Group Tracker Parameter Tuning Guide for the 3D People Tracking Demo

Rev 1.2

Table of Contents

Date	Comment
January, 2021	Rev1.0: – Initial version.
July, 2022	Rev1.1: – Added Support for Azimuth Tilt
May, 2023	Rev 1.2: – Changed documentation from “People Counting” to “People Tracking” to more accurately reflect the capabilities of the example.

1 Introduction and Scope

The purpose of this document is to explain the group tracker parameters and give the user general guidelines on tuning these parameters. In section 2, we briefly introduce the group tracker algorithm while in section 3 we describe the tracker layer parameters that can be configured through the CLI interface in the people tracking application demo. In section 4, we describe some commonly encountered cases in which the user may need to tune the parameters to obtain optimal performance of the tracker. For detailed description of the group tracker algorithm and implementation details please refer to the Group tracker implementation guide [1].

2 Group Tracker Module

In the people tracking Lab, the tracking layer is responsible for localizing and tracking a clustered group of points. The input to the tracking layer is the point cloud from the detection layer and the output is a target list. Each measurement point from the detection layer contains the range, azimuth angle, elevation angle, radial velocity and SNR values. A collection of these measurement points is called a point cloud. The tracking layer is expected to take the point cloud data as an input, perform target localization, and report the results as a target list. The target list is a set of trackable objects with certain attributes such as the position, velocity, physical dimensions etc. These attributes can be further used by the higher level layers for visualization and scene interpretation or as an input to the classifier for object identification and classification decisions.

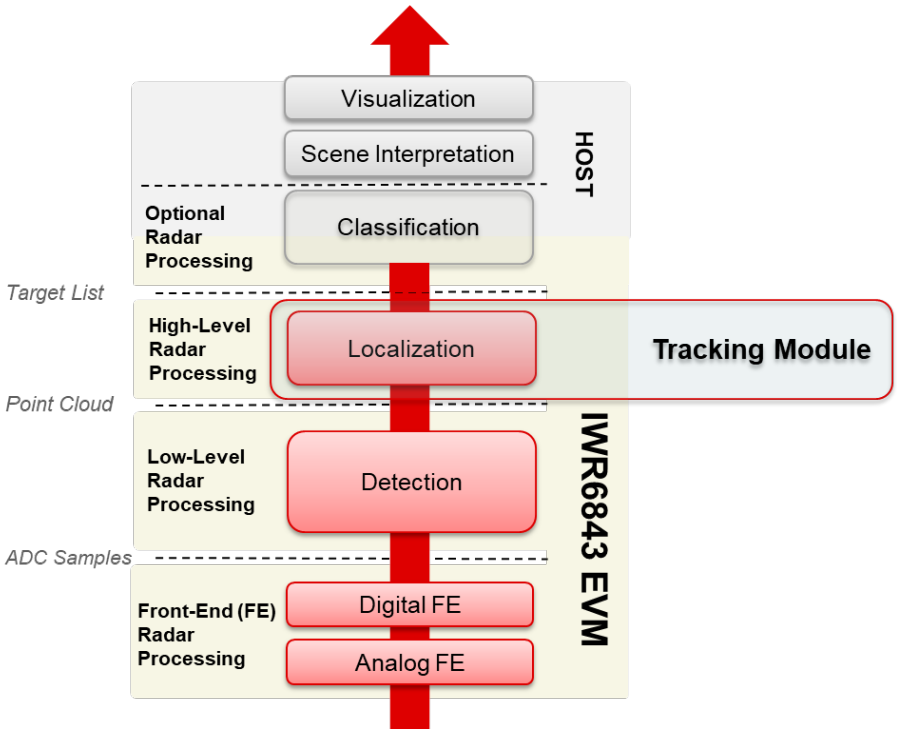

Figure 1. Tracking Module in the overall processing chain. The input is a point cloud from the detection layer while the output is a target list.

The basic building blocks of the Group tracker algorithm are briefly summarized below. For a detailed description please refer to Group tracker implementation guide [1]. The group tracker algorithm can be divided into the major functional blocks as shown below in Figure 2.

The input to the group tracker is a set of measurement points from the detection layer called the "point cloud". Each of the measurement point obtained from the detection layer includes in spherical co-ordinates the measured range, azimuth, elevation, and radial velocity of the point. The tracker motion model used is a 3D constant acceleration model characterized by a 9 element State vector S: $[x(n) \quad y(n) \quad z(n) \quad \dot{x}(n) \quad \dot{y}(n) \quad \dot{z}(n) \quad \ddot{x}(n) \quad \ddot{y}(n) \quad \ddot{z}(n)]$ in Cartesian space. It should be noted that the measurement vector is related to the state vector through a non-linear transformation (due to trigonometric operations required to convert from spherical to Cartesian coordinates). A variant of Kalman Filter called the Extended Kalman Filter (EKF) is used in the group tracker that linearizes the non-linear function using the derivative of the non-linear function around current state estimates. Please refer to the Group tracker implementation guide for more details on the algorithm [1] .

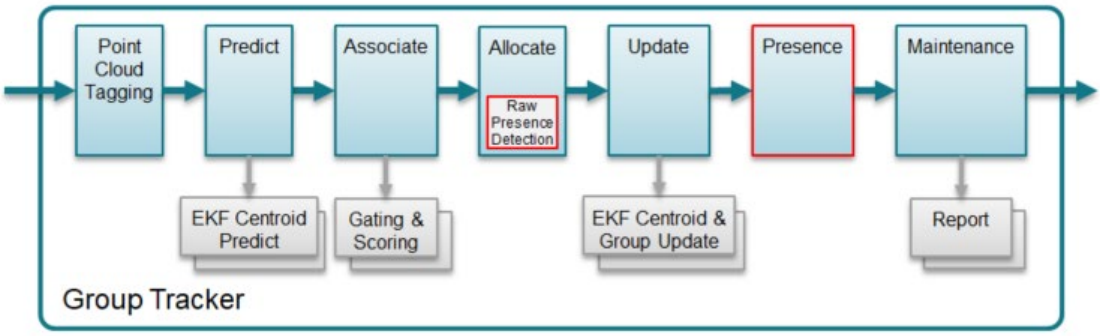

Figure 2. Block Diagram of Group Tracker

2.1 Point Cloud Tagging

As a first step, the measurement points that fall outside user configurable boundaries are appropriately tagged and ignored for further processing. These boundaries define the region of interest and typically correspond to the dimensions of the room or area of interest. Ignoring the points beyond the walls of a room also helps in eliminating ghost targets as some of these ghosts often fall outside the physical dimensions of the room.

2.2 Predict

The prediction step implements the standard prediction equations of the Kalman Filter. The input to the prediction is the position, velocity, and acceleration (i.e. state vector) in Cartesian co-ordinates and the corresponding state covariance matrix of a tracked object. Based on the prediction equations, the new position, velocity and acceleration *(*i.e. *a priori* state vector) and covariance matrices are estimated for each of the target.

2.3 Associate

In the Association step, a gate is created around each target. Mahalanobis distance is then computed between each measurement point and each of the targets/tracks. If the point falls within the gate of a track it is assigned a bidding score. Each point is given a score for each track that it qualifies to be gated with. Finally, the point is assigned to the track with which it has the highest bidding score. Please note that each point is only associated with a single track (based on the highest bidding score).

2.4 Allocate

All the points that don't get associated to an existing track in the association step go through an allocation step. Functionally, allocation is similar to clustering. In the allocation step, an arbitrary measurement point is first selected as a leading measurement and used for initialization of a cluster/candidate set. Each measurement is then checked if it falls within the user configurable velocity and distance thresholds of the cluster. If the measurement passes the check, it is added to the cluster and the centroid is recalculated. Once this is finished for all the measurement points, a few additional qualifying criteria are checked for the cluster. These tests include whether the cluster exceeds a minimum number of points, its mean velocity exceeds a velocity threshold, and if the points in the cluster have a strong enough combined SNR. If these qualifying criteria are passed, then a new tracking instance is created and the associated points in the cluster are used for initializing the tracker. The clusters which do not pass the qualifying criteria are ignored.

2.5 Update

As new measurements get associated with a track, the track centroid and the corresponding covariance matrix estimates are updated for each track based on these new associated measurements. In addition, the group track dispersion matrix and group covariance matrix is also computed in this step. The group covariance matrix is subsequently used in the Association step to form a gate around each track.

2.6 Presence

Presence detection determines if any target exists within the occupancy area. The algorithm combines "raw detection" and "known target in the area" indications. For "raw detection" indication, it re-uses the candidate set created by allocation process and checks it against the occupancy thresholds i.e. number of points in the set and their mean velocity exceeds the corresponding thresholds. For "known target in the area" the algorithm checks whether the known target measurement centroid is within the occupancy boxes.

2.7 Maintenance

Each track transitions through a life cycle of events. A track may be in the DETECT, ACTIVE or FREE state At the maintenance step, the state of the track might be changed or a track might be de-allocated.

3 Tracker Layer Configuration Parameters

The configuration parameters presented in this section are used to configure the tracker layer. They should be adjusted to match the use-cases based on the particular scenery and target characteristics. The users can set the configuration parameters using the configuration text file located in the corresponding people tracking demo (i.e., wall-mount or ceil mount). The configuration file formats are the same for both the 3D wall and ceiling mount scenarios, but will have different parameter values. Each line in the configuration file represents a command-line interface (CLI) message that configures several parameters. An example of tracker layer configuration for the ceiling mount is shown in Table 1.

	CLI Commands	
Tracking layer parameters	**staticBoundaryBox**	-3 3 -3 3 -0.5 3
	boundaryBox	-4 4 -4 4 -0.5 3
	sensorPosition	2.9 0 90
	gatingParam	3 2 2 3 4
	stateParam	3 3 6 20 3 1000
	allocationParam	20 20 0.05 20 1.5 20
	maxAcceleration	1 0.1 1
	trackingCfg	1 4 800 20 37 33 120 1
	presenceBoundaryBox	-4 4 -4 4 0.5 2.5

Table 1. Example of a tracker Configuration for the Ceiling Mounting position

The tracker layer parameters can be grouped into a few sets. A high-level description of the parameter sets and the corresponding CLI command are shown in Table 2. In the following subsections, we provide detailed information on each of the CLI commands.

	Parameter sets	CLI Commands	Description
1	**Scenery Parameters**	**boundaryBox** **staticBoundaryBox** **sensorPosition** **presenceBoundaryBox**	These define the dimensions of the physical space in which the tracker will operate. These also specify the radar sensor orientation and position. Any measurement points outside these boundary boxes will not be used by the tracker.
2	**Gating Parameters**	**gatingParam**	These determine the maximum volume and velocity of a tracked object and are used to associate measurement points with tracks that already exist. Points detected beyond the limits set by these parameters will not be included in the set of points that make up the tracked object.
3	**Allocation Parameters**	**allocationParam**	These are used to detect new tracks/people in the scene. When detected points are not associated with existing tracks, allocation parameters are used to cluster these remaining points and determine if that cluster qualifies as a person/target.

4	**State Parameters**	**stateParam**	The state transition parameters determine the state of a tracking instance. Any tracking instance can be in one of three states: FREE, DETECT, or ACTIVE.
5	**Max Accelaration Parameters**	**maxAcceleration**	These parameters determine the maximum acceleration in the lateral, longitudinal, and vertical directions.

Table 2. Configuration Parameters for the Group Tracker and their CLI commands

3.1 Scenery Parameters

These set of parameters allows the user to configure the dimensions of the physical space in which the tracker will operate. These also specify the radar sensor orientation and position. The current demo supports two sensor mount configurations (a) Wall mount (b) Ceiling mount as shown in Figure 3. It's convenient to define mathematical spaces in which the measured data is obtained, tracker operates and the output is visualized. We define the following spaces which are applicable to both wall and ceiling mount configurations

- World Space W : {X_w, Y_w, Z_w} is Cartesian with origin **O** at floor level
- Tracker Space T : {X_t, Y_t, Z_t} is Cartesian with origin at sensor
- Point Cloud Space P : {r, φ, θ , $\dot{r}$ } } is Spherical where the center of the antenna virtual array of the radar sensor is considered to be the origin. r is the radial distance, φ is the azimuth angle, θ the elevation angle and $\dot{r}$ is the radial velocity of a measured point with respect to the sensor axis.

The sensor mounting geometry and its relation to the different measurements spaces is shown in Figure 3. Please note that in the demo configurations

- Origin (O) is collocated with sensor projection to Z= 0 plane where Z = 0 corresponds to the scene floor.
- Sensor location S_w = {0, 0, H} is specified in the World Cartesian Space where H is the height of the sensor above the ground
- It's also useful to define a rotation matrix. If the sensor is tilted clockwise by an angle Θ about the axis X_w, then the rotation Matrix $\mathrm{R}_x(\Theta)$ is given as

$$\mathrm{R}_x(\Theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\Theta & sin\Theta \\ 0 & -\sin\Theta & cos\Theta \end{bmatrix}$$

- All configuration boxes defined by the user are also in the World Cartesian Space. In Figure 3, the boundary box is specified by the co-ordinate values { x_1, x_2, y_1, y_2, z_1, z_2,}

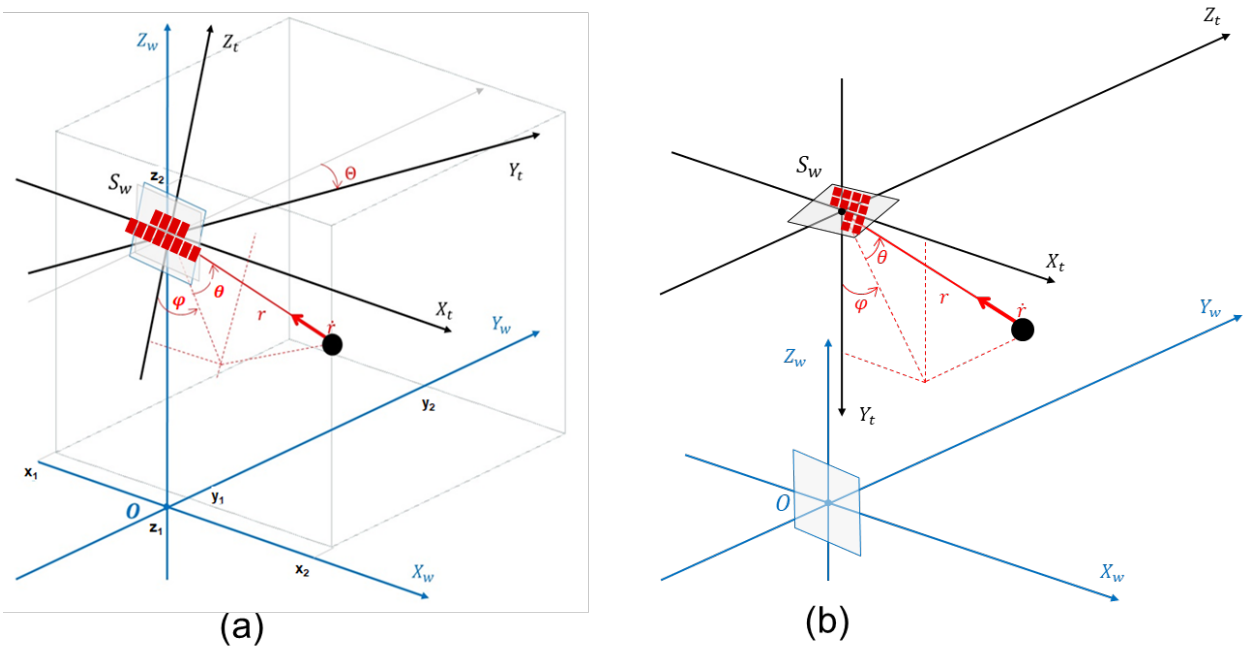

Figure 3. (a) Wall Mount Sensor (b) Ceil Mount Sensor

The scenery parameters are set through the following CLI commands that are supported in the current demo.

- **boundaryBox**
- **staticBoundaryBox**
- **presenceBoundaryBox**
- **sensorPosition**

These allow the user to configure the tracker with expected boundaries and areas of static behavior. Boxes are defined in meters in the world co-ordinates in Cartesian space (X,Y,Z). The sensor is assumed to be at (0,0,H) where H is the height of the sensor above the ground.

3.1.1 Boundary Box

The CLI command <boundaryBox> described in Table 3 allows the user to configure the physical dimensions of the space in which the tracker will operate. Any measurement points that fall outside this boundary will be discarded by the tracker. For example, boundary box can be configured to ignore the targets in the hallway outside the room, or to ignore potential ghost target that appear behind the room walls due to multipath reflections.

boundaryBox -4 4 -4 4 -0.5 3				
No	**Parameters**	**Example Value**	**Dim**	**Description**
1	**X-min** (float)	-4	m	Minimum horizontal distance with respect to the origin in the World co-ordinates
2	**X-max** (float)	4	m	Maximum horizontal distance with respect to the origin in the World co-ordinates
3	**Y-min** (float)	-4	m	Minimum vertical distance with respect to the origin in the World co-ordinates
4	**Y-max** (float)	4	m	Maximum vertical distance with respect to the origin in the World co-ordinates
5	**Z-min** (float)	-0.5	m	Minimum height with respect to the origin in the World co-ordinates. Note that Z = 0 corresponds to the ground plane. [In ceiling mount we see some valid reflections from below the ground hence we have a negative value]
6	**Z-max** (float)	3	m	Maximum height with respect to the origin in the World co-ordinates

Table 3. Boundary Box Configuration

3.1.2 Static Boundary Box

Static Boundary Box defines the zone in which targets are expected to become static for a long time. The static boundary box is configured through the CLI command <staticBoundaryBox> and its usage is described in Table 4.

staticBoundaryBox -3 3 -3 3 -0.5 3				
No	**Parameters**	**Default**	**Dim**	**Description**
1	**X-min** (float)	-3	m	Minimum horizontal distance with respect to the origin in the World co-ordinates
2	**X-max** (float)	3	m	Maximum horizontal distance with respect to the origin in the World co-ordinates
3	**Y-min** (float)	-3	m	Minimum vertical distance with respect to the origin in the World co-ordinates
4	**Y-max** (float)	3	m	Maximum vertical distance with respect to the origin in the World co-ordinates
5	**Z-min** (float)	-0.5	m	Minimum height with respect to the origin in the World co-ordinates. Note that Z = 0 corresponds to the ground plane
6	**Z-max** (float)	3	m	Maximum height with respect to the origin in the World co-ordinates

Table 4. Static Boundary Box Configuration

Static boundary box is used in the following way in the demo

- Static reflection points outside the static area are ignored

- When targets are within the static zone, and miss detection event occurs, the larger static threshold (static2freeThre**)** will apply for track de-allocation. When the targets are outside the area the smaller exit threshold (exit2freeThre) will be applied. Please refer to Section 3.4 for a description of these threshold values.
- Typically <staticBoundaryBox> is configured to be smaller than the boundary specified by the <boundaryBox> so that the exit condition can be triggered (i.e. smaller exit2freeThre applied to quickly delete the track). If they are set to be the same dimensions, then the tracker will not be able to quickly delete the tracks as it will not know whether the tracks have gone static or the target actually exited the boundary box.

3.1.3 Presence Boundary Box

The presence module in the tracker layer determines if any target exists within the occupancy area specified by the presence boundary box. The CLI command <presenceBoundaryBox> is used to configure the presence boundary box as shown in Table 5. The presence boundary box does not need to be smaller than the <boundaryBox>.

presenceBoundaryBox -3 3 -3 3 -0.5 3				
No	**Parameters**	**Default**	**Dim**	**Description**
1	**X-min** (float)	-3	m	Minimum horizontal distance with respect to the origin in the World co-ordinates
2	**X-max** (float)	3	m	Maximum horizontal distance with respect to the origin in the World co-ordinates
3	**Y-min** (float)	-3	m	Minimum vertical distance with respect to the origin in the World co-ordinates
4	**Y-max** (float)	3	m	Maximum vertical distance with respect to the origin in the World co-ordinates
5	**Z-min** (float)	-0.5	m	Minimum height with respect to the origin in the World co-ordinates. Note that Z = 0 corresponds to the ground plane
6	**Z-max** (float)	3	m	Maximum height with respect to the origin in the World co-ordinates

Table 5. Presence Boundary Box configuration

3.1.4 Sensor Position and Orientation

The CLI command <sensorPosition> can be used to specify the radar sensor orientation and position. **T**he sensor location S_w = {0, 0, sensorHeight} is specified in the World Cartesisan Space where sensorHeight is the height of the sensor above the ground plane i.e. Z = 0.

sensorPosition 2.9 0 90				
No	**Parameters**	**Example values**	**Dim**	**Description**
1	**sensorHeight** (float)	CM: 2.9 WM: 2.0	m	Height of the radar sensor above the ground plane.
2	**azimTilt** (float)	0	deg	The azimuth tilt of the sensor about the axis Z_w in Figure 3. A positive value indicates rotation from the positive X_w axis in the direction of the positive Y_w axis. Newly supported in Toolbox 4.12
3	**elevTilt** (float)	CM : 90 WM: 15	deg	The elevation tilt of the sensor about the axis X_w in Figure 3. A positive value indicates clockwise rotation with the tilt is towards the ground.

Table 6. Sensor Position Configuration. CM (Ceiling Mount). WM(Wall mount)

3.2 Gating Parameters

For each given track, a boundary is formed around the predicted centroid. The size of the boundary or gate should account for (a) Physical dimensions of the target (b) Expected target maneuver (c) Expected tracking error (d) Uncertainties in the detection layer. The boundary or gate around the track centroid is used to determine the points that can be associated with a given track.

Gating parameters are set through the CLI command <gatingParam>. These determine the maximum volume and velocity of a tracked object. Points detected beyond the limits set by these parameters will not be included in the set of points that make up the tracked object.

gatingParam 3 2 2 3 4				
No	**Parameters**	**Example values**	**Dim**	**Description**
1	**Gain** (float)	CM: 3 WM: 3	-	Gain of the gating function. It is set based on expected tracking errors and uncertainties of detection layer
2	**Limit-Width** (float)	CM: 2 WM: 2	m	Gating Limit in Width. It is set based on the physical dimensions and agility of the targets
3	**Limit-Depth** (float)	CM: 2 WM: 2	m	Gating Limit in Depth. It is set based on the physical dimensions and agility of the targets
4	**Limit-Height** (float)	CM: 3 WM: 3	m	Gating Limit in Height. It is set based on the physical dimensions and agility of the targets
5	**Limit-Velocity** (float)	CM: 4 WM: 4	m/s	Gating Limit in (radial) Velocity. It is set based on the motion of the targets

Table 7. Gating Parameters Configuration. CM (Ceiling Mount). WM(Wall mount)

- **Gain:** Gain is defined as the amount the track gating limits specified in the limit parameters below can grow to create the gating function. Each of the dimensions X (width), Y (depth), Z (height) and Velocity is multiplied by the *gain* to generate a gating function for the track. This value is set based on expected

tracking errors, expected target maneuver and uncertainties in the detection layer. We have empirically chose a value of 3 based on "three sigma rule" in statistics.

- **Limit-Width, Limit-Depth, Limit-Height, Limit-Velocity:** These values determine the limits of the gating function and are set based on physical dimensions and agility of the targets. Setting these too small will result in allocating multiple tracks for the single object, setting these too large will cause allocating single track for multiple objects. To set these limit values, the physical dimensions of the object that are to be tracked can be used as a good starting point and then the values further optimized based on test results.

3.3 Track Allocation Parameters

An attempt is made to associate each of the measurements (obtained from the point cloud in the detection layer) with existing tracking instances based on the gating parameters. The measurements that don't get associated with existing tracks are subjects for the allocation decision.

The allocation parameters determine if a **candidate point** can be clustered into an **allocation set.** To join the set, each point needs to be within <maxDistanceThre> and <maxVelThre> from the allocation sets centroid. Moreover, once the set is formed, it has to have more than <pointsThre> members and exceed the <maxVelThre> and SNR threshold values.

allocationParam 20 20 0.05 20 1.5 20

No	Parameters	Example values	Dim	Description
1	**snrThre** (float)	CM: 20 WM: 40	-	Minimum total SNR for the allocation set
2	**snrThreObscured** (float)	CM: 20 WM: 100	-	Minimum total SNR for the allocation set when obscured by another target
3	**velocityThre** (float)	CM: 0.05 WM: 0.1	m/s	Minimum radial velocity of the allocation set centroid
4	**pointsThre** (float)	CM: 20 WM: 20	-	Minimum number of points in the allocation set
5	**maxDistanceThre** (float)	CM: 1.5 WM: 0.5	m^2	Maximum squared distance between candidate centroid and centroid to be part of the allocation set
6	**maxVelThre** (float)	CM: 20 WM: 20	m/s	Maximum velocity difference between candidate and centroid to be part of the allocation set

Table 8. Allocation Parameter Configuration. CM (Ceiling Mount). WM(Wall mount)

- **snrThre:** This threshold is the minimum total SNR for the allocation-set. It's value is set to be roughly equal to the expected linear SNR value of a single detection point at 6m range multiplied by <pointsThre>. The total SNR for the allocation set is obtained by summing SNR values (in linear scale) of all the members of the allocation set. The total SNR is then compared with an internally derived SNR threshold value. The internally derived SNR threshold is a scaled version of <snrThre> where the scaling factor is range-dependent. Setting <snrThre> too low leads to long term errors.

- **snrThreObscured:** This threshold is the minimum total SNR for the allocation set when obscured by another target. The usage is similar to that of <snrThre> except that this value is used if the

allocation-set is obscured by another track. An allocation-set is declared obscured if it is behind AND has similar Doppler to an existing track. Usually <snrThreObscured> would be set larger than the <snrThre> to require stronger SNR for the obscured target to be allocated a track.

- **velocityThre:** Used to ignore allocation sets with very low radial velocity. On several occasions we might observe "spectral leakage" from strong static reflectors into adjacent Doppler bins. Setting <velocityThre> to a non-zero value prevents the static strong reflectors being allocated a track. However, setting a value too high can result in persons that are very stationary to not be allocated a track.
- **pointsThre:** Value is set based on the minimum expected number of points from a target.
- **maxDistanceThre:** This is a measure of how close (geometrically, in m^2) the point-candidate shall be to the centroid of the set-under-construction to join the set. It is similar to epsilon in DBSCAN algorithm. Note that the distance measure takes into account only the geometric difference (range, azimuth, and elevation) between the point and centroid and does not include the velocity and SNR of the measurement points.
- **maxVelThre:** Measure of how close (in radial velocity, m/s) the point-candidate shall be to the centroid of the set-under-construction to join the set. The parameter may be used to filter out obvious Doppler error.

3.4 State Transition Parameters

Any tracking instance can be in one of three states i.e. FREE, DETECT, or ACTIVE. The transition from one state to another is determined by the thresholds set in the state transition parameters. The state transition diagram and the associated thresholds are shown below

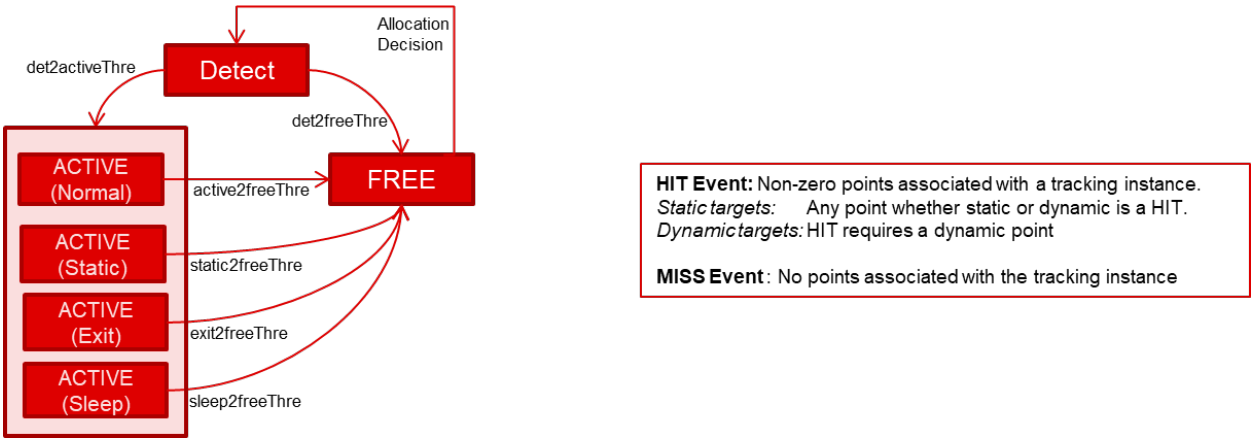

Figure 4. State Transition Diagram

A track on allocation is in the DETECT state. At each time instance, the track gets either a HIT or a MISS event (description of these events is in Table). After a sufficient number of HIT events, a track in DETECT state transitions to ACTIVE state. When a track is deleted, it is in the FREE state.

stateParam 3 3 6 20 3 1000

No	Parameters	Example values	Dim	Description
1	**det2actThre** (int)	CM: 3 WM: 3	Num of frames	In DETECT state; threshold for the number of continuous HIT events to transition from DETECT to ACTIVE state
2	**det2freeThre** (int)	CM: 3 WM: 3	Num of frames	In DETECT state; threshold for the number of continuous MISS events to transition from DETECT to FREE state
3	**active2freeThre** (int)	CM: 6 WM: 6	Num of frames	In ACTIVE state and NORMAL condition; threshold for the number of consecutive MISS events needed to transition from ACTIVE to FREE state
4	**static2freeThre** (int)	CM: 20 WM: 500	Num of frames	In ACTIVE state and STATIC condition; threshold for the number of continuous MISS events for a STATIC target in a static zone to transition from ACTIVE to FREE state
5	**exit2freeThre** (int)	CM: 3 WM: 5	Num of frames	In ACTIVE state and EXIT condition; threshold for the number of continuous MISS events for a target outside the static zone to transition from ACTIVE to FREE state. Determines the Maximum lifespan for the target outside the static box
6	**sleep2freeThre** (int)	CM: 1000 WM: 6000	Num of frames	Determines the Maximum lifespan for the target inside the static box

Table 9. State Transition Parameter Configuration. CM (Ceiling Mount). WM(Wall mount)

- **det2actThre:** a threshold for the number of continuous HITS to transition from DETECT to ACTIVE state
- **det2freeThre:** a threshold for the number of continuous misses to transition from DETECT to FREE state
- **active2freeThre:** Generic threshold for continuous misses in ACTIVE state when no special conditions (static2free/exit2free) apply. The corresponding counter is reset with either static or dynamic points associated
- **static2freeThre:** This threshold is applicable in a special condition when a track in ACTIVE state is declared to be STATIC inside a "static zone". The track is declared static if the velocity from the motion model is close to static (based on a threshold value). An assumption is made that the reason we don't have associated detection (i.e. MISS event) is because we removed them as "static clutter". In this case, we increment the MISS count, and use static2free threshold to "extend the life expectation" of the static targets.
- **exit2freeThre:** If an ACTIVE track is outside the "static zone", then the assumption is made that the reason we didn't get any associated points (i.e. MISS event) is that target is exiting. In this case, we use exit2free threshold to quickly free the exiting targets.
- **sleep2freeThre:** is the threshold for the maximum time target can be STATIC. There is separate counter that is reset only with dynamic point associated

3.5 Max Acceleration Parameters

These parameters specify the maximum amount that the acceleration can change in the lateral, longitudinal and vertical directions between frame periods. As an example, we are interested in modeling the position, velocity and acceleration of a target and we choose to use a 3D constant acceleration model. The assumption is that for each discrete time step the position and velocity of the object can potentially change however the acceleration would remain constant. However, in reality there is no guarantee that the targets acceleration will remain constant as there could be several unknown external un-modeled factors that can alter this assumption.

These max acceleration parameters try to model this deviation from our assumed motion model. Setting a large value will imply that we don't trust our motion model and hence the Prediction step is less reliable. Setting a low value implies that our motion model is very accurate and we do not expect much uncertainty.

maxAcceleration 1 0.1 1				
No	**Parameters**	**Example values**	**Dim**	**Description**
1	**Max Accel - X direction** (float)	CM: 1 WM: 0.1	m/s^2	Maximum amount that the target acceleration is expected to change in the X-direction betwee time-periods
2	**Max Accel - Y direction** (float)	CM: 0.1 WM: 0.1	m/s^2	Maximum amount that the target acceleration is expected to change in the Y- direction between time-periods
3	**Max Accel - Z direction** (float)	CM: 1 WM: 0.1	m/s^2	Maximum amount that the target acceleration is expected to change in the Z-direction in between time-periods

Table 10. Max Acceleration Parameter Configuration. CM (Ceiling Mount). WM(Wall mount)

3.6 Tracker Configuration Parameters

These parameters are used to enable the Tracker Module and determine the amount of memory to allocate based on maximum number of points (**maxNumPoints**) and tracks (**maxNumTracks).** It also configures the radial velocity parameters (initial velocity, velocity resolution, max velocity) and frame rate at which the tracker is to operate.

trackingCfg 1 4 800 20 37 33 120 1				
No	**Parameters**	**Example values**	**Dim**	**Description**

1	**enable** (int)	1	-	Set 1 to enable the group tracker
2	**IntialConfigParams** (int)	4	-	An index to the default internal tracker parameter structure array that initializes the tracker configurations for different use-cases. These internal tracker parameters will be used by default if the user does not set these through the corresponding CLI command
3	**maxNumPoints** (int)	800	-	Maximum number of Detection points per frame
4	**maxNumTracks** (int)	20	-	Maximum number of Targets to track at any given time
5	**maxRadialVelocity** (int)	37	(10*velocity in m/s)	Maximum Radial velocity. If the maximum radial velocity reported from the sensor is 5 m/s then <**maxRadialVelocity**> needs to be set to 50
6	**radialVelocityResolution** (int)	33	mm/sec	Radial velocity resolution in millimeter/sec that the configured chirp profile can provide
7	**deltaT** (int)	120	msec	Frame Rate. This should match the sensor chirp configuration
8	**boresightFilteringEnable** (int)	1	-	Set 1 to enable boresight filtering. This is used only in ceiling mount and ignored in the wall-mount use-case

Table 11. Tracker Configuration Parameters Configuration for ceiling mount

- **enable:** Set 1 to enable the tracker Module
- **IntialConfigParams:** An index to tracker parameter structure array that initializes the tracker configurations for different use-cases. These parameters are overwritten by the corresponding CLI command. For instance, the <stateParam> command will overwrite the state transition parameters populated by the initial configuration.
- **maxNumPoints:** Maximum number of measurement points per frame. The tracker Module allocates memory based on this parameter so it's important to choose a realistic number based on the use-case. This can be configured to a maximum of 1000 points depending on available memory.
- **maxNumTracks:** Maximum number of targets to track. The tracker Module allocates memory based on this parameter so it's important to choose a realistic number based on the use-case. This can be configured to a maximum of 200 tracks depending on available memory.
- **maxRadialVelocity:** Maximum radial velocity that sensor would report to the tracker module from the detection layer.
- **radialVelcoityResolution:** Radial Velocity resolution. Note that this value is specified in millimeter/sec
- **deltaT:** Frame time in millisecond

- **boresightFilteringEnable:** This flag is used only in ceiling-mount and ignored in the wall-mount use-case. The boresight angle is ± 6 degree at 2 meters. The filter starts at 2m from the device and extends through the floor.

4 Parameter and Performance Tuning

This section describes some commonly encountered scenarios in which in which the user may need to tune the configuration parameters discussed in Section 3 to obtain optimal performance of the tracker.

4.1 Too Few People Detected

There are multiple parameters that affect when a person can be positively identified and tracked. First, make sure that the chirp configuration that the device is configured to acquire data with extends to the range at which we want to detect people/targets and that the <boundaryBox> is set accordingly.

If there are reflection points coming from the person but the tracker is unable to allocate a track, then consider changing the track allocation parameters shown in Table 12. However, it should be kept in mind that lowering these parameters increases the chances of false detection.

No	Parameter Type	Field	Tuning Guide
1	Allocation Params	snrThre	Further the distance of the target from the radar, the lower the SNR of the detected points. Lowering the SNR threshold will lower the cumulative SNR that is required for an allocation set to be considered as a track.
2	Allocation Params	pointsThre	Fewer points are needed for an allocation set to be considered as a track if the pointsThre is lowered. Lowering this threshold might be beneficial for longer ranges as fewer points might be detected from a person at larger distances from the sensor
3	Allocation Params	velocityThre	Lowering the velocity threshold will allow a person making very tiny movements to still be tracked.

Table 12. Tuning Parameters if the number of count is less than expected

Another scenario in which too few people will be counted is when multiple people are close together which is discussed in section 4.3

4.2 Too Many People Detected (Ghosting)

There could be several reasons that the tracker is detecting too many people in the scene. One likely cause is multipath reflections (radar Energy reflected from a person being reflected again from a wall, ceiling, floor or some object) which will cause the tracker to produce a false detection. These false detections are called ghosts. Ghosts can be caused by multiple phenomena, but there are a few parameters that can be tuned to minimize or remove ghosting.

No	Parameter Type	Field	Tuning Guide
1	Scenery	boundaryBox	Start by properly setting the scenery parameters. Many ghosts caused by multipath reflections from the wall will appear outside of the detection area. These can be immediately ruled out if the scenery parameters are properly set.
2	Allocation	snrThre	Ghosts will usually have detection points with lower SNR

3	Allocation	pointsThre	Ghosts will usually have fewer detected points
4	State	det2actThre	det2active : Increase the amount of time the ghost has to exist before its state is changed to ACTIVE state. Useful if the Ghost only appears momentarily
5	State	det2FreeThre static2FreeThre acitive2FreeThre exit2FreeThre	det2free, static2free, actve2free, ext2free: Lowering these thresholds, the tracks will be freed faster

Table 13. Tuning Parameters if the number of people counted is larger than expected

Items # 2,3,4 in Table 13 can be used to reduce false detections. Ghosts will usually have fewer points with lower SNRs. Increasing the Points Threshold and SNR Threshold will stop the tracker from allocating these clusters as tracks. Increasing the <det2actThre> threshold will increase the amount of time the ghost has to exist before it is promoted to ACTIVE state. In the case where a ghost only appears momentarily, this can stop the tracker from tracking it.

If changing these parameters fails to stop ghosts from appearing, consider changing the parameters in item#5 i.e. <det2FreeThre>, <static2FreeThre>,<acitive2FreeThre>,< exit2FreeThre>. By lowering these thresholds, tracks will be freed faster, so the ghost will be tracked for a shorter period of time. However, lowering these may reduce the ability of the tracker to properly maintain a track on a real target, especially in a cluttered environment.

4.3 Resolving multiple people when close together

In some situations, the tracker may allocate one track to two or more people. This is likely to happen when people are near each other, and walking at the same pace in the same direction (a fairly common occurrence). Other times, the tracker may give one person multiple tracks.

To prevent these situations, consider changing the following parameters: All of these parameters will affect how points in the point cloud get distributed into different tracks.

No	Parameter Type	Field	Tuning Guide
1	Allocation	maxDistanceThre	The Allocation parameters will affect how tracks are created. Detected points are clustered, and each cluster can become a tracked person. Lowering these thresholds will force the point clusters that become tracked people to be smaller, increasing the differentiation between people, while raising them will allow the point clusters to be larger, decreasing differentiation between people.
2	Allocation	maxVelThre	
3	Gating	Gain	These parameters determine how close a measurement needs to be to be counted as part of an existing track. Lowering these values will help the tracker to separate the individual point clouds if multiple people will be walking near each other. However setting these values too low will result in allocating multiple tracks for a single person
4	Gating	Limit-Length	
5	Gating	Limit-Width	
6	Gating	Limit- Velocity	

Table 14 Tuning Parameters if unable to resolve multiple people

4.4 Tracks from stationary people (that were once moving) are disappearing

At times, it may happen that a tracker was successfully tracking a moving person but when the person stopped moving (e.g. sat down, standing very still) the track got de-allocated. This could be due to multiple reasons.

First, make sure that the static Boundary box is configured correctly i.e. the stationary target is within the static Boundary box. [Please refer to Section 3.1.2 for a description of the static boundary box].

The maximum lifespan for static targets inside the static box is determined by the state transition parameters <static2freeThre> and <Sleep2FreeThre> described in Section 3.4. Try increasing the values of these parameters which will help in keeping the track alive for a longer time duration.

If the particular use-case scenarios are likely to have people sitting very still or stationary then it is recommended to tune the detection layer parameters for increased motion sensitivity [Please see the detection layer tuning guide] [2].

No	Parameter Type	Field	Tuning Guide
1	Scenery	staticBoundaryBox	Make sure that the stationary target is within the static boundary boxes
2	State	Static2FreeThre Sleep2FreeThre	These thresholds are only applicable to static targets within the Static zone. Larger these values the longer the track will persist inside the static boundary box. If a static target disappears, then try increasing these values

Table 15. Tuning Parameters if the track associated with a stationary person is disappearing

4.5 Tracks from moving people are disappearing

Sometimes a track corresponding to a moving person might disappear. One likely cause for this is that the person/target is occluded by other objects in the scene and hence there won't be any reflection points detected from the target. The performance of the tracker is dependent on the continuity of the tracking process and occlusion of targets might cause a track to disassociate from its target. In these cases very little can be done. Most likely, these occlusions would lead to temporary short-time errors, and once the target is visible again, a track will be re-allocated.

Sometimes the track might get de-allocated if the reflected points from the target do not get associated with its track (i.e. they fall outside the gating function of the track). In these cases, increasing the values of the Gating Parameters can be considered. However, as mentioned in Section 4.3, setting the gating parameters too high can result in difficulty in resolving multiple close-by people.

A track from a moving person can also disappear, if the motion dynamics of the target violate the assumptions made in the tracker motion model. For instance, in the 3D constant acceleration model, we expect the acceleration of the target to be constant. If a moving target abruptly changes acceleration (e.g. makes a sharp turn), the motion model might not adequately capture this and can result in the target

track being lost. As discussed in Section 3.5, max acceleration parameters can be used to model these deviations from the assumed motion model. Setting these parameters to larger values will imply that our motion model has more uncertainty and the EKF will give comparatively less weightage to the Prediction step.

No	Parameter Type	Field	Tuning Guide
1	Max Acceleration Params	maxAcceleration	These max acceleration parameters try to model the deviation from our assumed 3D constant acceleration motion model. If a track from a moving target is disappearing then increasing this values can be considered. Setting a large value will imply that we don't trust our motion model and hence the Prediction step is less reliable.
2	Gating Params	Gain, Limit-depth, Limit-width, Limit-height,	These parameters determine how close a measurement needs to be to be counted as part of an existing track. If a track from a moving target is disappearing then increasing these values can be considered. However, keep in mind that setting the gating parameters too high can result in difficulty in resolving multiple close-by people.

Table 16. Tuning Parameters if the track associated with a moving person is disappearing

5 References

[1] "Group Tracker Algorithm and Implementation Guide, Texas Instruments."
[2] "3D People Counting Detection Layer Tuning Guide, Texas Instruments."