A Software Architecture for RGB-D People Tracking Based on ROS Framework for a Mobile Robot

Matteo Munaro, Filippo Basso, Stefano Michieletto,
Enrico Pagello, and Emanuele Menegatti

The authors are with the Intelligent Autonomous Systems Laboratory (IAS-Lab),
at the Department of Information Engineering of the University of Padova,
via Gradenigo 6B, 35131 - Padova, Italy.
{munaro,bassofil,michieletto,epv,emg}@dei.unipd.it

Abstract. This paper describes the software architecture of a distributed multi-people tracking algorithm for mobile platforms equipped with a RGB-D sensor. Our approach features an efficient point cloud depth-based clustering, an HOG-like classification to robustly initialize a person tracking and a person classifier with online learning to drive data association. We explain in details how ROS functionalities and tools play an important role in the possibility of the software to be real time, distributed and easy to configure and debug.

Tests are presented on a challenging real-world indoor environment and tracking results have been evaluated with the CLEAR MOT metrics. Our algorithm proved to correctly track 96% of people with very limited ID switches and few false positives, with an average frame rate above 20 fps. We also test and discuss its applicability to robot-people following tasks and we report experiments on a public RGB-D dataset proving that our software can be distributed in order to increase the framerate and decrease the data exchange when multiple sensors are used.

Keywords: People tracking, Robot Operating System, real-time, RGB-D data, mobile robots.

1 Introduction and Related Work

Autonomous service robots have to move and act in dynamic and populated environments, thus they must be endowed with fast and reliable perception skills. In particular, they must be able to distinguish people from the rest, predict their future positions and plan their motion in a human-aware fashion, according to their tasks.

People detection and tracking techniques have been widely studied in literature. Many works exist that use RGB cameras only ([8], [25], [5]) or 3D sensors only ([16], [23], [24], [6], [18]). However, when dealing with mobile robots, the need for robustness and real time capabilities usually led researchers to tackle these problems by combining appearance and depth information. In [3], both a

S. Lee, K.-J. Yoon, and J. Lee (Eds.): *Frontiers of Intelligent Auton. Syst.*, SCI 466, pp. 53–68.
DOI: 10.1007/978-3-642-35485-4_5

PTZ camera and a laser range finder are used in order to combine the observations coming from a face detector and a leg detector, while in [14] the authors propose a probabilistic aggregation scheme for fusing data coming from an omnidirectional camera, a laser range finder and a sonar system. Ess *et al.* [9],[10] describe a tracking-by-detection approach based on a multi-hypothesis framework for tracking multiple people in busy environments from data coming by a synchronized camera pair. The depth estimation provided by the stereo pair allowed them to reach good results in challenging scenarios, but their method relied on a people detection algorithm which took 30s to process each image. Stereo cameras continue to be widely used in the robotics community ([1],[21]), but the computations needed for creating the disparity map always impose limitations to the maximum frame rate achievable, thus leaving less room for further algorithms operating in series with the tracking one.

Fig. 1. Tracking output on some frames extracted from the test set

With the advent of reliable and affordable RGB-D sensors a rapid boosting of robots capabilities can be envisioned. For example, the Microsoft Kinect sensor allows to natively capture RGB and depth information at good resolution (640x480 pixels) and frame rate (30 frames per second). Even though the depth estimation becomes very poor over eight meters of distance and this technology cannot be used outdoors because the sunlight can change the infrared pattern projected by the sensor, it constitutes a very rich source of information that can be simply used on a mobile platform.

In [22] a people detection algorithm for RGB-D data is proposed, which exploits a combination of HOG and HOD descriptors. However, the whole frame is densely scanned to search for people, thus requiring a GPU implementation for being executed in real time. Also [7] relies on a dense GPU-based object detection, while [15] investigates how the usage of the people detector can be reduced using a depth-based ROI tracking. However, the obtained ROIs are again densely scanned by a GPU-based people detector. In [13] a tracking algorithm on RGB-D data is proposed, which exploits the multi-cue people detection approach described in [22]. It adopts an on-line detector that learns individual target models and a multi-hypothesis decisional framework. No information is given about the computational time needed by the algorithm and results are reported for some sequences acquired from a static platform equipped with three RGB-D sensors.

In our previous works ([2], [17]) a fast and robust algorithm for people detection and tracking has been proposed, that has been designed for mobile robots and reaches real time performance while relying on CPU-only computation. We reached state of the art tracking results on challenging scenarios in terms of accuracy and speed with a ROS-based implementation.

1.1 Robot Operating System

The *Robot Operating System* [19] by Willow Garage is an open-source, meta-operating system that provides services usually expected from an operating system, including hardware abstraction, low-level device control, message-passing between processes, package management, and tools and libraries useful for typical robotics applications, such as navigation, motion planning, image and 3D data processing. Among the many drivers provided with ROS there is also the OpenNI driver, which allows to interface with three RGB-D devices (the Microsoft Kinect, the Asus Xtion and the PrimeSense sensor) for obtaining their raw data. The primary goal of ROS is to support code reuse in robotics research and development and, in this direction, is designed to be as thin as possible and its libraries are ROS-agnostic and have clean functional interfaces.

In this paper, we describe the software architecture of a multi-people tracking and following algorithm based on RGB-D data and specifically designed for mobile platforms. In particular, we focus on how ROS has been used for development, optimization and debugging, providing also quantitative evalutation of our implementation choices. Moreover, we report tests to prove such a system to work also for tracking people with multiple sensors in a distributed configuration.

The remainder of the paper is organized as follows: in Section 2 an overview of the people tracking algorithm we implemented is given, while in 3 we detail how ROS tools and functionalities have been exploited for this purpose. Section 4 reports experimental results obtained by our method when tracking and following people from a mobile robot and when used in a multi-sensor and distributed people tracking scenario. In Section 5 we report conclusions and future works.

2 People Tracking Algorithm

In this section, we give a brief overview of the two main conceptual blocks of our people tracking system, whose single algorithms are described in [17]. As reported in Fig. 2, the RGB-D data acquired by the sensor are processed by a detection module that filters the point cloud data, removes the ground and performs a 3D clustering of the remaining points. The clustering method is composed by an *Euclidean Clustering* algorithm followed by a sub-clustering pipeline specifically designed for detecting persons even when close to each other or to the background. Furthermore, we apply a HOG-based people detection algorithm to the RGB image of the resulting clusters in order to keep only those that are more likely to belong to the class of people. The resulting output is a set of detections that are then passed to the tracking module.

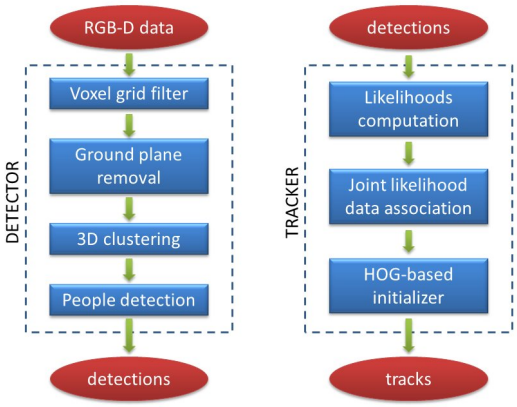


Fig. 2. Block diagram describing input/output data and the main operations performed by our detection and tracking modules

Our tracking algorithm performs detection-track association as a maximization of a joint likelihood composed by three terms: motion, color appearance and people detection confidence. A person classifier for every target is learned online for evaluating color appearance. It is based on the *Online Adaboost* algorithm [12] and on features extracted from the color histogram of the target. At the learning stage, new detections associated to a track are considered as positive examples for that track's classifier, while the other detections inside the image are chosen as negative examples. The HOG confidence is also used for robustly initializing new tracks when no association with existing tracks is found.

3 Software Architecture

This section details our algorithm by focusing on how ROS tools and functionalities have been used for the architectural design of our system, its implementation, debug and testing.

3.1 Modularity with Nodes, Topics and Message-Passing

ROS nodes are processes that perform computation and communicate with each other by asynchronous message passing. Messages are exchanged within a network by means of TCP or UDP and written/read to/from topics, that are named buses over which nodes exchange well defined types of messages.

ROS nodes structure highly incentives software modularity, so that most of the code can be reused also for other applications. In particular, sensors drivers are usually confined into single nodes, while data processing is implemented within other nodes. Following this policy we structured our code as composed

by five different nodes[1] interconnected as shown in Fig. 3. This figure is very similar to what can be obtained at runtime with the ROS tool `rxgraph`, but it has been re-drawn for better visualization with colors. Nodes are reported in green, topics in cyan and sensors in orange. This tool allows to easily understand the code structure and also to find bugs related to missed/wrong connections of nodes and topics. Here below, a description of the single nodes is reported:

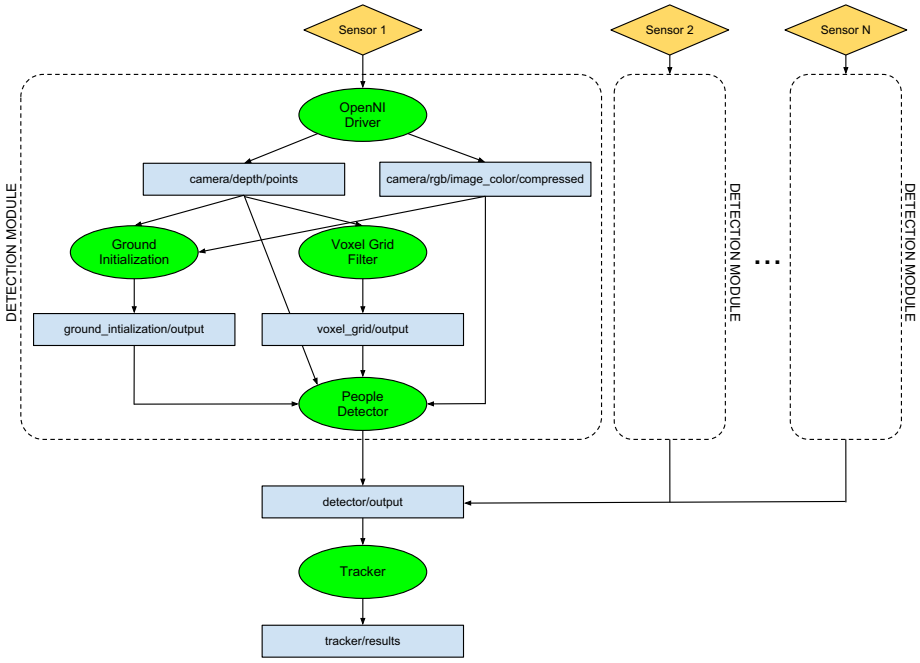


Fig. 3. Block scheme of nodes (green) and topics (cyan) interconnections

- `OpenNI_driver`: this node is provided by ROS repositories[2] and consists in the open source driver for the Microsoft Kinect, the ASUS Xtion and the PrimeSense sensor. It publishes their raw data both as colored/not colored point clouds and as RGB and depth images.

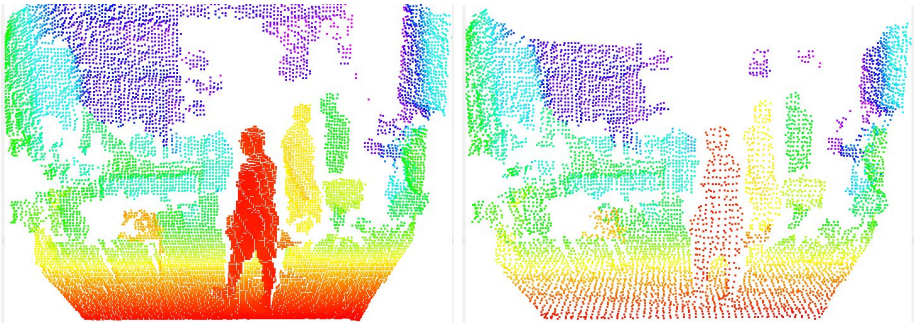
- `ground_initialization`: it is used for manual initialization of the ground plane at system startup. It takes as input sensor raw data and displays the RGB image, then a user is requested to click on three points referring to the ground. Given this input, it computes the ground plane parameters, that are then written to the corresponding topic. It is worth to notice that this node

[1] The second, fourth and fifth nodes have been implemented by the authors, while the first and the third nodes come as part of ROS libraries.

[2] `http://www.ros.org/wiki/openni_kinect`.

is executed only once, thus it is usefully structured as a separated node that terminates after the ground plane has been initialized.

- `voxel_grid_filter`: this node performs a smart downsampling of the raw point cloud and it is implemented in the Point Cloud Library[3] [20], a 3D data processing library integrated within ROS. At each frame, the space is subdivided into a set of voxels (volumetric pixels) and all points inside each voxel are approximated with the coordinates of their centroid. Other than for reducing the number of points, this operation is also useful for obtaining point clouds with approximately constant density, where points density no longer depends on their distances from the sensor. In that condition the number of points of a cluster is directly related to its real size. As an example, in Fig. 4 we compare the raw point cloud of the Microsoft Kinect sensor with the result of the voxel grid filtering when choosing the voxel size to be of 0.06m.



(a) Raw Kinect point cloud. (b) After voxel grid filtering.

Fig. 4. The effect of the voxel grid filter on Kinect 3D data

- `people_detector`: this node takes as input the raw sensor data and the output of the `voxel_grid_filter` node. It removes the ground plane from the latter point cloud and performs an efficient 3D clustering optimized for people detection that allows to obtain clusters candidate to belong to people. These candidates are then validated with a HOG people detector and good clusters properties are published to a topic as the output of this node. On the same output topic (in this case named `/detector/output`) more nodes of `people_detector` type can publish. That is the case illustrated in Fig. 3, where many RGB-D sensors are present and, for every sensor within the network, a detection block is run that publishes the corresponding detections to the common topic. This node exploits also the raw sensor data to extract the highest possible number of points from the clusters belonging to

[3] `http://www.pointclouds.org`.

people. The RGB values corresponding to these points are then written to the output topic in order to be used by the `tracker` node to compute color histograms of the targets.

- `tracker`: it has the task to perform prediction and data association by having as input the detections published to the topic `/detector/output`. As stated above, these detections can come from different sensors connected to different computers and they are all used for performing people tracking.

The main part of the algorithms performed by the descripted nodes is contained in callback functions, executed every time a new message is published to their corresponding input topic. In Table 1 we report the bandwidth occupied by publishing to the main topics reported in Fig. 3. These data are very useful since they highlight which messages are too heavy and could slow down the whole application because they have to be copied between two different memory locations. They have been obtained with the ROS command `rostopic bw topic_name`. For the topic relative to the raw XYZ point cloud (`camera/depth/points`) we report the measured bandwidth at three different depth resolutions, that can be set by means of the OpenNI driver. The requested bandwidth for topic `detector/output` is quite high because the raw RGB image is also passed from the detection module to the tracker. However, it could be easily decreased by an order of magnitude by transmitting the RGB image in compressed format by means of the `image_transport` ROS package.

Table 1. Bandwidth used (in MB/s) when publishing messages to the listed topics

Topic name	Bandwidth (MB/s)
camera/depth/points (VGA)	148
camera/depth/points (QVGA)	37
camera/depth/points (QQVGA)	9.3
camera/rgb/image_color/compressed	1.1
voxel_grid/output	8.1
detector/output	30

Nodelets for Avoiding Data Copying

Every node in ROS runs in a different process, thus interprocess communication is needed in order to exchange data between nodes. That means that the computer has to spend more time and memory for passing data between them. For such a reason, as an alternative to nodes, nodelets have been designed to provide a way to run multiple algorithms in the same process with zero copy transport between algorithms.

We implemented a nodelet version of our detection module where we substituted the nodes with nodelets managed by the same nodelet manager. In Table 2 we report the framerate of the people detector for our two different implementations and for three different resolutions of the sensor depth point cloud. As it

Table 2. Comparison between nodes and nodelets versions of our people detection software in terms of framerate (fps) and at different depth resolutions

	VGA	**QVGA**	**QQVGA**
nodes	14.5	21.7	28.1
nodelets	18.3	24.8	29.9

can be noticed, the nodelets implementation led to a higher framerate and the higher is the resolution, the higher is the framerate gain because we avoid to copy large amounts of data between different processes[4].

3.2 Easy Configuration with `tf`

Some of the most frequent errors when dealing with robots with multiple joints or with multiple sensors is related to wrong reference system transformations. ROS `tf` package has been designed to easily refer data to multiple reference systems, that can vary over time, and maintains the relationship between coordinate frames in a tree structure buffered in time. This package is distributed, in the sense that there is no central source of information and all the coordinate frames are known to all ROS components on any computer in the system. Moreover, there is no loss in accuracy when transforming data multiple times. These transforms can be published or listened to with a fixed rate over time and they can be visualized with both the ROS visualizer (`rviz`), that shows their position in real time, and with the command `rosrun tf view_frames`, that saves to a file the full tree of coordinate transforms. In Fig. 5, we report the `tf` tree showing the parent/child relationship of the coordinate frames related to our robotic platform. `tf` relative to the world map and physical links of the robot are colored in cyan, while the other colors highlight the frames referring to the different sensors mounted on the robot, namely sonars, laser range finder and Kinect. In Fig. 6 we report a photo of the robotic platform (a) we used for the people following application, together with its URDF model as it is visualized in `rviz` (b). In addition, also `tf` reference axes are drawn for every robot joint and every sensor (c). This visualization allowed to easily check the correctness of the reference frames. Moreover, `rviz` makes possible visualization with hardware in the loop, so that a direct comparison between the simulated and the real behavior of the robot can be performed.

4 Experiments

4.1 Tests with a Mobile Robot

We present here some results obtained with our tracking system on RGB-D video sequences collected in an indoor environment with the mobile robot shown in Fig. 6. It consists of a Pioneer P3-AT platform equipped with a Microsoft Kinect

[4] All the other tests reported in this paper refer to the implementation based on nodes.

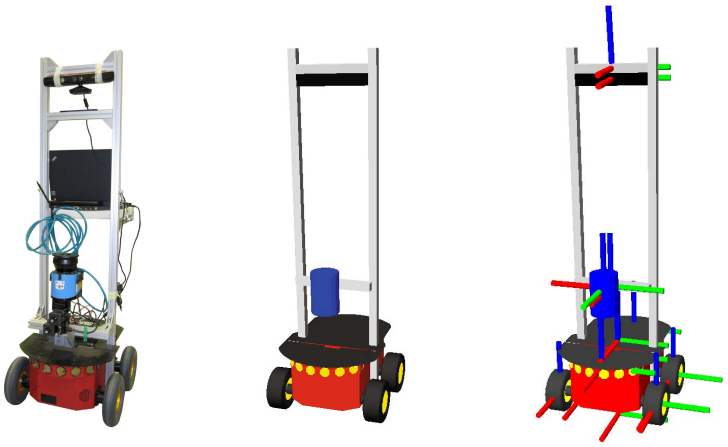
Fig. 5. Transforms (`tf`) tree representing the reference systems defined for our robotic platform and their parent-child relationships

sensor, which is endowed with a standard RGB camera, an infrared camera and an infrared projector. This low cost hardware can provide RGB-D data with 640 x 480 pixel resolution at 30 frames per second. In these tests we acquired depth data at QQVGA resolution, namely 160 x 120 pixels, for speed. It is worth to notice that this choice does not affect the accuracy achievable by our system, because of the voxel dimension we chose for the voxel grid filter we apply.

We performed tests while the robot was moving along one direction in three different scenarios of increasing difficulty:

1. no obstacle is present, people move with simple (linear) trajectories;
2. no obstacle is present, people move with complex trajectories and interact with each other;
3. obstacles (chairs, a whiteboard) are present, people move with complex trajectories and interact with each other.

Every video sequence extends over about 750 frames, thus the total test set includes 2371 frames, 6172 instances of people and 13 tracks that have been manually annotated on the RGB image and that constitute the ground truth.



(a) Test platform. (b) URDF model in `rviz`. (c) URDF model and `tf` in `rviz`.

Fig. 6. The robotic platform (a) used in the tests, its model displayed with the ROS visualizer (b) and the reference frames of every model joint (c)

The minimum distance between people is 0.2m while the minimum people-object distance is 0.05m.

For the purpose of evaluating the tracking performance we adopted the CLEAR MOT metrics [4], that consists of two indexes: MOTP and MOTA. The MOTP indicator measures how well exact positions of people are estimated, while the MOTA index gives an idea of the number of errors that are made by the tracking algorithm in terms of false negatives, false positives and mismatches. In particular, given that our ground truth does not consist of the metric positions of all persons, but of their positions inside the image, we computed the MOTP index as the average PASCAL index [11] (intersection over union of bounding boxes) of the associations between ground truth and tracker results by setting the validation threshold to 0.5. We computed the MOTA index with the following formula

$$MOTA = 1 - \frac{\sum_t (fn_t + fp_t + ID_t^{sw})}{\sum_t g_t} \tag{1}$$

where fn_t is the number of ground truth people instances (for every frame) not found by the tracker, fp_t is the number of output tracks instances that do not have correspondences with the ground truth, ID_t^{sw} represents the number of times a track corresponding to the same person changes ID over time and g_t is the total number of ground truth instances present in all frames.

In Table 3 we report, for every test sequence, the MOTP and MOTA indexes, the percentage of false positives and false negatives and the number of ID switches. The results are very good for these tests: the only ID switches are due to people who change motion direction when occluded by other people or

outside the camera field of view. In Fig. 1 we report some examples of correctly tracked frames from our test set. Different IDs are represented by different colors and the bounding box is drawn with a thick line if the algorithm estimates a person to be completely visible, while a thin line is used if a person is considered occluded.

Table 3. Tracking results for tests with a moving robot

	MOTP	MOTA	FP	FN	ID Sw.
Simple traj.	82.2%	95.8%	2.5%	1.6%	3
Complex traj.	83.5%	90.9%	4.7%	4.4%	1
With obstacles	83.3%	94.3%	4.7%	0.9%	3

4.1.1 People Following Module

As a further test for proving the robustness and the real time capabilities of our tracking method we implemented an additional module with the task of following a specific tracked person. At the implementation level this is done with a ROS node which reads the tracking results and drives the robot towards the person with the specified ID. It is worth to notice that we wrote this new node in `Python`, while the rest of the software has been written in `C++`, by exploiting the property of ROS messages to be language and platform-independent. We then asked the robot to follow a particular person along a narrow corridor with many lighting changes and a hall where many people are present. In Fig. 7 the output of the tracking algorithm is visualized for some frames collected during this experiment. In the first row it can be seen that the person is correctly tracked and followed by the platform along the corridor, while the lighting conditions considerably change. This is allowed by a good prediction of people position in ground plane coordinates, obtained with the use of the Unscented Kalman Filter that can also benefit of many available measures thanks to the high frame rate. In the second row the tracking (and following) robustness is shown when other people walk next to the followed person or between him and the robot. As a real world example, we show in Fig. 8 some screenshots of our robot that successfully managed to detect and track people within groups and to follow a particular person within a crowded environment only by means of data provided by the tracking algorithm.

4.2 Tests with a Distributed Tracking System

When doing people tracking from multiple cameras, it is useful to distribute the computational cost to multiple agents/computers. The advantage is two-fold: there is no need for a powerful computer as central unit and only selected data have to be sent over the network. If a software has been developed using ROS, distributing the computation among different machines becomes an easy task

Fig. 7. People following test. First row: examples of tracked frames while a person is robustly followed along a narrow corridor with many lighting changes. Second row: other examples of correctly tracked frames when other people are present in the scene.

Fig. 8. Sample images of our mobile robot following the person with the blue sweater within a crowded environment. It is worth to notice that the sweater has the same color of the carpet on the floor.

Table 4. Computers of the wired network used for the distributed tests

	PC1	**PC2**	**PC3**	**PC4**
CPU	Xeon Quad 3.10GHz	i5-520M 2.40 GHz	i7-620M 2.67Ghz	Core 2 Quad 2.66Ghz
RAM	4GB DDR3	4GB DDR3	4GB DDR3	4GB DDR3
Type	Desktop	Laptop	Laptop	Desktop

because a node makes no assumption about where in the network it runs, thus computation can be relocated at run-time to match the available resources.

In order to test our people tracking algorithm in a distributed configuration we used the *RGB-D People Dataset*[5] ([22,13]), which contains about 4500 RGB-D frames acquired from three vertically mounted Kinect sensors pointing towards adjacent, but not overlapping regions. As sketched in Fig. 3, we used three independent people detection modules (one for each sensor) that publish detection messages to the same topic and a tracking node that fuses them in tracks as they arrive. Detections coming from different sensors are referred to the same reference system by means of static `tf`. Data were pre-recorded in ROS `bag` files that contain Kinect depth in QQVGA resolution and RGB image in VGA resolution. We evaluated the tracking framerate of the whole process both when executed on a single computer and also when distributed on a heterogeneous computer network composed by PCs whose main specifications are reported in Table 4. PC1 and PC4 are desktop computers, while PC2 and PC3 are laptops suitable to be used on a mobile robot. As it can be seen in Table 5, the stream of a single Kinect of the dataset can be processed on PC1 at 28 fps by our detection module, while the whole detection and tracking algorithm runs at 26 fps. When processed on PC2, these framerates decrease at 23 and 19 fps respectively. When processing the whole *RGB-D People Dataset* (three detection modules and one tracking node) only on PC2, the framerate decreased at 15 fps only. Therefore, we tested a distributed configuration where the three detection modules have been run on PC2, PC3 and PC4 and the tracking node have been executed on PC1. All the computers were connected within a gigabit ethernet network. In this configuration, we measured a framerate of 58 fps as the sum of the frames processed by the three detection modules in the three computers and 31 fps for the tracker node in the fourth computer. As a result, we can notice the doubling of the tracking framerate. Moreover, it is worth to notice that the centralized processing of three Kinect streams has been possible because data were pre-recorded, while dealing with live acquisition in a centralized configuration could further decrease the tracking framerate or could not be possible due to bandwidth limitations of the USB bus on standard computers. A video with the qualitative results obtained with our distributed tracking system can be found at this link: `http://youtu.be/b70vLKFsriM`, while in Table 6 a quantitative comparison of our system with that in [13] is reported. For further reference, please refer to [17].

[5] `http://www.informatik.uni-freiburg.de/~spinello/RGBD-dataset.html`.

Table 5. Framerate of the detection and tracking modules with different test configurations

	Detector (fps)	**Tracker (fps)**
Single stream on PC1	28	26
Single stream on PC2	23	19
Three streams centralized on PC2	20	15
Three streams distributed	58	31

Table 6. Tracking evaluation with RGB-D People Dataset

	MOTP	**MOTA**	**FP**	**FN**	**ID Sw.**
Ours	73.7%	71.8%	7.7%	20.0%	19
[13]	N/A	78%	4.5%	16.8%	32

5 Conclusions and Future Works

In this paper we described the architectural design of a RGB-D people tracking software based on the ROS framework. We focused on how ROS tools and functionalities led to a modular code structure, easy to configure and to debug. Tests with a mobile robot were performed in scenarios of increasing complexity and our tracking system proved to be very effective even when dealing with strong occlusions and complex trajectories. Moreover, we proved we can reach real time performance and that our implementation can be distributed on multiple computers in order to increase the framerate and decrease the data exchange over the network with respect to a centralized computation.

As future works, we envision to decrease the bandwidth requested by the data exchange between the detection and tracking modules and to test this people tracking application when using data from multiple RGB-D sensors placed on each member of a team of autonomous robots that can communicate by means of a wireless network. Moreover, we are planning to release a new RGB-D dataset that could allow to measure the 3D accuracy obtainable when tracking people with consumer RGB-D sensors, such as Microsoft Kinect[6].

References

1. Bajracharya, M., Moghaddam, B., Howard, A., Brennan, S., Matthies, L.H.: A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle. International Journal of Robotics Research 28, 1466–1485 (2009)
2. Basso, F., Munaro, M., Michieletto, S., Pagello, E., Menegatti, E.: Fast and robust multi-people tracking from RGB-D data for a mobile robot. In: Lee, S., Cho, H., Yoon, K.-J., Lee, J. (eds.) Intelligent Autonomous Systems 12. AISC, vol. 193, pp. 265–276. Springer, Heidelberg (2012)

[6] For further information, please visit
http://www.dei.unipd.it/~munaro/KTP-dataset.html

3. Bellotto, N., Hu, H.: Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters. Auton. Robots 28, 425–438 (2010)
4. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. J. Image Video Process. 2008, 1:1–1:10 (2008)
5. Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L.: Robust tracking-by-detection using a detector confidence particle filter. In: IEEE International Conference on Computer Vision (October 2009)
6. Carballo, A., Ohya, A., Yuta, S.: People detection using range and intensity data from multi-layered laser range finders. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5849–5854 (2010)
7. Choi, W., Pantofaru, C., Savarese, S.: Detecting and tracking people using an rgb-d camera via multiple detector fusion. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1076–1083. IEEE (2011)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition 2005, vol. 1, pp. 886–893 (June 2005)
9. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: A mobile vision system for robust multi-person tracking. In: IEEE Conference on Computer Vision and Pattern Recognition 2008, pp. 1–8 (2008)
10. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: Moving obstacle detection in highly dynamic scenes. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Piscataway, NJ, USA, pp. 4451–4458 (2009)
11. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision 88, 303–338 (2010)
12. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, vol. 1, pp. 260–267 (2006)
13. Luber, M., Spinello, L., Arras, K.O.: People tracking in rgb-d data with online boosted target models. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011 (2011)
14. Martin, C., Schaffernicht, E., Scheidig, A., Gross, H.-M.: Multi-modal sensor fusion using a probabilistic aggregation scheme for people detection and tracking. Robotics and Autonomous Systems 54(9), 721–728 (2006)
15. Mitzel, D., Leibe, B.: Real-time multi-person tracking with detector assisted structure propagation. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). IEEE (2011)
16. Mozos, O., Kurazume, R., Hasegawa, T.: Multi-part people detection using 2d range data. International Journal of Social Robotics 2, 31–40 (2010)
17. Munaro, M., Basso, F., Menegatti, E.: Tracking people within groups with rgb-d data. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal (October 2012)
18. Navarro-Serment, L.E., Mertz, C., Hebert, M.: Pedestrian detection and tracking using three-dimensional ladar data. In: FSR, pp. 103–112 (2009)
19. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: Ros: an open-source robot operating system. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2009)
20. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 (2011)

21. Satake, J., Miura, J.: Robust stereo-based person detection and tracking for a person following robot. In: Workshop on People Detection and Tracking IEEE ICRA (2009)
22. Spinello, L., Arras, K.O.: People detection in rgb-d data. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011 (2011)
23. Spinello, L., Arras, K.O., Triebel, R., Siegwart, R.: A layered approach to people detection in 3d range data. In: Proc. 24th AAAI Conference on Artificial Intelligence, PGAI Track (AAAI 2010), Atlanta, USA (2010)
24. Spinello, L., Luber, M., Arras, K.O.: Tracking people in 3d using a bottom-up top-down people detector. In: IEEE International Conference on Robotics and Automation (ICRA 2011), Shanghai (2011)
25. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR 2001, vol. 1, pp. 511–518 (2001)