (77 to 81 GHz)
  - Phase noise at 1 MHz
    - –94 dBc/Hz (76 to 77 GHz)
    - –91 dBc/Hz (77 to 81 GHz)
- Built-in Calibration and Self-Test (Monitoring)
  - ARM Cortex®-R4F-based radio control system
  - Built-in firmware (ROM)
  - Self-calibrating system across frequency and temperature
- C674x DSP for FMCW signal processing
  - On-chip memory: 1.5MB
- Cortex-R4F MCU for object detection, and interface control
  - Supports autonomous mode (loading user application from QSPI flash memory)
- Integrated peripherals
  - Internal memories with ECC
  - Up to six ADC Channels
  - Up to two SPI Channels

- Up to two UARTs
- CAN interface
- I2C
- GPIOs
- Two-lane LVDS interface for raw ADC data and debug instrumentation

2.2.2 mmWave SDK

The mmWave SDK is split in two broad components: mmWave Suite and mmWave Demos.

2.2.2.1 mmWave Suite

mmWave Suite is the foundational software part of the mmWave SDK and includes smaller components:

- Drivers
- OSAL
- mmWaveLink
- mmWaveLib
- mmWave API
- BSS firmware
- Board setup and flash utilities

2.2.2.1.1 mmWave Demos

The SDK provides a suite of demonstrations that depict the various control and data processing aspects of an mmWave application. Data visualization of the demonstration's output on a PC is provided as part of these demonstrations:

- mmWave processing demonstration

3 Design Considerations

3.1 Use Case Geometry and Sensor Considerations

The IWR1642 is a radar-based sensor that integrates a fast FMCW radar front end with both an integrated ARM R4F MCU and TI C674x DSP for advanced signal processing.

The configuration of the IWR1642 radar front end depends on the configuration of the transmit signal as well as the configuration and performance of the RF transceiver, the design of the antenna array, and the available memory and processing power. This configuration influences key performance parameters of the system.

The key performance parameters at issue are listed with brief descriptions:

- Maximum range
  - Range is estimated from a beat frequency in the de-chirped signal that is proportional to the round trip delay to the target. For a given chirp ramp slope, the maximum theoretical range is determined by the maximum beat frequency that can be detected in the RF transceiver. The maximum practical range is then determined by the SNR of the received signal and the SNR threshold of the detector.
- Range resolution
  - This is defined as the minimum range difference over which the detector can distinguish two individual point targets, which is determined by the bandwidth of the chirp frequency sweep. The higher the chirp bandwidth is, the finer the range resolution will be.
- Range accuracy
  - This is often defined as a rule of thumb formula for the variance of the range estimation of a single point target as a function of the SNR.
- Maximum velocity

  - Radial velocity is directly measured in the low-level processing chain as a phase shift of the de-chirped signal across chirps within one frame. The maximum unambiguous velocity observable is then determined by the chirp repetition time within one frame. Typically this velocity is adjusted to be one-half to one-fourth of the desired velocity range to have better tradeoffs relative to the other parameters. Other processing techniques are then used to remove ambiguity in the velocity measurements, which will experience aliasing.
- Velocity resolution
  - This is defined as the minimum velocity difference over which the detector can distinguish two individual point targets that also happen to be at the same range. This is determined by the total chirping time within one frame. The longer the chirping time, the finer the velocity resolution.
- Velocity accuracy
  - This is often defined as a rule of thumb formula for the variance of the velocity estimation of a single-point target as a function of the SNR.
- Field of view
  - This is the sweep of angles over which the radar transceiver can effectively detect targets. This is a function of the combined antenna gain of the transmit and receive antenna arrays as a function of angle and can also be affected by the type of transmit or receive processing, which may affect the effective antenna gain as a function of angle. The field of view is typically specified separately for the azimuth and elevation.
- Angular resolution
  - This is defined as the minimum angular difference over which the detector can distinguish two individual point targets that also happened to have the same range and velocity. This is determined by the number and geometry of the antennas in the transmit and receive antenna arrays. This is typically specified separately for the azimuth and elevation.
- Angular accuracy
  - This is often defined as a rule of thumb formula for the variance of the angle estimation of a single point target as a function of SNR.

When designing the frame and chirp configuration for a traffic-monitoring use case, start by considering the geometry of the scene, the field of view in both azimuth and elevation, and the ranges of interest. As an example, assume a four-lane intersection with a radar sensor mounted overhead. Taking some assumptions on the sizes and positioning of lanes, medians, crosswalks, stop bars, and overhead sensor mountings (see Figure 7), an azimuth field of view of at least 25° will cover the stop bar and approaching 60 m of roadway.

Figure 7 shows an example traffic monitoring geometry.

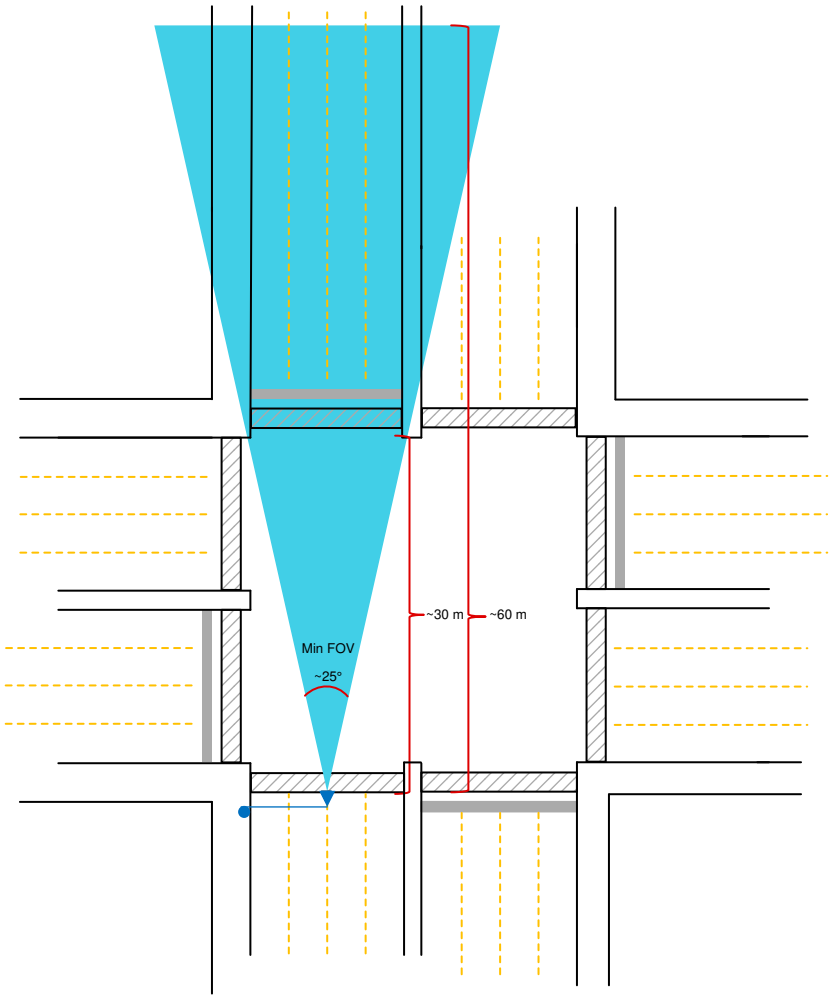


Figure 7. Sample Intersection

For this example assume that the antenna pattern enables this azimuth field of view with two transmit and four receive antennas for azimuth angle estimation, while in the elevation axis the field of view is a narrow 15° without elevation angle processing. Figure 8 shows the example elevation geometry, which shows a traffic-monitoring sensor mounted at a height of 7.5 m with a 15° elevation field of view and 7.5° downtilt.

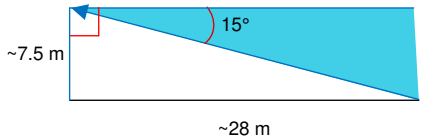


Figure 8. Sample Traffic Monitoring Sensor Mount

The IWR1642 EVM has a much wider azimuth field of view of 100° and a somewhat wider elevation field of view of 22°, but the EVM has sufficient antenna gain to achieve a 60-m range for vehicle detection, so the EVM will be used as a basis for a medium-range example chirp configuration.

The configurability of the IWR1642 allows for flexibility of design to fit different use-cases within traffic monitoring. After fixing the basic geometry of the intersection and the antenna pattern, the chirp design is carried out considering some target performance parameters and balancing the trade-offs among those parameters in the context of the IWR1642's device transceiver capabilities. In particular, consider the maximum range as a starting point. Two examples are outlined. One is for a medium range of 70 m and includes transmit multiple-input-multiple-output (MIMO) processing to increase the angular resolution. The other is a longer-range, 120-m design without the MIMO processing. In the following paragraphs a brief description of the tradeoff analysis is provided.

Given the IWR1642 maximum (complex) sampling rate of 6.25 Msps, the requirement for the maximum range determines the maximum beat frequency, which has to pass through the available IF chain of the radar sensor bandwidth. Beat frequency also depends on the chirp duration and the bandwidth of the chirp signal, which depend on the requirements for the range resolution and the maximum velocity, respectively. The finer the required range resolution, the longer the chirp duration must be. Therefore from the perspective of the maximum beat frequency (which is limited by the IF chain design), there exists a tradeoff between the requirements for the maximum range and the range resolution. Other things equal, the larger the maximum range, the coarser the range resolution must be.

The maximum velocity determines minimum chirp repetition period and indirectly the chirp duration. The higher maximum velocity, the shorter chirp duration and, in turn, the shorter chirp duration the higher beat frequency. Given the maximum range and maximum beat frequency (determined by the IF chain design), the requirements for the maximum velocity and the range resolution are the trade-off parameters. Other things equal, the higher maximum velocity the coarser range resolution and vice versa.

The requirements for the velocity resolutions for the traffic monitoring use case would normally be of the order of 1 km/h or coarser. The velocity resolution is inversely proportional to the number of chirp repetitions in a frame. However, the number of chirps in a frame also affects the SNR at the detector input—the more chirps in a frame, the higher SNR at the detector input. Therefore, although it may not be required to have very fine velocity resolution for the traffic monitoring system, the velocity resolution may need to be set finer such that a target of a desired RCS is detectable at the maximum range.

The overall performance of the radar traffic monitoring system depends also on transmit and receive antenna system design. In the case of the considered IWR1642 sensor, there are two transmit antennas separated by 2λ and four receive antennas separated by $\lambda/2$. This enables several modes of operations, two of them are used here:

1. Using only one transmit antenna and four receive antennas allows for the receive beamforming across four receive signal chains. In this case when using a simple classical receive beamforming, angular (azimuth) resolution is approximately given by $\Delta\theta = 2/(\cos(\theta)*Nr)$ where Nr is the number of receive antennas and where theta is the overall beam steering angle away from normal to the array. Given Nr = 4 here, $\Delta\theta_a = 30°$ for the range of azimuth angles of interest.
2. Using two transmit antennas (in an alternating, TDM fashion) and four receive antennas enables the TDM MIMO mode, which allows for the receive beamforming across eight (virtual) receive signal chains, Nr = 8. Given this, in the case of TDM MIMO: $\Delta\theta_b = 14.3°$.

There are obviously trade-offs associated with the selection of the two modes of operation above. TDM MIMO mode enables better angular resolution at the cost of reduced maximum velocity.

In both example cases, the maximum sampling frequency is selected in order to take advantage of the full IF bandwidth. The target maximum range is set. After setting the maximum range, the range resolution and maximum velocity are balanced in a trade-off to achieve the best range resolution while meeting the maximum velocity requirements. Increasing the velocity resolution to the practical limit of the internal radar cube memory also increases the effective range of the transceiver.

For more information regarding chirp parameters and MIMO mode, refer to the following resources:

- Radar Devices
- MIMO Radar

The example chirp design starts with the input parameters shown in Table 1. Note that the approach outlined in this section for the chirp and frame design is not the only approach that can be taken. Other approaches could point to other alternative designs with a different balance of the trade-offs.

Table 1. Performance Parameters of Two Example Chirp Designs on the IWR1642

KEY INPUT PARAMETERS		
PERFORMANCE PARAMETERS	MEDIUM-RANGE MIMO EXAMPLE	LONG-RANGE NON-MIMO EXAMPLE
Antenna pattern	Two Tx, four Rx in azimuth plane	One Tx, four Rx in azimuth plane
Maximum range	70 m	185 m
Range resolution	0.25 m	0.8 m
Maximum velocity(1)	27 km/h (7.5 m/s)(1)	65 km/h (18 m/s)(1)
Velocity resolution	1.7 km/h (o.47 m/s)	1.1 km/h (0.30 m/s)
Frame duration	50 ms	50 ms
ADC sampling rate	5.5 MSPS	5.5 MSPS
DERIVED CHIRP DESIGN PARAMETERS		
Chirp valid sweep bandwidth	600 MHz	186 MHz
Chirp time	56.64 µs	46.6 µs
Chirp repetition time	129.7 µs	54.6 µs
Number of samples per chirp	312	256
Nfft_range	512	256
Number of chirps per frame	32	118
Nfft_doppler	32	128
Radar cube size	512 KB	480 KB

(1) Additional processing can extend the maximum trackable velocity by 3x the chirp maximum velocity.

3.2 Low Level Processing Details

An example processing chain for traffic monitoring using the medium range chirp and frame design is implemented on the IWR1642 EVM.

The main processing elements involved in the processing chain consist of the following:

- Front end
  - Represents the antennas and the analog RF transceiver implementing the FMCW transmitter and receiver and various harware-based signal conditioning operations. This must be properly configured for the chirp and frame settings of the use case.
- ADC
  - The ADC is the main element that interfaces to the DSP chain. The ADC output samples are buffered in ADC output buffers for access by the digital part of the processing chain.
- EDMA controller
  - This is a user-programed, DMA engine employed to move data from one memory location to another without using another processor. The EDMA can be programed to trigger automatically and can also be configured to reorder some of the data during the movement operations.
- C674 DSP
  - This is the digital signal processing core that implements the configuration of the front end and executes the low level signal processing operations on the data. This core has access to several memory resources as noted further in the design description.

The Low Level Processing chain is implemented on the DSP. There are several physical memory resources used in the processing chain, which are described in Table 2.

Table 2. Memory Configuration

SECTION NAME	SIZE (KB) AS CONFIGURED	MEMORY USED (KB)	DESCRIPTION
L1D SRAM	16	16	Layer one data static RAM is the fastest data access for DSP and is used for most time-critical DSP processing data that can fit in this section.
L1D cache	16	Used as cache	Layer one data cache caches data accesses to any other section configured as cacheable. The LL2, L3, and HSRAM are configured as cacheable.
L1P SRAM	28	22	Layer one program static RAM is the fastest program access RAM for DSP and is used for most time-critical DSP program that can fit in this section.
L1P cache	4	Used as cache	Layer one cache caches program accesses to any other section configured as cacheble. The LL2, L3, and HSRAM are configured as cacheable.
LL2	256	227.3	Local layer two memory is lower latency than layer three for accessing and is visible only from the DSP. This memory is used for most of the program and data for the signal processing chain.
L3	768	580	Higher latency memory for DSP accesses primarily stores the radar cube and the range-Doppler power map. It is a less time-sensitive program. Data can also be stored here.
HSRAM	32	Currently unused	Shared memory buffer between the DSP and the R4F relays visualization data to the R4F for output over the UART in this design.

As described in Figure 9, the implementation of the traffic monitoring example in the signal-processing chain consists of the following blocks implemented as DSP code executing on the C674x core in the IWR1642:

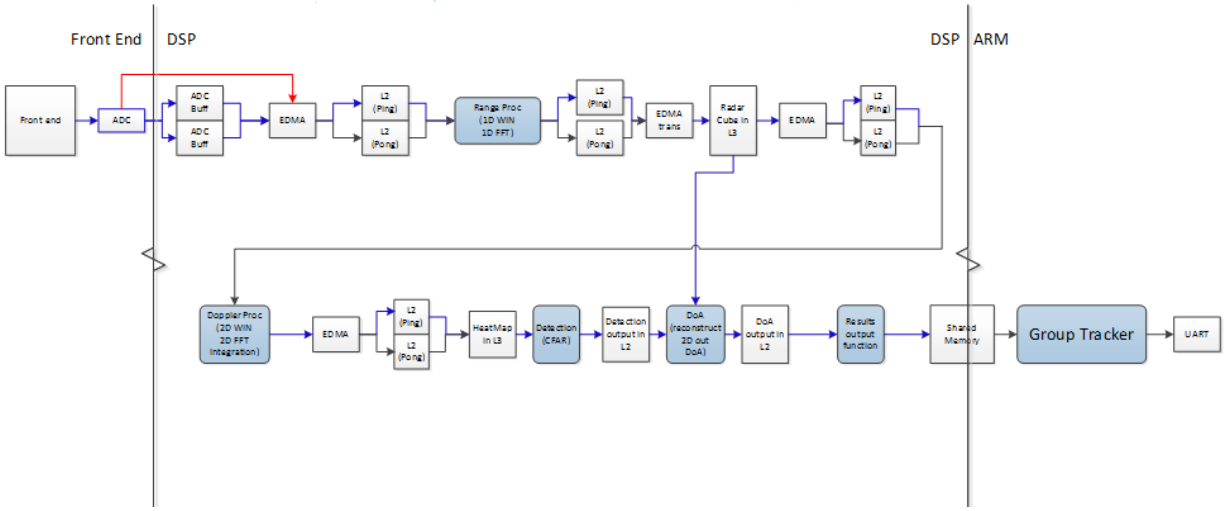


Figure 9. Low Level Signal Processing Chain

- Range processing
  - For each antenna, EDMA is used to move samples from the ADC output buffer to DSP's local memory. A 16-bit, fixed-point 1D windowing and 16-bit, fixed-point 1D FFT are performed. EDMA is used to move output from DSP local memory to radar cube storage in layer three (L3) memory. Range processing is interleaved with active chirp time of the frame. All other processing happens each frame, except where noted, during the idle time between the active chirp time and the end of the frame.
- Doppler processing, antenna combining
  - For each antenna, EDMA transfers data between radar cube in L3 and DSP local L2 memory. The DSP operations are, 16-bit fixed point 2D windowing, formatting from 16-bit fixed-point IQ to floating-point IQ, floating-point 2D FFT, and non-coherent combining of received power across antennas in floating point. The output range-Doppler power signal or heat map is stored in L3 memory separate from the radar cube. Note that per antenna, floating-point Doppler data is discarded to reduce memory storage.
- Range-Doppler detection
  - An algorithm is applied to the range-Doppler power mapping to find detection points in range and Doppler space. The algorithm consists of a first pass along range axis using cell averaging smaller of (CASO) CFAR, and a second pass along Doppler axis using cell averaging (CA) CFAR. Due to the data access pattern, the detection code accesses the integrated signal in L3 memory through the L1D cache. The output detected point list is stored in L2 memory.
- Angle estimation
  - For each detected point in range and Doppler space, the input to angle estimation is reconstructed by re-computing per antenna Doppler data from radar cube and applying Doppler compensation for 2-TX antenna TDM-MIMO.
  - In the case of TDM-MIMO with velocity ambiguity, additional processing is needed for Vmax extension. This is subsequently referred as angle correction for Vmax extension. 2 hypothesis of Doppler compensation are applied, one corresponding to speed of (2*N-1)*v, and one corresponding to speed of 2*N*v, for 2-TX antenna case. A beamforming search algorithm then will be performed on both hypothesis, and whichever returns the larger peak will be declared as the

correct angle estimates for this [range, Doppler] pair. The output is stored in the L2 memory, and then copied to the shared memory between DSP and ARM, along with range/Doppler estimation, as well as the detection SNR.

– The angle correction for Vmax extension is very sensitive to the antenna phase correctness, especially the Tx antenna phase. The Tx antenna phase non-ideality will introduce similar angle bias effect as that from Doppler effect. If Tx antenna phase error is not compensated, the angle correction for Vmax extension will result in a wrong angle which will show up as a strong biased ghost that will cause tracking error. This phase non-ideality needs to be compensated with a calibration procedure described in the "Range Bias and Rx Channel Gain/Offset Measurement and Compensation" section of the mmWave SDK documentation for the xwr16xx Out of Box Demo.

3.3 High Level Processing Details

3.3.1 Task Model

High level processing is implemented with two tasks: higher priority Mailbox Task, and lower priority Application Task. Once the system is configured, Mailbox task is pending on a semaphore, waiting for the frame ready message from DSP. Once awakened, Mailbox task copies relevant point cloud data from the shared memory into TCM, and posts the semaphore to an Application Task to run. It then creates the transport frame header, and initiates a DMA process for each part (TLV) of the frame. While DMA started sending data over UART, Mailbox task yields to lower priority Application Task. Once DMA process completes, additional DMA can be scheduled (example, TX2, and TX3). To achieve parallelism between Task processing and DMA, Transmit task sends current (Nth) point cloud TLV with previous (N-1)th target list and target index TLVs.

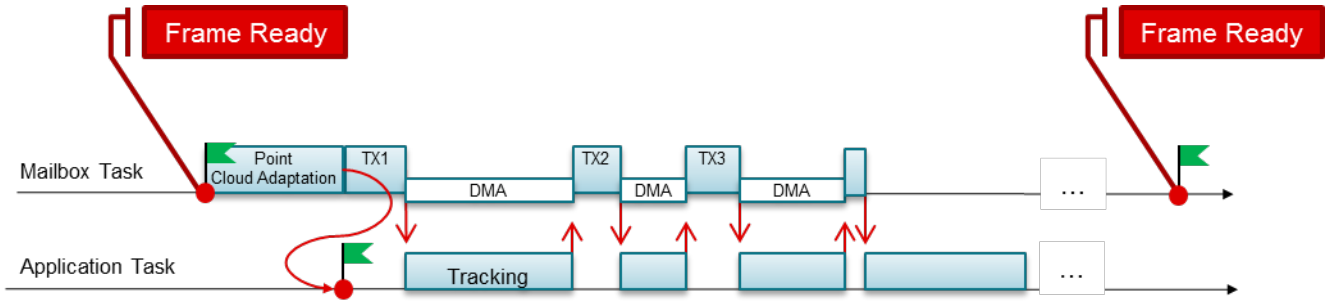


Figure 10. High Level Processing Task Model

Application Task executes Group Tracking Step.

3.3.2 Group Tracker

Tracking algorithm is implemented as a library. Application task creates an algorithm instance with configuration parameters that describe sensor, scenery, and behavior of radar targets. Algorithm is called once per frame from Application Task context. It is possible to create multiple instances of group tracker.

Figure 11 explains the steps the algorithm goes through during each frame call. Algorithm inputs measurement data in Polar coordinates (range, angle, Doppler), and tracks objects in Cartesian space. Therefore we use Extended Kalman Filter (EKF) process.

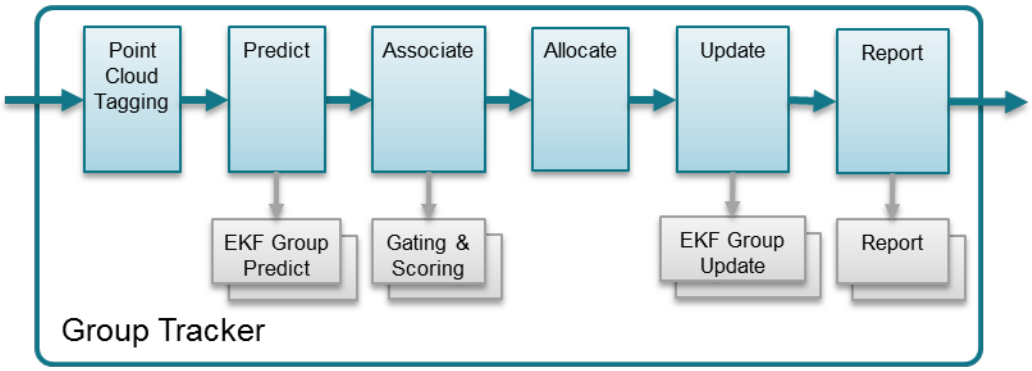


Figure 11. Group Tracking Algorithm

Point cloud input is first tagged based on scene boundaries. Some points may be tagged as “outside the boundaries”, and will be ignored in association and allocation processes.

Predict function estimates tracking group centroid for time n based on state and process covariance matrices estimated at time n-1. We compute a-priori state and error covariance estimations for each trackable object. At this step we also compute measurement vector estimations.

Association function allows each tracking unit to indicate whether each measurement point is “close enough” (gating), and if it is, to provide the bidding value (scoring). Point is assigned to a highest bidder.

Un-assigned points are going through an Allocate function. During the Allocation process, points are first joined into a sets based on their proximity in measurement coordinates. Each set becomes a candidate for allocation decision. It has to pass multiple tests to become a new track. Once passed, the new tracking unit is allocated.

During Update step, tracks are updated based on the set of associated points. We compute the innovation, Kalman gain, and a-posteriori state vector and error covariance. In addition to classic EKF, the error covariance calculation includes group dispersion in measurement noise covariance matrix.

The Report function queries each tracking unit and produces the algorithm output.

3.3.3 Configuration Parameters

The configuration parameters are used to configure Tracking algorithm. They shall be adjusted to match the user's use case based on particular scenery and targets characteristics. Parameters are divided into mandatory, and optional (advanced). Mandatory parameters are described below.

Table 3. Mandatory Configuration Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
maxNumPoints	250		Maximum Number of Detection Points per frame
maxNumTracks	20		Maximum Number of Targets to track at any given time
stateTrackingVector Type	2DA		2D={x,y,vx,vy} 2DA={x,y,vx,vy,ax,ay} 2DA is the only supported option 3D={x,y,z,vx,vy,vz} 3DA={x,y,z,vx,vy,vz,ax,ay,az}
initialRadialVelocity	-5	m/s	Expected target radial velocity at the moment of detection
maxRadialVelocity	N/A	m/s	Maximum absolute radial velocity reported by sensor. Shall match sensor chirp configuration
radialVelocityResolu tion	N/A	m/s	Minimal non-zero radial velocity reported by sensor. Shall match sensor chirp configuration
maxAccelerationX	0	m/s^2	Maximum targets acceleration in lateral direction. Used to compute processing noise matrix
maxAccelerationY	4	m/s^2	Maximum targets acceleration in latitudinal direction. Used to compute processing noise matrix
deltaT	N/A	ms	Frame Rate, shall match sensor chirp configuration
verbosityLevel	NONE		A bit mask representing levels of verbosity: NONE | WARNING | DEBUG | ASSOCIATION DEBUG | GATE_DEBUG | MATRIX DEBUG

First two parameters (maxNum Points, maxNumTracks) are used to dimension the tracking SW.

3.3.3.1 Advanced Parameters

Advanced parameters are divided into few sets. Each set can be omitted, and defaults will be used by an algorithm. The user is expected to modify needed parameters to achieve better performance.

3.3.3.1.1 Scenery Parameters

This set of parameters describes the scenery. It allows user to configure the tracker with expected boundaries, and areas of static behavior. User can define up to 2 boundary boxes, and up to 2 static boxes. Boxes coordinates are in meters, sensor is assumed at (0, 0) of Cartesian (X, Y) space. Table 4 lists these parameters.

Table 4. Scenery Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
numBoundaryBoxes	1U	---	Number of boundary boxes defined. Points outside boundary box will be ignored
boundaryBox[2]	{0.7f, 15.5f,15.f,75.f}, {0, 0, 0, 0}	m	(left, right, bottom, top}
numStaticBoxes	1U	---	Number of static boxes defined. Targets inside static box are allowed to persist as static
upperEntranstaticBox[2]	{1.7f, 14.5.f,16.f,50.f}, {0, 0, 0, 0}	m	(left, right, bottom, top}

3.3.3.1.2 Measurement Standard Deviation Parameters

This set of parameters is used to estimate standard deviation of the reflection point measurements. Table 5 lists these parameters.

Table 5. Measurements Standard Deviation Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
LengthStd	4/3.46	m	Expected standard deviation of measurements in target length dimension
WidthStd	1.5/3.46	m	Expected standard deviation of measurements in target width dimension
DopplerStd	1.0f	m/s	Expected standard deviation of measurements of target radial velocity

Typically, the uniform distribution of reflection points across target dimensions can be assumed. In such cases, standard deviation of the random variable within the interval [a, b] can be computed as: $\sigma = \frac{b-a}{\sqrt{12}}$.

For example, for the targets that are 1 m wide, standard deviation can be configured as: $\frac{1}{\sqrt{12}}$.

3.3.3.1.3 Allocation Parameters

The reflection points reported in point cloud are associated with existing tracking instances. Points that don't get associated are subjects for the allocation decision. Each candidate point is clustered into an allocation set. To join the set, each point needs to be within maxDistance and maxVelThre from the set's centroid. Once the set is formed, it has to have more than setPointsThre members, and pass the minimal velocity and SNR thresholds. Table 6 lists these parameters.

Table 6. Allocation Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
SNR threshold	60.0f	—	Minimum total SNR for the allocation set, linear sum of power ratios
setSNRObscThre	60.0f	---	Minimum total SNR for the allocation set, linear sum of power ratios, when obscured by another target

Table 6. Allocation Parameters (continued)

PARAMETER	DEFAULT	DIM	DESCRIPTION
Velocity threshold	1.0f	m/s	Minimum radial velocity of the allocation set centroid
Points threshold	3U	—	Minimum number of points in the allocation set
maxDistanceThre	2.8f	m^2	Maximum squared distance between candidate and centroid to be part of the allocation set
maxVelThre	2.0f	m/s	Maximum velocity difference between candidate and centroid to be part of the allocation set

3.3.3.1.4 State Transition Parameters

Each tracking instance can be in either FREE, DETECT, or ACTIVE state. Once per frame, the instance can get HIT (have non-zero points associated to a target instance) or MISS (no points associated) event.

When in FREE state, the transition to DETECT state is made by the allocation decision. See Section 3.3.3.1.3 for the allocation decision configuration parameters. When in DETECT state, use the det2active threshold for the number of consecutive hits to transition to ACTIVE state, or det2free threshold of number of consecutive misses to transition back to FREE state. When in ACTIVE state, the handling of the MISS (no points associated) is as follows:

- If the target is in the static zone *and* the target motion model is close to static, then assume that the reason for no detection is because they were removed as static clutter. In this case, increment the miss count, and use the static2free threshold to extend the life expectation of the static targets.
- If the target is outside the static zone, then the assumption is made that the reason we didn't get the points is that target is exiting. In this case, we use exit2free threshold to quickly free the exiting targets.
- Otherwise, (meaning target is in the "static zone", but has non-zero motion in radial projection) we assume that the reason of not having detections is that target got obscured by other targets. In this case, we continue target motion according to the model, and use active2free threshold.

Table 7 lists the parameters used to set this behavior.

Table 7. State Transition Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
det2activeThre	3U	—	In DETECT state; how many consecutive HIT events needed to transition to ACTIVE state
det2freeThre	10U	—	In DETECT state; how many consecutive MISS events needed to transition to FREE state
active2freeThre	20U	—	In ACTIVE state and NORMAL condition; how many consecutive MISS events needed to transition to FREE state
static2freeThre	2000U	—	In ACTIVE state and STATIC condition; how many consecutive MISS events needed to transition to FREE state
exit2freeThre	10U	—	In ACTIVE state and EXIT condition; how many consecutive MISS events needed to transition to FREE state

3.3.3.1.5 Gating Parameters

The gating parameters set is used in the association process to provide a boundary for the points that can be associated with a given track. These parameters are target-specific. Table 8 lists each of these parameters.

Table 8. Gating Function Parameters

PARAMETER	DEFAULT	DIM	DESCRIPTION
Volume	12	—	Gating volume
LengthLimit	8	m	Gating limit in length
WidthLimit	4	m	Gating limit in width
VelocityLimit	0	m/s	Gating limit in velocity (0 – no limit)

The gating volume can be estimated as the volume of the ellipsoid, computed as $V = \frac{4\pi}{3}abc$, where a, b, and c are the expected target dimensions in range (m), angle (rad), and doppler (m/s).

For example, consider a person as a radar target. For the target center, we could want to reach ±0.45 m in range (a = 0.9), ±3 degree in azimuth (b = 6π / 180), and ±5.18 m/s in radial velocity (c = 10.36), resulting in a volume of approximately 4.

In addition to setting the volume of the gating ellipsoid, the limits can be imposed to protect the ellipsoid from overstretching. The limits are the function of the geometry and motion of the expected targets. For example, setting the WidthLimit to 8 m does not allow the gating function to stretch beyond 8 m in width.

3.3.4 Memory Use

The Cortex-R4F uses tightly-coupled memories (256KB of TCMA and 192KB of TCMB). TCMA is used for program and constants (PROG), while TCMB is used for RW data (DATA). Memory use at the Cortex-R4F is summarized in the following tables. Table 9 lists the total memory footprint, indicating memory use.

Table 9. Cortex-R4F Memory Use

MEMORY	AVAILABLE (BYTES)	USED (BYTES)	USE (PERCENTAGE)
PROGRAM	261888	103170	39%
DATA	196608	171370	87%

Table 10 lists the memory used by the tracking algorithm in percentages to a total memory footprint. The tracking algorithm is instantiated with 250 maximum measurements in point-cloud input, and a maximum of 20 tracks to maintain at any given time.

Table 10. Group Tracking Algorithm Memory Use

MEMORY	AVAILABLE (BYTES)	USED BY GTRACK (BYTES)	USE (PERCENTAGE)
PROGRAM	103710	12609	12%
DATA	92056	14650	16%

3.4 System Timing

System timing is illustrated in Figure 12. Low level processing operates both on chirp by chirp, and frame by frame basis. High layer processing gets awaken once every frame.

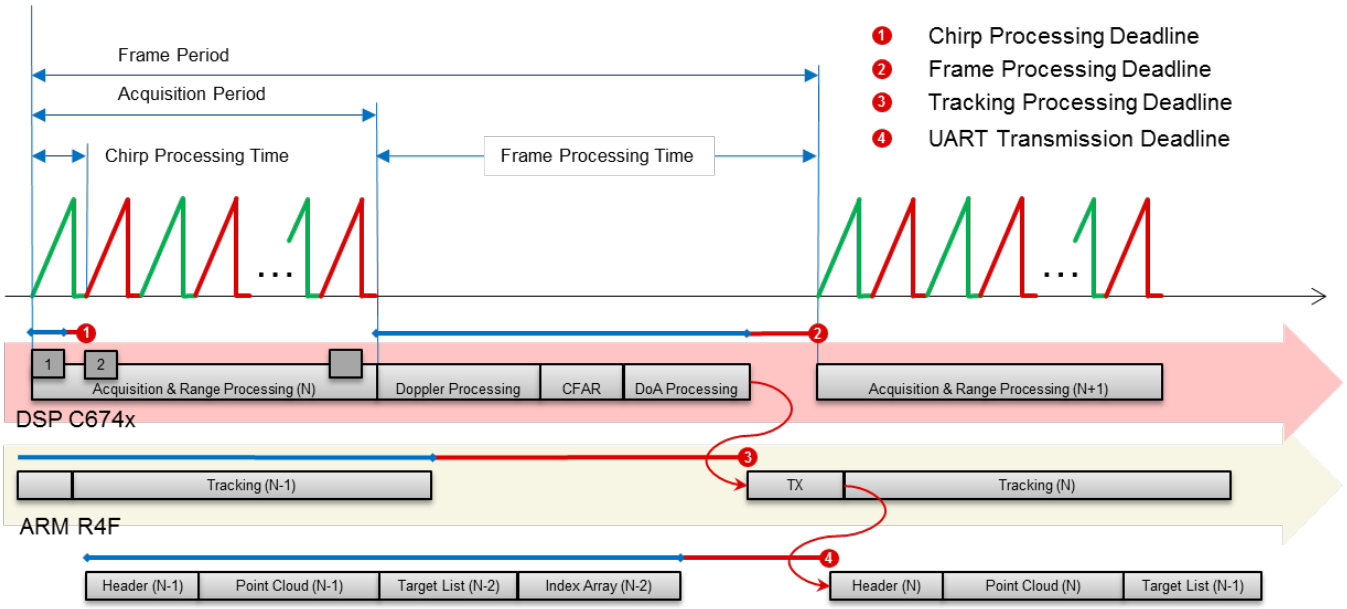


Figure 12. System Timing Diagram

Low level processing chain implemented in DSP C674x has two hard deadlines:

1. Chirp Processing deadline, which is defined as a latest time the acquisition and range processing for the given chirp shall complete. This is a hard deadline, and the available margin is used to estimate DSP loading during the acquisition period.
2. Frame Processing deadline, which is defined as a latest time the frame processing (Doppler, CFAR, and DoA) shall complete for a given frame. This is also a hard deadline, and DSP loading during frame processing can be estimated.

High level processing has two soft deadlines:

1. Tracking processing deadline, which is defined as a latest time the high level processing shall complete for a given frame.
2. Transport capacity deadline, which indicates the latest time the output data can appear at the interface. Since the amount of data sent varies from frame to frame, and system shall be tolerable to temporal overshoots, we would be looking at the long time average capacity values. These long time average values shall not exceed 90% of interface capacity.

3.4.1 DSP MIPS Summary

Table 11 provides benchmarked results and estimates the resulted loading of CPUs.

Table 11. DSP MIPS Usage Summary

	AVAILABLE TIME	USED TIME	LOADING
Active Chirp Time	64.65 us	37.5 us	58%
Frame Time	46 ms	15.5 ms	34%

3.4.2 ARM R4F Processing Time

The amount of time needed by ARM R4F to process input point cloud and deliver target information is a function of number of targets currently tracked, and number of measurements (points in input point Cloud) received. Figure 13 shows the results collected (blue) and a bound defined (red).

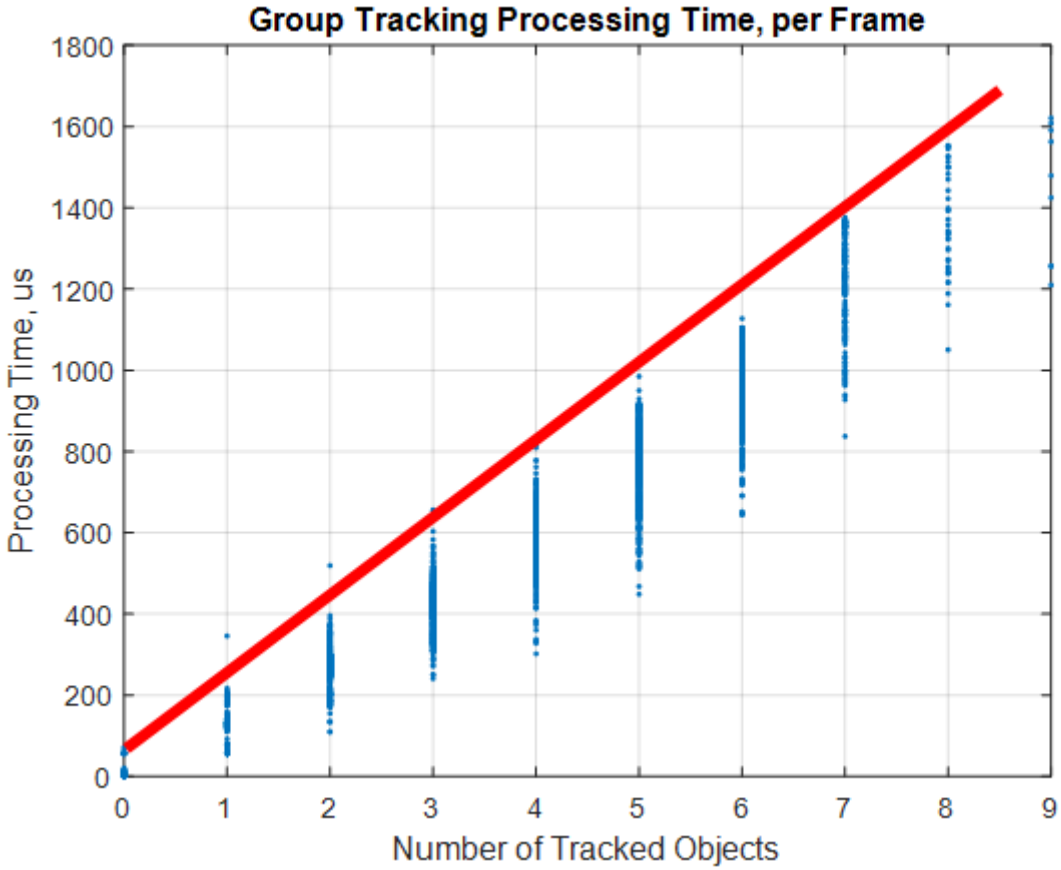


Figure 13. ARM R4F Processing Time

We observed that processing time increases linearly with number of tracking objects. With fixed number of tracking objects, complexity increases linearly with number of input points. Taking the worst case of number of points per frame of 250, we can derive a bound of about ≈ 200µs per tracking object. With frame time of 50ms, tracking 20 targets shall be completed in 4ms, which will consume <10% of CPU.

3.5 UART Communications

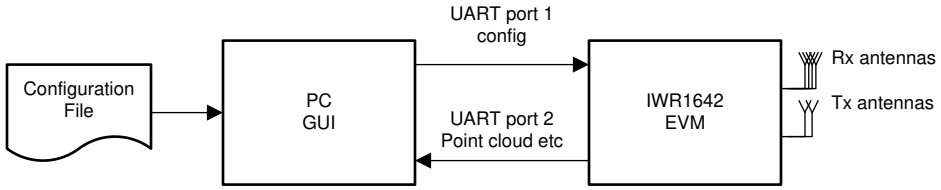


Figure 14. IWR1642 UART Communication

As illustrated in Figure 14, the example processing chain uses one UART port to receive input configuration to the front end and signal processing chain and uses the second UART port to send out processing results for display. See the information included in the software package for detailed information on the format of the input configuration and output results.

3.5.1 Input Configuration Format

Input configuration file is used to configure the sensor front end, lower level processing, and higher level processing. Input configuration formats are described in detail in chirpParams_TMdemo.xlsx

3.5.2 Output Results Format

Transport process at R4F outputs one frame every frame period. Frame has a fixed header, followed by variable number of segments in Tag/Length/Value (TLV) format. Each TLV has a fixed header, followed by variable size payload. Byte order is little endian.

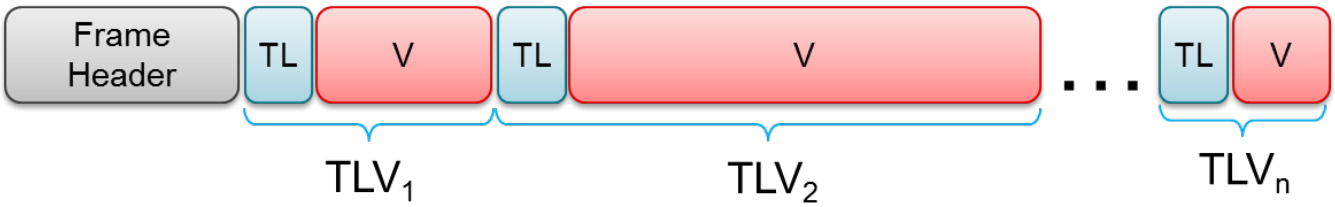


Figure 15. Output Results Format

4 Implementation Considerations

4.1 Floating-Point Versus Fixed-Point Implementation

The C674x DSP integrated in the IWR1642 offers a rich set of fixed-point and floating-point instructions. The floating-point instruction set can accomplish addition, subtraction, multiplication, and conversion between 32-bit fixed point and floating point—in single cycle for single-precision floating point and in one to two cycles for double-precision floating point. The majority of the single-precision, floating-point instructions are at the same speed as 32-bit fixed-point instructions (in fact the single-precision, floating-point FFT is almost as efficient as a 32-bit, fixed-point FFT). There are also fast instructions to calculate reciprocal and reciprocal square root in single cycle with 8-bit precision. With one or more iterations of Newton-Raphson interpolation, the user can achieve higher precision in a few tens of cycles. Another advantage of using floating-point arithmetic is that user can maintain both precision and dynamic range of the input and output signal without spending CPU cycles checking dynamic range of the signal or rescaling intermediate computation results to prevent overflow or underflow. These would enable user to skip or do less requalification of the fixed-point implementation of an algorithm, which makes algorithm porting much simpler and faster.

With the above, the 16-bit, fixed-point operations are two to four times faster than the corresponding single-precision, floating-point instructions. Trade-offs between precision, dynamic range, and cycles cost must be carefully examined to select suitable implementation schemes.

For the example of 1D FFT because the maximum effective ADC bit per sample is about 10 bits, under the noise and cluttered condition the output peak to average ratio is limited (not a delta function in the ideal case), data size does not expand between input and output. Because the deadline requirement for chirp processing is in general tight, a 16-bit, fixed-point FFT is used for the balanced dynamic range and SNR performance, memory consumption, and cycle consumption.

For the example of 2D FFT, there is additional signal accumulation in the Doppler domain, thus output signal peak tends to be very big. A single-precision, floating-point FFT is used, so adjusting the input signal level (which may cause SNR loss) or having a special FFT to have dynamic scaling for each butterfly is not required—both could have much higher cycle cost. In addition, because there is 2D windowing function before FFT, the data reformatting from 16-bit IQ to single-precision, floating-point and 2D windowing can be combined without additional cycle cost. The drawback of the floating-point FFT is output data size is doubled from input data size. The 2D FFT results cannot be stored back to the radar cube. For DoA detection, reconstructing the 2D FFT results per detected object is required at additional cycle cost.

For the example of clustering, the 16-bit fixed-point can safely cover the dynamic range and precision requirement of the maximum range and range resolution. The arithmetic involved is distance between two point and decision logic, which can be easily implemented using 16-bit, fixed-point multiplications instructions and 32-bit fixed-point condition check instructions. Therefore, a fixed-point implementation is used, which is about two times the cycle improvement of the floating-point implementation.

4.2 EDMA Versus DSP Core Memory Access

Enhanced direct memory access (EDMA) provides efficient data transfer between various memories with minimum DSP core intervention and cost. In general data movement in radar processing chain is very regular and ordered, whereas data from lower-level, slow memory is moved to higher-level faster memory for DSP processing then transfers back to lower-level memory for storage. Therefore, EDMA is the preferred way to accomplish most of these data movements. Specifically, the Ping-Pong scheme can be used for EDMA to parallelize data transfer and signal processing, so that at steady state, there is no overhead for data movement.

There are couple of scenarios that must consider the trade-offs between using EDMA and direct core access.

First, if there is irregular data access pattern for a processing module, using EDMA would be very cumbersome and sometimes impractical. For example of the two-pass CFAR algorithm used in the example signal processing chain, a CFAR-CASO search must be conducted in range domain then immediately conduct a CFAR-CA search in Doppler domain to confirm the results in the first pass. For this 2D alike search, using EDMA for data movement could be very cumbersome. Therefore, DSP is used to access L3 memory directly with L1D cache for L3 memory turned on. Cycle performance degraded to 1.8x to 2x of the entire power heat map stored in L2 memory (thus no EDMA involved). Flexibility is gained if required to change search order or do other algorithm tuning because no hardcoded EDMA is tied with this implementation, and there is no requirement to use any local buffer in L2 memory to store power heat map. With the current memory usage and cycle cost, it is a good design choice.

Secondly when the size of the data transfer is small, the EDMA overhead (setting up PaRamSet, triggering the EDMA, and checking the finish of EDMA) compared to the signal processing cost itself becomes bigger and might be more cycle efficient to use direct DSP access to L3 with L1 cache on. It has been observed for small 2D FFT size of 32, direct core access costs less cycles than using EDMA. In addition, code is simpler without Ping-Pong scheme and EDMA.

4.3 DSP Memory Optimization

In order to optimize the DSP memory, portions of the L1D and L1P are configured as SRAM.

There are 32KB of L1D and 32KB of L1P in C674x. Typically memory is configured as L1D cache and L1P cache as a whole, but for radar processing chain, the data and program memory footprint is relatively small, which makes it possible to carve out portion of L1D and L1P and use them as SRAM without any cycle performance impact.

In our implementation, 16KB of L1D are configured as L1 data cache. The remaining 16KB are configured as data SRAM. The EDMA input and output Ping-Pong buffers are allocated in this fast memory and shared between range processing and Doppler processing.

4KB of L1P is configured as L1 program cache. The remaining 28KB is configured as program SRAM, which holds the majority of the real-time frame-work code and all algorithm kernels except tracking.

With this implementation, 40KB of L2 memory was saved, which can be used for adding new algorithms or for other optimizations. There was approximately a 5% to 10% cycle improvement for range processing while no cycle penalty for other modules with data buffers in L2 or L3 memory was observed. Specifically for range processing because all functions are in L1P SRAM, all input and output buffers are in L1D SRAM and only FFT twiddle factors are in L2, but the FFT will be fetch to L1D cache and stay there for the all antennas and all chirps. There is very small cycle fluctuation because there is much less L1 cache operations at the background.

4.4 Radial Velocity Extension Support

Low level processing delivers radial velocity information for every detected point. Regardless of the true radial velocity of the object, Tthe value of this measurement is limited to the range

$$V_r \in \left[-V_{r,max}, \ldots, 0, \ldots + V_{r,max} - V_{r,min}\right]$$

where $V_{r,\ max}$ is maximum radial velocity of the sensor, and $V_{r,\ min}$ is radial velocity resolution. Both $V_{r,\ max}$ and $V_{r,\ min}$ are derived from RF chirp configuration. Therefore in the absence of extra processing, there will be ambiguity because the velocity measurement will wrap around.

High level processing uses radial velocity measurements to estimate target position and velocities in Cartesian space.

Therefore, theoretically, the radar-based traffic monitoring solution should be able to resolve any ambiguity on the target radial velocity. However, for the algorithm to work properly, additional work is required both at low and high level processing.

In the case of TDM-MIMO, Doppler effect for the detected point needs to be compensated for the second Tx antenna for angle estimation. Velocity ambiguity will further cause angle ambiguity when applying the Doppler compensation. It's the job of low level processing to correct the angle ambiguity. And it's the high level processing's job to unroll the velocity of the target based on target position and velocity estimation.

4.4.1 Low Level Processing Support of Radial Velocity Extension

4.4.1.1 Angle Estimation with TDM-MIMO and Velocity Ambiguity

For 2 Tx antenna TDM-MIMO, phase compensation needs to be done for the antenna samples received from the 2nd Tx antenna,

$$\Delta\varphi = 2\pi l2N \quad (1)$$

, where

$$l=[-N2,\ldots,N2-1] \quad (2)$$

is the Doppler index for the detected object, and N is the length of Doppler FFT.

In the case of velocity ambiguity, let v be the radial velocity of the detected object with Doppler index l from CFAR module. l'=l+i*N will represent radial velocity of v'=v+i*Vr,max, where i is an integer number, and l' will be physically the same as l caused by velocity ambiguity. In turn, the phase compensation for 2nd for v' will be

$$\Delta\varphi' = 2\pi l'2N=\Delta\varphi+i*+\pi \quad (3)$$

If we examine the formula above carefully, there are only 2 possible values of Doppler compensation for antenna samples from 2nd Tx antenna: H1=exp(-j*(Δφ+(2*k* π)) = exp(-j*Δφ), or H2=exp(-j*(Δφ+((2*k+1)* π)) = -exp(-j*Δφ).

In angular spectrum domain, as shown in Figure 16, when we apply Doppler compensation that correctly reflects the phase rotation from the true Doppler, we will see a spectrum shape in the blue curve, while the red curve represents the angular spectrum from compensation with a wrong hypothesis.

In the low level processing chain, we used a simple technique of just applying these two hypotheses of compensation factor to the antennas samples from 2nd Tx antenna, performing 2 angle estimations on 2 sets of data, and choosing the angle estimate corresponding to the larger peak from the angular spectrum. We call this angle correction for Vmax extension.

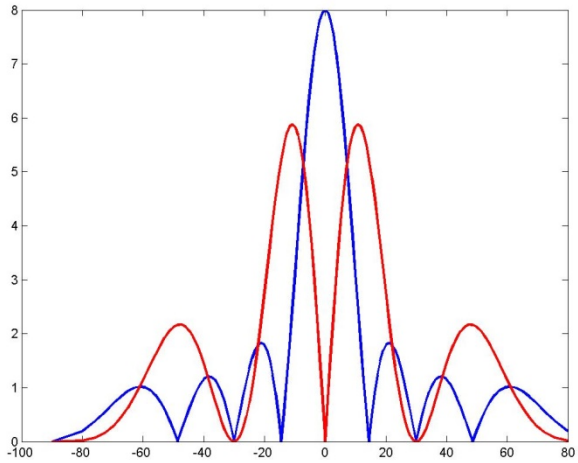


Figure 16. Angular spectrum comparison for antenna samples compensated with 2 hypotheses

4.4.1.2 Performance Evaluation of the Angle Correction for Vmax Extension

We used the raw data capture from mid-range chirp configuration, for which the maximum radial velocity (Vr, max) is 7.5 m/s, to verify the performance of the angle correction algorithm described in section Section 4.4.1.1.

The following plot is from an example of a sedan driving away from the sensor, at the boresight, with speed gradually increasing from below Vr,max, to beyond Vr,max but below 2*Vr,max.

In Figure 17, we color coded all the detected points with angle using Doppler compensation factor H1 as red points, and all the detected points with angle using Doppler compensation factor H2 as blue points. Shown in the upper left velocity estimates plot, velocity estimates started with positive small values while the car was driving away from the sensor with lower speed. As car speed increased, when exceeding the maximum velocity Vr,max, the velocity estimates wrapped around to -Vr,max and then continue increasing.

Ideally, all the detected points with speed below Vr,max should have selected angle estimation from compensating by H1, meaning they should be all red color. And all detected points with speed above Vr,max should have selected angle estimation from compensating by H2, meaning they should be all blue color.

The errors happen when detection points have impairments in the spatial domain, either due to higher noise, or stronger interferences, which is shown in the lower left plot within the two ellipses.

But as shown in Figure 17 lower left plot, the erroneous angle estimates are sparse, and none-biased. They make up about 6% of total detection points from tests we performed. And we have observed that the higher level processing is able to tolerate this error rate.

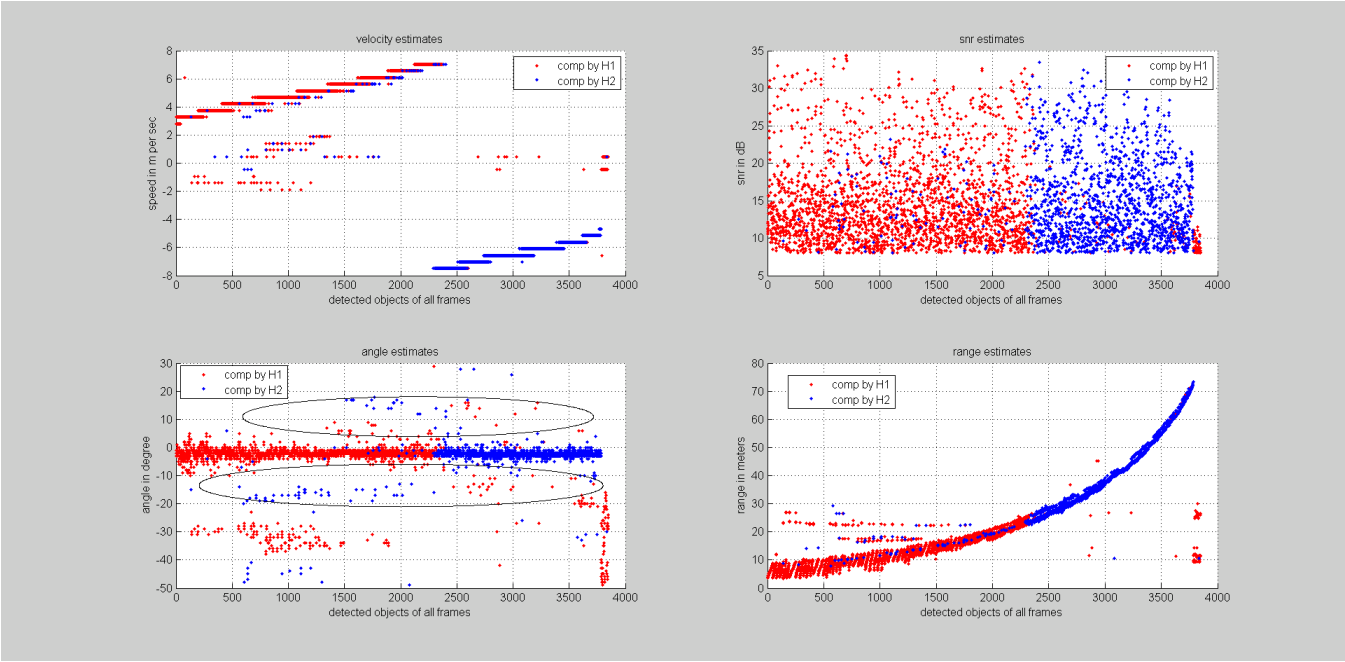


Figure 17. Angle correction performance plot

4.4.1.3 Sensitivity to Phase Non-ideality of the EVM Board

During the test, we have observed significant increasing of the angle correction/ compensation errors, with some EVM boards reaching 35% errors, due of antenna phase non-ideality from imperfection of the EVM antenna alignment. This will cause symptoms such as, wrong angle of the vehicle from both point cloud and tracker, or the point cloud of the vehicle suddenly jumping to another angle and hence track of the vehicle breaking into two tracks, or strong side ghost point cloud in addition to the vehicle point cloud and hence a ghost track created.

We strongly recommend user to calibrate the EVM using procedure described in section “Range Bias and Rx Channel Gain/Offset Measurement and Compensation” in mmwave SDK OOB demo document in (<SDK_INSTALL_PATH>/packages/ti/demo/xwr16xx/mmw/docs/doxygen/html/index.html). Once the calibration coefficients have been obtained for the board to be used for TM demo testing, they need to be copied to the corresponding chirp configuration file (such as mmw_tm_demo_ph2.cfg, etc) so that the low level processing chain can compensate the phase non-ideality to be able to estimate angle correctly.

4.4.2 High Level Processing Support of Radial Velocity Extension

High level processing takes the values of $V_{r,max}$ and $V_{r,min}$ as configuration parameters. Group tracking algorithm was modified to support the “unrolling” of radial velocity. The unrolling process differs in each tracking step.

At association step, we perform Gating & Scoring function for each measurement point $\{R_i, \varphi_i, V_{ri}\}$ and each existing track *n*. Figure 18 illustrates the process. For each existing track we use estimated radial velocity $V_{r,n}$ to unroll the input velocity. Essentially, we explore different unrolling hypothesis to find the closest target.

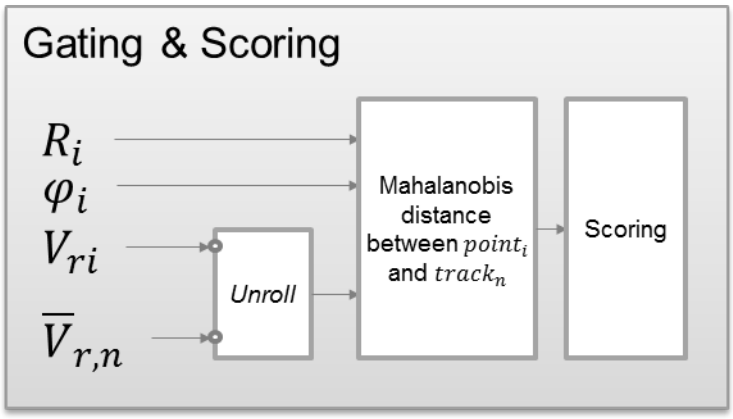


Figure 18. Velocity Unrolling in Association Step

The points not associated with any existing targets goes through allocation function. The unrolling function here uses the leading point's radial velocity $V_{r,\ i=0}$ to unroll the velocity of the other points in the allocation set.

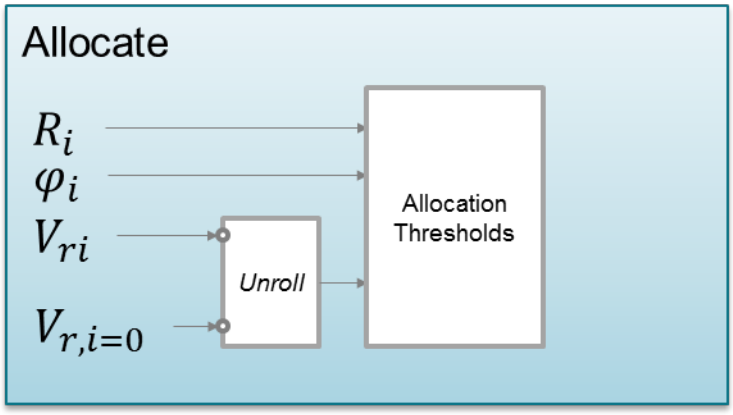


Figure 19. Velocity Unrolling in Allocation Step

All points associated with a given target are used in update step. Each tracker implements simple state machine to handle velocity unrolling. During initial steps, velocity is unrolled based on computed range rate R_n. Range rate is computed based on difference between range at allocation time and current range, divided by the time elapsed. Once range rate stabilizes, we monitor the error between predicted radial velocity and range rate. Once error diminishes, we switch unrolling base to the predicted radial velocity.

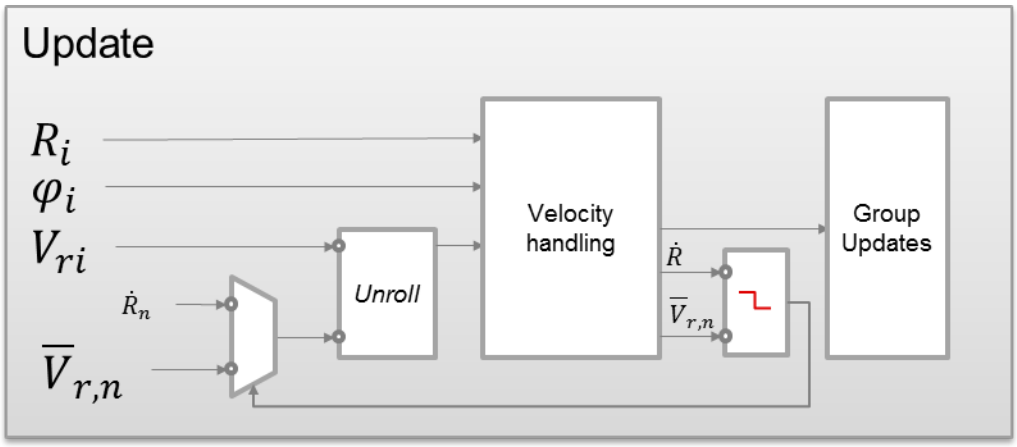


Figure 20. Velocity Unrolling in Update Step

The key factors here are that the performance of allocation step is not degraded because we use all points (even with wrong radial velocities) to pass the allocation step criteria's. We also observed good stability when switching to predicted velocities. This is due to cumulative properties of the range rate. After few (3-5) frames, we usually have good estimation of target's radial velocity

5 Hardware, Software, Testing Requirements, and Test Results

5.1 Required Hardware and Software

5.1.1 Hardware

The following hardware is required to get the demonstration running:

- IWR1642 EVM

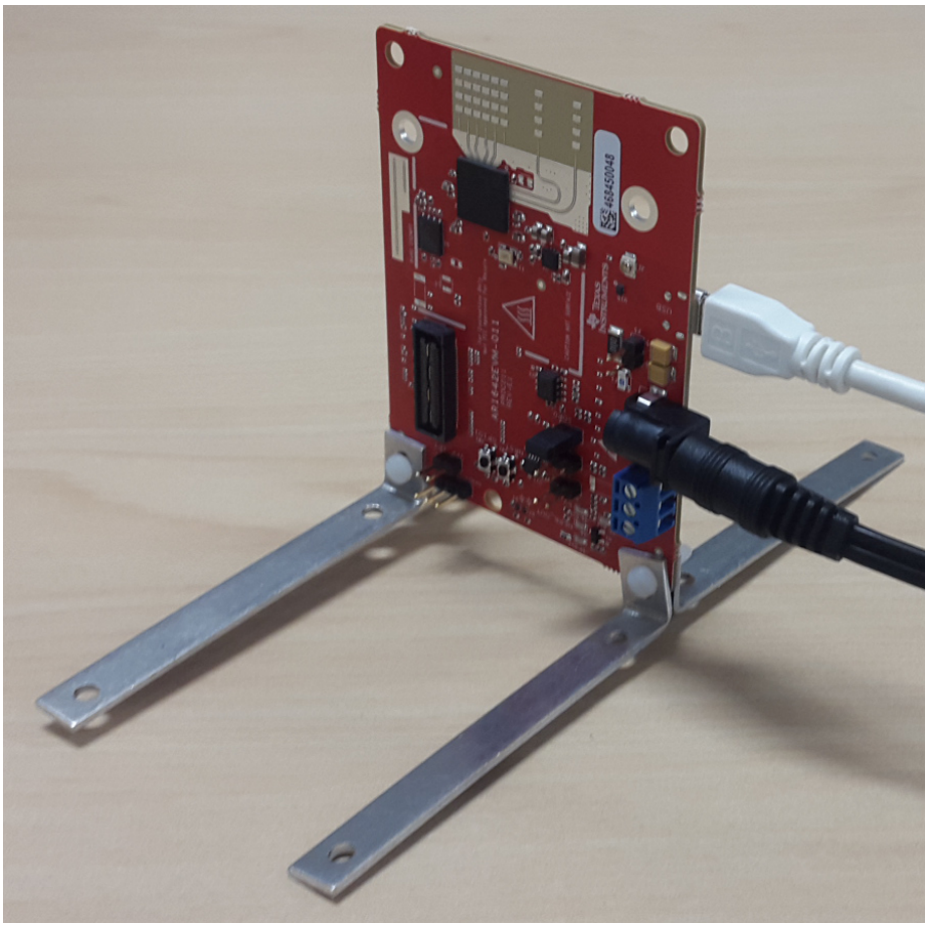


Figure 21. IWR1642 EVM

5.1.2 Software

The mmWave SDK can be downloaded from mmWAVE SDK.

To download the traffic monitoring application software used in this document, see the software link at TIDEP-0090. However, to get the latest version of the software, see the mmWave Industrial Toolbox Labs.

For Windows, the installation path should be as short as possible. Please use something like C:\tmdemo.

Upon installation, please follow the instructions provided in:

[TOOLBOX_INSTALL_PATH]\labs\traffic_monitoring

Details on how to run the pre-built binaries and how to rebuild the demonstration application are provided in the package.

5.2 Enabling Support for IWR6843 and IWR1843 Devices

The software, chirp configurations, and hardware setup details for the IWR6843 and IWR1843 are provided in the mmWave Industrial Toolbox Labs. Refer to the User's Guide available under the Traffic Monitoring section. Chirp configurations for these devices are provided with similar performance metrics as discussed in this reference design for the IWR1642 device. Additionally, these devices support a 3rd transmitter for 3D detected point clouds and tracking. The configurations for enabling 3D detections are also provided.

6 Testing and Results

6.1 Test Setup

In order to test traffic monitoring system, a test environment was constructed to approximate the scenes calculated from the geometry analysis. Figure 22 shows this test environment, which consists of a mark off for a four or five lane roadway with approximately 70 m of roadway length.

A *stop bar* was assumed to exist 20 m along the roadway as shown in Figure 23. Two sensor mounting positions were then defined based on the elevation and azimuth field of view of the IWR1642 EVM. One sensor mounting position was situated at 4.9 m above the ground over the center of the five lane roadway. The second sensor mounting position was situated at 4.9 m above the ground but placed off to the side of the four lane road as shown in Figure 24. Simulating a pole mount on the side of the road, the sensor was placed at a 20-m longitudinal distance from the stop bar and 2 m off to the side of the road with an approximately 10° azimuth angle towards the road as shown in Figure 25. In both cases the downtilt angle of the sensor was adjusted so that the field of view included about 5 meters in front of the stop bar at the near end of the road, as well as the far end of the road section which was approximately 70 meters away. This essentially simulated 50 meters of approach beyond the stop bar.

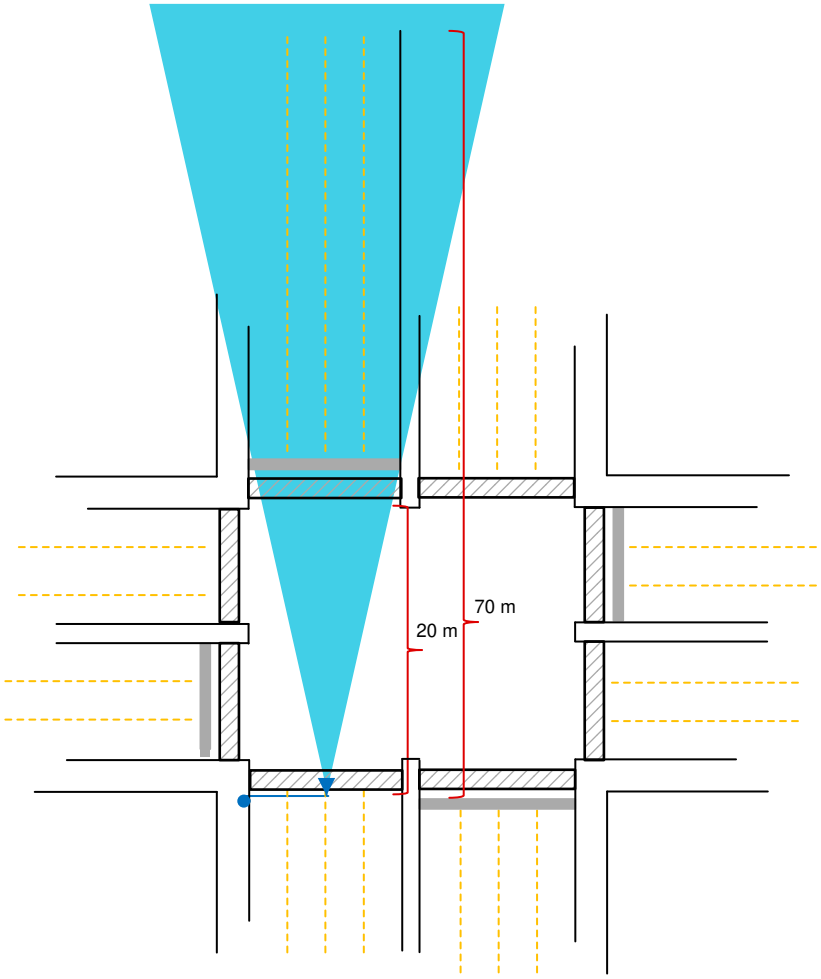


Figure 22. Test Setup Schematic Layout: First Sensor Mount

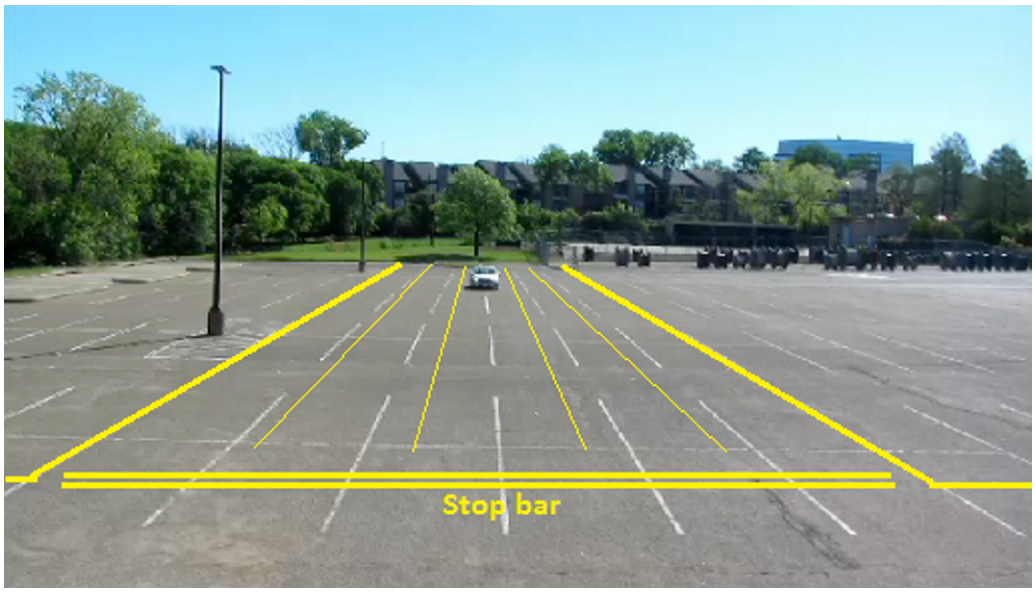


Figure 23. Test Setup Ground Truth First Sensor Mount

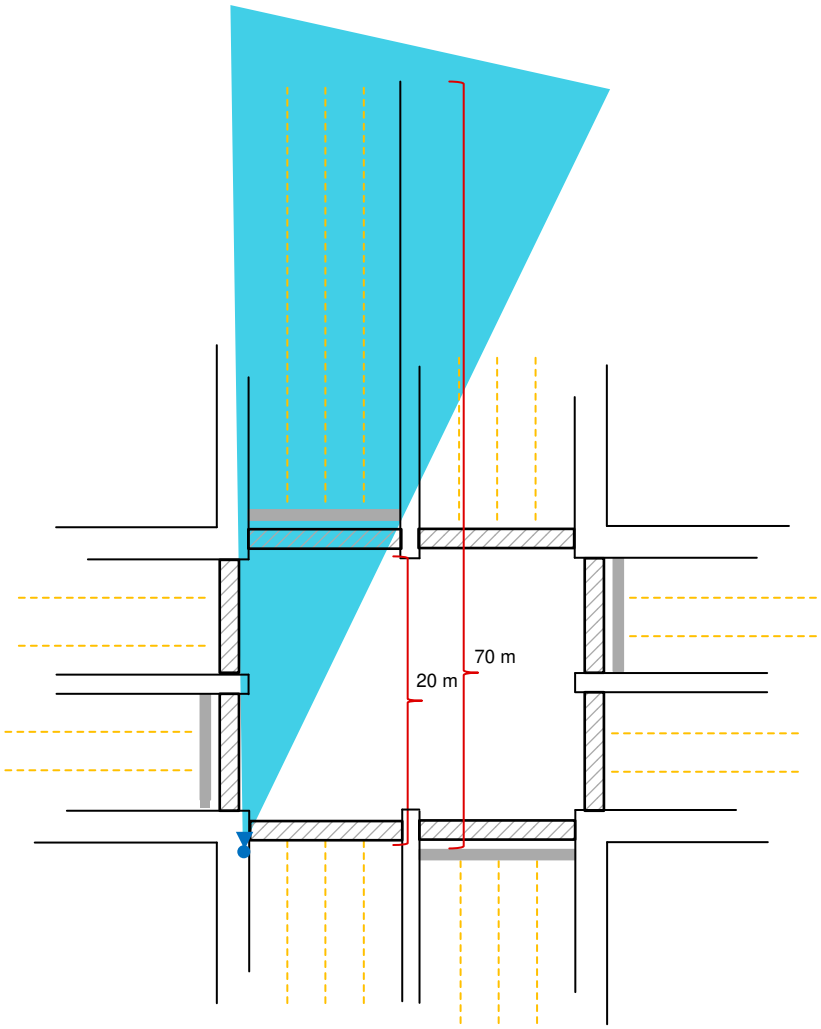


Figure 24. Test Setup Schematic Layout: Second Sensor Mount

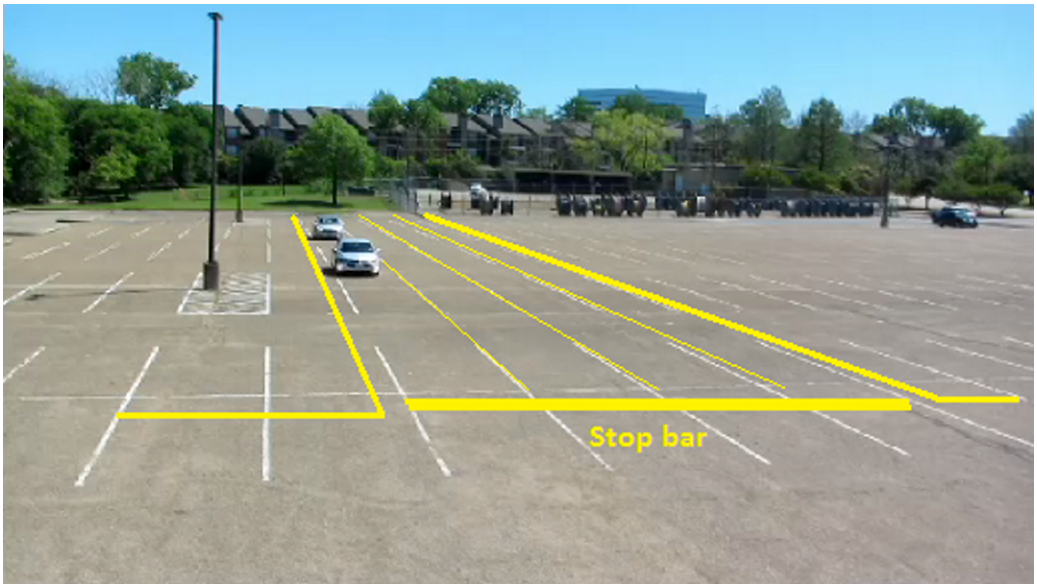


Figure 25. Test Setup Ground Truth Second Sensor Mount

6.2 Test Results

6.2.1 Range-Doppler Detection and Angle of Arrival (AoA) Estimation Test Results

Test results pertaining to the performance of the range-Doppler detector and the AoA estimator are presented for the medium-range MIMO chirp configuration described in Table 1. Results for three example test scenarios are described.

6.2.1.1 Test Scenario One Result

In this test scenario, the radar sensor is placed at 4.9 m above the ground over the middle of the street and approximately 20 m from the stop bar. One car moves towards the sensor at near constant velocity, starting from just outside the design target maximum range of 70 m and proceeding past the stop bar and out of the field of view.

Figure 26 illustrates the performance of the detector and the AoA estimator shortly after the car entered the design target maximum range. These results are generated from the processing chain described in Section 3.2 for a single-frame snapshot of the test data.

Figure 26(left) represents the ground truth picture (car marked by a red ellipse). Figure 26 (right) represents the range power profile. The thicker green line represents the zero Doppler and the other lines (thinner, multicolor) represent power profiles for non-zero Doppler values. The detections corresponding to the car are represented by the red round markers. The red ellipse indicates all detections corresponding to the moving car.

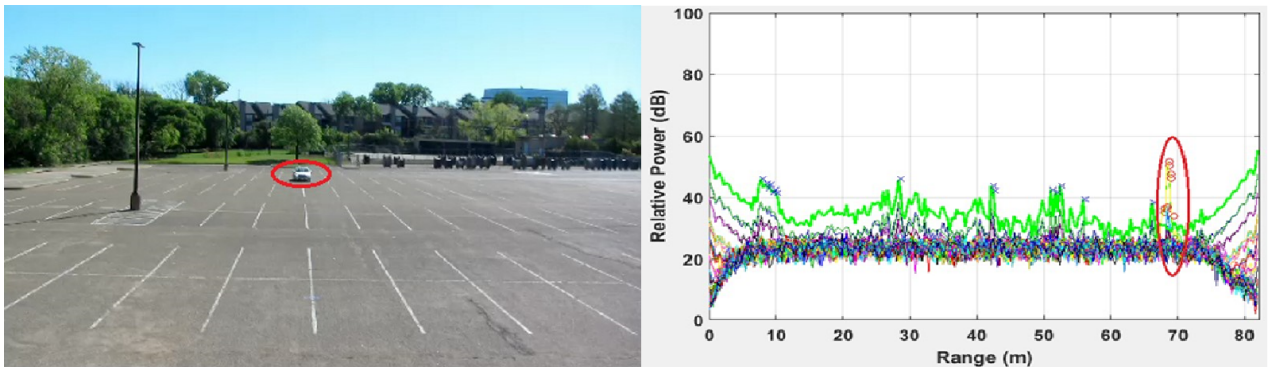


Figure 26. Scenario One: Ground Truth

Figure 27 (left) shows the range-Doppler power profile in the form of a heatmap. The red ellipse indicates the signal power corresponding to the moving car. Figure 27 (right) shows the detections in the x-y plane after AoA processing. The red ellipse indicates all detections corresponding to the moving car.

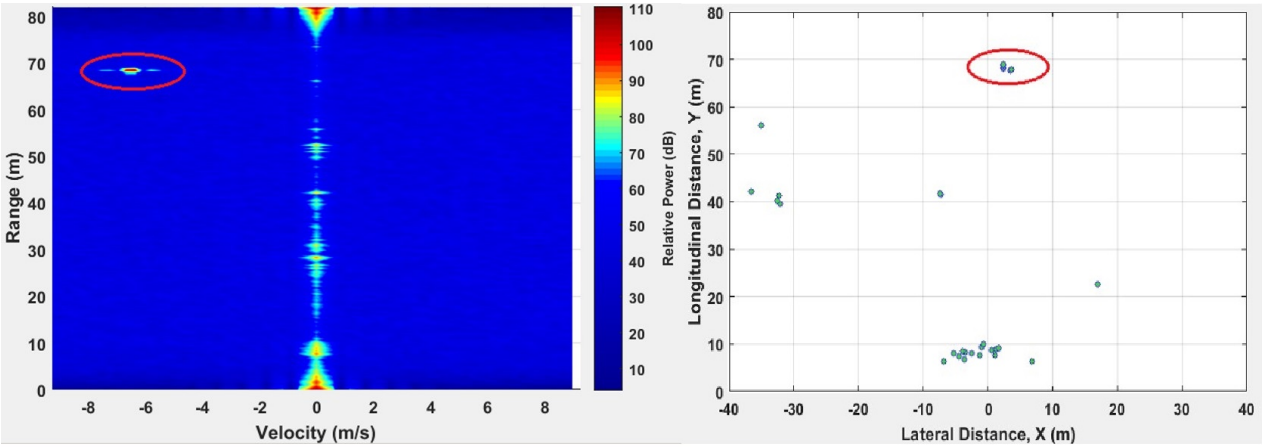


Figure 27. Scenario One: Heatmap Snapshot (left) and X-Y Detections (right)

6.2.1.2 Test Scenario One Observations

The results from test scenario one demonstrate the ability of the IWR1642 EVM with the medium range MIMO sensor configuration to detect a vehicle at the target maximum range of 70 m. Furthermore, the results also show the ability to estimate the radial velocity and, using the range and AoA data, the x-y positions of the reflected signals from the vehicle. In all three plots, some static objects and some static ground clutter can also be observed, which do not impede the ability to detect the moving vehicle.

6.2.1.3 Test Scenario Two Results

In this test scenario, the radar sensor is placed at 4.9 m above the ground, approximately 2 m from the edge of the assumed street, approximately at a longitudinal distance of 20 m from the assumed stop bar and with a 10° azimuth offset angle to the street. Two cars move towards the sensor one after the other in the nearest lane.

Figure 28 and Figure 29 illustrate the performance of the detector and the AoA estimator shortly after the second car entered the target maximum range These results are generated from the processing chain described in Section 3.2 for a single frame snapshot of the test data.

Figure 28(left) represents the ground truth picture (cars marked by red and blue ellipse). Figure 28(right) represents the range power profile. The thicker, green line represents the zero Doppler and the other lines (thinner, multicolor) represent power profiles for non-zero Doppler. The detections corresponding to a car are represented by red round markers. The red and blue ellipses indicate all detections corresponding to the two moving cars, closer and further away, respectively.

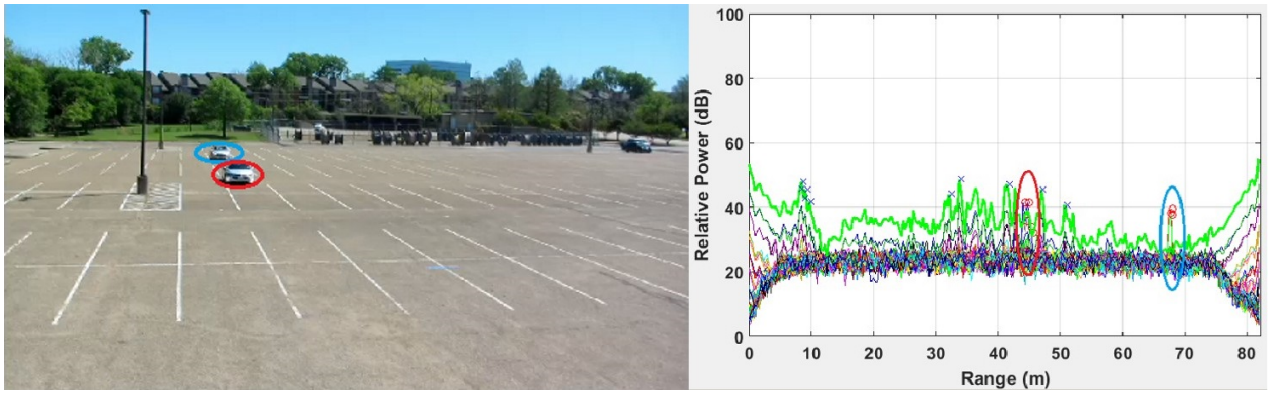


Figure 28. Scenario Two: Ground Truth (left) and Range Profile (right)

Figure 29(left) shows the range and Doppler power profile in the form of a heatmap. The red and blue ellipses indicate signal power corresponding to the two moving cars, closer and further away, respectively. Figure 29 (right) shows the detections in the x-y plane after AoA processing. The red and blue ellipses indicate all detections corresponding to the two moving cars, closer and further away, respectively.

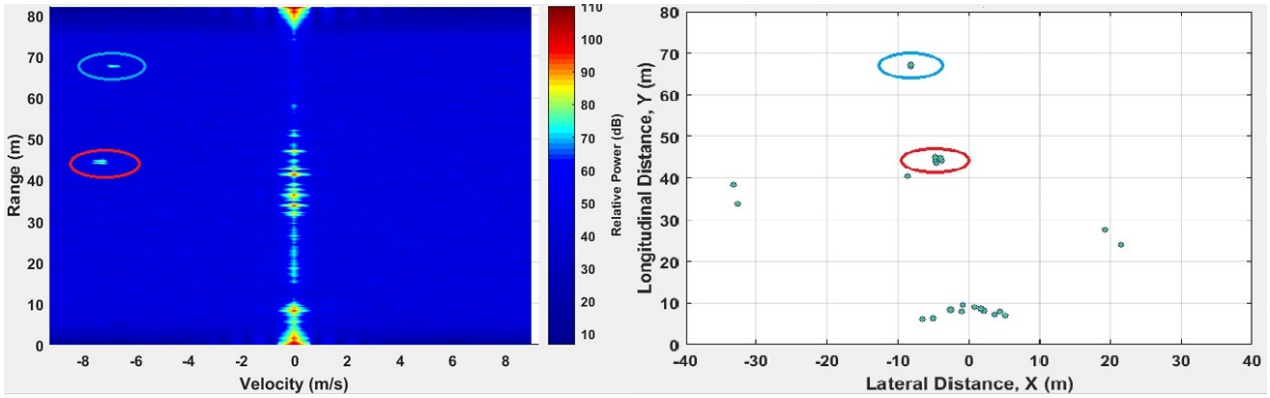


Figure 29. Scenario Two: Heatmap Snapshot (left) and X-Y Detections (right)

6.2.1.4 Test Scenario Two Observations

The results from test scenario two demonstrate the ability of the IWR1642 EVM with the medium range MIMO sensor configuration to detect multiple vehicles at the target maximum range of 70 m. Furthermore, the results also show the ability to estimate the radial velocity and, using the range and AoA estimates, the x-y positions of the reflected signals from the vehicles. In all three plots some static objects and some static ground clutter can also be observed. The data also indicates that the range-Doppler and AoA detection signals can form the basis for identifying and tracking multiple vehicles in a traffic monitoring scene.

6.2.2 Extended Range 120 to 200 m

Test results related to the performance of the range-Doppler detector and the AoA estimator are presented for the long range (120 - 200 m), non-MIMO chirp configuration described in Table 1. Recall that the chirp design goal was a maximum range of 120 m, but after setting all of the configuration parameters of the chirp design, the maximum range of the measurements extended out to approximately 200 m. The following tests assess the detector performance at 120 m and beyond.

6.2.2.1 Scenario Three Test Results

In this test scenario, the radar sensor is placed at 1.5 m above the ground in the middle of the street. One car moves towards the sensor at near constant velocity, starting from outside the design target maximum range of 120 m, approximately at 200 m, and proceeds past the stop bar and out of the radar field of view.

Figure 30 and Figure 31 illustrate the performance of the detector and the AoA estimator shortly after the car was detected for the first time and when the car was at about the design target maximum range. These results are generated from the processing chain described in Section 3.2 for a single-frame snapshot of the test data.

Car detected for the first time at approximately 175 m:

Figure 30 (left) represents the ground truth picture (car marked by a red ellipse). Figure 30 (right) represents the range power profile where the thicker green line represents the zero Doppler and other lines (thinner, multicolor) represent power profiles for non-zero Doppler values. The detections corresponding to the car are represented by the red, round markers. The red ellipse indicates all detections corresponding to the moving car.

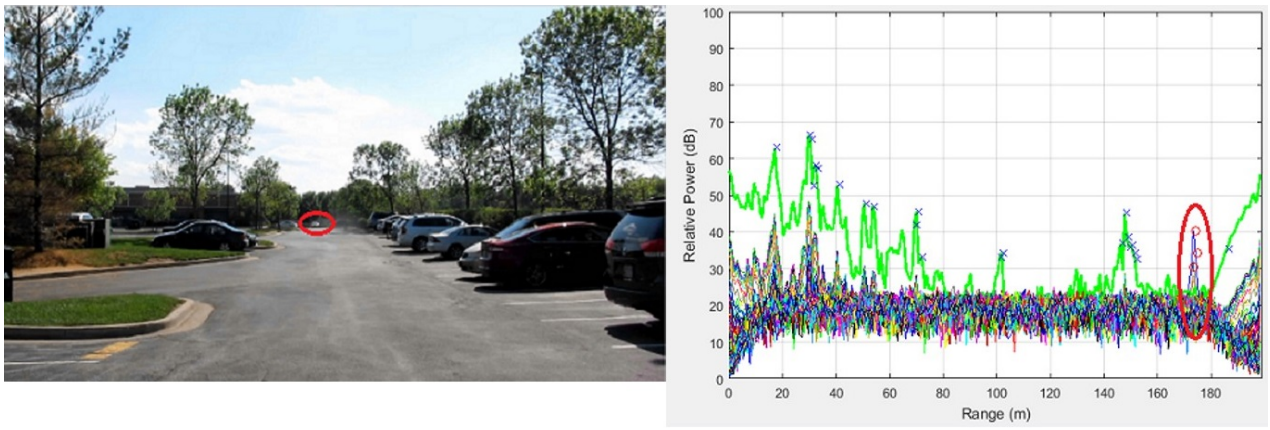


Figure 30. Scenario Three: 175-m Ground Truth (left) and Range Profile (right)

Figure 31 (left) shows the range-Doppler power profile in the form of a heatmap. The red ellipse indicates the signal power corresponding to the moving car. Figure 31 (right) shows the detections in the x-y plane after AoA processing. The red ellipse indicates all detections corresponding to the moving car.

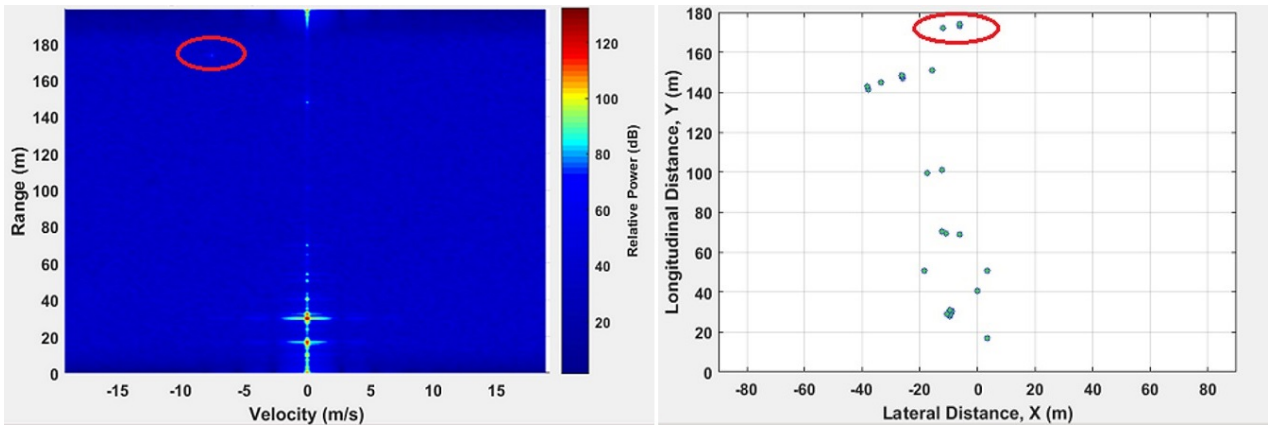


Figure 31. Scenario Three: 175-m Heatmap

Car at about the design target maximum range of 120 m:

Figure 32 (left) represents the ground truth picture (car marked by a red ellipse). Figure 32 (right) represents the range power profile where the thicker green line represents the zero Doppler and other lines (thinner, multicolor) represent power profiles for non-zero Doppler values. The detections corresponding to the car are represented by the red, round markers. The red ellipse indicates all detections corresponding to the moving car.

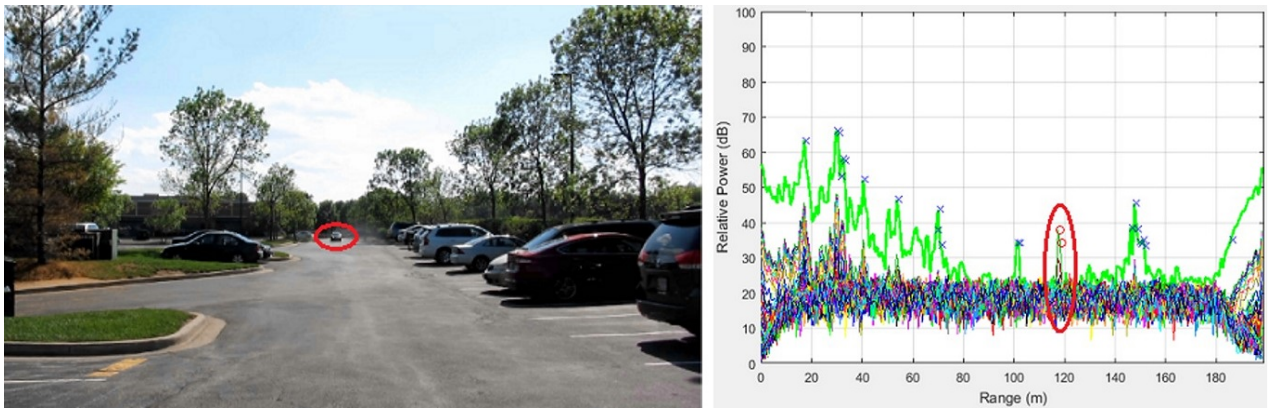


Figure 32. Scenario Three: 120-m Ground Truth (left) and Range Profile (right)

Figure 33 (left) shows the range-Doppler power profile in the form of a heatmap. The red ellipse indicates the signal power corresponding to the moving car. Figure 33 (right) shows the detections in the x-y plane after AoA processing. The red ellipse indicates all detections corresponding to the moving car.

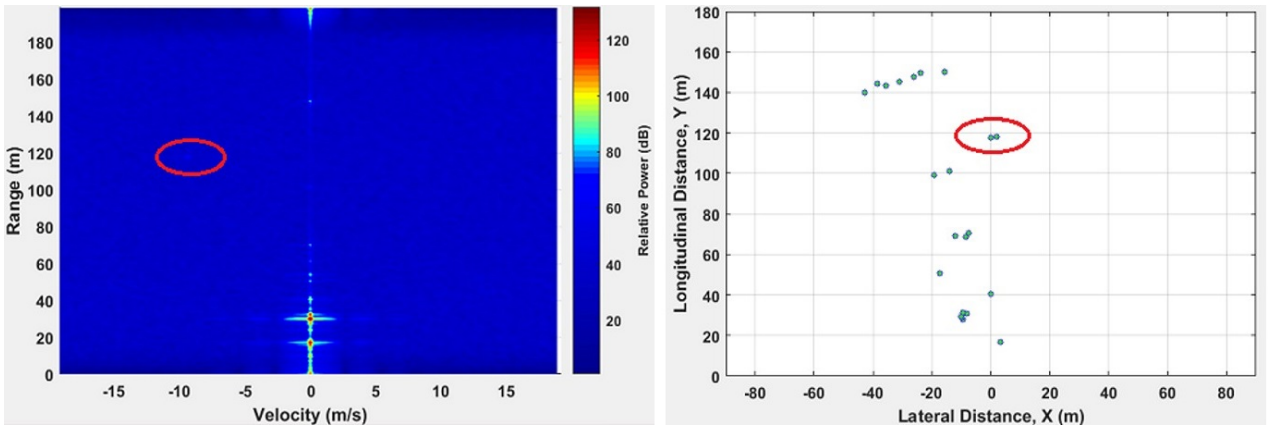


Figure 33. Scenario Three: 120-m Heatmap

6.2.2.2 Scenario Three Test Observations

The results from test scenario three demonstrate the ability of the IWR1642 EVM, with the large range, non-MIMO sensor configuration to detect a vehicle at the target maximum range of 120 m and beyond, with initial detection in this case at approximately 175 m. Furthermore, the results also show the ability to estimate the radial velocity and, using the range and AoA data, the x-y positions of the reflected signals from the vehicle. In all three plots some static objects and some static ground clutter can also be observed, which do not impede the ability to detect the moving vehicle. Similar range results are expected using a MIMO version of this chirp design as well. The main difference between the non-MIMO and MIMO designs would be a lower maximum velocity for the MIMO version.

7 System Level Performance

Traffic monitoring system is capable of detecting, and tracking targets at EVM, sending targets list to a Host PC for scene interpretation and visualization.

To asses system level performance of traffic monitoring system we construct appropriate test cases, record ground truth, and compute:

- accuracy of vehicle counting
- vehicle tracking reliability
- vehicle location estimations precision
- vehicle velocity estimation precision
- vehicle detection distance

Summary of performance results is presented in the table below. The test case and methodology on how we computed those values is explained in the subsections that follow.

Table 12. System Performance Parameters

PARAMETER	DIMENSION	MEASURED VALUE	SECTION
Vehicle Counting Reliability	%	97.8	7.1
Vehicle Tracking Reliability	%	86.2	7.2
Vehicle Location Precision	m	Xpos = 0.23, Ypos = 0.48	7.3
Vehicle Velocity Precision	m/s	Vx = 0.63, Vy = 0.44	7.4
Vehicle Detection Distance	m	54.7 (mean), 72.1 (max)	7.5

7.1 Vehicle Counting

7.1.1 Test Case Description

We simulated an intersection with 3 traffic lanes, stop line, and traffic signal. We run a test for 5 minutes, where vehicles are instructed to choose the line at random and obey the traffic signal. During the test, we first simulated the green light behavior, were vehicles are continuously passing the stop line. Then, we simulated the red light behavior, where the vehicles got stopped, (4 vehicles per lane were stopped waiting), then all moved away with simulated green signal.

Figure 34 illustrates test in process, at about 3.41 min, where two vehicles are fully stopped, and two other vehicles are approaching the stop line. At that moment, system was tracking 4 targets, 2 (moving) in the first lane, and 1 stopped in lanes 2 and 3.

System is capable of counting targets per traffic lane. Total of 35 vehicles were counted at that time; (12, 9, and 14) in each lane correspondently.

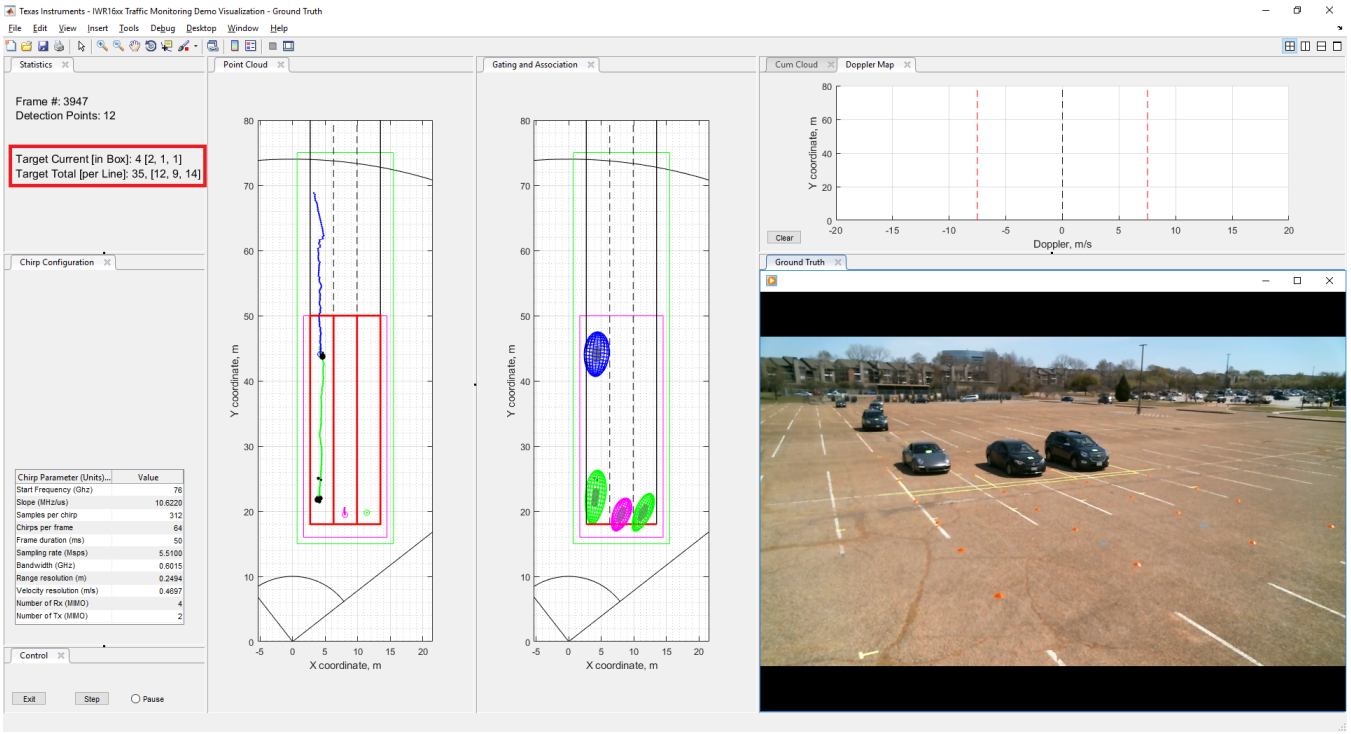


Figure 34. Vehicle Counting Illustration

After the test completed, we counted 44 vehicles, versus 45 per ground truth. The results are summarized in the table below.

Table 13. Vehicle Counting Reliability

LANE	NUMBER OF VEHICLES	NUMBER OF COUNTED VEHICLES	COUNTING RELIABILITY (%)
1	16	16	
2	12	12	
3	17	16	
Total	45	44	97.8%

7.2 Reliability of Vehicle Tracking

To asses tracking performance we define tracking reliability score. The score measures how many mistakes the tracker made in terms of missing tracks, false positives, mismatches, failures to recover tracks, etc.

7.2.1 Test Case Description

We processed the scene of about 2 minutes with 25 vehicles approaching, entering the intersection, crossing the stop line, and leaving.

7.2.2 Methodology

We define the measurements space as horizontal (no elevation) Cartesian plane. This plane has sensor in the origin, X axis is parallel to the stop line, and Y axis parallel to the traffic lanes.

At about the same time interval we capture the output of the tracker and high resolution video representing the ground truth. Output of the tracker is already in horizontal plane coordinates.

Video capture requires offline post processing. Below are the steps we performed:

- Identify vehicle location by a single point in horizontal plane (for example, point V1 at the picture below). This point is a vertical projection of the center of the vehicle's front bumper into a horizontal X-Y plane.
- Record each vehicle's point, in pixels, at regular video frame intervals.
- Calculate each vehicle trajectory in pixels.
- Interpolate the trajectory points to radar frame timestamps (50ms).
- Translate pixels into X-Y coordinates using projective geometry rules. explains the details.
  - Using known measurement points as a references, compute two vanishing points (VP1, and VP2)
  - Project each pixel into X and Y axis (V1 => V1x, V1y), and found the distances using cross ratios theorem.

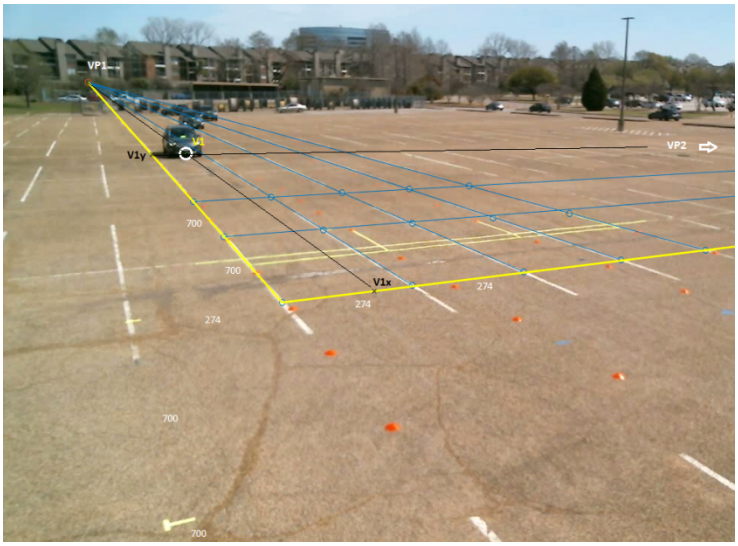


Figure 35. Translating Pixels to Real World Coordinates

Analyzing the output of the tracker, we look individually at the history of each tracking instance during the life cycle of the track: starting from the moment of allocation ending to the moment of freeing. At the moment of allocation we search for the closest vehicle point and assign it to the instance. We declare the tracking instance "good" if:

- At allocation time, there was a vehicle point "close enough". The closeness is measured by the Euclidean distance between the initial track position and closest unallocated vehicle point.
- The track is long enough. We declare an error if the track is short (less than 20 frames).
- The track shall be released at "exit zone". We declare an error if the track is released "too early".
- For every radar frame the track existed, we compute the distance between current track location and allocated vehicle position. If at any time the distance exceeds the desired threshold (4m), we declare an error.

The reliability score is a ratio on correctly tracked targets to a total number of tracks during long random run.

7.2.3 Results

Table 14. Vehicle Tracking Reliability

NUMBER OF VEHICLES	NUMBER OF GOOD TRACKS	TOTAL NUMBER OF TRACKS	TRACKING RELIABILITY %
25	25	29	86.2

The errors in tracking are typically due to early target detection, followed by few frames with no detections. As shown above, those errors may not be critical to vehicle counting.

7.3 Precision of Vehicle Position and Velocity Estimates

To compute expected errors in vehicle position and velocity estimations, we use the results of the previous test. In particular, we only look into the “good” tracks, where we had established a vehicle reference point at all times the track exists.

7.3.1 Test Case Description

We use the subset of the test described in Section 7.2.1.

7.3.2 Methodology

Figure 36 illustrates the evolution of Xpos value of each of 25 good tracks from the point of allocation (time tick 0) to the point of freeing. Those curves are multicolored. Green colors represent corresponding Xpos of the front of the vehicles as determined by video capture post processing.

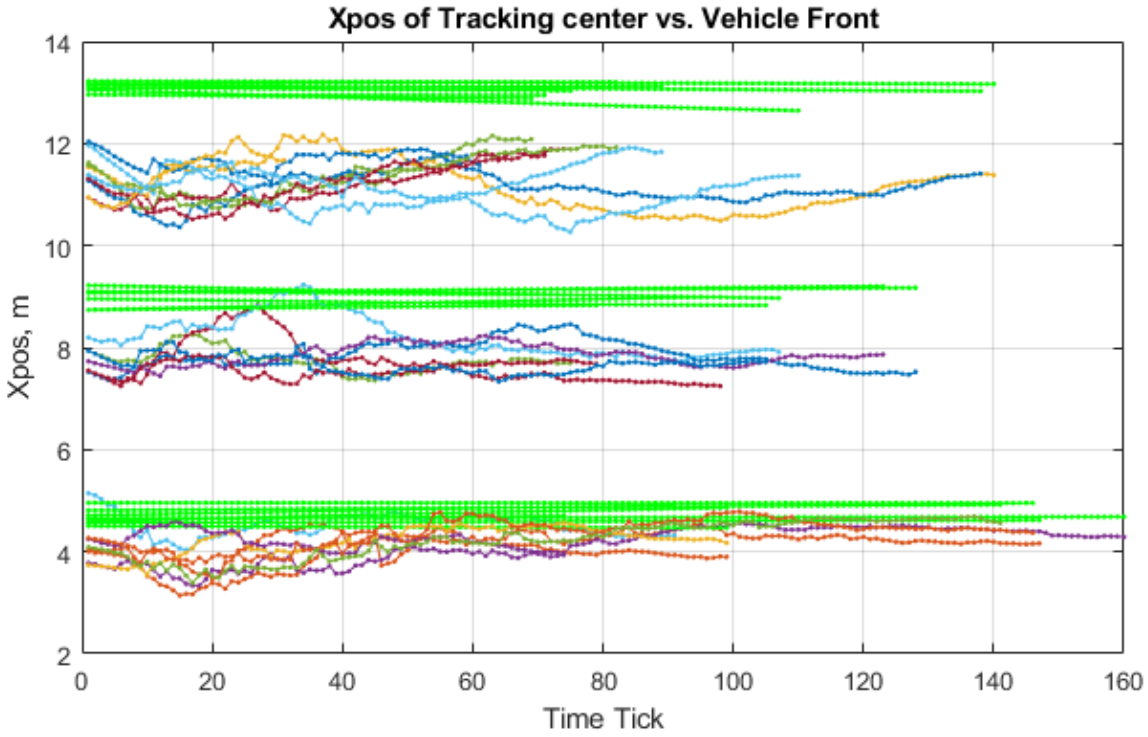


Figure 36. Tracking Xpos versus Vehicle Front

The bias can be explained as a distance between the vehicle point (defined as front center) and “point cloud” centroid. One can also observe that this bias is larger (about 1.2m) for the furthest lane, probably because we see more reflections from the sides of the car.

Because of the deterministic nature of this bias, it can be removed based on geometry knowledge, and we focus only on the standard deviation measure of Xpos.

Using similar methodology, Figure 37 shows the results we obtained after removing bias from position and velocity estimates. We present error statistic as a function of target distance from the X axis.

Figure 37. Positional Error - X

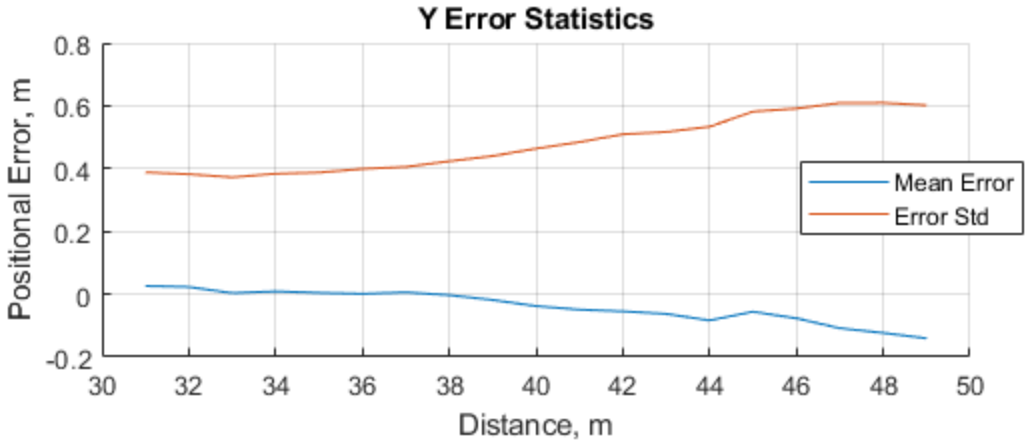


Figure 38. Positional Error - Y

Figure 39. Velocity Error - X

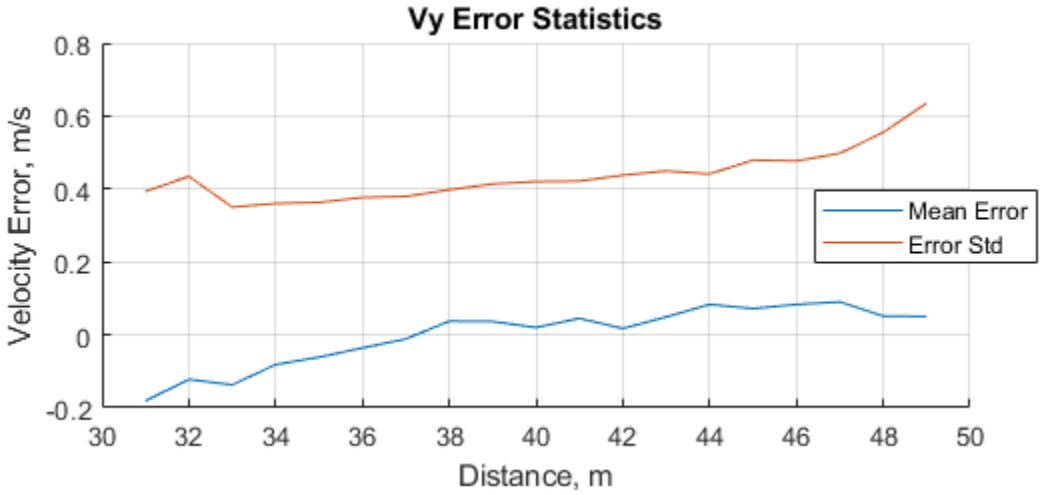


Figure 40. Velocity Error - Y

7.3.3 Results

Table 15. Vehicle Tracking Precision

TRACKING PRECISION	DIMENSION	ERROR, STD AT 40m OF DISTANCE
Latitude (Xpos)	m	0.23
Longitude (Ypos)	m	0.48
Velocity, Latitudinal (Vx)	m/s	0.63
Velocity, Longitudinal (Vy)	m/s	0.44

We observed that error statistics changes slightly as a function of target distance. Therefore, we reported the expected error at 40m distance.

7.4 Detection Distance

7.4.1 Test Case Description

We use the subset of the test described in Section 7.2.1.

7.4.2 Methodology

For each track, we record Y position at allocation time, then compute mean and max values.

7.4.3 Results

Table 16. Detection Distance

DETECTION DISTANCE	DIMENSION	DISTANCE FROM SENSOR
Mean Detection	m	54.7
Max Detection	m	72.1

7.5 System Behavior with Static Targets

The use case with multiple vehicles stopped at the stop line is probably the most challenging one. In this section we provide a description of expected system behavior.

When multiple vehicles are approaching the red light, the tracker shall have good estimation of each approaching target. When targets are slowing down, the tracker, using constant acceleration motion model, is expected to adapt to the velocity changes, and have reliable information that targets are approaching a full stop (becoming static). Once completely stopped, the detection layer is not expected to provide any reflection points. However, the tracker employs the logic where it maintains the properties (centroid positon, zero velocities/accelerations, as well as group dispersion and process covariance matrices) of the track. Therefore, it maintains the targets in static mode until they start moving again. Figure 41 shows one approaching vehicle at lane 1, and 9 static targets (three on each lane). Note that at a given frame, the detection points only correspond to a moving vehicle. Also note that tracker is missing 4th targets at lanes 2 and 3. This is because there were not enough reflection points for those vehicles (since they are still far away, and were partially obscured by other vehicles). However, once traffic starts moving, those missed targets will have a good chance to get a tracker allocated.

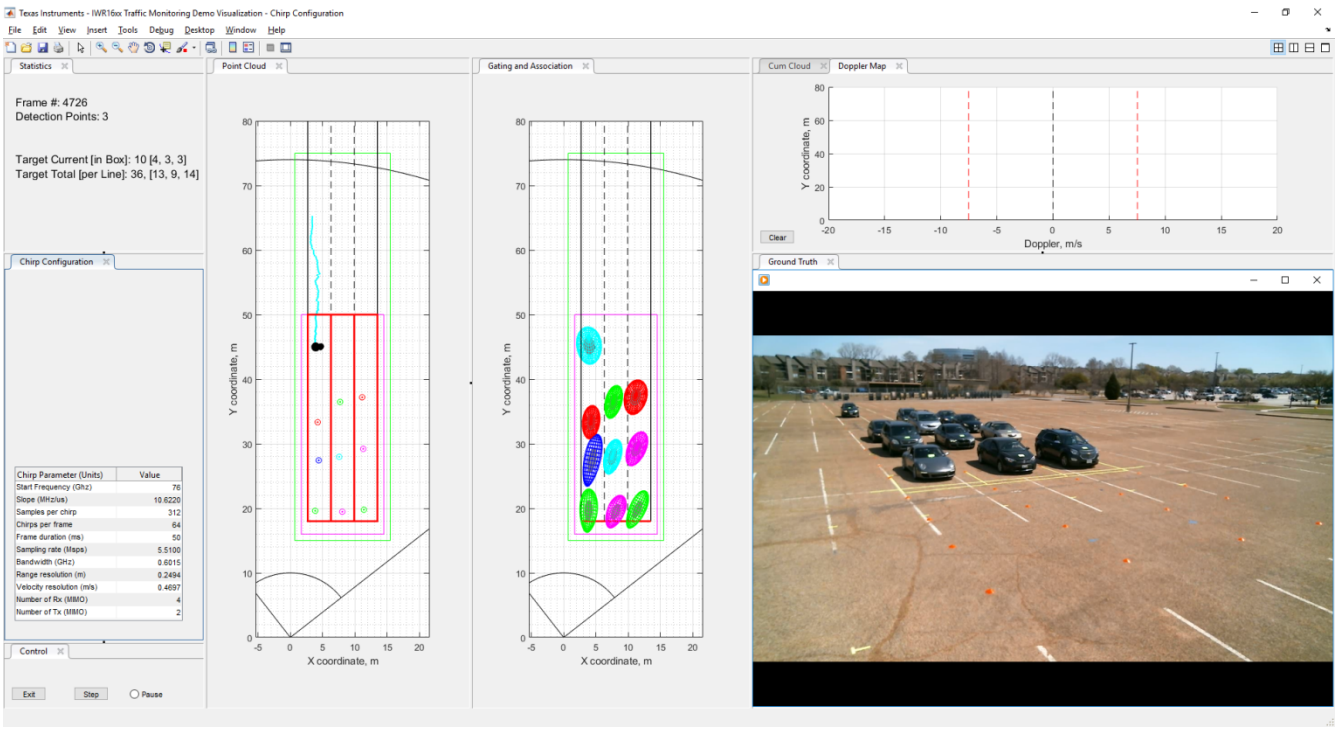


Figure 41. Vehicles at the Stop Line

The system as tested was configured with a static zone that starts at 50m distance (see purple square). Therefore, we shall be able to support 4 static cars per lane, totaling 12 cars. With few cars approaching and freeing, the tracker was configured to support maximum 20 cars at any given time.

7.6 System Behavior with Targets Moving Over Maximal Radial Velocity

The detection layer radial velocity measurements are always within the range [-Vmax, +Vmax], see black dots in the Doppler Map tab in Figure 42, limited by red dashed lines of Vmax = +/-7.5m/s. The capabilities of the tracking layer of deriving Cartesian velocities are demonstrated with a pink track in the Point Cloud Tab. The track was properly detected at about 70m of range. Reported radial velocities of multiple reflection points of +6.8m/s were immediately translated to correct target Cartesian velocities Vx (about 0m/s), and Vy (about -8m/s). Target velocity projected to radial direction is shown as pink circle

(also in the Doppler Map tab). Target with velocity over Vmax was correctly tracked through all the ranges; we observed target slow down as approaching the stop line. This illustrates tracker capability to disambiguate radial velocity measurements. Note that tracker isn't limited to any particular value of target velocity. In different test environments we observed proper velocity tracking for the targets moving at > 3x Vmax.

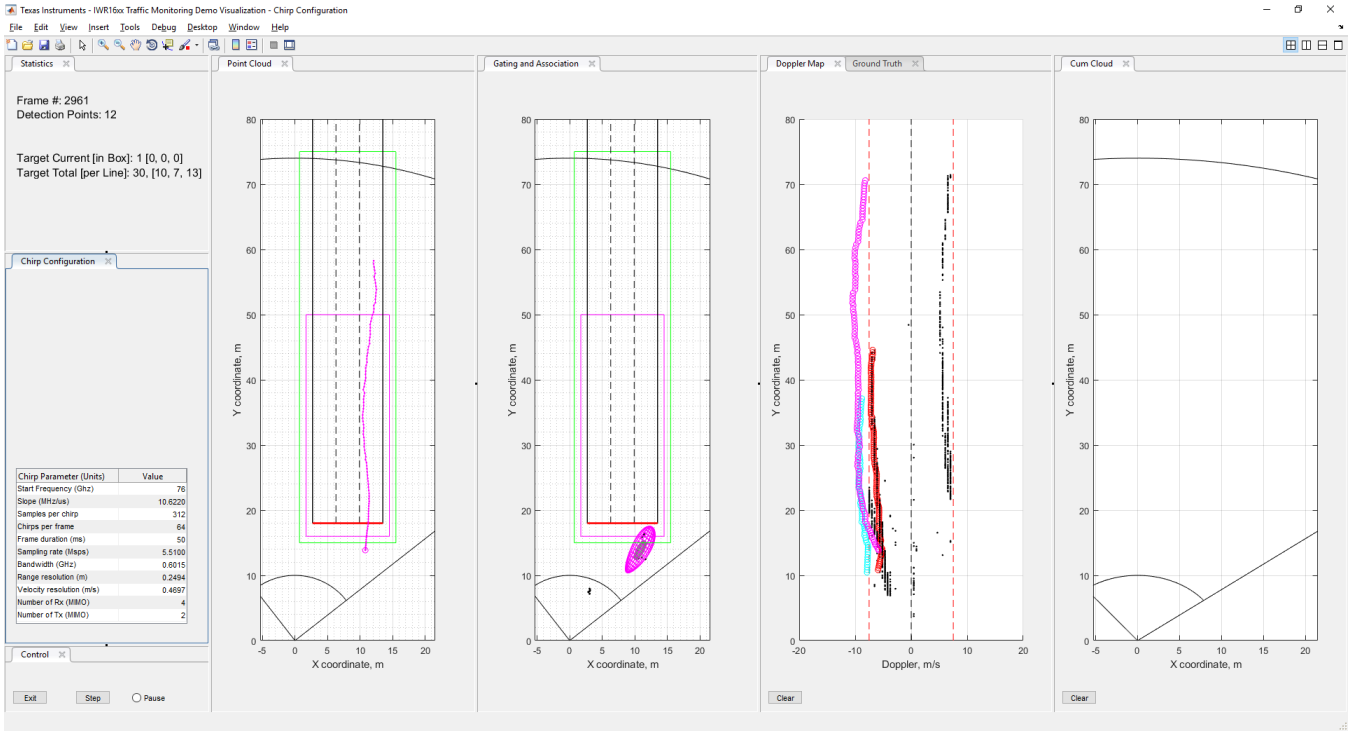


Figure 42. Vehicles Moving with Velocities over Vmax

8 Design Files

8.1 Schematics

To download the schematics, see the design files at TIDEP-0090.

8.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDEP-0090.

8.3 Altium Project

To download the Altium project files, see the design files at TIDEP-0090.

9 Software Files

To download the software files, see the design files at TIDEP-0090.

10 Related Documentation

1. Texas Instruments, *Robust traffic and intersection monitoring using millimeter wave sensors* , Marketing White Papers (SPYY002)
2. Texas Instruments, *IWR1642 Data Sheet* , Datasheet (SWRS212)
3. Texas Instruments, *IWR1642 Evaluation Module (IWR1642BOOST) Single-Chip mmWave Sensing Solution* , IWR1642BOOST User's Guide (SWRU521)
4. Texas Instruments, *IWR16xx/14xx Industrial Radar Family* , Technical Reference Manual (SWRU522)
5. Texas Instruments, *mmWave SDK* , Tools Folder
6. Martin Ester, Hans-Peter Kriegel, J&g Sander, Xiaowei Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise*, KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.

10.1 Trademarks

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from A Revision (April 2018) to B Revision **Page**

- Added Additionally, the IWR1843 and IWR6843 devices are also supported, giving the option for 3D object detection. Enabling 3D detection can improve robustness and accuracy of tracking with the ability to filter detected points based on elevation information. 1
- Added tool folder 1
- Added tool folder 1
- Deleted IWR1642 1
- Changed "76 GHz to 77 GHz" to "up to 4 GHz" 1
- Deleted IWR1642 with 1
- Deleted IWR1642 based on 77 GHz 3
- Added "The reference design provides test results and examples using the IWR1642BOOST EVM. The design has been further extended to enable traffic monitoring applications on IWR6843 and IWR1843 devices with the additional capability of 3D detection and tracking." 3
- Deleted 76 GHz to 81 3
- Added 4 GHz bandwidth 3
- Changed "IWR1642" to "mmWave sensors" 3
- Added "as an example of traffic monitoring capabilities using mmWave sensor. For details on enabling traffic monitoring applications on either the IWR6843 or IWR1843 devices refer to: Section 5.2 . The design considerations and enabled features in this document apply across all EVM platforms." 4
- Added For 2 Tx antenna TDM-MIMO, phase compensation needs to be done for the antenna samples received from the 2nd Tx antenna, ... 26
- Added is the Doppler index for the detected object, and N is the length of Doppler FFT. 26
- Changed "[INSTALL_PATH]\labs" to "[TOOLBOX_INSTALL_PATH]\labs\traffic_monitoring" 32
- Added new section "Enabling Support for IWR6843 and IWR1843 Devices" 32

Changes from Original (May 2017) to A Revision **Page**

- Changed Range from 120 to 180 m 1
- Added Link To End Equipment Page 1
- Updated to properly describe current CFAR implementation 6
- Deleted R4F section 13
- Changed to increase detail in angle estimation description. 15
- Removed Clustering and Tracking, Added High Level Processing Section 17
- Added Group Tracker Section 17
- Added Configuration Parameters Section. These parameters are updated to reflect TM 2.0 18
- Added ARM R4F Processing Time 23
- Removed in depth TLV description. 24
- Added Radial Velocity Extension Support 26
- Changed Section 6 changed to reflect new testing details and results. 32
- Added Section 7 40

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265