OpenPTrack: Open Source Multi-Camera Calibration and People Tracking for RGB-D Camera Networks

Matteo Munaro, Filippo Basso, Emanuele Menegatti

Department of Information Engineering, University of Padova, Via Gradenigo 6B, 35131 - Padova, Italy

Abstract

OpenPTrack is an open source software for multi-camera calibration and people tracking in RGB-D camera networks. It allows to track people in big volumes at sensor frame rate and currently supports a heterogeneous set of 3D sensors. In this work, we describe its user-friendly calibration procedure, which consists of simple steps with real-time feedback that allow to obtain accurate results in estimating the camera poses that are then used for tracking people. On top of a calibration based on moving a checkerboard within the tracking space and on a global optimization of cameras and checkerboards poses, a novel procedure which aligns people detections coming from all sensors in a *x-y-time* space is used for refining camera poses.
While people detection is executed locally, in the machines connected to each sensor, tracking is performed by a single node which takes into account detections from all over the network. Here we detail how a cascade of algorithms working on depth point clouds and color, infrared and disparity images is used to perform people detection from different types of sensors and in any indoor light condition.
We present experiments showing that a considerable improvement can be obtained with the proposed calibration refinement procedure that exploits people detections and we compare Kinect v1, Kinect v2 and Mesa SR4500 performance for people tracking applications. OpenPTrack is based on the *Robot Operating System* and the *Point Cloud Library* and has already been adopted in networks composed of up to ten imagers for interactive arts, education, culture and human-robot interaction applications.

Keywords: OpenPTrack, people tracking, RGB-D, open source, multi-camera, network calibration, human-robot interaction, Microsoft Kinect, Microsoft Kinect v2, Mesa SwissRanger, stereo.

1. Introduction

The ability to detect and track people in real time is useful for a variety of applications: from video surveillance to robotics, from education to art. With the advent of commercially available consumer RGB-D cameras, and continued efforts in computer vision research to improve multi-modal image and point cloud processing, robust person tracking from a single camera with the stability and responsiveness necessary to drive interactive applications is now possible at low cost [1]. However, these sensors usually have a limited field of view and working range, that only allows to track people in a narrow area. Moreover, people tracking is still more prone to errors when a person gets occluded by other people or objects. Exploiting multiple cameras in a network can solve both these issues, but the extension of people tracking algorithms to multi-camera scenarios is not straightforward.

For this purpose, we created OpenPTrack[1], an open source, BSD licensed, project launched in 2013 to create a scalable, multi-camera solution for person tracking that specifically aims to support applications in education, art, and culture. This project aims at enabling artists and creators to work with robust real-time person tracking in real-world projects. OpenPTrack aims to support *creative coders* in the arts, culture, and educational sectors who wish to experiment with real-time person tracking as an input for their applications. In or-

Email addresses: matteo.munaro@dei.unipd.it (Matteo Munaro), filippo.basso@dei.unipd.it (Filippo Basso), emg@dei.unipd.it (Emanuele Menegatti)

[1]Website: http://openptrack.org, repository: https://github.com/OpenPTrack/open_ptrack.

	OpenPTrack	**Single Kinect** (w/ Microsoft SDK or OpenNI/NiTE)	**Blob Tracking** (e.g., Community Core Vision)	**Augmented Reality** (e.g., AR Toolkit)	**Motion Capture Marker-based** (e.g., Vicon)	**RF Tracking** (e.g., Zebra)
Target audience	Education, Arts, Culture	Various	Various	Various	High-end Production	Industrial sensing
Core technology	Networked 3D imagers	3D Imager	2D Camera(s)	2D Camera	2D Cameras	Radio Frequency
Output Type	ID, 3D Centroid	ID, Skeletal Data	ID, 2D Centroid	ID, 3D Position, Orientation	ID, Dense Skeletal Data	ID, 3D Position
Max. Tracking Volume	Large	Small	Small to Medium	Small to Medium	Can be very large	Can be very large
Fusion of multiple views	Intrinsic	N/A	Up to developer	Up to developer	Intrinsic	N/A
Typical refresh rate (Hz)	30-60	30	15-30	30-60	60-120+	20-50
Lag (perceptual)	Low	Low	Low	Low	Very low	Medium
Maximum people tracked	Many	Typically 4	Tradeoff with volume	Tradeoff with volume	Many	Many
Person detection	Yes	Yes	Not usually	N/A	Yes	N/A
ID Stability	Medium	Medium to High	Low	High	High	Very high
3D Tracking	Yes	Yes	No	Yes	Yes	Yes
Skeletal Tracking	No	Yes	No	No	Yes	No
Must carry / wear something	No	No	No	Yes	Yes	Yes
Position accuracy	High	High	Varies greatly	Medium to High	Very high	Medium
Occlusion resistance	High in multi-imager nets	Low	Low	Low	High	Requires multiple tags
Visible light sensitivity	Minimal	Minimal	Yes	Yes	Some	None
IR light sensitivity	Depends on imager	Yes	Depends on imager	Depends on imager	Often	No
Costume sensitivity	Must be humanoid	Must be humanoid	None	Must show tag	Must wear marker suit	None
Multiple imager types	Yes	No	Yes	Yes	Yes	N/A
Typical setup time	Medium	Very low	Low to Medium	Low to Medium	Medium to High	Medium to High
App integration complexity	Low	Low	Low to Medium	Low to Medium	Medium to High	Low to Medium
Open source software	Yes	Usually	Usually	Usually	No	No
Off-the-shelf parts	Yes	Yes	Yes	Yes	No	No
Typical system cost	$$	$	$	$	$$$$	$$$

Figure 1: Comparison of popular tracking technologies. Techniques based on a single sensor have clear limitations in terms of tracking space and accuracy. OpenPTrack targets people tracking in large spaces, fusing multiple views for allowing to track many people with high resistance to occlusion, similarly to what can be obtained with motion capture systems or with radio frequency sensors. Nevertheless, it does not require the person to wear anything and the installation costs are considerably lower since it uses off-the-shelf parts and open source software.

der to allow application developers to easily use it for their work, we also focused on providing user-friendly calibration and tracking processes by automating most of the operations that were required to the user. The system allows to use a network of imagers to track the moving centroids (center of mass) of people within a defined area. These data are also provided as a simple JSON-formatted stream via UDP, which can be incorporated into creative coding tools like Max/MSP, Touchdesigner, and Processing, as well as a variety of other software languages and environments.

In addition to the applications listed above, OpenPTrack is also perfectly suited to be adopted in environments where a robot has to interact with a human, such as in a *RoboCup@Home* scenario [2] or when a human and a robot share the same workspace in a factory [3]. The constant and real time monitoring of people positions is a necessary condition to avoid any injury that an industrial robot could cause to a human.

OpenPTrack contains numerous state-of-the-art algorithms and pipelines for RGB and/or depth-based people detection and tracking, and has been created on top of a modular node based architecture, to support the maximum re-use of code and a distributed implementation. In particular, it is built on other open source libraries: the Point Cloud Library (PCL [4]), Open Source Computer Vision (OpenCV [5]) and Robot Operating System (ROS [6]). It currently works with the first and second generation Microsoft Kinect, Mesa Imaging Swissranger SR4500 and custom stereo cameras built with a pair of PointGrey cameras.

The main contributions of this work are:

- a real-time, easy to use, and precise calibration process that also exploits people detections for refining camera poses
- a quantitative comparison of three RGB-D sensors for the purpose of people tracking
- a number of new features and optimizations implemented in OpenPTrack with respect to the people detection and tracking algorithms in [1]

The remainder of the paper is organized as follows: In Section 2, we review the most recent works on RGB-D people detection and tracking and camera networks, while in Section 3 we present the sensors that can be used to build an OpenPTrack network. Section 4 describes the three stages of the procedure implemented in OpenPTrack for calibrating a network of RGB-D sensors in real time. Section 5 gives an overview of our detection algorithms and of the differences with [1], while Section 6 details an efficient version of the tracking node and Section 7 explains our efforts towards system usability. Experiments on network calibration and people tracking with different sensors are reported in Section 8, while some interactive applications exploiting OpenPTrack are described in Section 9. Conclusions are drawn in Section 10.

2. Related work

2.1. People tracking from RGB-D data

Since the introduction of low-cost RGB-D sensors, a number of people detection and tracking algorithms which exploit combined color and depth information have been proposed. Most of these approaches apply a sliding window technique to RGB and/or depth images for people detection ([7], [8]), thus requiring high parallelization with GPUs to obtain real time performance. Moreover, the preferred tracking algorithm in this context is a multi-hypothesis tracking, that is able to recover from failures, but is also computationally expensive, thus hardly usable in real time scenarios with many people.

Recent works ([9], [10], [1], [3], [11]) allow to avoid the sliding window approach that usually leads to analyze thousands of detection windows per image. These new methods exploit the assumption, also adopted by OpenPTrack, that people stand/walk on a ground plane and cluster algorithms on depth data to find a small number of Regions Of Interest (ROIs) which are candidates to contain people and are then classified with more robust and computational demanding algorithms.

An open source implementation of the people detection algorithms in [10] and [1] is available in the Point Cloud Library [4]. In ROS-Industrial *Human Tracker* repository[2], these algorithms are combined with other people detectors in a people detection cascade which obtains even more computational efficiency and which is combined with the tracking algorithm described in [3]. Also the code for the GPU-based and the CPU-based versions of the people detection and tracking software in [11] has been recently released[3].

2.2. Multi-camera people tracking

All of the software described above is targeted to track people from a single RGB-D camera which can move onboard a wheeled robot. However, none of these provides the possibility to perform people tracking in a distributed fashion by exploiting multiple cameras.

Some works exist on people tracking and re-identification from multiple 2D cameras. Some of them assume that the network has been calibrated ([12], [13]), while others perform tracking with uncalibrated cameras [14]. One of the main methods for determining spatial positions of people is to geometrically transform images based on a predetermined ground plane homography ([15], [16], [17]). Tracking can then be done based on the estimated ground plane positions. Unfortunately, none of these works address the problem of network calibration and they do not provide an open source implementation of the presented methods.

2.3. Multi-camera calibration

The problem of network calibration is not necessarily connected to the people tracking application and concerns a whole branch of works in literature. Auvinet et al. [18] proposed a new method for calibrating multiple depth cameras for body reconstruction by using only depth information. Their algorithm is based on plane intersections and the NTP protocol for data synchronization. The calibration achieves good results: even if the depth error of the sensor is 10 mm, the reconstruction error with three depth cameras is, in the best case, less than 6 mm. A drawback of their implementation is that they have to manually select the plane corners and, above all, they only deal with depth sensors, thus avoiding the possibility to add the color information to the fused data. Another approach to solve the calibration problem is the one proposed by Le and Ng [19]: they jointly calibrate groups of sensors. More specifically, each group is composed by a set of sensors that can provide a 3D representation of the world (e.g. a stereo camera, an RGB camera and a depth camera, etc.). First of all, they calibrate the intrinsics of each sensor, secondly they calibrate the extrinsic parameters of each group and then they calibrate the extrinsic parameters of each group with respect to all the others. Finally, they refine the calibration parameters of the entire system in one optimization step. Their experiments show that this method not only reduces the calibration error, but also requires little human intervention.

A procedure that jointly calibrates the pose of all the sensors does not accumulate errors like a calibration based on sensor pairs do. For this reason, we act in a similar way for performing the calibration that involves the use of a checkerboard.

The idea of using people detections for calibration has been recently explored in literature: in [20], detections are used for computing the intrinsic parameters of a camera together with the foot-head homology, while, in [21], they are exploited for estimating the plausible range of 3D position and height for people walking on the ground plane. In this work, instead, we use peo-

[2] https://github.com/ros-industrial/human_tracker/tree/develop.

[3] http://www.vision.rwth-aachen.de/software/realpdt/realpdt.

Table 1: Comparison of the sensors currently supported by OpenPTrack.

	Kinect v1	**Kinect v2**	**SR4500**	**Stereo**
Depth resolution	320x240	512x424	172x144	variable
Depth range [m]	0.8 - 8	0.4 - 10	0.4 - 9	variable
Intensity resolution	640x480	1920x1080	172x144	variable
Type of intensity	color	color/infrared	infrared	color/mono/infrared
Error VS distance	quadratic	nearly constant	nearly constant	quadratic
Crosstalk	yes	no	yes	no
Visible light sensitivity	no	no	no	high for color cam
Infrared light sensitivity	high	low	low	high for IR cam
Optimal frame rate (fps)	30	30	13	variable

ple detections for refining the relative pose of multiple cameras.

OpenPTrack, not only addresses the problem of people tracking in a distributed network of RGB-D sensors, but also proposes the whole pipeline necessary to perform the calibration of the sensor network. It extends state-of-the-art approaches in terms of scalability and ease of use in multi-camera scenarios, where the problems caused by occlusion and limited field of view can be solved.

A comparison table of some popular approaches to people tracking is reported in Figure 1. This table illustrates the gap that OpenPTrack aims to fill: scalable, robust, real-time person tracking using affordable off-the-shelf components and an open source codebase.

3. Supported sensors

The OpenPTrack library estimates people tracking information from point cloud data containing depth and intensity information. The intensity information can be derived from color, grayscale or infrared data, thus allowing to use OpenPTrack with a wide range of existing and future sensors. For all supported sensors, instructions for drivers installation and sensor calibration are provided. In Table 1, we report a comparison between the sensors currently supported by OpenPTrack.

3.1. Kinect v1

The first generation Microsoft Kinect[4] (Kinect v1) exploits a structured-light sensor for estimating depth information and a color camera for acquiring color. The main problems of using networks of Kinect v1 sensors derive from the high depth estimation errors at far range and from the crosstalk between sensors, that is the disturbance that is created when more sensors project their structured pattern to the same scene. Moreover, Kinect v1 depth estimation highly degrades in presence of external infrared light, e.g., sunlight, thus it is only suited to be used indoors. OpenPTrack could work also with *Asus Xtion* and *Primesense Carmine* sensors, that are very similar to Kinect v1, but they are not officially supported since they are not produced anymore.

3.2. Kinect v2

Kinect v2[5] estimates depth by exploiting the Time-of-Flight (ToF) principle [22], i.e., an array of emitters send out a modulated signal that travels to the measured point, gets reflected and is received by the CCD of the sensor. Kinect v2 is probably the best sensor to be used in multi-camera scenarios since its depth estimation error is nearly constant up to 10 meters [23] and there is no crosstalk between multiple sensors. This sensor can also be used outdoors to track people in the near range thanks to its low sensitivity to infrared light. As we will see, OpenPTrack uses Kinect v2 color image at the calibration stage, to exploit its high resolution, while it uses its infrared image for people detection and tracking, in order to be invariant to lighting conditions.

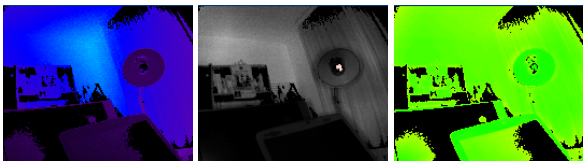
3.3. SwissRanger 4500

The Mesa SwissRanger SR4500[6] is another time-of-flight sensor, but with lower resolution than Kinect v2. Other than a depth map, it also provides an infrared intensity image and a confidence value for every pixel stating the confidence of depth data. An example of this kind of data is reported in Figure 2. Even if it has lower resolution than Kinect v1 and v2, depth data are more continuous and accurate than for Kinect v1, thanks to the exploitation of ToF technology. Its sensitivity to external lighting is similar to that of Kinect v2, but it does

[4] `http://www.xbox.com/it-IT/Xbox360/Accessories/kinect`.

[5] `http://www.microsoft.com/en-us/kinectforwindows`.

[6] `http://www.mesa-imaging.ch/products/sr4500`.



(a) Depth (b) Intensity (c) Confidence

Figure 2: Data produced by the Mesa SwissRanger SR4500.

suffer from crosstalk problems. However, the possibility to use heterogeneous sensors with OpenPTrack also allows to alternate different sensors within the network in order to reduce the crosstalk between sensors of the same type.

3.4. Stereo cameras

OpenPTrack works also with data coming from passive sensors, such as stereo cameras, that are the best sensors to be used outdoors. The possibility to use custom-built stereo pairs is provided, in order to allow the user to choose the resolution and baseline most suitable to his needs. For example, tracking could be extended to farther range than that obtained with the other supported sensors by choosing a wide baseline for the stereo pair. At the moment of writing, BlackFly[7] PointGrey cameras are fully supported for creating custom stereo cameras, from intrinsic calibration to tracking.

4. Network calibration

In order to fuse people detections coming from every sensor, all camera poses must be accurately known with respect to a common reference frame. In OpenPTrack, the calibration of the camera network consists of three stages, as depicted in Figure 3. At first, the RGB-D sensors must be intrinsically calibrated; then, the camera poses within the network are estimated by means of a checkerboard; finally, these poses are refined by exploiting people detections coming from all sensors.

4.1. Intrinsic calibration of the sensors

Intrinsic calibration consists in estimating focal length, optical center and distortion coefficients of the cameras composing the RGB-D sensors. If a sensor is composed of multiple cameras, also the relative pose between them must be calibrated. This calibration is separately performed for every sensor and OpenPTrack provides standard procedures to estimate the needed parameters by imaging a checkerboard from multiple viewpoints[8].

4.2. Real-time extrinsic calibration with a checkerboard

Unlike intrinsic calibration, extrinsic calibration involves all sensors at once, since it consists in acquiring images from all sensors while moving a checkerboard within the tracking space and in performing a global optimization of the camera and checkerboard poses. This procedure relies on ROS communication and networking capabilities, can be executed in little time and provides feedback in real time about the calibration status. In particular, once the network of computers and sensors is set up[9], the user is requested to move a checkerboard in front of the cameras. When the checkerboard is seen by a pair of cameras, the relative pose between the two is estimated in real time and the resulting reference axes are drawn within the ROS visualizer, as shown in Figure 4a. Then, every time a new camera sees the checkerboard at the same time of one of the already calibrated cameras, the new camera pose is estimated and added to visualization. After all cameras have been calibrated, the user is requested to place the checkerboard on the ground, as in Figure 4b. With this operation, the position of the ground plane with respect to each camera is calibrated.

The size of the checkerboard can be specified via configuration file. In the examples reported in this work, we used a 5x6 checkerboard with squares of 0.12 meters, that is usually recognized at distances up to 6 meters from cameras with VGA resolution, e.g. Kinect v1. The field of view overlap that is needed for two cameras to see the checkerboard can vary according to the checkerboard distance and its orientation with respect to both cameras, thus having real-time feedback of the calibration in the visualizer is very important for the user.

4.2.1. Optimization

The camera poses are optimized together with the checkerboard poses in a bundle adjustment fashion relying on the `calibration_toolkit`[10] [24] package and the *Ceres Solver* library [25] Since more than two cameras can see the checkerboard and more frames are collected over time, many constraints can be added to this

[7] `http://www.ptgrey.com/blackfly-usb3-vision-cameras`.

[8] Instructions on how to perform intrinsic calibration of the supported sensors are reported here: `https://github.com/OpenPTrack/open_ptrack/wiki/Intrinsic-Calibration`.

[9] All computers of the network must be time-synchronized.

[10] `https://github.com/iaslab-unipd/calibration_toolkit`.

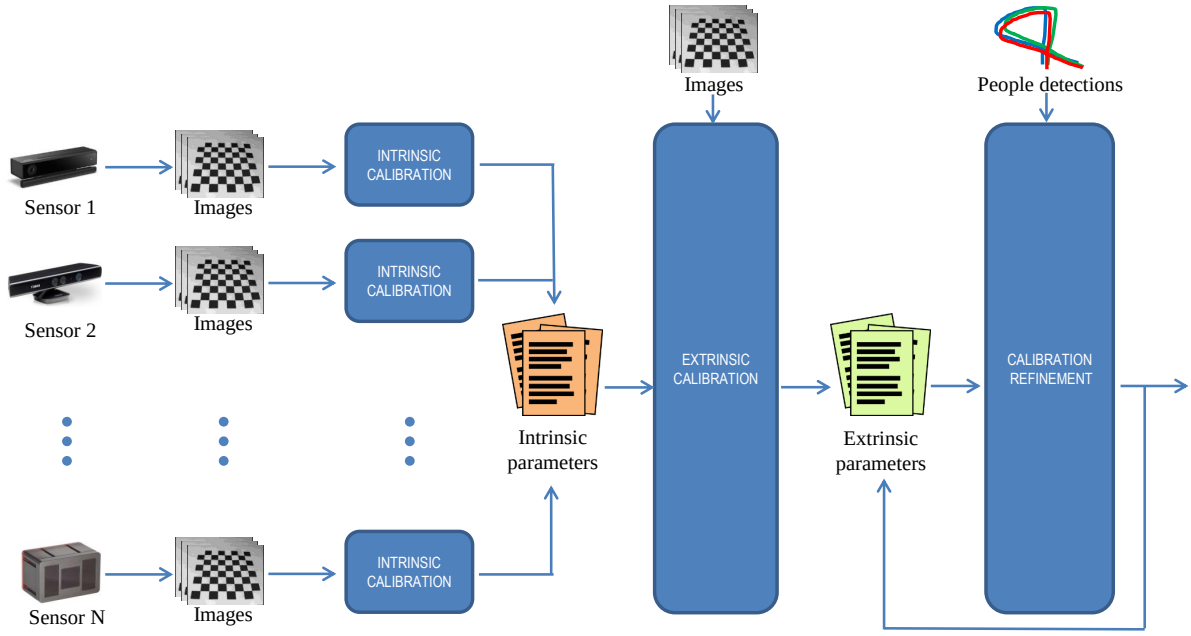


Figure 3: Calibration workflow in OpenPTrack. Three stages can be identified: Intrinsic calibration, extrinsic calibration and calibration refinement with people detections.

optimization problem. In particular, let B be a checkerboard with reference frame $\mathcal{B}$ and let ${}^{\mathcal{B}}\mathbf{B}$ be the set of its corners (in the checkerboard's coordinate system). Let also $\overline{\mathsf{C}} = \{\mathsf{C}_1, \mathsf{C}_2 \dots \mathsf{C}_N\}$ be the set of cameras.
According to the above notation, and supposing we have already performed K acquisition steps, we can easily enumerate the constraints the acquired data impose:

1. The pose of every camera $({}^{\mathcal{W}}C_n, n = 1 \dots N)$, with respect to the world reference frame $\mathcal{W}$, must be the same at each step k.
2. The pose of the checkerboard with respect to the world reference frame at step k, namely ${}^{\mathcal{W}}\mathcal{B}_k$, must be kept constant.

We can therefore design our error function e as

$$\mathrm{e} = \sum_{k=1}^{K} \left[\frac{1}{\sigma^2_{\mathrm{cam}}} \sum_{n=1}^{N} u_{nk} \cdot \mathrm{e}_{\mathrm{cam}}({}^{\mathcal{W}}C_n, {}^{\mathcal{W}}\mathcal{B}_k) \right], \qquad (1)$$

where $\mathrm{e}_{\mathrm{cam}}(\cdot)$ is the error we can compute on the image data, while σ_{cam} is a normalization factor. u_{nk} is instead an indicator function: it is equal to 1 if at step k camera C_n sees the checkerboard B, otherwise it is equal to 0.

The error e_{cam} is computed as the reprojection error of the checkerboard corners onto the image. For each corner $\mathbf{b} \in \mathbf{B}$ we call $\hat{\mathbf{b}}$ the corner extracted from the image in pixel coordinates. The reprojection error of the corners can be computed as

$$\mathrm{e}_{\mathrm{cam}}({}^{\mathcal{W}}C, {}^{\mathcal{W}}\mathcal{B}) = \sum_{\mathbf{b} \in \mathbf{B}} \left\| \mathrm{repr}_{\mathsf{C}}({}^{C}\mathbf{b}) - \hat{\mathbf{b}} \right\|^2, \qquad (2)$$

where $\mathrm{repr}_{\mathsf{C}}(\cdot)$ is the reprojection function that returns the pixel coordinates of a 3D point using camera C intrinsic parameters. Further details about the optimization algorithm can be found in [24].

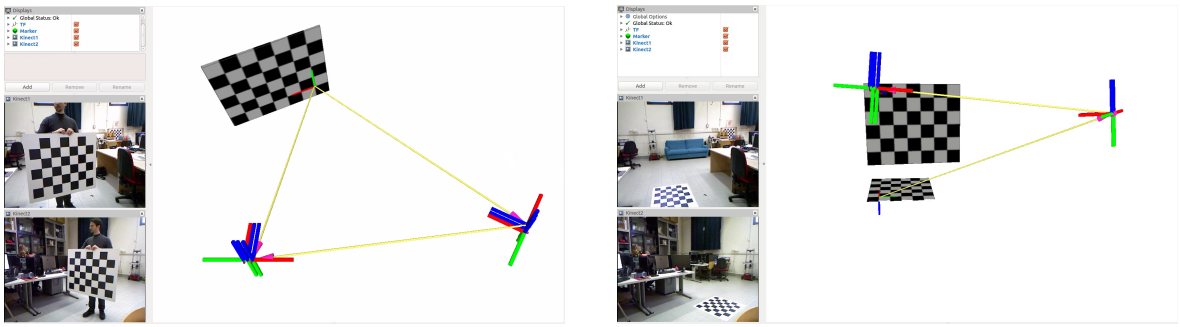
This optimization step is repeated at constant time intervals during the calibration phase and its results are used to update the visualizer in real time.

4.2.2. *Extrinsic calibration results*

When all cameras have been calibrated, the user is prompted to save the results. At this point, a tree containing the relative transformations between the cameras is produced. Moreover, the first transformation of the tree is the one connecting the ground reference frame (`world`) with the first calibrated camera.

This calibration procedure has been made generic so as to work with heterogeneous sensors. At the time of writing, all sensors reported in Section 3 can be calibrated together in the same OpenPTrack network.

Unlike Kinects, the SR4500 sensor is not coupled with a color camera. Since the basis of our extrinsic calibration method is an algorithm which finds checker-



(a) Pairwise calibration

(b) Ground plane calibration

Figure 4: Screenshots from the real-time extrinsic calibration procedure provided with OpenPTrack.

boards within color/grayscale images[11], we use SwissRanger intensity images for calibration in the same way we use Kinect color images. In order for the checkerboards to be detected on those intensity images, we perform a proper histogram equalization of the intensity values and an image upsampling to fit the minimum image resolution expected by the checkerboard finding algorithm.

In Figure 5, color images from three Kinects v1 and equalized intensity images from three SR4500 are shown, together with the output of our extrinsic calibration procedure visualized as reference frames of every sensor with respect to the ground reference frame.

In Figure 6, we propose a qualitative evaluation of two pairwise calibrations. In Figure 6a, the point clouds obtained by two Kinects v1 are shown when referred to a common reference frame obtained with calibration. The two Kinects were observing the scene with a difference in the point of view of about 120°. The accuracy in the point clouds alignment can be better appreciated if looking at the person's legs. In Figure 6b, instead, we report in false colors the point cloud obtained with a SwissRanger and, in RGB, the point cloud obtained with a Kinect v1. It can be noticed how our procedure allows to align data coming from these two different sensors. However, since every sensor has a different intrinsic distortion in estimating the depth, better results could be obtained if taking into account a correction of the depth maps as described in [26].

4.2.3. *Multi-frame ground plane estimation*

As stated in Section 4.2, the ground plane equation and the world reference frame are determined by positioning a checkerboard on the floor after that all cameras have been calibrated. However, when the checkerboard is placed on the floor but far from every camera, the results could be not satisfactory: in particular, the estimated ground plane could present a non-negligible rotation with respect to the real one. In many cases, printing a bigger checkerboard is not a viable solution to overcome this issue. Instead, we developed a further procedure, optional in OpenPTrack, that adds new constraints to the optimization cost function. In particular, the user is requested to place the checkerboard in different positions on the ground plane and the algorithm then imposes that all the detected checkerboards lie on the same plane by adding new geometrical constraints to the cost function. In fact, if we fix the plane π on which a checkerboard can move, the checkerboard pose can be defined by a 2D transform ${}^{\mathcal{P}}\mathcal{B}_2$ with respect to the reference frame of plane π, namely $\mathcal{P}$. This transform consists in two translations (t_x, t_y) and a 2D rotation θ.

So, let define a plane by means of its reference frame ${}^{\mathcal{W}}\mathcal{P}$, such that the x- and y-axes are on the plane and the z-axis is its normal. The pose of a checkerboard lying on π at step k can be written as

$$ {}^{\mathcal{W}}\mathcal{B}_k = {}^{\mathcal{W}}\mathcal{P} \cdot {}^{\mathcal{P}}\mathcal{B}_{2_k}, \tag{3} $$

where the 2D transform ${}^{\mathcal{P}}\mathcal{B}_{2_k}$ is wrapped into a 3D one to perform the matrix multiplication

$$ {}^{\mathcal{P}}\mathcal{B}_{2_k} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & t_x \\ \sin(\theta) & \cos(\theta) & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4} $$

It is then possible to refine both the plane and the checkerboard pose in the optimization by substituting (3) into (1).

[11] `findChessboardCorners` method in OpenCV.

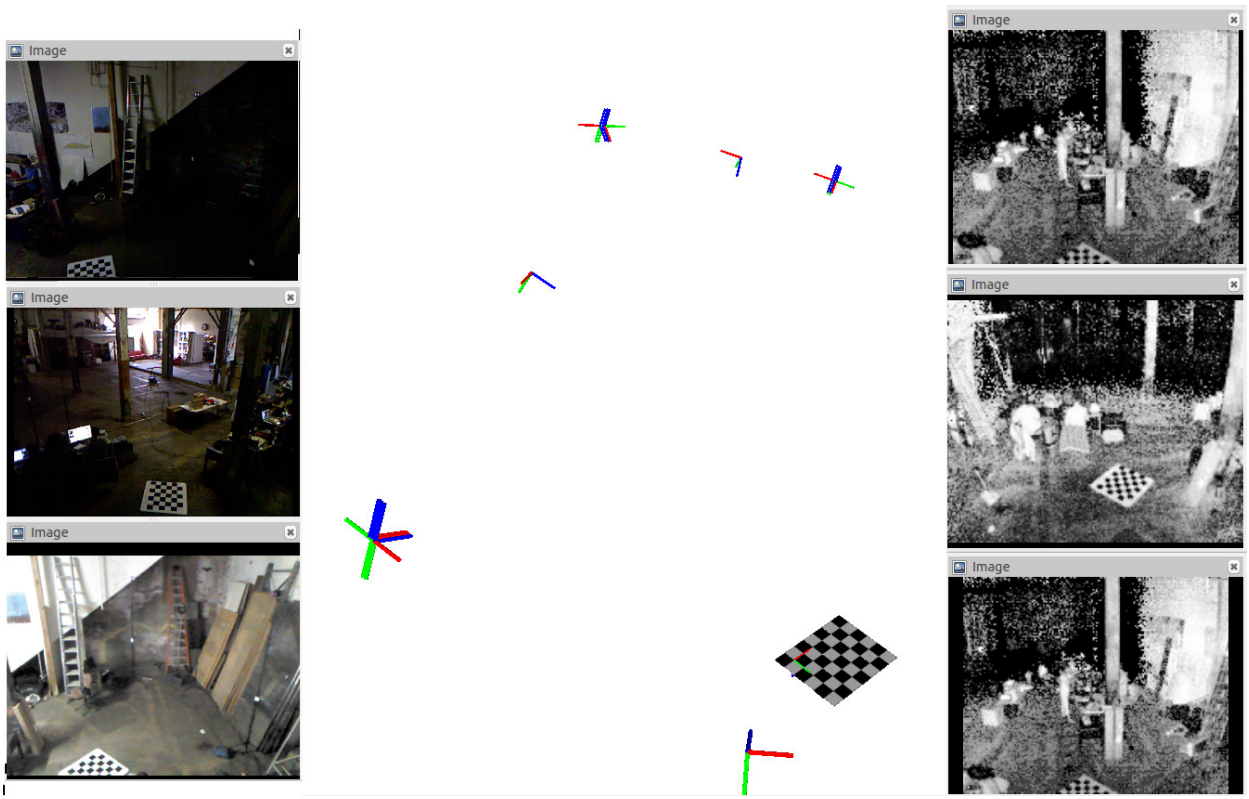
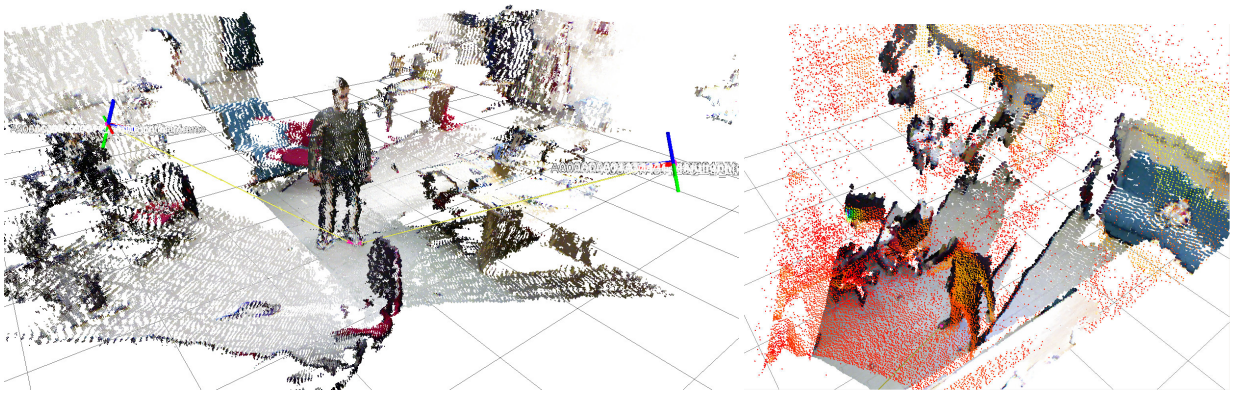


Figure 5: Extrinsic calibration results for a network composed of three Kinects v1 (images on the left) and three SwissRangers SR4500 (images on the right).



(a) Point clouds from two Kinects v1

(b) Point clouds from a Kinect v1 and a SwissRanger

Figure 6: Point cloud alignment based on extrinsic calibration obtained with OpenPTrack.

4.3. Calibration refinement with people detections

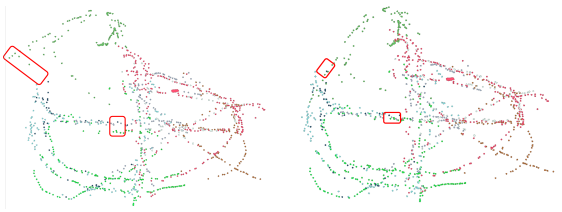
The network calibration procedure described in Section 4.2 estimates camera poses by exploiting the intensity images (color or infrared) of the sensors. However, people position in the tracking space is extracted from sensors depth (point cloud) data. For this reason, for tracking to work well, point clouds generated by each sensor should result as aligned as possible after calibration, otherwise the same person seen from different sensors could produce two different tracks, thus generating a *splitting*. To refine the calibration estimated in Section 4.2 so as to reflect a good alignment of sensors point clouds, we developed a novel procedure that, assuming that only one person is walking within the tracking space, exploits people detection trajectories coming from all the sensors and align them one to another. The input of this algorithm is then composed of trajectories D_i like those represented in Figure 7a, where different colors mean that they come from different sensors. A timestamp is also associated to each of those points. These trajectories are referred to the same reference frame according to the calibration results obtained with the procedure in Section 4.2. The main assumption of the refinement algorithm is that different points with the same timestamp should ideally coincide because only one person is present.

To exploit this constraint, we implemented the algorithm reported here below.

Algorithm 1 Calibration refinement

Input: $\mathbf{D} \leftarrow (D_i)$
for $i \leftarrow 1, N_{sensors}$ **do**
  $RT_i \leftarrow I_{4x4}$
  $D_i^w \leftarrow$ WarpInTime(D_i)
end for
$icp\_maxD^t \leftarrow icp\_maxD$
for $i \leftarrow 1, N_{iter}$ **do**
  $ave\_cloud^w \leftarrow$ AverageCloud$(\mathbf{D}^w)$
  $(\mathbf{RT}, \mathbf{D}^w) \leftarrow$ ICPToAveCloud$(\mathbf{RT}, \mathbf{D}^w, ave_cloud^w, icp_maxD^t)$
  $icp\_maxD^t \leftarrow icp\_maxD^t/2$
end for
$\mathbf{D} \leftarrow$ Transform$(\mathbf{D}, \mathbf{RT})$
$ave\_cloud^p \leftarrow$ ProjectToGround(ave_cloud^w)
$\mathbf{RT} \leftarrow$ ICPToAveCloud$(\mathbf{RT}, \mathbf{D}, ave_cloud^p, icp_maxD)$

Remembering that the world reference frame is on the floor and that Z is the normal to the ground plane, WarpInTime is a method that warps detection trajectories so as to substitute the Z coordinate with the time coordinate. The trajectories before and after this warping process can be seen in Figure 7a and Figure 9a, respectively. This preliminary step allows to impose the time-position constraint by globally aligning these point clouds extended in the time dimension. Without this step, a global alignment of detection point clouds would not consider time, thus trying to align also detections close in space but very distant in time.



(a) Before (b) After

Figure 7: People detections from all sensors (a) before and (b) after the calibration refinement procedure.

(a) Before (b) After

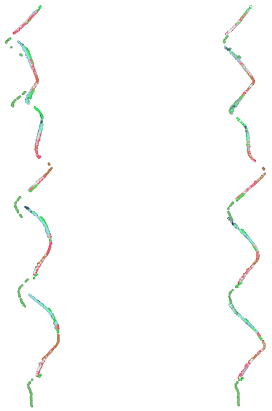
Figure 8: Side view: people detections from all sensors (a) before and (b) after the calibration refinement procedure.

As a reference cloud for aligning all detection clouds, an average of the different trajectories is computed in the $X - Y - time$ space. This is done by the AverageCloud method, while the alignment is performed with the Iterative Closest Point (ICP) [27] algorithm in ICPToAveCloud.

The alignment is iteratively repeated N_{iter} times (four by default) while decreasing the maximum distance allowed between corresponding points, so that the registration is refined in more steps.

The Transform method applies the estimated transformation to a detection cloud, while the ProjectToGround method computes a projection of a point cloud on a given plane, the ground plane in this case. These methods are exploited because a final alignment is performed only in the $X - Y - Z$ space starting from the alignment obtained until that point. This step ensure that the detection point clouds are correctly aligned also along the Z dimension.

In Figure 9b, the effect of the alignment in the $X - Y - time$ space can be appreciated. A clear improvement with respect to Figure 9a is visible because one trajectory was quite off before the alignment. The detection trajectories in $X - Y - Z$ space aligned according to the estimated final transformation are visualized in Figure 7b and Figure 8b from a top view and a side view, respectively. Some of the differences before and after the refinement are highlighted in red.



(a) Before (b) After

Figure 9: People detections warped in time (a) before and (b) after the calibration refinement procedure.

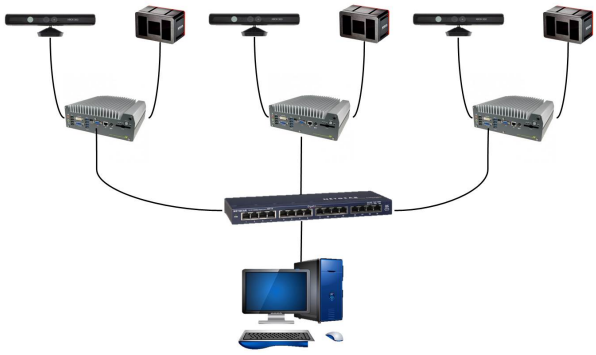


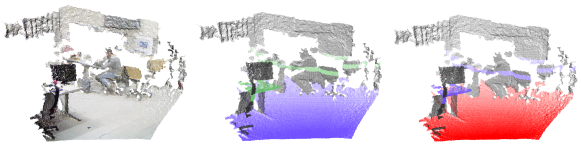
Figure 10: Example of camera network composed by three Kinect v1 and three SR4500. Sensors are directly connected to distributed PCs which perform people detection. Then, detections are sent through the network and read/used by the PC running the tracking algorithm.

5. Distributed people detection

OpenPTrack allows to perform people tracking within a camera network by distributing people detection and centralizing the tracking process, as proposed in [28]. As depicted in Figure 10, the sensors are directly attached to a computer which analyzes the data stream and performs people detection. Only the detections are sent through the network, in order to be fused at the tracking level after being referred to a common reference frame by means of calibration data.

5.1. Choice of ground plane for people detection

The people detection algorithms implemented in OpenPTrack rely on the ground plane assumption. Thus, the ground plane equation has to be estimated at startup. This plane can be extracted from point cloud data by finding all the planes by means of the `OrganizedMultiPlaneSegmentation` algorithm [29] in the Point Cloud Library and then by choosing, among them, the one that is fairly horizontal and placed under any other plane. As an alternative to the automatic selection, the user can manually select the ground by clicking on three point cloud points (manual selection), or he can select the ground plane among those produced by an automatic multi plane segmentation (semi-automatic selection). In Figure 11a, a sample point cloud is shown, while in Figure 11b the planes resulting from the segmentation are shown with different colors. Finally, the plane selected by the automatic procedure is highlighted in red in Figure 11c.



(a) (b) (c)

Figure 11: Automatic ground plane estimation: (a) sample point cloud, (b) output of the multi-plane segmentation, (c) output of the automatic ground plane selection (in red the chosen plane).

It is worth noting that all cameras within a network share the same ground plane equation and that this can be derived from calibration, without the need for estimating it separately for every sensor. However, there are scenarios where the ground plane estimate obtained with calibration is more accurate than the one obtained from analyzing the point cloud and some other scenarios where the opposite is true. Thus, in OpenPTrack, we implemented a heuristic approach that is more likely to provide the best estimate in every scenario:

1. the ground plane equation is estimated from point cloud data
2. the ground plane equation is also estimated from calibration data
3. if the ground estimated from the point cloud is not valid or it is too distant from the plane derived from calibration, the latter is kept as valid
4. if the ground estimated from the point cloud is close to the one derived from calibration, the former is used. In fact, the ground estimated with calibration could be less precise than that obtained directly with detection from the point cloud because the former suffers from the problem of error propagation in the camera poses.

5.2. People detection in color and infrared images

In OpenPTrack, algorithms for people detection in color, infrared and disparity images are used, depending on the sensor. As a first step, a depth-based clustering similar to the one proposed in [1] is always applied on point cloud data in order to determine clusters of points candidate to contain a person. The main difference with [1] is that, in OpenPTrack, people detection is performed in the ground plane reference frame instead than in the sensor reference frame, thus improving detection performance when the sensor is tilted.

For Kinect v1 and stereo cameras, that can produce color images, a HOG-based people detector [1] is then applied to these images in correspondence of the clusters extracted from the point cloud. Since Kinect v2 and SR4500 provide infrared images, the HOG-based people detector is applied to these images because they are invariant to visible lighting. For improving gradient estimation and thus HOG descriptors, the image contrast is enhanced with a histogram equalization algorithm. The choice of exploiting infrared images for people detection makes tracking work even with poor or no illumination.

It is also worth mentioning that, for SR4500, the point cloud points associated to low confidence values are filtered out before applying our depth-based clustering algorithm, since they are likely to be due to noise. In Figure 12, some people detection results obtained on SR4500 data are reported. As we will see in more detail in Section 8, even if SR4500 has considerably lower resolution than Kinect v1, our software allows to obtain comparable results.

5.3. People detection in disparity images

Kinect v1's infrared images cannot be used for people detection because of the dotted pattern that is projected all over the image. Thus, for making tracking work in every lighting condition also with this sensor, a people detection algorithm working on disparity images is also exploited, while the algorithm on color images is automatically disabled if lighting is not good. In summary, for Kinect v1:

- the depth-based clustering proposed in [1] is applied on point cloud data;
- if the mean luminance of the image is over a threshold, the HOG-based people detector [1] is applied to the RGB image at the clusters position;
- then, an Adaboost classifier based on Haar features extracted from the disparity map [3] is applied to the clusters resulting from the previous steps.

This pipeline allows to exploit both RGB and depth information for obtaining the best results when the color image is good, while using only depth data if the color image is too dark.

5.4. 3D background subtraction for static scenes

By default, OpenPTrack does not assume the background to be static to perform people detection, thus it works even if some objects in the workspace are moved while performing tracking. However, since background subtraction can improve people detection for static scenes, we also implemented this feature, that can be enabled via a configuration parameter. If this is the case, the background is learned by every detection node at system start-up or when this feature is enabled and then used to remove background points from the sensors point clouds. In particular, all point clouds acquired in the background learning stage are used to fill a PCL `octree` structure, that allows to efficiently retain the foreground points from new point clouds, so that people detection is performed only on the foreground.

6. Light-weight tracking algorithm

The tracking node is a centralized algorithm which receives detections from all over the network and performs data association every time a new set of detections arrives.

In [1], this is done as a maximization of a joint likelihood composed by three terms: motion, color appearance and people detection confidence. For evaluating color appearance, a person classifier for every target is learned online by using features extracted from the color histogram of the target and by choosing as negative examples also the other detections inside the image. Thus, the tracker receives from the detection nodes the appearance information of every detection together with its position information and runs a classifier update by means of the Online Adaboost algorithm [1] for every possible detection-track association.

When dealing with a low number of cameras in the network, the bandwidth necessary to transfer appearance information to the tracking node and the time to update the appearance classifiers are negligible, but, when the network is composed of dozens of cameras, they could lead to bandwidth saturation or high computational burden since the number of people and of detection messages to be processed could be considerably higher. Since, with OpenPTrack, we were targeting a light-weight tracking algorithm that could handle as many cameras as possible, we removed the appearance term from the joint likelihood proposed in [1], so

(a) (b) (c) (d)

Figure 12: People detection results on SR4500 infrared intensity images.

that only motion information and people detection confidence are used.

In particular, as motion term, we compute the Mahalanobis distance between track i and detection j as

$$D_M^{i,j} = \tilde{\mathbf{z}}_k^{\mathrm{T}}(i,j) \cdot \mathbf{S}_k^{-1}(i) \cdot \tilde{\mathbf{z}}_k(i,j) \tag{5}$$

where $\mathbf{S}_k(i)$ is the covariance matrix of track i provided by a filter and $\tilde{\mathbf{z}}_k(i,j)$ is the residual vector between measurement vector based on detection j and output prediction vector for track i:

$$\tilde{\mathbf{z}}_k(i,j) = \mathbf{z}_k(i,j) - \hat{\mathbf{z}}_{k|k-1}(i). \tag{6}$$

The values we compare with the Mahalanobis distance represent people positions and velocities in ground plane coordinates. Given a track i and a detection j, the measurement vector $\mathbf{z}_k(i,j)$ is composed by the position of detection j and the velocity that track i would have if detection j were associated to it.

An Unscented Kalman Filter is exploited to predict people positions and velocities along the two ground plane axes (x, y). As people motion model we chose a constant velocity model because it is good at managing full occlusions, as described in [30]. Given that the Mahalanobis distance for multinormal distributions is distributed as a chi-square [31], we use this distribution for defining a gating function for the possible associations.

People detection confidence is used for defining a policy for initializing, updating and removing tracks, as in [1].

As a future work, we envision to compute also compact yet effective signatures of people, similar to those in [32], which could allow for better data association while preserving the scalability properties of the system.

7. Ease of use and real-time feedback

One of the main focus points when designing OpenPTrack was ease of use and configuration also for non-expert users. For this reason, we developed a calibration procedure that provides real time feedback to the users and automatically distributes calibration files over the network. Moreover, all detection and tracking parameters can be changed in real time while the system is running by exploiting ROS Graphic User Interfaces implemented in the `dynamic_reconfigure`[12] package, as shown in Figure 13.

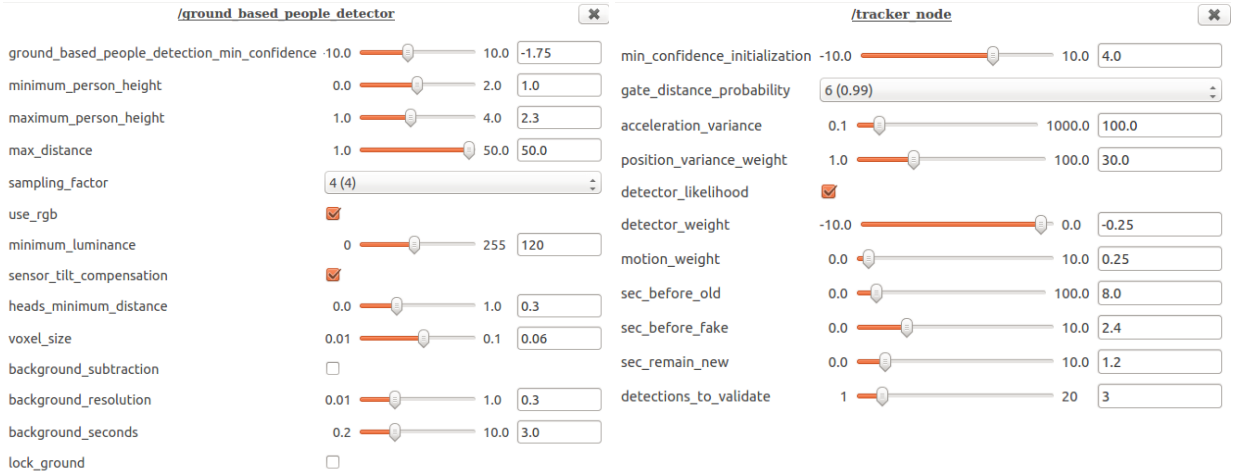
All tracking data, such as tracks ID, position, height and confidence, are also sent via UDP as JSON-formatted messages, thus being ready to be exploited by web-based applications. A moving average filtering node is also implemented and applied to the tracking output in order to let the user choose the desired tracking frame rate and the amount of smoothing in people trajectories.

A fundamental step for multi-sensor calibration and tracking to work is the time synchronization of the network, that is not yet automated, but it is foreseen as a future work by exploiting the Network Time Protocol (NTP[13]).

8. Experiments

So far, the OpenPTrack library has been used in real world scenarios for tracking people within networks composed of up to ten sensors. In Figure 14, an example of tracking output from a single Kinect v1 camera is reported: on the left, the tracks of four people are drawn with different colors, while, on the right, people detection results are shown for the current frame. It is worth noting that these results refer to a scenario where the cameras were four meters high and considerably tilted. Moreover, the background was very cluttered, but our combination of RGB and depth algorithms allowed to obtain robust detection results.

[12] `http://wiki.ros.org/dynamic_reconfigure`.
[13] `http://www.ntp.org`.



(a) Detection GUI

(b) Tracker GUI

Figure 13: Examples of Graphic User Interfaces for real time parameter selection for the detection and tracking nodes.

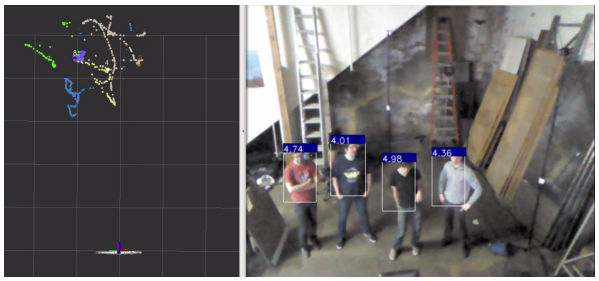
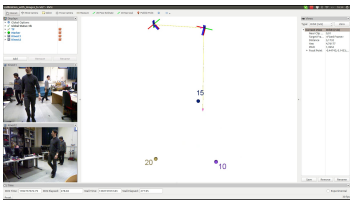
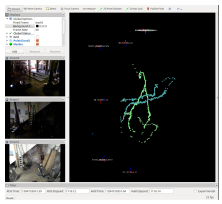


Figure 14: People tracking results: tracks reported with different colors on the left, detections at the current frame on the right.



(a) Three people tracked with two Kinect v1.



(b) Two people tracked with three Kinect v1 four meters high.

Figure 15: Some tracking results obtained with the OpenPTrack software.

In Figure 15a, the position of three people seen from two Kinect v1 are reported, while, in Figure 15b, we visualize the tracks produced by our algorithm when tracking two people from three Kinect v1 with overlapping field of views. Color images are reported on the left as a reference.

8.1. Quantitative evaluation of calibration refinement

The extrinsic calibration algorithm in OpenPTrack allows to calibrate networks of sensors in few seconds and visualize the calibration progress in real time. In Figure 16, we report the camera poses obtained with the algorithm described in Section 4.2 for a network composed of ten sensors: seven Kinect v1, two SR4500 and a stereo pair.

In Figure 17 and 18, we report another experiment on the calibration refinement method. It can be easily noticed that the detection point clouds are more correctly aligned after the refinement and that all the detections lie on the same plane.

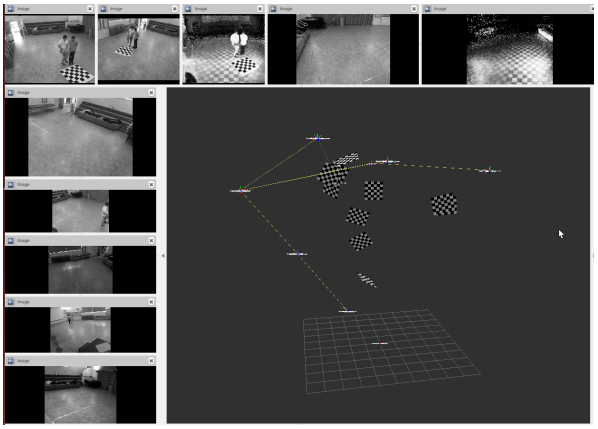
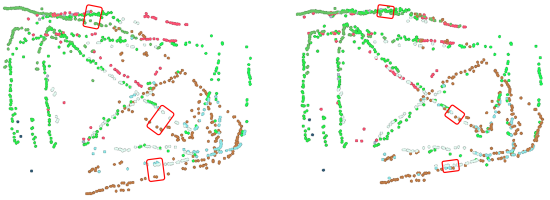


Figure 16: Calibration results with ten sensors: seven Kinect v1, two SR4500 and one stereo pair.



(a) Before (b) After

Figure 17: People detections from all sensors in Figure 16 (a) before and (b) after the calibration refinement procedure.

(a) Before (b) After

Figure 18: People detections from all sensors in Figure 16 (a) before and (b) after the calibration refinement procedure.

For a quantitative evaluation of refinement results, we measured the average distance between all detection clouds and the average cloud (introduced in algorithm 1). As reported in Figure 18, this error decreased of two to four times by applying the refinement. This test is reported for both the detection trajectories in Figure 7 (A) and in Figure 17 (B).

8.2. *Evaluation of people tracking with different sensors*

As reported in Section 3, OpenPTrack allows to use different types of sensors. For comparing them in terms of tracking capabilities, we performed tracking with Kinect v1, Kinect v2 and SR4500 on three videos[14] with different challenges: *Crowd* features seven people randomly moving in the range $1-9m$; in *Distance*, two people linearly move up to $10m$ of distance from the sensor; finally, in *Occlusions*, two people walk in the range $1-8m$ while being heavily occluded by background objects.

[14]These videos are not available for download at the moment of writing, but they could be released at this page afterwards: `http://www.dei.unipd.it/~munaro/research.html`.

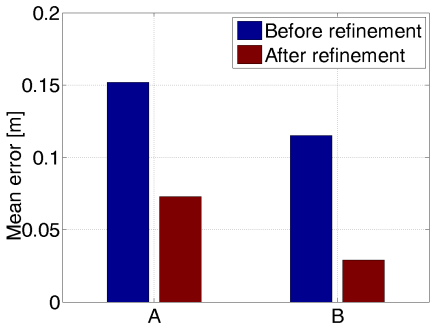


Figure 19: Mean error from average detection cloud before and after calibration refinement computed for the detection trajectories in Figure 7 (A) and in Figure 17 (B). They refer to the network illustrated in Figure 16.

Table 2: Tracking evaluation for *Crowd* video.

	MOTA	**MOTP**	**FP**	**FN**	**ID Sw.**
Kinect v2	60.98%	79.67%	9.3%	29.5%	9
Kinect v1	44.21%	75.53%	1.0%	54.2%	18
SR4500	49.48%	76.67%	8.7%	40.4%	25
Kinect v2 [1]	61.56%	80.45%	7.2%	30.7%	22
Kinect v1 [1]	41.22%	75.22%	2.5%	55.6%	27

Table 3: Tracking evaluation for *Distance* video.

	MOTA	**MOTP**	**FP**	**FN**	**ID Sw.**
Kinect v2	89.67%	73.86%	3.2%	7.1%	0
Kinect v1	54.54%	71.14%	1.6%	43.5%	4
SR4500	64.10%	78.50%	1.2%	34.5%	1
Kinect v2 [1]	89.20%	75.27%	2.3%	8.5%	0
Kinect v1 [1]	40.96%	71.94%	2.7%	56.0%	5

Quantitative tracking results on these videos are reported in Table 2, 3 and 4 where the CLEAR MOT metrics [33] is used to evaluate accuracy and precision. We also reported the results obtained with Kinect v1 and v2 by exploiting the method in [1], which uses also color information for data association.

From these data, it can be clearly stated the superiority of Kinect v2 with respect to the other sensors in detecting people since the rate of false negatives is considerably lower. Unlike Kinect v1, Kinect v2 does not present quantization in depth estimation, thus point clouds are better shaped also at a distance. With respect to SR4500, Kinect v2 takes advantage of the higher depth resolution. Since with Kinect v2 people are more continuously detected, they are also better tracked, thus explaining a lower number of identity (ID) switches. The performance of Kinect v1 and SR4500 is similar, with a little advantage for SR4500 thanks to its better precision (and no quantization error) in depth estima-

Table 4: Tracking evaluation for *Occlusions* video.

	MOTA	**MOTP**	**FP**	**FN**	**ID Sw.**
Kinect v2	48.14%	75.41%	6.2%	45.3%	6
Kinect v1	31.98%	72.84%	0.4%	67.1%	6
SR4500	30.42%	72.96%	4.1%	63.9%	11
Kinect v2 [1]	41.78%	75.57%	6.2%	51.8%	4
Kinect v1 [1]	12.20%	64.11%	2.0%	85.5%	6

(a) *Crowd* - K1 (b) *Crowd* - K2

(c) *Distance* - K1 (d) *Distance* - K2

(e) *Occlusions* - K1 (f) *Occlusions* - K2

Figure 20: Sample tracking results with Kinect v1 and v2 for the three test videos. For every track, its ID and the distance from the sensor is reported.

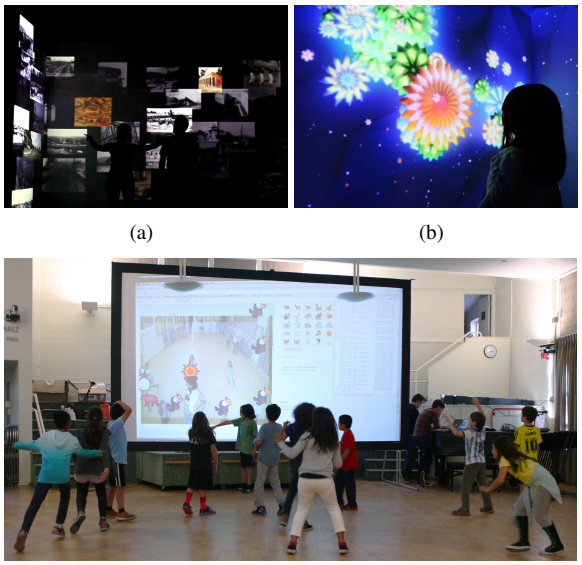
tion.

In Figure 20, we compare screenshots of tracking results obtained with Kinect v1 and v2 on the three test videos. This qualitative comparison confirms that the main difference between Kinect v1 and v2 is in detecting people at far range. As shown in Figure 20d, the high precision in Kinect v2 point clouds allowed to track a person at more than 9.5m and very close to the background.

From the comparison with [1], it can be noticed that OpenPTrack behaves in a similar way, even if not exploiting color information for data association. Sometimes, it also detects more people, probably because, in OpenPTrack, people detection is performed in the ground plane reference frame instead than in the sensor reference frame, thus improving detection performance when the sensor is tilted.

9. Applications

The OpenPTrack library has also been validated in three real world scenarios. The first was in March 2014 in Los Angeles, where three Kinect v1 and three SR4500 were mounted on top of a 8x6x4 meters indoor



(a) (b)

(c)

Figure 21: Applications in (a) culture, (b) art and (c) education based on OpenPTrack tracking software (images from http://openptrack.org).

pavilion and used for cooperative people tracking by the UCLA Interpretive Media Laboratory. The tracking output drove an interactive digital mural application showing images of the Los Angeles State Historic Park, which were selected and manipulated based on people's real-time position within the pavilion (Figure 21a). OpenPTrack calibration procedure enabled to easily calibrate this complex network in one minute with real time visualization of the calibration results and without the need for storing data to be processed offline. For this installation, one Kinect v1 and one SR4500 were connected to each of three PCs used for people detection. These PCs were fanless industrial PCs with Intel i7-3612QM @ 2.10GHz CPU. The detection frame rate was of about 30 fps for Kinect and 13 fps for SR5400. However, the SR4500 frame rate was limited by the exposure time chosen in order to have better depth estimates. The same OpenPTrack network was also used for *Whorl*[15](Figure 21b), an artistic installation showing flowers blooming and moving according to people position within the pavilion.

Subsequent installations in May and June 2014 at Indiana University's Rogers Elementary School and the UCLA Lab School tested OpenPTrack for enabling elementary school students to control a science simula-

[15] `http://openptrack.org/2014/10/whorl-made-with-openptrack`.

tion based on their body position and movement (Figure 21c), as part of the NSF-supported Science through Technology Enhanced Play (STEP) research project. The childrens' motion controlled on-screen avatars as they explored an embodied simulation of states of matter. These deployments provided practical experience in parameter adjustment for children (80-100 cm tall), calibration by end-users, and how to best support third-party applications using the data in the future. For the same purpose, in February 2015, OpenPTrack has been extensively tested for one month in the ten sensors configuration shown in Figure 16, which was composed of seven Kinect v1, two SR4500 and one stereo camera.

10. Conclusions

We introduced the main features of OpenPTrack, an open source project for scalable people tracking in heterogeneous networks of 3D sensors. It features a real-time, easy to use, and precise multi-camera calibration process that also exploits people detections for refining camera poses and modules for people detection in color, infrared and disparity images, in order to allow for tracking in every light condition. Furthermore, its light-weight tracking algorithm is able to fuse data coming from many sensors. A strong focus of this project is on the usability of the software by non-expert users since many users of this system are developers of applications based on the tracking output.

The reported experiments showed that a considerable improvement is obtained with the proposed calibration refinement method. We also compared the sensors supported by OpenPTrack in terms of accuracy and precision for people tracking in three different scenarios, showing the supremacy of Microsoft Kinect v2 over the other sensors. Finally, we saw how OpenPTrack is already used in a number of installations that exploit up to ten sensors for estimating motion of adults or kids for arts, education and culture applications.

11. Acknowledgements

The authors would like to thank Jeff Burke, Alexander Horn and Randy Illum for the extensive collaboration in designing and testing OpenPTrack. OpenPTrack has been sponsored by UCLA REMAP and Open Perception. Key collaborators include the University of Padova and Electroland. Portions of the work have been supported by the National Science Foundation (IIS-1323767).

References

[1] M. Munaro, E. Menegatti, Fast RGB-D people tracking for service robots, Journal on Autonomous Robots 37 (2014) 227–242.

[2] D. Holz, L. Iocchi, T. van der Zant, Benchmarking intelligent service robots through scientific competitions: The Robocup@Home approach, in: AAAI Spring Symposium: Designing Intelligent Robots, 2013.

[3] M. Munaro, C. Lewis, D. Chambers, P. Hvass, E. Menegatti, RGB-D human detection and tracking for industrial environments, in: In Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13), Padua, Italy, 2014.

[4] R. B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), in: International Conference on Robotics and Automation (ICRA) 2011, Shanghai, China, 2011, pp. 1–4.

[5] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source Robot Operating System, in: International Conference on Robotics and Automation (ICRA), 2009.

[7] M. Luber, L. Spinello, K. O. Arras, People tracking in RGB-D data with on-line boosted target models, in: International Conference On Intelligent Robots and Systems (IROS) 2011, 2011, pp. 3844–3849.

[8] D. Mitzel, B. Leibe, Real-time multi-person tracking with detector assisted structure propagation, in: International Conference on Computer Vision (ICCV) Workshops 2011, IEEE, 2011, pp. 974–981.

[9] F. Basso, M. Munaro, S. Michieletto, E. Pagello, E. Menegatti, Fast and robust multi-people tracking from RGB-D data for a mobile robot, in: 12th Intelligent Autonomous Systems Conference (IAS-12), Jeju Island, Korea, 2012, pp. 265–276.

[10] M. Munaro, F. Basso, E. Menegatti, Tracking people within groups with RGB-D data, in: Proc. of the International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, 2012, pp. 2101–2107.

[11] O. H. Jafari, D. Mitzel, B. Leibe, Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras, in: In Proceedings of IEEE International Conference on Robotics and Automation (ICRA'14), Hong Kong, China, 2014.

[12] J. M. Diaz, J.-B. Hayet, Color and motion-based particle filter target tracking in a network of overlapping cameras with multithreading and gpgpu, Acta Universitaria 23 (1).

[13] R. Vezzani, D. Baltieri, R. Cucchiara, Pathnodes integration of standalone particle filters for people tracking on distributed surveillance systems., in: P. Foggia, C. Sansone, M. Vento (Eds.), ICIAP, Vol. 5716 of Lecture Notes in Computer Science, Springer, 2009, pp. 404–413.

[14] G. Lian, J.-H. Lai, Y. Gao, People consistent labeling between uncalibrated cameras without planar ground assumption., in: ICIP, IEEE, 2010, pp. 733–736.

[15] D. Arsic, E. Hristov, N. H. Lehment, B. Hornler, B. Schuller, G. Rigoll, Applying multi layer homography for multi camera person tracking., in: ICDSC, IEEE, 2008, pp. 1–9.

[16] R. Eshel, Y. Moses, Homography based multiple camera detection and tracking of people in a dense crowd., in: CVPR, IEEE Computer Society, 2008.

[17] S. M. Khan, M. Shah, A multiview approach to tracking people in crowded scenes using a planar homography constraint., in: A. Leonardis, H. Bischof, A. Pinz (Eds.), ECCV (4), Vol. 3954 of Lecture Notes in Computer Science, Springer, 2006, pp. 133–146.

[18] E. Auvinet, J. Meunier, F. Multon, Multiple depth cameras calibration and body volume reconstruction for gait analysis, in: Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on, 2012, pp. 478–483.

[19] Q. Le, A. Ng, Joint calibration of multiple sensors, in: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, 2009, pp. 3651–3658.

[20] B. Micusik, T. Pajdla, Simultaneous surveillance camera calibration and foot-head homology estimation from human detections, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1562–1569.

[21] J. Liu, R. T. Collins, Y. Liu, Surveillance camera autocalibration based on pedestrian height distributions, in: Proceedings of the British Machine Vision Conference, Citeseer, 2011, p. 144.

[22] C. D. Mutto, P. Zanuttigh, G. M. Cortelazzo, Time-of-Flight Cameras and Microsoft Kinect, SpringerBriefs in Electrical and Computer Engineering, 2012.

[23] S. Zennaro, M. Munaro, S. Milani, A. Bernardi, P. Zanuttigh, E. Menegatti, Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications, in: Proceedings of the 2015 IEEE International Conference on Multimedia and Expo (ICME'15), Turin, Italy, 2015, pp. 1–6.

[24] F. Basso, R. Levorato, E. Menegatti, Online calibration for networks of cameras and depth sensors, in: In 12th Workshop on Non-classical Cameras, Camera Networks and Omnidirectional Vision (OMNIVIS 2014), Hong Kong, China, 2014.

[25] S. Agarwal, K. Mierle, et al., Ceres solver, `http://ceres-solver.org`.

[26] F. Basso, A. Pretto, E. Menegatti, Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple, in: In Proceedings of IEEE International Conference on Robotics and Automation (ICRA'14), Hong Kong, China, 2014.

[27] P. Besl, N. D. McKay, A method for registration of 3-d shapes, Pattern Analysis and Machine Intelligence, IEEE Transactions on 14 (2) (1992) 239–256. doi:10.1109/34.121791.

[28] M. Munaro, F. Basso, S. Michieletto, E. Pagello, E. Menegatti, A software architecture for RGB-D people tracking based on ros framework for a mobile robot, in: Frontiers of Intelligent Autonomous Systems, Vol. 466, Springer, 2013, pp. 53–68.

[29] A. Trevor, S. Gedikli, R. Rusu, H. Christensen, Efficient organized point cloud segmentation with connected components, in: 3rd Workshop on Semantic Perception Mapping and Exploration (SPME), Karlsruhe, Germany, 2013.

[30] N. Bellotto, H. Hu, Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters, Autonomous Robots 28 (2010) 425–438.

[31] C. L. Naberezny Azevedo, The multivariate normal distribution [online], `http://www.ime.unicamp.br/~cnaber/mvnprop.pdf`.

[32] M. Munaro, S. Ghidoni, D. Tartaro Dizmen, E. Menegatti, A feature-based approach to people re-identification using skeleton keypoints, in: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong (China), Elsevier, 2014.

[33] K. Bernardin, R. Stiefelhagen, Evaluating multiple object tracking performance: the CLEAR MOT metrics, Journal of Image Video Processing 2008 (2008) 1:1–1:10.