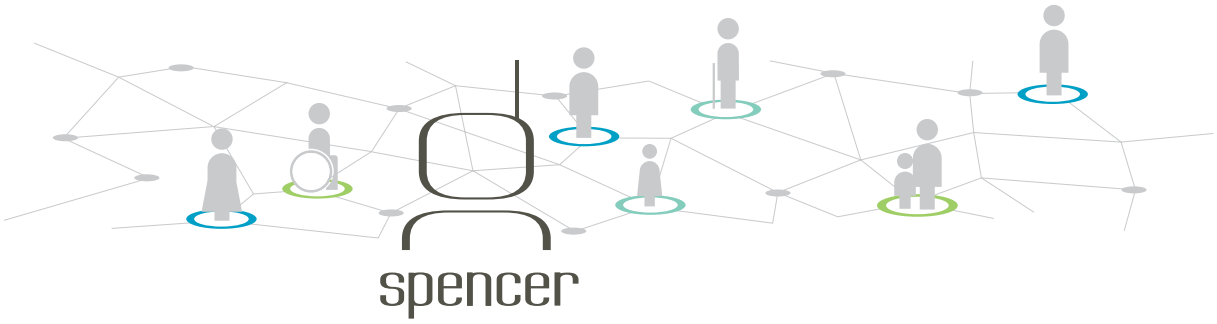



Grant agreement no: FP7-600877

# SPENCER:

## Social situation-aware perception and action for cognitive robots

Project start: April 1, 2013

Duration: 3 years

## DELIVERABLE 2.5

## Group detection and tracking from depth and vision data

Due date: month 34 (January 2016)

Lead contractor organization: ALU-FR

Dissemination Level: PUBLIC

# Contents

# Abstract

In this deliverable, we present the final, multi-modal people and group detection and tracking system used in SPENCER. The output of this system is used to guide a group of passengers to their goal at the airport, but has also been used in experiments as an input for socially compliant motion planning among groups of people. We briefly recapitulate our work on group detection via coherent motion indicator features, already mentioned in D2.4, followed by a description of our multi-modal people tracking system. This system has been evaluated in a series of experiments on both synthetic and real-world data, including an annotated dataset captured at the actual deployment area of the robot, Amsterdam-Schiphol airport. The system was tested successfully during the initial deployment at Amsterdam-Schiphol airport in December 2015, and only minor modifications had to be done for the final demonstration.


# 1 Introduction

Recognizing groups and interpreting their behavior are key to the understanding of populated environments. Therefore, task T2.3 deals with the development of a group detection and tracking framework that uses people detection and tracking from T2.1 as an input.

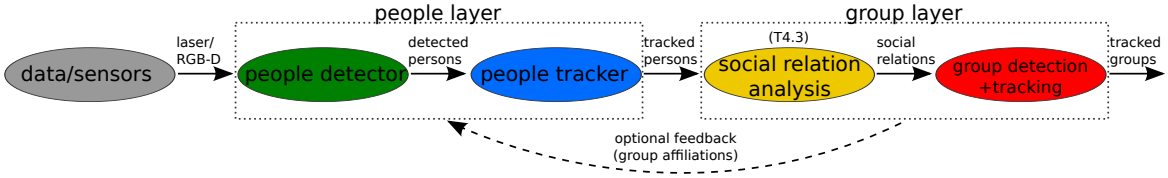



Figure 1: Overview of the data flow from sensor input to people detection and tracking to group detection and tracking. Feedback of group information into people tracking was studied experimentally in [5], but is not used in the final system deployed on the robot due to computational constraints.

In this deliverable, we first present the multi-modal people tracking framework, which has been significantly improved after deliverable D2.1 since it is one of the most crucial components running on the SPENCER robot platform, serving as an input to a series of social navigation modules. We then present our work on group detection via coherent motion indicator features, and the integration of all these components via the robot operating system (ROS).

### Improvements done after deliverables D2.1 and D2.4

Note that in this deliverable, we do not go into detail into the existing people and group tracking functionality that was already discussed in the previous deliverables D2.1 (People Detection and Tracking from Depth and Vision Data) and D2.4 (Group Detection and Tracking from Depth and Vision Data Early Prototype).

Building on top of these deliverables, the following improvements have been made for D2.5:

- The RGB-D-based person detectors (groundHOG, upper-body detector) by RWTH have been

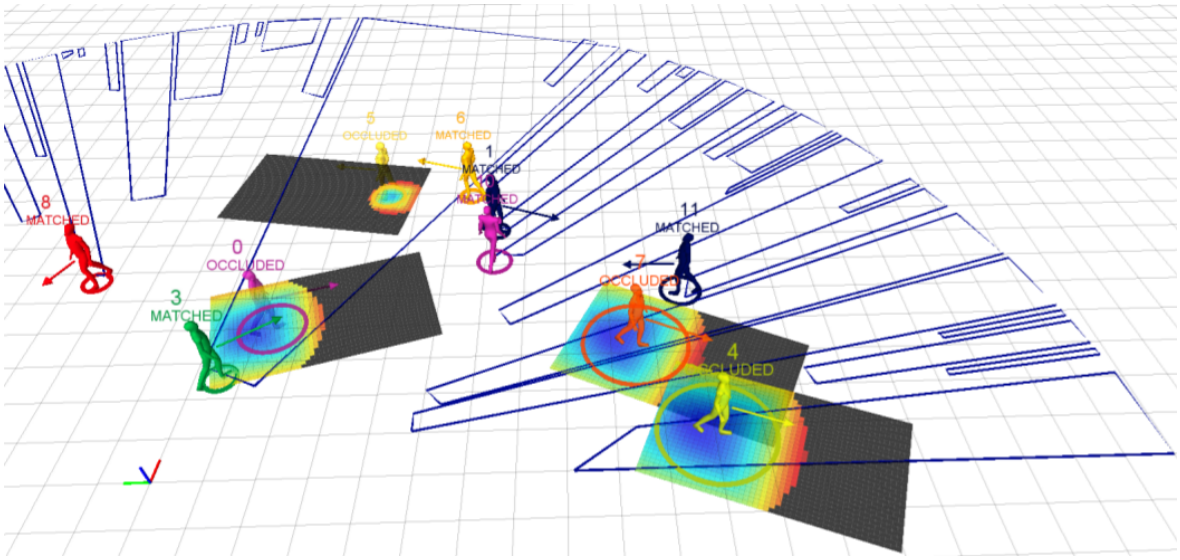



Figure 2: Integration of occlusion geodesics [8] into our people tracking framework, showing the reassignment costmaps for several occluded tracks.

fully integrated with the standalone nearest-neighbor tracker (NNT) by implementing an efficient detection-to-detection fusion scheme.

- We also integrated an additional RGB-D detector from PCL [6] for comparison purposes.
- We spent significant time on tuning the parameters of the NNT, implementing different motion models and an interacting multiple models filter, and a track initiation logic. As shown in [4] and [3], using these enhancements, the NNT can outperform much more complex data associations at only a fraction of the computational cost. For experimental purposes, we also integrated a recently proposed approach for geometric occlusion handling via occlusion geodesics [8] into the NNT, as shown in Figure 2.
- We implemented a wrapper for pySMAC[1] to allow for automatic tuning of tracker parameters via sequential model-based parameter optimization (see [4]).
- The entire people and group tracking stack has been tested successfully during the initial deployment at Amsterdam-Schiphol airport in December 2015.

## 2 Multi-modal people detection and tracking

Building on top of D2.4, where initially only a 2D laser-based people detection and tracking system developed by ALU-FR was deployed on the robot in October 2014, this system has now been integrated and successfully tested with two vision-based detectors from RWTH: A depth-based upper-body detector, and a monocular vision-based groundHOG detector. Additionally, the existing vision-based MDL tracker by RWTH has been modified such that it outputs tracked persons in the same

[1] https://github.com/automl/pysmac

format, allowing for comparability of the two approaches. A recent comparative study [4] of different methods on synthetic data and data from a pedestrian area has shown that e.g. a simple nearest-neighbor tracker can perform similar to, or even better than, an MHT in crowded environments due to the combinatorial explosion of possible track states.

In our most recent work, [3], we have conducted a comparison of different people tracking systems on a dataset recorded at the actual deployment area of the robot, Amsterdam-Schiphol airport. The examined systems include the MHT and NNT by ALU-FR, the MDL tracker by RWTH, and another variant of NNT used within the STRANDS EU FP7 project.

In a nutshell, our results indicate that:

- More complex data associations (such as MHT or MDL) are not necessarily better than simpler, less resource-intensive data association methods like NNT.
- Instead, implementation details such as track initiation logic, track deletion or geometric occlusion handling play a very large role.
- The most important factor influencing tracking quality is still the accuracy of the underlying person detector(s).
- In the sensory setup used in SPENCER, a combination of the close-range RGB-D upper-body detector, and the 360-degree 2D laser detector prove to be a good combination. As it turns out, also integrating the groundHOG detector for far-range vision can lead to worse results, as at far range the geometric accuracy of monocular vision is much lower than the accuracy of 2D laser.

Our (so far unpublished) results from experiments with the occlusion geodesics (Fig. 2), originally proposed by Possegger et al. [8] for people tracking from overhead video cameras, show that this extension can help to reduce the number of track mismatches, at the cost of an increased number of false positives. This trade-off that has to be made, also depends on the particular application scenario (tracking groups as long as possible e. g. for group guidance, vs. accurate short-term tracking as an input to social navigation).

# 3 Group detection and tracking

For group detection, as described in [5, 1], we build a social network graph where the nodes represent persons and the edge weights the likelihood of a positive social relation between each pair of persons. Figure 3 (left) shows such a social network graph. The social relation likelihoods are provided by task T4.3 (Social Relation Analysis), and can e.g. be derived using a probabilistic SVM classifier trained on coherent motion indicator features (relative distance, relative speed, relative orientation), using a dataset annotated with groups of pedestrians, as well as single individuals.

The social network graph is rebuilt at every single person tracking cycle. After it has been constructed, we perform graph-cutting at a manually specified threshold, to find clusters of persons that belong together.

The initially studied multi-model MHT [5, 1] is a monolithic tracker that integrates person-level and group-level tracking. We have proven the usefulness of this approach via established tracking

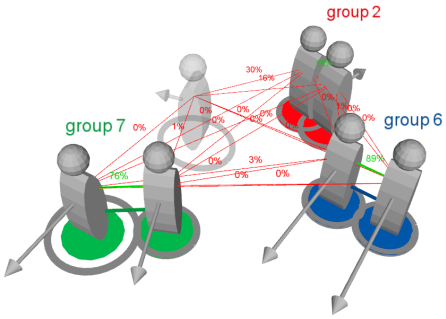



Figure 3: A social network graph between tracked persons, which are the nodes of the graph. The edge weights are to probabilities of positive social relations between each pair of persons, determined using a probabilistic SVM trained on coherent motion indicator features.

metrics, as it allows us to feed back information from group-level into person-level tracking (by e. g. adapting person-level occlusion probabilities within groups). However, this approach is also computationally complex (due to the combinatorial explosion of possible states), and thus requires a significant amount of CPU power which is usually limited on mobile platforms that also need to run other (perception) components such as obstacle detection, motion planning or mapping.

Due to the limited number of computational resources on the robot, we have therefore also implemented a significantly less complex *stand-alone group tracking module*. The group detection occurs via the same mechanism as described before, leading to a social network graph. Group clusters are then tracked by either associating group centroids (using a nearest-neighbor filter), or group compositions, in terms of individual group member IDs (by keeping a memory of previously seen group configurations). The latter approach has, in qualitative experiments, been shown to be more robust when group splits and merges happen frequently. For details of the implementation, see [2].

All these modules fully communicate with each other via ROS and can be flexibly combined and interchanged, so that e. g. also the RWTH vision-based detection and tracking system described in deliverable D2.1 can be used with this group tracking mechanism.

## 3.1 Qualitative results

These group tracking approaches have shown to provide valuable input for social navigation in cocktail party-like scenarios. They were used as an input to learning socially normative robot navigation behaviors, in a recent paper [7] that is described more closely in deliverable D5.2. As shown in Figure 4, the robot DARYL avoids splitting a group of people engaged in a conversation for which the tracker has estimated a high-probability pair-wise social relation. Without such an estimate, the robot drives in between the persons still respecting their personal spaces.

However, in real data captured at Amsterdam Schiphol, where people are often in a rush and the areas are at times very crowded, it is difficult to make out distinct groups even for a human annotator. But even in those cases where it is possible to discern individual groups, the inter-person distances are often so small that the SPENCER robot could not pass in between the persons at all, due to its width of around 85 centimeters plus some safety margin. Therefore, even without any dedicated group detection and tracking module, the existing obstacle and social compliance layers will already cause

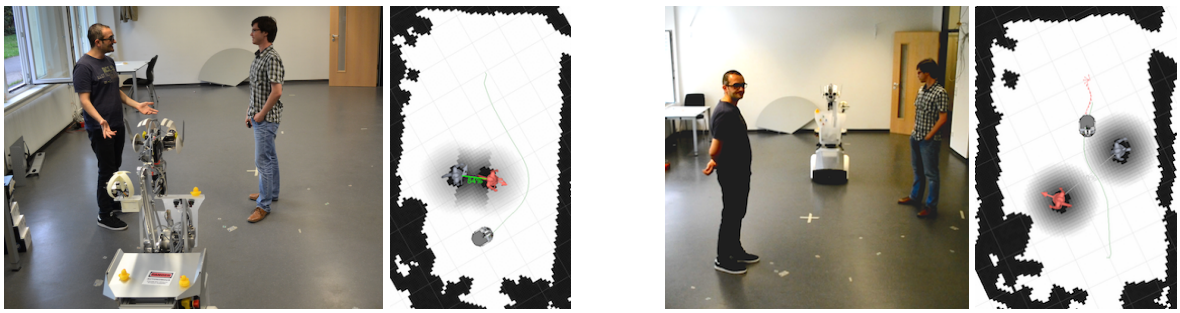


Figure 4: Experiments with the robot Daryl showing a socially normative behavior in a group of people. Setup and resulting costmaps are shown side-by-side, darker cells are higher cost, the robot path is shown in green. **Left:** two persons engaged in a conversation, the robot drives around them for social normativeness. **Right:** two persons without social relation, the robot drives in between them while respecting their personal spaces. For more details, see deliverable D5.2 and [7].

the robot to not cross in between group members.

Group tracking is, however, useful for the guidance task. In this case, group detection can be performed at the beginning of the scenario (where the group that is to be guided is standing in front of the robot), and the group can then be tracked by tracking its individual person tracks (based upon their IDs). This is sensible, because we still want to keep track of all group members even if one member temporarily increases the distance to the other group members (e.g. to avoid another person).

Qualitatively, this approach worked fine most of the time during the initial robot deployment at Amsterdam-Schiphol airport in December 2015. However, sometimes the robot lost track of the group completely if the group had to pass through very busy and crowded areas, such as the shopping plaza. In these cases, an appearance-based person reidentification module – which was unfortunately not within the scope of the project as per the DoW – would have been required, as solely tracking person trajectories is insufficient in these cases as noted also by human annotators.

We plan to record further quantitative metrics (on how often the guided group was lost and for how long people have been tracked) during the final robot deployment in March 2016.

## References

[1] Timm Linder and Kai O. Arras. Multi-model hypothesis tracking of groups of people in RGB-D data. In *IEEE Int. Conf. on Information Fusion (FUSION'14)*, Salamanca, Spain, 2014.

[2] Timm Linder and Kai O. Arras. People detection, tracking and visualization using ROS on a mobile service robot. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference. Studies in Systems, Decision and Control*. Springer, 2016.

[3] Timm Linder, Stefan Breuers, Bastian Leibe, and Kai O. Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'16)*, Stockholm, Sweden, 2016.

[4] Timm Linder, Fabian Girrbach, and Kai O. Arras. Towards a robust people tracking framework for service robots in crowded, dynamic environments. In *Assistance and Service Robotics Workshop (ASROB-15) at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2015*, Hamburg, Germany, 2015.

[5] Matthias Luber and Kai O. Arras. Multi-hypothesis social grouping and tracking for mobile robots. In *Robotics: Science and Systems (RSS'13)*, Berlin, Germany, 2013.

[6] M. Munaro, F. Basso, and E. Menegatti. Tracking people within groups with RGB-D data. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[7] Billy Okal and Kai O. Arras. Learning socially normative robot navigation behaviors using bayesian inverse reinforcement learning. In *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)*, Stocholm, Sweden, 2016.

[8] H. Possegger, T. Mauthner, P.M. Roth, and H. Bischof. Occlusion geodesics for online multi-object tracking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1306–1313, June 2014.

# Towards a Robust People Tracking Framework for Service Robots in Crowded, Dynamic Environments

Timm Linder Fabian Girrbach Kai O. Arras

***Abstract*— People tracking is an important prerequisite for socially compliant service robots that operate in human environments. In this paper, we take this challenge to new extremes by attempting to robustly track people in a 360-degree field around the robot in very crowded environments like a busy airport terminal. As a first contribution, we present a novel multi-modal people tracking framework that is modular, flexible and fully integrated with ROS. We believe that our framework can be of great benefit to researchers as it covers the entire people tracking pipeline, including powerful visualization and evaluation tools. Secondly, we present a number of simple extensions that can make tracking in crowded scenarios more robust. Finally, we compare our tracking method against a complex multi-hypothesis tracking system. Our results on real and synthetic data suggest that it is not the choice of data association which has the largest impact, but the underlying models that, for instance, control track initiation, deletion, and handling of occlusions. By showing that the simpler data association may be sufficient, we provide reasoning why the available resources on a resource-constrained mobile robot might be better spent on other tasks such as higher-level perception and reasoning.**

## I. Introduction

People tracking from a mobile platform in a first-person perspective has been studied in the robotics and computer vision communities for over a decade, and provides important functionality for assistance and service robots operating in human environments. It can lay the foundations to higher-level social processing such as socially aware motion planning [1], group detection and tracking [2], [3], social activity detection [4] or general human-robot interaction, and is crucial in person guidance tasks. The problem has basically been solved in simple scenarios with only a handful of persons walking in front of 2D laser [5] or RGB-D sensors [6], and good progress has been made in semi-crowded scenarios with 10–15 people simultaneously visible [7]–[10]. Few approaches are multi-modal in nature [11], [12].

However, larger numbers of persons need to be tracked when the entire 360-degree field of view around the robot shall be covered using an entire array of sensors, and not many systems have been evaluated in very complex and highly dynamic scenarios where over 30 persons can be present at the same time, such as in a very crowded airport terminal where our own service robot is going to be deployed (Fig. 1).

The authors are with the Social Robotics Lab, Dept. of Computer Science, University of Freiburg, Germany. http://srl.informatik.uni-freiburg.de, {linder,arras}@cs.uni-freiburg.de. This work has been partly supported by the European Commission under contract number FP7-ICT-600877 (SPENCER).

Fig. 1. Typical example of a crowded, highly dynamic situation in an airport terminal where we want to robustly and efficiently track persons with our service robot platform depicted on the bottom left.

In this paper, we present a multi-modal, highly modular and ROS-based people detection and tracking framework that, to our knowledge, is the most complete publicly available framework for this purpose, encompassing powerful visualization components, a group detection and tracking module, and implementations of different evaluation metrics for comparison with other approaches. Secondly, we discuss how a set of relatively simple extensions can make a person tracking system based upon very efficient nearest-neighbor (NN) data association more robust in challenging scenarios. We argue that the choice of data association method only plays a subordinate role, and what really matters are the models used to *e. g.* initiate and terminate tracks, handle occlusions, and predict human motion. We demonstrate this in a first set of experiments, where we compare our non-probabilistic nearest-neighbor method to a proven multi-hypothesis tracker that is significantly more complex in terms of implementation, parameter finding, and computational requirements. Our core tracking algorithm runs at less than 30% CPU load on a single core in complex scenarios, leaving enough computational resources for higher-level perception and social reasoning components.

Finally, we present a way of automatically tuning tracking parameters with regard to multi-target tracking metrics via an existing hyperparameter optimization library. Even in a simple NN-based system, there can be over 20 inter-dependent parameters that can affect tracking performance and require expert knowledge when tuned manually. These parameters are often part of noise models which significantly abstract

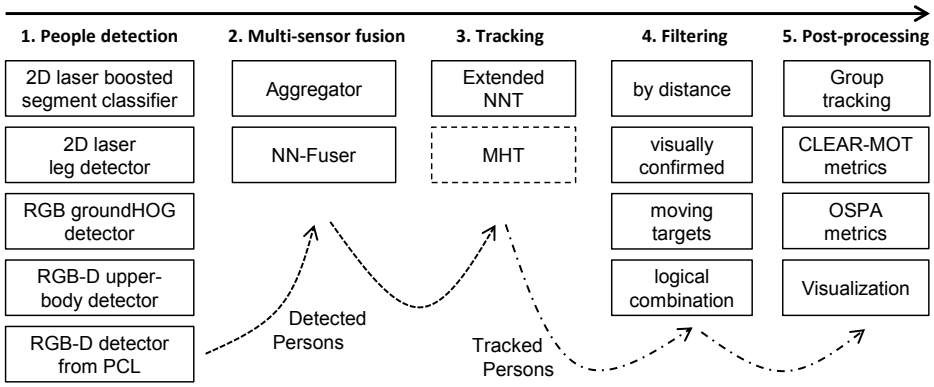



Fig. 2. Components and data flow between the 5 processing stages of our people tracking framework. All components are implemented as separate, reusable ROS modules, incorporating our own and third-party detection modules, tracking performance metrics, and powerful visualization components. All components, except for the one with dashed outline, are available as open-source.

from the underlying models of reality, and can at times be even counter-intuitive to use. Therefore, automated parameter tuning can significantly ease the burden of deploying our tracking framework in new scenarios.

## II. Our framework

Fig. 2 gives an overview of the main components of our modular people tracking framework. All of these components are fully integrated into ROS and will be made publicly available on GitHub[1] at publication of this paper. In the following, we will briefly describe the most important components, starting from the left at the detection layer.

### A. Detection

*2D laser:* For people detection in 2D laser range data, we have re-implemented a variant of the boosted laser segment classifier from [13] and integrated it with ROS. After a separate ROS node has segmented the laser scans using jump-distance clustering or a more accurate, but computationally more complex agglomerative hierarchical clustering (AHC), the detector computes a set of geometric 2D features on each segment which are then fed to the classifier. An additional high-recall blob detector which coarsely classifies the same kind of segments based upon their number of points and overall width has also been integrated.

We also integrated a publicly available leg detector[2] into our framework, which can provide better results if the sensor is mounted close to the ground such that both legs are visible as individual echoes in the laser scan. In our experiments with the sensor at 70–80 cm height, however, this detector (after parameter tuning[3], with the existing learned model) did not provide better results than our boosted segment classifier.

*Monocular vision and RGB-D:* For person detection in RGB-D, the existing depth template-based upper-body detector described in Jafari et al. [10], which runs in real-time at 20-30 Hz on the CPU, as well as their CUDA-based, monocular groundHOG detector have been integrated. We also extended the RGB-D person detector from [14], which applies a HOG classifier on candidate regions extracted from a depth-based height map, with GPU acceleration.

*Fusing detections:* For multi-sensor people tracking, our framework currently uses a flexible detection-to-detection fusion scheme configured fully via XML files. This allows to combine multiple sensor cues even when the particular tracking algorithm was not specifically designed to receive detection input from multiple sources. Using greedy nearest-neighbor association, we first fuse detections from sensors with overlapping fields of view (*e. g.* front laser, front RGB-D) and then aggregate the resulting sets of detections that do not overlap (*e. g.* front and rear detections).

All detectors integrated into our framework output detections which adhere to the same ROS message format. A *DetectedPerson* comprises a position vector $\mathbf{z}'$ and its uncertainty $R'$ in a sensor-specific 3D coordinate frame, a scalar detection confidence, and some meta-data. $R'$ can, for example, vary as a function of the person's distance to the sensor.

### B. Tracking

In this section, we describe a new tracking system developed with robustness and computational efficiency in mind specifically for deployment on mobile service robots in crowded environments. Using a relatively cheap set of extensions from the target tracking community to systematically tackle shortcomings of current systems in such scenarios, we want to improve robustness without having to resort to multi-hypothesis tracking methods that are orders of magnitudes more complex in terms of implementation and computational requirements.

In our tracking system, detections arrive in their sensor-specific coordinate frame and are instantaneously transformed into a locally fixed frame (based upon robot odometry) that does not move with the robot. This ensures that the motion prediction of tracked persons is independent from the robot's ego-motion. In the resulting set of measurements $Z = \{\mathbf{z}_1, ..., \mathbf{z}_n\} \subset \mathbb{R}^2$, we drop the $z$ coordinate as we currently only track in 2D world coordinates.

[1] https://github.com/spencer-project/spencer_people_tracking

[2] http://wiki.ros.org/leg_detector

[3] With original parameters, the detector had very low recall on our data.

*Motion prediction:* We predict the target motion by maintaining an Extended Kalman filter for each individual person. At our detection rate of around 30 Hz, human motion can in most scenarios be assumed to be locally linear, leading to the Nearly Constant Velocity model with state vector $\mathbf{x} = [x, \dot{x}, y, \dot{y}]^T$ and the transition matrix

$$F = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

to predict a future state $\hat{\mathbf{x}} = F\mathbf{x}$, where $t$ is the length of the tracking cycle. We use additive process noise $Q$ to account for small changes in the velocity of the human motion. The process noise level $q_{\mathrm{L}}$ varies with the dynamics of the application scenario:

$$Q = \begin{bmatrix} \frac{1}{3}t^3 & \frac{1}{2}t^2 & 0 & 0 \\ \frac{1}{2}t^2 & t & 0 & 0 \\ 0 & 0 & \frac{1}{3}t^3 & \frac{1}{2}t^2 \\ 0 & 0 & \frac{1}{2}t^2 & t \end{bmatrix} q_{\mathrm{L}}. \tag{2}$$

*Data association:* Correctly associating new detections with existing tracks is crucial for good tracking performance. On the other hand, recent research [15] suggests that for people tracking from a mobile platform, with increasing environment complexity the difference in performance between various approaches such as Nearest-Neighbor (NN), Joint Probabilistic Data Association (JPDA) [5] and Probability Hypothesis Density (PHD) filters [15] diminishes.

Due to the significantly lower complexity that scales well with the number of tracked persons, we employ different variations of NN data association. The Global NN approach solves the complete linear assignment problem between detections and tracks using the Hungarian method [16] or the faster Jonkers-Volgenant [17] method, whereas Greedy NN searches for minima in the cost matrix in a greedy fashion. For all these approaches we use the Mahalanobis distance between track prediction and detection as assignment cost.

### C. Track post-processing and visualization

Often, higher-level reasoning components are not interested in all tracks that are maintained internally by the people tracking system. Therefore, we provide a series of post-processing modules which filter the output such that it, for instance, only includes visually confirmed tracks, non-static persons, the $n$ persons closest to the robot (useful for human-robot interaction), or a logical combination thereof. A standalone group detection and tracking module, based upon the coherent motion indicator features and the social network graph described in [3], has also been added.

The visualizations in this paper have been generated using a set of custom and highly configurable plugins for the ROS visualization tool *RViz* that are part of our framework.

## III. Extensions for more robust tracking

### A. Better motion prediction in dynamic scenarios

We provide a bank of first- and second order motion models, which cover different aspects of human motion. The variety of human motion has to be considered especially in crowded environments, where people are forced to change their motion according to the dynamics of the environment, which may lead to sudden stops or curvy trajectories. In addition to the Nearly Constant Velocity (CV) model (Eq. 2), the framework comprises the following motion models: Brownian Motion (Wiener Process), Nearly Constant Acceleration (CA) and Nearly Coordinated Turn (CT). Slight random variations of the constant term are modelled via additive white Gaussian noise. To capture the variety of human motion, the models may be combined inside an Interacting Multiple Models filter (IMM), which further helps to reduce the dependency on the process noise level $q_L$ that strongly influences tracking performance but is a difficult to configure parameter due to the trade-off between tracking precision and robustness to sudden maneuvers.

### B. Track initiation

Especially in sparse 2D laser range data, false positive detections can occur at high rates. To reduce the resulting number of 'ghost' tracks, we use a rule-based approach to confirm track creation. The *Track Initiation Logic* [18] was proven to perform well in a statistical and practical analysis for radar tracking [19]. For a track to be confirmed, it has to pass a gating step based upon the Euclidean distance between the measurement $\mathbf{z}$ and the current measurement prediction $H\hat{\mathbf{x}}$ of the track candidate, as well as a gating test that restricts the linear velocities to an interval $[v_{\min}, v_{\max}]$. If $v_{\min} > 0$, track initiation is restricted to moving targets, which prevents tracks being created from human-like objects such as trees or columns. As an extension to the original method, in each tracker cycle the velocity gating is performed recursively for each observation with regard to all observations already associated to the track candidate $c_i$, as shown in Alg. 1.

### C. Track deletion logic

In the base version of our tracking system, we delete occluded tracks after a fixed number of tracking cycles $n_{\mathrm{del}}$ if no matching detection could be found. To prevent possible false alarm tracks from staying in the system for too long, we add a discrimination between 'young' and 'mature' tracks, where the latter is a track that over its lifetime has been matched at least $n^{\mathrm{mat}}$ times. In case a young track is occluded, it is deleted after being without detection for $n_{\mathrm{del}}^{\mathrm{yng}}$ cycles, whereas a mature track is deleted after a larger number of steps $n_{\mathrm{del}}^{\mathrm{mat}} > n_{\mathrm{del}}^{\mathrm{yng}}$ without detection.

## IV. Experiments and Results

Since the focus of this paper is on tracking, and not detection, for the purpose of the following experiments we restrict ourselves to using 2D laser range data. The presented

**Algorithm 1:** Cascaded logic for track initiation

**Data**: unmatched detections $Z^u$, initiation candidates $C$
**Params**: min. match count $n_{\min}$, velocity thresholds
**Result**: new tracks $T_{\text{new}}$ to be initiated
**foreach** *existing initiation candidate* $\mathbf{c}_i \in C$ **do**
  **foreach** $\mathbf{z}_j \in Z^u$ **do**
    Predict next measurement from state of $\mathbf{c}_i$
    Check distance between prediction and $\mathbf{z}_j$
    Check velocity between $\forall \mathbf{z} \in Z_{c_i}$ and $\mathbf{z}_j$
    **if** *checks passed* **then**
      $Z^u = Z^u \setminus \{\mathbf{z}_j\}$
      **if** $|Z_{c_i}| \geq n_{\min}$ **then**
        $C = C \setminus \{\mathbf{c}_i\}$
        Add $\mathbf{c}_i$ to $T_{\text{new}}$
      **else**
        Add $\mathbf{z}_j$ to $Z_{\mathbf{c}_i}$
        $n_{\text{miss},\mathbf{c}_i} = 0$
  **if** $\mathbf{c}_i$ *has no matching detection* $\mathbf{z} \in Z^u$ **then**
    $n_{\text{miss},\mathbf{c}_i} = n_{\text{miss},\mathbf{c}_i} + 1$
    **if** $n_{\text{miss},\mathbf{c}_i} > n_{\text{miss, max}}$ **then**
      Delete initiation candidate: $C = C \setminus \{\mathbf{c}_i\}$
**foreach** $\mathbf{z}_j \in Z^u$ **do**
  $C = C \cup$ new candidate $\mathbf{c}_k$ with $Z_{\mathbf{c}_k} := \{\mathbf{z}_j\}$

tracking method with its extensions is independent of the sensor modality used for detecting people.

### A. Datasets

We evaluate the proposed system on two datasets of different complexity. Exemplary screenshots are shown in Fig. 3, with statistics on these two datasets in Table I. The first one is a popular 2D laser dataset recorded with a stationary SICK LMS-291 laser scanner at the Main Station in Freiburg, Germany. The second, new dataset is significantly more complex in terms of crowd dynamics and number of tracks within sensor range at a given time. This dataset has been synthetically generated using a combination of the pedestrian simulator *PedSim*, and the robot simulator *Gazebo*. Using our ROS wrapper of PedSim[4], we have modelled a scenario similar to the one our service robot will encounter in an airport terminal (Fig. 1), with large flows of people moving around corners towards specific goals, static groups and single persons spread all over the environment, and other people queuing up. Person interactions are modelled in PedSim using the social force model after Helbing [20], and pedestrian positions are then fed to Gazebo to continuously reposition 3D meshes in a scene in order to generate 2D laser scans via raycasting from a simulated mobile platform. While this approach obviously cannot correctly depict reality in all its detail, especially in terms of background complexity, it allows us to quickly study different scenarios, robot behaviors, and obtain exact groundtruth without need for manual annotations.

[4] https://github.com/srl-freiburg/pedsim_ros

| | Frame Count | | Track Count | | |
|---|---|---|---|---|---|
| Dataset | Total | Annotated | Total | Avg | Max |
| Main Station | 33,204 | 6,000 | 160 | 16.7 | 27 |
| PedSim | 4,965 | 4,965 | 90 | 78.6 | 90 |

TABLE I
DATASET STATISTICS

### B. Evaluation metrics

A commonly used measure for evaluating multi-object tracking performance are the CLEAR-MOT metrics [21]. Besides counting false positives (FP), false negatives (FN) and ID switches (ID), they define an aggregate error measure called MOT Accuracy (MOTA) as

$$\text{MOTA} = 1 - \frac{\sum_k (\text{FP}_k + \text{FN}_k + \text{ID}_k)}{\sum_k \text{GT}_k},$$

where $k$ is the tracking cycle index. The optimal MOTA score is 1.0, and MOTA can reach negative values if the tracker makes more errors than there are ground truth objects GT over the entire dataset duration.

As discussed extensively in [22], MOTA scores can vary between implementations and are dependent on meta-parameters such as the matching distance threshold $\theta_d$ and the way in which track hypotheses are assigned to groundtruth objects. Therefore, it is important to use the same metrics implementation when comparing different tracking approaches. While in our previous work [2], the tracking metrics were tightly integrated into the tracker, our framework provides a standalone Python implementation of CLEAR-MOT metrics as a separate ROS node that can be used to evaluate any kind of tracking algorithm compatible with our ROS message definitions. In our version, we compute groundtruth correspondences using a variant of the Hungarian method based upon Euclidean distances between object centroids in 2D world coordinates, with $\theta_d = 1\text{m}$.

Although MOTA can give a good impression of overall tracking performance, it is questionable if it alone can be a sufficient measure of tracking performance, as it weights all types of errors equally. Depending on the application scenario, some errors might have larger consequences (*e. g.* switching track IDs in a person guidance scenario). In our results, we therefore list these error types separately. Our framework also incorporates an implementation of the OSPA metrics [23], which we did not use in this work.

### C. Experimental setup

All of our experiments were conducted on a high-end gaming laptop equipped with a quad-core Intel Core i7-4700 MQ processor and 8 GB of RAM under Ubuntu 14.04 with ROS Indigo. This is the actual computer platform used for this purpose on our service robot. For all experiments, we used the same boosted 2D laser segment classifier with agglomerative hierarchical clustering at a linkage threshold of 0.2m, pre-trained on annotated data from a SICK LMS-291 laser scanner at 75 cm height. We output detections with

Fig. 3. Example groundtruth tracks of the datasets used in our experiments, and originating laser endpoints (in grey). *Left:* Freiburg Main Station dataset, recorded with a stationary LMS 291 laser scanner. The camera image is only for visualization purposes. *Right:* Synthetic dataset generated by combining the pedestrian simulator *PedSim* with *Gazebo* to simulate 2D laser range data via raycasting (small pictures). Our simulation is modelled after a highly crowded, real airport scenario where people disembark from an airplane and enter the airport terminal. The simulated robot is driving through the $30\text{m} \times 30\text{m}$ scene for exploration and equipped with two laser scanners at 70 cm height.

a fixed position uncertainty of $R = \text{diag}(0.1, 0.1)$ up to a maximum range of 20m, after which laser echoes become very sparse and often consist of less than 3 points.

As an additional baseline method for comparison, we use a variant of the multi-hypothesis tracker (MHT) after Reid *et al.* [24] and Cox & Hingorani [25] with explicit occlusions labels [26]. We use $n$-scanback pruning with $n = 30$ and limit the maximum time $t_{\max}$ per tracking cycle in which an arbitrary number of hypotheses may be generated to $0.03\text{s}$. The EKF parameters for the CV motion model are identical to the ones used for the NNT. The probabilities $p_{\text{det}}$, $p_{\text{occ}}$, $p_{\text{del}}$ are $0.7$, $0.27$ and $0.03$ as in [2]. The poisson rates for new tracks and false alarms were hand-tuned on our scenarios to $\lambda_{\text{new}} = 0.005$ and $\lambda_{\text{fal}} = 0.005$. We additionally enforce occluded tracks to be deleted after at most 10 cycles without detection, as they otherwise stay in the system for too long, leading to extremely bad MOTA scores in the very dynamic PedSim scenario.

### D. Parameter tuning

Finding the correct parameter configuration is crucial to achieve good tracking results. Even in a simple NN-based system, there can be over 15 inter-dependent, performance-relevant parameters that often require expert knowledge for manual tuning. In particular, our experience is that i) the process noise level $q_L$, ii) the measurement covariances $R$, iii) track initiation- and iv) track deletion thresholds can have a high performance impact and are difficult to estimate.

We therefore integrated pySMAC[5], a Python wrapper for the hyperparameter optimization tool SMAC [27], with our extended NN tracker to identify well-performing parameter sets. SMAC allows to define categorical, integer and float parameter ranges and boundary conditions to be met. It uses a given performance metric, in our case MOTA, to fit a surrogate model in the form of a random forest. The model is used for prediction of promising configurations, which are then optimized using a combination of Bayesian optimization and local search.

## V. Results

Table II shows tracking performance results of our extended NNT as well as the MHT baseline on the moderately crowded Freiburg Main Station and the very crowded PedSim datasets. We also show quantitative results of our tracking framework in different scenarios on our YouTube channel[6].

### A. Data association

The results for the different data association methods, namely NN with multiple associations per detection, Global NN and Greedy NN, show that the sub-optimal solution of the assignment problem of the greedy method yields nearly the same results as the optimal solution found by applying the Jonkers-Volgenant [17] algorithm for the optimal solution. The NN variant that allows a detection to be assigned to multiple tracks yields higher results in MOTA, but also a higher number of ID switches. This can be explained by the fact that a track with a missing detection converges towards its nearest neighbor, which after a number of missed detections results in a track duplicate.

### B. Track initiation logic

The track initiation logic by itself drastically reduces the FP rate by around half, but also leads to an increased miss rate for three different reasons. First, by requiring at least $n_{\min} = 6$ matches in our case for a track confirmation, each track's appearance is delayed by the same number of tracking cycles. Second, for targets outside of our velocity boundaries of $[v_{\min}, v_{\max}] := [0.2, 2]$, such as static persons, no tracks are initiated. Lastly, for targets that only infrequently trigger a detection, the number of consecutive allowed misses $n_{\text{miss, max}}$ might be too low.

### C. Track deletion logic

Our track deletion logic has been tuned to delete young tracks after $n_{\text{del}}^{\text{yng}} = 20$ cycles and mature tracks after $n_{\text{del}}^{\text{yng}} = 50$ cycles. A track is being considered mature after at least $n^{\text{mat}} = 100$ matches. As can be seen from the results in Table II, this initially leads to a worse MOTA score compared to the baseline NN tracker due to the increase in false positives, as certain tracks are now allowed to survive for a longer time compared to the case without deletion logic where they are already deleted after $n_{\text{del}} = 5$ cycles.

However, once deletion and initiation logic are combined, the overall MOTA score increases because the two extensions

[5] https://github.com/automl/pysmac

[6] https://youtube.com/spencereuproject

| Method | Main Station dataset MOTA | ID | FP% | Miss% | Hz | PedSim dataset MOTA | ID | FP% | Miss% | Hz |
|---|---|---|---|---|---|---|---|---|---|---|
| Global NN | 71.9% | 1280 | 8.4% | 18.6% | 6997 | 80.4% | 4962 | 12.4% | 5.7% | 1323 |
| Greedy NN, multi-association | 70.7% | 1439 | 9.8% | 18.3% | 6766 | 78.8% | 5627 | 14.1% | 5.5% | 1403 |
| Greedy NN | 71.9% | 1279 | 8.4% | 18.6% | 6976 | 80.5% | 4968 | 12.4% | 5.7% | 1061 |
| +Extended initiation logic | 67.2% | 781 | 2.5% | 30.0% | 7069 | 78.9% | 3112 | 4.9% | 15.3% | 1260 |
| +Deletion logic | 66.7% | 463 | 20.1% | 12.7% | 5732 | 71.1% | 1842 | 25.0% | 3.3% | 747 |
| +Base initiation logic | 72.6% | 274 | 6.6% | 20.1% | 6816 | 80.8% | 1234 | 9.1% | 9.6% | 1025 |
| +Extended initiation logic | 73.3% | 306 | 7.2% | 19.3% | 6472 | 82.2% | 1315 | 9.4% | 7.9% | 935 |
| +IMM (CV + CV) | 73.3% | 311 | 7.2% | 19.3% | 3433 | 82.2% | 1272 | 9.4% | 8.0% | 606 |
| MHT $t_{\max} = 0.03$s | 72.2% | 815 | 9.4% | 17.4% | 33 | 71.3% | 3670 | 23.0% | 4.8% | 32 |

TABLE II

TRACKING PERFORMANCE COMPARISON

augment each other well. Combining the two, MOTA on the Main Station dataset rises from 71.9% to 73.3%, with an impressive reduction in ID switches from 1279 to 306. Similarly on the PedSim dataset, the number of ID switches is reduced by around 75%. Here, the extended version of the track initiation logic that recursively performs velocity gating against all previous detections that were already associated with the candidate, achieves 0.7-1.4% higher MOTA scores than the basic version which just performs velocity gating on the latest associated detection.

### D. IMM

The IMM results in Table II were achieved using two CV models with different process noise levels $q_{\mathrm{L}_1} = 0.035$ and $q_{\mathrm{L}_2} = 0.267$. The parameters and the overall choice of motion models were found by automatic parameter optimization via PySMAC. While MOTA did not improve by adding the IMM, the number of ID switches went down slightly by $3.6\%$ on the more challenging PedSim dataset. Additional qualitative experiments in our lab, where multiple persons interacted with our robot in a narrow environment, showed a reduction in track losses and subsequent ID switches. We believe that in such human-robot interaction scenarios, the IMM can lead to more obvious improvements, because the human subjects often try to 'play' with the robot and trick the tracking system into errors by performing erratic maneuvers. This happens less often in our recorded datasets, which do not include explicit human-robot interactions and therefore contain fewer sharp turns and persons stopping abruptly.

### E. Comparison to other systems

Compared to the baseline nearest-neighbor tracker, the hypothesis-oriented MHT [26] – as expected – achieves better scores on the Main Station dataset[7]. On the highly challenging PedSim dataset, however, the MHT underperforms even after carefully tuning its parameters. We believe this to be due to the combinatorial explosion of possible data associations given the high track count[8]. This would necessitate a very high number of hypotheses, which is infeasible to maintain within the given cycle time limit of 30 ms. No improvement could be achieved by further raising the cycle limit to *e. g.* 100 ms.

Compared to the NNT with all extensions, the bare MHT performs 1% worse on the Main Station dataset, but 11% worse on the PedSim data. We believe the higher number of ID switches in MHT output, compared to the NNT, is due to frequent switching of the global best hypothesis. This is a well-known problem in multi-hypothesis tracking that is not straightforward to solve. For the sake of fairness, we want to note that we would expect tracking performance to increase by several percent if extensions such as the track initiation logic from Sec. V-B were also incorporated into the MHT.

### F. Runtime performance

In the last column of Table II, we show the median of the extrapolated processing rates of the tracking algorithms based upon actually measured cycle times (without taking the detection stage into account). All methods have been hand-optimized for runtime performance. For an equal comparison, and with the application scenario of the service robot with limited on-board processing capabilities in mind, we restrict the tracker's CPU usage to a single thread.

It can be seen that the NNT, even with our extensions, is extremely efficient due to its simplicity, being able to theoretically process over 5000 tracking cycles per second in moderately crowded scenarios (Main Station), and still over 600 in very crowded scenarios (PedSim). The entire tracking framework, with 2 separate laser detectors for front and rear, runs in real-time at 35 Hz on our robot platform, with the tracker itself consuming less than 30% of a single CPU core even in crowded scenarios.

Looking at the cycle rates of the basic NNT on the PedSim dataset, which are in the order of around 1000 Hz, it easily becomes apparent why in such highly crowded scenarios with over 30 tracks in sensor range at a time, a hypothesis-oriented multi-hypothesis approach cannot succeed without massive parallelization. Since in the MHT, the data association step performed by the NNT needs to be repeated for every single hypothesis, we cannot expect more than $1000\,\mathrm{Hz} : 500\,\mathrm{hyp.} \approx 2\,\mathrm{Hz}$ on a single CPU core assuming

[7]These results are 8% worse than the baseline results reported in [2], because we track targets up to a maximum range of 20 meters (instead of 12m) and due to use of a different CLEAR-MOT implementation.

[8]In [26], for only four tracks up to 1000 hypotheses are generated.

we want to generate at least 500 hypotheses. Even on a processor capable of executing 8 threads in parallel, the expected frame rate would drop below 20 Hz without yet running any detection or higher-level perception components.

## VI. Conclusion

In this paper, we have presented a modular, ROS-based framework for people tracking in crowded environments. As we have demonstrated by the integration of multiple person detectors, some of them from third-party sources, and two different tracking methods, our framework is easily extensible. We believe that our framework can be of benefit to researchers in service and assistance robotics, human-robot-interaction, and people tracking in particular, as it covers the entire people tracking pipeline, including powerful visualization and evaluation tools. Secondly, we have demonstrated that the choice of data association method matters less than expected. For practical applications on resource-constrained service robots, simple data association methods combined with effective extensions like a track initiation logic can be a better choice than highly complex multi-hypothesis approaches.

In future work, we want to extend our evaluation to multi-modal sensor data and compare quantitatively against further publicly available state-of-the art tracking methods (e. g. [10]) on challenging data captured in a crowded airport environment (Fig. 1).

To resolve data association ambiguity during extended occlusions, we plan to integrate appearance-based cues whenever persons enter the field of view of an RGB-D sensor on our robot. In these cases, we could also inhibit the track initiation logic to allow detection of standing persons, if the RGB-D detector is sufficiently confident. Finally, we want to integrate other promising methods from the literature that have, to our knowledge, never been combined in a single system, including feedback of group information into person-level tracking [2], motion prediction informed by a social force model [8], and occlusion geodesics [28].

## References


[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1726–1743, Dec. 2013.

[2] M. Luber and K. O. Arras, "Multi-hypothesis social grouping and tracking for mobile robots," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, 2013.

[3] T. Linder and K. O. Arras, "Multi-model hypothesis tracking of groups of people in RGB-D data," in *IEEE Int. Conf. on Information Fusion (FUSION'14)*, Salamanca, Spain, 2014.

[4] B. Okal and K. O. Arras, "Towards group-level social activity recognition for mobile robots," in *IROS 2014 Workshop on Assistance and Service Robotics in a Human Environment*, Chicago, USA, 2014.

[5] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with mobile robots using sample-based joint probabilistic data association filters," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 99–116, 2003.

[6] F. Basso, M. Munaro, S. Michieletto, E. Pagello, and E. Menegatti, "Fast and robust multi-people tracking from RGB-D data for a mobile robot," in *Intelligent Autonomous Systems 12*, ser. Advances in Intelligent Systems and Computing, S. Lee, H. Cho, K.-J. Yoon, and J. Lee, Eds. Springer Berlin Heidelberg, 2013, vol. 193, pp. 265–276.

[7] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[8] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, USA, 2010.

[9] M. Luber, L. Spinello, and K. O. Arras, "People tracking in RGB-D data with online-boosted target models," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.

[10] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[11] C. Martin, E. Schaffernicht, A. Scheidig, and H.-M. Gross, "Multi-modal sensor fusion using a probabilistic aggregation scheme for people detection and tracking," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 721 – 728, 2006, selected papers from the 2nd European Conference on Mobile Robots (ECMR 2005) 2nd European Conference on Mobile Robots.

[12] N. Bellotto and H. Hu, "Multisensor-based human detection and tracking for mobile service robots," *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, vol. 39, no. 1, pp. 167–181, 2009.

[13] K. O. Arras, O. Martínez Mozos, and W. Burgard, "Using boosted features for the detection of people in 2d range data," in *Proc. of the Int. Conf. on Robotics & Automation*, 2007.

[14] M. Munaro, F. Basso, and E. Menegatti, "Tracking people within groups with RGB-D data," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2012.

[15] J. Correa, J. Liu, and G.-Z. Yang, "Real time people tracking in crowded environments with range measurements," in *Social Robotics*, ser. Lecture Notes in Computer Science, G. Herrmann, M. Pearson, A. Lenz, P. Bremner, A. Spiers, and U. Leonards, Eds. Springer International Publishing, 2013, vol. 8239, pp. 471–480.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, pp. 83–97, 1955.

[17] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[18] Y. Bar-Shalom, *Tracking and Data Association*. San Diego, CA, USA: Academic Press Professional, Inc., 1987.

[19] Z. Hu, H. Leung, and M. Blanchette, "Statistical performance analysis of track initiation techniques," *Signal Processing, IEEE Transactions on*, vol. 45, no. 2, pp. 445–456, Feb 1997.

[20] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995.

[21] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *Journal of Image Video Processing*, vol. 2008, 2008.

[22] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, June 2013, pp. 735–742.

[23] B. Ristic, B.-N. Vo, D. Clark, and B.-T. Vo, "A metric for performance evaluation of multi-target tracking algorithms," *Signal Processing, IEEE Transactions on*, vol. 59, no. 7, pp. 3452–3457, July 2011.

[24] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. on Automatic Control*, vol. 24, no. 6, 1979.

[25] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 18, no. 2, 1996.

[26] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, USA, 2008.

[27] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[28] H. Possegger, T. Mauthner, P. Roth, and H. Bischof, "Occlusion geodesics for online multi-object tracking," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1306–1313.

# Taking a Closer Look at People Tracking in Challenging Environments Using a Novel Multi-Modal Evaluation Framework

Timm Linder[1] Stefan Breuers[2] Bastian Leibe[2] Kai O. Arras[1]

***Abstract*— Tracking people is a key technology for robots and intelligent systems in human environments. Many person detectors, filtering methods and data association algorithms for people tracking have been proposed in the past 15+ years in both the robotics and computer vision communities, achieving decent tracking performances from static and mobile platforms in real-world scenarios. However, little effort has been made to compare those methods, analyze their performance using different sensory modalities and study their impact on different performance metrics. In this paper, we propose a fully integrated real-time multi-modal laser/RGB-D people tracking framework for moving platforms in environments like a busy airport terminal. We conduct experiments on two challenging new datasets collected from a first-person perspective, one of them containing very dense crowds of people with up to 30 individuals within close range at the same time. We consider four different, recently proposed tracking methods and study their impact on seven different performance metrics, in both single and multi-modal settings. We extensively discuss our findings, which indicate that more complex data association methods may not always be the better choice, and derive possible future research directions.**

## I. Introduction

People tracking from a first-person perspective using a mobile sensor platform has been studied in the robotics and computer vision communities for over a decade, and various detection methods for different sensor modalities, as well as tracking algorithms have been proposed. Often, complex, generic data association methods are combined with extensions that are specific to the application domain to better deal with the frequent occlusions in such environments and model people's behavior [1], [2]. However, very recent work [3] reminds us that although complex data association methods, such as JPDAF or MHT, have been shown to deliver better performance in application areas with high-clutter environments like radar tracking [4], no systematic comparison between simpler and more complex data association methods has been performed for people tracking, where false positive detections occur systematically rather than randomly. Also, most systems focus only on a single sensor modality, and are tested in simple environments with only few tracked persons and limited dynamics.

In this paper, we want to go one step further and examine how well some recent, publicly available tracking methods perform in challenging, highly crowded and dynamic scenarios such as a busy airport terminal (Fig. 1). Following recent trends in the computer vision community towards a standardized benchmark for multi-object tracking methods [5], [6], and to enable a fair comparison of different tracking methods, we integrate them into a common framework and provide them with the same set of detections as input.

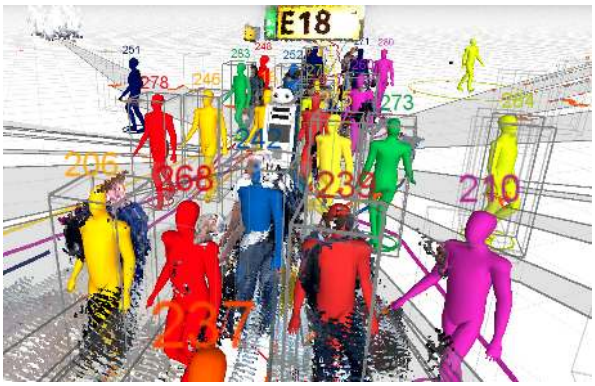



Fig. 1. Typical example of a crowded, dynamic situation in an airport terminal with frequent occlusions where we want to robustly and efficiently track persons from a first-person perspective with our mobile service robot platform, which can (barely) be seen in the middle of the picture.

Our contributions are:

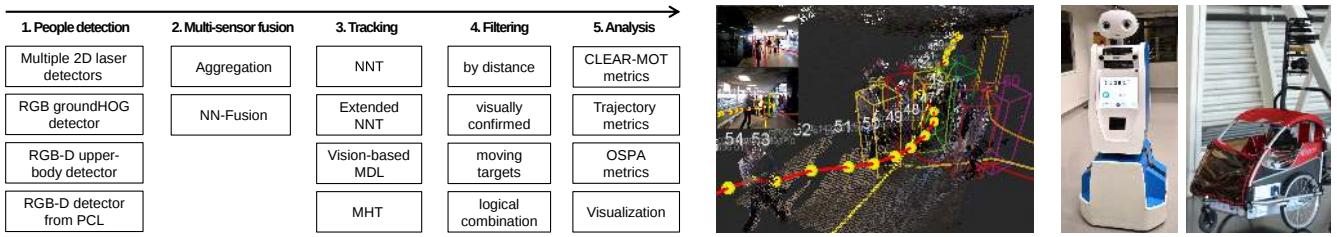
- An extensive ROS-based framework that provides the tooling for the systematic evaluation of multi-modal people tracking algorithms under identical conditions
- A comparison of four state-of-the-art real-time tracking systems that have been integrated into the framework, with focus on both tracking quality and runtime performance – including a proven MDL-based tracking approach from the computer vision community [7]
- Experiments on 2 challenging new datasets with RGB-D and 2D laser data from a first-person perspective
- A thorough discussion of strengths and weaknesses of current methods in these scenarios, and possible future directions of research.

Large parts of our framework, including our new multi-modal annotation tool and the parameters used to obtain our results, will be made publicly available as open source, to allow researchers to quickly reproduce our results on their own datasets and to evaluate their own algorithms using our framework.

[1] Social Robotics Laboratory, University of Freiburg, Germany. http://srl.informatik.uni-freiburg.de, {linder,arras}@cs.uni-freiburg.de

[2] Computer Vision Group, RWTH Aachen University, Germany. http://www.vision.rwth-aachen.de/, {breuers,leibe}@vision.rwth-aachen.de

This work has been partly supported by the European Commission under contract number FP7-ICT-600877 (SPENCER).




(a) Our modular, multi-modal tracking framework (b) Multi-modal annotation tool (c) Robot (d) Sensor platf.

Fig. 2. (a) All components of our framework are implemented as separate, reusable ROS modules, most of them open source. So far, we have integrated four existing tracking methods [7]–[10] into our framework for a fair comparison under identical conditions. (b) Our new multi-modal track annotation tool, based upon *RViz*. The 3D visualization encompassing RGB-D and laser point clouds as well as annotated trajectories (red line with yellow waypoint markers), along with 2D camera views with projected annotations (small images), significantly speeds up the annotation process. (c) Our service robot platform, equipped with front- and rear-facing 2D laser and RGB-D sensors. (d) Our mobile sensor platform, in a similar sensor configuration.

## II. Related work

People detection and tracking are of high interest to both robotics and computer vision. While both communities have, rather individually, made significant progress in the past, there has recently been a trend towards combining multiple methods and modalities as the available computational power for real-time detection and tracking is becoming larger, also on mobile robot platforms.

In robotics, often laser sensors are used to cover a large field of view, especially for mapping and navigation. The methods to detect people in this setup are based on simple ad-hoc classifiers looking for local minima in the scan [11], [12], or more elaborate person detectors [13]. In vision, camera and RGB-D sensor information is used as an input stream to the detection pipeline. Often HOG features are used to detect full bodies [14], [15], while upper-body detectors like [7] are better suited to detect nearby persons.

In both areas, missed detections and false alarms need to be compensated by a tracking algorithm, which often uses some form of data association method. [9] and [16] use the most simple NN method or NN-JPDA. Another NN tracker, [8], is using a more sophisticated track initiation and deletion logic and interacting multiple models (IMM). The more complex multi-hypothesis tracking methods [17], [18] are known to outperform simple NN methods in radar tracking [4], and have also been used for people tracking purposes [10] along with extensions such as a social force model or person-level feedback from group-tracking [1], [19]. The vision based MDL-tracker [7] is also loosely based on MHT, but formulates it as a quadratic pseudo-boolean optimization problem, solved via mininum description length (MDL). As it was specifically designed for visual data, it allows for the incorporation of an appearance model. Most trackers use an Extended Kalman Filter (EKF) to incorporate a constant velocity motion model of pedestrians.

With higher computational power, it has become possible to use multi-modal sensor platforms, equipped with both laser and RGB-D sensors, especially on service robots. This combination makes it possible for the robots to deal with the challenges in their highly crowded and dynamic field of operation. While a few multi-modal systems have been presented in the past [9], [20], [21], to the best of our knowledge, a consistent comparison of different people tracking approaches in a multi-modal setup in challenging environments is still missing in robotics. Even in the vision community, a standardized baseline evaluation of existing tracking methods has just begun [6]. Being aware of the challenges of groundtruth evaluation, as discussed in [5], we aim at providing a reusable, multi-modal framework that enables a consistent, comparative evaluation.

## III. Our framework

Fig. 2a gives an overview of the main components of our modular people tracking framework. All of these components are fully integrated into ROS and will be made publicly available on GitHub[1] at publication of this paper. In the following, we will briefly describe the most important components, starting from the left at the detection layer.

### A. Detection

*2D laser.* For people detection in 2D laser range data, we use a random forest classifier trained on the laser features described in [13]. While our ROS-based implementation, using classifiers from the OpenCV library, also allows to use other classifiers such as Adaboost or SVM, the random forest (with 15 trees and maximum depth of 10) performed best on our manually annotated training/test data set recorded in a pedestrian zone [19] using a SICK LMS 500 laser scanner at a height of 70 cm and 0.25 degrees angular resolution. After a separate ROS node has segmented the laser scans using a variant of jump-distance clustering, the detector computes a set of geometric 2D features on each segment which are then fed to the classifier.

*Monocular vision and RGB-D.* For person detection in RGB-D, the existing depth template-based upper-body detector described in Jafari et al. [7], which runs in real-time at 20-30 Hz on the CPU, as well as their CUDA-based, monocular groundHOG detector [14] have been integrated. We also extended the RGB-D person detector from [22], which applies a HOG classifier on candidate regions extracted from a depth-based height map, with GPU acceleration.

[1] https://github.com/spencer-project/spencer_people_tracking

*Fusing detections.* For multi-sensor people tracking, our framework allows to flexibly combine detections from multiple modalities using an easy-to-setup detection-to-detection fusion scheme that works even when the particular tracking algorithm was not specifically designed to cope with detection input from multiple sources, as most of the trackers in our evaluation. Using greedy NN association, we first fuse detections from sensors with overlapping fields of view (*e. g.* front laser, front RGB-D) and then aggregate the resulting sets of detections that do not overlap (*e. g.* front and rear detections). As association cost, we use either the Euclidean or Mahalanobis distance between individual detections, or a cost computed in polar coordinates that penalizes discrepancies in distance less heavily (mainly useful for 2D image-based detectors that do not output precise depth estimates).

All detectors integrated into our framework output detections which adhere to the same ROS message format. A *DetectedPerson* comprises a position vector $\mathbf{z}'$ and its uncertainty $R'$ in a sensor-specific 3D coordinate frame, a scalar detection confidence, and some meta-data. This clearly defined interface allows to easily integrate additional detectors into the system, and provides the interface to the tracking module.

### B. Tracking

Up to now, we have integrated four different tracking systems into our framework for comparison purposes. We will shortly outline these approaches in the following:

*Nearest-neighbor tracker [9].* This is a very fast tracker based upon a nearest-neighbor data association that has recently been integrated into ROS by the authors of [9]. Motion prediction is performed via an Extended Kalman Filter (EKF) using a constant velocity (CV) motion model, and tracks are initiated if a minimum number of detections occur within a small radius. Track deletion takes place if the track covariance exceeds a certain limit. A more advanced NN-JPDAF association method, not used in our experiments, is also provided.

*Extended nearest-neighbor tracker [8].* Also based upon greedy NN data association, this recent work of our own was developed with robustness and computational efficiency in mind especially in highly crowded scenarios. Compared to [9], it includes a velocity-based track initiation logic that only initiates new tracks if a given amount of detections appear with a consistent velocity profile that is compatible with typical human walking speeds. Track deletion occurs when the number of tracking cycles without matching detection exceeds a certain threshold, and a distinction is made between 'young' and 'mature' tracks that have already existed for a while; young tracks are deleted after a smaller number of cycles, since they often represent false alarms. For motion prediction, an IMM approach is used that combines multiple CV and coordinated turn models.

*Multi-hypothesis tracker [10].* As an additional baseline method for comparison, we use a variant of the multi-hypothesis tracker (MHT) after Reid *et al.* [17] and Cox & Hingorani [18] with explicit occlusions labels [10]. This probabilistic tracker does not incorporate any track initiation logic. Instead, new track creation is modelled via a Poisson process. Various extensions (such as incorporation of group-level feedback or a social force model) have been proposed in the past, but are not used in our experiments.

*Vision-based MDL tracker [7].* This method is based on the work of Leibe *et al.* [23] and uses the tracking framework of [24], [25] to build an overcomplete set of track hypotheses, similar to MHT. Via bi-directional EKF new trajectories are generated for the current frame by following the motion model backwards in time, while existing trajectories are extended from the last to the current frame. Each track then receives an individual score based on the appearance, motion model and confidence of inlier detections, while the interaction cost between tracks takes into account physical overlap and shared detections. Selecting the best subset of hypotheses from the score matrix is then formulated as a quadratic binary problem and solved in an MDL fashion by the multi-branch method of [26].

In all of the examined tracking systems, person detections arrive in their sensor-specific coordinate frame and are instantaneously transformed into a locally fixed frame (based upon robot odometry) that does not move with the robot. This ensures that the motion prediction of tracked persons is independent from the robot's ego-motion. In the resulting set of measurements $Z = \{\mathbf{z}_1, ..., \mathbf{z}_n\} \subset \mathbb{R}^2$, we drop the $z$ coordinate as we only track in 2D world coordinates.

### C. Groundtruth annotation and evaluation

To our knowledge, no multi-modal track annotation tool for 2D/3D laser, RGB-D and stereo data is currently publicly available. Our new ROS-based multi-modal annotation tool, partially shown in Fig. 2b, leverages the powerful visualization capabilities of the ROS visualization tool *RViz* and *rqt* to enable annotating people directly in 3D world space in the RGB-D and laser point clouds. For reference purposes, annotations and 2D laser scans are also projected into the camera images. By placing trajectory waypoints at regular intervals (e. g. every 0.5 sec or 2.0 sec, depending on the dynamics of the scene) and interpolating in between, the annotation process is significantly sped up.

For tracking performance evaluation purposes, we have integrated and extended a publicly available Python implementation of the CLEAR-MOT metrics, and implemented further trajectory-based metrics (both described in Sec. IV-B), as well as the OSPA metrics [27] (not used in this work).

## IV. Experiments

### A. Datasets

For our experiments, we have recorded two entirely new, multi-modal datasets (cf. Fig. 3). The *Motion Capture Sequence* has been recorded in a narrow lab environment in front of our robot platform, shown in Fig. 2c, which remains stationary throughout this sequence. The recorded sensor data includes a 190-degree frontal 2D laser scan from a

(a) Motion Capture Sequence

(b) Airport Sequence 03

Fig. 3. Example RGB frames from our new datasets.

SICK LMS 500 sensor, and the data from a front-facing Asus Xtion Pro Live RGB-D sensor. In this four-minute sequence, which we will make publicly available, four persons that wear motion capture markers on their heads for groundtruth acquisition are moving around in highly dynamic and erratic patterns, very frequently occluding each other and stopping or accelerating abruptly. This dataset is mainly intended to simulate human-robot interaction, in which case people often 'play' with the robot, or try to challenge its tracking abilities.

The second sequence, *Airport Sequence 03*, is part of a much larger dataset that was recorded at a major European airport, used by around 150,000 passengers each day, using a moving sensor platform (Fig. 2d) that closely replicates the sensory setup on our robot. The dataset includes 2D laser range data recorded from two back-to-back SICK LMS 500 scanners at 70 cm height, covering a *full 360-degree horizontal field of view* around the robot. It also includes RGB-D data from two Asus sensors mounted in horizontal orientation and facing into forward and rearward direction.

In the first half of this sequence, the sensor platform remains stationary and observes a dense flow of passengers disembarking from an airplane. In the second half, the platform joins the flow of people towards a large, open area inside the terminal. During the entire 4-minute sequence, the platform is almost constantly surrounded by 20–30 persons that follow various motion patterns at different walking speeds and undergo many severe occlusions. In total, 172 ground truth tracks have been manually annotated using our new multi-modal annotation tool. For our experiments, we ignore all groundtruth tracks at a distance of greater than $12.0$m as correctly annotating tracks becomes highly challenging above this distance due to extreme occlusions and increasing inaccuracy in sensor calibration.

### B. Evaluation metrics

A commonly used measure for evaluating multi-object tracking performance are the CLEAR-MOT metrics [28]. Besides counting false positives (FP), false negatives (FN) and ID switches (IDS), they define an aggregate error measure called MOT Accuracy (MOTA) as

$$\mathrm{MOTA} = 1 - \frac{\sum_k (\mathrm{FP}_k + \mathrm{FN}_k + \mathrm{IDS}_k)}{\sum_k \mathrm{GT}_k},$$

where $k$ is the tracking cycle index. The optimal MOTA score is 1.0, and MOTA can reach negative values if the tracker makes more errors than there are ground truth objects GT over the entire duration of the dataset.

As discussed extensively in [5], MOTA scores can vary between implementations and are highly dependent on meta-parameters such as the matching distance threshold $\theta_d$ and the way in which track hypotheses are assigned to groundtruth objects. In our version, we compute groundtruth correspondences using a variant of the Hungarian method based upon Euclidean distances between object centroids in world coordinates, with $\theta_d = 1\mathrm{m}$. We ignore all correspondences where the groundtruth track is physically occluded, which is determined by searching for associated laser points within a radius of $0.3m$ of the annotated position, shifted towards the sensor origin by $0.2m$ to take into account that the laser sensor only perceives the surface of the person.

One caveat is that the number of ID switches (IDS) has very low influence on overall MOTA, as FP and FN counts are often significantly higher in comparison. Therefore, the absolute number of ID switches is often used as a second, separate measure when evaluating people tracking performance. However, a tracking system with lower track recall (i. e. which tracks less persons by, for instance, initiating tracks very reluctantly) almost certainly generates less ID switches. Very recent research in the computer vision community [6], which we adopt here, instead motivates to compute the *relative number of ID switches*, rIDS, defined as a product of the absolute number of ID switches and the inverse of the recall over all frames, $\mathrm{IDS} \cdot \mathrm{GT}/\mathrm{TP}$.

Finally, we also compute the trajectory-based measures of mostly tracked (MT) and mostly lost (ML) persons [29], denoting the number of groundtruth tracks that have been tracked for more than 80% or less than 20% of their length.

### C. Experimental setup

All of our experiments were conducted on a high-end gaming laptop equipped with a quad-core Intel Core i7-4700 MQ processor and 8 GB of RAM under Ubuntu 14.04 with ROS Indigo. Each single experiment has been run at least 3 times and metrics have been averaged to ensure stable results that are not negatively affected by the not fully deterministic message passing, synchronization and transform lookups in ROS. For the computationally more complex experiments on the airport dataset sequence, we have pre-recorded all detections to ensure that all tracking algorithms are always fed with the same input for a fair comparison.

### D. Parameter selection

As highlighted in [5], for evaluations of multi-person tracking it is important that parameters of the tracking algorithm are tuned on a separate validation dataset to verify its generalization capabilities and avoid overfitting. With this in mind, we carefully tuned all of our algorithms on separate, similar, but not identical datasets. Specifically, for tuning the NNT [9], the Extended NNT [8] and the MHT [10], we used the laser-based *Freiburg Main Station* dataset (cf. e. g. [1]), as well as a synthetically generated dataset via a combination of the pedestrian simulator *PedSim* and *Gazebo* (see [8] for

details). The vision-based MDL tracker has been tuned on the *ETH* dataset [30] recorded in a pedestrian zone.

## V. Results

In Tables I–IV, we present quantitative results of the different tracking methods for each modality on our datasets. Qualitative results are available on our YouTube channel[2]. As the MDL tracker [7] currently only supports image-based detections from the upper-body and groundHOG detectors, we only use it in experiments with the front RGB-D sensor.

### A. Comparison of different tracking approaches

Comparing the results of the different tracking approaches under identical conditions (sequence, modality), we note that the simple NN approaches often generate the best MOTA score. This might be due to a lower number of parameters, which could result in better generalization capabilities regarding new scenarios. The Extended NNT is superior to the NNT in terms of MOTA and FP%, most likely due to its track initiation and deletion logic. Especially on the *Motion Capture Sequence* with four groundtruth tracks, one or two ghost tracks are enough to cause bad FP scores for methods without a sophisticated initiation logic, such as NNT and MHT. Interestingly, both of these perform very similarly in most of the tested scenarios concering MOTA and FP%. On the other side, the miss ratio is often the highest for the Extended NNT, and caused by delayed track initiation.

Both multi-hypothesis methods seem to suffer from frequent switching between hypotheses, a problem well known in multi-hypothesis tracking. This results in a high number of relative ID switches. However, in front RGB-D only (Tab. I), the MDL-Tracker gives best FP% and thus a MOTA score comparable to the one of Extended NNT. Note that MHT might obtain better results if parameters such as new track rates were re-tuned on the datasets used for testing, or if it were allowed to correct its decisions when running in an offline-manner, i.e., fixed-lag smoothing. Nevertheless, this can be problematic for real-time motion planning applications due to the introduced delay, and lowers MOTA when comparing always against the most recent groundtruth.

The simple NNT tracker dominates in the number of consistently tracked targets, i.e., higher MT and lower ML, due to a more straightforward initiation of tracks.

### B. Laser-only vs. multi-modal detections

Next, we want to investigate the benefits of the multi-modal sensor platform and the use of both 2D laser and RGB-D sensors. On the airport sequence, incorporating vision-based detections from groundHOG and upper-body increases the number of mostly tracked targets. This leads to a higher track recall and lower miss ratio for all approaches, at the cost of an increased FP%, ultimately resulting in a lower MOTA score (Tab. III). A more sophisticated fusion scheme of the different detector outputs might yield some improvement, however, a visual inspection of our naïve fusion scheme reveals no immediately apparent problems.

[2] https://youtube.com/spencereuproject

(a) (b) (c)

Fig. 4. Typical failure cases from the HOG detector caused by (a) clutter, (b) reflections on the floor or (c) on the walls. Reflections on walls and glass surfaces can sometimes also cause false laser detections.

Instead, further experiments reveal that the HOG detector causes many false alarms (Fig. 4) and provides imprecise depth estimates for distant persons, obtained by projecting image footpoints onto the estimated ground plane. Tab. IV shows the multi-modal result without HOG, but still using upper-body detections from the RGB-D sensor. The FP% decreases, but unfortunately also MT goes down and miss rate increases. Anyhow, general tracking quality improves, which is reflected in the highest MOTA score for each tracking approach so far using this configuration.

On the *Motion Capture Sequence*, all methods struggle with an extremely high FP%, except for Extended NNT, whose extensive track initiation logic can again compensate for false alarms. The resulting discrepancy of the MOTA scores is huge (75% vs. 8-15%). It seems here that the laser detector – instead of HOG – is responsible for most of the false positives, often in chairs and other furniture: when using only front RGB-D (Tab. I), FP% is around 50%-points lower.

### C. Filtering detections by a static map

As a static map of the environment is often available for navigation purposes anyway, we want to examine its use for false positive suppression. We rasterize a circle of 15 cm radius at the detection's position onto the given occupancy grid map. If less than 90 percent of all grid cells are free, we reject the detection. As we filter on the detection level, the process can be applied to any detector.

As no map had been recorded in the airport environment, we restrict this experiment to the *Motion Capture Sequence* (Tab. V), where it leads to an increase in MOTA of 15–65 percentage points for the different tracking approaches.

### D. Runtime performance

In the last column of each table, we show the median of the extrapolated processing rates of the tracking algorithms based upon actually measured cycle times (without taking the detection stage into account). All examined tracking systems are implemented in C++. Note that the rate of MHT is fixed to 30 Hz, generating as many hypotheses per cycle as possible in this time frame (at lower rates, the performance gets worse due to less frequent updates of the EKF).

The simple NNT is about 3 times computationally more efficient than the Extended NNT. Both outperform the two more complex methods MDL and MHT by two orders of

| Method | Airport Sequence 03 MOTA | rIDS | FP% | Miss% | MT | ML | Hz | Motion Capture Sequence MOTA | rIDS | FP% | Miss% | MT | ML | Hz |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NNT [9] | 27.7% | 227 | 39.4% | **32.5%** | **92** | **47** | **13701** | 60.7% | **131** | 23.6% | **14.3%** | 4 | 0 | **20726** |
| Extended NNT [8] | **44.4%** | **210** | 13.1% | 42.1% | 63 | 60 | 4287 | **69.8%** | 151 | 7.8% | 20.9% | 4 | 0 | 5637 |
| MHT [10] | 26.9% | 338 | 39.4% | 33.0% | 87 | 51 | 28 | 57.9% | 173 | 24.7% | 15.6% | 4 | 0 | 28 |
| MDL-Tracker [7] | 43.7% | 428 | **12.5%** | 43.1% | 36 | 59 | 53 | 60.7% | 373 | **4.8%** | 31.3% | 1 | 0 | 139 |

TABLE I

ONLY FRONT RGB-D DETECTIONS (SMALL FOV)

| Method | Airport Sequence 03 MOTA | rIDS | FP% | Miss% | MT | ML | Hz | Motion Capture Sequence MOTA | rIDS | FP% | Miss% | MT | ML | Hz |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NNT [9] | 59.7% | **236** | 20.8% | **19.3%** | **112** | 27 | **6184** | 25.6% | **54** | 70.4% | **3.3%** | 4 | 0 | **17968** |
| Extended NNT [8] | **62.8%** | 331 | **3.4%** | 33.5% | 68 | 35 | 2307 | **68.8%** | 60 | **25.3%** | 5.2% | 4 | 0 | 4988 |
| MHT [10] | 58.9% | 700 | 16.6% | 23.9% | 85 | **26** | 29 | 28.0% | 85 | 67.3% | 3.8% | 4 | 0 | 28 |

TABLE II

ONLY LASER DETECTIONS (LARGE FOV)

| Method | Airport Sequence 03 MOTA | rIDS | FP% | Miss% | MT | ML | Hz | Motion Capture Sequence MOTA | rIDS | FP% | Miss% | MT | ML | Hz |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NNT [9] | 45.7% | 325 | 36.4% | **17.7%** | **123** | **19** | **4590** | 14.8% | **55** | 81.7% | **2.7%** | 4 | 0 | **15703** |
| Extended NNT [8] | **62.1%** | **313** | **8.2%** | 29.4% | 96 | 26 | 2005 | **74.9%** | 58 | **20.1%** | 4.3% | 4 | 0 | 4690 |
| MHT [10] | 46.3% | 692 | 34.9% | 18.2% | 117 | 22 | 31 | 8.6% | 74 | 87.6% | 2.9% | 4 | 0 | 29 |

TABLE III

MULTI-MODAL DETECTIONS (LARGE FOV)

| Method | Airport Sequence 03 MOTA | rIDS | FP% | Miss% | MT | ML | Hz | Motion Capture Sequence MOTA | rIDS | FP% | Miss% | MT | ML | Hz |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NNT [9] | 62.1% | **226** | 18.7% | **19.0%** | **114** | 27 | **6100** | 18.1% | **52** | 77.6% | 3.7% | 4 | 0 | **15857** |
| Extended NNT [8] | **64.2%** | 262 | **3.3%** | 32.4% | 77 | 33 | 2222 | **77.4%** | 62 | **16.5%** | 5.4% | 4 | 0 | 4744 |
| MHT [10] | 60.2% | 676 | 17.2% | 22.0% | 97 | **24** | 29 | 17.8% | 76 | 77.7% | **3.6%** | 4 | 0 | 28 |

TABLE IV

MULTI-MODAL DETECTIONS WITHOUT HOG (LARGE FOV)

magnitudes, and require less than 10 percent CPU usage even in very crowded environments.

## VI. DISCUSSION

In the following, we discuss the most important conclusions that we can draw from our experiments on the crowded airport dataset and the dynamic motion capture sequence.

### A. Influence of detector performance

A major observation that we made during our experiments is that detector performance is the single, most important factor influencing tracking performance which goes far beyond the impact of the chosen tracking algorithm. In a nutshell, none of the examined tracking methods deal really well with high false positive rates. During initial experiments, we used a 2D laser detector trained on a different sensor model, and using a less restrictive selection of positive and negative training samples. This detector caused extremely bad MOTA scores between $-3.3$ and $-1.3$ due to enormous false positive rates (> 200%). None of the examined methods could cope with this high number of false positives, which occur systematically and repeatedly at the same locations. Although the track initiation logic of the extended NNT was able to suppress a significant amount of the false positives, MOTA still did not exceed $-1.3$. Even though multi-hypothesis trackers have been shown to work well in (random) high clutter in radar tracking [4], the worst MOTA score was obtained using the MHT, which does not possess any dedicated track initiation logic. After incorporating the static occupancy grid map to filter out false detections beforehand, MOTA scores of all examined approaches became positive, but were still significantly below the levels presented in Sec. V.

### B. Integrating 2D image-based detections

While a vision-based tracker can significantly benefit from 2D image-based detections (e. g. from HOG) that extend its maximum tracking distance beyond the useful working range

| | Motion Capture Sequence | | | | |
|---|---|---|---|---|---|
| Method | MOTA | rIDS | FP% | Miss% | Hz |
| NNT [9] | 78.0% | **50** | 18.5% | 2.9% | **19050** |
| Extended NNT [8] | **89.4**% | 59 | **5.3**% | 4.6% | 4926 |
| MHT [10] | 73.8% | 71 | 21.9% | 3.4% | 28 |

TABLE V

MULTI-MODAL DETECTIONS (LARGE FOV) + STATIC MAP

of RGB-D sensors (around 6–7m), their depth estimates are often very imprecise. In a multi-modal setup where precise laser measurements are available ($\sigma \approx 3$cm), using HOG detections as a direct input to the tracking algorithm may therefore be detrimental. Instead, laser-based detections may be a better choice to cover far detection ranges, while image-based detections could be used to validate laser detections visually, if an association can be established.

### C. The choice of tracking parameters

Our experience shows that the correct choice of parameters significantly outweighs the choice of data association. Tracking approaches with only few parameters may generally be the preferrable choice, as they may generalize better towards new scenarios. Especially complex, probabilistic multi-hypothesis approaches often require re-learning or manual tuning by an expert of parameters such as new-track or deletion Poisson rates that depend on the given scenario and can vary with location and time (e. g. when a new plane arrives at an airport and the passengers start disembarking). Also, automatic parameter learning approaches as outlined in [8], [31] may help to simplify the process. To make our results easier to reproduce and allow researchers from other fields (e. g. HRI) to benefit from our findings, we will share all parameter configurations used in our experiments online.

### D. Trade-off between FPs, miss rate and ID switches

Another lesson we learn from our experiments is that the choice of parameters greatly depends on the desired application scenario. There appears to be no universal set of parameters that fully accommodates all requirements, as a trade-off has to be made between attaining a low false positive count, a low miss rate, and a low number of ID switches. The first two can be important when using the people tracker output for socially aware navigation, since high false positive rates (i. e. ghost tracks) could freeze the robot, while missed tracks can cause the robot to behave impolitely or even endanger people. On the other hand, in person guidance scenarios, it is of utmost importance to maintain the ID of a tracked person as long as possible, while false positives might not be such a large issue.

As shown in our experiments, low false positive rates can be achieved by a dedicated track initiation logic, pre-filtering on a static map, and early track deletion. The first two options can cause the tracker to miss certain (e. g. static) tracks, while the last option may result in ID switches if the track suddenly reappears after an occlusion.

### E. Importance of standardized tracking metrics

Even minor differences in tracking metrics implementation or its parameters can have significant influence on results. We agree with findings from the vision community [5] which underline the importance of using a standardized evaluation script and the same detection input for all tracking systems. Our proposed tracking framework is, to our knowledge, the first that allows for *multi-modal* data annotation in RGB-D, 2D/3D laser and potentially stereo data, and enables a systematic evaluation and comparison of different tracking methods and detectors in a joint framework.

### F. Which tracking approach to choose?

Finally, we attempt to answer the question which of the examined tracking methods to choose for real-time people tracking from a mobile platform in very crowded and dynamic environments. Looking at the multi-modal results on the motion capture sequence (Table III, right), we see that the same, underlying NN data association method of [8], [9] delivers an astonishing difference in MOTA performance of 60%, depending on the presence or lack of a dedicated track initiation logic. On the other hand, on the airport sequence, we observe only a a 0.5% difference in MOTA between simple NN and complex MHT data association. Therefore, as already hinted at in [8], it appears that incorporating promising tracking extensions (e. g. [1], [2]) into a simple data association scheme might be the way to go. The computation time which is saved by refraining from using a more complex multi-hypothesis data association method could instead be spent on higher-level perception, or to improve detector performance, which has a high impact as previously discussed. Both of the discussed NN-based approaches are relatively easy to configure, show good performance on our test datasets, and run at low CPU usage (<10% on a single core) – which is crucial on a mobile service robot platform that also needs to localize itself, plan and navigate.

Here, the hypothesis-oriented MHT after Reid [17] and Cox & Hingorani [18] may also be at disadvantage in very crowded environments. Since the entire state of the scene is represented within each single hypothesis, a very large number of hypotheses may be needed to adequately represent all likely combinations of possible track states. In [10], up to 1000 hypotheses are generated for just 4 person tracks, each one involving the same data association that the NN-based methods only need to perform once. Generating as many hypotheses as possible within a given time window, as in our experiments, ensures a certain minimum cycle rate to be met, but may result in only few hypotheses being generated.

Finally, scenarios where some delay in decision making can be tolerated, such as offline video analysis or static observation of people behavior, allow for a different mode of evaluation where the delayed selection of the best hypothesis can be taken into account, by deferring matching with the groundtruth by a certain number of tracking cycles. We believe that in these cases, the multi-hypothesis approaches [7], [10] can show their full potential and attain higher scores.

### G. Future directions of research

Using solely detectors with relatively low false-positive rate, the difference in tracking metrics between various tracking approaches and implementations becomes surprisingly small. Visually analyzing the remaining ID switches that still occur on the airport and motion capture sequences, we believe that in these cases, the motion model provides insufficient information and full person reidentification is required. Implementing a robust reidentification module can be very challenging in scenarios such as the airport, which is used by over 150,000 passengers per day. An open question is still how to deal with tracks that first re-appear in 2D laser and need to be assigned a preliminary ID, before potentially getting visually re-identified as a previously seen person; in person guidance scenarios, this issue potentially needs to be dealt with on the task planning level.

## VII. Conclusion

In this paper, we have presented a multi-modal people tracking framework into which we have integrated four existing tracking approaches of varying complexity, in order to study them on two challenging new, multi-modal datasets – one of them recorded from a static platform in a highly dynamic HRI scenario, and another one from a moving platform inside a crowded airport terminal. We have carefully analysed the performance of these existing methods with regard to multiple tracking metrics under different multi-modal configurations, identified and extensively discussed their strengths and weaknesses, shared some learned lessons and drawn conclusions that may guide possible future directions of research. Finally, we want to encourage other researchers to integrate their own detectors and tracking algorithms into our framework, and share their results.

## References


[1] M. Luber and K. O. Arras, "Multi-hypothesis social grouping and tracking for mobile robots," in *Proceedings of Robotics: Science and Systems*, 2013.

[2] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[3] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[4] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, 1999.

[5] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, 2013.

[6] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1504.01942

[7] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[8] T. Linder, F. Girrbach, and K. O. Arras, "Towards a robust people tracking framework for service robots in crowded, dynamic environments," in *Assistance and Service Robotics Workshop (ASROB-15) at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[9] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide, "Real-time multisensor people tracking for human-robot spatial interaction," in *Workshop on Machine Learning for Social Robotics at International Conference on Robotics and Automation (ICRA)*, 2015.

[10] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[11] A. Fod, A. Howard, and M. Mataríc, "Laser-based people tracking," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[12] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research (IJRR)*, vol. 22, no. 2, 2003.

[13] K. O. Arras, O. Martínez Mozos, and W. Burgard, "Using boosted features for the detection of people in 2d range data," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[14] P. Sudowe and B. Leibe, "Efficient use of geometric constraints for sliding-window object detection in video," in *Computer Vision Systems*, 2011.

[15] L. Spinello and K. O. Arras, "People detection in RGB-D data," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[16] N. Bellotto and H. Hu, "Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters," *Autonomous Robots*, vol. 28, no. 4, 2010.

[17] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. on Automatic Control*, vol. 24, no. 6, 1979.

[18] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. on Pattern Analysis & Machine Intell.*, vol. 18, no. 2, 1996.

[19] T. Linder and K. O. Arras, "Multi-model hypothesis tracking of groups of people in RGB-D data," in *IEEE Int. Conf. on Information Fusion (FUSION'14)*, 2014.

[20] E. Schaffernicht, C. Martin, A. Scheidig, and H.-M. Gross, "A probabilistic multimodal sensor aggregation scheme applied for a mobile robot," in *KI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, 2005, vol. 3698.

[21] L. Spinello, R. Triebel, and R. Siegwart, "Multiclass multimodal detection and tracking in urban environments," *The International Journal of Robotics Research*, vol. 29, no. 12, 2010.

[22] M. Munaro, F. Basso, and E. Menegatti, "Tracking people within groups with RGB-D data," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[23] B. Leibe, K. Schindler, N. Cornelis, and L. V. Gool, "Coupled object detection and tracking from static cameras and moving vehicles," *IEEE Trans. on Pattern Analysis & Machine Intell.*, vol. 30, no. 10, 2008.

[24] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "Robust multiperson tracking from a mobile platform," *IEEE Trans. on Pattern Analysis & Machine Intell.*, vol. 31, no. 10, 2009.

[25] K. Schindler, A. Ess, B. Leibe, and L. Van Gool, "Automatic detection and tracking of pedestrians from a moving stereo rig," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 6, 2010.

[26] K. Schindler, U. James, and H. Wang, "Perspective n-view multibody structure-and-motion through model selection," in *Proc. of the European Conf. on Comp. Vision (ECCV)*, 2006.

[27] B. Ristic, B.-N. Vo, D. Clark, and B.-T. Vo, "A metric for performance evaluation of multi-target tracking algorithms," *Signal Processing, IEEE Transactions on*, vol. 59, no. 7, 2011.

[28] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *Journal of Image Video Processing*, 2008.

[29] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid-boosted multi-target tracker for crowded scene," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[30] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[31] M. Luber, G. D. Tipaldi, and K. O. Arras, "Better models for people tracking," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

# People Detection, Tracking and Visualization using ROS on a Mobile Service Robot

Timm Linder, Kai O. Arras

Social Robotics Laboratory, University of Freiburg
Georges-Köhler-Allee 074, 79110 Freiburg i. Br., Germany
{linder,arras}@cs.uni-freiburg.de

http://github.com/spencer-project

**Abstract.** In this case study chapter, we discuss the implementation and deployment of a ROS-based, multi-modal people detection and tracking framework on a custom-built mobile service robot during the EU FP7 project SPENCER. The mildly humanized robot platform is equipped with five computers and an array of RGB-D, stereo and 2D laser range sensors. After describing the robot platform, we illustrate our real-time perception pipeline starting from ROS-based people detection modules for RGB-D and 2D laser data, via nodes for aggregating detections from multiple sensors, up to person and group tracking. For each stage of the pipeline, we provide sample code online. We also present a set of highly configurable, custom RViz plugins for visualizing detected and tracked persons and groups. Due to the flexible and modular structure of our pipeline, all of our components can easily be reused in custom setups. Finally, we outline how to generate test data using a pedestrian simulator and Gazebo. We conclude with quantitative results from our experiments and lessons that we learned during the project. To our knowledge, the presented framework is the functionally most complete one that is currently available for ROS as open-source software.

**Keywords:** People detection, people tracking, group tracking, perception, service robot, mobile robot, sensors, visualization

## 1 Introduction

In this chapter, we discuss our experiences with the implementation and deployment of a ROS-based people detection and tracking framework on a complex, custom mobile service robot platform equipped with a large array of sensors.

### Contributions of the Book Chapter

One key contribution of this chapter is a set of reusable message definitions for a people and group detection and tracking pipeline. In our research, we demonstrated that these definitions can successfully be applied across different

sensor modalities and real-time detection and tracking algorithms, for which we provide exemplary implementations. Based upon these message definitions, we also provide a reusable and highly configurable set of visualization plugins for the standard ROS visualization tool, RViz. To our knowledge, the resulting people detection and tracking framework is the functionally most complete one that is publicly available for ROS at present.

The code for the detection and tracking modules, as well as our message definitions and visualization plugins, can be found online[1] in a Git repository, mostly under a BSD license. The content of this repository needs to be cloned into a ROS workspace and built using `catkin_make`. An up-to-date list of dependencies can be found in the accompanying Readme file. Our components have been tested on ROS Hydro and Indigo on 64-bit Ubuntu systems.

## 2 Background of the SPENCER Project

The aim of the EU FP7 research project SPENCER is to develop algorithms for service robots that can guide groups of people through highly dynamic and crowded pedestrian environments, such as airports or shopping malls, while behaving in a socially compliant manner by e. g. not crossing in between families or couples. Possible situations that such a robot could encounter are visualized in Fig. 1. To this end, robust and computationally efficient components for the perception of humans in the robot's surroundings need to be developed.

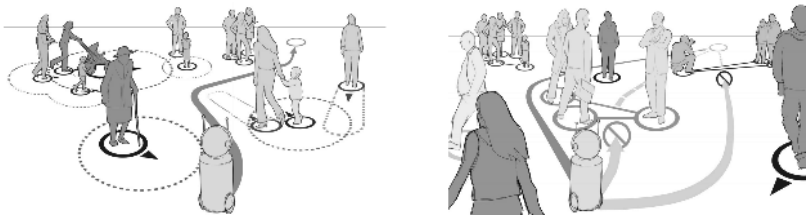


**Fig. 1.** Typical situations encountered by a mobile service robot in crowded pedestrian environments, such as shopping malls or airports. To be able to behave in a socially compliant way while driving, by for instance not crossing through a group, the robot needs to gain a precise understanding of the persons in its environment.

### 2.1 Robot Hardware and Sensory Setup

As none of the commercially available robot platforms offered the computational and sensory capabilities required for the research in SPENCER, a custom mobile robot was developed by an industrial partner within the project. The differential-drive robot platform built for SPENCER, shown in Fig. 2 (left), is around 2

[1] `https://github.com/spencer-project/spencer_people_tracking`

meters tall and equipped with two onboard Intel Core i7-3520M machines for planning and navigation, an i3-2120 machine for interaction, two i7-4700MQ gaming laptops with nVidia GeForce GTX 765M for perception and one embedded PowerPC system for low-level motion control. All systems communicate over a gigabit ethernet connection and, except for the embedded system, run ROS Indigo on Ubuntu 14.04.

The robot is equipped with two front- and two rear-looking RGB-D sensors, a front-facing stereo camera system and a pair of 2D laser scanners offering, when combined, a 360-degree coverage. For human-robot interaction, a touchscreen and a boarding pass reader have been integrated. Custom ROS wrappers for communication with the embedded system and the robot's head joints were developed as part of the project.

**Fig. 2.** *Left:* Picture and renderings of the SPENCER robot platform. *Right:* The custom-built mobile data capture platform, including a custom odometry solution using quadrature encoders in dynamo housings on both main wheels, connected to a microcontroller (bottom right picture). Both platforms contain numerous ROS-based software components distributed over multiple computers.

**Mobile Data Capture Platform** To obtain groundtruth data for the design, training and testing of new algorithms, we also built a mobile data capture platform (Fig. 2, right) consisting of an off-the-shelf bicycle trailer equipped with sensors in a similar configuration as on the actual robot. Wheel odometry for localization is obtained using a custom microcontroller solution that communicates with a ROS node on a laptop. For data recording using *rosbag*, we used 3 high-end laptops with fast SSDs. Overall, using this platform, we captured over 1.4 TB of data at a major European airport to train our perception components.

### 2.2 People Tracking Pipeline

Figure 3 shows the real-time people and group detection and tracking pipeline developed in the context of the SPENCER project.

Starting with sensory data such as 2D laser scans, RGB-D point clouds or stereo or monocular camera images, we first detect people using detectors devised specifically for the particular sensor modality (Sec. 3). The resulting person

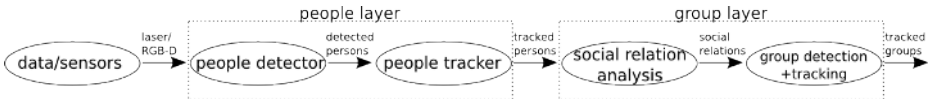



**Fig. 3.** People and group tracking pipeline developed during the SPENCER project.

detections are then fed into a person tracker (Sec. 4), which integrates the information over time and attempts to maintain a consistent ID for a given person for as long as possible. A simple approach for tracking entire groups of persons by estimating their spatial and social relations is presented in Sec. 5.

All of this becomes more complex if information from multiple detectors operating on different sensor modalities, such as 2D laser and RGB-D, shall be combined to make tracking more robust and cover a larger field of view (Sec. 6). Finally, using the powerful RViz visualization tool and custom plugins developed by us as well as a custom SVG exporter script, the outputs of the tracking pipeline can be visualized (Sec. 7). In Sec. 8, we briefly outline how to generate test data for experiments using a pedestrian simulator. Finally, we show qualitative results and discuss runtime performance in Sec. 9.

The entire communication between different stages of our pipeline occurs via ROS messages defined in the corresponding sections, which we explain in detail to encourage reuse of our components in custom setups. The architecture allows for easy interchangeability of individual components in all stages of the pipeline.

## 3 People Detection

In this section, we present ROS message definitions and exemplary implementations for detecting people in RGB-D and 2D laser range data. We start by a short summary of existing research in this field.

While still relatively expensive, 2D laser range finders offer data at high frequencies (up to 100 Hz) and cover a large field of view (usually around 180 degrees). Due to the sparseness, the sensor data is very cheap to process, but does not offer any appearance-based cues that can be used for people detection or re-identification. Early works based on 2D laser range data detect people using ad-hoc classifiers that find local minima in the scan [10, 27]. A learning approach is taken by Arras et al. [2], where a classifier for 2D point clouds is trained by boosting a set of geometric and statistical features.

In RGB-D, affordable sensors are gaining popularity in many indoor close-range sensing scenarios since they do not require expensive disparity map calculations like stereo cameras and usually behave more robustly in low lighting conditions. Spinello and Arras [28] proposed a probabilistically fused HOD (histogram of oriented depths) and HOG (histogram of oriented gradients) classifier. Munaro et al. [21] present a person detector in RGB-D that uses a height map-based ROI extraction mechanism and linear SVM classification using HOG features, which has been integrated into the Point Cloud Library (PCL). Jafari

et al. [12] at close range also use a depth-based ROI extraction mechanism and evaluate a normalized depth template on the depth image at locations where the height map shows local maxima corresponding to heads of people. For larger distances, a GPU-accelerated version of HOG is used on the RGB image.

### 3.1 ROS Message Definitions

In the following, we present our message definitions for person detection. Due to the multi-language support of ROS, the resulting messages can be processed by any kind of ROS node, regardless if implemented in C++, Python, Lua or Lisp. We want to emphasize that our message definitions are intentionally kept as simple and generic as possible, to allow reuse over a wide range of possible sensor modalities and detection methods. Therefore, for instance, we do not include image bounding boxes or visual appearance information which would be specific to vision-based approaches and not exist e. g. in 2D laser range data.

Our definition of a detected person is similar to a geometry_msgs/PoseArray, but additionally, for each detection we also specify a unique detection ID, a confidence score, a covariance matrix and the sensor modality. The detection ID allows the detection to be matched against e. g. the corresponding image bounding box, published under the same detection ID on a separate ROS topic. The covariance matrix expresses the detector's uncertainty in the position (and orientation, if known). It could, for instance, be a function of the detected person's distance to the sensor, and is therefore not necessarily constant. The confidence score can be used to control track initialization and to compute track scores. Finally, information about the modality can be used by a tracking algorithm to, for example, assign a higher importance to visually confirmed targets.

A spencer_tracking_msgs/**DetectedPerson** thus has the following attributes:

- *detection_id* [uint64]: Unique identifier of the detected person, monotonically increasing over time. To ensure uniqueness, different detector instances, should use distinct ID ranges, by e. g. having the first detector issue only IDs that end in 1, the second detector IDs that end in 2, and so on.

- *confidence* [float64]: A value between 0.0 and 1.0 describing the confidence that the detection is a true positive.

- *pose* [geometry_msgs/PoseWithCovariance]: Position and orientation of the detection in metric 3D space, along with its uncertainty (expressed as a $6 \times 6$ covariance matrix). For unknown components, e. g. position on the $z$ axis or orientation, the corresponding elements should be set to a large value[2]. The pose is relative to the coordinate frame specified in the DetectedPersons message (see below).

[2] We usually use a value of $10^5$ to indicate this. If set to infinity, the covariance matrix becomes non-invertible, causing issues later on during tracking.

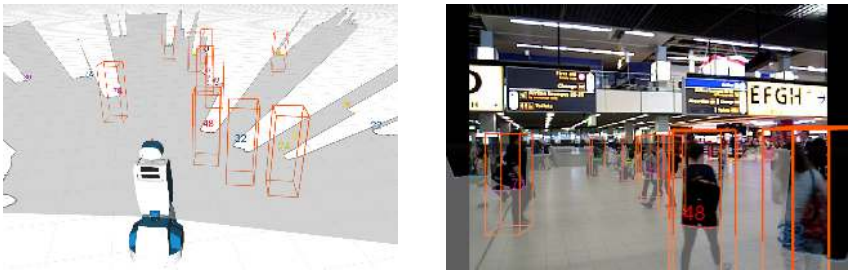



**Fig. 4.** *Left:* Area visible to the front laser scanner (in grey), candidate laser scan segments (with numbers that identify the segments), and laser-based person detections (orange boxes). *Right:* Projection of the laser-based person detections into a color image of the scene. Segment 72 on the left border is a false negative classification. As the 2D laser scanner has a very large horizontal field of view (190 degrees), the camera image does not cover the entire set of laser scan segments (e. g. segment 24 on the right).

– *modality* [string]: A textual identifier for the modality or detection method used by the detector (e. g. RGB-D, 2D laser). Common string constants are pre-defined in the message file.

A **DetectedPersons** message aggregates all detections found by a detector in the same detection cycle, and contains the following attributes:

– *header* [std_msgs/Header]: Timestamp and coordinate frame ID for these detections. The timestamp should be copied from the header of the sensor_msgs/LaserScan, Image or PointCloud2 message that the detector is operating on. Similarly, the coordinate frame can be a local sensor frame as long as a corresponding transformation into the tracking frame (usually "odom") exists in the TF hierarchy.

– *detections* [array of DetectedPerson]: The array of persons that were detected by the detector in the current time step.

### 3.2 Person Detection in 2D Laser Data

**Boosted Laser Segment Classifier** For people detection in 2D laser range data (Fig. 4), we use a re-implementation of [2] using an Ada-Boost implementation from the OpenCV library. The classifier has been trained on a large, manually annotated data set consisting of 9535 frames captured using our mobile data capture platform in a pedestrian zone, with the laser scanner mounted at about 75 cm height. We used an angular scan resolution of 0.25 degrees and annotated detections up to a range of 20 meters.

Prior to classification, the laser scan is segmented by a separate ROS node using either jump distance or agglomerative hierarchical clustering with a distance threshold of 0.4m. The combined laser-based segmentation and detection system is launched via the command-line

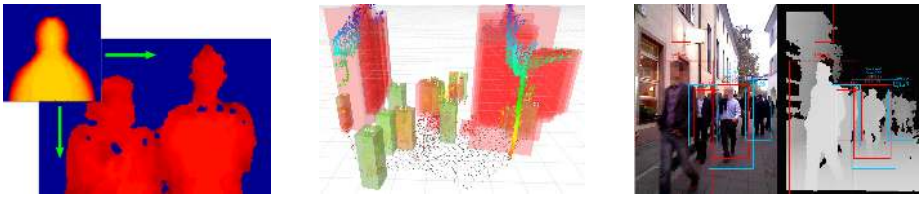


**Fig. 5.** Different RGB-D detectors which we integrated into our framework. *Left:* Upper-body detector from [12] which slides a normalized depth template over the depth image. *Middle:* RGB-D detector from PCL which first extracts regions of interest, visualized here by boxes, from the point cloud [21]. *Right:* Combo-HOD (histogram of oriented depths) detector [28] (closed-source).

```
roslaunch srl_laser_detectors
    adaboost_detector_with_segmentation.launch
```

and expects sensor_msgs/LaserScan messages on the /laser topic and publishes the resulting detections at /detected_persons. The names of these topics can be reconfigured via parameters passed to the launch file. A short example dataset for testing, and instructions on how to play back this recorded data, can be found online in our `README` file.

**Leg Detector** We also provide a wrapper to make the existing `leg_detector` ROS package compatible with our message definitions. This package needs to be downloaded and built separately[3]. The underlying algorithm uses a subset of the 2D features also included in our implementation of [2], but first tracks both legs separately and in our experience works best if the laser sensor is mounted very close to the ground, below 0.5m height.

### 3.3 Person Detection in RGB-D

**Upper-Body Detector** We modified the publicly available close-range upper-body detector by [12] (Fig. 5, left) to also output DetectedPersons messages. It operates purely on depth images and is launched via:

```
roslaunch rwth_upper_body_detector upper_body_detector.launch
```

The input and output topics can be configured via the `camera_namespace` and `detected_persons` parameters. It is assumed that a ground plane estimate is published on the topic specified by the `ground_plane` parameter, which can e. g. be achieved using the `ground_plane_fixed.launch` file provided in the `rwth_ground_plane` package.

[3] `http://wiki.ros.org/leg_detector`

**Further RGB-D Detectors** In the `pcl_people_detector` package, we also integrated the RGB-D person detector by [21] from the Point Cloud Library (PCL), which extracts regions of interest from a depth-based height map and then applies a linear HOG classifier. We extended the code to output markers for visualization of ROIs (Fig. 5, middle) and output DetectedPersons messages. Configurable parameters are documented in a launch file that can be started via:

```
roslaunch pcl_people_detector start.launch
```

Likewise, as a proof of concept, a closed-source implementation of Combo-HOD (histogram of oriented depths) [28] was integrated (Fig. 5, right).

### 3.4 Person Detection in Monocular Vision

**groundHOG** We also integrated the GPU-accelerated medium- to far-range groundHOG detector from [12] into our framework. It is configured in a similar way as the upper-body detector described above, and also requires a groundplane estimate to narrow down the search space for possible person detections. On a computer with a working CUDA installation, it can be launched via:

```
roslaunch rwth_ground_hog ground_hog_with_GP.launch
```

As can be seen from these examples, it is very easy to integrate new detectors into our people tracking framework. This makes it possible to easily swap out detectors in experiments, as well as to draw upon a common visualization toolkit for displaying the detections along with raw sensor data in 3D space (Sec. 7).

## 4 People Tracking

The output of a person detector just represents a single snapshot in time and may be subject to false alarms and missed detections. To gain a more long-term understanding of the scene, to filter out spurious misdetections and to extract trajectory information and velocity profiles, the detected persons need to be associated over time, a process called *people tracking*. The goal of a people tracking system is to maintain a persistent identifier for the same person over time, as long as the person remains visible in the scene and while bridging short moments of occlusion.

Different multi-target data association and tracking algorithms have been studied in the context of person tracking. Simpler algorithms such as the Nearest-Neighbor Standard Filter (NNSF), Global Nearest Neighbor (GNN) [4, 22] or Nearest-Neighbor Joint Probabilistic Data Association (NNJPDA) [3] make hard data association decisions, whereas other variants including the Joint Probabilistic Data Association Filter (JPDAF) [27] use soft assignments. All of these methods are single-hypothesis algorithms that at the end of each tracking cycle, only keep the most likely data association hypothesis in memory. In contrast, multi-hypothesis tracking approaches (MHT) [25, 8, 1, 12] use a hypothesis tree

which allows to correct wrong initial decisions at a later point in time. This, however, comes at the price of higher computational complexity and complex implementation.

### 4.1 ROS Message Definitions

As in the detection case, we tried to keep the message definitions for people tracking as generic as possible such that they can be re-used across a large variety of different people tracking algorithms. In the following sections, we illustrate concrete implementations that we have experimented with during the project, all of which yield the following output.

A **TrackedPerson**, according to our definition, possesses the following attributes:

- *track_id* [uint64]: An identifier of the tracked person, unique over time.
- *is_matched* [bool]: False if no matching detection was found close to the track's predicted position. If false for too long, the track usually gets deleted.
- *is_occluded* [bool]: True if the person is physically occluded by another person or obstacle. False if the tracking algorithm cannot determine this.
- *detection_id* [uint64]: If is_matched is true, the unique ID of the detection associated with the track in the current tracking cycle. Otherwise undefined.
- *pose* [geometry_msgs/PoseWithCovariance]: The position and orientation of the tracked person in metric 3D space, along with its uncertainty (expressed as a $6 \times 6$ covariance matrix). The pose is relative to the coordinate frame specified in the TrackedPersons message (see below).
- *twist* [geometry_msgs/TwistWithCovariance]: The linear and possibly angular velocity of the track, along with their uncertainties.
- *age* [duration]: The time span for which this track has existed in the system.

The **TrackedPersons** message aggregates all TrackedPerson instances that are currently being tracked:

- *header* [std_msgs/Header]: The coordinate frame is usually a locally fixed frame that does *not* move with the robot, such as "odom". The timestamp is copied from the incoming DetectedPersons message; this is important to synchronize DetectedPersons and TrackedPersons messages to be able to look up details about a DetectedPerson identified by its detection_id.
- *tracks* [array of TrackedPerson]: All persons that are being tracked in the current tracking cycle, including occluded and unmatched tracks.

### 4.2 People Tracking Algorithms Used in Our Experiments

During the SPENCER project, we have conducted experiments with different person tracking algorithms of varying complexity. Next to a computationally cheap nearest-neighbor algorithm which we will describe in the following section, we converted an existing hypothesis-oriented multi-hypothesis tracker from our past work [1, 17] into a ROS package[4] to exploit the visualization capabilities and interchangeability of detectors of our framework. Further experiments were conducted using a vision-based track-oriented multi-hypothesis tracker[5] [12].

A detailed study of which person tracking algorithm to choose under which circumstances is subject of our ongoing research. We believe that in general, the choice of models (e. g. for motion prediction, occlusion handling, inclusion of visual appearance) has a greater influence on overall tracking performance than the tracking algorithm itself. Here, various methods have been proposed in the past, e. g. the use of a motion model influenced by social forces [19], adapting occlusion probabilities and motion prediction within groups [17], improved occlusion models [16], or an online-boosted RGB-D classifier that learns to tell apart tracks from background or other tracked persons [18, 22].

### 4.3 Example: Nearest-Neighbor Tracker

As an illustrative example, we show how a person tracking system can be implemented using a very simple nearest-neighbor standard filter, for which the full code is provided online. Combined with the extensions discussed in the following section, the system can in most situations already track persons more robustly than other publicly available ROS-based people tracking implementations that we are aware of (e. g. in `wg-perception/people`).

The nearest-neighbor data association method greedily associates a detection with the closest (existing) track, if that track's predicted position is closer than a certain gating threshold. While not as robust as more advanced methods like multi-hypothesis tracking (MHT), it is very simple to implement and real time-capable even with a high number of tracks (up to 40–80). In our experience, such single-hypothesis methods deliver satisfactory results if there are not too many closely spaced persons in the scene and the detector has a relatively low false alarm rate and high recall, which can also be achieved by fusing information from multiple detectors in different modalities (Sec. 6).

A new tracking cycle starts once a new, and potentially empty, set of DetectedPersons is received by the tracker from the detector. The resulting actions that are performed are summarized in the following ROS callback:

*People Tracking Workflow*

```
1 void newDetectedPersonsCallback(DetectedPersons& detections)
2 {
```

[4] Not (yet) publicly available due to open questions on licensing.
[5] ROS version to be made available in the `rwth_pedestrian_tracking` package.

```
3   transformIntoFixedFrame(detections);
4   predictTrackStates();
5   predictMeasurements();
6   Pairings pairings = performDataAssociation(m_tracks, detections);
7   updateTrackStates(pairings);
8   initNewTracks(detections, pairings);
9   deleteObsoleteTracks();
10 }
```

First, in line 3, all detections are transformed into the locally fixed "odom" coordinate frame, by looking up the robot's current position in the world as reported by odometry using the `tf` library. The states of all existing tracked persons – internally represented as a 4D vector $(x, y, \dot{x}, \dot{y})$ of position and velocity – are predicted into the future (line 4), e. g. using a Kalman Filter, and converted into measurement predictions by dropping their velocity components (line 5). Then, an association matrix is constructed which contains the distance between predicted and real measurement position for each possible pairing of detections and tracks. This usually involves a gating step in which incompatible pairings are discarded beforehand. The nearest-neighbor standard filter then starts searching for the pairing with minimum distance until all tracks and/or detections have been associated. In line 7, the states of all matched tracks are updated with the position of the associated detection, or in the case of an unmatched track, the state prediction is used as the new state. Finally, new tracks are initialized from detections that could not be associated with any track (line 8). Tracks that have not been associated with a detection for too long (i. e. occluded) are deleted.

### 4.4 Improving Robustness of Tracking

For more robust tracking of maneuvering persons, we extended this tracking method with an interacting multiple models (IMM) approach that dynamically mixes four different motion models depending on the evolution of a tracked person's trajectory and the prediction error: 1) A constant velocity (CV) model with low process noise, 2) a CV model with high process noise, 3) a Brownian motion model, 4) a coordinated-turn model. In lab experiments, this combination helped to reduce the number of track losses caused by abruptly stopping or turning persons, without us having to manually tune process noise levels, especially in smaller or very crowded environments where motions are less linear.

Our implementation also optionally subscribes to a second DetectedPersons ROS topic, on which detections of a high-recall, low-confidence detector (e. g. a primitive laser blob detector) can be published. After regular data association, we perform another round of data association on these detections only with tracks that have not yet been associated with a detection (from the high-precision detector). This simple modification can improve tracking performance significantly if the main detector has a low sensitivity.

Finally, an implementation of a track initialization logic as described in [6] helps to prevent false track initializations in case of spurious misdetections.

**Launching the People Tracker**

The nearest-neighbor tracker implementation comes with a launch file configured with parameters that yielded good results in our test cases. We recommend the reader to copy this launch file and use it as a starting point for own experiments:

```
roslaunch srl_nearest_neighbor_tracker nnt.launch
```

### 4.5 Tracking Metrics

For performance evaluations, we wrapped two publicly available Python implementations of the CLEAR MOT [5] and OSPA [26] multi-target tracking metrics into ROS nodes that follow our message conventions. These metrics are helpful for tuning parameters of the tracking system, as well as evaluating completely new tracking algorithms. They assume that annotated groundtruth tracks are given in the form of time-synchronized TrackedPersons messages, and can be found in the `spencer_tracking_metrics` package.

## 5 Group Tracking

For socially compliant navigation, which is the main research objective in the SPENCER project, tracking just individual persons is not sufficient. If the task of the robot is to guide a group of persons through a shopping mall or an airport, or to avoid crossing through groups of people, knowledge of groups in the surroundings is important. In the latter case, we regard as a group a closely spaced formation of standing or walking persons. In the former case, a more long-term definition of *group* might be necessary, in which a person can, for instance, briefly leave the formation to avoid opposing traffic before re-joining the group.

Group tracking has been studied before in the computer vision community on photos and movies [29, 9, 7]. Methods have also been developed for group tracking using overhead cameras in the context of video surveillance [30, 23, 14, 24]. For people tracking on mobile robots in a first-person perspective, a multi-model multi-hypothesis tracker for groups has been proposed by Lau et al. [13], which was extended by Luber and Arras [17] and Linder and Arras [15].

In the following, we outline a few basic principles for online group detection and tracking, and describe sample code that we provide online.

### 5.1 Social Relation Estimation

Before detecting groups, we first estimate the pairwise social relations of all tracked persons by feeding the output of the people tracker into a social relation estimation module. This module, based upon the work described in [15], builds a social network graph as shown in Fig. 6 (left), where the edge weights encode the likelihood of a positive social relation between a pair of persons. To estimate these likelihoods, we rely on *coherent motion indicators*, which are

motion-related features that were found to indicate group affiliation between people in large-scale crowd behavior analysis experiments [20]. They consist of relative spatial distance, difference in velocity and difference in orientation of a given pair of tracks. Using a probabilistic SVM classifier trained on a large dataset annotated with group affiliations, these features are mapped to a social relation probability that indicates the strength of the relation. This method allows for an easy integration of additional social cues (such as persons' eye gaze, body pose) in the future. An implementation and a trained SVM model are provided in the `spencer_social_relations` package. Figure 7 shows the corresponding SocialRelation and SocialRelations ROS messages that constitute the resulting social network graph.

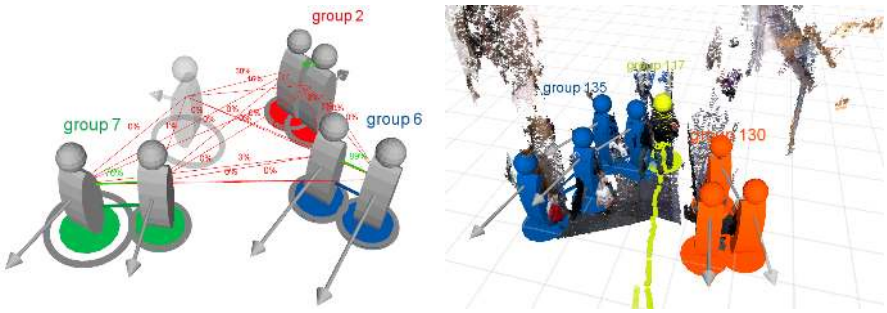



**Fig. 6.** *Left:* Example of a social network graph, where the edge weights encode the likelihood of a positive social relation between persons. Colored circles below each person indicate resulting group affiliations and grey covariance ellipses visualize position uncertainty. *Right:* Three different groups that are being tracked in RGB-D.

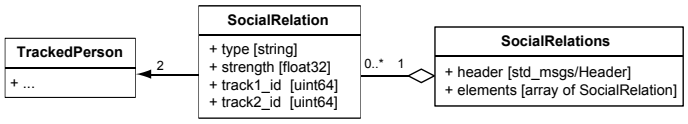



**Fig. 7.** Message definitions for the social network graph. The strength of a social relation is specified as a real number between 0.0 and 1.0. The type string can be used to distinguish between different types of relations, e. g. spatial, family or romantic.

### 5.2 Group Detection and Tracking

To detect groups, we perform graph-cutting in the social network graph by removing all edges below a fixed edge weight threshold. The remaining connected components of the graph then become the resulting groups.

A simple group tracking algorithm can now be implemented by treating groups as separate entities, and tracking their centroids. This data association might fail if the group splits apart very suddenly, causing a large shift in the resulting new centroids. Alternatively, it is possible to track just the composition of a group, in terms of its individual group members, and then keep a memory of previously seen group configurations. This works as long as at least one person in the group is always visible. A simple Python-based implementation of this approach can be found in the `spencer_group_tracking` package. The output is in the form of TrackedGroups messages, as outlined in Fig. 8.

In ongoing research shown in Fig. 6 (right), we use a more complex approach which interleaves a group-level data association layer with regular person-level data association in a multi-model multi-hypothesis tracker [17, 15]. By explicitly modelling group formation in terms of merge and split events, groups can be tracked more robustly. Internally, a prototype of this system has already been integrated with our visualization and detection framework and yielded good experimental results. While currently not publicly available, this system uses exactly the same message interfaces as the provided Python code.

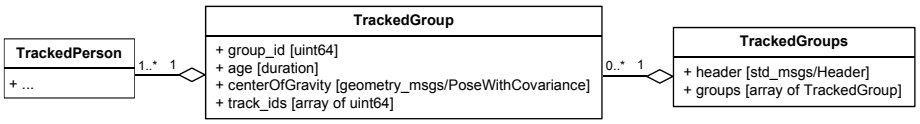



**Fig. 8.** Message definition for a single tracked group, a collection of all tracked groups in one cycle, and their relation towards tracked persons.

## 6 Multi-Modal Tracking

In this section, we describe how we can fuse the output of multiple detectors, potentially operating in different sensor modalities, to make people tracking more robust and extend the robot's field of view. A broad overview of different strategies for fusing multi-modal detections for general tracking purposes is given in [4]. Here, we mainly focus on fusion at the detection level, as track-to-track fusion involves associating two or more trajectories with each other, which is computationally more complex and not straightforward. Our framework allows for two different fusion strategies at the detection level, namely detection-to-detection and detection-to-track fusion.

### 6.1 ROS Message Definitions

To keep a memory of which detections have been fused together, we define a CompositeDetectedPerson message (Fig. 9). Retaining this information is essential for components at later stages in the perception pipeline, like a human

attribute classifier that takes as an input the regions of interest found by a vision-based person detector. If these ROIs likewise bear a corresponding detection ID, they can later on be associated with the correct tracked person such that the extracted information can be smoothed over time.

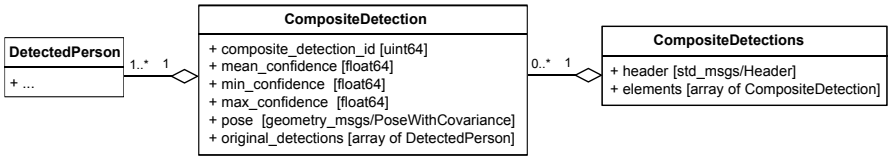



**Fig. 9.** Message definitions for composite detections. Retaining the information which original detections have been combined into a composite detection or a track is important for perception components at later stages in the pipeline.

In order to provide all of our fusion components with a streamlined, homogenous interface, we initially convert all DetectedPersons messages into CompositeDetectedPersons via a converter node, even if there is only a single DetectedPerson involved. This allows for easy chaining of the components.

### 6.2 Strategies for Fusion at the Detection Level

**Detection-to-Detection Fusion** Fusing detections to detections has got the advantage that the resulting composite detections can be fed into any existing (unimodal) tracking algorithm, without requiring special provisions to cope with information from multiple detectors. This approach can also be helpful if the tracking algorithm itself is computationally very complex and scales badly with the number of detections. For detection-to-detection fusion, we have implemented a series of nodelets[6] which can be used to flexibly compose a fusion pipeline by means of roslaunch XML files:

- *Converter nodelets* for converting DetectedPersons messages into CompositeDetectedPersons messages, and vice versa.

- *Aggregator nodelets* which simply concatenate two sets of CompositeDetectedPersons. Useful for combining detections from sensors with non-overlapping fields of view, in which case no data association is required.

- *Fusion nodelets* that perform data association between two sets of CompositeDetectedPersons, e. g. using a nearest-neighbor data association technique.

[6] A powerful mechanism to dynamically combine multiple algorithms into a single process with zero-copy communication cost. See `http://wiki.ros.org/nodelet`.

**Detection-to-Track Fusion** Although presently not realized, our framework also allows for detection-to-track fusion. As the tracker's knowledge of a track's previous trajectory can help with the association of the incoming detections, this approach may provide better results in case of very crowded environments. In this case, the fusion stage needs to be implemented within the tracker itself. It then becomes the tracker's responsibility to publish a corresponding CompositeDetectedPersons message to let other perception components know which original detections have been fused.

### 6.3 Post-Processing Filters

Often, higher-level reasoning components are not interested in all tracks that are output by the people tracking system. Therefore, we provide a series of post-processing filters for TrackedPersons messages such that the output, for instance, only includes visually confirmed tracks, non-static persons, or the $n$ persons closest to the robot (useful for human-robot interaction). These are implemented in `spencer_tracking_utils` and can be set up in a chain if needed.

### 6.4 Multi-Modal People Tracking Setup on the Robot

On the robot platform, we run the front laser detector, the front upper-body RGB-D detector, the front HOG detector, and the fusion-related nodes on a single laptop. Likewise, the detectors for the rear-oriented sensors are executed on the second laptop, as well as the people tracking system itself. The lower RGB-D sensor on each side is tilted downwards and currently only used for close-range collision avoidance. Since all components, including the laser-based segmentation as a pre-processing step, are implemented as individual ROS nodes, they are separate system processes and therefore automatically make full use of modern multi-core CPUs.

Figure 10 shows a configuration which we used during experiments with our robot platform. Due to the flexible configuration in roslaunch XML files, different setups can easily be tested without modifying any source code. Of course, if the used people tracker implementation directly supports multiple detection sources, most of the detection-to-detection fusion stage can be left out (up to a possible aggregation of front and rear detections). We also ran experiments with an integrated person- and group tracker [15] that combines all of the steps in the tracking stage in order to feed back information from group-level tracking into person tracking, while still publishing TrackedPersons, SocialRelations and TrackedGroups messages. In all these cases, the remaining components including detectors, visualization, post-processing filters and evaluation metrics can still be used as-is.

### 6.5 Exemplary Launch File

As an inspiration for own experiments, we provide a launch file for the multi-modal people tracking setup that we used on our robot:

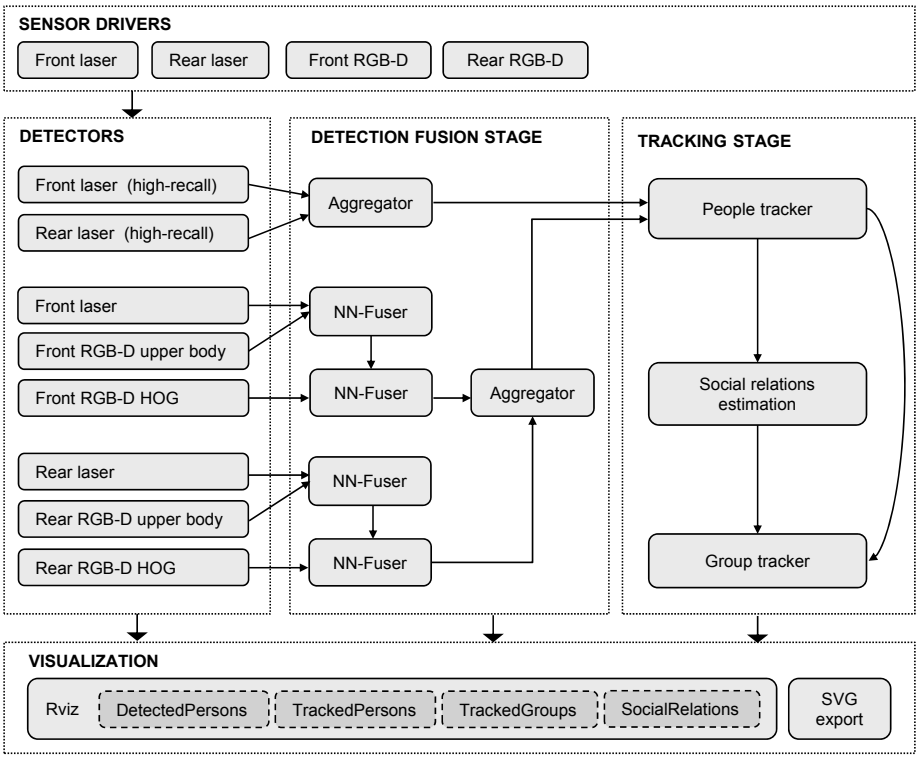



**Fig. 10.** A possible multi-modal people and group detection and tracking architecture that can be configured using our framework. Arrows represent ROS message flows. This setup was used during experiments on the SPENCER robot platform. Rounded rectangles represent reusable components that can be run as separate ROS nodes, and therefore easily be distributed across different computers. Connections between nodes are configured entirely in roslaunch XML without having to modify source code.

```
roslaunch spencer_people_tracking_launch tracking_on_robot.launch
```

Alternatively, we provide a launch file that just uses a single RGB-D sensor:

```
roslaunch spencer_people_tracking_launch
    tracking_single_rgbd_sensor.launch height_above_ground:=1.6
```

## 7 Visualizing the Outputs of the Perception Pipeline

Powerful visualization tools can be of great help when analyzing the performance of a complex perception pipeline. The standard visualization tool for sensor data which is shipped with ROS is RViz, which uses the feature-rich OGRE graphics engine as its visualization backend. The core idea behind RViz is that it provides

different visualization plugins *(displays)* for different types of ROS messages, e. g. sensor_msgs/LaserScan or nav_msgs/Odometry.

In general, RViz provides two ways of implementing custom visualizations:

- *Markers and marker arrays*, using existing displays provided by RViz and message types from the `visualization_msgs` package. They allow to easily display primitives such as lines, triangles, texts, cubes or static 3D meshes. The pose, dimensions and color of the shape are specified within the message itself, published by a ROS node in any supported programming language.

- *RViz plugins*, written in C++, can benefit from the entire set of capabilities offered by the OGRE graphics engine. As each display comes with a property panel based on the UI framework Qt, they can easily be customized within the RViz user interface. RViz automatically takes care of persisting any settings when the user saves the configuration, which allows to load entire visualization setups via a single mouse-click.

In our experience, markers allow to quickly implement component-specific visualizations without requiring much developer effort. On the other hand, RViz plugins are great for more complex visualizations that are reused often. They offer a better end-user experience (due to the ability to change settings on-the-fly inside the RViz GUI) at the cost of larger implementation effort.

### 7.1 Custom RViz Visualization Plugins

During the course of the SPENCER project, we were often in need of visualizing the outputs of our people tracking system and changing visualization settings on-the-fly, for instance during live robot experiments and for demonstration videos and publications. Therefore, in the `spencer_tracking_rviz_plugin` package, we developed a series of custom RViz plugins for displaying detected persons, tracked persons, social relations, tracked groups and human attributes[7]. They are automatically discovered by RViz and include features such as:

- Different visual styles: 3D bounding box, cylinder, animated human mesh
- Coloring: 6 different color palettes
- Display of velocity arrows
- Visualization of the 99% covariance ellipse for position uncertainty
- Display of track IDs, status (matched, occluded), associated detection IDs
- Configurable reduction of opacity when a track is occluded
- Track history (trajectory) display as dots or lines
- Configurable font sizes and line widths

In the Results section in Fig. 14, we show a number of different visualizations generated using our RViz plugins. For the first row of that figure, we additionally

[7] Such as gender and age group, not discussed here in detail.

used the existing RViz *Camera* display to project 3D shapes into a camera image. Examples of the social relations and tracked groups display are shown in Fig. 6, left and right. As opposed to other RViz displays, these displays subscribe to two topics at the same time (e. g. social relations and tracked persons).

### 7.2 URDF Model for Robot Visualization

To visualize the robot in RViz, we built a URDF (Unified Robot Description Format) file for the SPENCER robot, which defines the different joints of the platform, including fixed joints at the base of the robot and the sensor mounting points. The `robot_state_publisher` and `joint_state_publisher` packages use the URDF to publish a hierarchy of transforms (TF tree) for transforming between local sensor and robot coordinate frames as shown in Fig. 11 (right).

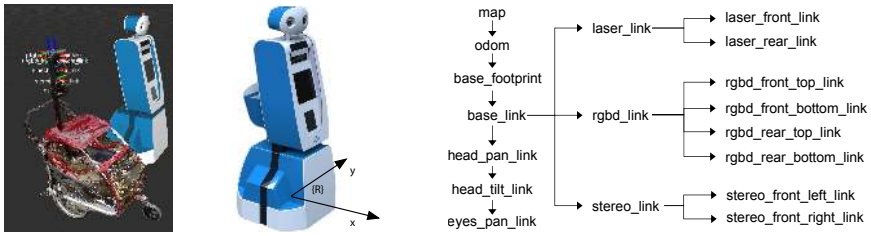



**Fig. 11.** *Left:* Robot models (based upon URDF descriptions) of the SPENCER robot platform and the mobile data capture platform. *Middle:* Robot coordinate frame. *Right:* Simplified version of the TF hierarchy used on the SPENCER robot.

To create the visual robot model shown in Fig. 11 (left), we used a CAD model provided by the robot manufacturer and first removed any unnecessary components inside the robot's shell that are not visible from the outside using FreeCAD. We then exported a single Collada (DAE) file for each movable part and simplified the resulting mesh to reduce its polygon count. This can be done using software such as MeshLab or 3DS Max. We then manually aligned the visual parts of the model against the TF links specified in the URDF, by cross-checking with RViz and the CAD file.

For the mobile data capture platform, we used a point cloud registration software and an RGB-D sensor that we rotated around the platform at different azimuths and elevations to generate a visual representation. We built a transform hierarchy similar to that of the actual robot, to ensure that the same perception algorithms can be run on recorded and live data without modifications.

### 7.3 ROS-based SVG Exporters

For the visualization of detected and tracked persons and groups, we additionally provide a Python script in the `srl_tracking_exporter` package for exporting

scalable vector graphics (SVGs). This is useful to analyze person trajectories from a 2D top-down view of the scene, which can be animated to display the evolution of tracks over time. In contrast to videos recorded from Rviz, they support free zooming to permit the analysis of smaller details (e. g. the cause of track identifier switches, by looking at the positions of individual detections). The resulting SVGs (Fig. 12) are best viewed in a web browser such as Chrome.

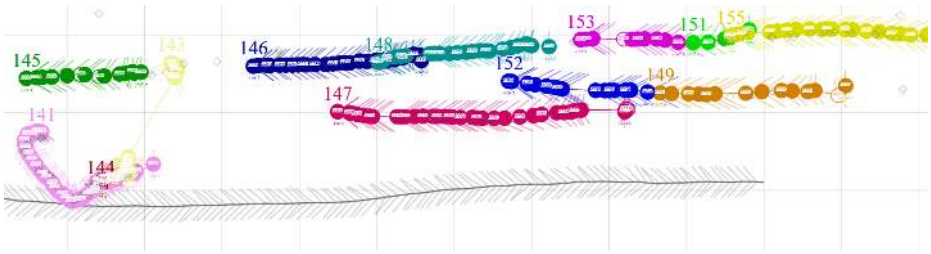



**Fig. 12.** Scalable and optionally animated vector graphic showing a 2D top-down view of a scene with several person tracks and their track IDs. Detections are shown as small diamonds. Arrowheads symbolize velocity, with the bottom grey track showing robot odometry. The user can zoom into the SVG to read detection IDs and timestamps.

## 8 Integration with 3rd-Party Simulation Tools

Often, experimenting with real, recorded data can be cumbersome due to the need for manual groundtruth annotations and because it can be difficult to capture or provoke specific situations without scripting the behavior of the agents in the scene. For example, testing socially compliant motion planning in combination with a real people tracking system (including occlusions) proves to be difficult from pre-recorded datasets as the sensor position is decided at the time of recording. Testing on the real robot, however, can bring up its own challenges, where lack of open space, limited physical access to the robot, hardware issues or discharged batteries are just a few obstacles to overcome.

### 8.1 Integration with the Robot Simulator Gazebo

For this reason, we integrated our framework with the robot simulator Gazebo to generate simulated sensor data of moving persons from a moving robot in static environments (Fig. 13, left). We simulate 2D laser scan data with a noise model matching that of the real sensor on the basis of datasheet specifications.

As simulating the environment becomes computationally very complex with more than a dozen persons in the scene, we disable the Gazebo physics engine and instead manually update the agents' and robot's position at 25 Hz using the Gazebo ROS API. Person shapes are approximated by static 3D meshes that

are processed by a GPU-accelerated raytracing algorithm in Gazebo to simulate laser scans. The resulting sensor_msgs/LaserScan is published on ROS and fed to our laser-based person detector, and RViz for visualization (Fig. 13, right).

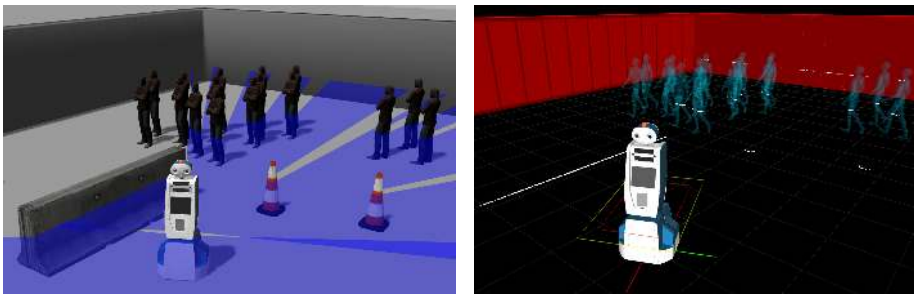


**Fig. 13.** *Left:* Positions of simulated pedestrians are constantly sent to the Gazebo simulator via ROS, which then generates simulated laser scans and again publishes them via ROS. These simulated laser scans can then be fed into the people tracking pipeline for synthetic experiments. *Right:* Moving pedestrians in a ROS adaptation of PedSim, the pedestrian simulator, visualized in RViz. The pedestrian behavior is modelled using a social force model from behavioral sciences.

### 8.2 Integration with the Pedestrian Simulator PedSim

Simulating realistic pedestrian motion behaviors is a research topic of its own. Here, we use the publicly available PedSim library[8], which simulates pedestrian crowd behaviors using a social force model from behavioral sciences [11]. We wrapped the library into a ROS node and added an RViz-based visualization of persons and obstacles using visualization markers (visible in Fig. 13, right). The positions of the simulated pedestrians are published as TrackedPersons messages, which can serve as a groundtruth to evaluate tracking performance (see Sec. 4.5). A separate converter node then sends these positions to Gazebo, where they are used to position 3D meshes as described in the previous section.

## 9 Results

In this section, we show qualitative results of our tracking system, discuss runtime performance and summarize important lessons that we learned during the project. As research in SPENCER is still going on at the time of this writing, we are currently preparing a detailed quantitative study of our tracking system, including a comparison of different multi-modal tracking approaches, for a publication at the end of the project.

[8] `http://pedsim.silmaril.org/`

### 9.1 Qualitative Results

In Fig. 14, we show illustrative results of our tracking system running on data recorded with our mobile data capture platform (Fig. 2) during a crowded situation at an airport, shortly after passengers have disembarked from an airplane. At the detection stage, we observe that the different detectors (2D laser, RGB-D upper-body, groundHOG in RGB) complement each other well and significantly increase the field of view around the robot in which persons can be tracked. In group guidance scenarios, it can make sense to filter out tracks in a post-processing step which have not been visually confirmed (shown in different color in the third row of Fig. 14), as especially the laser-based detector can sometimes trigger false alarms when objects appear like humans in the laser scan.

Multiple videos which show our tracking system in action on both the real robot platform and pre-recorded datasets can be found on our YouTube channel[9].

### 9.2 Runtime Performance

Our system runs in real-time at 20–25 Hz using the configuration described in Sec. 6.4 on two Intel Core i7-4700MQ gaming laptops with 8 GB RAM and nVidia GeForce GTX 765M graphics[10]. These laptops are dedicated to perception tasks on the robot. We avoid expensive streaming of RGB-D images over network by executing the RGB-D detectors directly on the laptop where the corresponding sensor is connected via USB. The computationally most expensive components are the person detectors, each requiring about one CPU core, with the people tracker itself requiring only 10–15% of a single CPU core when 20–30 tracks are simultaneously being tracked.

### 9.3 Lessons Learned

Finally, we want to note down a few points that we learned about ROS and our hardware during the course of our project:

**Parallelization:** A complex perception pipeline such as the one developed in SPENCER requires a high amount of computational power. We learned that ROS can be a great help here due to its node-based modular structure, which easily allows to split up complex processing tasks into separate processes that can be run on different CPU cores or even different computers, without requiring extensive reconfiguration.

**Collaboration:** We think it is important to devise mock components early on in the project to enable other collaborators to test their dependent components. Due to the multi-language support in ROS, these can be simple Python scripts that e. g. publish fake detections.

[9] `http://www.youtube.com/spencereuproject`

[10] The GPU is only used by the groundHOG person detector. For a single RGB-D sensor and 2D laser scanner, a single laptop is sufficient.

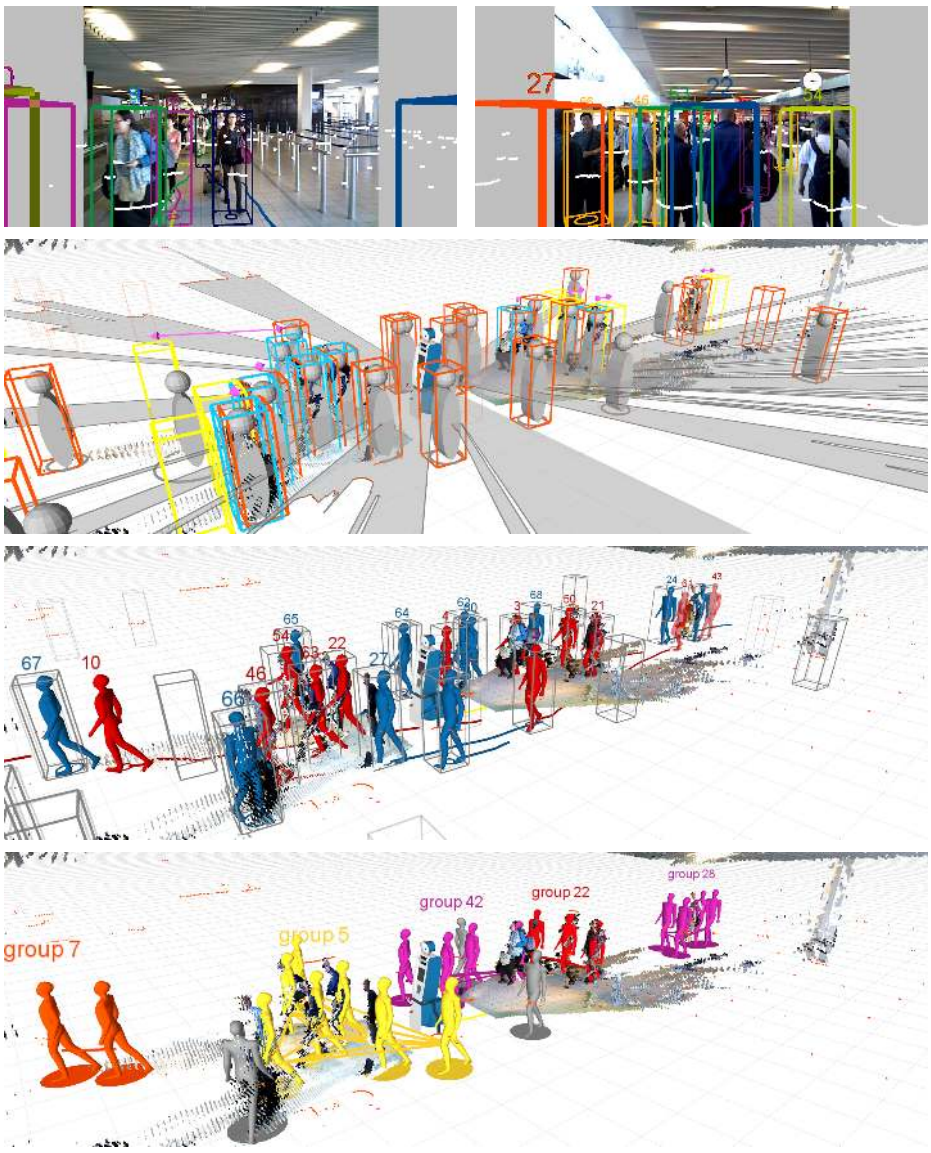



**Fig. 14.** Multi-modal people detection and tracking from a mobile robot in a crowded airport environment. *First row:* Color images of the upper rear and front RGB-D sensors, with projected laser points and bounding boxes of tracked persons. *Second row:* Detections of 2D laser (orange boxes), upper-body detector (cyan) and groundHOG (yellow) on top of front and rear RGB-D and 2D laser point clouds. Fused detections are shown as grey cylinders. The area visible to the laser scanners is shaded in grey. *Third row:* Resulting person tracks. As opposed to blue persons which are only tracked in laser, red tracks have been confirmed a number of times by one of the visual detectors. *Last row:* Tracked groups that were detected via coherent motion indicator features.

**Transform issues:** We often had to deal with transform-related issues, which is natural due to the high number of sensor coordinate frames involved. Here, we found the tools `tf_monitor`, `tf_echo`, `view_frames` of the `tf` package as well as the TF display in RViz to be of great help. Especially the first-mentioned tool allows to identify latency issues that can arise when the clocks of the computers on the robot are not properly synchronized, which will cause transform lookups to fail. The `roswtf` tool lists unconnected ROS topic subscriptions, which helps when dealing with over 300 concurrently active topics.

**Hardware:** In terms of hardware, instead of using gaming laptops onboard a robot, in the future we would rather use traditional computers with extra GPU and USB PCIe cards to circumvent the USB bus sharing issues which we repeatedly encountered with the first-generation RGB-D sensors, since most laptops nowadays only expose a single USB bus. It might also be worthwhile to consider using many small-form-factor PCs with integrated GPUs, as opposed to few high-end computers. Due to the modular structure of ROS, individual nodes can easily be spread across different computers.

## 10 Conclusion

In this chapter, we presented a multi-modal, ROS-based people detection and tracking framework developed for and tested on a large mobile service robot. We believe that due to its modular structure with clearly defined interfaces between components, our framework can easily be re-used on other robot platforms equipped with similar sensor configurations. Also, as we found out in our own research, standardized message interfaces allow to easily replace individual components of the pipeline, such as the core tracking algorithm, by a different implementation for comparative purposes. When doing so, the user can leverage our wide infrastructure of existing visualization, evaluation and simulation tools to quickly set up a usable tracking system. To our knowledge, the presented people tracking framework is the functionally most complete one that is currently available as open-source ROS software.

In future work, we want to extend our tracking framework with person re-identification capabilities, the lack of which is currently the most apparent bottleneck when dealing with lengthy occlusions of tracked persons.

**Acknowledgements** This work has been supported by the EC under contract number FP7-ICT-600877 (SPENCER). The authors would like to thank Fabian Girrbach for implementing several important extensions to the tracking system; Stefan Breuers from the Computer Vision Group at RWTH Aachen for MHT experiments and integrating the RGB-D and monocular vision detectors into the framework; and Christian Dondrup from the Lincoln Centre for Autonomous Systems Research for the original ROS integration of these components.

## References


1. Arras, K.O., Grzonka, S., Luber, M., Burgard, W.: Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Pasadena, California, USA (2008)
2. Arras, K.O., Mozos, O.M., Burgard, W.: Using boosted features for the detection of people in 2D range data. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Rome, Italy (2007)
3. Bar-Shalom, Y., University of California, L.A.U.E.: Multitarget-Multisensor Tracking: Applications and Advances. No. 3 in Radar Library, Artech House (2000)
4. Bar-Shalom, Y., Willett, P., Willett, P., Tian, X.: Tracking and Data Fusion: A Handbook of Algorithms. YBS Publishing (2011)
5. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: The CLEAR MOT metrics. J. Image Video Process. 2008, 1:1–1:10 (2008)
6. Blackman, S., Popoli, R.: Design and Analysis of Modern Tracking Systems. Radar Library, Artech House (1999)
7. Choi, W., Savarese, S.: A unified framework for multi-target tracking and collective activity recognition. In: Proc. of the European Conf. on Comp. Vision (ECCV). pp. 215–230. Florence, Italy (2012)
8. Cox, I., Hingorani, S.: An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 18(2), 138–150 (1996)
9. Ding, L., Yilmaz, A.: Inferring social relations from visual concepts. In: IEEE Int. Conf. on Comp. Vis. (ICCV). Barcelona, Spain (2011)
10. Fod, A., Howard, A., Matari, M.J.: Laser-based people tracking. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Washington DC (2002)
11. Helbing, D., Molnar, P.: A social force model for pedestrian dynamics. Phys. Rev. E 51 (1995)
12. Jafari, O.H., Mitzel, D., Leibe, B.: Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Hong Kong, China (2014)
13. Lau, B., Arras, K.O., Burgard, W.: Multi-model hypothesis group tracking and group size estimation. Int. Journal of Social Robotics 2(1), 19–30 (2010)
14. Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B.: Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In: ICCV Workshop on Modeling, Simul. and Vis. Analysis of Large Crowds (2011)
15. Linder, T., Arras, K.O.: Multi-model hypothesis tracking of groups of people in RGB-D data. In: IEEE Int. Conf. on Information Fusion (FUSION'14). Salamanca, Spain (2014)
16. Luber, M., Tipaldi, G.D., Arras, K.O.: Better models for people tracking. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Shanghai (2011)
17. Luber, M., Arras, K.O.: Multi-hypothesis social grouping and tracking for mobile robots. In: Robotics: Science and Systems (RSS'13). Berlin, Germany (2013)
18. Luber, M., Spinello, L., Arras, K.O.: People tracking in RGB-D data with online-boosted target models. In: Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS). San Francisco, USA (2011)
19. Luber, M., Stork, J.A., Tipaldi, G.D., Arras, K.O.: People tracking with human motion predictions from social forces. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Anchorage, USA (2010)

20. Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., Theraulaz, G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. PLoS ONE 5(4) (2010)
21. Munaro, M., Basso, F., Menegatti, E.: Tracking people within groups with RGB-D data. In: Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS) (2012)
22. Munaro, M., Menegatti, E.: Fast RGB-D people tracking for service robots. Autonomous Robots 37(3), 227–242 (2014)
23. Pellegrini, S., Ess, A., van Gool, L.: Improving data association by joint modeling of pedestrian trajectories and groupings. In: Proc. of the European Conf. on Comp. Vision (ECCV). Heraklion, Greece (2010)
24. Qin, Z., Shelton, C.R.: Improving multi-target tracking via social grouping. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Providence, RI, USA (2012)
25. Reid, D.B.: An algorithm for tracking multiple targets. Trans. on Automatic Control 24(6) (1979)
26. Schuhmacher, D., Vo, B.T., Vo, B.N.: A consistent metric for performance evaluation of multi-object filters. Signal Processing, IEEE Transactions on 56(8), 3447–3457 (2008)
27. Schulz, D., Burgard, W., Fox, D., Cremers, A.B.: People tracking with mobile robots using sample-based joint probabilistic data association filters. The International Journal of Robotics Research 22(2), 99–116 (2003)
28. Spinello, L., Arras, K.O.: People detection in RGB-D data. In: Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS). San Francisco, USA (2011)
29. Wang, G., Gallagher, A., Luo, J., Forsyth, D.: Seeing people in social context: recognizing people and social relationships. In: Proc. of the European Conf. on Comp. Vision (ECCV). Heraklion, Greece (2010)
30. Yu, T., Lim, S.N., Patwardhan, K.A., Krahnstoever, N.: Monitoring, recognizing and discovering social networks. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Miami, FL, USA (2009)

## 11 Authors' Biographies

Timm Linder is a PhD candidate at the Social Robotics Laboratory of the University of Freiburg, Germany. His research in the SPENCER project focusses on people detection and tracking in 2D laser range and RGB-D data, as well as human attribute classification in RGB-D. He is an experienced C++ and Python developer and has been a ROS user for over two years. He contributed to the ROS visualization tool RViz and played a major role in the deployment and integration of the software stack on the SPENCER robot platform.

Kai O. Arras is an assistant professor at the Social Robotics Laboratory of the University of Freiburg, and coordinator of the EU FP7 project SPENCER. After his diploma degree in Electrical Engineering at ETH Zürich, Switzerland and a PhD at EPFL, he was a post-doctoral research associate at KTH Stockholm, Sweden and at the Autonomous Intelligent Systems Group of the University of Freiburg. He also spent time as a senior research scientist at Evolution Robotics, Inc. and later became a DFG Junior Research Group Leader at the University of Freiburg.