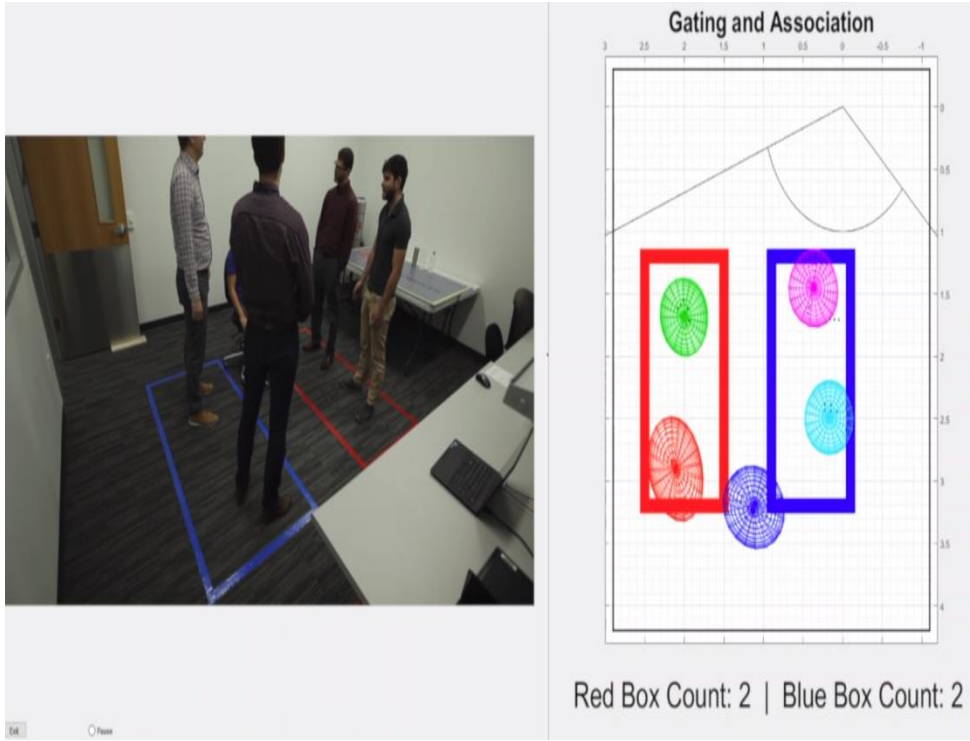
# TI mmWave Labs- People Counting Demo

## Setup and Customization Guide

# People Counting Demo Overview

- Demonstrates the use of TI mmWave Single chip Radar IWR1642 to count and track multiple people simultaneously.
- Detection and tracking algorithms run onboard the IWR1642 device to localize people and track their movement with a high degree of accuracy.
- Clutter removal enables static objects like chairs, tables to be removed
- Source code, CCS project and the Matlab GUI are part of this package
- Demo needs to be set up & customized for the area of interest.



# Demo Setup and Customization

There are 2 major steps involved to get to a desired people counting demo:

## EVM installation

The EVM needs to be installed at a proper height and with an elevation tilt to maximize the energy reflection from the objects and hence to get to a rich point cloud. This means directing the majority of transmitted energy towards the area of interest for better reflected energy reception. After placing the EVM, some Scenery Parameters will have to be tuned to reflect the location and position of the EVM in its environment.

## GUI and Tracking Parameter Customization

Demo customization is done in **GUI matlab** code for setting up the demo visualization

The group tracker needs to be configured for various parameters in task_app.c file under the project pplcount_16xx_mss, to work on the input point cloud data for localization and tracking via. These parameters include

**Scenery Parameters**

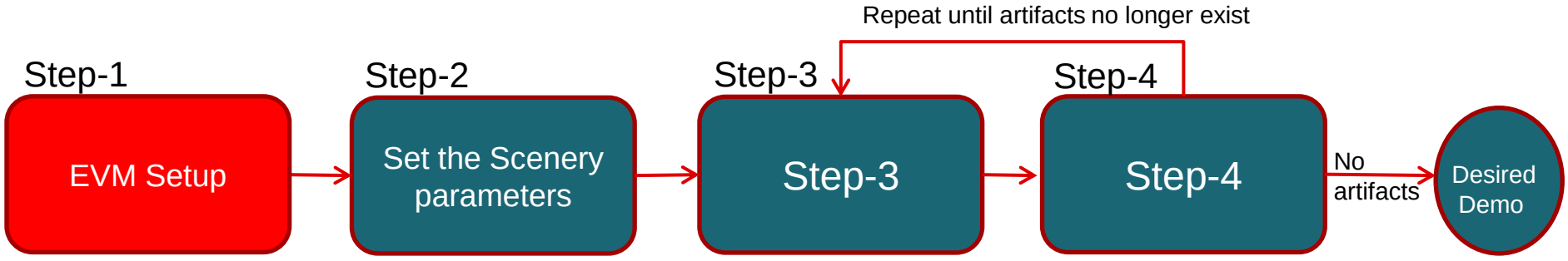
**Allocation Parameters**

**State Transition Parameters**

and the **d**emo needs to be recompiled & built after any change to the above tracking parameters

.

# Steps for Demo setup



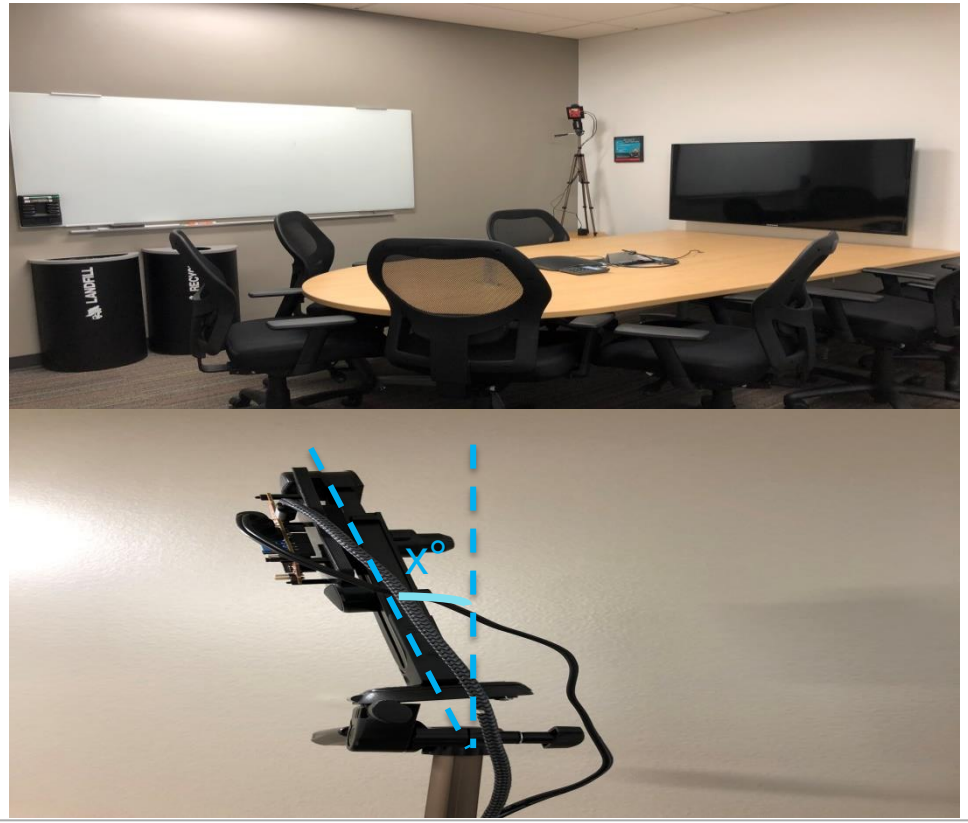

EVM setup to get to a better detection resulting in rich point cloud

Change scenery parameters for tracker according to area of interest

Run the Demo, observe the artifacts, if any

Based on the artifacts seen, further customize the tracker using state transition and allocation parameters

# EVM Installation




- Height:

  EVM needs to be installed at least 2-2.5mtrs above ground

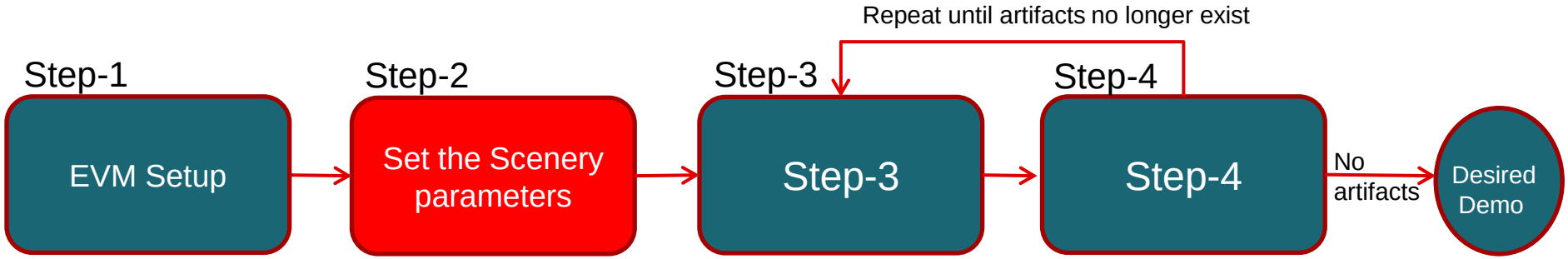
- Tilt in Elevation

  EVM needs to be tilted as shown in the picture with a angle x deg where 10 < x < 45 deg.

  X=10 deg works better for conference/meeting rooms

The elevation tilt helps focus the transmitted energy towards the area of interest and achieve better reflection which results in a good detection thus a rich point cloud.

# Steps for Demo setup



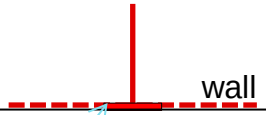
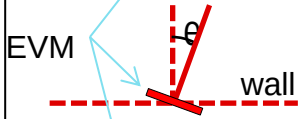
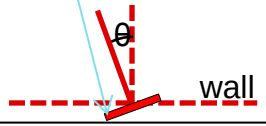

EVM setup to get to a better detection resulting in rich point cloud

Change scenery parameters for tracker according to area of Interest

Run the Demo, Observe the artifacts, if any

Based on the artifacts seen further customize the tracker using State transition and allocation parameters

# Set the Scenery Parameters - Chirp

| Top view Depiction | Azimuth Angle Tilt | .cfg file convention |
|---|---|---|
|  | 0 | 90 |
|  | +20 | 70 |
|  | -20 | 110 |

Note: EVM tilt to the right is +ve & to the left is -ve

- The EVM setup-azimuth angle shift needs to be input to the Radar Sensor through the .cfg file
- The API
  - trackingCfg 1 2 250 20 200 50 80

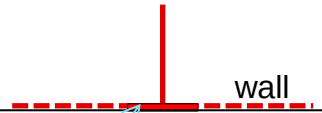
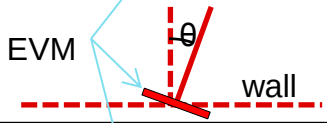
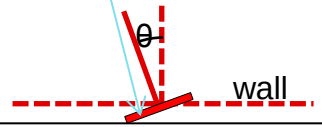
The above highlighted(blue) values needs to be changed accordingly

Example: for a Azimuth angle tilt of 0deg which corresponds to value of 90

trackingCfg 1 2 250 20 200 50 90

# Set the Scenery Parameters - GUI

The azimuth angle shift of the EVM needs to be communicated to the GUI. This helps to align the boxes in the 2 box type of demo to the EVM appropriately

| Top view Depiction | Angle of Installation |
|---|---|
|  | 0 |
|  | +20 |
|  | -20 |

The “scene.azimuthTilt“ parameter in the GUI needs to be configured as below

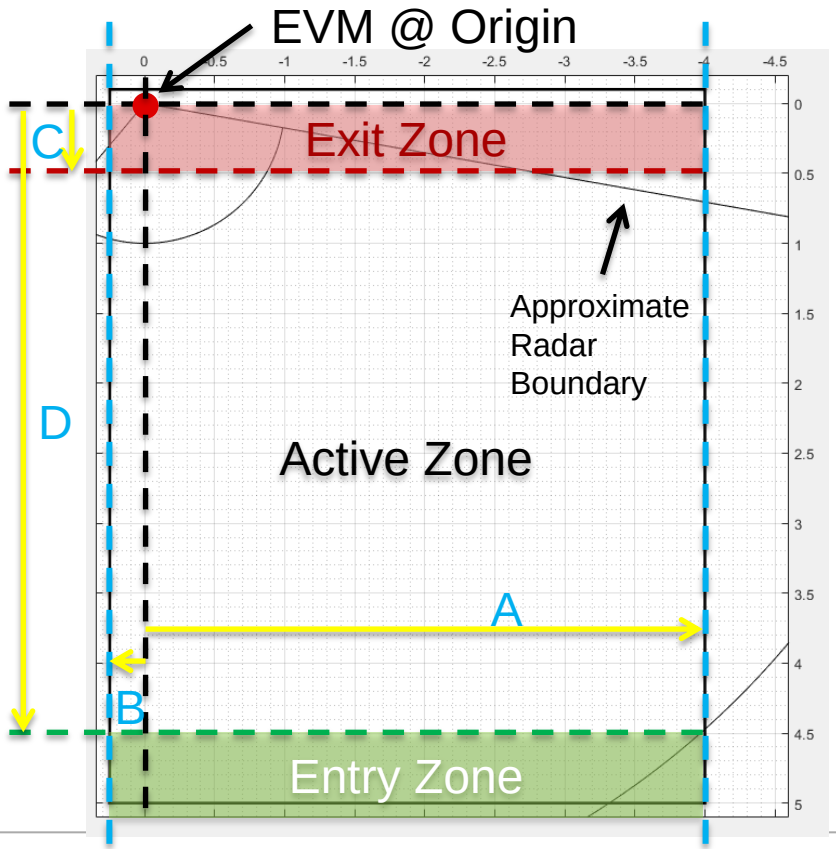
“scene.azimuthTilt = (+/-Theta)*pi/180”

For Theta=20 deg,

scene.azimuthTilt = (20)*pi/180

Note: EVM tilt to the right is +ve & to the left is -ve

# Set the Scenery Parameters – Active Zone




- After Setting up the EVM Please measure the distance A,B,C and D for the area of Interest.
- These scenery parameters would be input at two places
  - Matlab GUI
  - appSceneryParamTable[2]={leftWall,rightWall,lowerEntrance,upperEntrance}

Where

A-leftWall,
B-rightWall
C-lowerEntrance
D-upperEntrance

# Set the Scenery Parameters - GUI

**GUI Customizations:**

The GUI must be customized to represent the active zone described in the previous slide. Below are the parameters that control the Active zone.

1. Demo environment –Scenery parameters
    1. wall.left = -1*str2num(A);
    2. wall.right = str2num(B);
    3. wall.front = str2num(C);
    4. wall.behind = -1*str2num(D);

For A,B,C,D refer to previous slide

# Set the Scenery Parameters – Group Tracker

## 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.

Refer to next slide for a visual description of these parameters. These should be the same as the GUI Parameters.

appSceneryParamTable[2]={leftWall,rightWall,lowerEntrance,upperEntrance}

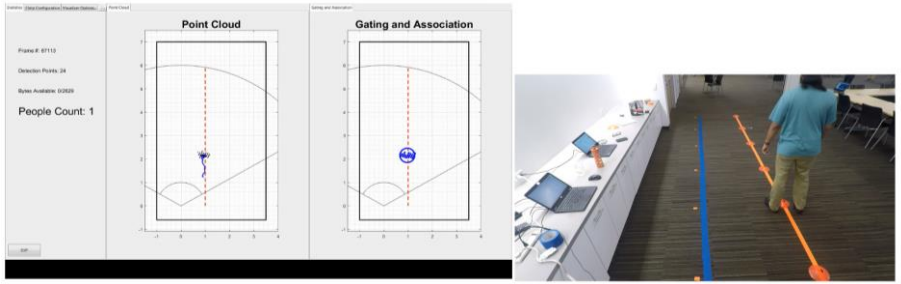
| Parameter | Default | | Description |
|---|---|---|---|
| leftWall | -1.5 | m | Position of the left wall, in meters, set to -100 if no wall. Points behind the wall will be ignored |
| rightWall | 1.5 | m | Position of the right wall, in meters, set to 100 if no wall. Points behind the wall will be ignored |
| lowerEntrance | 1 | m | Entrance area lower boundary, in meters; set to 0 if not defined. |
| upperEntrance | 4.5 | m | Entrance area lower boundary, in meters; set to 100 if not defined. |

# Set the Scenery Parameters – Further GUI Customization

The Demo can be set-up with or without subzones that count occupants:

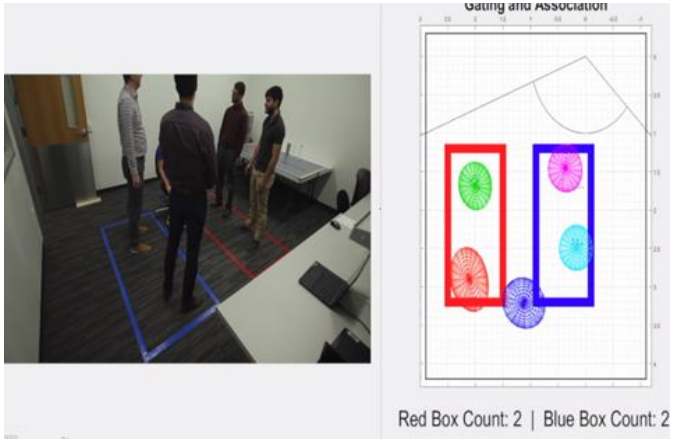
**No Subzones:**

For such a visualization set the parameter,
Scene Run choice = UI_Nobox;




**Subzones:**

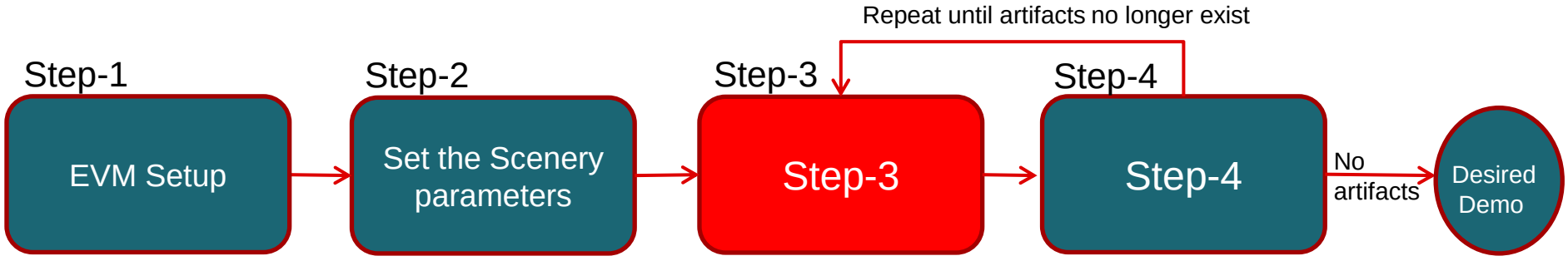
For such a visualization, set the parameter:
Scene Run choice = 2Box;



# Tracking Parameter Customization

- The tracking task parameters are set in task_app.c
- Find these starting at line 115
- The program must be recompiled after you make changes

# Steps for Demo setup




EVM setup to get to a better detection resulting in rich point cloud

Change scenery parameters for tracker according to area of Interest

Run the Demo, Observe the artifacts, if any

Based on the artifacts seen further customize the tracker using State transition and allocation parameters

# Tracking Parameter Customization

Run the demo once with the test scenarios in the area of interest.
Observe the performance and note down the artifacts seen.

# Tracking Parameter Customization

## 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 Points Threshold members, and pass the minimal velocity and SNR thresholds to start tracking.

appAllocationParamTable[2]={SNRThreshold,VelocityThreshold,Pointsthreshold,MaxDistanceThre,MaxVelThre}

| Parameter | Default | | Description |
|---|---|---|---|
| SNR Threshold | 250 | - | Minimum total SNR for the allocation set, linear sum of power ratios |
| Velocity Threshold | 0.1 | m/s | Minimum radial velocity of the allocation set centroid |
| Points Threshold | 5 | - | Minimum number of points in the allocation set |
| maxDistanceThre | 1 | $m^2$ | Maximum squared distance between candidate and centroid to be part of the allocation set |
| maxVelThre | 2 | m/s | Maximum velocity difference between candidate and centroid to be part of the allocation set |

# Tracking Parameter Customization

## 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 .

Once in **FREE** state, the transition to **DETECT** state is made by the allocation decision.
Once in **DETECT** state, 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.

Once in **ACTIVE** state, the handling of the MISS (no points associated) is as follow:

- If the target is in the “static zone” then assumed to be removed by static clutter removal and the miss count is incremented and static2free threshold is used to “extend the life expectation” of the static targets. The object continues to exist till the static2free number of frames-time

- If the target is outside the static zone, then the target is considered exiting. In this case, exit2free threshold is used to quickly free the exiting targets.

- If target is in the “static zone”, but has non-zero motion in radial projection hence not enough points, active2free threshold is used to determine when to free the target.

# Tracking Parameter Customization

## State Transition Parameters

appStateParamTable[2]={det2activeThre,det2freeThre,active2freeThre,static2freeThre,exit2freeThre}

| Parameter | Default | | Description |
|---|---|---|---|
| **det2activeThre** | **10** | **-** | In DETECT state; how many consecutive HIT events needed to transition to ACTIVE state |
| **det2freeThre** | 5 | - | In DETECT state; how many consecutive MISS events needed to transition to FREE state |
| **active2freeThre** | 10 | - | In ACTIVE state and NORMAL condition; how many consecutive MISS events needed to transition to FREE state |
| **static2freeThre** | 100 | - | In ACTIVE state and STATIC condition; how many consecutive MISS events needed to transition to FREE state |
| **exit2freeThre** | 5 | - | In ACTIVE state and EXIT condition; how many consecutive MISS events needed to transition to FREE state |

# Tracking Parameter Customization

## Gating Parameters

Gating parameters set is used in association process to provide a boundary for the points that can be associated with a given track. These parameters are target-specific.

appStateParamTable[2]={Volume,{LengthLimit,WidthLimit,VelocityLimit}}

| Parameter | Default | | Description |
|---|---|---|---|
| **Volume** | **4** | | **Gating volume** |
| **LengthLimit** | 3 | m | Gating Limit in length |
| **WidthLimit** | 2 | m | Gating Limit in width |
| **VelocityLimit** | 0 | m/s | Gating Limit in velocity (0 – no limit) |

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 , we could want to reach +/- 0.45m in range ($a = 0.9$), +/- 3 degree in azimuth ($b = 6\pi/180$), and +/- 5.18m/s in radial velocity ($c = 10.36$), resulting in volume about 4.

In addition to setting the volume of the gating ellipsoid, the limits can be imposed to protect ellipsoid from overstretching. The limits are the function of the geometry and motion of the expected targets.

For example, setting WidthLimit to 8m will not allow the gating function to stretch beyond 8m in width

# Tracking Parameter Customization

## Measurement Standard Deviation Parameters

This set of parameters is used to estimate standard deviation of the reflection point measurements. These rarely need to be changed.

appVariationParamTable[2]={LengthStd,WidthStd,DopplerStd}

| Parameter | Default | | Description |
|---|---|---|---|
| LengthStd | 1/3.46 | m | Expected standard deviation of measurements in target length dimension |
| WidthStd | 1/3.46 | 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 can be computed as below.

$$\sigma = \frac{b - a}{\sqrt{12}}$$

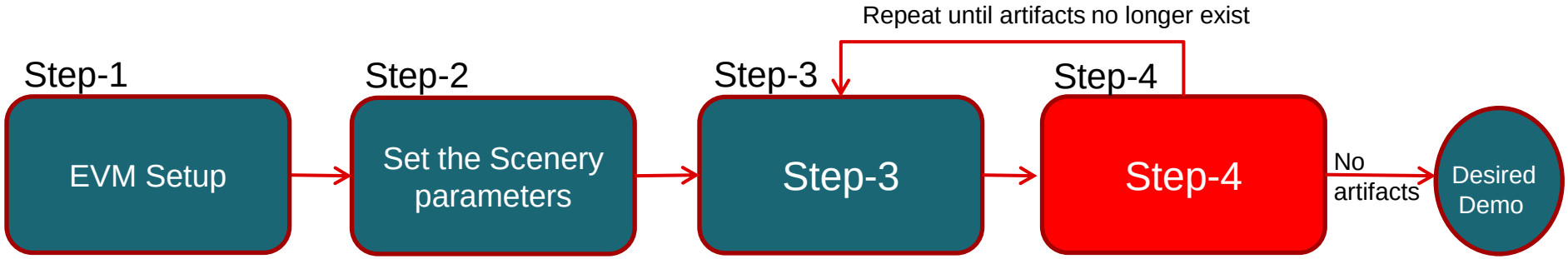
For example, for the targets of 1m wide, standard deviation can be configured as $\frac{1}{\sqrt{12}}$.

# Tracking Parameter Customization

Use the allocation and state transition parameters to mitigate the artifacts seen. The table below lists certain common Artifacts observed and the parameter that could be changed to achieve a satisfactory Demo. Though the list is not robust, but should give a direction of approach towards a solution

| Artifact | Parameter /Suggestion |
|---|---|
| Few or no detected points in the point cloud. | Check by tilting the EVM more in Elevation and observe if the detected points increase |
| Target generating detected points, but not becoming active until it is close (~2.5 meters) to the radar. | Decrease the **det2activeThre.** |
| Multipath reflections | Increase **Points Threshold**<br>Increase **SNR Threshold** |

# Steps for Demo setup



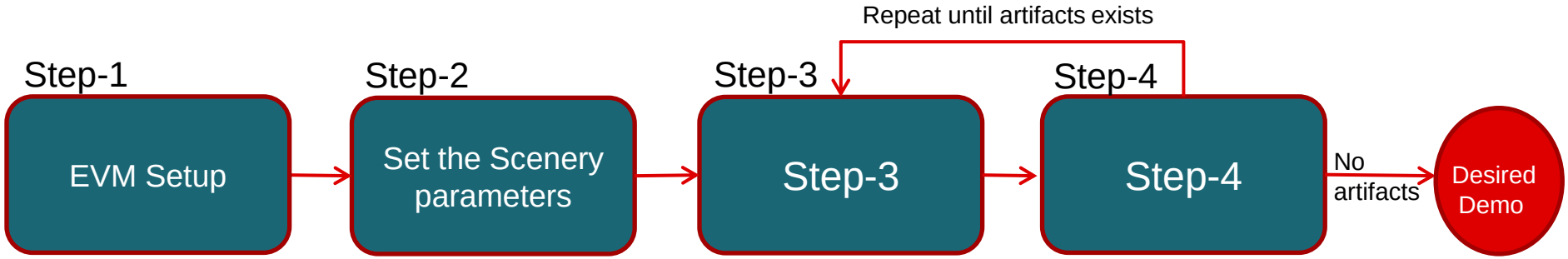

EVM setup to get to a better detection resulting in rich point cloud

Change scenery parameters for tracker according to area of Interest

Run the Demo, Observe the artifacts, if any

Based on the artifacts seen further customize the tracker using State transition and allocation parameters

# Steps for Demo setup




EVM setup to get to a better detection resulting in rich point cloud

Change scenery parameters for tracker according to area of Interest

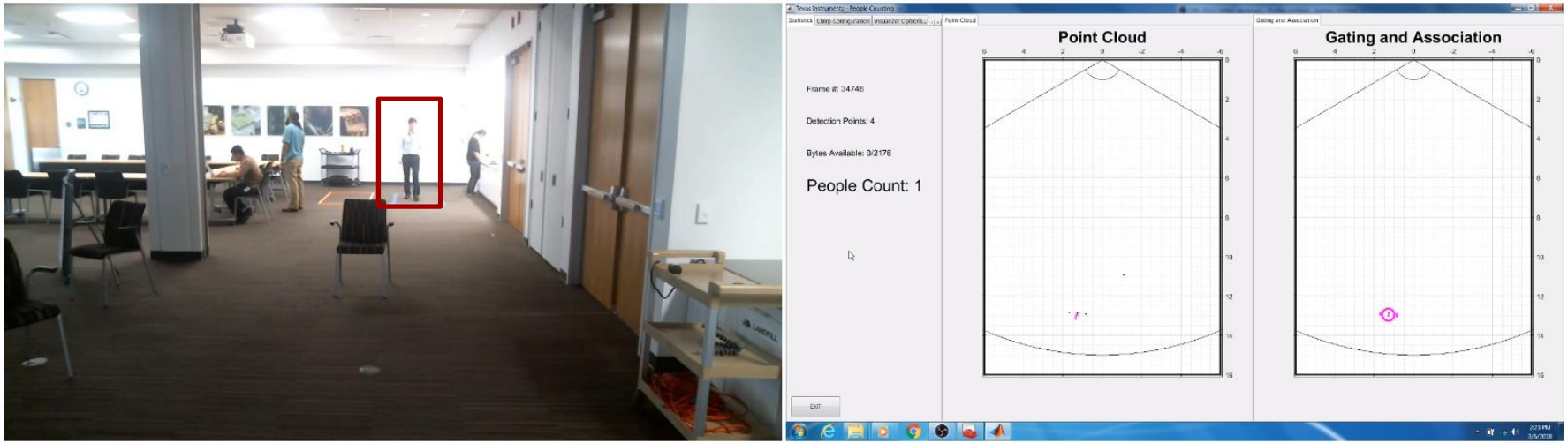
Run the Demo, Observe the artifacts, if any

Based on the artifacts seen further customize the tracker using State transition and allocation parameters

# Tuning Example – Increase Range to 12 m

- Key Parameters:
  - SNR Threshold – decrease from default
  - Det2active Threshold – decrease from default
  - Points threshold – consider lowering if necessary
- Explanation – at distance, a target will have lower SNR, less detected points, and be less likely to be detected in each frame. By lowering SNR, the tracker can classify less powerful signals as objects. By lowering points threshold, the tracker can classify less detected points as human. By lowering the Det2Active threshold, the tracker will classify a detected cluster as human faster, meaning an approaching person will be detected farther away
- Key Tradeoff – If these parameters are set too low, false detection is likely

# Tuning Example – Increase Range to 12 m

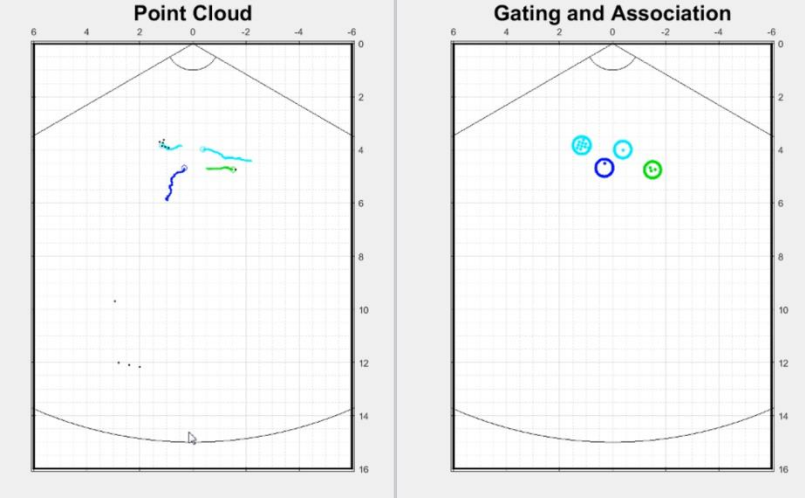



Target is Detected at 13 meters with these configuration changes.

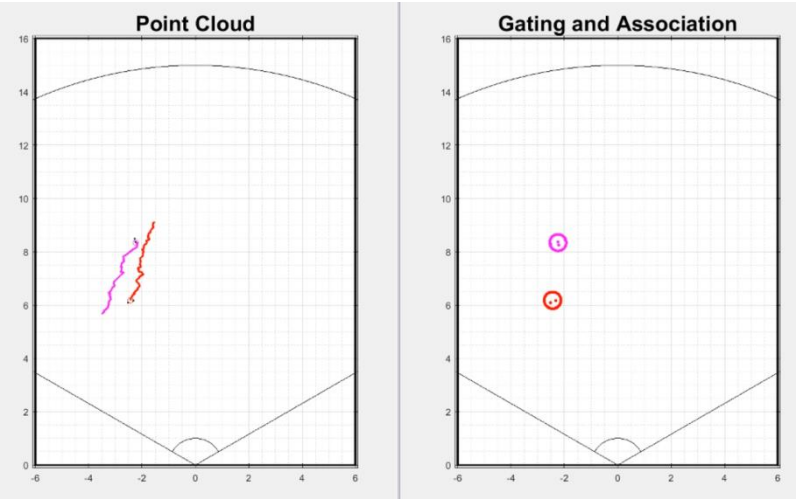
# Removing Issues when two People cross paths

- When two people cross paths, the tracker will sometimes confuse them or create ghosts
- Key Parameter
  - MaxDistance Threshold – change to 0.5 from 1
- Explanation – when two people pass each other, the detected points that make up there centroid can be confused by the tracker. When these points are confused, the centroid can be lost, or transferred to a different target
- MaxDistance Threshold determines the maximum distance a point can be from the centroid and still be allocated to it. By lowering this, the tracker cannot confuse the allocated points of two tracked people when they cross paths.
- Key Tradeoff – Large objects may appear as multiple objects

# Removing Issues when two People cross paths




Above you can see that the tracker spawned two ghosts when the targets crossed.




After the fix, you can see from red and blues path that the tracker kept them separate when crossing.

# Example Chirp Configurations

| Parameter | 5m | 12m |
|---|---|---|
| Starting frequency (GHz) | 77 | 77 |
| Maximum range, Rmax (m) | 6 | 15 |
| Range resolution (cm) | 4.9 | 12 |
| Maximum velocity (km/h) | 18.64 | 18.9 |
| Velocity resolution (km/h) | 0.297 | 0.297 |
| Maximum velocity (m/s) | 5.18 | 5.25 |
| Velocity resolution (m/s) | 0.08 | 0.08 |
| Idle time (us) | 30 | 30 |
| ADC valid start time (us) | 10 | 10 |
| Periodicity (ms) | 50 | 50 |
| Valid sweep bandwidth (MHz) | 3061.22 | 1250 |
| Ramp slope (MHz/us) | 60 | 25 |
| Sampling frequency (Msps) | 2.50 | 2.50 |
| Number of samples per chirp | 128 | 128 |
| Ramp end time | 62.20 | 62.20 |
| Chirp repetition period(us) | 184.0 | 184.0 |
| No of Samples per Chirp | 128 | 128 |
| No of Chirps | 128 | 128 |
| Active frame time(ms) | 23.55 | 23.55 |
| Radar cube size (KB) | 512 | 512 |

- There are two configuration given in the table. One for 6m and other for the 15m configuration.
- The same configurations can be used if the area of interest's –Rmax is 5 or 12m
- Customer could also modify the configuration using the system estimator tool on ti.com to suite the needs provided the Radar cube size is maintained at 512KB.

Note:
Its is highly recommended to design a chirp configuration to support slightly higher range than desired Rmax as the edge effect (windowing and other causes) several range bins near Rmax are excluded from detection

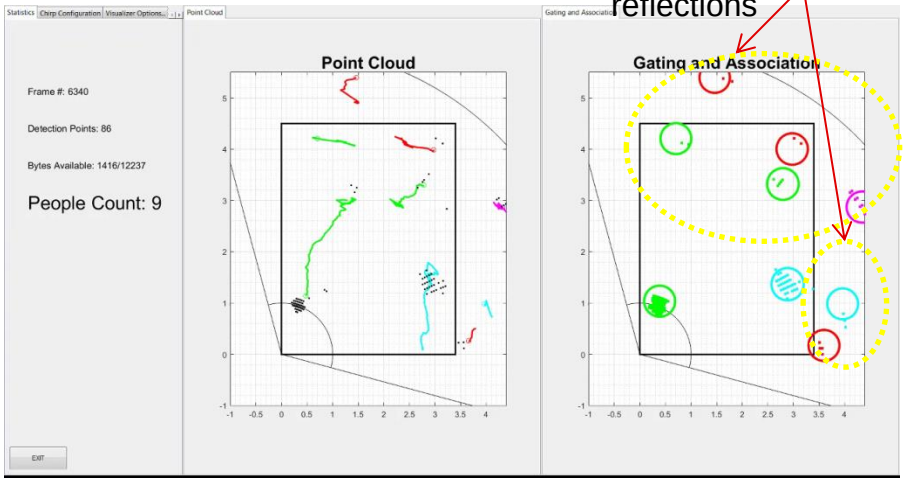
Example: If desired Rmax=12m, design a chirp configuration for 15mtrs

# Ghost Objects

- Sometimes, ghost objects appear where there are no people and can be counted as a person by the tracker
- Key Parameters to reduce Ghosts
  - appSceneryParamTable
  - SNR Threshold
  - Det2active Threshold
  - Points Threshold
- Explanation: Ghosts are mostly caused by multipath reflections off walls, ceiling, ground, or other clutter, especially metal.
  - To ignore multipath reflections off walls measure the distance of the radar to the walls. Then set the valuesin the appSceneryParamTable so that points that appear behind the walls are ignored.
  - Multipath reflections normally will have less points, lower SNR, and be temporary (ex an object such as a metal beam can cause ghost reflections when a person passes by the beam). Therefore a strategy to reduce ghosts is to increase SNR thresholds, Det2active threshold, and Points Threshold.
- Key Tradeoff – Setup time to measure the dimensions of the room.  If the SNR thresholds, Det2active threshold, and Points Threshold are set too high, real objects can be missed. Especially objects at further range.

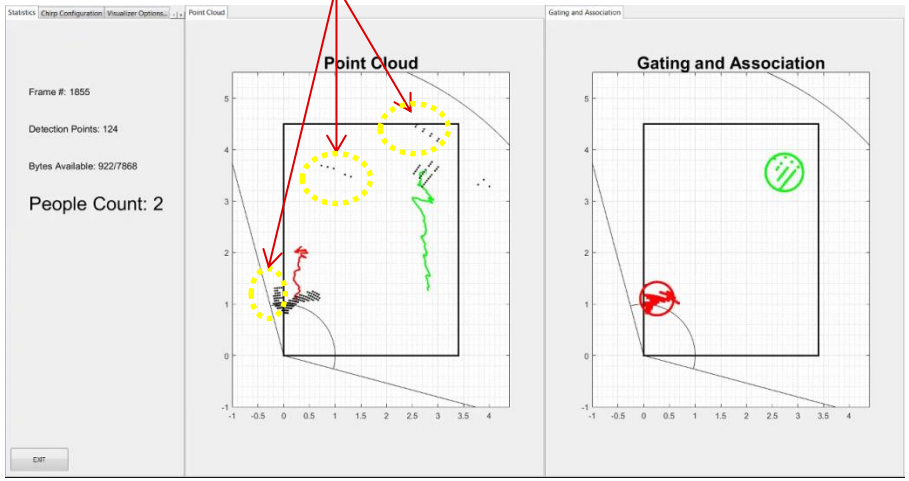
# Ghost Objects

Ghosts from multipath reflections




Even though there are only two people in the scene, ghost objects result from multipath reflections off the walls and other objects in the room.

Ignored multipath reflections




After the fix, you can see that even though there are multipath reflections they are not considered objects.