KNOWLEDGE BASE APPLICATIONS TO
ADAPTIVE SPACE-TIME PROCESSING, VOLUME IV: KNOWLEDGE-BASED TRACKING

ITT Systems
Technology Service Corporation
Charles Morgan and Lee Moyer

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-SN-RS-TR-2001-146 Vol IV (of VI) has been reviewed and is approved for publication.

APPROVED:  
MICHAEL C. WICKS  
Project Engineer

FOR THE DIRECTOR:  
ROBERT G. POLCE, Chief  
Rome Operations Office  
Sensors Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/SNRT, 26 Electronic Pky, Rome, NY 13441-4514. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.
This report describes a knowledge-based tracking capability that will support a space-time adaptive processing (STAP) environment. The effort described in this report resulted in development of a multiple target tracker and the design and testing of several knowledge-based rules.

Three types of tracking filters are described:

1. An uncoupled two-state alpha-beta filter with position and velocity component states,
2. An uncoupled three-state Kalman filter with position, velocity, and acceleration component states, and
3. An extended four-state Kalman filter with both x and y position and velocity component states.

The first two filters use a one-dimensional measurement vector containing the report position component. The extended Kalman filters use a three-dimensional measurement vector containing x and y report position and "pseudo" Doppler, the latter defined as range times range rate.
# Table of Contents

1.0 Introduction and Overview ................................................................. 1

2.0 Tracker Description ............................................................................ 2

2.1 Basic Tracker ....................................................................................... 2

2.2 Knowledge Based Rules ....................................................................... 4

2.2.1 Tracking Legend .............................................................................. 5

2.2.2 Maneuver/Obstacle Rules ............................................................... 5

2.2.3 Shadow Rule ..................................................................................... 9

2.2.4 Discrete Rule .................................................................................. 10

3.0 Software Description ........................................................................... 11

Module Name: Tracker_GUI_7 ................................................................. 13

        Setup_Tracker_6 ............................................................................. 14

        Run_Tracker_8 .............................................................................. 17

        put_Table_R_2 ............................................................................. 18

        get_Table_R_2 ............................................................................. 20

        put_Table_Trk_4 .......................................................................... 21

        get_Table_Trk_4 .......................................................................... 23

        add_Table_Trk_4 .......................................................................... 24

        Cleanup_Table_Trk ....................................................................... 25

        print_Table_T ............................................................................... 26

        Display_Trk_Data_5 ...................................................................... 27

        set_Trackstate ............................................................................... 28

        set_TrackHM_stat ......................................................................... 29

        set_Track_gates ............................................................................. 30
# Table of Contents con’t

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shadow_Update</td>
<td>31</td>
</tr>
<tr>
<td>set_Track_Shadow</td>
<td>32</td>
</tr>
<tr>
<td>Predict_shlx_x</td>
<td>33</td>
</tr>
<tr>
<td>Kal_b_pred</td>
<td>35</td>
</tr>
<tr>
<td>Kal_c_pred</td>
<td>36</td>
</tr>
<tr>
<td>smooth_shlx</td>
<td>37</td>
</tr>
<tr>
<td>Kal_b_smth</td>
<td>39</td>
</tr>
<tr>
<td>Kal_c_smth</td>
<td>40</td>
</tr>
<tr>
<td>first_smooth_shlx</td>
<td>42</td>
</tr>
<tr>
<td>abtrack_init</td>
<td>43</td>
</tr>
<tr>
<td>track_init_2</td>
<td>44</td>
</tr>
<tr>
<td>track_init_c</td>
<td>45</td>
</tr>
<tr>
<td>Assign_Cov_manu</td>
<td>46</td>
</tr>
<tr>
<td>get_def_Cov_manu</td>
<td>48</td>
</tr>
<tr>
<td>Get_Semiaxes_3</td>
<td>49</td>
</tr>
<tr>
<td>E_Gate_Semiaxes_3</td>
<td>50</td>
</tr>
<tr>
<td>get_sig_track</td>
<td>53</td>
</tr>
<tr>
<td>Scale_Shadow_Gates</td>
<td>54</td>
</tr>
<tr>
<td>Rotate_xy2xyp</td>
<td>55</td>
</tr>
<tr>
<td>Test_E_Gate</td>
<td>56</td>
</tr>
<tr>
<td>Test_Smile_Gate</td>
<td>57</td>
</tr>
<tr>
<td>Pred_Shadow_Test</td>
<td>59</td>
</tr>
<tr>
<td>Test_In_Shadow</td>
<td>60</td>
</tr>
</tbody>
</table>
**Table of Contents con't**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete_Test</td>
<td>61</td>
</tr>
<tr>
<td>E_Discrete_Test</td>
<td>62</td>
</tr>
<tr>
<td>TR_Assoc_Max_T</td>
<td>63</td>
</tr>
<tr>
<td>TR_Assoc_Min_D</td>
<td>64</td>
</tr>
<tr>
<td>cmp_track_age</td>
<td>65</td>
</tr>
<tr>
<td>get_track_ang</td>
<td>66</td>
</tr>
<tr>
<td>get_pred_ang</td>
<td>67</td>
</tr>
<tr>
<td>update_error</td>
<td>68</td>
</tr>
<tr>
<td>Load_P_DET</td>
<td>69</td>
</tr>
<tr>
<td>Load_dBuf_4</td>
<td>70</td>
</tr>
<tr>
<td>plot_dBuf</td>
<td>71</td>
</tr>
<tr>
<td>Figure 1-1</td>
<td>Tracker GUI Options.................................................. 2</td>
</tr>
<tr>
<td>Figure 2-1</td>
<td>High Level Tracker Flow............................................... 3</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>Uncompensated Tracker Response for Maneuvering Target........... 6</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>Compensated Tracker Response for Maneuvering Target Track Oriented Elliptical Gates used with 0.5g Across-Track Acceleration Allowance....... 7</td>
</tr>
<tr>
<td>Figure 2-4</td>
<td>Compensated Tracker Response for Maneuvering Target “Smile” Shaped Centripetal Gates used with (0.0 – 0.5)g Across-Track Acceleration Allowance.............................................. 8</td>
</tr>
<tr>
<td>Figure 2-5</td>
<td>Tracker Response for Target Flying through Shadow No Shadow Rule Applied................................................................. 9</td>
</tr>
<tr>
<td>Figure 2-6</td>
<td>Tracker Response for Target Flying through Shadow Shadow Rule Applied............................................................ 10</td>
</tr>
<tr>
<td>Figure 2-7</td>
<td>Tracker Response for Target Passing Near a Discrete Discrete Rule Applied............................................................. 11</td>
</tr>
</tbody>
</table>
1.0 Introduction and Overview

The purpose of this effort has been to provide ITT Systems & Sciences with a knowledge based tracking capability that will support a space time adaptive processing (STAP) environment. This has included the development of a basic multiple target tracker and the design and testing of several knowledge based rules. A rule book containing 25 potential knowledge based rules was developed and is presented in Volume V.

For the purpose of ITT’s application, the main elements of the tracker software can be imbedded in a larger STAP simulation by removing the GUI. The key tracking modules are setup_Tracker_6 and Run_Tracker_8.

The Run_Tracker_module allows the use of three types of tracking filters. These include:

1. an uncoupled two state alpha beta filter with position and velocity component states,
2. an uncoupled three state Kalman filter with position, velocity, and acceleration component states, and
3. an extended four Kalman filter with both x and y position and velocity component states.

The first two filters use a one dimensional measurement vector containing the report position component. The extended Kalman filter uses a three dimensional measurement vector containing x and y report position and “pseudo” Doppler, the latter defined as range times range rate.

The tracking software as delivered has been imbedded in a GUI structure that makes it easy to exercise the tracker under a variety of conditions. Using the GUI, the user can interactively select existing or new target scenarios and tracking options prior to tracker execution. Figure 1-1 shows the list of available options.
2.0 Tracker Description

The first part of this section contains a narrative description of the basic multiple target tracker. Detailed documentation has been provided in Section 3. This is followed by a discussion of three knowledge based tracking rules that can be used to support the STAP processor. Simple tracking simulations have been provided to illustrate the use of these rules.

2.1 Basic Tracker

A high level description of the multiple target tracker function is shown below in Figure 2-1. The tracker processes reports on a per scan basis and makes use of a track table that contains several track attributes as well as kinematic information. The key attributes include the tracks identification number, its state value, which is a measure of track quality, and its status. Track status can be dropped, tentative, or firm. In the current software, a dropped status is assigned to a track whose state has been demoted to zero, a tentative status is assigned to a new
track formed by an unused report with a state value initialized to one, and a firm status is given to any track with a state value greater than one. The tracker’s function consists of a correlation section together with association and track maintenance sections.

The correlation section is performed by testing for inclusion each current report against each extrapolated track gate and forming a binary correlation matrix with ones in the capture positions. Tentative track gates formed from unassociated reports are extrapolated by centering a circle of kinematically-determined maximum radium about the report, whereas firm track gates are extrapolated by using the track filter prediction equations. Track gate sizes are typically a function of both the measurement and prediction uncertainties, as well as the track’s maneuver status. A distance matrix of the same size as the correlation matrix is also formed containing the report-to-gate center distances.

The association and maintenance section uses the above correlation and distance error arrays to assign reports and tracks in a unique manner. In the event that multiple reports are common to a given track gate, or multiple tracks capture a common report, the association logic will resolve the conflict. Furthermore, any reports that are not assigned to an existing track will be used to spawn a new track designated to have a tentative status. Two simple association logics are currently available. There is a “nearest neighbor” logic that assigns the closest captured report to a given track and there is a “venerable track” logic that associates the oldest common track to a given report.
After all tracks and reports have been associated, the track state promotion logic is applied. Track states are updated using either a "hit" or a "miss" table, depending on whether they correlate with a report. These state tables are usually designed to require specified numbers of successive hits and misses before a track is declared as firm or dropped. Two examples are given below. In Case 1, tables Hit1 and Miss1 allow a tentative track to build up to firm status at state level 4 after three successive hits, and to demote with each miss until it reaches state 0 where it is dropped. In Case 2, tables Hit2 and Miss2 show a more complex strategy. A tentative track promotes up to a firm status of 3 after two hits. However, there are hold states 4, 5 and 6 which allow the track to recover its firm status more quickly after a single miss.

<table>
<thead>
<tr>
<th>Case 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
</tr>
<tr>
<td>Hit1</td>
</tr>
<tr>
<td>Miss1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
</tr>
<tr>
<td>Hit2</td>
</tr>
<tr>
<td>Miss2</td>
</tr>
</tbody>
</table>

Once track state logic has been applied, the track table is updated. All tracks with state values of 0 are removed from the table. Furthermore, all reports that were not associated with extant tracks are used to spawn new tentative tracks that are added to the table.

### 2.2 Knowledge Based Rules

Knowledge based rules make use of extended map and intelligence information, and are used to improve the tracker's ability to support STAP processing requirements. Key issues include the tracking of targets in regions adjacent to large discrete devices, and in shadow zones that are blocked from the radar's line of sight. Using map information the tracker should also be able to anticipate a target maneuver that will be required to avoid obstacles. Queued with this information, the tracker will apply appropriately shaped track gates that enhance its capability of capturing reports while maintaining reasonable gate sizes. Four rules are discussed below, along with simple tracking simulations.

In the following examples single target tracks are displayed graphically with the following conventions:
2.2.1 Tracking Legend:

+ Predicted gate center position
* Measured report position
o Coasted track position
• Extrapolated position of a missed detection
d Dropped tentative track
D# Dropped firm track
# Corresponds to age (scans) of dropped track

In addition, all the results were obtained using an uncoupled Kalman tracking filter and all simulations assumed a ten second scan period.

2.2.2 Maneuver/Obstacle Rules:

Both alpha beta and Kalman tracking filters do a good job with targets that move along a constant heading with a fixed speed. Deviations from a straight path cause prediction errors to occur and can ultimately result in a dropped track. Therefore it is important, whenever possible, to anticipate target maneuvers by several scans. This allows time to make such adjustments as increasing the gate size and track gain, or using a shaped gate to allow for across track deviations caused by target turning.

If a track approaches an obstacle whose across track extent is H, a maneuver can be anticipated to occur within a time extent no longer than $T_{\text{max}}$. For this discussion, assuming a constant target speed $v$ and a maximum possible acceleration $A_{\text{max}}$, this extent is given by:

$$T_{\text{max}} = \frac{\rho}{v} \cos^{-1} \left(1 - \frac{H}{\rho} \right)$$

where

$$\rho = v^2 / A_{\text{max}}$$

denotes the radius of curvature of the target turn required to clear the obstacle. Somewhere within this time period the tracker should apply its maneuver logic.

Figures 2-2 to 2-4 use the same section of track to illustrate the effect of different gating strategies on the tracker's ability to handle a maneuvering target. Performance is computed for a constant speed (250 meters/sec), high SNR (20 dB at mid range) Swerling 1 target, as it approaches an obstacle (shaded rectangular region) from below and begins to turn after the twenty third scan.

Figure 2-2 shows the response of an uncompensated tracking filter using a standard track gate centered on the predicted gate center, and oriented along and across range with a size dependent on both the measurement and prediction errors. No target acceleration has been
assumed and no maneuver noise has been added. While the straight line section is handled adequately, the initial track begins to lose the target after the turn begins, whereupon it is demoted to a dropped status on the next four scans. The three kilometer circular gates indicate newly spawned tentative tracks that were subsequently updated to firm status.

Figure 2-2: Uncompensated Tracker Response for Maneuvering Target

The first maneuvering target rule specifies the use of shaped elliptical gates. Figure 2-3 shows the same target as tracked using a combination of two gates, both centered on the predicted gate center. One gate is oriented along and across range, with a size determined by range and angle measurement uncertainties. The second gate is oriented along and across the target's instantaneous track and its size is a function of the maximum turning acceleration of the target, assumed here to be 0.5g units. Reports capture occurs if the measurement falls within either gate. Except for a missed report early in the linear part of the trajectory, causing the gate to swell initially and then settle down, the track is maintained throughout the maneuver. Each lower case d symbol indicates a dropped tentative track. This occurs when such a track fails to capture a report on the following scan. The large three kilometer circles denote tentative tracks that were successfully promoted to firm status. Finally, the dots occurring in both Figures 2-2 and 2-3, shown extrapolated from the straight line section of the approaching track, represent missed detections. While actual gates existed for these cases, they have been drawn here only or situations in which reports were captured.
Figure 2-3: Compensated Tracker Response for Maneuvering Target. Track Oriented Elliptical Gates used with 0.5g Across-Track Acceleration Allowance
The second maneuvering target rule specifies the use of a gate whose shape is determined solely by centripetal turning mechanics. Figure 2-4 shows the target being tracked using a “smile shaped” gate whose shape is determined by the kinematic constraints imposed on a constant speed target undergoing a centripetal maneuver. Let $t$ range over the time interval from the last track update till the next predicted report acquisition, $T_{\text{scan}}$ seconds later. The maneuver envelope is given by the $xy$ locus of points, oriented along and across the track, and generated by the equations:

$$x = vt + \rho \sin \theta$$

$$y = \rho (1 - \cos \theta)$$

$$\theta = v(T_{\text{scan}} - t)/\rho$$

where the radius of curvature $\rho$ is defined as above. For this gate, a track capture occurs if the measurement ellipse, centered on the measured report and oriented along and across range, intersects the smile locus. As in the previous example, this gate is able to maintain the track throughout the target maneuver. One advantage of this gate is that it has a relatively small area as compared with other maneuver gates and this makes it less likely to capture any false alarms.

**Figure 2-4:** Compensated Tracker Response for Maneuvering Target. “Smile” Shaped Centripetal Gates used with $(0.0 - 0.5)$ g Range of Across-Track Acceleration Allowance
2.2.3 Shadow Rule:

The shadow rule provides a means of preserving firm tracks that enter regions shadowed from the radar line of sight. If the predicated track gate center falls within a designed shadow region, both the track state and gate size will be frozen. Upon emerging from the shadow, state promotion resumes and the gate size will not be allowed to exceed a maximum value. In the tracker scenario used in Figures 2-5 and 2-6, only a minimal amount of acceleration noise, 0.05 g units, was used in order to maintain a straight line coasting of the track through the shadow. As previously, the target has a speed of 250 meters per second and the update scan time is 10 seconds.

Figure 2-5 shows a section of tracker response, when no shadow rule is in effect, for a target approaching a shadow zone (shaded rectangle) from below. The three dots denote extrapolated positions of the initial track where no reports were captured. After four demotions the firm track that initially entered the shadow was dropped, as indicated by the D14 symbol. A new tentative track was spawned, and promoted, once the predicted gate positions moved beyond the shadow. Note that the D symbol has been placed at the last updated track position, just prior to entering the shadow, where the track was 14 scans old.

Figure 2-5: Tracker Response for Target Flying through Shadow
No Shadow Rule Applied

Figure 2-6 shows the track history for the same target-shadow scenario when the shadow rule was in place. The circles in the shadow denote coasted track positions at which the track state was held fixed. Once the predicted gate position emerged from the shadow, there was a
moderate increase in gate size, after which it settled down, and the original firm track continued undisturbed.

Figure 2-6: Tracker Response for Target Flying through Shadow
Shadow Rule Applied

2.2.4 Discrete Rule:

By tagging large radar returns, or discretes, the STAP processor can exclude regions containing them from its covariance matrix element formation and thereby not use up limited degrees of freedom on their cancellation. The discrete rule allows the tracker to coast through any region containing one of these tagged returns and to essentially ignore it. If a known discrete falls within a track gate, that track will be treated as if in a shadow and will not be updated.

In Figure 2-7, the same target speed, update time, and acceleration noise have been used as in the shadow rule examples. A discrete has been inserted in the target trajectory as shown. As indicated, the tracker preserves the state value of 4 as the predicted gate positions passes through the discrete.
The previous discussion looked at the implementation of four specific tracking rules. However, many more rules that were considered relevant to the STAP problem were developed during the course of this tracking study. In addition to the rule topics discussed here, issues regarding the assignment of target priority, detection threshold, state promotion logic, and other features were considered. A collection of twenty-five such rules are presented in the "Knowledge-Based Tracker Rule Book" that is included in Volume V of this report. Included with each rule is a discussion of its rationale, its impact on the overall knowledge based system and interface requirements that might exist between the tracker, controller and radar.

3.0 Software Description

All tracking software used in this effort has been written in Matlab 5.1. A total of 49 modules are listed and documented below. These can be subdivided into three groupings consisting of user interface, main tracker, and tracker utilities. The first two modules belong to the user interface group. These are Tracker_GUI_7, which provides the graphical user interface for the overall tracking simulation, and Setup_Tracker_6, which sets up the interface between the tracker and the scenario generator. The main tracker group contains the multiple target tracking module, Run_Tracker_8, which carries out all prediction, smoothing, and association operations on report data on a per scan basis. Finally, the tracker utilities group contains all of the support modules that are used by the tracker. It should be noted that some of the modules supporting the user interface group have not been documented since that portion of the software is going to be removed by the customer and replaced with their own hooks into the STAP simulation software.
**User Interface Modules**

Tracker_GUI_7  
Setup_Tracker_6

**Main Tracker Module**  
Run_Tracker_8

**Tracker Utility Modules**  
put_Table_R_2  
get_Table_R_2  
put_Table_Trk_4  
get_Table_Trk_4  
add_Table_Trk_4  
Cleanup_Table_Trk  
print_Table_Trk  
Display_Trk_Data_5  
State_Update  
set_Trackstate  
set_TrackHM_stat  
set_Track_gates  
Shadow_Updates  
set_Track_Shadow  
predict_shlx_x  
Kal_b_pred  
Kal_c_pred  
smooth_shlx  
Kal_b_smth  
Kal_c_smth  
first_smooth_shlx  
abtrack_init  
track_init_2
Module Name: Tracker_GUI_7

Calling Module: None

Called Modules: All_Defaults_4
Demo_Defaults_4
Read_Scenario_2
Write_Scenario_2
Track_Data_Gen_4
Radar_menu_3
Tracker_Opt_menu_2
Setup_Tracker_6
Run_Tracker_8

Inputs: None

Outputs: None

Globals:
***Radar_menu_3***
first_in_Radar
f_Ghz N_hits SNR0_dB Prf_kHz
DRng_m Az_mrad_3dB Dop_hz
Pfa SNR_fa_dB
***Scenario_menu***
script_name
Tracker_Opt RESP
***Track_Filter_menu***
Tfilt_RESP
***Tracker_param_menu***
first_in_Tr_parm
Tfilt_RESP
Mult_LP Mult_HP radius_TENT_LP radius_TENT_HP
Man_amt alpha beta Pri_T
Turn_int
gate_case
***Tracker_prom_menu***
first_in_Tr_prom
Hit_Tbl_LP Miss_Tbl_LP Hit_Tbl_HP Miss_Tbl_HP
*** ***
fig_no_T
Time X_meas Y_meas pDop_meas report_ID
X_tru Y_trum pDop_tru
S11 S12 S13 S22 S23 S33
ang_pred Angle (radians) of prediction point
ang_trk Angle (radians) of track
gate_on Draw track gate flag (1=> yes)
print_Trk_Tbl Print track gate flag (1=> yes)
text_on Label track with text (1=> yes)
N_shadow Number of shadow zones in scenario
x_sh_LO Array of low x shadow tile values
x_sh_HI Array of high x shadow tile values
y_sh_LO Array of low y shadow tile values
y_sh_HI Array of high y shadow tile values
In_Shadow In shadow flag (1=> yes)
Nshadow_dwell Number of consecutive dwells in shadow

Globals:
Mult
Table_R
Table_T
DROPPED TENT FIRM
LP HP
SUM1_ERROR SUM2_ERROR
P_DET
HP_Buf
dBuf
Display_cnt

Description:

Initializes and sets up variables to be used by the Run_Tracker_8 program. Puts scenario generator outputs in a form usable by tracker. Also adds false alarms to scenario generation data.
<table>
<thead>
<tr>
<th>Module Name:</th>
<th>Run_Tracker_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling Modules:</td>
<td>None (script file)</td>
</tr>
</tbody>
</table>
| Called Modules: | put_Table_R_2  
get_Table_R_2  
get_Table_Trk_4  
put_Table_Trk_4  
add_Table_Trk_4  
predict_shlx_x  
first_smooth_shlx  
smooth_shlx  
Get_Semiaxes_3  
Test_E_Gate  
Test_Smile_Gate  
get_def_Cov_manu  
Assign_Cov_manu  
State_Update  
set_Trackstate  
set_TrackHM_stat  
Pred_Shadow_Test  
Shadow_Update  
Discrete_Test  
set_Track_Shadow  
get_track_ang  
get_pred_ang  
update_error  
TR_Assoc_Max_T  
Display_Trk_Data_5  
print_Table_T  
Load_dBuf_4  
plot_dBuf  
Cleanup_Table_Trk |
| Inputs: | setup by Tracker_GUI_7 and  
Setup_Tracker_6 |
| Outputs: | None |
| Globals: | None |
| Description: | The multiple target tracker processes report data collected during each scan and performs three basic functions: track-report correlation, association, and maintenance. |
**Module Name:** put_Table_R_2  
**Calling Module:** Run_Tracker  
**Called Modules:** None

**Inputs:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Time of radar blips</td>
</tr>
<tr>
<td>X_meas, Y_meas</td>
<td>Measured position (km)</td>
</tr>
<tr>
<td>FDop_meas</td>
<td>Measured pseudo Doppler (km*km/sec)</td>
</tr>
<tr>
<td>X_tru, Y_tru</td>
<td>True position (km)</td>
</tr>
<tr>
<td>FDop_tru</td>
<td>True pseudoDoppler (km*km/sec)</td>
</tr>
<tr>
<td>S11, ..., S33</td>
<td>Measurement Covariance</td>
</tr>
<tr>
<td>sig_Rng_km</td>
<td>Range Measurement error (km)</td>
</tr>
<tr>
<td>sig_Az_rad</td>
<td>Azimuth measurement error (radians)</td>
</tr>
<tr>
<td>sig_Rdot_kmps</td>
<td>Range rate measurement error (km/sec)</td>
</tr>
<tr>
<td>freq_GHz</td>
<td>Radio frequency of radar (GHz)</td>
</tr>
<tr>
<td>Det_Level</td>
<td>Detection level</td>
</tr>
<tr>
<td>Prior_intel</td>
<td>Priority status</td>
</tr>
<tr>
<td>Manu_intel</td>
<td>Maneuver status</td>
</tr>
<tr>
<td>report_ID</td>
<td>Trajectory ID of report</td>
</tr>
<tr>
<td>Nrep</td>
<td>Number of reports calling Module: Run_Tracker</td>
</tr>
</tbody>
</table>

**Called Modules:** None

**Inputs:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Time of radar blips (sec)</td>
</tr>
<tr>
<td>X_meas, Y_meas</td>
<td>Measured position (km)</td>
</tr>
<tr>
<td>FDop_meas</td>
<td>Measured pseudo Doppler (km*km/sec)</td>
</tr>
<tr>
<td>X_tru, Y_tru</td>
<td>True position (km)</td>
</tr>
<tr>
<td>FDop_tru</td>
<td>True pseudoDoppler (km*km/sec)</td>
</tr>
<tr>
<td>S11, ..., S33</td>
<td>Measurement Covariance</td>
</tr>
<tr>
<td>sig_Rng_km</td>
<td>Range Measurement error (km)</td>
</tr>
<tr>
<td>sig_Az_rad</td>
<td>Azimuth measurement error (radians)</td>
</tr>
<tr>
<td>sig_Rdot_kmps</td>
<td>Range rate measurement error (km/sec)</td>
</tr>
<tr>
<td>freq_GHz</td>
<td>Radio frequency of radar (GHz)</td>
</tr>
<tr>
<td>Det_Level</td>
<td>Detection level</td>
</tr>
<tr>
<td>Prior_intel</td>
<td>Priority status</td>
</tr>
<tr>
<td>Manu_intel</td>
<td>Maneuver status</td>
</tr>
<tr>
<td>report_ID</td>
<td>Trajectory ID of report</td>
</tr>
<tr>
<td>Nrep</td>
<td>Number of reports calling Module: Run_Tracker</td>
</tr>
<tr>
<td>scan</td>
<td>Current scan index</td>
</tr>
</tbody>
</table>

**Outputs:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table_R</td>
<td></td>
</tr>
</tbody>
</table>

18
Globals: Table_R

Description:

Load the report table buffer for the current scan. All arrays are indexed as (report, scan) and were created by a scenario generator.
Module Name: get_Table_R_2

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:
- rep

Outputs:
- T_m
- X_m, Y_m
- FDop_m
- Cov_m_vec
- sig_rng_km_m
- sig_crng_km_m
- sig_Rdot_kmps_m
- freq_GHz
- Det_m
- priority_in
- maneuver_in
- report_source
- X_tru_m, Y_tru_m
- FDop_tru_m
- error_status

Blip time (sec)
Measured position (km)
Measured Doppler (km*km/sec)
Measured covariance vector
Range measurement error (km)
Cross range measurement error (km)
Range rate measurement error (km?sec)
Radar rf frequency (GHz)
Detection level
Priority Status
Maneuver status
Trajectory ID
True position (km)
True pseudoDoppler (km*km/sec)
not used

Globals: Table_R

Description:

Fetch report table data for current scan.
Module Name: put_Table_Trk_4

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

trk: Track index
TID: Track ID number
status: Track status (drop, tentative, firm)
state: Track state (quality index; 0 => dropped)
report: Captured report index
scan1: Scan index when track became firm
trk_type: Tracker type (alpha beta, unc/extd Kalman)
Z_m: Measurement state vector
Cov_m_vec: Measurement covariance vector
T_m: Measurement of time of captured report
priority: Priority status of track
maneuver: Maneuver status of track
Z_p: Prediction state vector
Cov_p_vec: Prediction covariance vector
Z_s: Smoothed state vector
Cov_s_vec: Smoothed covariance vector
HM_flg: Track capture flag (0 => miss, 1 +. hit)
ang_trk: Track angle (radians)
x_s_2LST: Last track x position (km)
y_s_2LST: Last track y position (km)
T_s_2LST: Last track time (sec)
semi_rng_T: along range semi axis (km)
semi_crng_T: cross range semi axis (km)
semi_trk_T: along track semi axis (km)
semi_ctrk_T: cross track semi axis (km)
In-Shadow: Shadow status flag (0 => not in)
Nshadow_dwells: Number of successive dwells in shadow
gate_case: Track gating choice (0:3)
Accel_max: Maximum acceleration (km/sec*sec)
sig_rng_km: range measurement error (km)
sig_crng_km: cross range measurement error (km)
sig_Rdot_kmps2: range rate meas error (km*km/sec)

Outputs:

error_status
Table_T
Globals: Table_T

Description:

Updates track table data for current scan and report index.
<table>
<thead>
<tr>
<th>Module Name:</th>
<th>get_Table_Trk_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling Module:</td>
<td>Run_Tracker_8</td>
</tr>
<tr>
<td>Called Modules:</td>
<td>None</td>
</tr>
<tr>
<td>Inputs:</td>
<td>trk</td>
</tr>
<tr>
<td>Outputs:</td>
<td>TID</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>sig_Rdot_kmps2</td>
</tr>
<tr>
<td>Globals:</td>
<td>Table_T</td>
</tr>
</tbody>
</table>

**Description:**

Fetch track table data for current scan and report index.
**Module Name:** add_Table_Trk_4  
**Calling Module:** Run_Tracker_8  
**Called Modules:** None  

**Inputs:**  
- Ntrk  
- num_new_trks  
- TID_New  
- ...  
- sig_Rdot_kmps2  
  
  same as inputs to put_Table_Trk4  

**Outputs:**  
- error_status  
- Table_T  

**Globals**  
- Table_T  

**Description:**  
Insert new tentative tracks into track table.
Module Name: Cleanup_Table_Trk
Calling Module: Run_Tracker_8
Called Modules: None
Inputs: None
Outputs:
   Ntrk  Number of valid tracks
Globals: Table_T
Description:

Updates track table for next scan. Sorts Table_T by column 3 (state) and removes all zero states. Counts number of remaining valid tracks and returns this value.
Module Name: print_Table_T

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

- Ntrk
- sup

Number of track sets to be printed (max = 10)
vector of Table_T indices to be printed

Outputs:

prints to command tool

Globals:

Table_T

Description:

Prints track table information as columns (one per track). Support vector sup is subset of {1:86} corresponding to fields in Table_T.
Module Name: Display_Trk_Data_5

Calling Module: Run_Tracker_8

Called Modules: Draw_Grid
get_Table_Trk_4
get_Table_R_2
Assign_Cov_manu
predict_shlx_x
get_sig_track
get_pred_ang
E_Gate_Semiaxes
Draw_Smile_Gate
Draw_E_Gate_2

Inputs:

Ntrk Number of tracks to be plotted
scan current scan index
scan_period scan time (sec)
sig_rng_km_nom not used
sig_Az_rad_nom not used
plot_vec plot scale: (xmin, xmax, ymin, ymax)
x_v, y_v x and y grid point sets
dx, dy x and y grid spacing
gate_on Draw track gate flag (1 => yes)
text_on Print Text gate
fig_no plot figure number
symbol_1 rng-cross rng aligned gate symbol
symbol_2 trk-cross trk aligned gate symbol
semi-max maximum gate size allowed

Outputs: graphical display in figure (fig_no)

Globals:

Display_cnt
Table_R Table_T
DROPPED TENT FIRM LP HP
Mult_LP Mult_HP radius_TENT_LP radius_TENT_HP
sig_trk_km sig_ctrk_km gate_case
Cov_man20 Cov_man30 Cov_man_acc_x Cov_man_acc_y

Description:

The module serves as a diagnostic and display tool for the multi target tracker. It displays predicted and measured positions prior to track table cleanup and draws track gates centered on predicted positions.
Module Name: set_Trackstate
Calling Module: Run_Tracker_8
Called Modules: None
Inputs:
    state
    trk
Outputs: None
Globals: Table_T
Description:
Inputs state value into track table at index trk.
Module Name: set_TrackHM_stat

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:
- HM_flg: Capture/shadow status of track
- trk: Track index

Outputs: None

Globals: Table_T

Description:
Inputs the track capture/shadow status into track table.
Module Name: set_Track_gates

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

- semi_rng_T: Along range semi axis of ellipse gate
- semi_crng_T: Cross range semi axis
- semi_trk_T: Along track semi axis
- semi_ctrk_T: Cross track semi axis
- trk: Track table index

Outputs: None

Globals: Table_T

Description:

Inputs ellipse gate semi axes into track table corresponding to index trk.
Module Name: Shadow_Update
Calling Module: Run_Tracker_8
Called Modules: set_Track_Shadow set_TrackHM_stat

Inputs:

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trk</td>
<td>Track index into Table_T</td>
</tr>
<tr>
<td>In_Shadow</td>
<td>Shadow status flag (1 =&gt; in shadow)</td>
</tr>
<tr>
<td>Nshadow_dwells_last</td>
<td>Number of scans input trk in shadow</td>
</tr>
<tr>
<td>HM_flg_last</td>
<td>Capture/shadow status of input trk</td>
</tr>
</tbody>
</table>

Outputs: None

Globals: Table_T

Description:

Updates track shadow status corresponding to index trk. If track is in a shadow, the number of shadow dwells is incremented and the capture status is set to the number 2. Otherwise, the number of dwells is set to zero and the capture status maintained.
Module Name: set_Track_Shadow

Calling Module: Run_Tracker_8 Shadow_Update

Called Modules: None

Inputs:
- Nshadow_dwells: Number of consecutive dwells in shadow
- In_Shadow: in shadow status flag
- trk: Index into Table_T

Outputs: None

Globals: Table_T

Description:
Inputs shadow parameters into track table corresponding to index trk. These include (1) the # of consecutive dwells of track in a shadow, and (2) the in-shadow status flag (0 => not in shadow, 1 => in shadow).
Module Name: predict_shlx_x  
Calling Module: Run_Tracker_8  
Called Modules: Kal_b_pred, Kal_c_pred  

Inputs:  
- type: Tracking filter type (1:3 +. alpha beta, uncoupled Kalman, and extended Kalman)  
- Z_s_last: Smoothed state vector at last scan  
- Cov_s_last_vec: Smoothed covariance array at last scan  
- Cov_man_x: Maneuver covariance matrix of x component  
- Cov_man_y: Maneuver covariance matrix of y component  
- Cov_man: Maneuver covariance matrix of extended Kal  
- dT: Time interval from last update to present  

Outputs:  
- Z-P: Prediction state vector  
- Cov_p_vec: Prediction covariance matrix  

 Globals: None  

Description:  
Computes predicted state vector and Covariance array, Z_p and Cov_p_vec, respectively. The method used depends on the tracking filter type specified. The maneuver covariance matrices provide a means for inputting acceleration noise to increase track gate size for the Kalman filter cases  

Equations:  

alpha beta:  

\[ X_p = x_{s\_last} + v_{x\_last} \times dT \]  
\[ Y_p = y_{s\_last} + v_{y\_last} \times dT \]  
\[ Vx_p = v_{x\_last} \]  
\[ Vy_p = v_{y\_last} \]  

where the components X_p, Vx_p, etc., are related to the state vectors Z_p, Z_s_last, etc. by stacking the x and y components of position, velocity, and acceleration  

\[ Z= [X;Vx;Ax;Y;Vy;Ay] \]
Uncoupled Kalman:
   see Kal_b_pred

Extended Kalman:

   see Kal_c_pred
Module Name: Kal_b_pred

Calling Module: predict_shlx_x

Called Modules: None

Inputs:
- \( Z_{s\_in} \) 3x1 Smoothed input state vector (either component)
  [Position; Velocity; Acceleration]
- \( dt\_in \) Time since last update
- Cov_manu 3x3 Covariance matrix of maneuver noise
- Cov_s_in 3x3 Smoothed input covariance matrix (either component)
  [Position; Velocity; Acceleration]

Outputs:
- \( Z_{p\_out} \) 3x1 Prediction state vector (corresponding component)
- Cov_p 3x3 Prediction covariance matrix

Globals: Table_T

Description:
Computes 3 x 1 prediction state vector and 3 x 3 prediction covariance matrix for the case of an uncoupled Kalman filter. All input and output state vectors are assumed to be 3 x 1, and all covariance matrices are 3 x 3.

Equations:
\[
\phi = \begin{bmatrix}
1 & dT & 0.5(dT)^2 \\
0 & 1 & dT \\
0 & 0 & 1 \\
\end{bmatrix}; 3 \times 3 \text{ state transition matrix}
\]

\( Z_{p\_out} = \phi * Z_{s\_in} \)

\( \text{Cov}_p = \phi * \text{Cov}_s\_in * \phi' + \text{Cov}_\text{manu} \)

(\( \phi' \) denotes the transpose of \( \phi \))
Module Name: Kal_c_pred

Calling Modules: predict_shlx_x

Called Modules: None

Inputs:

<table>
<thead>
<tr>
<th></th>
<th>4 x 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_s_in</td>
<td>Smoothed input state vector</td>
</tr>
<tr>
<td></td>
<td>([X_s_{\text{in}}; Vx_{s_{\text{in}}}; Vy_{s_{\text{in}}}])</td>
</tr>
<tr>
<td>dt_in</td>
<td>Time since last update</td>
</tr>
<tr>
<td>Cov_manu</td>
<td>4 x 4</td>
</tr>
<tr>
<td></td>
<td>Covariance matrix of maneuver noise</td>
</tr>
<tr>
<td>Cov_s_in</td>
<td>4 x 4</td>
</tr>
<tr>
<td></td>
<td>Smoothed input covariance matrix</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th></th>
<th>4 x 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_p_out</td>
<td>Prediction state vector</td>
</tr>
<tr>
<td></td>
<td>([X_{p_{\text{out}}}; Vx_{p_{\text{out}}}; Y_{p_{\text{out}}}; Vy_{p_{\text{out}}}])</td>
</tr>
<tr>
<td>Cov_p</td>
<td>4 x 4</td>
</tr>
<tr>
<td></td>
<td>Prediction covariance matrix</td>
</tr>
</tbody>
</table>

Globals: None

Description:

Computes 4 x 1 prediction state vector and 4 x 4 prediction covariance matrix for the case of an extended Kalman filter. All input and output state vectors are assumed to be 4 x 1, and all covariance matrices are 4 x 4.

Equations:

\[
\phi = \begin{bmatrix} 1 & dT & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dT \\ 0 & 0 & 0 & 1 \end{bmatrix}; 4 \times 4 \text{ state transition matrix}
\]

\[
Z_{p_{\text{out}}} = \phi \ast Z_{s_{\text{in}}}
\]

\[
Cov_{p} = \phi \ast Cov_{s_{\text{in}}} \ast \phi' + Cov_{man}
\]

\((\phi' \text{ denotes the transpose of } \phi)\)
Module Name: smooth_shlx

Calling Module: Run_Tracker_8

Called Modules:
Kal_b_smth
Kal_c_smth

Inputs:
- **type**: Tracking filter type (1:3 => alpha beta, uncoupled Kalman, and extended Kalman)
- **Z_m**: Measurement vector (x;y;pDop)
- **Z_p**: Prediction state vector (X;Vx;AxA;Y;Vy;Ay)
- **Cov_p_vec**: Prediction cov array (Cov_p_x;Cov_p_y)reshaped to 1 x 18
- **Cov_m_vec**: Meas cov array 1 x 9
- **dT**: Time interval from last update to present

Outputs:
- **Z_s**: Smoothed state vector (X;Vx;AxA;Y;Vy;Ay)
- **Cov_s_vec**: Smoothed cov array (Cov_s_x;Cov_s_y)reshaped to 1 x 18
- **Gain_mat**: Gain matrix (not normally used by tracker)

Globals: alpha beta

Description:
Computes smoothed state vector and Covariance array, Z_s and Cov_s_vec, respectively. Also returns a gain matrix (not normally used) which can be used for diagnostic purposes. The smoothing method used depends on the tracking filter type specified.

Equations:

**alpha beta:**

\[
\begin{align*}
  Dx &= x_m - X_p \\
  x_s &= X_p + \alpha \cdot Dx \\
  Vx_s &= Vx_p + \beta \cdot Dx/dT \\
  Dy &= y_m - Y_p \\
  y_s &= Y_p + \alpha \cdot Dy \\
  Vy_s &= Vy_p + \beta \cdot Dy/dT
\end{align*}
\]

where

\[
Z_m = [x_m; y_m; pDop_m]
\]

and Z_p and Z_s are 6 x 1 vectors of the position, velocity and acceleration components
Uncoupled Kalman:  see Kal_b_smth

Extended Kalman:  see Kal_c_smth
Module Name: Kal_b_smth

Calling Module: smooth_shlx

Called Modules: None

Inputs:

- \( Z_m \) - 3x1 Measurement vector \([x_m; y_m; pDop_m]\)
- \( Z_{p\text{-out}} \) - 3x1 Prediction state vector (either component) \([\text{Position; Velocity; Acceleration}]\)
- \( \text{Cov}_p \) - 3x3 Covariance matrix of maneuver noise
- \( \text{Cov}_{s\text{-in}} \) - 3x3 Prediction covariance matrix
- \( \text{var\_meas} \) - 3x3 Measurement variance

Outputs:

- \( Z_{s\text{-out}} \) - 3x1 Smoothed output state vector (either component) \([\text{Position; Velocity; Acceleration}]\)
- \( \text{Cov}_{s\text{-out}} \) - 3x3 Smoothed output covariance matrix
- \( \text{Res} \) - 3x1 Residual error vector (Innovations)
- \( \text{dist\_Res\_2} \) - Statistical distance
- \( S \) - 3x1 Gain matrix

Globals: None

Description:

Computes 3x1 smoothed state vector and 3x3 smoothed covariance matrix for the case of an uncoupled Kalman filter. All input and output prediction and smoothed state vectors are assumed to be 3x1, and all covariance matrices are 3x3.

Equations:

(for either x or y component)

\[ M = [1 0 0]; \] Measurement Matrix (1 Measurement x 3 states)
\[ I_{\text{Cov\_Res}} = 1/(M * \text{Cov\_p} * M' + \text{var\_meas}) \]
\[ S = \text{Cov\_p} * M' * I_{\text{Cov\_Res}}; \] Gain Matrix (3 states x 1 measurement)
\[ Z_{s\text{-out}} = Z_{p\text{-out}} + S * (Z_m - M * Z_{p\text{-out}}) \]
\[ \text{Cov}_{s\text{-out}} = (I - S * M) * \text{Cov\_p} \]
\[ \text{Res} = (Z_m - M * Z_p); \] Innovations
\[ \text{dist\_Res\_2} = \text{Res} * I_{\text{Cov\_Res}} * \text{Res}'; \] Statistical distance
Module Name: Kal_c_smth

Calling Modules: smooth_shlx

Called Modules: None

Inputs:

<table>
<thead>
<tr>
<th>Input</th>
<th>Shape</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_m</td>
<td>3 x 1</td>
<td>Measurement vector (3 x 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[x_m; y_m; pDop_m]</td>
</tr>
<tr>
<td>Z_p_out</td>
<td>4 x 1</td>
<td>Prediction state vector [X; Vx; Y; Vy]</td>
</tr>
<tr>
<td>Cov_p</td>
<td>4 x 4</td>
<td>Prediction covariance matrix</td>
</tr>
<tr>
<td>Cov_meas</td>
<td>3 x 3</td>
<td>Measurement Covariance</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th>Output</th>
<th>Shape</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_s_out</td>
<td>4 x 1</td>
<td>Smoothed output state vector [X; Vx; Y; Vy]</td>
</tr>
<tr>
<td>Cov_s_out</td>
<td>4 x 4</td>
<td>Smoothed output covariance matrix</td>
</tr>
<tr>
<td>Res</td>
<td>3 x 1</td>
<td>Residual error vector (Innovations)</td>
</tr>
<tr>
<td>dist_Res_2</td>
<td></td>
<td>Statistical distance</td>
</tr>
<tr>
<td>S</td>
<td>4 x 3</td>
<td>Gain Matrix</td>
</tr>
</tbody>
</table>

Globals: None

Description:

Computes 4 x 1 smoothed state vector and 4 x 4 smoothed covariance matrix for the case of an extended Kalman filter. All input and output prediction and smoothed state vectors are assumed to be 4 x 1, and all covariance matrices are 4 x 4.

Equations:

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
M_{31} & M_{32} & M_{33} & M_{34}
\end{bmatrix}; \text{Measurement matrix (3 Measurements x 4 states)}
\]

where

\[
M_{31} = 0.5 \times Z_{p\text{.out}} \quad (2)
\]
\[
M_{32} = 0.5 \times Z_{p\text{.out}} \quad (1)
\]
\[
M_{33} = 0.5 \times Z_{p\text{.out}} \quad (4)
\]
\[
M_{34} = 0.5 \times Z_{p\text{.out}} \quad (3)
\]
Note that

\[(pDop)_{est} = M_{31} \cdot X_p + M_{32} \cdot Vx_p + M_{33} \cdot Y_p + M_{34} \cdot Vy_p\]

\[I_{Cov\_Res} = (M \cdot Cov_p \cdot M' + Cov_{meas})^{-1}\]

\[S = Cov_p \cdot M' \cdot I_{Cov\_Res}; \text{ 4 x 3 Gain Matrix (4 states x 3 measurements)}\]

\[Res = (Z_m - M \cdot Z_{p\_out}); \text{ 3 x 1 Innovations}\]

\[Z_{s\_out} = Z_{p\_out} + S \cdot Res \text{ 4 x 1}\]

\[Cov_{s\_out} = (I - S \cdot M) \cdot Cov_p \text{ 4 x 4}\]

\[dist_{Res\_2} = Res; \ast (I_{Cov\_Res}) \ast Res\]
Module Name: first_smooth_shlx

Calling Module: Run_Tracker_8

Called Modules:
- abtrack_init
- track_init_2
- track_init_c

Inputs:
- type: Track filter type (1:3 => alpha beta, Uncoupled Kalman, and extended Kalman)
- Z_m_last: Measured vector (last update)
- Z_m: Measured vector (current)
- Cov_m_vec: Meas Covariance array (current)
- dT: Time interval (sec) from last update

Outputs:
- Z_s_out: Smoothed state vector
- Cov_s_vec: Smoothed covariance array

Globals: None

Description:
Performs smoothing of tentative tracks

Equations:
- alpha beta: see abtrack_init
- Uncoupled Kalman: see track_init_2
- Extended Kalman: see track_init_c
Module Name: abtrack_init

Calling Module: first_smooth_shlx

Called Modules: None

Inputs:

(x1, y1) Last updated track position
(x2, y2) Current track position
dT12 Time interval (sec)

Outputs:

(x_s, y_s) Smoothed track position
(vx_s, vy_s) Smoothed tracked velocity
ang_trk Track angle (radians)

Globals: None

Description:

Does a two point initialization of an alpha beta tracker using current and last track positions and their time interval.

Equations:

\[ D_x = (x_2 - x_1) \]
\[ x_s = x_2 \]
\[ v_x_s = \frac{D_x}{dT_12} \]
\[ D_y = (y_2 - y_1) \]
\[ y_s = y_2 \]
\[ v_y_s = \frac{D_y}{dT_12} \]
\[ \text{ang}_{\text{trk}} = \arctan \left( \frac{D_y}{D_x} \right) \]
Module Name: track_init_2
Calling Module: first_smooth_shlx
Called Modules: None

Inputs:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delt</td>
<td>Time (sec) between current and last update</td>
</tr>
<tr>
<td>z_pos_1</td>
<td>Last position component update (x or y)</td>
</tr>
<tr>
<td>z_pos_2</td>
<td>Current position component (x or y)</td>
</tr>
<tr>
<td>var_pos_m_2</td>
<td>Measurement variance of position component</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_s_in</td>
<td>Smoothed state vector (3 x 1)</td>
</tr>
<tr>
<td>P_s_in</td>
<td>Smoothed covariance matrix (3 x 3)</td>
</tr>
</tbody>
</table>

Globals: None

Description:

Does a two point initialization of an uncoupled Kalman tracker. This routine is applied separately to both the x and y components of target motion.

Equations:

\[
\begin{align*}
\text{pos}_s &= \text{z}_\text{pos}_2 \\
\text{vel}_s &= (\text{z}_\text{pos}_2 - \text{z}_\text{pos}_1)/\text{Delt} \\
\text{Z}_s_\text{in} &= [\text{pos}_s; \text{vel}_s; 0] \\
\text{P}_s_\text{in} &= \begin{bmatrix}
\text{P}_{x11} & \text{P}_{x12} & 0 \\
\text{P}_{x12} & \text{P}_{x22} & 0 \\
0 & 0 & 0
\end{bmatrix}
\end{align*}
\]

where

\[
\begin{align*}
\text{P}_{x11} &= \text{var}_\text{pos}_m_2 \\
\text{P}_{x12} &= \text{P}_{x11}/\text{Delt} \\
\text{P}_{x22} &= \text{P}_{x11}/(\text{Delt})^2
\end{align*}
\]
Module Name: track_init_c

Calling Module: first_smooth_shlx

Called Modules: None

Inputs:
- Delt: Time (sec) between current and last update
- (x1, y1): Last position update
- (x2, Y2): Current position
- sig_xx: Measurement covariance xx comp
- sig_yy: Measurement covariance yy comp
- sig_xy: Measurement covariance xy comp

Outputs:
- Z_s_in: 4 x 1 Smoothed state vector
- P_s_in: 4 x 4 Smoothed covariance matrix

Globals:
- Time (sec) between current and last update
- Last position update
- Current position
- Measurement covariance xx comp
- Measurement covariance yy comp
- Measurement covariance xy comp

Description:

Does a two point initialization to be used with the extended Kalman track filter.

Equations:

Let $T = \text{Delt}$, $T^2 = T^2$

$$\begin{aligned}
sig_x &= \sqrt{\text{sig}_{xx}}, \quad sig_y = \sqrt{\text{sig}_{yy}} \\
pos_{sx} &= x_2, \quad pos_{sy} = y_2; \text{ smoothed positions} \\
vel_{sx} &= (x_2 - x_1)/T, \quad vel_{sy} = (y_2 - y_1)/T; \text{ smoothed velocities} \\
Z_s_{in} &= [\text{pos}_{sx}; \text{vel}_{sx}; \text{pos}_{sy}; \text{vel}_{sy}] \\
P_{s_{in}} &= \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\
P_{12} & P_{22} & P_{23} & P_{24} \\
P_{13} & P_{23} & P_{33} & P_{34} \\
P_{14} & P_{24} & P_{34} & P_{44} \end{bmatrix}
\end{aligned}$$

where

$$\begin{aligned}
P_{11} &= \text{sig}_{xx}, \quad P_{12} = 1.5 * P_{11}/T, \quad P_{13} = \text{sig}_{xy}; \quad P_{14} = 1.5 * \text{sig}_x * \text{sig}_y/T \\
P_{22} &= 6.5 P_{11}/T^2; \quad P_{23} = P_{14}, \quad P_{24} = \text{sig}_{xy}/T^2 \\
P_{33} &= \text{sig}_{yy}, \quad P_{34} = 1.5 * P_{33}/T \\
P_{44} &= 6.5 * P_{33}/T^2
\end{aligned}$$
Module Name: Assign_Cov_manu
Calling Module: Run_Tracker_8
Called Modules: get_def_Cov_manu

Inputs:
- trk_type
  - Track filter type
- scan
  - Current scan
- manu_rep
  - Maneuver anticipation status flag
- rep_source
  - Maneuver anticipation interval (scan lo, scan hi)

Outputs:
- Cov_man
  - Maneuver covariance matrix
- Cov_man_x
- Cov_man_y

Globals:
- Cov_man20
- Cov_man30
- Cov_man_acc_x
- Cov_man_acc_y
- Turn_int

Description:
Sets up the maneuver covariance matrix. If a maneuver has been anticipated for the current scan then maneuver noise is applied. Otherwise a small value (essentially zero) is loaded into the matrix.

Equations:

\[
\text{Accel}_{\text{max}} = \text{Maximum maneuver acceleration (km/s}^2\text{)}
\]

\[
\text{var}_{\text{acc}} = (\text{Accel}_{\text{max}})^2
\]

\[
\text{var}_{\text{vel}} = (\text{Accel}_{\text{max}} \times \text{scan}\_\text{period})^2
\]

\[
\text{var}_{\text{pos}} = (0.5 \times \text{Accel}_{\text{max}} \times (\text{scan}\_\text{period})^2)^2
\]

\[
\text{Cov}_\text{man} = \begin{bmatrix}
\text{var}_{\text{pos}} & 0 & 0 \\
0 & \text{var}_{\text{vel}} & 0 \\
0 & 0 & \text{var}_{\text{acc}}
\end{bmatrix}; \text{trk\_type} \neq 3
\]
\[ Cov_{\text{man}} = \begin{bmatrix} \text{var\_pos} & 0 & 0 & 0 \\ 0 & \text{var\_vel} & 0 & 0 \\ 0 & 0 & \text{var\_pos} & 0 \\ 0 & 0 & 0 & \text{var\_vel} \end{bmatrix}; \text{trk\_type}=3 \]

\[ Cov_{\text{man\_x}} = Cov_{\text{man\_y}} = Cov_{\text{man}} \]
Module Name: get_def_Cov_manu

Calling Module: Assign_Cov_manu

Called Modules: None

Inputs:
- trk_type: Track filter type

Outputs:
- Cov_man: Maneuver covariance matrix
- Cov_man_x
- Cov_man_y

Globals:
- Cov_man20
- Cov_man30

Description:
Loads default values into maneuver covariance matrices.

Equations:

\[
\begin{align*}
\text{Cov\_man} &= \text{Cov\_man20}; & \text{trk\_type} \neq 3 \\
\text{Cov\_man} &= \text{Cov\_man30}; & \text{trk\_type} = 3 \\
\text{Cov\_man\_x} &= \text{Cov\_man\_y} = \text{Cov\_man};
\end{align*}
\]

where

\[
\begin{align*}
\text{Cov\_man20} &= \text{le}-10 * \text{eye} \ (3) \\
\text{Cov\_man30} &= \text{le}-6 * \text{eye} \ (4)
\end{align*}
\]
Module Name: Get_Semiaxes_3

Calling Module: Run_Tracker_8

Called Modules: E_Gate_Semiaxes_3
get_sig_track
Scale_Shadow_gates

Inputs:

In_Shadow Current in shadow flag (1 => yes)
In_Shadow_last Last update in shadow flag
Nshadow_dwells_last Number of successive dwells in shadow as of last update
gate_case Gate type flag (0:3)
trk_type Track filter type (1:3)
Mult Gate size multiplier
Accel_max Maximum assumed acceleration
dT Time (sec) since last update
ang_pred Angle (rad) of predicted point wrt radar cs
ang_trk Angle (rad) of track wrt radar cs
sig_rng_km_m Range measurement error (km)
sig_crg_km_m Cross range measurement error (km)
Z_s_last Smoothed state vector (last update)
Z_p Prediction state vector (current)
Cov_p_vec Prediction covariance array (current)
semi_rng_T_last Along range semi axes (last)
semi_crg_T_last Cross range semi axes (last)
semi_trk_T_last Along track semi axes (last)
semi_ctrk_T_last Cross track semi axes (last)
semi_max Maximum allowed semi axis size

Outputs:

semi_rng_T Along range semi axes (current)
semi_crg_T Cross range semi axes (current)
semi_trk_T Along track semi axes (current)
semi_ctrk_T Cross track semi axes (current)

Globals: None

Description:

Computes the semi axes for the two orientations of elliptical gates used. The first type is oriented along and across range and the second is oriented along and across track. Tracks that are in a shadow have their gate sizes frozen. Otherwise, track gates sizes are determined as a function of the gate case and track filter type selected. This function is done by the routine E_Gate_Semiaxes_3.
Module Name: E_Gate_Semiaxes_3

Calling Module: Get_Semiaxes_3

Called Modules: Rotate_xy2xyp

Inputs:

- gate_case: Gate type flag (0:3)
- trk_type: Track filter type (1:3)
- Mult: Gate size multiplier
- Z_p: Prediction state vector (current)
- Cov_p_vec: Prediction covariance array (current)
- ang_pred: Angle (rad) of predicted point wrt radar cs
- sig_rng: Range measurement error (km)
- sig_crng: Cross range measurement error (km)
- ang_trk: Angle (rad) of track wrt radar cs
- sig_trk: Along track measurement error (km)
- sig_ctrk: Cross track measurement error (km)
- semi_max: Maximum allowed semi axes size

Outputs:

- semi_rng: Along range semi axes (km)
- semi_crng: Cross range semi axes (km)
- semi_trk: Along track semi axes (km)
- semi_ctrk: Cross track semi axes (km)

Globals: None

Description:

Computes semi axes for elliptical gates oriented along/cross range, and oriented along/cross track. Results depend on which of three gate cases are chosen (cases 1 and 2 require a Kalman filter). In each case the along/cross track gate sizes are computed as scaled versions of the along/cross track measurement errors. However, the along/cross range oriented gate sizes are case dependent. Gate case 1 combines in an rss fashion (1) range and cross range measurement errors from the last track update, and, (2) current prediction covariance estimates in x and y, projected onto the range/cross range axes. The composite gate is formed by multiplying this result by a scale factor. Gate case 2 uses only the scaled prediction covariance estimates in x and y to form the gate. Gate case 3 uses the scaled along/cross range measurement errors to form this gate.

Equations:
The quantities sig_rng and sig_crng are measurement errors along range and cross range computed from the last report captured by a given track and stored in the track table. The quantities sig_trk and sig_ctrk are based on kinematical assumptions and are computed in get_sig_track.

alpha beta filter:

\[
\begin{bmatrix}
\text{semi_rng} \\
\text{semi_crng} \\
\text{semi_trk} \\
\text{semi_ctrk}
\end{bmatrix} =
\begin{bmatrix}
\text{sig_rng} \\
\text{sig_crng} \\
\text{sig_trk} \\
\text{sig_ctrk}
\end{bmatrix} \times \left\{ \begin{array}{ll}
1.5 \text{Mult}, & (\text{gate_case} < 3) \\
\text{Mult}, & (\text{gate_case} = 3)
\end{array} \right.
\]

Kalman filters:

Get prediction uncertainty along x and y directions \((\sigma_{px}, \sigma_{py})\) from the covariance array Cov_p_vec.

\((\text{gate_case} = 1)\)

\[\Delta \theta = (\text{ang_pred} - \text{ang_trk})\]
Compute projections of \(\sigma_{px}\) and \(\sigma_{py}\) onto the range-cross range axes.

\[
\begin{bmatrix}
\sigma_{px'} \\
\sigma_{py'}
\end{bmatrix} =
\begin{bmatrix}
\cos(\Delta \theta) & \sin(\Delta \theta) \\
-\sin(\Delta \theta) & \cos(\Delta \theta)
\end{bmatrix}\begin{bmatrix}
\sigma_{px} \\
\sigma_{py}
\end{bmatrix}
\]

Form the “root sum square” (Rss) of \(\sigma_{px'}\) and \(\sigma_{py'}\) with the measurement errors

semi_rng = Mult * \(\sqrt{(\sigma_{px'})^2 + (\text{sig_rng})^2}\)

semi_crng = Mult * \(\sqrt{(\sigma_{px'})^2 + (\text{sig_crng})^2}\)

semi_trk = Mult * \(\text{sig_trk}\)

semi_ctrk = Mult * \(\text{sig_ctrk}\)

\((\text{gate_case} = 2)\)

\[\theta = \text{ang_trk}\]

Compute projections of \(\sigma_{px}\) and \(\sigma_{py}\) onto track-cross track axes.

\[
\begin{bmatrix}
\sigma_{px''} \\
\sigma_{py''}
\end{bmatrix} =
\begin{bmatrix}
\cos(\theta) & \sin(\theta) \\
-\sin(\theta) & \cos(\theta)
\end{bmatrix}\begin{bmatrix}
\sigma_{px} \\
\sigma_{py}
\end{bmatrix}
\]

51
\[
\begin{bmatrix}
\text{semi}_\text{rng} \\
\text{semi}_\text{crng} \\
\text{semi}_\text{trk} \\
\text{semi}_\text{ctrk}
\end{bmatrix}
= \text{Mult} \ast
\begin{bmatrix}
\text{sig}_\text{rng} \\
\text{sig}_\text{crng} \\
\sigma_{px} \\
\sigma_{py}
\end{bmatrix}
\]

\text{gate}_\text{case} = 3

\[
\begin{bmatrix}
\text{semi}_\text{rng} \\
\text{semi}_\text{crng} \\
\text{semi}_\text{trk} \\
\text{semi}_\text{ctrk}
\end{bmatrix}
= \text{Mult} \ast
\begin{bmatrix}
\text{sig}_\text{rng} \\
\text{sig}_\text{crng} \\
\text{sig}_\text{trk} \\
\text{sig}_\text{ctrk}
\end{bmatrix}
\]
Module Name: get_sig_track

Calling Module: Get_Semiaxes_3

Called Modules: None

Inputs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_s</td>
<td>Smoothed state vector</td>
</tr>
<tr>
<td>Accel_max</td>
<td>Maximum allowed acceleration (km/s*s)</td>
</tr>
<tr>
<td>dT</td>
<td>Time (sec) since last update</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sig_trk_km</td>
<td>Along track uncertainty (km)</td>
</tr>
<tr>
<td>sig_ctrk_km</td>
<td>Cross track uncertainty (km)</td>
</tr>
</tbody>
</table>

Globals: None

Description:

Computes along/cross track uncertainties. The along track uncertainty is computed as
0.5*A*(dT)^2 with the along track acceleration A assumed bounded by 0.5g. The across
track uncertainty is assumed to be due to constant speed turning only.

Equations:

Define the following quantities:

\[ \text{Acc}_{\text{at}} = \text{along track acceleration} \leq \text{Accel} \_\text{max} \]
\[ \text{V} \_\text{kmps} = \text{estimated track speed} \]
\[ \rho_{\text{km}} = \text{radius of curvature of turn (km)} \]
\[ \text{Ang} = \text{angle of turn voer time } dT \text{ (radians)} \]
\[ \rho_{\text{km}} = (\text{V} \_\text{kmps})^2/\text{Accel} \_\text{max} \]
\[ \text{ang} = \text{V} \_\text{kmps} \times dT/\rho_{\text{km}} \]
\[ \text{sig}\_\text{trk} \_\text{km} = 0.5 \text{Acc}_{\text{at}} (dT)^2 \]
\[ \text{sig}\_\text{ctrk} \_\text{km} = \rho_{\text{km}} (1 - \cos (\text{Ang})) \]
Module Name: Scale_Shadow_Gates

Calling Module: Get_Semiaxes_3

Called Modules: None

Inputs:
- gate_scale: Gate size multiplier
- semi_rng_T_in: Along range semi axis (input)
- semi_crng_T_in: Cross range semi axis (input)
- semi_trk_T_in: Along track semi axis (input)
- semi_ctrk_T_in: Cross track semi axis (input)

Outputs:
- semi_rng_T_out: Along range semi axis (output)
- semi_crng_T_out: Cross range semi axis (output)
- semi_trk_T_out: Along track semi axis (output)
- semi_ctrk_T_out: Cross track semi axis (output)

Globals: None

Description:
Multiplies input gate semi axes by a scale factor.
Module Name: Rotate_xy2xpyp

Calling Module: E_Gate_Semiaxes_3

Called Modules: None

Inputs:

- ang_rad: Rotation angle (rad)
- Z: Input two component vector

Outputs:

Rotated two component vector

Globals:

None

Description:

Rotates the two component vector Z through angle ang_rad.

Equations:

\[ \theta = \text{ang\_rad} \quad \text{and} \]

Let \[ \text{Rot} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \]

\[ Z_p = \text{Rot} \times Z \]
Module Name: Test_E_Gate

Calling Module: Run_Tracker_8

Called Modules: Rotate_xy2xyyp

Inputs:

- $Z_m$: Measurement state vector (x;y;pDop)
- $Z_p$: Prediction state vector (x;vx;ax;y;vy;ay)
- ang_rotn: Rotation angle wrt radar cs (radians)
- semi_along: Semi axis length along rotated x
- semi_across: Semi axis length along rotated y

Outputs:

- In_E_Gate: Inclusion test flag (1 => inclusion)

Globals:

None

Description:

Tests if measured point $Z_m$ lies within an ellipse which is (1) oriented at an angle "ang_rotn" with respect to the radar coordinate system, (2) centered on the predicted position, $Z_p$, and, (3) has semi axes of length semi_along and semi_across. Returns a value of zero if test fails, and a value of one if test passes.

Equations:

Let $(X_m, Y_m)$ and $(X_p, Y_p)$ denote the measured predicted points, respectively. Transform the predicted-measurement position error into the local coordinate system of the ellipse, centered on the predicted point. The transformed residual errors $(\Delta X', \Delta Y')$ are

$$\begin{bmatrix} \Delta X' \\ \Delta Y' \end{bmatrix} = \text{Rot} \begin{bmatrix} X_p - X_m \\ Y_p - Y_m \end{bmatrix}$$

where

$$Y_e = \text{semi_across} \cdot \sqrt{1 - (\Delta X'/\text{semi_along})^2}$$

The condition for inclusion of $(X_m, Y_m)$ within the ellipse is

$$|\Delta Y'| \leq Y_e \quad \text{and} \quad |\Delta X'| \leq \text{semi_along}$$
Module Name: Test_Smile_Gate

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

- \( T_s \): Scan period (sec)
- \( Z_m \): Measurement state vector \((x;y;pDop)\)
- \( Z_p \): Prediction state vector \((x;vx;ax;y;vy;ay)\)
- \( Z_s_{\text{last}} \): Smoothed state vector (last update)
- \( x_{\text{last}}_{\text{km}} \): \(x\) position (last update)
- \( y_{\text{last}}_{\text{km}} \): \(y\) position (last update)
- \( ag_{HI} \): Maximum turn acceleration (g units)
- \( \text{semi}_\text{rng} \): Measurement ellipse semi axis along range
- \( \text{semi}_\text{crng} \): Measurement ellipse semi axis cross range

Outputs:

- \( \text{In}_{\text{gate}}_{\text{m}} \): Inclusion test flag (1 => included)

Globals: None

Description:

Tests if measurement ellipse intersects centripetal maneuver “smile” shaped gate.

Equations:

The smile gate envelope is defined by two equations that are parameterized by the time \( t \), ranging from the current report time to one scan period later. Let \( v \) denote the constant target speed and \( \rho \) be the radius of curvature of the turn. Then

\[
\rho = \frac{v^2}{9.8 \times ag_{HI}} \text{ and } \\
\text{Ang} = \frac{v (T_s - t)}{\rho}.
\]

With these the equations for the maneuver envelope in the local track coordinate system, centered on the last smoothed position, are

\[
x' = vt + \rho \sin(\text{ang}) \\
y' = \rho (1 - \cos(\text{ang}))
\]

The gate test is performed by:
(1) Transforming the boundary points \((x'_s, y'_s)\) to the radar coordinate system

\[
\begin{bmatrix}
  x'_s \\
  y'_s
\end{bmatrix} = \begin{bmatrix}
  x_{last\_km} \\
  y_{last\_km}
\end{bmatrix} + \begin{bmatrix}
  \cos(\theta_{tk}) & -\sin(\theta_{tk}) \\
  \sin(\theta_{tk}) & \cos(\theta_{tk})
\end{bmatrix} \begin{bmatrix}
  x'_s \\
  y'_s
\end{bmatrix}
\]

where \(\theta_{tk}\) denotes the instantaneous track angle.

(2) Transforming \((x_s, y_s)\) to local coordinate system centered on the measured point and oriented at the angle of the measured point \(\theta_m\).

\[
\begin{bmatrix}
  \Delta X'_s \\
  \Delta Y'_s
\end{bmatrix} = \begin{bmatrix}
  \cos(\theta_m) & \sin(\theta_m) \\
  -\sin(\theta_m) & \cos(\theta_m)
\end{bmatrix} \begin{bmatrix}
  x_s - x_m \\
  y_s - y_m
\end{bmatrix}
\]

(3) Test each maneuver envelope point for inclusion within the measurement ellipse

\[
|\Delta X'_s| \leq semi\_rng \\
|\Delta Y'_s| \leq Y_e
\]

\[
Y_e = semi\_crng \sqrt{1 - (\Delta X''/semi\_rng)^2}
\]
Module Name: Pred_Shadow_Test
Calling Module: Run_Tracker_8
Called Modules: get_def_Cov_manu
predict_shlx_x
Test_In_Shadow

Inputs:
- trk_type: Type of tracking filter (1:3)
- Z_s_last: Smoothed state vector (last update)
- Cov_s_last_vec: Smoothed covariance array (last update)
- dT: Time interval (sec) since last update

Outputs:
- Z_p: Prediction state vector
- In_Shadow: Shadow status flag (1 => in shadow)

Globals:
- Cov_man20 Cov_man30
- x_sh_LO x_sh_HI y_sh_LO y_sh_HI

Description:
Tests if predicted position falls within a shadow region. Sets flag to 1 if in shadow. [The shadow zones were specified as a set of rectangular tiles by the scenario generator. The array x_sh_LO, x_sh_HI, y_sh_LO, and, y_sh_HI specify low and high positions of each time.]
Module Name: Test_In_Shadow

Calling Module: Shadow_Test

Called Modules: None

Inputs:
- Z_p: Prediction state vector (x;vx;ax;y;vy;ay)
- x_sh_LO: Array specifying low x position of shadow tiles
- x_sh_HI: Array specifying high x position of shadow tiles
- y_sh_LO: Array specifying low y position of shadow tiles
- y_sh_HI: Array specifying high y position of shadow tiles

Outputs:
- In_Shadow: In shadow status flag (1 => in shadow)

Globals: None

Description:
Tests if predicted position falls within any of the shadow tiles that were specified by the scenario generator.
Module Name: Discrete_Test

Calling Module: Run_Tracker_8

Called Modules: E_Discrete_Test

Inputs:

- \( Z_p \) Prediction state vector
- gate_case Gate case flag (1:3)
- ang_pred Angle (rad) of predicted point
- semi_rng_T Semi axis length of along range ellipse
- semi_crng_T Semi axis length of cross range ellipse
- ang_trk Angle (rad) of track
- semi_trk_T Semi axis length of along track ellipse
- semi_ctrk_T Semi axis length of cross track ellipse

Outputs:

- In_Discrete Discrete capture flag (1 => discrete present)

Globals: None

Description:

Tests if discrete point falls within either of two gates oriented along/across range or along/cross track.

Equations:

\((\text{gate\_case} = 1)\)

Each discrete point is tested for inclusion in a single ellipse, centered on the predicted point, and oriented along range/cross range. The semi axes are given by semi_rng_T and semi_crng_T.

\((\text{gate\_case} > 1)\)

Each discrete point is tested for inclusion if either of two ellipses, both centered on the predicted point. The first ellipse is the same as defined above. The second ellipse is oriented along track/cross track and has semi axes given by semi_trk_T and semi_ctrk_T.
<table>
<thead>
<tr>
<th>Module Name:</th>
<th>E_Discrete_Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling Module:</td>
<td>Discrete_test</td>
</tr>
<tr>
<td>Called Modules:</td>
<td>Test_E_Gate</td>
</tr>
</tbody>
</table>

**Inputs:**
- Z_p
- ang_rotn
- semi_along
- semi_across

- Prediction state vector \((x;vx;ax;y;vy;ay)\)
- Rotation angle (rad) of ellipse gate wrt radar cs
- Semi axis length of gate along rotated x
- Semi axis length of gate along rotated y

**Outputs:**
- In_Discrete

- Discrete inclusion flag \((1 \Rightarrow \text{discrete in})\)

**Globals:**
- Ndiscrete X_D_Vec Y_D_Vec

**Description:**
Tests if any of a set of discrete points falls within an elliptical gate: (1) centered on the predicted point, (2) oriented at an angle "ang_rotn" wrt radar coordination system, (3) having semi axes lengths of semi_along along the rotated x axis and of length semi_across along the rotated y axis.

The discrete locations were specified by the scenario generator and given here by the arrays X_D_Vec and Y_D_Vec.
Module Name: TR_Assoc_Max_T

Calling Module: Run_Tracker_8

Called Modules: cmp_track_age

Inputs:
- Corr: Binary correlation matrix (trks, reps)
- Dist: Distance matrix (trks, reps)
- NAT_in: Number of active tracks in Table_T
- Nrep_in: Number of reports from current scan
- scan: Current scan

Outputs:
- Update: Binary array of updated tracks (1 => updated)
- unAssoc: Binary array of unassociated tracks
- unused: Binary array of unused reports
- Asgn_Reps: Array of reports assigned to each track

Globals: Table_T

Description:

Assigns unique reports to corresponding track. If a report is common to multiple tracks then it is assigned to the oldest track.
Module Name: TR_Assoc_Min_D

Calling Module: Run_Tracker_8

Called Modules: cmp_track_age

Inputs:
- Corr
- Dist
- NAT_in
- Nrep_in
- scan

  Binary correlation matrix (trks, reps)
  Distance matrix (trks, reps)
  Number of active tracks in Table_T
  Number of reports from current scan
  Current scan

Outputs:
- Update
- unAssoc
- unused
- Asgn_Reps

  Binary array of updated tracks (1 => updated)
  Binary array of unassociated tracks
  Binary array of unused reports
  Array of reports assigned to each track

Globals: Table_T

Description:

Assigns unique reports to corresponding track. If multiple reports are common to a given track then the closest is assigned to the track.
Module Name:           cmp_track_age
Calling Module:        TR_Assoc_Max_T
Called Modules:        None

Inputs:
    trk         Track index into Table_T
    scan        Current scan

Outputs:
    age        Track age since first became firm

Globals:               Table_T

Description:

Computes age of a track in scan units. Age is defined as time since track became firm.
Module Name: get_track_ang
Calling Module: Run_Tracker_8
Called Modules: None
Inputs:
- Z_p
- Z_s_last
Prediction state vector
Smoothed state vector (last update)
Outputs:
- ang_trk
Angle of track (radians)
Globals: None
Description:
Computes track angle in radians with respect to the radar coordinate system.
Module Name: get_pred_ang
Calling Module: Run_Tracker_8
Called Modules: None
Inputs: 
  $Z_p$  Prediction state vector
Outputs: 
  ang_pred  Prediction of Angle
Globals: None
Description:

Computes angle of predicted point with respect to the radar coordinate system.
Module Name: update_error

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:
- N_Hits: Number of times track captured a report
- Z_tru: True state vector
- Z_p: Prediction state vector
- TID: Track ID number

Outputs:
- mean_error: Position error (predicted – true) averaged over N_Hits
- sigma_error: Standard deviation of position error

Globals:
- SUM1_ERROR
- SUM2_ERROR

Description:
Computes running mean and standard deviation of position error between predicted and true values as a function of track ID number. Stores running first and second moments in global buffers SUM1_ERROR and SUM2_ERROR.
Module Name: Load_P_DET
Calling Module: Run_Tracker_8
Called Modules: None

Inputs:
- Ntk_max: Number of tracks in P_DET buffer
- scan: Current scan index
- N_HITS: Number of captures for track
- N_EVENTS: Number of captures plus misses for track

Outputs: None

Globals: Table_T_P_DET

Description:
Computes fraction of captures for specific track over its evolution. Stores results in global buffer P_DET.
Module Name: Load_dBuf_4
Calling Module: Run_Tracker_8
Called Modules: get_Table_Trk_4

Inputs:
  Ntrk  Current number of tracks
  dBuf_last_cnt  Number of dropped HP tracks in buffer dBuf at last scan

Outputs:
  dBuf_cnt  Number of dropped HP tracks in buffer dBuf currently

Globals:
  dBuf
  Table_T
  DROPPED TENT FIRM
  DP LP

Description:

Updates the high priority dropped track buffer each scan. The buffer dBuf has three components: (time, x position, y position)
Module Name: plot_dBuf

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:
- dBuf_new
- plot_vec

Outputs: None

Globals: dBuf

Description:
Plots dropped high priority tracks each scan.