multilaser_surveillance

This package provides tools to perform surveillance on a known area. The area is watched by a number of fixed 2D laser range finders.

Multimodal tracking is based on the perception stack of the STRANDS project. This stack makes use of `BayesTracking`, a library of Bayesian tracking. For more info, read *Real-time multisensor people tracking for human-robot spatial interaction* by Dondrup and Bellotto.

Steps / pipeline:

1) `MapBuilderWatcher` - map building mode - Build the map based on the stream of laser scans.

2) `MapBuilderWatcher` - surveillance mode - Compare the streams of laser scans to the map and detect outliers w.r.t. the map

3) `2dclusterer` - cluster outliers into continuous blobs, and publish their barycenter.

4) `bayes_people_tracker` - convert the discontinuous blobs barycenters into tracks, using Unscented Kalman Filter.

Licence

BSD

Authors

- Package maintainer: Arnaud Ramey (arnaud.a.ramey@gmail.com)
- `strands_perception_people`: STRANDS project
- `BayesTracking` library: Nicola Bellotto (nbellotto@lincoln.ac.uk)

Compile and install

ROS Indigo + catkin

Compile with catkin_make:

```
$ roscd ; cd src
$ git clone https://github.com/strands-project/strands_perception_people.git
$ git clone https://github.com/LCAS/bayestracking.git
$ git clone https://github.com/wg-perception/people.git
$ rospack profile
$ catkin_make --only-pkg-with-deps multilaser_surveillance
```

Run

1) Build the map. The created map is shown in `rviz` : it corresponds to the purple cells, obtained by accumulating the laser scans and inflating of a constant radius around them. The map is automatically saved in `multilaser_surveillance/data/maps`.

```
$ roslaunch multilaser_surveillance stage_arenes.launch  mode:=build
```

2) Perform surveillance. The map is loaded from the same folder.

```
$ roslaunch multilaser_surveillance stage_arenes.launch
```

1) & 2) `MapBuilderWatcher` - map building & surveillance

modes

Parameters

- `~frames [string]`, default `""`. Semi-colon-separated list of the frame of each 2D scan defined in `~scan_topics`. Must be non empty and of the same size as `~scan_topics`.
- `~static_frame [string]`, default `"/map"`. The static frame for the map. The scans are converted into this frame.
- `~mode [string]`, default `"surveillance"`. Accepted values are `"build"` and `"surveillance"`.
- `~map_prefix [string]`, default `"mymap"`. Where to save or load (according to the mode) the map file. Corresponds to a `.csv` and a `.png` file (these extensions are aggregated to the parameter value).
- `~scan_topics [string]`, default `""`. Semi-colon-separated list of topics of 2D scans. Must be non empty and of the same size as `~scan_topics`.
- `~xmin, ~ymin, ~xmax, ~ymax [double, meters]`, default -10,-10,10,10 meters. ONLY IN BUILD MODE. Minimum and maximum values for the map boundaries. Scan values out of this range (in map coordinates) will be discarded.

Subscriptions

- `$(scan_topics) [sensor_msgs/LaserScan]` The different scan streams published by the laser range finders.
- `/tf [tf2_msgs/TFMessage]` The transforms between the frame of each scan and the `static_frame`.

Publications

- `/map [nav_msgs/OccupancyGrid]` The map, shaped as an occupancy grid. Rate: max 1 Hz.
- `/marker [visualization_msgs/Marker]` Amarker showing the outliers as a red point cloud, and the devices as arrows. Rate: max 1 Hz.
- `/outliers [sensor_msgs/PointCloud]` ONLY IN SURVEILLANCE MDOE. The point cloud of all points not belonging to the map. Rate: upon reception of each scan, max 100 Hz.
- `/scan [sensor_msgs/PointCloud]` The merged scan of all lasers. Rate: upon reception of each scan, max 100 Hz.

3) 2dclusterer

Parameters

- `~cluster_tolerance [double, meters]`, default 0.1 meter. The maximum distance between two points to consider them as belonging to the same cluster.

Subscriptions

- `/cloud [sensor_msgs/PointCloud]` The 2D point cloud to cluster, in `(x, y)`.

Publications

- `/cluster_centers [geometry_msgs/PoseArray]` The array containing the center of each cluster. For each center, the orientation is set to 0.
- `/marker [visualization_msgs/Marker]` Amarker showing the clusters as a colored point cloud, where the color corresponds to the cluster ID. Rate: upon reception of each PointCloud.

4) bayes_people_tracker

Parameters

- `detectors.yaml [YAML file]` AYAML file configuring the different detectors. See STRANDS doc for more details.

Subscriptions

- `~detectors/*/topic [geometry_msgs/PoseArray]` For each detector configured in the YAML file, the corresponding PoseArray topic.

Publications

- `/people_tracker/pose_array [geometry_msgs/PoseArray]` The pose of each object, obtained by Bayesian filtering.

Troubleshooting

Problem: The Bayesian tracker does not create tracks if my detector frame-rate is below 5 Hz (200 ms).

Explanation: By default, the BayesTracking multitracker creates tracks if it receives detections at least every 200 ms, cf. constructor:

```
MultiTracker(unsigned int sequenceSize = 5, double sequenceTime = 0.2)
```

And the embedded `MultiTracker` embedded in `people_tracker/simple_tracking.h` uses the default constructor:

```
MultiTracker<FilterType, 4> mtrk; // state [x, v_x, y, v_y]
```

Solution: change `sequenceTime` in MultiTracker instantiation: open `people_tracker/simple_tracking.h`

and change the line

```
SimpleTracking() {
```

for:

```
SimpleTracking() : mtrk(5, .5) {
```