EARS Map Objects

version 0.2
MATLAB format specification

Christine Evers, c.evers@imperial.ac.uk

October 30, 2014

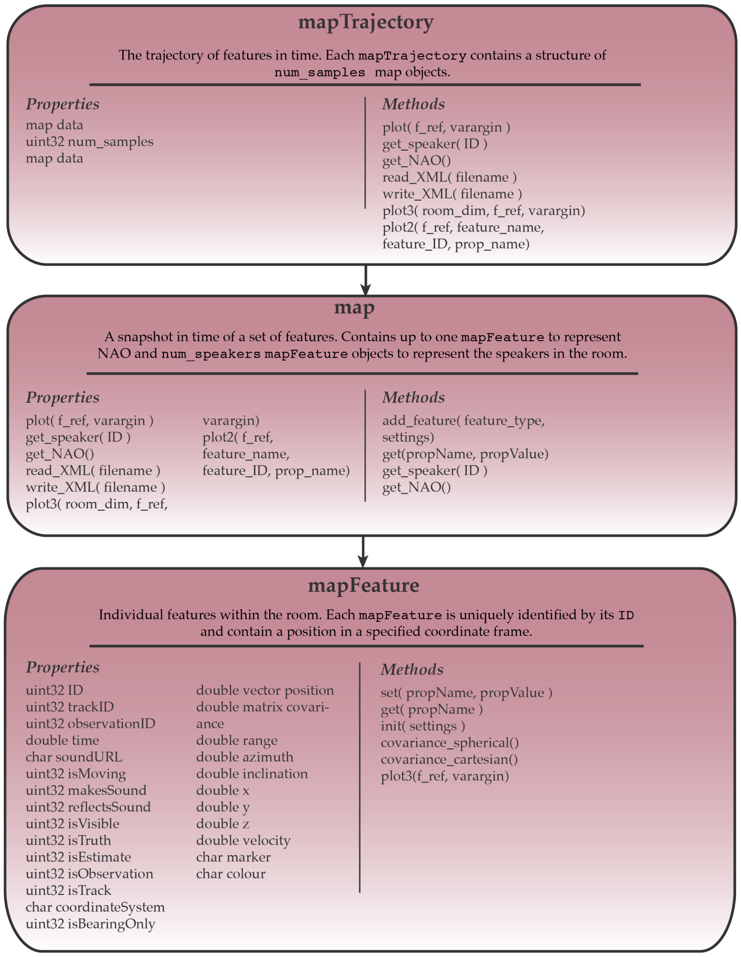
Contents

1 Introduction

A *map* is a snapshot in time, consisting of a collection of features within a room. *Map features* are defined as the robot, surrounding speakers or other sound sources, as well as features of the room such as wall or surrounding obstacles.

As time progresses, moving objects create trajectories in space.[1] This time-variance of maps is reflected using *map trajectories* that combine the information in each map with time.

The Embodied Audition for RobotS (EARS) map library therefore consists of three types of objects: `mapTrajectory`, `map` and `mapFeature`. The object hierarchy is summarised in Fig. 1.



mapTrajectory

The trajectory of features in time. Each `mapTrajectory` contains a structure of `num_samples` map objects.

Properties

map data
uint32 num_samples
map data

Methods

plot(f_ref, varargin)
get_speaker(ID)
get_NAO()
read_XML(filename)
write_XML(filename)
plot3(room_dim, f_ref, varargin)
plot2(f_ref, feature_name, feature_ID, prop_name)

map

A snapshot in time of a set of features. Contains up to one `mapFeature` to represent NAO and `num_speakers` `mapFeature` objects to represent the speakers in the room.

Properties

plot(f_ref, varargin)
get_speaker(ID)
get_NAO()
read_XML(filename)
write_XML(filename)
plot3(room_dim, f_ref, varargin)
plot2(f_ref, feature_name, feature_ID, prop_name)

Methods

add_feature(feature_type, settings)
get(propName, propValue)
get_speaker(ID)
get_NAO()

mapFeature

Individual features within the room. Each `mapFeature` is uniquely identified by its `ID` and contain a position in a specified coordinate frame.

Properties

uint32 ID
uint32 trackID
uint32 observationID
double time
char soundURL
uint32 isMoving
uint32 makesSound
uint32 reflectsSound
uint32 isVisible
uint32 isTruth
uint32 isEstimate
uint32 isObservation
uint32 isTrack
char coordinateSystem
uint32 isBearingOnly
double vector position
double matrix covariance
double range
double azimuth
double inclination
double x
double y
double z
double velocity
char marker
char colour

Methods

set(propName, propValue)
get(propName)
init(settings)
covariance_spherical()
covariance_cartesian()
plot3(f_ref, varargin)

Figure 1: Hierarchy of map library

[1] In this context, even stationary speakers could be considered time-varying as head movement and slight body rotations lead to a time-varying source-sensor geometry.

2 mapFeature

mapFeature objects are defined in a specified coordinate frame and are of a specified order / dimension. E.g., a Cartesian mapFeature of order = 3 natively stores its data as position $= \begin{bmatrix}\texttt{x} & \texttt{y} & \texttt{z}\end{bmatrix}^T$ with a covariance

$$\texttt{covariance} = \begin{bmatrix}\sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{xz}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & \sigma_{yz}^2 \\ \sigma_{zx}^2 & \sigma_{zy}^2 & \sigma_{zz}^2\end{bmatrix}$$

The spherical coordinates can be directly requested from Cartesian mapFeature objects using

2.1 Properties

Name	Data type	Description
order	uint32	Order / dimension of the object. Must be specified at initialisation, cannot be changed once set.
ID	uint32	Unique ID of the mapFeature within the map object.
trackID	uint32	Unique track ID of the object. This is used to identify the trajectory of a track over time. ID is typically assigned after the association stage in tracking
observationID	uint32	The ID of the EARO object used to update the mapFeature.
time	uint32	Time the mapFeature was last changed. Typically corresponds to the time stamp of the map.
soundURL	char	URL to either a wav file or location in memory.
isMoving	0 | 1	Binary flag indicating if object is moving (1) or stationary (0)
makesSound	0 | 1	Binary flag indicating if object is actively producing sound (1) or is silent (0)
reflectsSound	0 | 1	Binary flag indicating if object is reflecting sound (1) or is sound absorbent (0).
isVisible	0 | 1	Binary flag indicating if object is visible (1) or obscured (0).
isTruth	0 | 1	Binary flag indicating if object corresponds to ground truth.
isEstimate	0 | 1	Binary flag indicating if object corresponds to an estimate.
isObservation	0 | 1	Binary flag indicating if object corresponds to observations.
coordinateSystem	'cartesian' | 'spherical'	Coordinate system that is used for natively storing data.
isBearingOnly	0 | 1	Binary flag indicating if object contains angle information only (i.e., no range).
position	double	Cartesian position vector of object of dimension order ×1.
covariance	double	Covariance matrix of object in space coordinateSystem.
range	double	Range of object [m]
azimuth	double	Azimuth of object [rad]
inclination	double	Inclination of object [rad]
marker	see LineSpec documentation	Marker used for plotting facility
colour	see ColorSpec documentation	3×1 colour vector used for plotting facility
isTrack	0 | 1	Binary flag indicating if mapFeature corresponds to a track.

2.2 Dependent properties

Dependent variables can be read, but cannot be written to. Their value is determined by dependency on another object property.

Name	Data type	Description	Dependency
`isTrack`	`0 | 1`	Binary flag indicating if `mapFeature` corresponds to a track.	`trackID`

2.3 Methods

The methods listed in the following are available to the map class.

Name	Description
`mapFeature(ID)`	Constructor. Input `ID` is optional
`init`	Initialisation of object.
`set`	Setter as per standard `hgsetget`.
`get`	Getter as per standard `hgsetget`.
`covariance_cartesian`	Returns the covariance in Cartesian coordinates.
`covariance_spherical`	Returns the covariance in spherical coordiantes.
`plot3`	Generates a 3D plot of the object position.

3 map

3.1 Properties

Name	Data type	Description
NAO	mapFeature	Up to one NAO feature can be contained in each map object, describing the NAO position and properties at this time stamp.
speaker	mapFeature	Zero or more mapFeature object can be contained in each map object, describing the speaker positions and properties at this time stamp.
num_speakers	uint32	Number of speakers contained within map object.
assigned_IDs	uint32	Vector of object IDs contained to the map object.
assigned_trackIDs	uint32	Vector of track IDs contained in the map object.

3.2 Methods

The methods listed in the following are available to the map class.

Name	Description
map	Constructor.
get	Getter as per standard hgsetget.
add_feature	Adds either a NAO or speaker feature to the map object.
get_speaker	Input: ID of desired speakers. Returns the corresponding speaker mapFeature.
get_NAO	Returns the NAO mapFeature.

4 mapTrajectory

4.1 Properties

Name	Data type	Description
data	map	Contains one map object per time stamp.
num_samples	uint32	Number of samples contained within mapTrajectory object.

4.2 Methods

The methods listed in the following are available to the class.

Name	Description
mapTrajectory	Constructor.
get	Getter as per standard hgsetget.
get_speaker	Input: ID of desired speakers. Returns the corresponding trajectory of the speaker's mapFeature objects over time.
get_NAO	Returns the trajectory of NAO's mapFeature objects over time.
plot3	Generates a 3D plot over time of all the features contained in the mapTrajectory object.
movie	Generates a movie using plot3.
plot2	Generates a 2D plot of requested properties (e.g., range, azimuth) over time and plots the 95^{th} confidence interval around the line.
write_XML	Writes the mapTrajectory object to XML.
read_XML	Reads a pre-specified mapTrajectory object from an XML file.

5 Appendix

5.1 Setters

The map class inherits from `hgsetget` for implementation of `set` and `get` methods, which is a subclass of the `handle` class. As such, parameters can be set using one of two methods:

```
% Option 1:
set( obj, ‘varname’, varval);

% Option 2:
obj.varname = varval;
```

where `varname` is the name of the property to set, `varval` is the value to set the property and obj is the instance of the map class. Consider for example the following example:

```
% Update the position of the map object to [3;1;0.8].
mymap = map();
% Set order of position vector first (dependency of position)
set( mymap, 'order', 3 );

% Option 1:
set( mymap, 'position', [3;1;0.8] );

% Option 2:
mymap.position = [3;1;0.8];
```

5.2 Getters

Getters operate in a similar to the setter methods, i.e., two options can be used to request values of properties:

```
% Option 1:
requested_value = get( obj, 'varname' )

% Option 2:
requested_value = obj.varname;
```

Consider for example the following example:

```
% Initialise map - as described above:
mymap = map();
set( mymap, 'order', 3 );
set( mymap, 'position', [3;1;0.8]);

% Option 1:
myposition = get( mymap, 'position' );

% Option 2:
myposition = mymap.position;
```