USER MANUAL

SmartTracker

Revision: *1*

Date: *2018-11-15*

Table of contents

History

Document revision	Date	Change log	Author
1	2018-10-31	Initial release	JK, CN

Documents

[1] User Manual Target-Viewer rev 13

Brief Manual Description

In this manual, specifically the *SmartTracker* is documented.
If you need further information about the *Target-Viewer* software, please read the *Target-Viewer* User Manual which you have received with the Target-Viewer setup package.

List of abbreviations

RADAR	Radio Detection And Ranging
RV target list	Contains target information such as radial velocity, range, azimuth angle etc.
object list	Contains information about tracked objects such as counters, object id, velocity, direction, distance etc.
ST	*SmartTracker*
ITL	*InnoSenT Tracker Library*
DLL	Dynamic Link Library
SO	Shared Object

Table 1: List of abbreviations

1. Introduction to SmartTracker

The *SmartTracker* software module is used for tracking objects like pedestrians, bicycles, cars etc.

In combination with an *InnoSenT iSYS* RADAR system, *SmartTracker* raises your security application to the next level.

Features:

- Multi-target tracking of maneuvering objects
- Suitable for multiple applications
- Robust towards temporal blocked line of sight
- 2+1 classification
- Artificial neural network for tracking and classification

Input of *SmartTracker* is the RV target list from a compatible *iSYS* RADAR system. It performs tracking and outputs an object list. This provides the opportunity to monitor the position, velocity and direction of these objects. Each object is classified and can be identified by its unique object ID.

In addition to the tracking process a classificator is integrated into the *SmartTracker*. The classificator is able to distinguish between pedestrians, vehicles and other objects.

The *SmartTracker* is included into the *InnoSenT Tracker Library* (ITL).
The ITL is part of the *InnoSenT Target-Viewer*. Besides this, the ITL is also capable of being integrated into other applications on various Windows and Linux distributions.

The *Target-Viewer* is a graphical user interface for visualization of targets, received by the *iSYS* RADAR sensor, and tracks, processed by the *SmartTracker*.

Sensor Compatibility

The *SmartTracker* is compatible with the following sensors:

- iSYS-5011
- iSYS-5021

Hardware Requirements

PARAMETER	CONDITION	MIN	TYP	MAX	UNIT
memory		182		200	kB
update rate	ARMv7 IMX6Q (790MHz)		19	22	ms
update rate	Intel Core i7-6820HQ (2.70GHz)		1		ms

2. SmartTracker description

2.1. How does the process flow look like?

The compatible sensor generates a RV target list, which consist of the radial velocity, range, angle and radar cross section info for all targets measured. The *SmartTracker* needs this target list as input, performs a tracking of those targets over multiple frames and generates itself an object list. The object list which can be received by the tracker consists of information about current position, velocity, direction and object states such as track quality. Each object has a unique object ID for identification.

The *SmartTracker* is integrated in the *InnoSenT Tracker Library* (ITL). In the ITL all necessary memory for tracking will be allocated by calling the Init-Function. Setting the default values proves that the ST performs with best possible performance for the configured sensor. When receiving targets from the sensor the tracker will be executed with RV target list as input argument. The object list or track list can be received after tracker execution.

Unloading the ITL will free all the allocated memory.

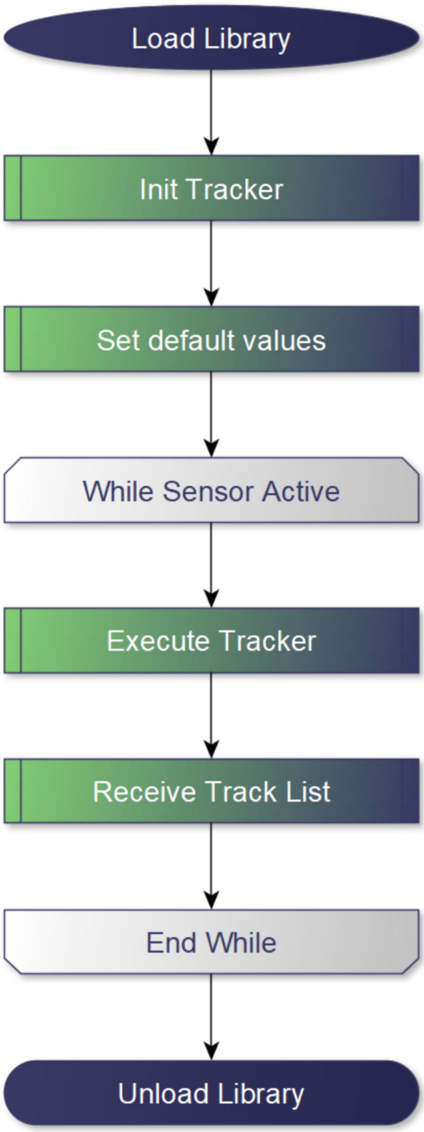

Figure 1: SmartTracker process flow

2.2. How to implement the SmartTracker in own applications

The *SmartTracker* is integrated in the *InnoSenT Tracker Library*. The ITL can be loaded as any other DLL in Windows or Shared Object in Linux. For more detailled information the example project shows how to integrate the ITL in your project. The example project is used for iSYS-5021 configured with Ethernet target list output. For this reason in addition to the ITL the *EthernetAPI* is needed for running this example.

All necessary libraries, source and header files can be found in the example project folder. The readme explains how to build and run the example project.

3. SmartTracker process flow and definition

This chapter contains the documentation of the *SmartTracker* functions needed for initialization, execution and parametrization.

3.1. Interface functions

3.1.1. Initialization of tracker

Function-Call	**itl_init_tracker** (cycleTime_s)
Purpose	Initialization of tracker and setting default parameters. Has to be called first, because of memory allocation and *SmartTracker* initiation. Input is the cycle time of the sensor. It is the constant time difference between two radar measurements.

3.1.2. Set default values

Function-Call	**itl_set_default_values** (productCode)
Purpose	This function is used for initialization of the tracker with sensor depending default values.

3.1.3. Tracker main function

Function-Call	**itl_execute_tracker** (pTargetListRV, nrOfTargets)
Purpose	Execution of tracker algorithm. The input arguments are the complete Radial-Velocity-Target-List and the number of targets received by the sensor.

3.1.4. Get tracker output

Function-Call	**itl_receive_track_list** (pTrackList, pNrOfTracks)
Purpose	This function provides the object / track list and the number of active tracks.

3.1.5. Set installation height

Function-Call	**itl_set_installation_height** (height)
Purpose	It is necessary to set the height of the iSYS-Radar System as good as possible to get the best possible performance.

3.1.6. Get installation height

Function-Call	**itl_get_installation_height** (pHeight)
Purpose	Used to receive the current height which is set in tracker.

3.1.7. Set installation angle

Function-Call	**itl_set_installation_angle** (angle)
Purpose	It is necessary to set the elevation angle of the iSYS-Radar System as good as possible to get the best possible performance. 0 degree means, that the sensor is installed horizontally.

3.1.8. Set installation angle

Function-Call	**itl_get_installation_angle** (pAngle)
Purpose	Used to receive the current elevation angle which is set in tracker.

3.1.9. Reset tracks

Function-Call	**itl_reset_tracks** ( )
Purpose	All active and inactive tracks will be deleted.

3.1.10. Set Ignore Zones

Process step	**Step 1.4**
Function-Call	**itl_set_ignore_zones** (pIgnoreZones, number of IgnoreZones)
Purpose	Interferers or reflections inside the observed area can cause unwanted tracks. To prevent false alarms they can be suppressed by using *Ignore Zones*. *Ignore Zones* have the following features: - a track may not be initialized inside an *Ignore Zone* - a track may move through an *Ignore Zone*, if it was confirmed outside before This function is used to set all *Ignore Zones*, you want to define, at once. The maximum number of *Ignore Zones* which can be set is defined by BT_MAX_NR_OF_IGNORE_ZONES. The maximum number of vertices of one *Ignore Zones* is defined by BT_MAX_NR_OF_POINTS_PER_IGNORE_ZONE.

3.1.1. Get Ignore Zones

Process step	**Step 1.4**
Function-Call	**itl_get_ignore_zones** (pIgnoreZones)
Purpose	This function is used to get all *Ignore Zones* at once. Note: The size of the struct is defined by BT_MAX_NR_OF_POINTS_PER_IGNORE_ZONE and BT_MAX_NR_OF_IGNORE_ZONES.

4. Appendix

4.1. Return Error Codes

Error Code	Description
ITL_OK	No error occurred.
ITL_ERROR_PROCESSING	Error in the Smart Tracker processing
ITL_ERROR_MEMORY_ALLOCATION	Occurs when it is not possible to allocate the required size of memory for the tracker application.
ITL_ERROR_PARAMETER	Occurs when a value of a parameter is out of range or when a parameter is not supported.
ITL_ERROR_STRUCT_SIZE	Size of a transmitted struct is not the size which is expected.
ITL_ERROR_PRODUCT_CODE	Tracker is called with an unsupported product.

4.2. Defines

Target state	Description
BT_MAX_NR_OF_TARGETS	The number of targets which can be processed by the *Smart Tracker* in one frame.
BT_MAX_NR_OF_TRACKS	The number of tracks which can be processed by the *Smart Tracker* in one frame.
BT_MAX_NR_OF_IGNORE_ZONES	Defines the number of *Ignore Zones* which can be set.
BT_MAX_NR_OF_POINTS_PER_IGNORE_ZONE	Defines the number of vertices of an *Ignore Zones*.

4.3. Target List

Target state	Description
f32_rcs_m2	Radar Cross Section of a target (in square meter)
f32_range_m	Range of target to sensor (in meter)
f32_velocity_mps	Radial velocity of a target to sensor (in meter per second)
f32_angleAzimuth_deg	Azimuth Angle of a target to sensor (in degree)
f32_reserved1	Unused
f32_reserved2	Unused

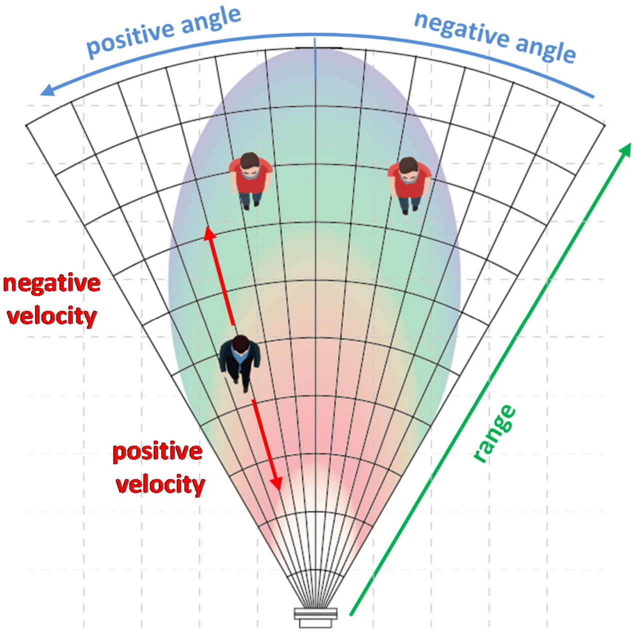

Figure 2: target information contained in target sub frame

4.4. Object List

Track state	Description
ui32_objectID	Each object has a unique identifier.
ui16_ageCount	Specifies the number of frames a track is existing.
ui16_predictionCount	Defines the number of consecutive frames in which a track could **not** assign a target.
ui16_staticCount	Defines the number of consecutive frames in which a track is marked as static.
f32_trackQuality	The quality of a track is an indicator how plausible and stable it is. The quality is defined from 0 to 100%. Note: Tracks with low quality will be deleted.
classID	The *SmartTracker* performs an internal classification to distinguish three different classes for identification. Types of classes: Pedestrian, Vehicle and Other.
f32_positionX_m	Current position of a track in x-direction.
f32_positionY_m	Current position of a track in y-direction.
f32_velocityX_mps	Current velocity of a track in x-direction.
f32_velocityY_mps	Current velocity of a track in y-direction.
f32_directionX	Current moving direction of a track in x-direction. Note: The x- and y-direction is normed to the unit vector.
f32_directionY	Current moving direction of a track in y-direction. Note: The x- and y-direction is normed to the unit vector.

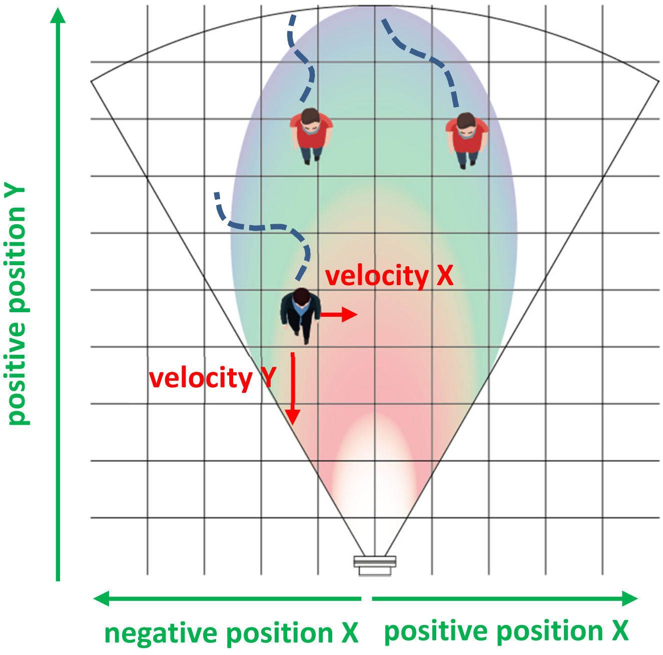

Figure 3: Object information contained in object list

InnoSenT GmbH
Am Roedertor 30
97499 Donnersdorf
GERMANY

Tel.: +49 95289518-0
E-Mail: info@innosent.de
www.innosent.de