trackerTOMHT

Multi-hypothesis, multi-sensor, multi-object tracker

Description

The `trackerTOMHT` System object™ is a multi-hypothesis tracker capable of processing detections of many targets from multiple sensors. The tracker initializes, confirms, predicts, corrects, and deletes tracks. Inputs to the tracker are detection reports generated by `objectDetection`, `fusionRadarSensor`, `irSensor`, or `sonarSensor` objects. The tracker estimates the state vector and state vector covariance matrix for each track. The tracker assigns detections based on a track-oriented, multi-hypothesis approach. Each detection is assigned to at least one track. If the detection cannot be assigned to any track, the tracker creates a track.

Any new track starts in a *tentative* state. If enough detections are assigned to a tentative track, its status changes to *confirmed*. If the detection already has a known classification (the `ObjectClassID` field of the returned track is nonzero), that track is confirmed immediately. When a track is confirmed, the multi-object tracker considers the track to represent a physical object. If detections are not assigned to the track within a specifiable number of updates, the track is deleted. For an overview of how the tracker functions, see Algorithms.

To track objects using the multi-hypothesis tracker:

1. Create the `trackerTOMHT` object and set its properties.
2. Call the object with arguments, as if it were a function.

To learn more about how System objects work, see What Are System Objects?

Creation

Syntax

```
tracker = trackerTOMHT
tracker = trackerTOMHT(Name,Value)
```

Description

`tracker = trackerTOMHT` creates a `trackerTOMHT` System object with default property values.

example

`tracker = trackerTOMHT(Name,Value)` sets properties for the multi-object tracker using one or more name-value pairs. For example, `trackerTOMHT('FilterInitializationFcn',@initcvukf,'MaxNumTracks',100)` creates a multi-object tracker that uses a constant-velocity, unscented Kalman filter and allows a maximum of 100 tracks. Enclose each property name in quotes.

Properties

expand all

Unless otherwise indicated, properties are *nontunable*, which means you cannot change their values after calling the object. Objects lock when you call them, and the `release` function unlocks them.

If a property is *tunable*, you can change its value at any time.

For more information on changing property values, see System Design in MATLAB Using System Objects.

`TrackerIndex` — Unique tracker identifier
`0` (default) | nonnegative integer

`FilterInitializationFcn` — Filter initialization function
`@initcvekf` (default) | function handle | character vector

`MaxNumTracks` — Maximum number of tracks
`100` (default) | positive integer

`MaxNumSensors` — Maximum number of sensors
`20` (default) | positive integer

`MaxNumDetections` — Maximum number of detections
`Inf` (default) | positive integer

`OOSMHandling` — Handle out-of-sequence measurement (OOSM)
`'Terminate'` (default) | `'Neglect'`

`StateParameters` — Parameters of track state reference frame
`struct([])` (default) | `struct array`

`MaxNumHypotheses` — Maximum number of hypotheses to maintain
`5` (default) | positive integer

`MaxNumTrackBranches` — Maximum number of track branches per track
`3` (default) | positive scalar

`MaxNumHistoryScans` — Maximum number of scans maintained in the branch history
`4` (default) | positive integer

`AssignmentThreshold` — Detection assignment threshold
`30*[0.3 0.7 1 Inf]` (default) | positive scalar | 1-by-3 vector of positive values | 1-by-4 vector of positive values

`ConfirmationThreshold` — Minimum score required to confirm track
`20` (default) | positive scalar

`DeletionThreshold` — Maximum score drop for track deletion
`-7` (default) | scalar

`DetectionProbability` — Probability of detection used for track score
`0.9` (default) | positive scalar between 0 and 1

`FalseAlarmRate` — Probability of false alarm used for track score
`1e-6` (default) | scalar

`Beta` — Rate of new tracks per unit volume
`1` (default) | positive scalar

`Volume` — Volume of sensor measurement bin
`1` (default) | positive scalar

`MinBranchProbability` — Minimum probability required to keep track
`.001` (default) | positive scalar

`NScanPruning` — N-scan pruning method
`'None'` (default) | `'Hypothesis'`

`HasCostMatrixInput` — Enable cost matrix input
`false` (default) | `true`

`HasDetectableBranchIDsInput` — Enable input of detectable branch IDs
`false` (default) | `true`

`OutputRepresentation` — Track output method
`'Tracks'` (default) | `'Hypothesis'` | `'Clusters'`

`HypothesesToOutput` — Indices of hypotheses to output
`1` (default) | positive integer | array of positive integers

`NumTracks` — Number of tracks maintained by tracker
nonnegative integer

`NumConfirmedTracks` — Number of confirmed tracks
nonnegative integer

Usage

To process detections and update tracks, call the tracker with arguments, as if it were a function (described here).

Syntax

```
confirmedTracks = tracker(detections,time)
confirmedTracks = tracker(detections,time,costMatrix)
confirmedTracks = tracker( __ ,detectableBranchIDs)
[confirmedTracks,tentativeTracks,allTracks] = tracker( __ )
[confirmedTracks,tentativeTracks,allTracks,analysisInformation] = tracker( __ )
```

Description

`confirmedTracks = tracker(detections,time)` returns a list of confirmed tracks that are updated from a list of detections, `detections`, at the update time, `time`. Confirmed tracks are corrected and predicted to the update time.

`confirmedTracks = tracker(detections,time,costMatrix)` also specifies a cost matrix, `costMatrix`.

To enable this syntax, set the `HasCostMatrixInput` property to `true`.

`confirmedTracks = tracker( __ ,detectableBranchIDs)` also specifies a list of expected detectable branches, `detectableBranchIDs`.

To enable this syntax, set the `HasDetectableBranchIDsInput` property to `true`.

`[confirmedTracks,tentativeTracks,allTracks] = tracker( __ )` also returns a list of tentative tracks, `tentativeTracks`, and a list of all tracks, `allTracks`.

`[confirmedTracks,tentativeTracks,allTracks,analysisInformation] = tracker( __ )` also returns information, `analysisInformation`, useful for track analysis.

Input Arguments

expand all

`detections` — Detection list
cell array of `objectDetection` objects

`time` — Time of update
scalar

`costMatrix` — Cost matrix
real-valued N-by-M matrix

`detectableBranchIDs` — Detectable branch IDs
real-valued M-by-1 vector | real-valued M-by-2 matrix

Output Arguments

expand all

`confirmedTracks` — Confirmed tracks
array of `objectTrack` objects | array of structures

`tentativeTracks` — Tentative tracks
array of `objectTrack` objects | array of structures

`allTracks` — All tracks
array of `objectTrack` objects | array of structures

`analysisInformation` — Additional information for analyzing track updates
structure

Object Functions

To use an object function, specify the System object as the first input argument. For example, to release system resources of a System object named `obj`, use this syntax:

```
release(obj)
```

expand all

Specific to `trackerTOMHT`

Common to All System Objects

Examples

collapse all

Track Two Objects Using trackerTOMHT

Create the `trackerTOMHT` System object with a constant-velocity Kalman filter initialization function, `initcvkf`.

Open Live Script

```
tracker = trackerTOMHT('FilterInitializationFcn',@initcvkf, ...
    'ConfirmationThreshold',20, ...
    'DeletionThreshold',-7, ...
    'MaxNumHypotheses',10);
```

Update the tracker with two detections having nonzero `ObjectClassID`. The detections immediately create confirmed tracks.

```
detections = {objectDetection(1,[10;0],'SensorIndex',1, ...
    'ObjectClassID',5,'ObjectAttributes',{struct('ID',1)}); ...
    objectDetection(1,[0;10],'SensorIndex',1, ...
    'ObjectClassID',2,'ObjectAttributes',{struct('ID',2)})};
time = 2;
tracks = tracker(detections,time);
```

Find and display the positions and velocities.

```
positionSelector = [1 0 0 0; 0 0 1 0];
```

```
velocitySelector = [0 1 0 0; 0 0 0 1];
positions = getTrackPositions(tracks,positionSelector)
```

```
positions = 2×2

   10.0000         0
         0   10.0000
```

```
velocities = getTrackVelocities(tracks,velocitySelector)
```

```
velocities = 2×2

     0     0
     0     0
```

Algorithms

expand all

Tracker Logic Flow

Assignment Thresholds for Multi-Hypothesis Tracker

Data Precision

References

[1] Werthmann, J. R.. "Step-by-Step Description of a Computationally Efficient Version of Multiple Hypothesis Tracking." In *International Society for Optics and Photonics*, Vol. 1698, pp. 228-301, 1992.

[2] Blackman, S., and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artech House Radar Library, Boston, 1999.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using MATLAB® Coder™.

Version History

Introduced in R2018b

See Also

Functions

`getTrackPositions` | `getTrackVelocities`

Objects

`objectDetection` | `trackingKF` | `trackingEKF` | `trackingUKF` | `trackingCKF` | `trackingPF` | `trackingMSCEKF` | `trackingGSF` | `trackingIMM` | `trackingABF` | `objectTrack` | `fusionRadarSensor` | `sonarSensor` | `irSensor` | `trackerGNN`

Blocks

Track-Oriented Multi-Hypothesis Tracker