UNIVERSITÀ DEGLI STUDI DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

SCUOLA DI DOTTORATO DI RICERCA IN: Ingegneria dell'Informazione

INDIRIZZO: Scienza e Tecnologia dell'Informazione

CICLO: XXX

REAL-TIME RGB-DEPTH PERCEPTION OF HUMANS FOR ROBOTS AND CAMERA NETWORKS

Direttore della Scuola: *Ch.mo Prof. Andrea Neviani*
Supervisore: *Ch.mo Prof. Emanuele Menegatti*

Dottorando:
Marco Carraro

to Silvia

Abstract

This thesis deals with robot and camera network perception using RGB-Depth data. The goal is to provide efficient and robust algorithms for interacting with humans. For this reason, a special care has been devoted to design algorithms which can run in real-time on consumer computers and embedded cards.

The main contribution of this thesis is the 3D body pose estimation of the human body. We propose two novel algorithms which take advantage of the data stream of a RGB-D camera network outperforming the state-of-the-art performance in both single-view and multi-view tests. While the first algorithm works on point cloud data which is feasible also with no external light, the second one performs better, since it deals with multiple persons with negligible overhead and does not rely on the synchronization between the different cameras in the network.

The second contribution regards long-term people re-identification in camera networks. This is particularly challenging since we cannot rely on appearance cues, in order to be able to re-identify people also in different days. We address this problem by proposing a face-recognition framework based on a Convolutional Neural Network and a Bayes inference system to re-assign the correct ID and person name to each new track.

The third contribution is about Ambient Assisted Living. We propose a prototype of an assistive robot which periodically patrols a known environment, reporting unusual events as people fallen on the ground. To this end, we developed a fast and robust approach which can work also in dimmer scenes and is validated using a new publicly-available RGB-D dataset recorded on-board of our open-source robot prototype.

As a further contribution of this work, in order to boost the research on this topics and to provide the best benefit to the robotics and computer vision community, we released under open-source licenses most of the software implementations of the novel algorithms described in this work.

Sommario

Questa tesi tratta di percezione per robot autonomi e per reti di telecamere da dati RGB-Depth. L'obiettivo è quello di fornire algoritmi robusti ed efficienti per l'interazione con le persone. Per questa ragione, una particolare attenzione è stata dedicata allo sviluppo di soluzioni efficienti che possano essere eseguite in tempo reale su computer e schede grafiche consumer.

Il contributo principale di questo lavoro riguarda la stima automatica della posa 3D del corpo delle persone presenti in una scena. Vengono proposti due algoritmi che sfruttano lo stream di dati RGB-Depth da una rete di telecamere andando a migliorare lo stato dell'arte sia considerando dati da singola telecamera che usando tutte le telecamere disponibili. Il secondo algoritmo ottiene risultati migliori in quanto riesce a stimare la posa di tutte le persone nella scena con overhead trascurabile e non richiede sincronizzazione tra i vari nodi della rete. Tuttavia, il primo metodo utilizza solamente nuvole di punti che sono disponibili anche in ambiente con poca luce nei quali il secondo algoritmo non raggiungerebbe gli stessi risultati.

Il secondo contributo riguarda la re-identificazione di persone a lungo termine in reti di telecamere. Questo problema è particolarmente difficile in quanto non si può contare su feature di colore o che considerino i vestiti di ogni persona, in quanto si vuole che il riconoscimento funzioni anche a distanza di giorni. Viene proposto un framework che sfrutta il riconoscimento facciale utilizzando una Convolutional Neural Network e un sistema di classificazione Bayesiano. In questo modo, ogni qual volta viene generata una nuova traccia dal sistema di people tracking, la faccia della persona viene analizzata e, in caso di match, il vecchio ID viene riassegnato.

Il terzo contributo riguarda l'Ambient Assisted Living. Abbiamo proposto e implementato un robot di assistenza che ha il compito di sorvegliare periodicamente un ambiente conosciuto, riportando eventi non usuali come la presenza di persone a terra. A questo fine, abbiamo sviluppato un approccio veloce e robusto che funziona anche in assenza di luce ed è stato validato usando un nuovo dataset RGB-Depth registrato a bordo robot.

Con l'obiettivo di avanzare la ricerca in questi campi e per fornire il maggior beneficio possibile alle community di robotica e computer vision, come contributo aggiuntivo di questo lavoro, abbiamo rilasciato, con licenze open-source, la maggior parte delle implementazioni software degli algoritmi descritti in questo lavoro.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Perception provides an essential feedback for autonomous robots and artificial intelligences to interact with the environment. A machine which is not able to *see* what is happening around itself, is also not able to interact safely with the environment and to make intelligent decisions. Moreover, perception is a very important feedback to recover from the inevitable failures an autonomous system will encounter when it has to interact with the real-world.

The goal of this thesis is to provide efficient algorithms to solve or to strengthen the perception of autonomous robots and camera networks when humans are involved. Designing algorithms that allow interaction with humans is particularly challenging, since the focus is not just on the quality of the outcome, but also on the achievable framerate.

Let us consider an assembly line of the future, where robots and humans will cooperate without security fences. It is of paramount importance that the robot will be able to recognize humans and the movements they are making in real-time. In this way, it will be able to take fast decisions and movements to prevent collisions with them. For similar reasons, also Human-Computer-Interaction (HCI) scenarios can benefit from such algorithms. Indeed, in order to enable an effective interaction, only a short-time response from the system can be tolerated. For those reasons, one of the main focus of this thesis is to solve different real-world perception problems in which humans are involved, while providing not just effective solutions, but also efficient ones.

The visual data considered as input for the algorithms described are composed of the color image, also called RGB image, and the depth image. The depth image encodes the distance with respect to the sensor of each real world point projected to the camera image plane. Such data enables to disambiguate many real-world situations that cannot be addressed using just color information. Indeed, while a color image is very informative, it sensibly changes with the scene illumination (e.g. let us think about a picture shot during the night). Moreover, some false positives are impossible to be detected using just 2D data, since there is no way to understand whether an information is real or just printed. This situation is depicted in Figure 2.1, where a person is holding a picture depicting a person. Depth information gives us enough information to discriminate such false positives and, moreover, when using active sensor to compute it, it is reliable also in presence of little or no light in the scene. To deal with such problems, our algorithms take advantage of RGB-D data. More formally, an input frame F, is composed of a synchronized couple of images, the color image (RGB) and the depth (D).

Since the topics we addressed are very important for the robotics and computer vision community, we released most of the implemented algorithms in an open-source Human-Computer-Interaction library: OpenPTrack[1].

The remainder of the thesis and the problems addressed are summarized as follows:

- Chapter 2 briefly describes how to compute the depth data. Moreover, we present an efficient system able to generate 3D data structures (i.e. Point Cloud) when using a Kinect v2 together with a GPU-enabled computer or embedded-card.

- Chapters 3 and 4 describe two novel algorithms we developed for solving the 3D Body Pose Estimation (3D-BPE) problem in RGB-D camera networks. The target is to find the 3D body joint locations of the people in the scene, without the aid of any marker. Chapter 3 addresses this problem by fusing information at the depth level. In this way, it is possible to create a synthetic frontal-view for the depth image from which a state-of-the-art single-view 3D-BPE is applied. On the other hand, Chapter 4 takes advantage of the recent advances using Convolutional Neural Networks (CNNs) to compute the single-view 3D-BPE from each camera and fuse them by means of Kalman filtering. While the contribution presented in Chapter 3 works also with no light, since it rely only on the point cloud, Chapter 4 outperforms it with respect to two aspects. Indeed, it can compute the poses of multiple people in the scene with negligible overhead and the synchronization between the different cameras is no longer required. In this way, each user can build the different network nodes following his needs and possibilities.

- Chapter 5 addresses the problem of the long-term re-identification of people in RGB-D camera networks. Nowadays, several state-of-the-art tracking libraries

[1]https://github.com/openptrack/open_ptrack_v2

are able to detect and track people smoothly. However, in presence of heavy occlusions or when a person re-enters the Field-of-View (FoV) of the network, those system are not able to recognize persons, initializing new tracks for them. In order to overcome this problem, we propose a system that exploits a CNN-based solution for face recognition and a Bayesian-inference classifier to recognize whether a person has already been seen by the network. Whenever a person is classified as already seen, the previous track ID is assigned to its track.

- Chapter 6 describes a contribution in the field of Ambient Assisted Living. Undetected falls are one of the main risk of early deaths for elder people living lonely at home. This Chapter shows a novel contribution to detect fallen people from a mobile robotic platform that is in charge of patrolling the home.

- Finally, Chapter 7 draws the conclusions of this thesis.

Moreover, Appendices A and B present additional contributions. In particular, Appendix A describes the open-source prototype for Ambient Assisted Living (AAL) we used to validate the algorithm proposed in Chapter 6. On the other hand, Appendix B presents a RGB-D dataset we recorded and publicly released about detection of fallen persons. The dataset is composed of several frames taken from two different environments on-board our mobile assistive robot.

1.1 Publications

The works we will describe in this thesis have been presented in the following International Conferences and Journals publications:

- **M. Carraro**, M. Munaro, J. Burke and E. Menegatti, Real-time marker-less multi-person 3D pose estimation in RGB-Depth camera networks, *arXiv Preprint, arxiv:1710.06235*, 2017. [15]

- Y. Zhao, **M. Carraro**, M. Munaro and E. Menegatti, Fast Multiple Object Tracking in RGB-D Camera Networks, in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, IEEE*, 2017. [130]

- M Antonello, **M Carraro**, M Pierobon and E Menegatti, Fast and Robust Detection of Fallen People from a Mobile Robot, in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, IEEE*, 2017. [3]

- R. Illum, **M. Carraro**, E. Menegatti and J. Burke, OpenPTrack: Real-time, Multi-camera Computer Vision Infrastructure for Artists, In *Workshop on Towards an artist-in-the-lab framework in conjunction with IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. [47]
- K. Koide, E. Menegatti, **M. Carraro**, M. Munaro and J. Miura, People Tracking and Re-Identification by Face Recognition for RGB-D Camera Networks, In *Mobile Robots (ECMR), 2017 European Conference on*, 2017. [57]
- **M Carraro**, M Munaro and E Menegatti Cost-efficient RGB-D smart camera for people detection and tracking, *Journal of Electronic Imaging 25 (4), 041007-041007* (JEI-2016) [16]
- **M Carraro**, M Munaro and E Menegatti, A powerful and cost-efficient human perception system for camera networks and mobile robotics, *International Conference on Intelligent Autonomous Systems, 485-497 (IAS-14)*, 2016 [17]
- **M Carraro**, M Munaro, A Roitberg and E Menegatti, Improved skeleton estimation by means of depth data fusion from multiple depth cameras, *International Conference on Intelligent Autonomous Systems, 1155-1167 (IAS-14)*, 2016 [18]
- **M. Carraro**, M. Antonello, L. Tonin and E. Menegatti, An Open Source Robotic Platform for Ambient Assisted Living. *AIRO@ AI* IA, 3-18*, 2015 [14]

Chapter 2

RGB and Depth data

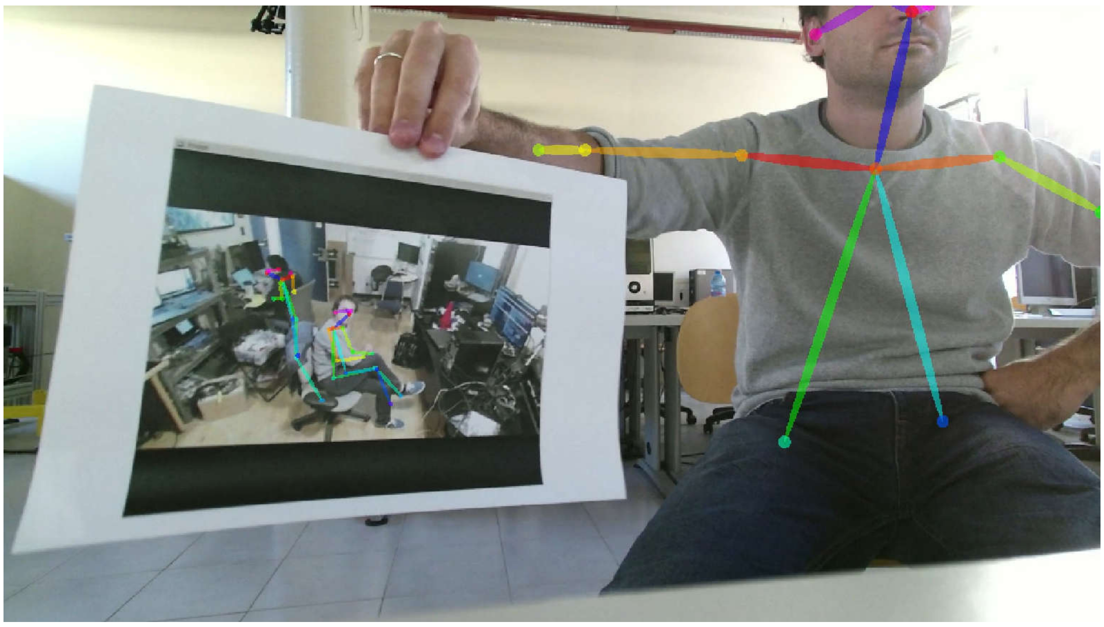


Figure 2.1: An example of the application of a state-of-the-art body pose estimation algorithm [13] on an image. As it can be seen, the algorithm finds proposals also in the picture kept by the real person even if there is no actual person there.

Depth information is required to build the data structures that enables computers and robots to directly reason in 3D. The majority of works in the Computer Vision and Robotics community rely on RGB-only information. While many state-of-the-art Deep Learning techniques [13, 37, 93–95] demonstrated to achieve impressive results using only this type of data, such systems do not work in common robotics situations as when there is absence of information (i.e. images shot during the night with no illumination). Moreover, given the nature of the method used, they generate false positives caused by mirror reflections or pictures held by persons as depicted in Figure 2.1. In order to overcome this problem, the RGB information is often enriched with the depth information; this type of data is called RGB-D. The goal of this chapter is to explain the major techniques to compute the depth information and the sensors we used in our works to compute such data. We also describe a novel contribution (see Section 2.2) regarding the creation of an efficient library to compute point cloud data structures using depth and images from a Kinect v2 sensor.

2.1 RGB-D sensors

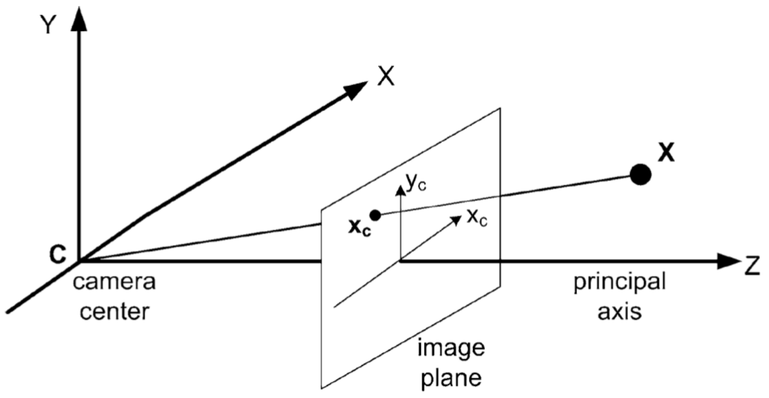


Figure 2.2: Perspective geometry of the pinhole camera model. Image courtesy of [84]

Figure 2.2 depicts the 3D perspective projection of the pinhole camera model. Given a real world point $\mathbf{X}$, this is projected to the pixel $\mathbf{x_c}$ obtained by intersection of the $\mathbf{X}-\mathbf{C}$ ray and the camera image plane. Given the world to camera transformation matrix $[R|\mathbf{t}]$, also known as the extrinsic parameters, and the intrinsics parameters K, which converts from metric units to pixels, Equation 2.1 shows how $\mathbf{X}$ and $\mathbf{x_c}$ are correlated.

$$\lambda \begin{bmatrix} \mathbf{x_c} \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K[R|\mathbf{t}] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

Using Equation 2.1 is possible to *project* a 3D point $\mathbf{X}$ to the image point $\mathbf{x_c}$. To be able to find the depth, we need to solve the inverse problem, i.e. given an image point $\mathbf{x_c}$ find the real 3D world point $\mathbf{X}$ that generated $\mathbf{x_c}$. Equation 2.1 does not give enough constraints to solve this problem, given that all the points which lie on the $\mathbf{X}-\mathbf{C}$ ray

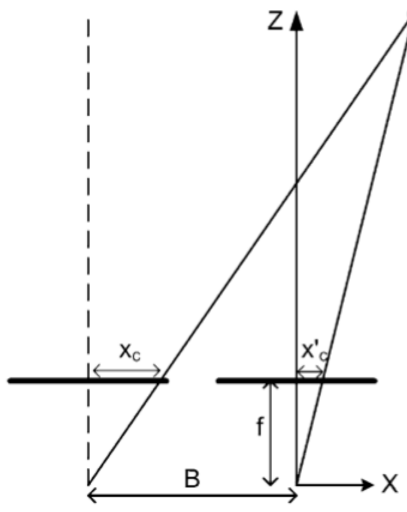


Figure 2.3: Stereo geometry. Image courtesy of [84]

and the image plane would be possible solutions. However, recent advances in Deep Learning using CNNs [59, 109] are demonstrating the effectiveness of such methods to fulfill the lack of mathematical constraints with experience. Such methods provide impressive results, nonetheless, they work on RGB data and they are therefore affected by the same problems discussed before, so they are not considered in the remainder of this chapter.

In order to add a constraint to solve this problem, a passive solution is to use an additional camera. The couple of cameras compose a stereo system and the depth image is calculated starting from the two synchronized RGB images. Figure 2.3 shows how a common world point is correlated to the two images captured by the stereo couple. In particular, the distance Z of the world point is inversely proportional to the disparity d calculated as $d = x_c - x'_c = fB/Z$ where f is the focal length and B is the distance between the cameras (i.e. the baseline). While stereo cameras theoretically solve the problem, they present three disadvantages:

- we need to be sure to find correspondences in the two images which are projected to the same 3D point. It is computationally expensive to find such correspondences;
- when there is no or little texture the number of correspondences are low, resulting in missing depth information;
- the depth estimation error increases quadratically with the inverse of the baseline.

Active sensors are able to overcome the texture problems of the passive sensors by *creating* texture using a projector. There are two main categories of active sensors: (i) structured-light sensors and (ii) Time-of-Flight (ToF) sensors. The structured-light sensors compute the depth by projecting an infrared pattern on the scene and then capturing how this pattern deforms. The most famous sensor which uses this technique is the Microsoft Kinect 360 launched in 2010 and depicted in Figure 2.4. On the other hand, Time-of-flight based sensors compute the depth information by using a matrix of emitters/receivers and measuring the phase shift between an emitted infrared signal and the reflected one. The most famous sensor in this category is the Microsoft Kinect One also known as Microsoft Kinect v2 (see Figure 2.4). Given the importance of such sensors for research purposes, the two Kinect sensors have been thoroughly compared in different state-of-the-art works [100, 127]. Overall, the second generation Kinect performs better in terms of reconstruction quality and error estimation with respect to increasing distances, even presenting the typical ToF problems such as flying pixels generated by sharp edges or vertexes and missing depth information due to infrared absorption of some materials. The majority of the works described in this thesis use the data provided by the Kinect v2.

While active sensors normally perform better than passive ones in indoor environments, they present some major drawbacks. In particular, stereo cameras have to be preferred in outdoor environments and generally when the maximum distance with respect to the sensor has not to be bounded. Indeed, the technology used with active sensors suffers from infrared interference in presence of solar light and it has a maximum working distance.

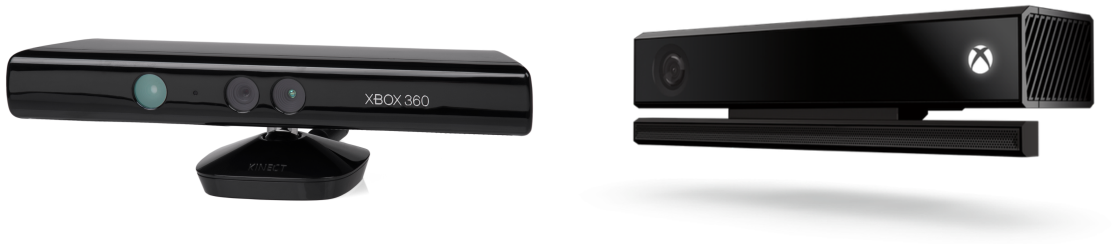


Figure 2.4: The two generation Kinect sensors. On the left the first generation, while on the right the Kinect v2.

Given a synchronized and extrinsically calibrated depth image D and color image I is possible to create a point cloud $P = \{(x, y, z, r, g, b) \in \mathbb{R}^6\}$. This is a data structure commonly used in robotics [18, 69, 70, 74, 97] which encodes the three-dimensionality of the scene. Given the nature of the point cloud creation (see Equation 2.1), this could be efficiently generated using massive parallel architectures as Graphical Processing Units (GPUs). To this end, Section 2.2 describes an efficient library to compute point clouds with a smart camera composed of a Microsoft Kinect v2 and a NVidia Jetson TX1.

2.2 An efficient perception node for mobile robotics

In the last years, human detection and tracking algorithms proved to be useful not only in surveillance, but also in different applications like service robotics, Ambient Assisted Living (AAL), culture and arts installations.

People perception was boosted by the introduction of new low cost RGB-D sensors such as the Microsoft Kinect v1 and v2 (see Figure 2.4). While Kinect v2 performances are better than the one achievable with the first-generation, it is even more computationally demanding of its predecessor, because of the large amount of data of each frame (tens of megabytes per frame at 30 frames per second). To keep the pace with the sensor maximum frame rate, a dedicated graphic card is required by its Linux driver. As a result, this sensor is not likely to be used on mobile robots fitted with embedded computers and requires powerful laptop (or industrial) computers.

In this chapter, we demonstrate the use of an NVidia Jetson TK1 embedded computer as an acquisition and processing unit for the Kinect v2 sensor. This is enabled by the new software library we propose in this work. The complete system is tested and demonstrated by using a ROS (Robot Operating System) [90] perception node for people detection and tracking from RGB-D data. We show that the integrated hardware and software system is well suited for application in service robots and also in distributed perception networks composed by dozens of nodes. The main advantage of the proposed solutions are high data processing throughput and the low power consumption of the Jetson embedded platform. The proposed system is tested and validated in distributed RGB-D camera network which uses OpenPTrack [70] (the open-source people tracking library). In summary, the contribution described in this chapter is twofold:

- We propose a new library which permits the usage of the Kinect v2 with CUDA-capable embedded systems and we demonstrate the validity of this work obtaining suitable frame-rates for real-world applications as people detection and tracking.
- We release this library as open-source[1] together with a ROS bridge [2] to make it

[1]https://github.com/OpenPTrack/libfreenect2
[2]https://github.com/OpenPTrack/kinect2_bridge

work out-of-the-box with the most popular framework for the robotics community.

The remainder of the chapter is organized as follows: Section 2.2.1 reviews the state-of-the-art of RGB-D sensors. Section 2.2.2 explains the features of the new library we developed. In Section 2.2.3, experiments are reported and results are shown in terms of acquisition and people tracking frequency. Finally, Section 2.2.4 draws the conclusions of this work.

2.2.1 State-of-the-art

RGB-D sensors

When dealing with mobile robots or complex surveillance scenarios, two dimensional information is not always sufficient to obtain reliable results of detection and tracking in real time. Furthermore, passive 3D solutions such as stereo cameras require additional processing for computing depth information and they are not able to estimate depth for lowly textured areas. For these reasons, the advent of active and low-cost 3D sensors, such as Microsoft Kinect, significantly improved the research on autonomous mobile robotics and computer vision. The first-generation Kinect is an RGB-D sensor that provides both color and depth data at VGA resolution. Depth is estimated by means of an active triangulation process [54] between an infrared pattern projector and an infrared camera, i.e., the position of each 3D point is the intersection of the optical rays corresponding to a dot of the projector and the one of the considered pixel in the infrared camera.

The Kinect is widely used in computer vision and robotics for Simultaneous Localization and Mapping (SLAM) [27], people detection and tracking [70, 71, 76], short-term and long-term people re-identification [31, 73, 79], AAL [14, 28, 35] and many other applications. Besides the wide usage of this sensor, it has the drawback of not being able to estimate depth information outdoors because the infrared component of the sunlight interferes with the pattern projected by Kinect. Furthermore, the depth estimation error increases quadratically with the distance [6]. To overcome these problems, in late 2013, Microsoft released the second generation of the Kinect sensor. This new RGB-D camera relies on the continuous wave time-of-flight [38] technology to infer depth, that is an array of emitters sends out a modulated signal that travels to the measured point, gets reflected and is received by the CCD of the sensor.

The sensor acquires a 512 x 424 depth map and a 1920 x 1080 color image at 15 to 30 fps depending on the lighting condition, since the sensor exploits an auto-exposure algorithm. Kinect v2 outperforms its predecessor on several aspects. In particular, it works outdoor up to four meters and depth accuracy remains constant while increasing the distance [127]. However, since the data resolution is higher than for Kinect v1, processing Kinect v2 data is computationally more demanding and turns out to be unsuitable for embedded systems.

In the work described in this chapter, in order to overcome this problem, we provided a new Linux driver and ROS wrapper for the Kinect v2 so that they allowed to obtain color and point cloud data at more than 20 fps with the NVidia Jetson, a CUDA-capable embedded system.

While Kinect v2 is directly supported in Microsoft Windows with a free driver and SDK provided by Microsoft, the only driver available in Linux is unofficial and open source and is called *libfreenect2*[3].

In this chapter, we also used NVidia CUDA [80], a scalable library for exploiting the General Purpose GPU (GP-GPU) computing on NVidia GPUs. Our work improved a first working library [122] developed in CUDA. The numerical comparisons between the different versions of these libraries are presented in Section 2.2.3.

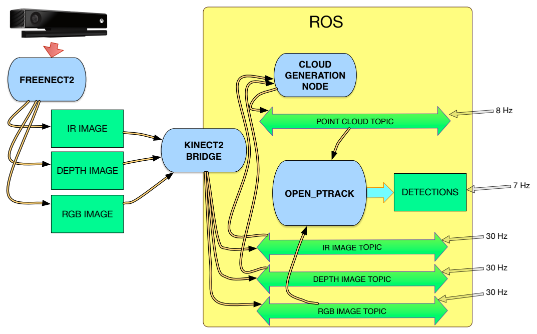
People detection and tracking in camera networks

The ability to autonomously detect and track humans in camera networks is one of the most important issues in robotics and computer vision applications. The problem can be split into two different sub-problems: (1) perform people detection and tracking within a view of a single camera and (2) maintain the same ID for the same person seen by different cameras [117].

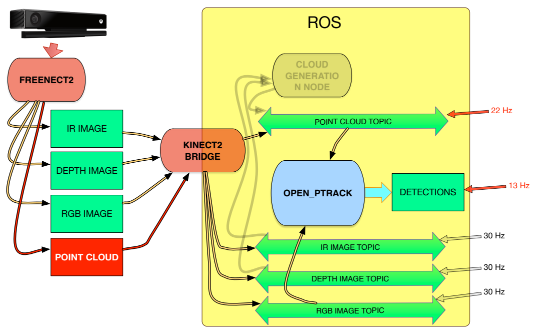
For solving (1), a wide set of works in literature relies on RGB data alone [53, 112], while, recently, new methods were developed for using RGB-D data to perform this task [49, 69, 70]. The problem of associating the correct ID among different cameras (2) is often solved based on the knowledge of camera poses and by exploiting features extracted from the person motion and appearance. OpenPTrack is an open source software for multi-camera calibration and people tracking in RGB-D camera networks. It allows scalable, robust and real-time person tracking using affordable off-the-shelf components, such as Kinect v2, and an open source codebase. It constitutes a powerful tool for enabling interactive experiences for education, arts and culture, but it is also exploited for guaranteeing people safety in industrial environments [75].

The OpenPTrack nodes which use the Kinect v2 are usually equipped with a powerful computer with a dedicated GPU because the sensor is eager of performance. The

[3]`https://github.com/OpenKinect/libfreenect2`



(a) The state-of-the-art data flow



(b) The proposed data flow

Figure 2.5: High level representations of the two data flows. On the left, the state-of-the-art data flow of the OpenPTrack system using the Kinect v2 and the Jetson embedded system. The 3D point cloud is computed outside the *libfreenect2* library by the ROS nodelet `cloud_generation_node`. On the right, the proposed data flow of the same system. The 3D point cloud is now directly computed within the new version of *libfreenect2* and streamed by the new version of *kinect2_bridge*, thus the external nodelet is no longer required.

use of these computers causes space problems, high costs and high power consumption. Moreover, the OpenPTrack network can potentially be made of dozens of nodes, thus amplifying these problems. Therefore, the use of embedded systems as the NVidia Jetson can fix these issues, allowing the building of large networks.

2.2.2 Methodology

Our objective is to acquire data from the Kinect v2 sensor with the NVidia Jetson TK1 at high frame rate and integrate the camera into an OpenPTrack network to perform people detection and tracking.

The state-of-the-art Linux driver for the Kinect v2, *libfreenect2*, is not able to perform the operation needed by OpenPTrack at a framerate suitable for people tracking. For this reason, in order to improve the performance, we use NVidia CUDA, shifting computational burden from the ARM CPU to the GPU of the embedded system.

In Figure 2.5a, the overall state-of-the-art system needed to perform people detection with Kinect v2 is shown. At first, the Linux driver for Kinect v2, *libfreenect2*, is used to obtain the raw data from the sensor. OpenPTrack is based on the ROS middleware, thus, in order to interface the Kinect camera with this people tracking library, we need a ROS wrapper of the *libfreenect2* driver. This wrapper is implemented in the *kinect2_bridge* ROS package, that reads sensory data obtained from the driver and streams them to ROS topics whenever a ROS node requests them. In this work, we adapt also this wrapper to be compatible with the proposed version of *libfreenect2*. The algorithm exploited by the standard version of the driver for computing depth image, infrared (IR) image and point cloud is reported in Algorithm 1 at lines 0-8, while lines 8-18 show the proposed one. The same algorithm is also illustrated in Figure 2.6.

The RGB information that comes from the sensor does not need any additional computation, so we store it directly without passing it to the GPU. The operations needed to transform the raw depth and infrared data in the final data needed by the tracking library are all pixel-wise or consist of operations on the neighborhood of each pixel. These types of functions are implementable in CUDA, thus lowering the final computational complexity from $O(N)$ to $O(1)$, with N number of pixels. Furthermore, in our approach, the generation of the point cloud from IR and depth data is not computed any more by the ROS wrapper, but directly within the driver, thus shifting computational burden from the ARM CPU to the GPU of the embedded system. Each parallel function we implemented requires as input the number of CUDA threads that will concurrently operate. Since the dimension of the depth and infrared images is 512x424, we designed a grid of 512 threads per block with $\left\lfloor \frac{512*424+(512-1)}{512} \right\rfloor = 424$ blocks [99].

OpenPTrack, the library we use for performing people detection, requires as input from the Kinect v2 a point cloud filled with 3D points *colored* with the corresponding intensity obtained from the infrared image. Here, the infrared is preferred to the RGB because the former is constant also in the dark. However, to help the people detection module, an intensity rescaling operation has to be performed on the intensity image in order to improve its contrast, thus helping people detection. Also this computation is performed in our version of the *libfreenect2* driver by exploiting the parallelization achievable with CUDA. In the next sections, we detail the important steps developed in this work.

Memory management

When exploiting GPU processing, the typical bottleneck is the overhead due to the data transfer between the central memory and the GPU memory [98]. To prevent these passages to affect the overall performance of our algorithm, we pre-allocate in the GPU memory the exact space needed by the infrared and depth images and the point cloud. This way, we avoid new allocations whenever a new frame is acquired by the sensor, thus making the GPU only overwrite the previous frame information.

The memory passages performed by our algorithm consist of the copy of the input data from CPU to GPU (only IR and depth raw images) and of those needed to transfer

Algorithm 1 The point cloud generation algorithm implemented in libfreenect2 and the efficient one

INPUT: a frame $F = (I, D, K)$ where I is the raw infrared image, D is the raw depth image and M is the camera calibration matrix
OUTPUT: The final infrared image $\widehat{I}$, the final depth image $\widehat{D}$ and the point cloud P

```
 1: procedure POINT_CLOUD_GENERATION($F = (I, D, K)$)
 2:     $P \leftarrow \emptyset$
 3:     for each pixel $p \in D$ and the correspondent $\widehat{p} \in \widehat{D}$ do
 4:         $\widehat{p} \leftarrow$ computeDepth($p$)
 5:     for each pixel $p \in I$ and the correspondent $\widehat{p} \in \widehat{I}$ do
 6:         $\widehat{p} \leftarrow$ computeIR($p$)
 7:     for each pixel $p_d \in \widehat{D}$ and the correspondent pixel $p_i \in \widehat{I}$ do
 8:         $P \leftarrow P \cup$ compute3DPoint($p_d$, $p_i$, $M$)
 9: procedure EFFICIENT_POINT_CLOUD_GENERATION($F = (I, D, K)$)
10:     $P \leftarrow \emptyset$
11:     $Image\_size \leftarrow 512 * 424$
12:     $Block\_size \leftarrow 512$
13:     $Grid\_size \leftarrow \left\lfloor \frac{Image\_size + (Block\_size - 1)}{Block\_size} \right\rfloor$
14:     memoryCopyFromCPUToGPU($D$,$I$)
15:     $\widehat{D} \leftarrow$ computeDepth$\langle\langle\langle Grid\_size, Block\_size \rangle\rangle\rangle$($D$)
16:     $\widehat{I} \leftarrow$ computeIR$\langle\langle\langle Grid\_size, Block\_size \rangle\rangle\rangle$($I$)
17:     $P \leftarrow$ computePointCloud$\langle\langle\langle Grid\_size, Block\_size \rangle\rangle\rangle$($\widehat{D}$,$\widehat{I}$,$K$)
18:     memoryCopyFromGPUToCPU($\widehat{D}$,$\widehat{I}$,$P$)
```

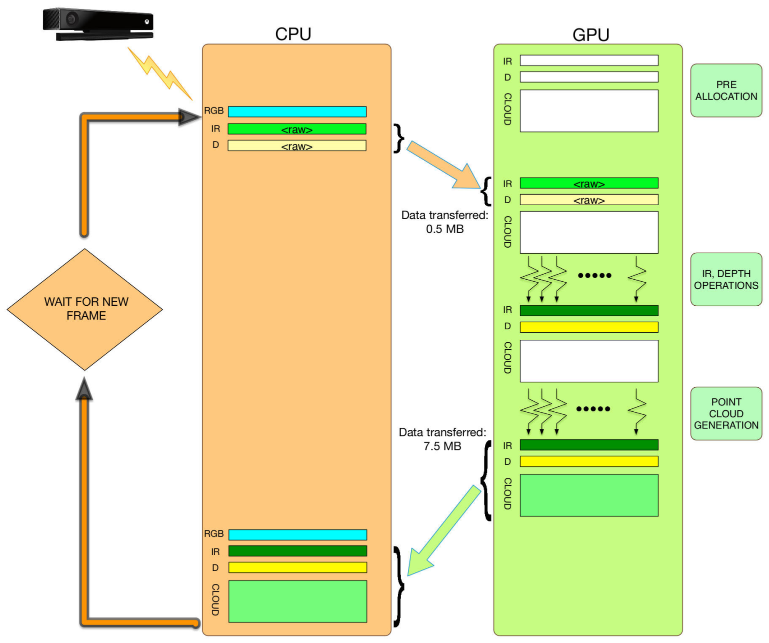


Figure 2.6: The processing flow performed by our library for each single frame. At start-up, the space for the needed data is allocated once in the GPU memory, then, for each frame, the data coming from the CPU memory are processed to achieve the final data. The data transferred from CPU to GPU consist of the raw IR and depth data, while the data transferred from GPU to CPU consists of the final IR, depth and point cloud obtained after the parallel computations have been made.

the output data from GPU to CPU central memory at the end of GPU processing. Figure 2.6 highlights the memory transfers performed by our algorithm. In particular, the memory transfers are less than 500 KB for the first transfer (from CPU to GPU) and the same quantity plus 7 MB (the point cloud) for the transfer-back. The maximum data transfer required by the application is then $7.5 * 30 = 225$MB/s which is about the 1.51% of the total Jetson GPU bandwidth and the 0.067% of a NVidia Geforce GTX Titan Black total GPU bandwidth.

Point Cloud generation

A 3D point cloud is the typical input of several algorithms in 3D computer and robot vision [1, 44]. This data structure is needed also by OpenPTrack to perform people detection while being robust to light changes. This data type is built from three pieces of information: the depth map, the infrared image and the intrinsic parameters of the sensor. The point cloud is computed at each new frame after that the depth data have been processed and become available. Given the equations of 3D perspective

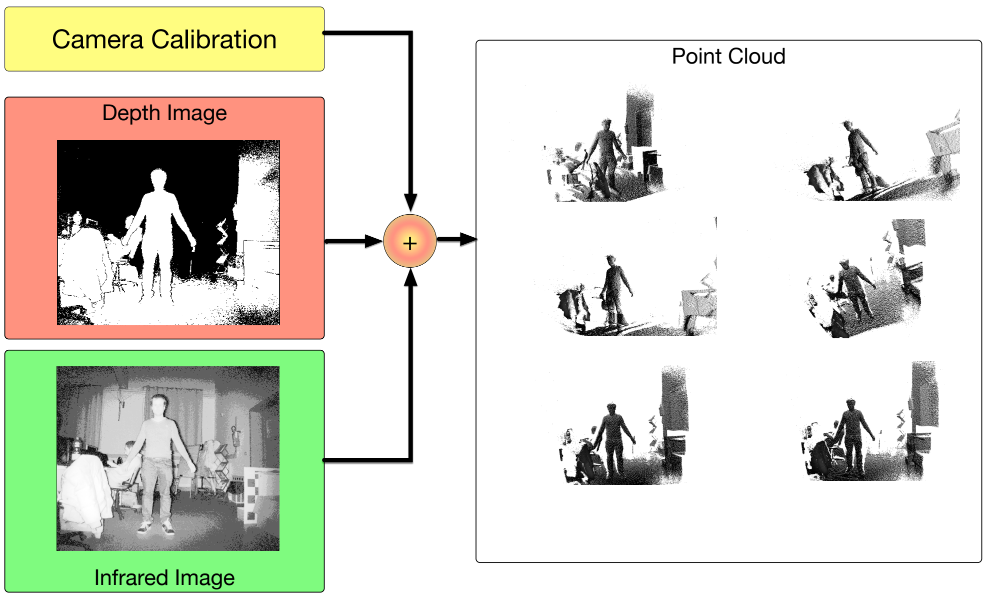


Figure 2.7: The infrared point clouds are obtained from the depth and infrared images and by exploiting the intrinsic parameters of the sensor. Example of six views of the resulting cloud are reported on the right.

projection (simplified from Equation 2.1):

$$\begin{bmatrix} x \\ y \\ d \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \Rightarrow \begin{cases} X = \frac{(x-c_x)d}{f_x} \\ Y = \frac{(y-c_y)d}{f_y} \\ Z = d \end{cases} \tag{2.2}$$

where (x, y) are the coordinates of a pixel in the depth image, d is the measured depth, f_x, f_y, c_x, c_y are the intrinsic parameters of the sensor and represent the focal lengths and the optical centers of the camera, we can obtain (X,Y,Z), the 3D coordinates of the correspondent point in the point cloud.

For what concerns color information, in order to maintain the same structure as for point cloud colored with RGB information, we consider all the three R, G and B channels and fill them with i, that is the infrared intensity of the pixel (x, y), in the IR image. Indeed, people detection performed on infrared information is more robust to changes in visible light. We fill the three R, G and B channels with the same i values in order to have the same algorithms working both on real RGB-colored point clouds and intensity-colored clouds. It is worth noting that, the additional computational burden due to the filling and use of three identical intensity channels is actually negligible. The space in memory allocated for the cloud is then filled in with $P = (X,Y,Z,R,G,B)$ points. Once the point cloud is filled, it is transferred back to the CPU (bottom part of

libraries used	Point Cloud topic framerate (Hz)	OpenPTrack detection framerate (Hz)
ours with the Jetson TX1	**30**	**23**
ours with the Jetson TK1	**22**	**13.7**
[122]	8	7

Table 2.1: Frame rate comparison with a Jetson TK1 embedded system and a Jetson TX1 embedded system.

libraries used	Point Cloud topic framerate (Hz)	OpenPTrack detection framerate (Hz)
ours	**30**	**25.4**
[122]	30	22

Table 2.2: Frame rate comparison with a high-end laptop.

Figure 2.6). An illustration of this process and some examples of the resulting point cloud are shown in Figure 2.7.

The `kinect2_bridge` wrapper

The `kinect2_bridge` wrapper is an executable which streams data from the `libfreenect2` driver library to the ROS topics whenever these data are requested. We had to adapt this wrapper because our version of the driver directly generates the point cloud that was computed by an external ROS node with the standard versions of the driver and the wrapper (see Figure 2.5). This choice allowed to save time avoiding to allocate, transfer and fill each point cloud.

2.2.3 Experiments

To show the improvements of the proposed system with respect to the state-of-the-art, we compared the outputs of two implementations of our system with a people detection and tracking system presented in scientific literature [122]. Table 2.1 compares the output of the software library proposed in [122] with two implementations of our system which used as processing unit respectively: the NVidia Jetson TK1 and the more recent (and much more powerful) NVidia Jetson TX1. The analysis is performed in terms of the following parameters: numbers of processed point clouds per second and people detection frequency. As reported in Table 2.1, the publishing rate of the point cloud gets to 22 point clouds streamed per second with an improvement of a factor 3. The people detection frame rate doubles with our approach reaching 13.7 frames per second, that enables real-time people tracking with continuous tracks.

The same test has also been performed by substituting the Jetson embedded system with an high-end laptop, with an Intel i5-4210M CPU and a NVidia Quadro K1100M

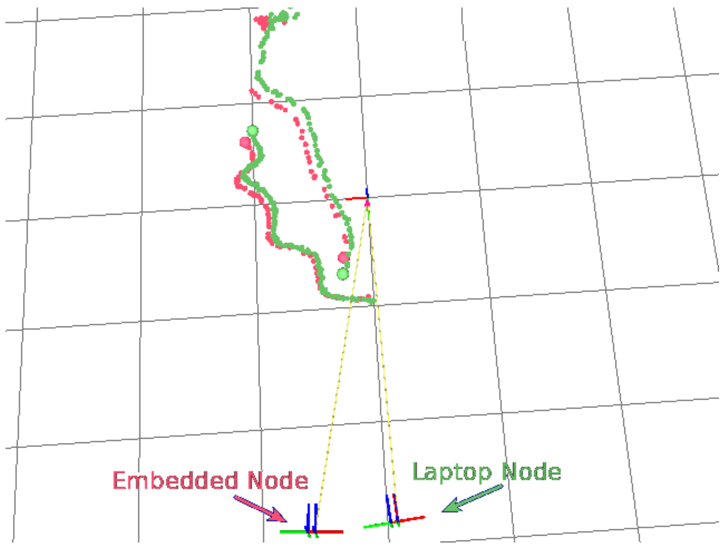


Figure 2.8: Output of OpenPTrack people tracking algorithm. Two people are moving in the environment and are simultaneously observed by the two tracking systems:a Kinect v2 attached to a high-end laptop (green tracks) and a Kinect v2 attached to the embedded system NVidia Jetson TK1 (red tracks). The data of the two tracks are perfectly overlapped, a small offset of the two tracks is introduced afterwards in the picture only for visualization purposes.

GPU. The frame rates reported in Table 2.2 prove that the proposed system increases the overall performance also on a machine with CUDA-enabled graphics card. A qualitative comparison of the output of the people detection module of OpenPTrack run by the embedded system and by the laptop computer has been performed by comparing the tracks of two moving people (see Figure Figure 2.8in which the two people are correctly detected by both nodes). The track generated by the embedded node (in red) is slightly less dense than the one generated by the laptop (in green) because of the differences in frame rate outlined in Tab 2.1 and 2.2. The tracking output data from the two systems are perfectly overlapped, a small offset of the two tracks is introduced afterwards in the picture only for visualization purposes.

These tests confirm that the hardware and software system proposed creates a compact, powerful and cost-efficient perception system (here tested as a people tracking system) based on a Kinect v2 attached to an embedded computer with CUDA processing capability. The hardware and software system we proposed in this work is of major interest also for the mobile robotics community, because on board of a mobile robot weight and power consumption are an issue. Indeed, processing data on board of a mobile robot with an embedded computer (which can be powered with 12V) instead of a laptop (which needs to be powered with a 220V) while keeping real-time performance is a great advantage. Moreover, the software library we developed outputs all the information needed by state-of-the-art robust navigation algorithms and it is compatible with ROS, the de-facto standard used in the robotics community.

2.2.4 Conclusion

In this chapter, we presented a hardware and software perception system which is powerful and cost-efficient. It has been designed for use in camera networks and mobile robotics. We tested the proposed system with people detection and tracking algorithms. It is composed of an embedded computer (i.e. the NVidia Jetson TK1), an RGB-D camera (i.e. the Kinect v2), and a multi-camera people tracking system (i.e. OpenPTrack, an open-source ROS-based software library).

The main contribution of this chapter is a novel software library to process at high frame-rate the heavy stream of data coming from a Kinect One exploiting the GPU capabilities of the Jetson TK1. A second major contribution is the wrapper which makes the *libfreenect2* library compatible with the ROS middleware. The proposed system was tested performing people detection in a RGB-D camera network using OpenPTrack. The results shows that our system allowed to triple the rate of generation of point clouds and to double the people detection frame rate with respect to the state-of-the-art library [122].

This work is not only fundamental for distributed perception systems composed of many perception units, where each unit has to be as powerful and cheap as possible, but also as perception unit for mobile robots, given its compatibility with ROS.

To provide the best benefit for the computer vision and robotics community, but also for developers of human-computer interaction applications, we released all the code implemented as open source within the OpenPTrack repository[4] [16].

Given the possibility to efficiently generate point cloud data as demonstrated, Chapter 3 will present a contribution about the efficient 3D human body pose estimation using just point cloud information obtained by an RGB-D camera network.

[4]https://github.com/OpenPTrack/libfreenect2/tree/jetson-dev
https://github.com/OpenPTrack/kinect2_bridge/tree/jetson-dev

Chapter 3

Markerless 3D body pose estimation fusing depth information between RGB-D cameras

The capability to segment the body parts or, more generally, to estimate a skeleton of a person in an unsupervised way is fundamental for many applications: from health-care to ambient assisted living, from surveillance to action-recognition and people re-identification. The introduction of affordable RGB-D cameras as the Microsoft Kinect, has given a boost to the research in this area and many marker-less skeleton estimation algorithms were born, as Shotton's human pose recognition [102] and Buys' body pose detection [12]. However, all these systems perform better when the subject is seen frontally with respect to the depth camera, mainly because most of the training examples they were trained with referred to this pose.

In this chapter, we want to overcome this problem when multiple cameras are available, thus generating a virtual depth image of the subject warped in frontal view after having fused the depth information coming from the cameras. Moreover, we propose an improvement to Buys' body pose detector [12], here used for skeleton estimation, by adding a preliminary people detection phase for background removal and performing an alpha-beta tracking on the final skeleton joints. The system has been tested on sequences of two freely moving persons imaged by a network composed of two first-generation Microsoft Kinect sensors.

Summarizing, the contribution of this chapter is two-fold:

- We introduce a novel multi-view method to estimate the skeleton of a person based on the fusion of the 3D information coming from all the sensors in the network and a subsequent warping to a frontal pose;

- We improve the body pose detector in [12] by removing background points from the input depth image with a people detection phase and by adding a joint tracking filter to the output of the detector.

The remainder of the chapter is organized as follows: Section 3.1, reviews the state-of-the-art of both single-camera and multi-camera skeleton tracking algorithms, while Section 3.2 gives an overview of our system. In Section 3.3, we describe the multi-view data fusion part of our system, while in Section 3.4 we describe the skeleton estimation algorithm we used and how we improved it. Finally, Section 3.5 details the experiments we performed and the results we achieved and in Section 3.6 conclusions are drawn.

3.1 Related Work

The skeleton of a person gives important cues on what the person is doing (action-recognition) [96, 115], who is the person viewed (people re-identification) [68, 72, 73], what are his/her intentions (surveillance) [45] and how are his/her health conditions (health-care) [25]. Furthermore, the wide literature on people tracking [69, 75, 76] demonstrates its usefulness for both security applications and human-robot interaction. One of the most important works on skeleton tracking is the one by Shotton et al. [102], which trains a random forest to recognize the body parts of a person with a huge training dataset composed of real and synthetic depth images of people. The classifier, licensed by Microsoft for entertainment applications, achieves good performance and works in real-time. The system is released within the Microsoft Kinect SDK and only works with Windows-based computers. Another work released as open-source within the *Point Cloud Library* [97] is the work of Buys et al. [12], which uses an approach similar to Shotton's. In our work, we use this latter body-part detector, that we also improved by adding a people detection pre-processing phase and an alpha-beta tracking algorithm.

Intelligent surveillance systems rely more and more on camera network cooperation. Indeed, more cameras are able to cover more space and from multiple views, obtaining better 3D shapes of the subject and decreasing the probability of occlusions.

Recent works relies on camera collaboration in network to enhance skeletal estimation. In [52], the skeleton obtained by single RGB images is fused with the skeleton estimated from a 3D model composed with the visual hull technique. The visual hull is used for refining the pose obtained from the single images. In [62], a skeleton is computed for every camera from a single image and then these estimates are projected to 3D and intersected in space. The work by Gao et al. [33] addresses this problem by registering a 3D model to the scanned point cloud obtained by two Kinects. This work is very accurate but unfeasible for real-time purposes given the 6 seconds needed to process each frame. The work of Yeung et al. [125] proposes a solution to the same problem with two Kinects that can be used in real-time. In particular, they uses two orthogonal Kinects and fuse the skeletons obtained from the Microsoft SDK with a constrained optimized framework.

In this work, we exploit the multi-view information at the depth level, leaving the skeleton estimation as the last part of the pipeline. In this way, we are able to obtain better skeletons also when the single ones are potentially noisy or when they have some not tracked joints. Moreover, we minimize the skeleton estimation error by warping the fused data to a frontal view, given that the skeleton estimation is best performed from frontally viewed persons.

3.2 System Overview

Figure 3.1 provides an overview of our system. A network composed of two first-generation Microsoft Kinect is considered, but the extension to a higher number of cameras is straightforward. At each new frame, the Kinects compute the 3D point cloud of the scene and the people detector segments only the points belonging to the persons in the scene. Afterwards, we transform the point clouds to a common reference frame given that the network is calibrated and we fuse the point cloud data after performing a fine registration with the Iterative Closest Point(ICP) algorithm [9]. The multi-view cloud obtained is then rotated and reprojected to a virtual image plane so as to generate a depth map of the persons seen from a frontal view. Then, body parts detection is performed on this virtual depth map and the joint position is computed from the body segmentation and tracked with an alpha-beta tracking filter. The obtained skeleton can then be reprojected to either of the original images. In Sections 3.3 and 3.4, we will better review each step of the proposed method.

3.3 Multi-view data fusion

State-of-the-art body part detectors [12, 102] perform poorly in presence of occlusions. This case often occurs when a person is side-viewed by a camera, and having more cameras in the scene does not ensure that one of them sees the person frontally. For this reason, our system exploits the perception network to perform data fusion and

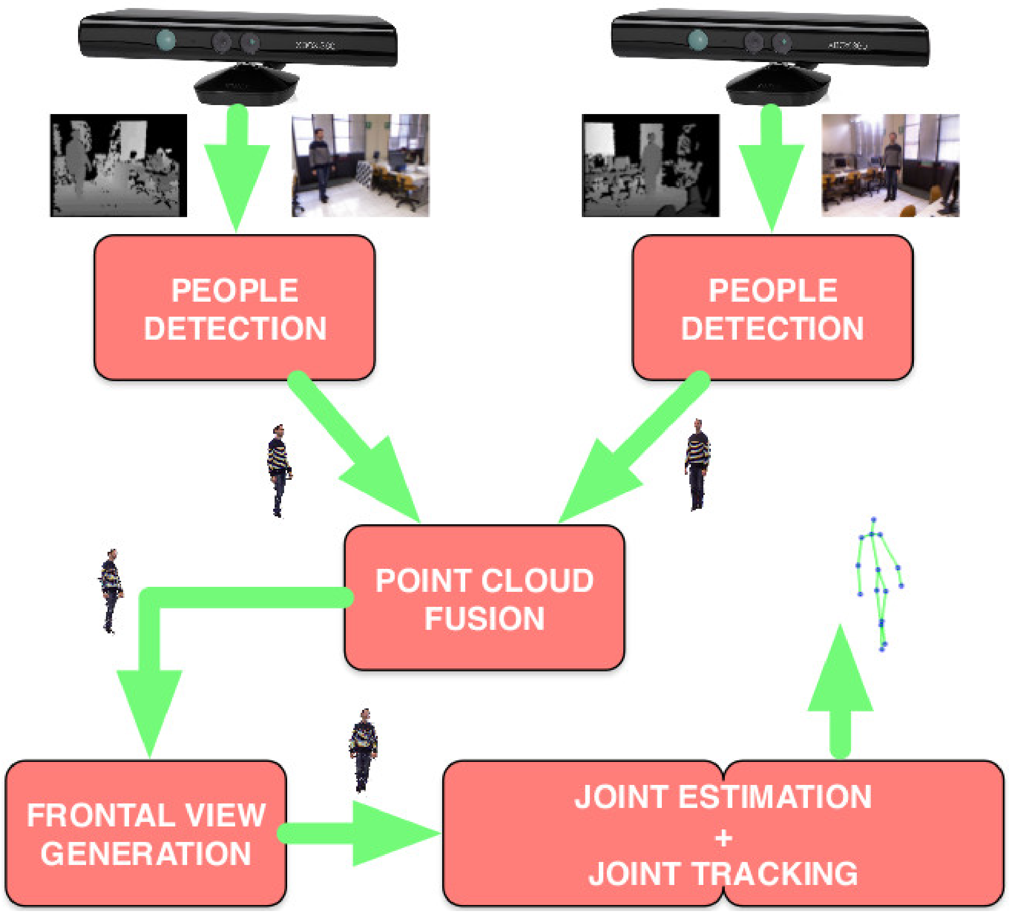


Figure 3.1: Overview of the proposed system.

frontal view generation in order to provide to the body part detector a more complete depth image of the person in the scene, thus improving the final performance.

3.3.1 People detection for background removal

The body part detector in [12] poorly estimates the lower body parts of a person when the ground plane is visible under the feet of the person or when the person is too close to the background. To overcome this problem, we added a people detection phase as a preprocessing for background removal. In this way, we build a new depth image where all the background points are set to a big depth value (e.g. 10 meters), so that the Random Forest in [12] can easily discard them from belonging to the foreground person. As for the people detector, we exploit the RGB-D people detection in [69, 76], that is publicly available in the Point Cloud Library and allows to robustly detect people and provide the point cloud points belonging to them. Then, these 3D points are reprojected to 2D to create a masked image which can be used instead of the entire depth image, improving the output of the original body part detector.

3.3.2 Point cloud fusion

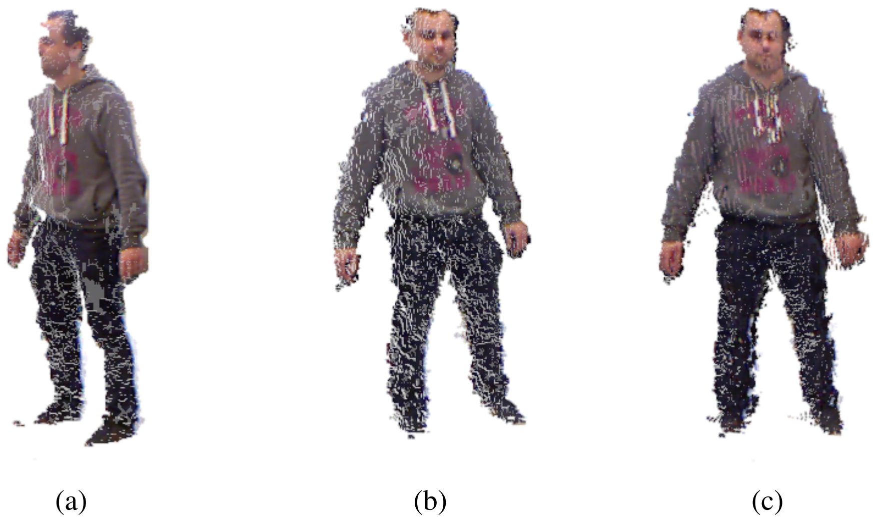


Figure 3.2: An example of the depth data fusion process. In a) the point cloud obtained from the C_1 camera, in b) the point cloud obtained from the C_0 camera and in c) the final fused cloud.

The information coming from multiple cameras is here exploited at the depth level, fusing the point clouds by means of the Iterative Closest Point algorithm. In particular, considering the network of two cameras C_0, C_1 we used for the experiments, we first obtain the segmented point clouds P_0, P_1 by means of the people detector and then, given the extrinsic parameters of the network, we refer these point clouds to a common *world* reference frame. After this transformation, the resulting point clouds P_1^w and P_0^w are finely registered by means of an ICP algorithm in order to account for depth estimation errors intrinsic of the sensors [127] or possible inaccuracies in the extrinsic calibration of the network. In formulas, we obtain the point clouds:

$$P_0^w = \mathfrak{T}_0^w(P_0) \tag{3.1}$$

$$P_1^w = \mathfrak{T}_1^w(P_1) \tag{3.2}$$

$$P_{total}^w = P_0^w \oplus \mathfrak{T}_{ICP}(P_1^w) \tag{3.3}$$

where, $\mathfrak{T}_i^j$ represents the transformation from the i reference frame to the j reference frame and $\mathfrak{T}_{ICP}$ is the transformation obtained by performing ICP with the point cloud P_0^w as the target cloud and the P_1^w as the source cloud. In Figure 3.2, an example of this process is shown. In order to lower the time to compute $\mathfrak{T}_{ICP}$, we calculate this transformation by using two downsampled versions of P_0^w and P_1^w and by limiting to 30 the number of iterations.

3.3.3 Frontal view generation

The best skeleton estimation comes from frontal-viewed persons. For this reason, we want to warp the total point cloud P_{total}^w obtained at Section 3.3.2 to be frontal with respect to the camera we chose as a reference, here C_0. In order to obtain this result, as shown in Figure 3.3, we project the points of P_{total}^w to the ground, that is the xOy plane of the *world* reference frame, thus obtaining a 2D shape that usually resembles an ellipsoid O of points.

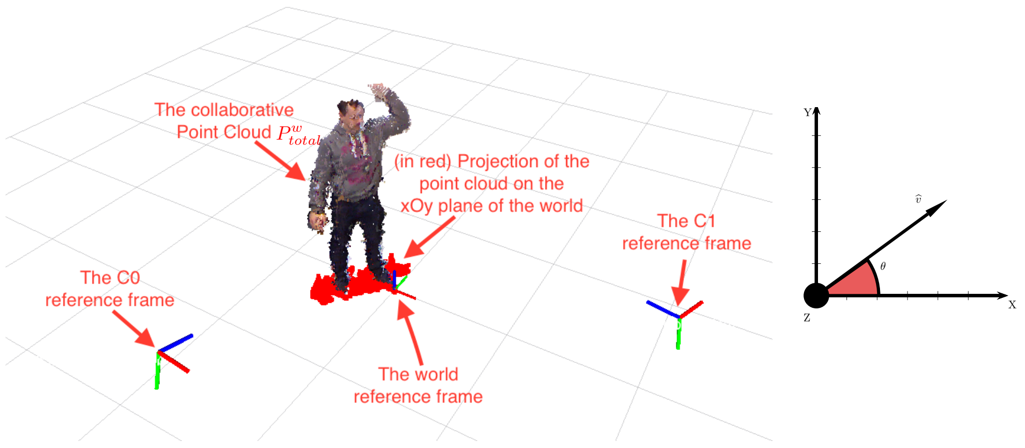

The collaborative Point Cloud P_{total}^w
(in red) Projection of the point cloud on the xOy plane of the world
The C1 reference frame
The C0 reference frame
The world reference frame
Y
$\hat{v}$
θ
Z
X

Figure 3.3: The frontal view generation phase. On the left there are the reference system of our Kinects and the common world reference system, the collaborative cloud obtained with ICP after the people detection phase and the same cloud projected on the xy plane of the world reference frame (visible in red). On the right, it is shown a top-view of the world reference frame, the vector representing the principal component $\hat{v}$ and the angle θ which is used for the frontal view warping. Best viewed in color.

We then calculate the principal components [91] of O in order to find a vector $\hat{v}$ with the same direction of the major axis of O, that is then used to rototranslate the original P_{total}^w with M:

$$M = \begin{bmatrix} R & \mathbf{t} \\ 0^{1x3} & 1 \end{bmatrix} \tag{3.4}$$

where R is the rotation matrix which rotates a cloud of θ around the *world* z-axis and

T the translation to bring the final point cloud to be centered on the *world* reference frame. In order to compute R, we need to compute θ, which is the angle between $\widehat{v}$ and $u_x = (1,0,0)$. In formulas, we have:

$$\theta = arccos\left(\frac{\widehat{v} \cdot u_x}{|\widehat{v}||u_x|}\right) \tag{3.5}$$

$$R = \begin{pmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

$$\mathbf{t} = -\begin{pmatrix} k_x \\ k_y \\ 0 \end{pmatrix}, \quad K = (k_x, k_y, k_z) = \frac{\sum_{i=0}^{|P^w_{total}|} P^w_{total}(i)}{|P^w_{total}|} \tag{3.7}$$

where K is the centroid of the total point cloud before the rototranslation. We can now obtain the desired frontal-view point cloud as:

$$P^w_{fv} = \{p = (x_p, y_p, z_p)^T \mid \exists q \in P^w_{total}, p = Mq\} \tag{3.8}$$

3.4 Body skeleton estimation

In this chapter, we use the algorithm described in [12] to perform body part detection that is open source and available in the Point Cloud Library. This detector takes as input a depth image, that is then classified by a Random Forest. For this reason, the multi-view and frontal point cloud P^w_{fv} obtained in Section 3.3.3 has to be projected to 2D in order to create a virtual depth-image $D_{virtual}$ that could be processed by the body part detector.

3.4.1 Virtual depth image generation

The virtual depth image $D_{virtual}$ is estimated by projecting the points in P^w_{fv} to the image plane of the C_0 camera, that has been taken as a reference. However, this process often leaves some holes in the generated image. We thus implemented a hole-filling procedure that fills the holes with the nearest valid points until a threshold distance. In formulas:

$$D_{virtual} = \{d_{ij} = (i,j) \in \mathbb{N}^2 | i \in (0,480), j \in (0,640)\} \tag{3.9}$$

$$d_{ij} = \begin{cases} P^0_{fv}(i,j), & (i,j) \text{ provides a valid point in } P^0_{fv} \\ P^0_{fv}(\overline{i,j}), & (\overline{i,j}) = argmin\{||(i,j) - (\widehat{i,j})|| < t | P^0_{fv}(\widehat{i,j}) \text{ is valid}\} \\ 10000, & \text{otherwise} \end{cases} \tag{3.10}$$

In Figure 3.4, a comparison of a sample image with and without the hole-filling procedure is shown. This hole-filled depth map is then provided as input of the body part detector.

Figure 3.4: On the left, the resultant point cloud re-projection to the image plane of the reference camera. On the right, the re-projection after the hole-filling procedure.

3.4.2 Joint estimation

The body part detector [12] assigns one of the 24 labels defined at training time to each pixel of the depth map and calculates the blobs of the coherent voxels with the same label.

From this preliminary segmentation, we then compute the positions of the skeleton joints in two steps. At first, we address the problem of the possibility of a single label being assigned to multiple coherent groups of voxels. This issue can be solved by combining the coherent voxel groups into a single blob or sorting them by their size with the largest one being selected for the joint calculation. Although the results of these simple methods were satisfying, the body part positions were imprecise in certain cases, especially as it comes to the smaller body parts such as hands and elbows. An improvement was achieved by building an optimal tree of the body parts, starting from the *Neck* as the root blob and further recursively estimating the child-blobs. This method is based on a pre-defined skeleton structure, which settles whether two body parts are connected as well as certain constraints regarding the expected size of the limbs.

In the second phase, the 3-D position of the joint is calculated from the selected blob. In most cases, the 3D centroid of the corresponding blob point cloud provides a good estimate for the joint position. An exception to this is the *Hip* blob, which also contains a large part of the torso. Besides, the *Shoulder* and *Elbow* joints are special cases which are described below.

Shoulders The shoulder position is calculated from the corresponding chest blob. Inside the blob point cloud, we estimate the voxel V_y_max with the maximum Y-value. We further build a sub-group of voxels belonging to the chest blob and having the distance to the V_y_max below a certain threshold (10 cm) and use the centroid of this sub-blob as the final position.

Elbows If the elbow blob was detected, we use the normal approach calculating the centroid of the blob. Otherwise, we estimate the point inside the arm blob, which has the longest distance from the previously estimated Shoulder joint.

Hips We define a certain threshold and build a sub-group of voxels belonging to the lower part of the hip blob. The centroid of this sub-group is used as the result position.

3.4.3 Joint tracking over time

The *Alpha-Beta filter* detailed in Algorithm 2 was implemented on top of the standard joint calculation to assure a consistent and continuous motion over time. This deterministic approach estimates the new position based on the predicted and measured position, with the weight of the measured position given by the parameter α, while β shows the weight of the velocity update.

Careful tuning of the α and β parameters is necessary to achieve the best results. Additionally, we have modified the update parameters for the hand joints, which usually have higher velocities then other body parts.

3.5 Experiments

We tested the different steps of the proposed approach with two series of RGB-D frames recorded from a network of two first-generation Microsoft Kinect. In these sequences, two different freely-moving persons were performing different movements.

Algorithm 2 Applying the Alpha-Beta filter for joint tracking at a timestep t.

INPUT:

- $Xm = [Xm_1, ..., Xm_n]$ - measured values of m body joints;
- $Xp = [Xp_1, ..., Xp_n]$ - previous values of m body joints;
- $V = [V_1, ..., V_n]$ - velocities of the body joints

OUTPUT:

- $X = [X_1, ..., X_n]$ - estimated joint positions;
- $V = [V_1, ..., V_n]$ - updated velocities:

```
1: procedure FILTERING($Xm, Xp, V$)
2:     Calculate the predicted position: $X_k = Xp_k + V_k * dt$
3:     Difference between measured and predicted: $R_k = Xm_k - X_k$
4:     New joint position value: $X_k = X_k + \alpha * R_k$
5:     New joint velocity value: $V_k = V_k + (\beta * R_k)/dt$
```

For measuring the accuracy, we considered the following skeleton estimation error:

$$\varepsilon = \frac{\sum_{frames} \frac{\sum_{joints} ||pos_{estim} - pos_{actual}||}{N_{joints}}}{N_{frames}} \tag{3.11}$$

where the ground truth for joint position pos_{actual} has been manually annotated. The system used for testing the methods proposed is an Ubuntu 14.04 machine with an Intel core i7-4770 CPU and a NVidia Geforce GTX 670 GPU. In Table 3.1, we reported a quantitative comparison of skeleton estimation with our methods and the original one in terms of Equation 3.11. In this table, we report also a baseline multi-view approach at the skeleton level in which each fused skeleton $\widehat{S}$ is the average of the single-view skeletons S_0 and S_1. While the original method [12] is independent from the background for the body pose estimation, the joint estimation algorithm is not and this cause the large ε obtained in our tests. Adding a people detection step, thus improve exponentially the performance gained by the joint estimator and our joint-tracking filter maintains the performance while smoothing the joints estimated.

Our novel multi-view approach outperforms both the single-view skeleton estimation and the baseline multi-view method we used. The results achieved are from 20% to 33% better than the single ones and up to 24% better than the baseline skeleton-based multi-view approach. The computational burden needed for computing a skeleton us-

	first person	second person
[12]	97.82	139.50
ours single-view with PD	34.35	35.01
ours single-view with PD and JT	34.50	35.35
baseline multi-view at the skeleton level	30.50	29.65
proposed multi-view approach at the depth level	**23.26**	**28.22**

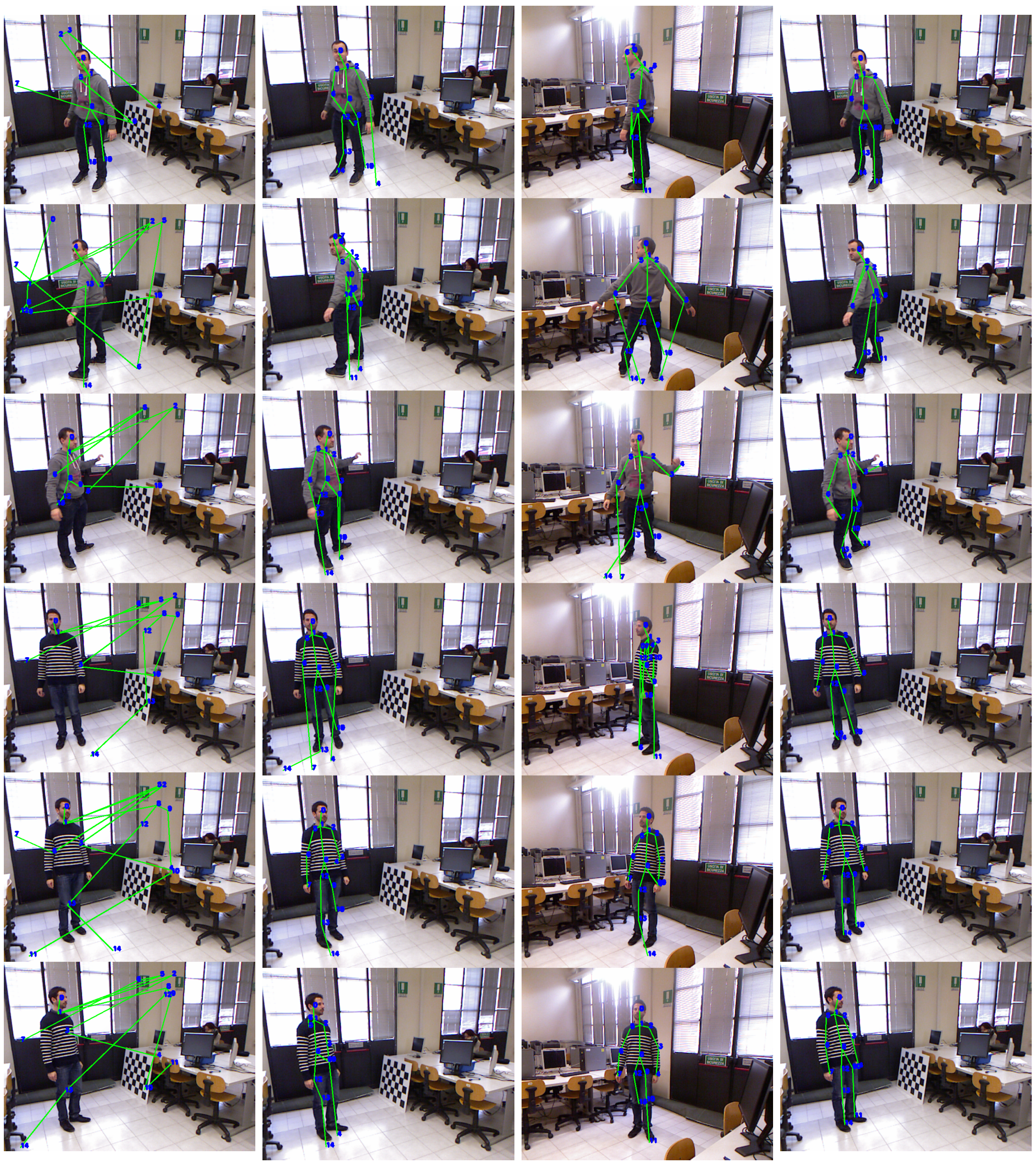
Table 3.1: The performance achieved by the original method [12] and our method. PD stands for people detection and JT for joint tracking.

ing a consumer i7-based computer is around 100 *ms* (60 *ms* for computing the virtual depth image plus 40 *ms* for the PCL skeleton computation) using a non-optimized version of the code. In Figure 3.5, we reported a qualitative comparison of skeleton estimation with these techniques.

3.6 Conclusions

In this chapter, we addressed the problem of human skeleton estimation and tracking in camera networks. We proposed a novel system to fuse depth data coming from multiple cameras and to generate a frontal view of the person in order to improve the skeleton estimation that can be obtained with state-of-the-art algorithms operating on depth data. Furthermore, we improved single-camera skeletal tracking by exploiting people detection for background removal and joint tracking for filtering joint trajectories. We tested the proposed system on hundreds of frames taken from two Kinect cameras, obtaining a great improvement with respect to state-of-the-art skeletal tracking applied to each camera. The proposed approach can be also applied to real-time scenarios given the low computational burden required.

The presented solution is effective, but it still presents problems when the person is partially seen by the camera network, resulting in a partial virtual depth image. Moreover, the approach is not scalable with the number of people, resulting in longer computational time required with multiple persons in the scene. To overcome this limitation, Chapter 4 will show an alternative solution to the same problem addressed. Last years, have seen impressive deep learning solutions that drastically enhanced the single-view quality of body pose estimators even with non-frontal views on RGB images. Chapter 4 takes advantage of this technology proposing a body pose estimation system that computes the 2D skeleton of all the people in the scene in one-shot. Then, the proposed system computes the 3D pose of all the persons with negligible overhead by fusing the outcomes of the single cameras by means of Kalman filtering. In this way, there is also no need of synchronization between the different network nodes allowing low-cost and heterogeneous camera networks. However, when the RGB image is not informative enough as during the night with no external light, the solution presented in this chapter is still to be preferred over the one presented in Chapter 3.



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

Figure 3.5: Some sample frames of the dataset we used for testing the proposed approach. Each row represents a frame. The different columns represent:
a) [12] on the C_0 stream;
b) ours with PD and JT on the C_0 stream;
c) ours with PD and JT on the C_1 stream;
d) our multi-view approach re-projected on the C_0 camera.

Chapter 4

Real-time and multi-person 3D Body Pose Estimation

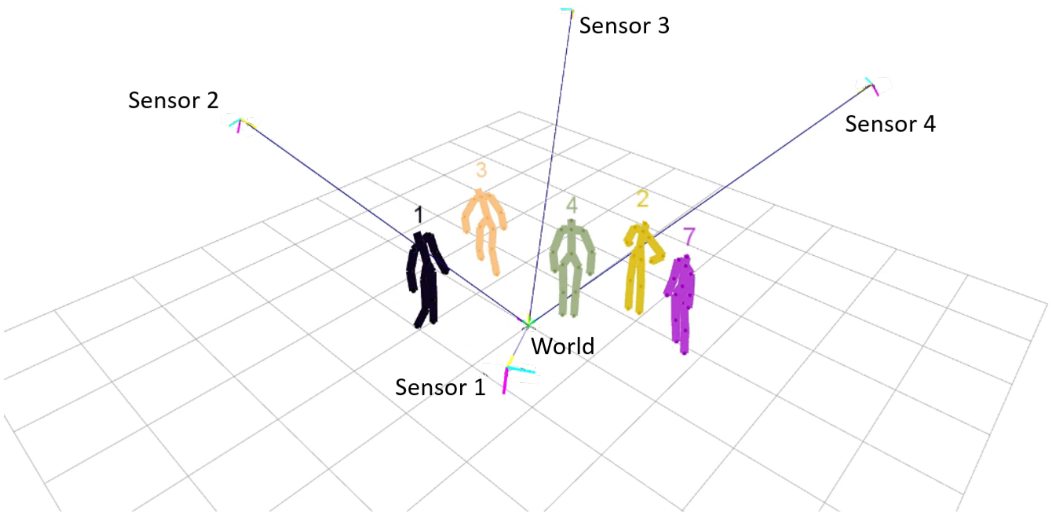


Figure 4.1: An example of a single-frame output provided by the system we are proposing. In the figure, five persons are seen from a network composed of four Microsoft Kinect v2. The system computes the body pose estimation of the persons in the scene with respect to a common world reference frame, set during the extrinsic calibration of the network.

The human body pose is rich of information. Many algorithms and applications, such as action recognition [42,114,126], people re-identification [36], Human-Computer-Interaction (HCI) [50] and industrial robotics [66,67,105] rely on this type of data. As already seen in Chapter 2, the recent availability of smart cameras [16, 17, 131] and affordable RGB-Depth sensors as the first and second generation Microsoft Kinect, allow to estimate and track body poses in a cost-efficient way. However, using a single

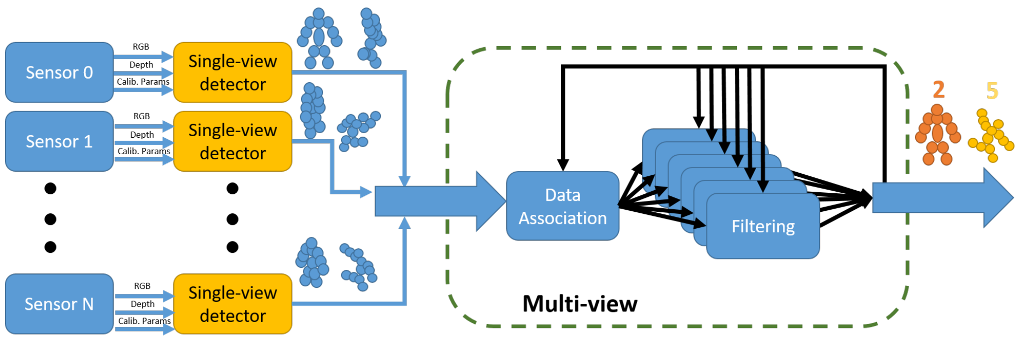


Figure 4.2: The system overview. The camera network is composed of several RGB-D sensors (from 1 to N). Each single-view detector takes the RGB and Depth images as input and computes the 3D skeletons of the people in the scene as the output using the calibration parameters K. The information is then sent to the multi-view central node which is in charge of computing the final pose estimation for each person in the scene. First, a data association is performed to decide which pose detection is belonging to which pose track, then a filtering step is performed to update the pose track given the detection.

sensor is often not reliable enough because of occlusions and Field-of-View (FOV) limitations. For this reason, a common solution is to take advantage of camera networks. Nowadays, the most reliable way to perform human Body Pose Estimation (BPE) is to use marker-based motion capture systems. These systems show great results in terms of accuracy (usually, less than 1mm), but they are very expensive and require the users to wear many markers, thus requiring a long set-up time and impairing a natural interaction because the user feels ”instrumented”. Moreover, these systems usually require offline computations and possibly long computational times, especially in complex scenarios with many markers and people, while the system we propose provides immediate results. A real-time response is usually needed in security applications, where person actions should be detected in time, or in industrial applications, where human motion is predicted to prevent collisions with robots in shared workspaces. Aimed by those reasons, the research on marker-less motion capture systems has been particularly active in recent years.

In this chapter, we propose a novel system to estimate the 3D human body pose in real-time. To the best of our knowledge, this is the first open-source and real-time solution to the multi-view, multi-person 3D body pose estimation problem. Figure 4.1 depicts our system output. The system relies on the feed of multiple RGB-D sensors (from 1 to N) placed in the scene and on an extrinsic calibration of the network. In this work, this calibration is performed with the `calibration_toolkit` [5][1]. The multi-

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

view poses are obtained by fusing the single view outcomes of each detector, that runs a state-of-the-art 2D body pose estimator [13, 118] and extend it to 3D by means of the sensor depth. The contribution of the chapter is two-fold: i) we propose a novel system to fuse and update 3D body poses of multiple persons in the scene and ii) we enriched a state-of-the-art single-view 2D pose estimation algorithm to provide 3D poses. As a further contribution, the code of the project has been released as open-source as part of the OpenPTrack [70, 74] repository. The proposed system is:

- *multi-view*: The fused poses are computed taking into account the different poses of the single-view detectors;

- *asynchronous*: The fusion algorithm does not require the different sensors to be synchronous or have the same frame rate. This allows the user to choose the detector computing node accordingly to his needs and possibilities;

- *multi-person*: The system does not make any assumption on the number of persons in the scene. The overhead due to the different number of persons is negligible;

- *scalable*: No assumptions are made on the number or positions of the cameras. The only request is an offline one-time extrinsic calibration of the network;

- *real-time*: The final pose framerate is linear to the number of cameras in the network. In our experiments, a single-camera network can provide from 5 fps to 15 fps depending on the Graphical Processing Unit (GPU) exploited by the detector. The final framerate of a camera network composed of k nodes is about the sum of their single-view framerate;

- *low-cost*: The system relies on affordable low-cost RGB-D sensors controlled by consumer GPU-enabled computers. No specific hardware is required.

The remainder of the chapter is organized as follows: in Section 4.1 we review the literature regarding human BPE from single and multiple views, while Section 4.2 describes our system and the approach used to solve the problem. In Section 4.3 experimental results are presented, and, finally in Section 4.4 we present our final conclusions.

4.1 Related Work

4.1.1 Single-view body pose estimation

Since a long time, there have been a great interest about single-view human BPE, in particular for gaming purposes or avatar animation. Shotton et al. [102] proposed the skeletal tracking system licensed by Microsoft used by the XBOX console with the first-generation Kinect. This approach used a random forest classifier to classify the different pixels as belonging to the different body parts. This work inspired an open-source approach that was released by Buys et al. [12]. This same work was then improved by adding the OpenPTrack people detector module as a preprocessing step [18]. Still, the performance of the detector remained very poor for non frontal persons.

In these last years, many challenging Computer Vision problems have been finally resolved by using *Convolutional Neural Networks* (CNNs) solutions. Also single-view BPE has seen a great benefit from these techniques [13,19,48,86]. The impressive pose estimation quality provided by those solution is usually paid in terms of computational time. Nevertheless, this limitation is going to be leveraged with newer network layouts and Graphical Processing Units (GPU) architectures, as proved by some recent works [13, 19]. In particular, the work of Cao et. al [13] was one of the first to implement a CNN solution to solve people BPE in real-time using a bottom-up approach. The authors were able to compute 2D poses for all the people in the scene with a single forward pass of their CNN. This work has been adopted here as part of our single-view detectors.

4.1.2 Multi-view body pose estimation

Multiple views can be exploited to be more robust against occlusions, self-occlusions and FOV limitations. In [26] a Convolutional Neural Network (CNN) approach is proposed to estimate the body poses of people by using a low number of cameras also in outdoor scenarios. The solution combines a generative and discriminative approach, since they use a CNN to compute the poses which are driven by an underlying model. For this reason, differently of the work presented here, the collaboration of the users is required for the initialization phase.

In our previous work [18], we solved the single-person human BPE by fusing the data of the different sensors and by applying an improved version of [12] to a virtual depth image of the frontalized person. In this way, the skeletonization is only

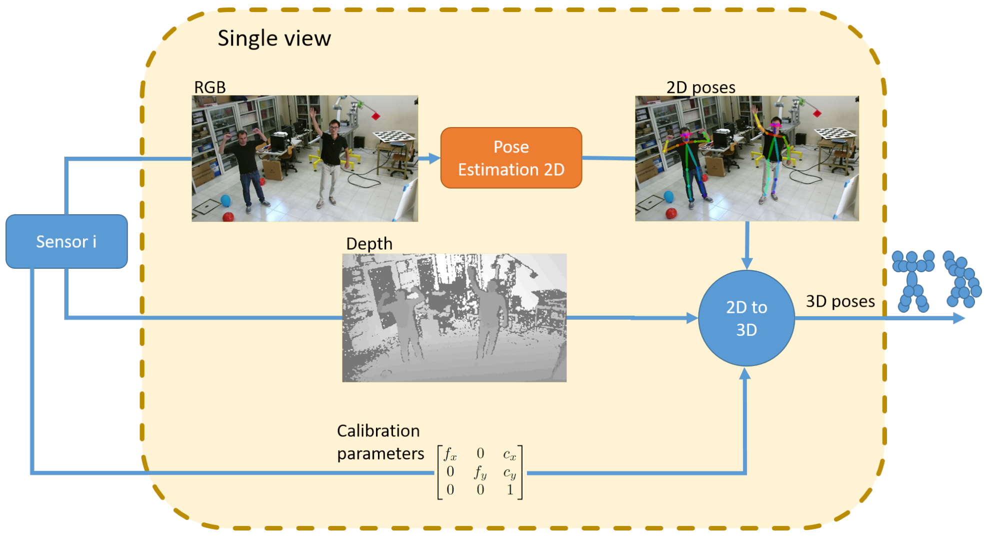


Figure 4.3: The single-view pipeline followed for each sensor. At each new frame composed of a color image (RGB), a depth image and the calibration parameters, the 3D pose of each person in the scene is computed from the 2D one. Then, the results are sent to the central computer which will compute the multi-view result.

performed once, on the virtual depth map of the person in frontal pose.

In [33], a 3D model is registered to the point clouds of two Kinects. The work provides very accurate results, but it is computationally expensive and not scalable to multiple persons.

The authors of [62] proposed a pure geometric approach to infer the multi-view pose from a synchronous set of 2D single-view skeletons obtained using [123]. The third dimension is computed by imposing a set of algebraic constraints from the triangulation of the multiple views. The final skeleton is then computed by solving a least square error method. While the method is computationally promising (skeleton computed in 1s per set of synchronized images with an unoptimized version of the code), it does not scale with the number of persons in the scene.

In [55] a system composed of common RGB cameras and RGB-D sensors are used together to record a dance motion performed by a user. The fusion method is obtained by selecting the best skeleton match between the different ones obtained by using a probabilistic approach with a particle filter. The system performs well enough for its goal, but it does not scale to multiple people and requires an expensive setup.

In [52] the skeletons obtained from the single images are enriched with a 3D model computed with the visual hull technique.

In [125] two orthogonal Kinects are used to improve the single-view outcome of both sensors. They used a constrained optimization framework with the bone lengths as hard constraints. While the work provides a real-time solution and there are no hard assumption on the Kinect positions, it was tested just with one person and two orthogonal Kinect sensors.

Similarly to many recent works [52, 55, 62], we use a single-view state-of-the-art body pose estimator [13], but we augment its results with the third dimension and we then combine the multiple views to improve the overall quality. To the best of our knowledge, no previous work was able to deal with multiple persons in the scene, while providing real-time outcomes.

4.2 System Design

Figure 4.2 shows an overview of the proposed system. It can be split into two parts: i) the single view detector, which is the same for each sensor and it is executed locally and ii) the multi-view part which is executed just by a central node of the network called master computer. In the single-view part (see Figure 4.3), each detector estimates the 2D body pose of each person in the scene using an open-source state-of-the-art single-view body pose estimator. In this work, we use the OpenPose[2] [13, 118] library, but the overall system is totally independent of the single-view algorithm used. The last operation made by the detector is to compute the 3D positions of each joint returned by OpenPose. This fusion is done by exploiting the depth information coming from the RGB-D sensor used. The 3D skeleton is then sent to the master computer for the fusion phase. This is done by means of multiple Kalman Filters used on the detection feeds, as explained in Section 4.2.3.

4.2.1 Camera Network setup

The camera network can be composed of several RGB-D sensors. In order to know the relative position of each camera, we calibrate the system using a solution similar to our previous works [70, 74]. From this passage we fix a common *world* reference frame $\mathscr{W}$ and we obtain a transformation $\mathscr{T}_{\mathscr{C}}^{\mathscr{W}}$, for each camera C in the network, which transforms points in the camera coordinate system C to the $\mathscr{W}$ reference system.

4.2.2 Single-view Estimation of 3D Poses

Each node in the network is composed of an RGB-D sensor and a computer to elaborate the images. Let ${}^{\mathfrak{R}}\mathfrak{F} = \{{}^{\mathfrak{R}}C, {}^{\mathfrak{R}}D\}$ be a frame captured by the detector $\mathfrak{R}$ and composed of the color image C and the depth image D all in the $\mathfrak{R}$ reference frame.

[2] https://github.com/CMU-Perceptual-Computing-Lab/openpose

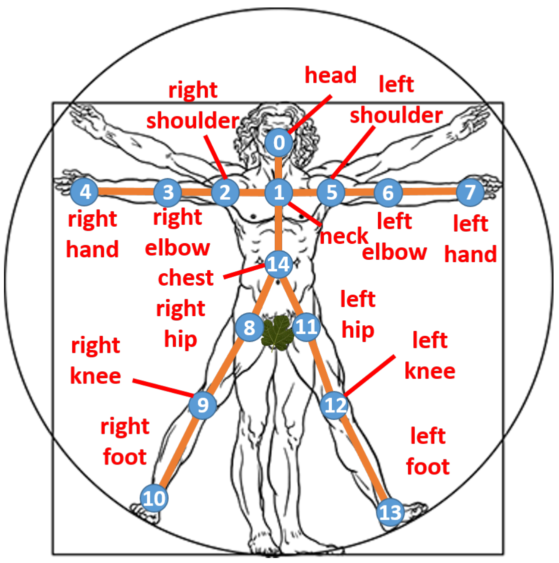


Figure 4.4: The human model used in this work.

The color and depth images in $\mathfrak{F}$ are considered as synchronized. We then apply *OpenPose* to ${}^{\mathfrak{R}}C$ obtaining the raw two dimensional skeletons $\overline{\mathfrak{S}} = \{\overline{S_0}, \overline{S_1}, \dots, \overline{S_k}\}$. Each $S = \{j_i \,|\, 0 \leq i \leq m\} \in \overline{\mathfrak{S}}$ is a set of 2D joints which follows the human model depicted in Figure 4.4. The goal of the single-view detector is to transform $\overline{\mathfrak{S}}$ in the set of skeletons $\widehat{\mathfrak{S}} = \{\widehat{S_0}, \widehat{S_1}, \dots, \widehat{S_k}\}$ where each $\widehat{S} \in \mathfrak{S}$ is a three dimensional skeleton. Given the RGB image I, let's consider a point $p = (x_p, y_p) \in I$ and its corresponding depth $d = proj(x_p, y_p)$ where the function $proj(\cdot)$ is obtained by means of extrinsic calibration of the depth and camera imagers of the RGB-D sensor. Considering (f_x, f_y) and (c_x, c_y) respectively the focal length and the optical center of the sensor, the relationship to compute the 3D point ${}^{\mathfrak{R}}P = ({}^{\mathfrak{R}}X, {}^{\mathfrak{R}}Y, {}^{\mathfrak{R}}Z)$ in the camera reference system $\mathfrak{R}$ is explained in Equation 4.1.

$$p = \begin{bmatrix} x_p \\ y_p \\ d \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathfrak{R}}X \\ {}^{\mathfrak{R}}Y \\ {}^{\mathfrak{R}}Z \end{bmatrix} = K \cdot {}^{\mathfrak{R}}P \tag{4.1}$$

Since the depth data is potentially noisy or missing, instead of directly considering $proj(\cdot)$ as the depth value, we refine it by calculating the median of the set $\mathfrak{D}(p)$. The calculation of the depth d associated to the point $p = (x_p, y_p)$ is shown in Equations 4.2 and 4.3.

$$\mathfrak{D}(p = (x_p, y_p)) = \{(x,y) \,|\, ||(x,y) - (x_p, y_p)|| < \varepsilon\} \tag{4.2}$$

$$d = \phi(p) = \underset{\alpha \neq 0}{\text{median}}\{\alpha = proj(x,y) \,|\, (x,y) \in \mathfrak{D}(p)\} \tag{4.3}$$

Given $\overline{\mathfrak{S}}$, we then proceed to the calculation of $\widehat{\mathfrak{S}}$ as shown in Equation 4.4.

$$\begin{gathered}\forall 0 \leq j < k, \quad \overline{S_j} = \{\overline{j_i} = (x_i, y_i) \,|\, 0 \leq i < m\} \in \overline{\mathfrak{S}}, \\ \widehat{S_j} = \left\{ \widehat{j_i} = \begin{bmatrix} |K^{-1} \cdot \overline{j_i}|_x \\ |K^{-1} \cdot \overline{j_i}|_y \\ \phi(\overline{j_i}) \end{bmatrix}, 0 \leq i < m \right\} \in \widehat{\mathfrak{S}}\end{gathered} \tag{4.4}$$

Once the set $\widehat{\mathfrak{S}}$ is available, the detector sends it to the master computer for the multi-view fusion computation.

4.2.3 Multi-view fusion of 3D poses

The master computer fuses the different information it receives from the single-view detectors in the network. One of the common limitations in motion capture systems is the necessity to have synchronized cameras. Moreover, off-the-shelves RGB-D sensors, such as the Microsoft Kinect v2, do not have the possibility to trigger the image acquisition. In order to overcome this limitation, our solution merges the different data streams asynchronously. This allows the system to work also with other RGB-D sensors or other low-cost embedded machine which can provide different frame-rates. At time t, the master computer maintains a set of tracks $\mathfrak{T} = \{T_0, T_1, ..., T_l\}$ where each pose tracked T_i is composed of the set of states obtained by m different Kalman Filters, one per each joint, i.e: $T_i = \{\mathscr{S}(\mathscr{K}_{i0}), \mathscr{S}(\mathscr{K}_{i1}), ..., \mathscr{S}(\mathscr{K}_{im})\}$. The additional Kalman Filter $\mathscr{K}_{im}$ is needed for the data association algorithm. Indeed, the first operation the master performs is to associate the different skeleton observations to its $\mathfrak{T}$. At time $t+1$, it may arrive a detection $\widehat{\mathfrak{S}_i} = \{\widehat{S_0}, \widehat{S_1}, ..., \widehat{S_k}\}$ from the sensor i of the network. The master computer first refers the detection to the common *world* coordinate system $\mathscr{W}$ (see Section 4.2.1) using the following transformation:

$$^{\mathscr{W}}\widehat{\mathfrak{S}_i} = \mathscr{T}_i^{\mathscr{W}} \cdot \widehat{\mathfrak{S}_i} = \{\mathscr{T}_i^{\mathscr{W}} \cdot S_j \,|\, \forall S_j \in \widehat{\mathfrak{S}_i}\} \tag{4.5}$$

Then, it associates the different skeletons in $^{\mathscr{W}}\widehat{\mathfrak{S}_i}$ as new observations for the different tracks in $\mathfrak{T}$ if they belong to them, or initializes new tracks if some of the skeletons do not belong to any $T_i \in \mathfrak{T}$. At this stage, the system also decides if a track is old and has to be removed from $\mathfrak{T}$. This step is important to prevent $\mathfrak{T}$ to grow big causing time computing problems with systems which are running for long time. We refer to this phase as *data association*. Algorithm 3 shows how it is performed.

The data association is done by considering the centroid of each skeleton S con-

Algorithm 3 The algorithm performed by the master computer to decide the association between the different skeletons in a detection and the current tracks.

INPUT:

- ${}^{\mathscr{W}}\widehat{\mathfrak{S}}_{\mathfrak{i}} = \{S_0, S_1, ..., S_{k-1}\}$ - a new detection set from sensor i expressed in the world reference frame
- $\mathfrak{T} = \{T_0, T_1, ..., T_{l-1}\}$ - the current set of tracked person poses.
- ε - maximum distance for a detection to be considered for the association

OUTPUT:

- $\mathscr{M} = \{(S_i, T_j) \in^{\mathscr{W}} \widehat{\mathfrak{S}}_{\mathfrak{i}} \times \mathfrak{T}\}$ - the association between the pose tracked and the new observations
- $\mathscr{N} \subseteq^{\mathscr{W}} \widehat{\mathfrak{S}}_{\mathfrak{i}}$ - the detections without an association. They will initialize a new track.
- $\mathfrak{T}_o \subseteq \mathfrak{T}$ - the tracks without an associated observations. They will be considered for removal if they remain without an observations for a long time.

```
1:  procedure DATA_ASSOCIATION(${}^{\mathscr{W}}\widehat{\mathfrak{S}}_{\mathfrak{i}}$, $\mathfrak{T}, \varepsilon$)
2:      $\mathfrak{T}_o \leftarrow \emptyset$
3:      $C \leftarrow \mathbf{0}_{k \times l}$
4:      for each $T_i \in \mathfrak{T}$ do
5:          for each $S_j \in^{\mathscr{W}} \widehat{\mathfrak{S}}_{\mathfrak{i}}$ do
6:              $x_t(j) \leftarrow centroid(S_j)$
7:              $z_t(i,j) \leftarrow$ *v that $T_i$ would have if $S_j$ were associated to it*
8:              $\widehat{z}_{t|t-1}(i) \leftarrow$ *prediction step of $\mathscr{K}_{im}$*
9:              $\Sigma_t(i) \leftarrow \Sigma_t(\mathscr{K}_{im})$
10:             $\tilde{z}_t(i,j) \leftarrow z_t(i,j) - \widehat{z}_{t|t-1}(i)$
11:             $C_{ij} \leftarrow \tilde{z}_t^T(i,j) \cdot \Sigma_t(i)^{-1} \cdot \tilde{z}_t(i,j)$
12:     $X \leftarrow solve\_Munkres(C)$
13:     for $i \in [0, l-1]$ do
14:         for $j \in [i+1, k-1]$ do
15:             if $X_{ij} = 1$ and $C_{ij} < \varepsilon$ then
16:                 $\mathscr{M} \leftarrow \mathscr{M} \cup \{(S_j, T_i)\}$
17:                 * update $\mathscr{K}_{im}$ with $S_j$ *
18:     $\mathscr{N} \leftarrow \{S_i \mid \nexists T_j, (S_i, T_j) \in \mathscr{M}\}$
19:     $\mathfrak{T}_o \leftarrow \{T_i \mid \nexists S_j, (S_j, T_i) \in \mathscr{M}\}$
20:     return $\mathscr{M}, \mathscr{N}, \mathfrak{T}_{\mathfrak{o}}$
```

tained in the detection ${}^{\mathscr{W}}\mathfrak{S}_{\mathrm{i}}$. The centroid is calculated as the chest joint $j_{14} \in S$, if this is valid, otherwise it is replaced with a weighted mean of the neighbor joints. Lines [6-9] of Algorithm 3 refers to the calculation of a cost associated to the case if the detection pose S_j would be associated to the track T_i. To calculate this, we consider the Mahalanobis distance (Line 11) between the likelihood vector at time t $\tilde{z}_t(i,j)$ and $\Sigma_t(\mathscr{K}_{i,x_t})$: the covariance matrix of the Kalman filter associated to the centroid of T_i. At this point, we are trying to solve the assignment problem with C as the cost matrix. This is solved by the Hungarian algorithm run on C; Line 11 refers to the use of the Munkres algorithm which efficiently computes the optimal matrix X with a 1 on the optimal couples and a 0 on the rest of them. Nevertheless, this algorithm does not consider a maximum distance between tracks and detections. Thus, it may happen that a couple is wrongly associated in the optimal assignment. For this reason, when inserting the optimal couples in $\mathscr{M}$, we check also if the cost of the couple in the initial cost matrix C is below a threshold.

Once solved the data association problem, we can assign the tracks ID to the different skeletons. Indeed, we know which are the detection at the current time t belonging to the tracks in the system and, additionally, we know also which tracks need to be created (i.e. new detections with no associated track) and the tracks to consider for the removal. Let n be the number of people in the scene, we used a set of Unscented Kalman Filters $\mathfrak{K} = \{\mathscr{K}_{ij}, 0 \leq i < n, 0 \leq j \leq m\}$ where the generic $\mathscr{K}_{ij} \in \mathfrak{K}$ is in charge of computing the new position of the joint j of the person i at time t, given the new detection received from one of the detectors at time t and the prediction of the filter $\mathscr{K}_{ij}$ computed from the previous position at time $t-1$ of the same joint j.

The state of each Kalman Filter $\mathscr{K}_{ij}$ is six-dimensional: $\mathscr{S}(\mathscr{K}_{ij}) = \{x, y, z, v_x, v_y, v_z\} \in \mathbb{R}^6$ composed by the position and the velocity of the track. As motion model we used a constant velocity model, since it is good to predict joint movements in the small temporal space between two good detections of that joint.

4.3 Experiments

The algorithm described in this chapter does not require any synchronization between the cameras in the networks. This fact makes particularly difficult to find a fair comparison between our proposed system and other state-of-the-art works. Thus, in order to provide useful indication on how our system performs, we recorded and manually annotated a set of RGB-D frames while a person was freely moving in the field-of-view of a 4-sensors camera network. Although our system is able to track

		r-shoulder	r-elbow	r-wrist	r-hip	r-knee	r-ankle
single-camera network	MAF_{30}	>100	>100	>100	>100	>100	>100
	MAF_{40}	>100	>100	>100	>100	>100	>100
	Ours	**54.9** ± **58.6**	**42.4** ± **47.4**	**42.4** ± **40.0**	**51.7** ± **43.7**	**54.5** ± **31.0**	**63.3** ± **34.2**
2-camera network	MAF_{30}	62.0 ± 33.0	62.9 ± 32.0	63.1 ± 34.5	76.4 ± 30.6	75.9 ± 27.4	88.3 ± 35.6
	MAF_{40}	83.7 ± 41.8	84.0 ± 40.9	83.1 ± 43.7	99.2 ± 40.4	96.3 ± 38.0	>100
	Ours	**20.7** ± **17.2**	**21.0** ± **17.5**	**24.3** ± **17.5**	**22.4** ± **16.7**	**42.8** ± **17.2**	**59.7** ± **28.6**
4-camera network	MAF_{30}	28.7 ± 16.4	31.0 ± 16.9	32.2 ± 22.5	40.2 ± 15.0	48.7 ± 12.8	58.6 ± 21.2
	MAF_{40}	38.4 ± 21.2	40.8 ± 21.7	41.6 ± 26.3	50.7 ± 19.4	56.2 ± 16.7	66.0 ± 24.5
	Ours	**22.7** ± **18.9**	**21.3** ± **18.5**	**26.3** ± **19.9**	**23.9** ± **18.0**	**46.5** ± **19.7**	**55.9** ± **25.1**
		l-shoulder	l-elbow	l-wrist	l-hip	l-knee	l-ankle
single-camera network	MAF_{30}	>100	>100	>100	>100	>100	>100
	MAF_{40}	>100	>100	>100	>100	>100	>100
	Ours	**77.7** ± **74.4**	**79.1** ± **82.7**	**70.0** ± **61.8**	**97.8** ± **30.3**	**57.5** ± **38.9**	**69.2** ± **37.6**
2-camera network	MAF_{30}	83.3 ± 33.4	85.8 ± 37.8	94.8 ± 45.4	>100	85.4 ± 35.5	93.3 ± 37.0
	MAF_{40}	>100	>100	>100	>100	>100	>100
	Ours	**32.1** ± **23.0**	**33.4** ± **26.3**	**39.8** ± **35.1**	**98.3** ± **21.2**	**39.9** ± **18.3**	**58.6** ± **27.1**
4-camera network	MAF_{30}	41.5 ± 17.9	39.9 ± 19.6	44.7 ± 29.5	94.1 ± 26.1	52.1 ± 17.8	57.8 ± 27.9
	MAF_{40}	53.0 ± 23.2	52.7 ± 24.6	57.6 ± 33.1	96.6 ± 30.8	61.2 ± 23.1	67.5 ± 31.6
	Ours	**22.5** ± **22.1**	**26.7** ± **25.9**	**31.8** ± **29.7**	**95.4** ± **22.0**	**45.1** ± **20.5**	**49.1** ± **25.2**

Table 4.1: The results of the experiments. Each number represents the mean and the standard deviation of the re-projection error on a reference camera (see Equation 4.6)
.

multiple persons in the scene, it is particularly time-consuming to manually annotate a dataset, therefore, for the experimental setup we considered a dataset involving just one person.

We compare our algorithm with a baseline method called MAF (Moving Average Filter), in which the outcome of the generic joint i at time t is computed as an average of the last k frames. In order to be as fair as possible, we fixed $k \geq 30$ to provide comparable results in terms of smoothness. This baseline operates the same identical data association performed by our algorithm and explained in Section 4.2.3.

We also demonstrated the effectiveness of the multi-view fusion by comparing our results with the poses obtained by considering just one and two cameras of the same network. As metrics, we report the average reprojection error with respect to one of the cameras, C_0. Equation 4.6 shows how this error is calculated with ${}^{\mathscr{W}}P$ as the generic joint expressed in the world reference system and p^* as the corresponding, manually annotated, ground truth:

$$e_{\text{repr}} = |p^* - K \cdot \mathscr{T}_{\mathscr{W}}^{C_0} \cdot {}^{\mathscr{W}}P| \tag{4.6}$$

Table 4.1 shows the results we achieved. As depicted, the proposed method outperforms the baseline in all the cases: single-view, 2-camera network and 4-camera network. In the first two cases (single and 2-camera network) the improvement is from 50% to 60%, while, when multiple views are available, it is from 18% to 32%. It is also interesting to note that the most noisy joints are the ones relative to the legs as confirmed by other state-of-the-art works [13, 48, 86].

4.3.1 Implementation Details

The system has been implemented and tested with Ubuntu 14.04 and Ubuntu 16.04 operating system using the Robot Operating System (ROS) [90] middleware. The code is entirely written in `C++` using the `Eigen`, `OpenCV` and `PCL` libraries.

4.4 Conclusions and Future Works

In this chapter we presented a framework to compute the 3D body pose of each person in a RGB-D camera network using only its extrinsic calibration as a prior. The system described outperforms the one shown in Chapter 3 by the fact that it works with multiple people at the same time with negligible overhead and there is no need of synchronization between the data stream coming from the cameras. Indeed, the system does not make any assumption on the number of cameras, on the number of persons in the scene, on their initial poses or clothes and the nature of the tracker presented does not require the cameras to be synchronous. In our experimental setup we demonstrated the validity of our system over both single-view and multi-view approaches. In order to provide the best service to the Computer Vision community and to provide also a future baseline method to other researchers, we released the source code under the BSD license as part of the OpenPTrack library[3].

As future works, we plan to add a human dynamic model to guide the prediction of the Kalman Filters to further improve the performance achievable by our system (in particular for the lower joints) and to further validate the proposed system on a new RGB-Depth dataset annotated with the ground truth of the single links of the persons' body pose. The ground truth will be provided by a marker based commercial motion capture system.

While 3D human body pose estimation is a great feature for allowing rich HCI systems, camera networks should also be able to detect and recognize the single persons. Indeed, this is a very important capability for assistive robots or for security purposes. Chapter 5 goes in this direction by presenting a solution to re-identify persons in RGB-D camera networks over days.

[3] `https://github.com/openptrack/open_ptrack_v2`

Chapter 5

People re-identification by means of face recognition in camera networks

Since several years, people detection and tracking is a very hot research topic that has attracted much attention in the computer vision community. This is due to the particular importance of this task in different domains such as surveillance as well as autonomous systems and culture. People tracking is used mostly for security reasons (e.g. detect and avoid collisions with pedestrians) or for artistic creation (e.g. animate interactive walls whenever a human is passing by). For different applications, though, the identity of each person becomes important. For example, an Ambient Assisted Living (AAL) robot, in charge of following a certain person, is supposed to always recognize that specific person, also among many other people and in presence of occlusions that may cause the robot to temporarily loose its track.

When multiple cameras are available in a network, it is also possible to share the knowledge between the nodes, building distribute algorithms that boost the recognition performance. Many works proposed a solution to similar scenarios [7, 64, 79]. They achieved good performance by using appearance features robust to pose and illuminance together with sophisticated feature comparison methods. However, those works provide a good solution using only the RGB information relative to the clothes of the

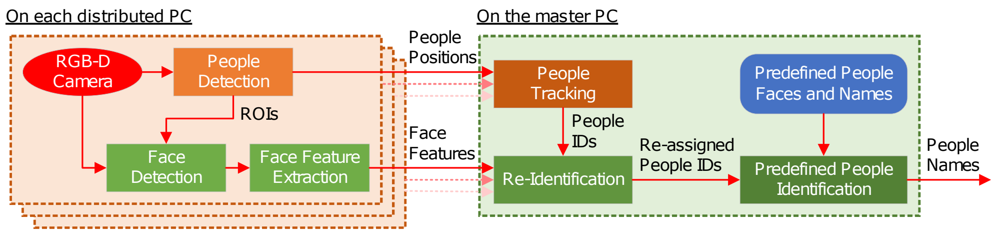


Figure 5.1: System overview.

subjects in the scene. Therefore, they cannot identify people over days as requested by many long-term service scenarios.

People re-identification[1] research has historically used several soft biometrical features (e.g. skeletal lengths [72], gait recognition [8, 58]). Such features are good to identify people in different days, since they are not invariant over time. However, when dealing with large number of persons with similar physiques, they may fail to correctly identify each person. In order to boost the reliability of the re-identification, the chosen features should be discriminative and robust to changes in illumination and viewpoint. Given that the face provides the most discriminative and reliable features to identify persons [129], this work proposes a new approach to leverage them in order to re-identify people.

Face recognition is one of the most studied field in Computer Vision [4, 11, 83, 87, 106, 107] . Nevertheless, the face has not been used as well for re-identification purposes in large camera networks, where faces are not always completely visible and there are pose, illumination and expression (PIE) changes. One of the most famous work that tried to use faces for this purpose was [7]. Nonetheless, faces were used as an additional assistance to appearance features and always for short-term re-identification purposes. Our goal is to propose a system that can re-identify people over days.

PIE issues represented a big challenge and they were very difficult to solve with traditional methods (e.g. EigenFaces [110], Local Binary Patterns [128], Scale-Invariant Feature Transform (SIFT) [34]). Recently, Deep Learning based techniques [83, 101, 106, 107] show big improvements in this sense, almost approaching the perfection in accuracy (99.6%) using one of the most challenging dataset for face recognition, the Labelled Faces in the Wild (LFW) [46]. Face visibility is still considered as an open problem, though.

In this chapter, we propose a novel system for re-assigning people ID in a people tracking framework using face recognition exploiting multiple RGB-D cameras. Indeed, camera networks could provide multiple point of views, reducing the face visibility issues. As people tracking system we used *OpenPTrack* [70], an RGB-D camera network-based people tracking framework. Our proposed system learns faces online and re-identifies (or identifies for the first time) people whenever they appear in the Field-of-View (FoV) of the camera network. The design of our network is similar to the OpenPTrack one, where a master computer is in charge of fusing the different people detection coming from the network nodes. In our work, the master computer is

[1]In this work, we used the term re-identification to refer to the recognition of persons over days considering the camera network as a whole. Historically, the same term has also been used to refer to the recognition of persons between different cameras, usually with non-overlapping FoVs.

also in charge of re-assigning ID to the tracks, using the face recognition results from the distributed nodes.

The chapter's contribution can be summarized as two-fold:

- we proposed a Bayesian inference-based face classification method. It reliably recognize a person using the confidences obtained by a Deep Neural Network (DNN)-based face recognition library. In our experiments we demonstrated how this type of inference is better than the baseline;
- we released the code of the project as open-source. The project is completely modular and uses the ROS [90] middleware. We also demonstrated this by making the network collaborate with a mobile robot (described in Appendix A).

The remainder of the chapter is organized as follows: Section 5.1 shows an overview of the system, while Section 5.2 describes the proposed face recognition framework. In Section 5.3 the integration of our system with a robotic platform is shown. Finally, Sections 5.4 and 5.5 show the experiments we made to validate the system and the conclusions, respectively.

5.1 System Overview

Figure 5.1 depicts the system overview. We track people by means of *OpenPTrack* [74] (Figure 5.2 shows a snapshot of the system) and, in order to recognize people faces, we exploit an open-source DNN-based face representation described in the work of Amos et. al. [11].

Each distributed node is in charge of providing to the master node a set of single-frame people detections and face features. For this reason, for each person tracked, each node extracts a Region of Interest (ROI) using a fixed-proportional window on top of the bounding box of the person considered. The face is then found inside this ROI and its signature calculated using OpenFace [11]. In this way, the network communication is efficient, since the data each node is sending (people detection and face signatures) is very light and our system can grow up to a large and dense network without occurring in bandwidth problems.

While the master node of the network computes people tracks by applying the OpenPTrack algorithm, our proposed system re-identifies people by comparing the signatures it is receiving. In order to achieve this goal, we exploit the concept of track, that connects several detections and face signatures of the same person (potentially unknown) in different time slots. Thus, by integrating this information, our algorithm builds a discriminative online descriptor for each person in the system. Finally, by comparing the descriptor to the known faces gallery it decides whether the person is known (the system re-assigns to the track the ID of the person in the system and updates his/her descriptor) or unknown. In this case, the system adds the computed descriptors and person ID to the gallery for future re-identification.

As an additional feature, the system also provides an offline people identification capability. Indeed, it is possible to add a set of people faces together with their names to the gallery in order to start the system using previous knowledge. This capability is very useful to build rich Human-Robot-Interaction (HRI) systems as well as AAL robots [14].

5.2 Proposed Method

5.2.1 People Detection and Tracking

In order to compute the tracks of the different people in the scene using a RGB-D camera network, we used *OpenPTrack* [70], an open-source people tracking library, as a barebone of our system.

The algorithm first segments and removes the ground plane from a point cloud and then clusters the remaining points. Finally, a classification phase discriminates each cluster in person or not person. The classification is performed by means of a Support Vector Machine (SVM) trained on the Histograms-of-Gradients (HOG) computed on the re-projected cluster on the original depth image. The people detection is performed by each sensor in the network. Then, the outcomes are sent to a master computer which is in charge of adding/updating/removing the people tracks in the system. This is performed by combining the outcomes solving the global nearest neighbor data association and using Kalman Filtering.

In Figure 5.2 an example of an artistic OpenPTrack installation is shown. The library provides real-time tracks of people as long as each person is visible. On the other hand, whenever the system looses a person track and then the same person re-appears, he acquires a new track ID. The goal of this work is to enable the network to recognize whenever such case happens and solve it by re-assigning the old ID to the new track. The only method to perform such task is by using long-term people re-identification, since a short-term re-id approach would not recognize people in different days.

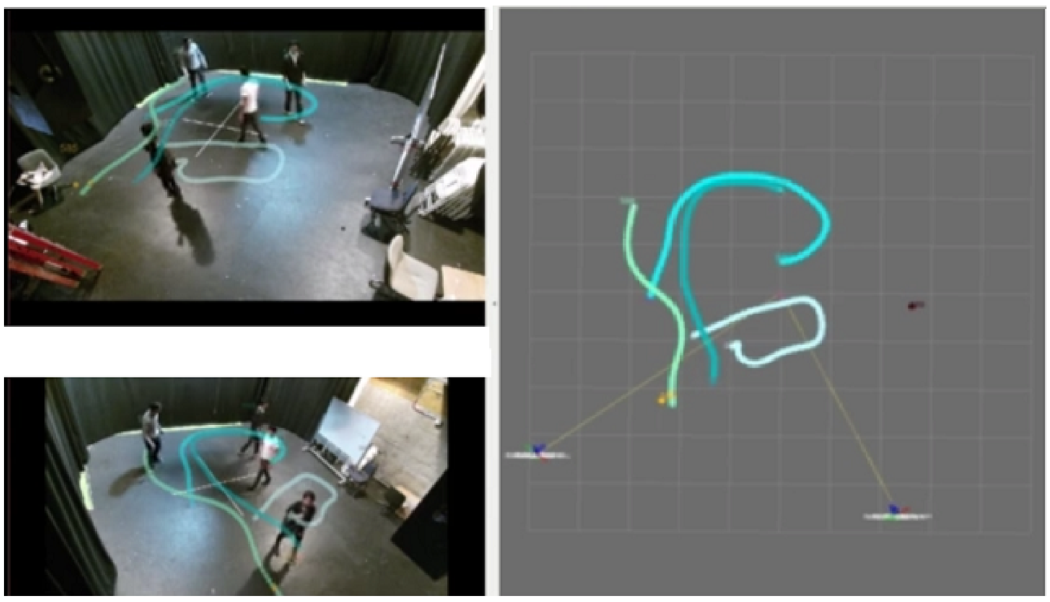


Figure 5.2: An example of OpenPTrack, a people detection and tracking framework using a RGB-D camera network. The system provides robust people tracking. However, person re-identification capability is necessary to recover the track of a person left from the camera view. [74]

5.2.2 Face Detection

The goal of our work is to provide a system that can work in real-time. For this reason, to detect faces we used a top-down approach. Firstly, we calculate the face ROIs exploiting the bounding boxes returned by OpenPTrack together with the people detection results. The ROIs are obtained by projecting a rectangle with fixed metric dimension (0.2m) to the top of each bounding box. Secondarily, we apply a Histograms-Of-Gradients (HOG) and cascaded SVM-based face detector [56] just on the ROIs. This method can find faces even when they are partially occluded, but it is quite slow if run on an entire image. Once the face is found, we then obtain the face features by using OpenFace. Figure 5.3 depicts an example of the face detection strategy we used. The different boxes indicate the ROIs and the detected faces respectively in green and in red.

5.2.3 Deep Neural Network-based Face Features

We use the OpenFace [11] library to generate face features to use in our algorithm. The library is an open-source implementation of FaceNet [101]. The goal of FaceNet is to compute a 128-dimensional feature vector which describes a face. The deep neural network behind the library is trained using a triplet loss function. It takes three

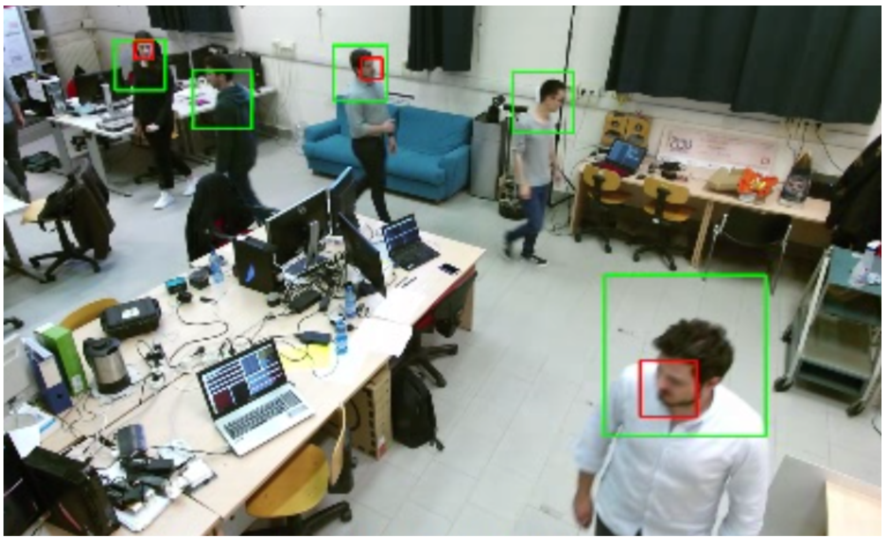
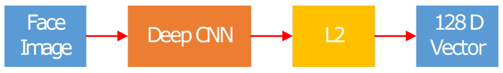
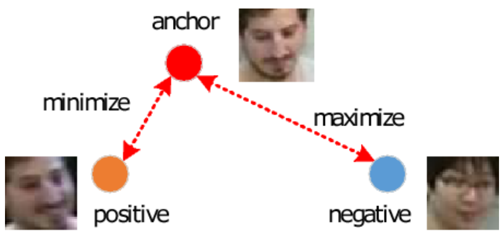


Figure 5.3: The result of the face detection approach on a frame. ROIs (visible in green boxes), are calculated from the people detections. The regions classified as faces are shown in red. As it can be seen, some faces are not detected due to occlusion. However, by integrating this information the number of unseen faces in a camera network remains low.

training data as input composed of a training data as an *anchor*, a negative data (i.e. with different identity as the anchor) and a positive data. The triplet loss function used minimizes the distance between the anchor-positive pair, while maximizing the one between the anchor-negative pair. As a result, features of faces of the same person will have a close L2 distance, while faces of different people will be far away from each other.



(a) Network Architecture.



(b) Triplet loss function.

Figure 5.4: *FaceNet* framework [101]. This framework employs a deep convolutional neural network and the triplet loss function to obtain discriminative face representations.

5.2.4 People Re-identification by Face Recognition

Algorithm 4 describes how the re-identification works. *face_list* represents the list of known faces in the system (i.e. faces which are learnt online and added offline), while *track_list* represents the people tracks currently seen by the system. If a person is not recognized by the system (i.e. just entered in the FoV of the camera network) his/her face features are accumulated and classified into the online learnt faces. If the confidence obtained is high enough, the algorithm finishes, since the person is considered as re-identified. Otherwise, we consider him/her as a new person seen for the first time. For the time the track is active, the faces are continuously updated and his/her ID is removed from *face_list* to avoid false positives. The ID is then added back to *face_list* when the person leaves the FoV of the network.

Algorithm 4 The people re-identification by face recognition algorithm.

INPUT:

- face_list - a set of known faces
- track_list - a set of tracks in the system

```
1:  procedure REIDENTIFICATION(face_list, track_list)
2:      for each track $\in$ track_list do
3:          track.face_images $\leftarrow$ track.face_images $\cup$ observed face images
4:          if track.face_ID has not been assigned then
5:              result $\leftarrow$ classify(track.face_images, face_list)
6:              if result.confidence $<$ threshold then
7:                  continue
8:              else
9:                  if result is known person then
10:                     track.face_ID $\leftarrow$ result.face_ID
11:                     result.face_images $\leftarrow$ result.face_images $\cup$ track.face_images
12:                     face_list $\leftarrow$ face_list $\setminus$ result
13:                 else
14:                     track.face_ID $\leftarrow$ new face_ID
15:         if track is not alive then
16:             track_list $\leftarrow$ track_list $\setminus$ track
17:             face_list $\leftarrow$ face_list $\cup$ (track.face_ID, track.face_images)
```

5.2.5 Bayesian Inference-based Face Classification

In order to reject failures, the system integrates recognition results as they come from the nodes in the network. The simplest way to perform the integration is thresholding, i.e. accumulate results for a fixed number of frames using as decision algorithm a majority voting. However, this simple approach does not take into account the differences in confidence of the single observations, affecting the final classification accuracy.

In this work, we propose a reliable classifier of the observed faces based on a Bayesian inference-based face classification. We used $p(x_{ij})$ as the probability of the i-th track to be of the same identity as the known face j. As special cases, $p(x_{i0})$ is considered as the probability that the i-th track is a person seen for the first time from the system, while $p(x_{0j})$ is considered as the probability that the person with identity j is not yet tracked by the system. In this way, we can define the following constraints:

$$\sum_{k=0}^{N} p(x_{kj}) = 1 \quad (j \neq 0) \tag{5.1}$$

$$\sum_{k=0}^{M} p(x_{ik}) = 1 \quad (i \neq 0) \tag{5.2}$$

where N and M are respectively the number of tracks in the system and the number of known faces. The probabilities can be represented as a table (see Fig. 5.5). The system updates this probability table with face images observed from people tracks. Using the Bayesian theorem, the posterior probability of $p(x_{ij})$ under an observation y is:

$$p(x_{ij}|y) \propto p(x_{ij})p(y|x_{ij}) \tag{5.3}$$

Let x_i^t be the face observed by the track i at time t, x_j^k the k-th face image of the j-th known face and $F(\cdot)$ the face feature extraction function, we can define the distances d_{ij} using Equation 5.4. We use those distances as observations.

$$d_{ij} = \min_k \|F(x_i^t) - F(x_j^k)\|, \tag{5.4}$$

The likelihood function $p(y|x_{ij})$ is modeled by randomly selecting wrong and correct pairs of faces from the LFW [46] dataset. We selected 60000 and 30000 samples of negative and positive pairs, respectively. We show in Figure 5.6 the distribution of the distances between the selected pairs. We fitted a skew normal distribution to each

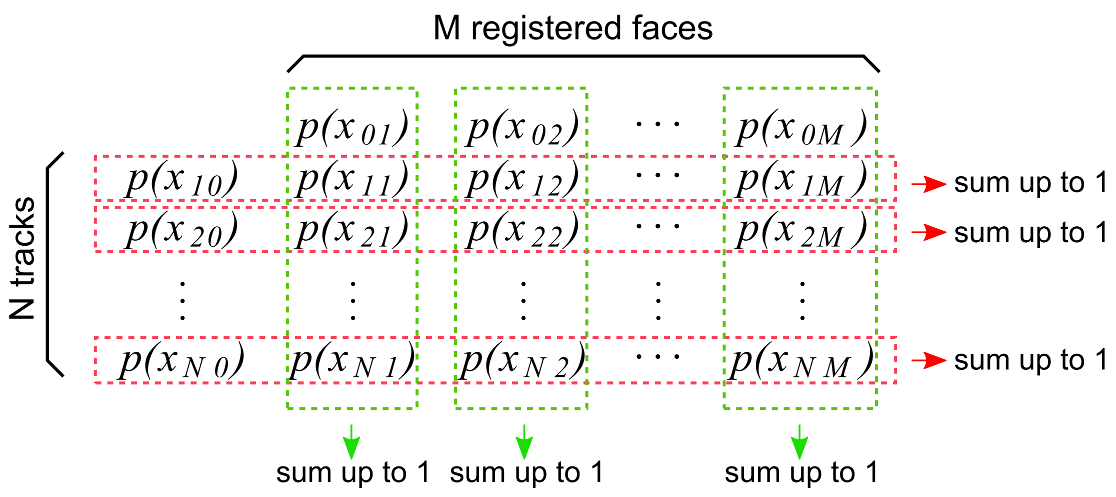

M registered faces
N tracks
$p(x_{01})$ $p(x_{02})$ $\cdots$ $p(x_{0M})$
$p(x_{10})$ $p(x_{11})$ $p(x_{12})$ $\cdots$ $p(x_{1M})$ sum up to 1
$p(x_{20})$ $p(x_{21})$ $p(x_{22})$ $\cdots$ $p(x_{2M})$ sum up to 1
$p(x_{N0})$ $p(x_{N1})$ $p(x_{N2})$ $\cdots$ $p(x_{NM})$ sum up to 1
sum up to 1 sum up to 1 sum up to 1

Figure 5.5: The posterior probability table. The element at (i, j) is the posterior probability that the i-th track has the same identity as the j-th learned face. Rows and columns are iteratively normalized so that they sum up to 1 by using Shinkhorn iteration.

distribution using a maximum likelihood estimation and gradient descent method (they are indicated as solid lines in Figure 5.6). We denote such distributions as $N_p(x)$ (positive distribution) and $N_n(x)$ (negative distribution). Finally, we define the likelihood $p(y|x_{ij})$ from d_{ij} as following:

$$p(y|x_{ij}) = \begin{cases} N_p(d_{ij}) & (j = i) \\ c \cdot N_n(d_{ij}) & (j = 0) \\ N_n(d_{ij}) & otherwise \end{cases} \tag{5.5}$$

using c as a constant to model the tendency that an observed face image is produced from an unknown person. Using a larger c, the framework tends to classify the observed face as an unknown one. In this work, we fixed $c = 1$, but the system works well with most values of c.

After updating the probability table using this computation, it usually violates Equations 5.1 and 5.2. For this reasons, we normalize each row and column by applying the Shinkhorn iteration [108]. The algorithm first normalizes the row and then normalizes the columns until they satisfy Equations 5.1 and5.2. Given the final probability table, if $p^(x_{ij}) = \max\{p_{x_ij}\}$ is larger than a threshold, we classify the i-th track as the j-th known face. The threshold we use for this work is 0.95.

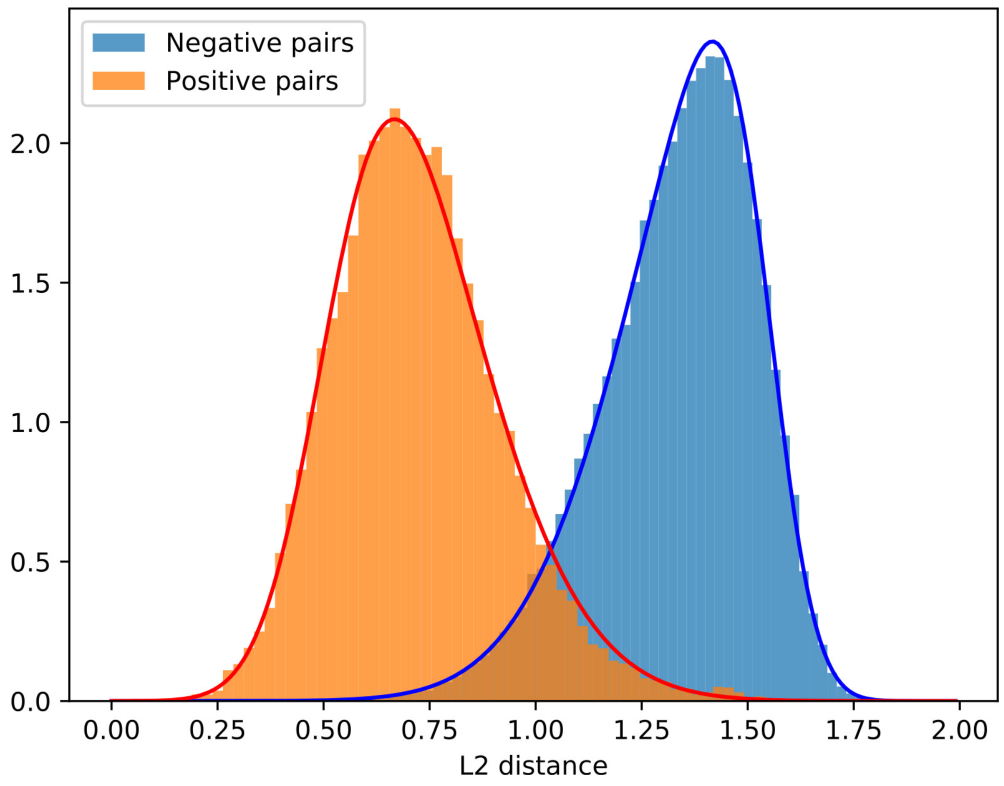


Figure 5.6: The distributions of L2 distance of the positive and the negative face pairs. The solid lines indicate the skew normal distributions fitted to the distributions.

5.3 Cooperation with a Mobile Robot

The proposed system is modular and built on top of the Robot Operating System (ROS) middleware. As a result, it is easy to make a robot cooperate with the network in order to perform Human-Robot-Interaction tasks. As a demonstration, we implemented a task in which a mobile robot based on a Turtlebot2 (see Figure 5.11) is asked to deliver a cup to a certain person. The robot navigates through the environment by using the ROS Navigation stack[2] over a pre-built map of the environment. In order to deliver the cup, the robot should know the transformation between its reference system to the camera network's one. To compute this, we manually aligned the robot map to the subset of the points in the range [0.1 - 0.3m] obtained by the merged point cloud of the camera network (see Figure 5.7). This passage could be completely automatic by using the Iterative Closest Point (ICP) algorithm together with an initial guess that could be obtained with an initial alignment based on the map features. Figure 5.8 depicts the cooperation between the camera network and the robot.

[2]`http://wiki.ros.org/navigation`

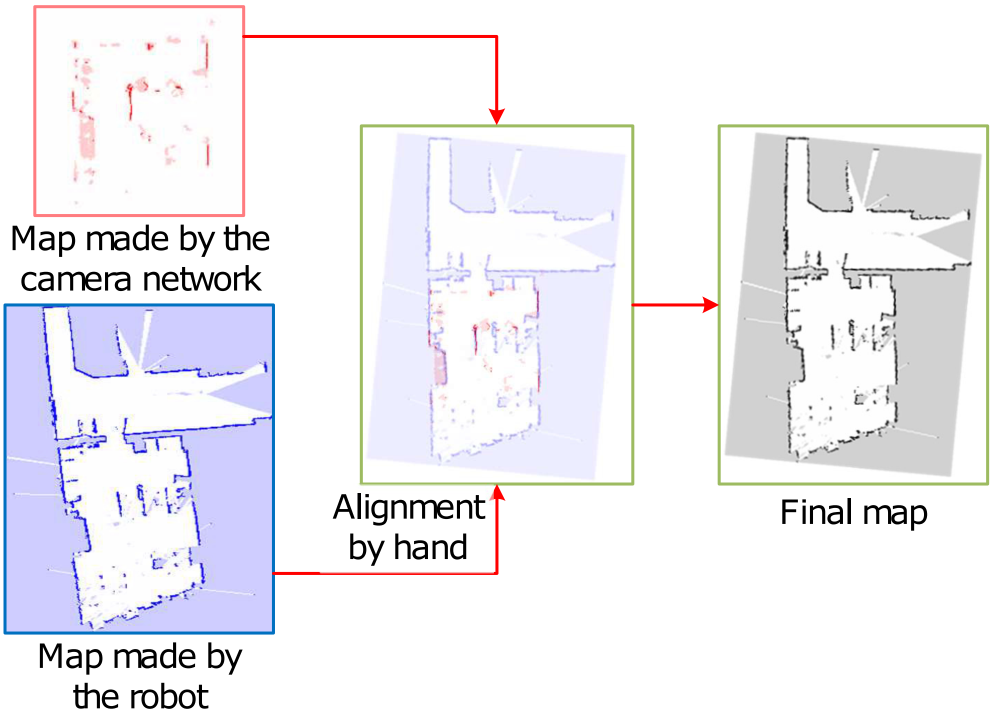


Figure 5.7: Aligning an environmental map made by the robot to the one made by the camera network. An occupancy grid map is created from point clouds obtained from the camera network, and then the maps are manually aligned.

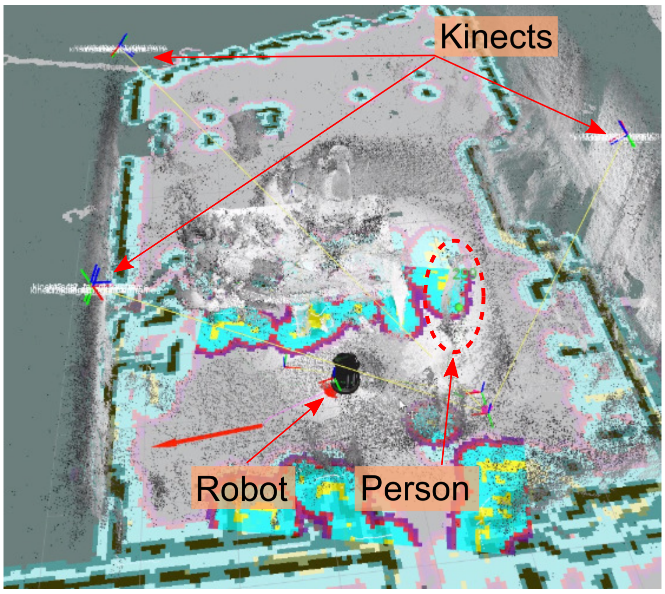


Figure 5.8: The integration of the system and a mobile robot. We can see the robot, the point clouds obtained from the Kinects, and the tracked person in the same view.

5.4 Experiments

5.4.1 Person Re-identification Experiment

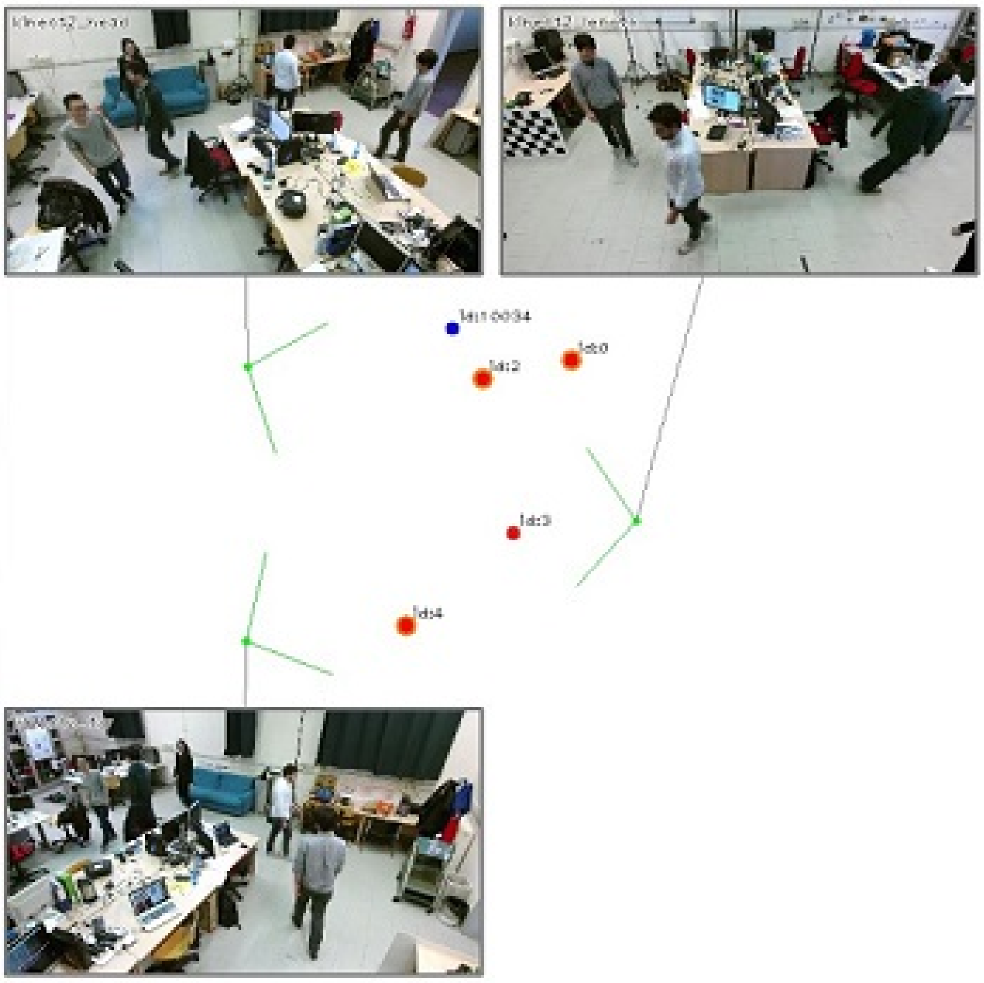


Figure 5.9: A snapshot of the experiments. Three Kinects are placed so that they cover the environment. The dots indicate the tracked people.

In order to evaluate our performance, we considered two RGB-D sequences involving six subjects recorded in different days. Four subjects appear in both sequences, but they changed clothes and, in some cases, hairstyle. Thus, appearance based re-identification would fail to re-identify them. In Figure 5.9 a snapshot of the experiment is visible. We used a three-Kinect v2 camera network that covers the scene. Table 5.1 shows some statistics about the dataset. During the recordings, the subjects moved out from the network's FoV causing the system to loose their tracks 49 and 58 times in the two sequences, respectively.

For the sake of comparison, we applied our Bayesian-inference based classification together with a state-of-the-art face recognition algorithm using landmark detection with SIFT features [87] and simple thresholding as classification approach. About the time for the execution of our approach, the face detection and extraction took about 15 msec per frame on each distributed node, while the master computer performed the re-identification in about 10 msec. The network was composed of consumer laptops with high-end i7 CPUs.

Table 5.2 describes the re-identification performance we achieved. With *success* and *failure* we indicate the number of tracks successfully and wrongly recovered. As it can be seen, the sum of frames considered (*failure* plus *success*) is different with the different methods. This is due to the fact that *OpenFace* discards a face when it does not have sufficient landmarks on it, while [87] yields more face features. Our proposed approach significantly outperforms the state-of-the-art.

Using the thresholding approach, the decision is made observing a fixed number of face images. This causes a worse failure rate, since this technique does not take into account the distances of the different re-identification scores. Nevertheless, the proposed Bayesian inference classifier waits to classify until a significant difference is seen. Moreover, it converges quickly as soon as the difference is noticed. This results in better re-id rate and better re-id time than the thresholding technique. In Figure 5.10 we show some examples of the results we achieved on both sequences. The system is able to re-identify most of the subjects in the sequences regardless their appearance (different clothes or hairstyle). In the same Figure, we also show the only recognition error we encounter using the Bayesian inference approach. The last column of Table 5.2 shows the problem of the face-based re-identification approaches. Indeed, the *recover_rate* is the number of tracks successfully recovered among all of them. Our approach raises the recover rate from 31% to 56%, but it is powerless for most of the tracks of the sequences. The reason is that the re-id approach relies on the face and, therefore, it cannot re-identify a track as long as his/her face is not visible. In order to overcome this problem, it is possible to add cameras to the network and combine the face re-id with other soft-biometric features to exploit the best of both methods.

Table 5.1: Long-term re-identification dataset summary

	duration [s]	# of subjects	# of lost
Seq. 1	162	6	49
Seq. 2	218	6	58

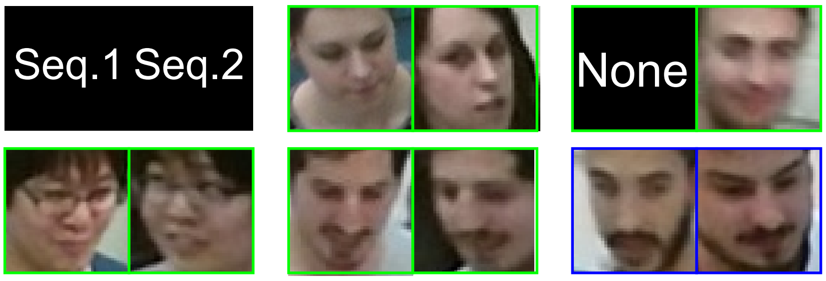


Figure 5.10: Our long-term re-identification results. In green, some examples of successful re-identification. In blue, the only re-identification error done using our proposed approach.

5.4.2 Experiment with a Mobile Robot

In order to demonstrate the possibility of building Human-Robot-Interaction applications using our system, we implement an experiment using a Turtlebot2 platform. The task assigned to the robot is to navigate though the environment and deliver an object (a cup) to a specific person. The camera network is trained with a couple of images and names (including the target person's one). As soon as the camera network sees and re-identifies the target person, the robot starts to navigate towards his position to complete his task (see Figure A.1).

Figure 5.12 exemplifies the lab setting of the experiment. The camera network is composed of three sensors and three persons are involved. At the beginning of the experiment, each person stares in front of a sensor. The camera network was trained with six different people and we successfully repeated the experiment three times changing the target each time.

In Figure 5.13 we show some pictures about the experiment. The person stares at the sensor and the system correctly re-identifies him as the target person (see Figure 5.13 (a)). As soon as the re-identification successfully ends, the system communicates his position and his name to the robot which moves to him (Figure 5.13 (b)). The robot arrives to his destination and communicates to the person to take the cup addressing him with his name (Figure 5.13 (c)) and the person takes the cup (Figure 5.13 (d)).

Table 5.2: Evaluation of Re-identification Accuracy

	success	failure	re-id time	recover rate
[87]	34	54	2.12	0.318
ours w/o Bayesian	51	7	4.59	0.477
ours w/ Bayesian	60	1	4.24	0.561

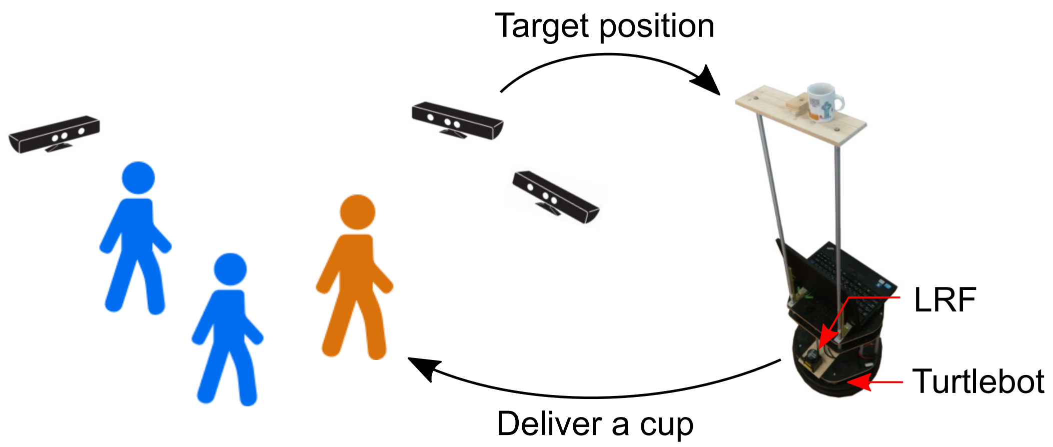


Figure 5.11: The experimental setup. Three persons are involved, in orange the target person. The system communicates the position of the target person to the robot which delivers the cup to him.

In all the experiments, the system successfully identified the target person, and the robot delivered the cup to the target person. The experimental result shows that the proposed system can be integrated with a mobile robot system, and it allows a rich HRI.

5.5 Conclusions and Future Work

In this chapter, we proposed a person re-identification system for camera networks. Our solution addresses the main problem of nowadays people tracking libraries which is the re-association of the people track ID (and name, when it is available). The system is implemented on top of *OpenPTrack*, an open-source people detection and tracking library and uses *OpenFace* to recognize the extracted faces. The recognition is done by exploiting multiple faces using a proposed Bayesian inference-based classification algorithm that allows reliable re-identification. The system has been evaluated using two RGB-D sequences shot in different days involving several people. Moreover, we demonstrated the integration of our system with real robots.

While the system shows great improvements with respect to the state-of-the-art, the visibility of a face in a camera network is still a difficult problem to solve, especially when cameras are mounted in high positions. In order to deal with this problem, we are planning to merge our solution with other long-term re-identification methods which use soft-biometric features as bone lengths or geodesic features.

Another future work is to prove the person re-identification capabilities of our approach using a wider and openly available dataset comparing our results with other

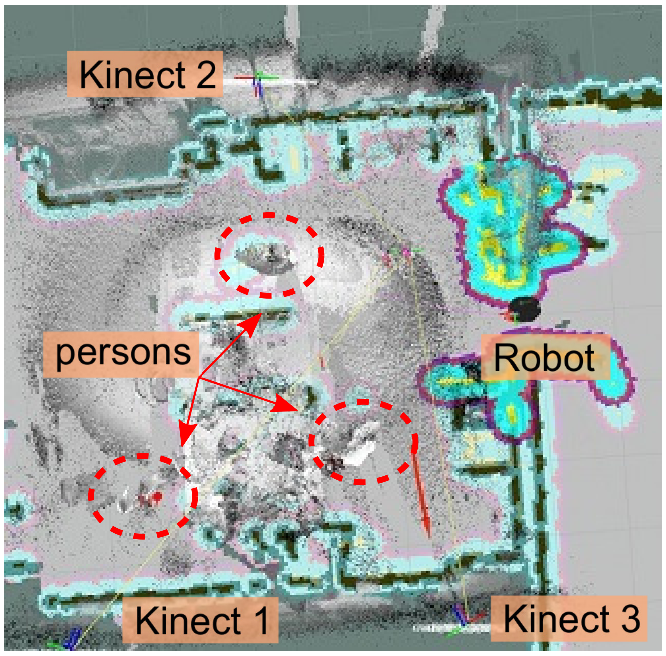


Figure 5.12: A top-view of the interactive map while the experiment was running. We used three sensors and three persons were involved in the experiment.

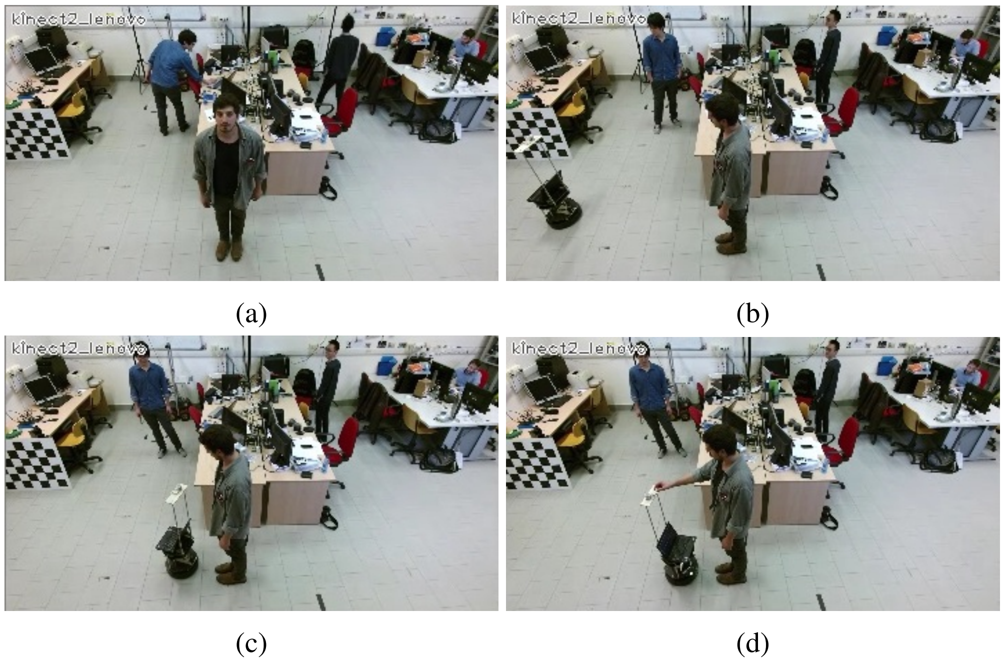


Figure 5.13: Some examples of the cup delivery experiment. (a) shows the person staring at the camera while the system re-identifies him. (b) shows the robot who is approaching his target, while (c) and (d) show, respectively, the robot calling the name of his target and the target who is taking the cup

long-term people re-identification approaches.

Camera networks provide an effective solution to detect different useful features as body pose or person identity. This rich and reliable information can complement the information gathered by a mobile robot boosting the robot's performances. Indeed, the camera network can feed this information about the humans acting in the environment to improve the robot perception as proved in this chapter. Nevertheless, camera networks are not always usable because of privacy problems or installation difficulties. For this reason, Chapter 6 describes a novel contribution for detecting fallen people in a home environment in which all perception and data processing is done on board of a single mobile robot.

Chapter 6

Fast and robust fallen people detection from a mobile robot

Recent studies [22, 43] are proving that the population pyramid is turning upside down. People who are aged 65 and over in 2015 (i.e. 8.5% of the entire population) will increase to 16.7% by the end of 2050. This is why new needs are requested to allow those people to maintain an active social life at their own home. While the number of social health care facilities as well as the number of personal caregivers are not sufficient to satisfy those needs, social robots have been studied to fill this gap. In recent years, many promising results were proposed in this fields as the new Softbank's Pepper or new research platforms as the health care robot Pearl [88], Hobbit [28, 29], Max [41], ASTRO [20] or our prototype Orobot [14] described in Appendix A. The main topics proposed by such works are not just about keeping a safe home environment by monitoring anomalies, but also to enable robots to be friendly companions able to enhance the social life of the elderly people without invading their privacy. While camera networks as the ones presented in Chapters 3 and 4 could be effectively used to find a fallen person or also to capture the act of falling, they are seen as intrusive by the elderly people. Moreover, a camera network installation is most times expensive and not possible in all the scenarios.

In this context, this chapter presents our solution to one of the most common source of harms, i.e. undetected falls [63]. Indeed, while it is very unlikely for a mobile robot alone to capture the act of a fall while it patrols, it can be of paramount importance to detect people lying on the floor in time to raise an alarm.

The contributions of the chapter are summarized as follows:

- we propose a pure-3D, real-time approach to detect fallen people lying on the floor which is suitable also for cluttered and dimmer scenes;

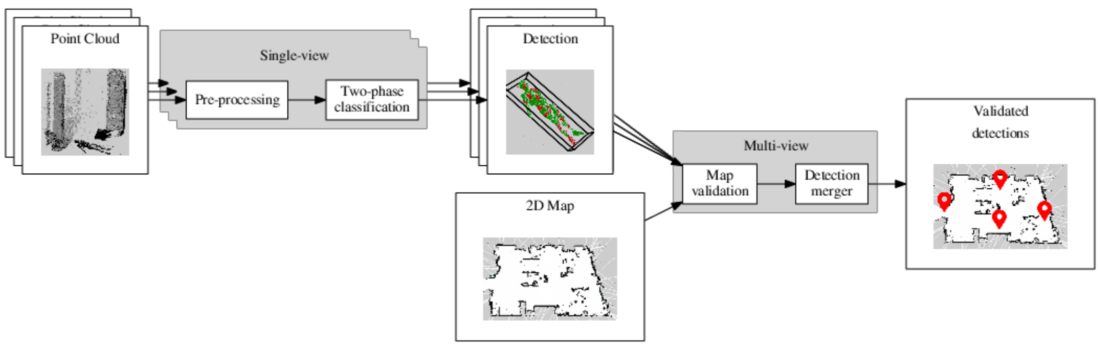


Figure 6.1: An overview of the proposed algorithm. The approach can be split into two different modules running in parallel. The *single-view* detector is in charge of detecting people lying on the floor, while the *multi-view* rejector takes advantage of the 2D map and the multiple view-points to reject false positives. The final output can be summarized as the robot 2D map enriched with the detection of fallen people indicated with red placeholders.

- a novel way to suppress false positives by integrating two basic robot functionalities: 2D navigation and mapping. The robot knows where it is in the room and therefore can suppress false positives taking advantage of multiple viewpoints;
- we registered a new dataset[1] for our experiments and we released it to provide benefit and a baseline for future comparisons. The dataset consists of static and dynamic sequences composed of 15 different persons shot in 2 different environments. For a description, see Appendix B.

The chapter is organized as follows. In Section 6.1 we review the literature about fallen people detection and people detection in general. In Section 6.2 we outline our novel algorithm. Firstly, we give an overview of the entire system and then we analyze the single-view approach and the integration with the robot functionalities. Section 6.3 shows the results we achieved using the IAS-Lab Fallen People dataset(described in Appendix B), while Section 6.4 draws our conclusions and future works we will focus on.

6.1 Related Work

People detection, also known as pedestrian detection, is one of the most studied problems in the Computer Vision community. Since a couple of years, the wide success of Deep Neural Networks (DNN) are changing the way we approach to this problem. In particular, several works [2, 13, 118] showed great results in detecting people and

[1]`http://robotics.dei.unipd.it/117-fall`

estimating their body poses. This kind of algorithms perform well also at detecting people lying on the floor, given the nature of the training done by the Neural Networks described. On the contrary, they require RGB images and, therefore, they cannot work in dimmer scenes, a common situation a mobile robot has to face while patrolling home. Furthermore, such works and in general DNN-based solutions, require high-end GPUs to work in real-time. This is a problem for a mobile robot which should be as light-weight as possible to enhance battery life. For this reason, equipping an high-end Graphical Processing Unit (GPU) on-board is most times unfeasible. For these reasons, DNN-based solutions are considered off-topic for this kind of application. Our approach is based on the point cloud built from RGB-D data, which is robust to illumination condition and computationally feasible. We take advantage of two recent semantic segmentation approaches [120, 121], to extract fast features computed on three-dimensional patches of clusters and learning contextual information among them using Conditional Random Fields (CRF) and 3D Entangled Forests, respectively.

As stated before, falls are the main source of harm for lonely elders at home. For this reason, the literature provides different solution to detect them. Mostly, they require the elders to wear a specific device. Those techniques are very popular given the recent disruption of open-source platforms which are powerful, small and modular at the same time [85]. Usually, such wearable devices embed accelerometers [10, 60, 61]. While those works permits to get useful motion data, they suffer from the difficulty to distinguish normal actions as sitting or lying down with respect to a fall. Moreover, they require the elder to collaborate to wear the device, situation which is not always possible to check or to ensure. A common solution to such problems consists of installing environmental sensors as microphones [89], infrared, vibration sensors [124] or cameras for people detection and tracking as shown in recent works [24, 35, 119] and commercial systems[2]. Such systems represent an effective solution to the falls detection problem. Nevertheless, they require special installations and are less accepted by elderly, who often see the cameras as an invasion to their privacy. The recent growth of Ambient Assisted Living (AAL) projects [14, 20, 28, 29, 41] proves that robots could be an alternative solution to assist elderly at home.

To the best of our knowledge, there exist just few approaches to detect people lying on the floor (i.e. [81, 113, 116]). In particular, [116] and [113] are similar to our purposes, since they propose solutions suitable for mobile robots. In [116], the authors tried to extract fallen people lying on the ground by using just RGB images. They exploited a deformable part-based model to work with multiple viewpoints to

[2]http://www.nively.com

make lying people detection invariant to the viewpoint. The authors of [113] use a similar approach to our work. They proposed a pipeline working on depth images to find candidates using an Euclidean clustering segmentation. The candidates are then layered along an axis to be more robust against occlusions and classified by means of SVM using Histograms of Local Surface Normals. The problem of this approach is the segmentation. Indeed, it is particularly difficult to segment people if those are near furniture. In our proposed approach we address this problem by using two classifiers. Unfortunately, neither the code or datasets used to validate the approach in [113] are available to compare our approach to theirs. In [81], people lying on the floor are detected by looking for their heads using RGB-D data. The work seems to require that the head of the subject is always visible and has not been tested in real-world scenarios with cluttered scenes. Remarkably, none of the approaches described took advantage of the knowledge stored in the mobile robot used, i.e. the local 2d map used for navigation purposes and the availability of multiple viewpoints.

6.2 Approach

Figure 6.1 describes an overview of the approach for detecting fallen people we are proposing in this chapter. The pipeline can be split in two sub-modules, (i) the single-view detector in charge of detecting people from single frames and (ii) the false positive rejection phase. The data used by the single-view detector is the three-dimensional point cloud generated using RGB-D data and the calibration of the sensor. Firstly, we pre-process the input cloud in order to cut all the non-important information above a Region Of Interest (ROI). This region corresponds to all the points with a fixed height with respect to the ground floor. Then, the resultant point cloud is segmented into small patches of voxels with similar appearance. At this point, we take advantage of a two classification phase. The first one classifies each patch as part of a person or not. This phase is important so that we can use Euclidean Clustering just on the positive patches to obtain a fallen person cluster invariant to occlusions. Finally, the second classifier is in charge of discriminating whether a cluster contains a person or not. The single-view classifier proved to be robust against clutter and performs well in different illumination conditions. Nevertheless, in order to further improve the overall performance, we exploit the 2D map and the multiple viewpoints available from the mobile robot to reject false positives.

6.2.1 Patch-based Detection of Fallen People

Each point cloud has to be pre-processed in order to reject noisy points and useless information that can compromise the runtime performance of the overall algorithm and generate false positives. The point cloud is filtered to contain just the floor and the points between it and a parallel plane with an height of about 0.7 m. The floor is then removed since it is not interesting to our purposes using a RANSAC based segmentation [30]. Since we are removing the floor while the robot is moving, the robot motion may affect the estimated floor plane. To make more robust the floor removal, we estimate two floor planes, (i) the first one using the points on the first half of the cloud close to the robot and (ii) the second one on the second half. We tuned the distance considered with real-time tests and we choose a distance of 3 m that proved to be a good split. Unfortunately, when dealing with ToF cameras as the Kinect v2, the sensor used by our prototype Orobot (described in Appendix A), we need to deal with sensory noise as flying pixels. To this end, a soft statistical outlier removal is applied using a set of 50 neighbors and 0.3 as standard deviation.

The single-view algorithm is based on two recent works in the field of semantic segmentation of objects and scene structures [120, 121]. The algorithm can be split in the following four subtasks:

1. over-segmentation of the point cloud in 3D patches with similar appearance;
2. classification of the 3D patches as part of a person lying on the floor, or not;
3. clusterization of close positive patches using Euclidean Clustering;
4. classification of the clusters as positive (i.e. contains a lying person) or negative

The proposed algorithm proved to be robust to occlusions as demonstrated in the Experiments we made (explained in Section 6.3).

The algorithm used for the over-segmentation into homogeneous 3D patches is the Voxel Cloud Connectivity Segmentation (VCCS) [82]. Figure 6.2 shows an example of the application of the VCCS algorithm on a prefiltered cloud. The goal of this approach is to preserve the edges of the scene and objects boundaries, looking for patches that describe the same point cloud, but drastically reducing the amount of data. The algorithm requires a set of parameters, we used the following:

- 0.06 as voxel resolution. This parameter affects the number of voxels to be generated. We chose a good trade-off between the computational time to compute the patches and to obtain patches with sufficient points;

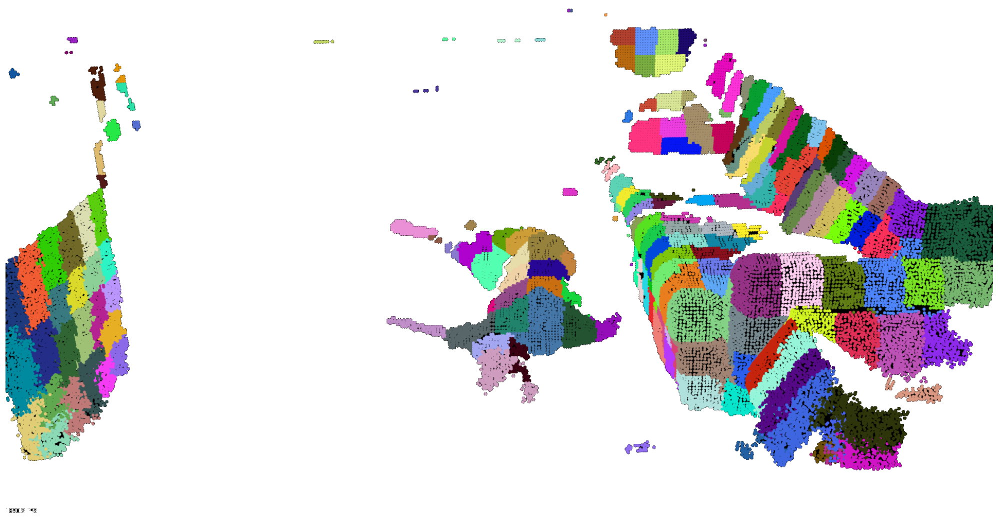


Figure 6.2: The first step applied by the single-view algorithm. The pre-filtered point cloud is divided in patches which clusterize similar structures. Each patch is visualized with a random color.

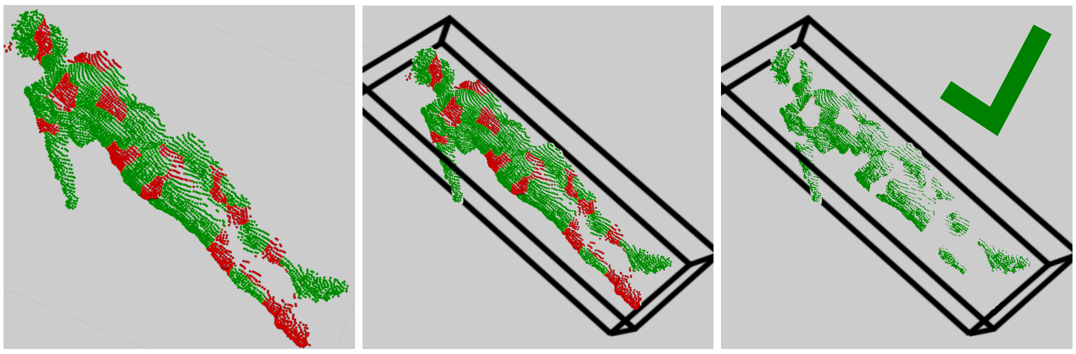
- 0.12 as seed resolution. This represents a good trade-off between over-segmenting also thinner parts as arms or legs and to obtain big patches;
- 0.0 as color importance. We do not want to rely on the RGB information to allow our algorithm to work also in dimmer scenes. Therefore, voxels do not consider the color information;
- 1.0 as spatial importance. This parameters has been kept as suggested in [104];
- 4.0 as normal importance. This parameters has been kept as suggested in [104].

Each three-dimensional patch generated by the VCCS algorithm is then classified as positive (i.e. part of a lying person) and negative (i.e. not part of a lying person). For this reason, we associate to each patch a feature vector $\mathbf{x_1}$. The features have been chosen as suggested by recent semantic segmentation works [120, 121]. We left out the color features, considering just the the geometric ones. We calculated λ_0, λ_1 and λ_2 from the scatter matrix of the patch, while others are calculated from the Oriented Bounding Box (OBB) obtained including all the patch points. Table 6.1 shows the list of chosen features.

The calculated feature is passed to a binary Support Vector Machine to classify whether the patch belongs to a person or not. The SVM uses a Radial Basis Function (RBF) whose parameter have been selected using grid search and k-fold validation. The final misclassification cost C has been set to 62.5, while γ to 0.51. As shown in

Features	Dimensionality
Compactness (λ_0)	1
Planarity ($\lambda_1 - \lambda_0$)	1
Linearity ($\lambda_2 - \lambda_1$)	1
Angle with floor plane (mean and std. dev.)	2
Height (top, centroid, and bottom point)	2
OBB dimensions (width, height and depth)	3
OBB face areas (frontal, lateral and upper)	3
OBB elongations ($\frac{height}{width}, \frac{depth}{width}, \frac{height}{depth}$)	3
Total number of features	16

Table 6.1: The features used by the first classifier for each 3D patch.



(a) (b) (c)

Figure 6.3: The last three steps performed by the single-view algorithm. (a) shows the outcome of the SVM classifier. In red, the negative patches, while in green, the positive ones. (b) shows the Euclidean clustering of the positive patches, while (c) shows the application of the second SVM classifier on the OBB of the cluster.

Figure 6.3(a), at this point a scene could be composed by a set of positive and negative patches. This phase enable us to reject the clutter and to clusterize the possible persons lying on the floor. Figure 6.3(b) and (c) show the idea behind the next two steps performed to find the fallen people location.

Differently from [113], two different sets of patches (positives and negatives) enable the possibility to apply the Euclidean clustering without the risk to include clutter that may cause a cluster to fail the recognition. Firstly, some positive patches can be easily recognized. Indeed, if a cluster presents a density of false positive patches particularly high, it can be filtered out. Then, the negative patches are rejected, and the Euclidean clusters are extracted from the point cloud composed only of positive patches using a large distance threshold of 1.0 m.

The OBB of each cluster is calculated. Therefore, the number of positive patches and the dimensions of its OBB are representative whether a cluster contains a person or not. Each cluster is encoded with a feature vector $\mathbf{x_2}$ of size 9. Table 6.2 shows the complete list of components in $\mathbf{x_2}$. One of the most important feature has proved to be the sample distances to the hyperplane returned by the former SVM. They have been organized in a 4-bins histogram for the distance intervals $[0, 0.25)$, $[0.25, 0.5)$, $[0.5, 1)$, $[1, \infty)$. Each histogram bin is filled with the positive patches whose distance falls in the correspondent interval. For this reason, this histogram provides 4 additional features (one per interval). We used another binary SVM classifier with RBF kernel trained with the set of $\mathbf{x_2}$. Again, grid-search and k-fold validation brought to choose a misclassification cost C of 312.5 and the bandwidth γ of $2.25e-3$.

Table 6.2: List of features calculated for each 3D cluster and their dimensionality.

Features	Dimensionality
OBB dimensions (width, height and depth)	3
Number of positive patches	1
Percentage of positive patches	1
4-bin histogram of positive patch confidences	4
Total number of features	9

6.2.2 Map Verification

The single-view algorithm has been tested with the Kinect v2 sensor used by the Orobot prototype (see Appendix A). The robot uses a 2D map of the environment built in advance using the equipped laser range finder (LRF). Exploiting this map the robot computes a collision-free plan taking into account static obstacles as walls or furniture. In order to prevent collision with dynamic obstacles as persons or chairs or new objects in the environment, the robot builds a dynamic map from the LRF data while it is moving which is used to locally change the static plan. The static map is used for our purposes, since it contains useful information to reject false positives. Indeed, true positives cannot be located inside walls or where large furniture items are located. For this reason, all the single-view detections are checked with those constraints. The static map is defined as a set of cells $S = \{Cell_i, 0 \leq i \leq N\}$, where:

$$Cell_i = \begin{cases} -1 & \text{unknown content} \\ 0 & \text{free space} \\ 0 < n \leq 1 & \text{probability to be occupied} \end{cases} \tag{6.1}$$

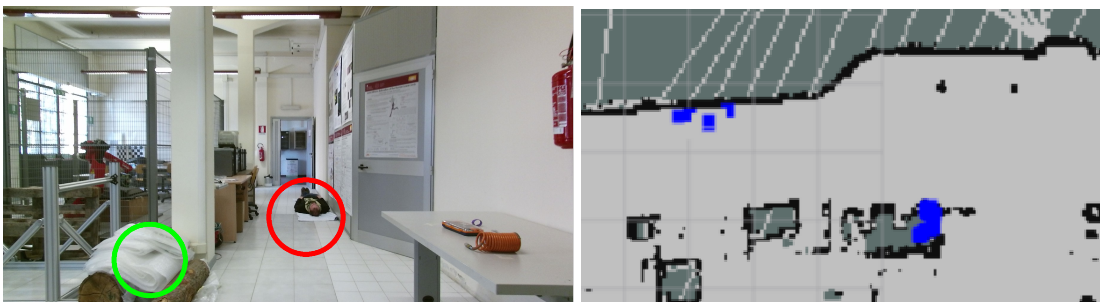


Figure 6.4: The false positive rejection performed by using the map verification phase. (a) a furniture item raising a false positive (in green), while (b) describes the detections performed by the algorithm. The detections relative to the false positives fall on the occupied space of the map and are, therefore, rejected.

State-of-the-art 2D SLAM algorithms as [39, 40] provide the transformation needed to project each single-view detection from the camera coordinate system to the map coordinate system so that it will be located in a cell map $Cell_i$. If the value of $Cell_i$ is -1 or occupied (i.e. $K \leq Cell_i \leq 1$ with $K = 0.30$), the detection is considered as a false positive and then rejected. We are assuming that the map has no unknown space wherever this is reachable by the mobile robot or its sensor. This assumption is always met by maps built in indoor environments for robot patrolling. Figure 6.4 shows an example of successful rejection of a false positive. The single-view detector found a person lying on the floor indicated by the green box raised by a trunk covered with some furniture in our lab environment. The map allows to reject this false positive since that area is already occupied. More importantly, the map verification phase can reject false positives coming from artifacts due to glass walls or mirrors.

6.2.3 Merging Detections from Multiple Vantage Points

Using a moving robot to find fallen people allows to use also another interesting feature that enhances the detection performances. Indeed, while it patrols, is very likely that the robot camera covers many part of the scene multiple times form different point of views. Therefore, the location of the detections obtained by the single-view detector and refined with the map verification can be easily tracked. Indeed, due to different viewpoints, a detection may be a true negative or a false positive. Additionally, experimental data shows that the detection rate of false positives is lower than the one obtained by true positives. Another contribution of this chapter is then to exploit multiple vantage points in order to further improve detection performances.

At this point, the single-view outcomes refined by the map verification are considered as input of the multi-view algorithm that exploits multiple view-points to reject false positives. The output of the algorithm is a set $\mathfrak{P}$ of validated fallen people locations p_i in the map; formally $\mathfrak{P} = \{p_i, 0 \leq i \leq g\}$, using g as the total number of people. Considering a new detection $d = (loc, t)$, where $d.t$ is the timestamp when the detection occurs and $d.loc$ is the fallen person location in the map reference system, we update the set of clusters $\mathfrak{C} = \{C_i : 0 \leq i \leq n\}$ where each cluster C_i is defined as:

$$C_i = \{d_j : ||d_j.loc - d_m.loc|| < \overline{th}, \forall j, m \in [0, k-1]\}, \tag{6.2}$$

where $\overline{th}$ is a threshold which controls when two detections will belong to the same cluster and k is the cardinality of the cluster (i.e. the total number of detections in it).

The multi-view algorithm, implemented in a concurrent thread, updates the set $\mathfrak{P}$ as soon as a new detection arrives and it also deletes the old ones from $\mathfrak{C}$. This is mandatory to maintain a lightweight representation of each cluster $\mathfrak{C}$ and also to discard false positives, whose detection rate is typically low. The algorithm is described in Algorithm 5, using the following parameters: Lines 3-7 are about the detection rejection performed in a cluster, while Lines 8-14 are about the cluster classification as person or not a person checking its cardinality and the detection rate.

In our tests, the best parameter configuration used is shown in Table 6.3. We would like to point out that using a frame rate allows to set a lower $\widehat{n}$, therefore preventing over-fitting on particular environments. The concurrent thread executes the Algorithm every 10 seconds. Figure 6.5 shows the algorithm in action.

Parameter	Value
$\widehat{t}$	60 s
$\overline{th}$	1 m
$\widehat{f}$	1 Hz
$\widehat{n}$	5

Table 6.3: The best parameter configuration from our tests for Algorithm 5 in our tests.

6.3 Training and results

Detecting fallen people is a challenging task. This is also pointed out by the lack of public datasets and methods to solve this problem. In order to allow future comparison and to boost the research in this field, we released the IASLAB-RGBD Fallen Person

Algorithm 5 The concurrent procedure run periodically to find lying people locations exploiting multiple vantage points.

INPUT:

- $\mathfrak{C}$: the clusters of detections (already filtered with the map verification) in the system.
- $\widehat{f}$: minimum detection rate a false positive should have;
- $\widehat{n}$: minimum cluster cardinality to be considered as a true positive;
- $\widehat{t}$: maximum detection age a candidate can have before being discarded;
- $\mathfrak{P}$: set of locations of fallen people found until now.

OUTPUT:

- $\mathfrak{P}$: set of updated locations of the fallen people

```
1: procedure VALIDATE_CLUSTERS($\mathfrak{C}$, $\mathfrak{P}$, $\widehat{t}$, $\widehat{f}$, $\widehat{n}$)
2:     for each $C_i \in \mathfrak{C}$ do
3:         for $j \in [0, k-2]$ do
4:             for $o \in [j+1, k-1]$ do
5:                 if $|d_o.t - d_j.t| > \widehat{t}$ then
6:                     $index \leftarrow$ARG_MIN$(d_o.t, d_j.t)$
7:                     $C_i \leftarrow C_i \setminus d_{index}$
8:         $t_m \leftarrow \min_{d \in C_i}\{d.t\}$
9:         $t_M \leftarrow \max_{d \in C_i}\{d.t\}$
10:        $f_i \leftarrow \frac{||\mathfrak{C}||}{t_M - t_m}$
11:        if $f_i \geq \widehat{f}$ and $||C_i|| \geq \widehat{n}$ then
12:            $loc_i \leftarrow \sum_{d \in C_i} \frac{d.loc}{||C_i||}$
13:            $\mathfrak{P} \leftarrow \mathfrak{P} \cup loc_i$
14:            $\mathfrak{C} \leftarrow \mathfrak{C} \setminus C_i$
15:    return $\mathfrak{C}$, $\mathfrak{P}$
```

Dataset[3] (see Appendix B) as a further contribution.

In this section we will explain how the classifiers described above have been trained and the results we achieved using the aforementioned dataset. The first SVM of the single-view algorithm has been trained and validated computing thousands of patches extracted from frames taken from the static part of the dataset and the *Lab A* sequence. The testing phase was performed on different frames taken from the static part and from the *Lab B* sequence. In particular, the static part provides the positive samples.

[3]http://robotics.dei.unipd.it/117-fall

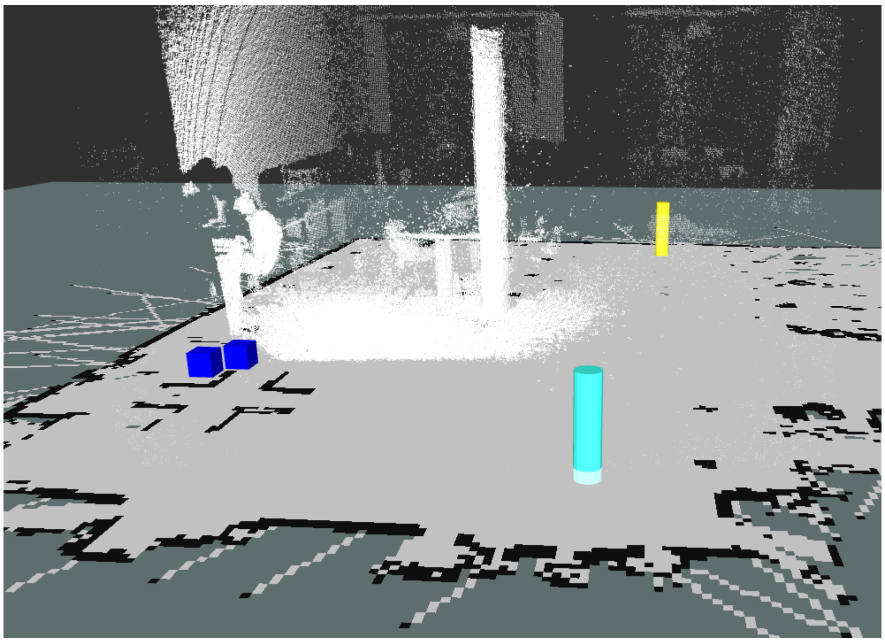


Figure 6.5: An example of the different outcomes using the multi-view approach. The Figure depicts the 2D map overlayed with the input cloud (white points). If the single-view detections (indicated as blue cubes) meet the criteria (time and distance), they are aggregated in one cluster containing a person(randomly colored cylinders)

On the web page of the dataset, the 70-30 train/test split is available. On the other hand, the negative samples have been extracted from the *Lab A* (we extracted 24 frames out of 15932 for training), from *Lab B* (32 frames out of 9391 for testing) and from the NYU Depth Dataset v2 [103] (35 frames out of 1449, for training). This is a wide used dataset for semantic scene understanding and it contains thousands of indoor scenes. We considered also another split: we used the static part and *Lab B* for training we tested on *Lab A*.

For the second classifier we considered an analogous case. We trained and validated on positive samples (in this case, clusters not patches) extracted from the static part, while we extracted negatives from the *Lab A* sequence and tested on samples of the *Lab B* sequence. Analogously, we considered another split, i.e. training on *Lab B* and testing on *Lab A*.

Not only the *single-view detector* but also the *multi-view analyser* has been tested on *Lab B*. Indeed, both *Lab A* and *Lab B* comprehend the entire robot transformation tree. Given that the position of the fallen people in the 2D map is known, this allows to calculate the performance indices automatically by checking if the location of the detected cluster centroid is close (at a distance less or equal to 1 m) to the ground truth centroid of a person position in the 2D map.

The performance on this dataset are measured by means of four common metrics, i.e. the detection *accuracy*, *precision*, *recall* and $F_{0.5}$ score. Let TP, TN, FP and FN be respectively the true positives, true negatives, false positives and false negatives, we can define the metrics as shown in Equations 6.3,6.4,6.5 and 6.6.

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{6.3}$$

$$precision = \frac{TP}{TP+FP} \tag{6.4}$$

$$recall = \frac{TP}{TP+FN} \tag{6.5}$$

$$F_{0.5} = \frac{(1+0.5^2)*precision*recall}{0.5^2*precision+recall}, \tag{6.6}$$

In particular, the $F_{0.5}$ score can be seen as an harmonic average of recall and precision promoting more the last one. This metric has already been used in another similar work by Volkhardt et. al [113]. Additionally, given the lack of available state-of-the-art work regarding fallen people detection, for the sake of comparison, we defined and compared with a baseline based on Euclidean custer extraction. In this way, we will focus on the use of the patches importance also in cluttered scenes. Moreover, we show also the running times of each part of the algorithm to show that it is suitable for running in real-time on mobile robots.

6.3.1 Validation

The methods presented in this chapter have been thoroughly evaluated on both sequences of the dataset described in Appendix B. Since the approach is composed of a double-step classification, we validated both classifiers separately to demonstrate the efficacy of both of them. We evaluated using both *Lab A* and *Lab B* sequences. Both classifiers have been trained using just part of the frames contained in *Lab A*, while they are seeing *Lab B* for the first time. The results are presented for each of the 3 contributions, i.e. the single-view algorithm and the two modules of the multi-view approach, the map rejection phase and the multiple vantage points validation. In the results, we used the baseline (B) in order to provide a useful comparison. B is based on the segmentation on candidates using Euclidean clustering with 0.10 m as distance threshold and 0.06 m as voxel resolution. We also want to point out that the distance threshold of the Euclidean clustering is far low than the one required by our by our approach (1 m). When the candidates are extracted, they are labelled using their OBB dimensions with a pre-trained Random Forest classifier. This choice is fair, since it permits to find a low number of clusters, while maximizing the number of true positives.

In order to find the optimal set of parameters of the RBF kernel used by the SVM, we used grid search and K-fold cross validation. In particular, the misclassification cost C was free to vary in the $[0.1, 500]$ interval with logarithmic step 5, while the bandwidth γ was free to vary in the $[10^{-5}, 0.6]$ interval with logarithmic step 15. For te cross-validation we used $K = 10$. Table 6.4 reports the SVM performances averaged on both splits of the test sets.

Table 6.4: Performances of the classifiers on their test sets. We averaged the results on both splits.

Method	Accuracy	Precision	Recall	$F_{0.5}$
Classifier 1 (C1)	0.90	0.87	0.86	0.87 ± 0.06
Classifier 2 (C2)	0.96	0.97	0.96	0.97 ± 0.02

Tables 6.5 and 6.6 show. As it can be seen, the proposed patch-based method outperforms the baseline not only in precision (i.e. how good is the classifier to predict correctly given one of its prediction), but also in recall (i.e. how good is the classifier to correctly predict given the ground truth predictions). Moreover, the performance are further improved by applying the map verification step.

Table 6.5: Performance comparison on the *Lab A*.

Method	Accuracy	Precision	Recall	$F_{0.5}$
Baseline (B)	0.88	0.65	0.33	0.54
Single-view (SV)	0.87	0.73	0.85	0.75
SV+Map verification (MV)	**0.92**	**0.85**	**0.85**	**0.85**

Table 6.6: Performance comparison on the *Lab B*, never seen before by both classifiers.

Method	Accuracy	Precision	Recall	$F_{0.5}$
Baseline (B)	0.84	0.64	0.26	0.50
Single-view (SV)	0.88	0.80	**0.86**	0.81
SV+Map verification (MV)	**0.90**	**0.87**	0.81	**0.86**

In Table 6.7, we report also the testing result of the multi-view detection merging described in Section 6.2.3. As expected, the method proved to be useful. We tested it using all the eight sequences acquired in the dataset in both environments. It is particularly interesting to notice that the navigation path followed by the robot slightly changes, even if the environment is the same because of dynamic obstacles and the different positions of the lying persons. Analyzing the results, each person on the *Lab A* sequence is always detected exactly once. We detected a false positive (i.e. generated by the trunk depicted in Figure 6.4a) in one of the 4 sequences, while the same false positive is rejected in the other 3. On the other hand, analyzing the 4 patrolling tasks of the *Lab B* sequence (which has never been seen before by the classifiers) we obtained no false positives (all rejected) and the persons are all detected exactly once.

Table 6.7: Performance of the final outcome provided by the algorithm (i.e. single-view, map verification and multi-view) on the *Lab A* and *Lab B* sequences.

Environment	TP/P	FP
Lab A (sequence 1)	4/4	0
Lab A (sequence 2)	4/4	1
Lab A (sequence 3)	4/4	0
Lab A (sequence 4)	4/4	0
Lab B (sequence 1)	4/4	0
Lab B (sequence 2)	4/4	0
Lab B (sequence 3)	4/4	0
Lab B (sequence 4)	4/4	0

In conclusion, Figure 6.6 shows some qualitative examples of the results achieved. The single-view detector is able to locate fallen people also in cluttered scenes (i.e. Figure 6.6(a),(b),(c) and (d)). Also in presence of noisy data that can fool the single-view detector (i.e. reflections due to glass surfaces), or close objects, the final algorithm can reject the false positives (see 6.6(e)(f)). Figure 6.6(g),(h) shows the final locations of the lying people in the two environments.

6.3.2 Runtime Analysis

Table 6.8 shows the running times we achieved using an unoptimzed version of the code with the single-view detector. The multi-view and map verification part are computationally negligible, so we do not report them. As it can be seen, the algorithm is very efficient and it is, therefore, a good choice for a mobile robot. Even with this version the algorithm could be used in real-time with an average rate of 7.72 fps.

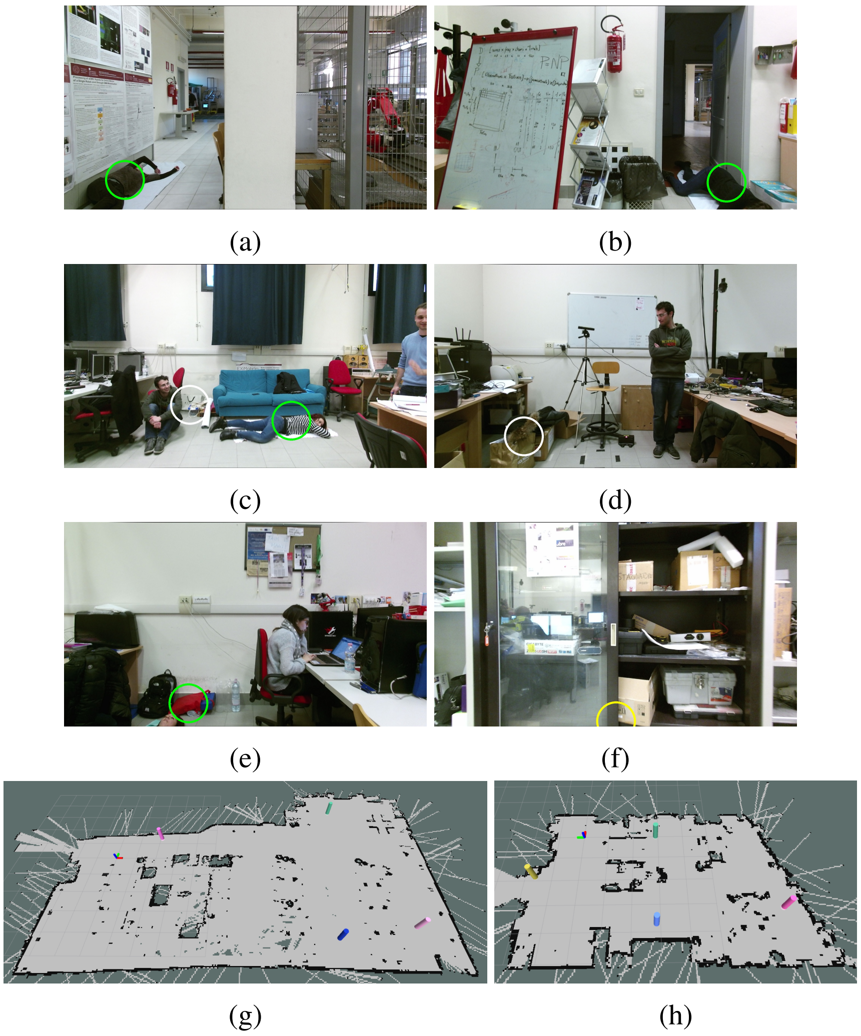


Figure 6.6: Qualitative results on the IASLAB-RGBD Fallen Person Dataset: (a)(b) even if the lying people can be very close to the wall or other scene elements, the *single-view detector* can find them at a high detection rate; (c)(d) the *single-view detector* can discard fake lying people, see the white circles; (e)(f) the *single-view detector* may find some false positives in the presence of clutter (several close objects) or high noise (glass surfaces); (g)(e) the *multi-view analyser* can reject both FP like in (e) thanks to the low frame rate or in (f) thanks to the map validation (yellow circle).

The machine we used to test it is a Dell inspiron 15 7000 which has an Intel-Core i7-6700HQ CPU with 4 cores clocked at 2.60GHz and 16GB of RAM.

6.4 Conclusions

In this chapter, we proposed an efficient and robust system that detects people lying on the floor from a mobile robot. The approach has proven to be effective and invariant

Table 6.8: Average runtimes of the main steps of the proposed algorithm on our test machine (Intel Core i7-6700HQ CPU, 2.60GHz x 4).

Processing Stage	Runtime
Pre-processing and Oversegmentation	10.27 fps
Patch Feature Extraction	105.98 fps
SVM Classification 1 (per patch)	0.84 μs
Cluster Feature Extraction	2639.56 fps
SVM Classification 2 (per cluster)	0.04 μs
Total runtime	7.72 fps

to pose and distance. The system is composed by two novel contributions: (i) a single-view algorithm which is composed by two classifiers that segment the fallen people by rejecting clutter and (ii) a map verification phase and multi-view fusion that improve the performances rejecting false positives obtained from (i). We validated the approach on the IASLAB-RGBD Fallen Person Dataset, presented in Appendix B showing great improvements with regard to a baseline approach. As future work, we would like to provide results also about the ability of the system to semantically segment the person on the floor and not just the ability to detect fallen persons. We also plan to extend the experiments using larger datasets involving multiple people and in real-world environments as houses and apartments.

Chapter 7

Conclusions

Perception of humans is really important for autonomous robots: first, for safety reasons and secondly to enable an effective interaction. In this thesis, we presented how to enrich people perception in camera networks and on board of mobile robots by exploiting RGB-D data. We showed also how robots and camera networks can collaborate in order to overcome their intrinsic limitations, i.e. the static nature of a camera network and the limited perceptive horizon of a mobile robot.

In Chapter 2, we introduced the concept of RGB-D data and the hardware and software technologies to capture and process RGB-D data in real time. This type of data is of paramount importance for robotic systems to enable a smooth interaction with the humans and the environment. The main contribution of the chapter is a novel efficient library (encapsulated in a perception node for ROS) to generate point clouds representing the RGB-D data. By using OpenPTrack as a test-bed, we demonstrated that our work greatly improves the state-of-the-art by tripling the point cloud generation rate and by doubling the people detection frame rate. This contribution enables real-time perception of humans using high-resolution point cloud data.

In Chapters 3 and 4, we presented two different novel approaches for solving the 3D human body pose estimation for RGB-D camera networks which proved to be effective in complementary scenarios. Chapter 3 proposed a depth-only approach. The depth data gathered by the different cameras are fused by an algorithm which computes a synthetic multi-view depth image for the user in the scene. The depth image is calculated from a frontal-view warping of the merged point cloud. The 3D pose of the user is obtained using a single-view state-of-the-art skeletal tracker applied to the synthetic multi-view depth image. The results showed that our approach not only provides good multi-view pose-invariant skeletons, but also improved the single-view quality of the considered pose estimator.

As an alternative approach, Chapter 4 proposed a novel 3D pose estimation algorithm based on RGB-D data, that exploits the camera network by fusing the processed 3D data instead of raw depth data. This is done by associating data at the tracker level and updating a set of Kalman filters, one for each 3D body joint. Experimental results show great improvements with respect to the baseline methods considered. The contribution of this chapter has been also released as open-source in the OpenPTrack v2.0 repository [1]. The system described in this chapter outperforms the results presented in Chapter 3 by allowing the tracking of multiple bodies at the same time with negligible overhead. Moreover, the system do not require data synchronization anymore, allowing the user to build an heterogeneous and cost-efficient network of sensors following his needs and possibilities. On the other hand, the solution presented in Chapter 3 has to be preferred when no light is available since it works only on depth data.

We believe the contributions described can give an important boost to improve the performances of current motion-capture systems and to open new applications of motion capture techniques in commercial sectors in which the current professional marker-based motion capture systems are too expensive. Many researchers are working in order to be able to remove the use of markers, while keeping the overall tracking quality of marker-based solutions and possibly with a low number of cameras. In this way, constraints on the users can be removed and the installation costs of such systems can be reduced, enabling a wider usage of them. Indeed, our work proved to effectively and precisely work relying only on RGB-D consumer sensors like Kinect v.2 and similar. Another scenario that could be explored to enhance our contributions is the possibility of overcome the static nature of the cameras in the network. Indeed, recent advances in Visual SLAM (e.g. [77, 78]) may be exploited to update the extrinsic calibration of the system, allowing the cameras to be mounted on robots, e.g. as a swarm of drones or heterogeneous team of mobile robots. As a result, the system would be able to move the positions of the cameras in order to maximize the information gathered by each single camera and not just by processing the data coming from prefixed locations. Moreover, the system would be also able to adapt itself to new environments regardless of the light conditions, also where common motion capture system cannot be installed.

In Chapter 5 we proposed a solution to the long-term people re-identification problem in a camera network. Many people detection and tracking libraries, as OpenPTrack, do not provide information regarding the identity of the persons, i.e. they assign new IDs whenever a new track is created. Our contribution exploits the face of each person

[1] https://github.com/openptrack/open_ptrack_v2

to recognize its identity and to re-associate the correct ID and name to each new track. The results achieved show great improvements, while highlighting the need of additional soft-biometrics features as the gait or the skeleton lenghts in order to cope with the situation when the face of a person is not visible from any camera. In this chapter, we also demonstrated a successful collaboration between the robot and the camera network in order to identify a target person and deliver an object to him/her. Similar solutions are achievable using the rich information obtained by the works described in Chapters 3 and 4 enabling the robot to perceive the movements of humans and act accordingly.

One of the possible application of the intelligent camera networks we are introducing with the works described in this thesis, is in the field of Ambient Assisted Living. Undetected falls are one of the main reasons of early deaths for lonely elderly people living at home. Chapter 6 proposed a fast algorithm to detect fallen people lying on the ground. The algorithm is composed of a single-view double-classifier enriched with a multi-view false positive rejection phase performed using the knowledge available from a mobile robot as multiple views and the map of the environment. Indeed, while camera networks are able to cover more space allowing also the detection of the act of falling, they are not always possible to be installed and they are usually seen as a privacy invasion by the people who needs assistance at home. The experiments and analysis described proved that the algorithm is robust against clutter, which is the main problem of state-of-the-art classifiers, and the multi-view and map verification phases substantially improved the quality. To cope with the lack of dataset of fallen people, we tested our algorithms on a new RGB-D dataset described in AppendixB recorded on-board our assistive robot, which has been made publicly available[2].

In the future, we can expect that robots and humans will cooperate to perform different tasks: from assembly lines in industry to rehabilitation at home, from cultural activities to ambient assisted living. For this reason, the algorithms used should be effective as well as efficient. We believe that the contributions of this thesis are paving the way to new developments in this direction, providing fast and effective inputs for a richer Human-Robot Interaction.

[2]http://robotics.dei.unipd.it/117-fall

Appendices

Appendix A

Orobot: an open-source prototype for Ambient Assisted Living

Last years have seen a worldwide lengthening of life expectancy [111] and, as a consequence, an increment of advanced assistive solutions for integrated care models. In this context, ICT can enhance home assistance services for elderly people and thus, the economical burden for health-care institutions. Indeed, recent studies report that 89% wish to stay at home for sentimental reasons or because they cannot afford nursing homes [51]. In addition, the shortage of professional caregivers is a well-known issue and relatives or friends must face with emotional distress negatively impacting also their productivity at work [92].

In this perspective, home robots will play a crucial role. Not only they will keep their house safe by monitoring and detecting anomalies or sources of hazards but they can also be companions able to enhance their social life, e.g. by better connecting them with their relatives and friends. Examples of recent projects that have tried to develop such a kind of systems are Hobbit [28, 29], Astro [20, 21] and Giraffplus [23]. The difficulties in creating a robust and reliable prototype and the encountered challenges in computer vision and autonomous robotics have been well-explained in [28].

This Appendix aims at presenting an open-source[1] and practical solution for an autonomous robotic platform for home care. The final goal is to develop a set of artificial intelligence services for indoor autonomous and safe navigation, fall detection, people recognition, speech interaction and telepresence. This platform has been used to validate the works described in Chapters 5 and 6.

In Section A.1 and A.2 respectively, the prototype and the tasks currently implemented are described. We have developed these functions using the *ROS: Robot Oper-*

[1]https://bitbucket.org/iaslab-unipd/orobot

ating System [90] middleware, which provides many algorithms and a standard communication framework. The need for standards and open solutions have been already pointed out in [65]. This project enables research in many fields like scene understanding, human robot interaction, socially assistive/intelligent robotics and sensor integration.

A.1 Hardware Configuration

Our prototype of home robot, shown in Figure A.1, is built on top of a commercial open-source mobile platform, the Turtlebot 2[2], which is already equipped with odometry, a gyro, bumpers, cliff sensors, wheel drop sensors and a docking IR receiver. Furthermore, this platform is a smart choice also because of the available API to interface with ROS and the existence of a worldwide community working with it. For our purposes, we have added an Hokuyo URG-04LX-UG01 2D scanning laser rangefinder and a Microsoft Kinect v2. The former is placed at a height of 15 cm so as to easily detect low obstacles; the latter at a height of almost 115 cm which is the best trade-off for the people detection at difference distances. The robot control is accomplished by a Lenovo Y50-70 laptop with Ubuntu 14.04 and ROS Indigo.

A.2 Software and Testing

The software of our robot is entirely based on ROS. The standard robot's task is the complete visit of the house while registering all events around it. The robot has also the capability of auto-docking, i.e. to autonomously return to the docking station for recharging. The next sections will better describe the important modules we used to achieve the goal of patrolling an house.

A.2.1 Laser Filtering

The raw data acquired with the laser scanner is subject to many inaccuracies due to the technology it is based on. As an example, there are issues with transparent and black surfaces producing frequent outliers. Nonetheless, the laser scanner is the core sensor for the navigation so the less inaccuracies occur, the better the navigation algorithm performs. For this reason, we have adopted an interpolation filter from the ROS packet *laser_filters*[3]. The laser streams data to the *raw_scan* ROS topic; the filter

[2]http://www.turtlebot.com/

[3]http://wiki.ros.org/laser_filters

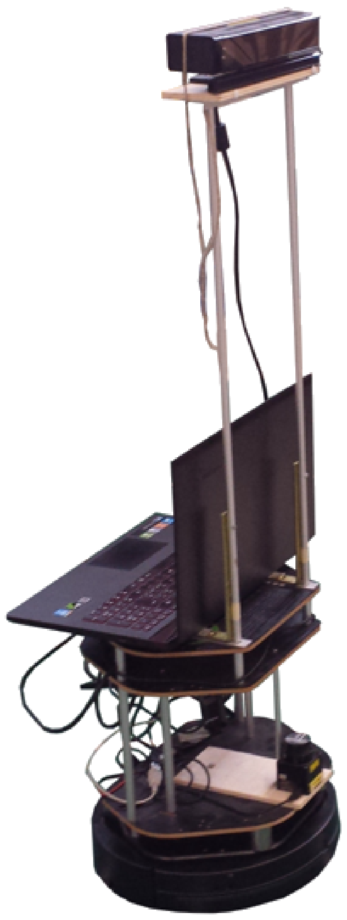


Figure A.1: Orobot: an open-source prototype for Ambient Assisted Living

reads, processes and finally publishes the filtered data to the *scan* ROS topic. This one is subscribed by the robot functionalities. In Figure A.2 the comparison of the same map built with the *raw_scan* topic (left) and the filtered one (right) is shown. In the unfiltered map we can see how the outliers can result in extremely noisy borders. In contrast, the interpolated map shows significant improvements.

A.2.2 Mapping and Navigation

Building a map is a core functionality in mobile robotics. In literature, this problem is known as *Simultaneous Localization And Mapping (SLAM)* and there exist several implementations in ROS. For this purpose, we are exploiting the well-known technique described in [39] and implemented in the *GMapping*[4] ROS package. We have tested it successfully in several natural environments (e.g., office, corridors, homes). Once

[4]`http://wiki.ros.org/gmapping`

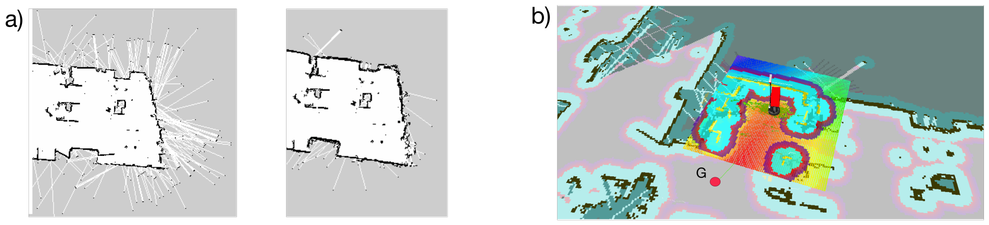


Figure A.2: a) Comparison of the map built with the raw data from the laser scanner (left) and the data filtered with an interpolation filter (right). b) View of the static and dynamic maps while the robot is navigating to the goal *G*.

we have a static map representing the environment, a method for reaching a point maintaining the localization within the map during the robot movement is of concern. The technique used is based on *AMCL* [32] *(Adaptive Monte Carlo Localization)* ROS packet. This algorithm is based on two maps: the static one and the dynamic one which is computed on-line. For reaching a goal *G* (Figure A.2b), it computes a static plan from its initial position to the goal basing the decision on the static map. After this phase, the control of the movements of the robot passes to the dynamic map which is computed in real-time. With this map, the AMCL algorithm checks whether the robot is correctly localized using the current sensor data and it re-localizes the robot. The static plan is then changed in case of dynamic obstacles in order to avoid them. If there are no possible new plans, the robot tries to rotate around itself to see if other paths are available given the dynamic map, otherwise it fails to reach the goal with an error which can be caught and managed.

Appendix B

IAS-Lab RGBD Fallen Person Dataset

In the literature is very difficult to find a dataset for evaluating fallen people detection approaches. For this reason, as one of the contribution of this thesis, we released a new dataset which involves several people in different environments. The dataset has been publicly released[1].

The IAS-Lab RGB-D Fallen Person Dataset consists of several RGB-D frame sequences containing 15 different people. We acquired the dataset in two different lab rooms, the *Lab B* and the *Lab A*. The different frames were taken with the on-board Microsoft Kinect v2 sensor of our Orobot prototype (see Appendix A) while it was patrolling the rooms. While the *Lab A* sequence is shot in a bigger room, the *Lab B* one is taken from a smaller scenario which contains clutter, a sofa and is, therefore, more similar to a real domestic scene. The *Lab B* is also more challenging, since it contains glass surfaces that generate noisy data. The dataset can be split in three parts:

1. Part 1 includes 360 RGB-D frames acquired from 3 static pedestals. It is composed of several views of 10 people, which have been asked to lie in 12 different poses, 6 from the back and 6 from the front. Each person has been manually segmented in 3D. Figure B.1 show some examples;

2. Part 2 includes 4 sequences of RGB-D frames, for a total of 15932 frames, acquired from a mobile robot during its patrolling task in the *Lab A*. People lie in 4 different fixed locations. Figure B.2 depicts some examples;

3. Part 3 includes 4 sequences of RGB-D frames, for a total of 9391 frames, acquired from a mobile robot during its patrolling task in the *Lab B*. People lie in 4 different fixed locations. Figure B.3 shows some examples.

[1]`http://robotics.dei.unipd.it/117-fall`

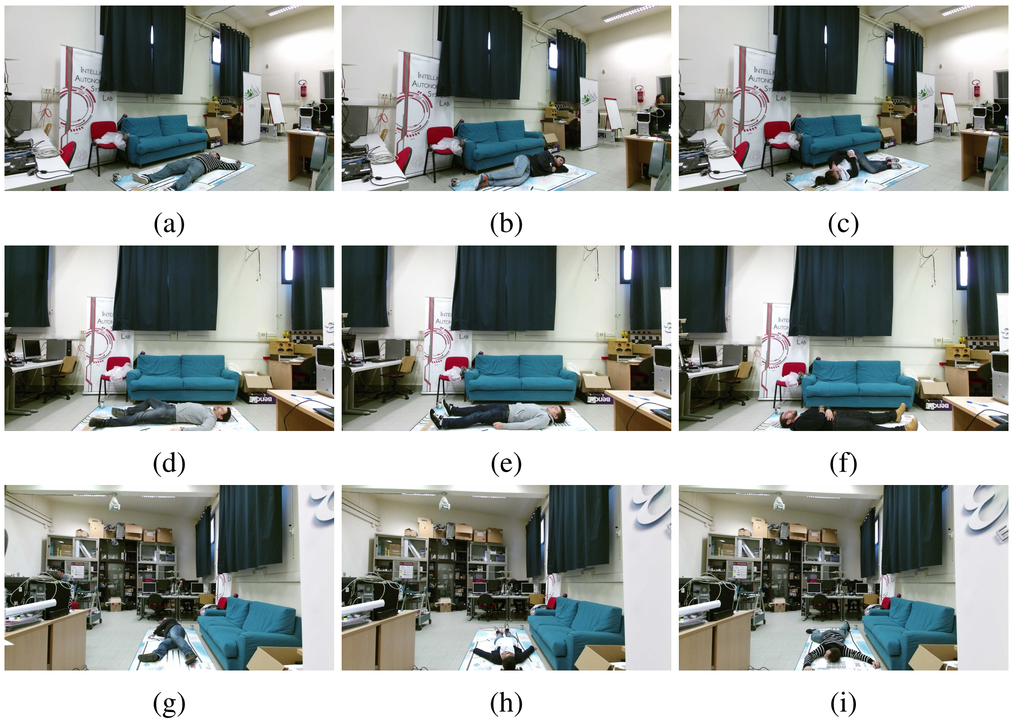
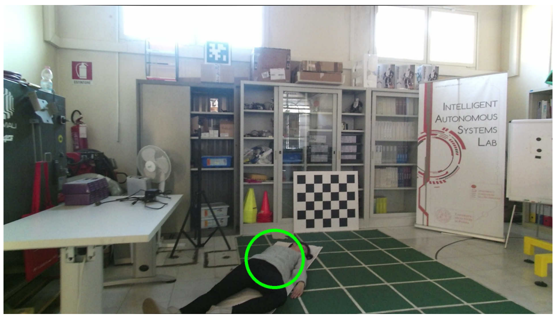
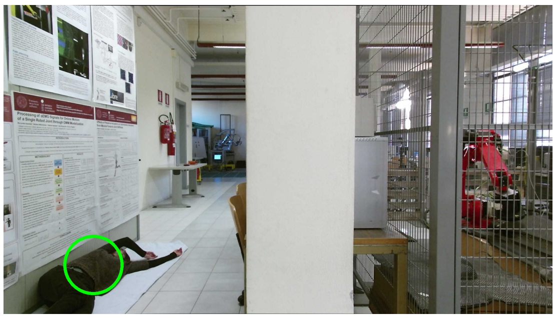


Figure B.1: Examples of the static part of the dataset. (a), (b) and (c) show three example frames acquired from the first pedestal position, while (d), (e) and (f) show the second pedestal position and (g), (h), (i) the third one. The people were free to position the limbs as they wanted, the only constraint was about were to place the back and the head.

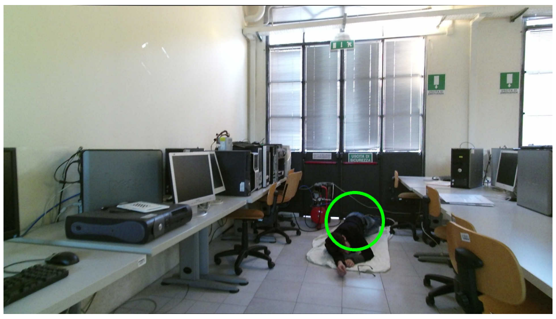
We organized the different sequences to reflect the training and testing split we used for our work [3](see Chapter 6).



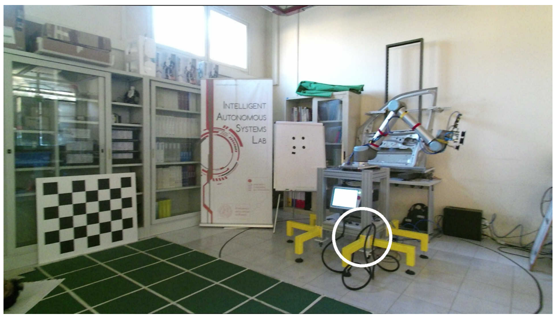
(a)



(b)



(c)



(d)

Figure B.2: Example frames about the Part 2 of the dataset. The frames have been captured from the results of the algorithm described in Chapter 6. The green circles show true positives found by the algorithm, while the white one that can be seen in (d) shows a correctly rejected detection.

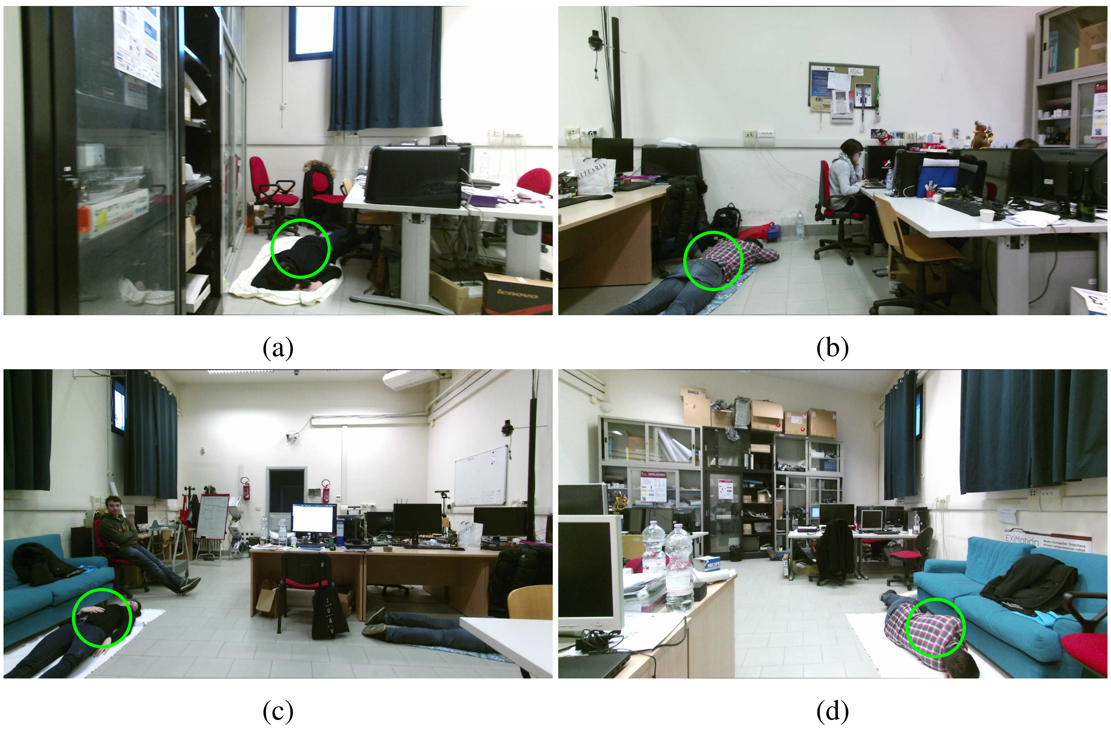


Figure B.3: Example frames about the Part 3 of the dataset. The frames have been captured from the results of the algorithm described in Chapter 6. The green circles show true positives found by the algorithm.

Bibliography

[1] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze. Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2104–2111. IEEE, 2013.

[2] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. S. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. In *BMVC*, volume 2, page 4, 2015.

[3] M. Antonello, M. Carraro, M. Pierobon, and E. Menegatti. Fast and robust detection of fallen people from a mobile robot. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4159–4166, Sept 2017.

[4] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–10. IEEE, 2016.

[5] F. Basso, R. Levorato, and E. Menegatti. Online calibration for networks of cameras and depth sensors. In *OMNIVIS: The 12th Workshop on Non-classical Cameras, Camera Networks and Omnidirectional Vision-2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, 2014.

[6] F. Basso, A. Pretto, and E. Menegatti. Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6244–6249. IEEE, 2014.

[7] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, apr 2014.

[8] A. Bedagkar-Gala and S. K. Shah. Gait-assisted person re-identification in wide area surveillance. *Computer Vision - ACCV 2014 Workshops*, pages 633–649, 2015.

[9] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[10] J. Boyle and M. Karunanithi. Simulated fall detection via accelerometers. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 1274–1277. IEEE, 2008.

[11] A. Brandon, B. Ludwiczuk, and S. Mahadev. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[12] K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, and C. Pantofaru. An adaptable system for rgb-d based human body detection and pose estimation. *Journal of visual communication and image representation*, 25(1):39–52, 2014.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, July 2017.

[14] M. Carraro, M. Antonello, L. Tonin, and E. Menegatti. An open source robotic platform for ambient assisted living. *Artificial Intelligence and Robotics (AIRO)*, pages 3–18, 2015.

[15] M. Carraro, M. Munaro, J. Burke, and E. Menegatti. Real-time marker-less multi-person 3d pose estimation in rgb-depth camera networks. *arXiv preprint arXiv:1710.06235*, 2017.

[16] M. Carraro, M. Munaro, and E. Menegatti. Cost-efficient rgb-d smart camera for people detection and tracking. *Journal of Electronic Imaging*, 25(4):041007–041007, 2016.

[17] M. Carraro, M. Munaro, and E. Menegatti. A powerful and cost-efficient human perception system for camera networks and mobile robotics. In *International Conference on Intelligent Autonomous Systems*, pages 485–497. Springer, Cham, 2016.

[18] M. Carraro, M. Munaro, A. Roitberg, and E. Menegatti. Improved skeleton estimation by means of depth data fusion from multiple depth cameras. In *International Conference on Intelligent Autonomous Systems*, pages 1155–1167. Springer, Cham, 2016.

[19] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4733–4742, June 2016.

[20] F. Cavallo, M. Aquilano, M. Bonaccorsi, R. Limosani, A. Manzi, M. C. Carrozza, and P. Dario. On the design, development and experimentation of the astro assistive robot integrated in smart environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4310–4315. IEEE, 2013.

[21] F. Cavallo, M. Aquilano, M. Bonaccorsi, I. Mannari, M. Carrozza, and P. Dario. Multidisciplinary approach for developing a new robotic system for domiciliary assistance to elderly people. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 5327–5330. IEEE, 2011.

[22] A. Chan, Y. Saito, and J.-M. Robine. International perspectives on summary measures of population health in an aging world, 2016.

[23] S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, J. Gonzalez, L. Karlsson, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo, et al. Giraffplus: Combining social interaction and long term monitoring for promoting independent living. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, pages 578–585. IEEE, 2013.

[24] R. Cucchiara, A. Prati, and R. Vezzani. A multi-camera vision system for fall detection and alarm generation. *Expert Systems*, 24(5):334–345, 2007.

[25] G. Diraco, A. Leone, and P. Siciliano. An active vision system for fall detection and posture recognition in elderly healthcare. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 1536–1541. IEEE, 2010.

[26] A. Elhayek, E. de Aguiar, A. Jain, J. Thompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. Marconiconvnet-based marker-less motion capture in outdoor and indoor scenes. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):501–514, 2017.

[27] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.

[28] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, et al. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems*, 75:60–78, 2016.

[29] D. Fischinger, P. Einramhof, W. Wohlkinger, K. Papoutsakis, P. Mayer, P. Panek, T. Koertner, S. Hofmann, A. Argyros, M. Vincze, et al. Hobbit-the mutual care robot. In *Workshop on Assistance and Service Robotics in a Human Environment Workshop in conjunction with IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2013, 2013.

[30] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[31] F. Fleuret, H. B. Shitrit, and P. Fua. Re-identification for improved people tracking. In *Person Re-Identification*, pages 309–330. Springer, 2014.

[32] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343–349, 1999.

[33] Z. Gao, Y. Yu, Y. Zhou, and S. Du. Leveraging two kinect sensors for accurate full-body motion capture. *Sensors*, 15(9):24297–24317, 2015.

[34] C. Geng and X. Jiang. Face recognition using SIFT features. In *International Conference on Image Processing*, pages 3277–3280. IEEE, 2009.

[35] S. Ghidoni, S. M. Anzalone, M. Munaro, S. Michieletto, and E. Menegatti. A distributed perception infrastructure for robot assisted living. *Robotics and Autonomous Systems*, 62(9):1316–1328, 2014.

[36] S. Ghidoni and M. Munaro. A multi-viewpoint feature-based re-identification system driven by skeleton keypoints. *Robotics and Autonomous Systems*, 90:45–54, 2017.

[37] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[38] S. B. Gokturk, H. Yalcin, and C. Bamji. A time-of-flight depth sensor-system description, issues and solutions. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 35–35. IEEE, 2004.

[39] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2432–2437. IEEE, 2005.

[40] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[41] H.-M. Gross, S. Mueller, C. Schroeter, M. Volkhardt, A. Scheidig, K. Debes, K. Richter, and N. Doering. Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5992–5999. IEEE, 2015.

[42] F. Han, X. Yang, C. Reardon, Y. Zhang, and H. Zhang. Simultaneous feature and body-part learning for real-time robot awareness of human behaviors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2621–2628, May 2017.

[43] W. He, D. Goodkind, and P. Kowal. An aging world: 2015. Technical report, 2016.

[44] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[45] J.-W. Hsieh, Y.-T. Hsu, H.-Y. M. Liao, and C.-C. Chen. Video-based human movement analysis and its application to surveillance systems. *Multimedia, IEEE Transactions on*, 10(3):372–384, 2008.

[46] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[47] R. Illum, M. Carraro, E. Menegatti, and J. Burke. Openptrack: Real-time, multi-camera computer vision infrastructure for artists. In *Workshop on Towards an artist-in-the-lab framework in conjunction with IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[48] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[49] O. H. Jafari, D. Mitzel, and B. Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5636–5643. IEEE, 2014.

[50] A. Jaimes and N. Sebe. Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, 108(1):116–134, 2007.

[51] L. Jeannotte, M. J. Moore, et al. *The State of aging and health in America 2007*. Merck Company Foundation, 2007.

[52] A. Kanaujia, N. Haering, G. Taylor, and C. Bregler. 3d human pose and shape estimation from multi-view imagery. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 49–56. IEEE, 2011.

[53] A. Kandhalu, A. Rowe, R. Rajkumar, C. Huang, and C.-C. Yeh. Real-time video surveillance over ieee 802.11 mesh networks. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 205–214. IEEE, 2009.

[54] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[55] Y. Kim. Dance motion capture and composition using multiple rgb and depth sensors. *International Journal of Distributed Sensor Networks*, 13(2):1550147717696083, 2017.

[56] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[57] K. Koide, E. Menegatti, M. Carraro, M. Munaro, and J. Miura. People tracking and re-identification by face recognition for rgb-d camera networks. In *Mobile Robots (ECMR), 2017 European Conference on*, 2017.

[58] K. Koide and J. Miura. Identification of a specific person using color, height, and gait features for a person following robot. *Robotics and Autonomous Systems*, 84:76–87, oct 2016.

[59] B. Li, Y. Dai, H. Chen, and M. He. Single image depth estimation by dilated deep residual convolutional neural network and soft-weight-sum inference. *arXiv preprint arXiv:1705.00534*, 2017.

[60] Q. Li, J. Stankovic, M. Hanson, A. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 138–143. IEEE, 2009.

[61] U. Lindemann, A. Hock, M. Stuber, W. Keck, and C. Becker. Evaluation of a fall detector based on accelerometers: A pilot study. *Medical and Biological Engineering and Computing*, 43(5):548–551, 2005.

[62] M. Lora, S. Ghidoni, M. Munaro, and E. Menegatti. A geometric approach to multiple viewpoint human body pose estimation. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6. IEEE, 2015.

[63] S. Lord, C. Sherrington, H. Menz, and J. Close. *Falls in older people: risk factors and strategies for prevention*. Cambridge University Press, 2007.

[64] R. Mazzon, S. F. Tahir, and A. Cavallaro. Person re-identification in crowd. *Pattern Recognition Letters*, 33(14):1828–1837, oct 2012.

[65] M. Memon, S. R. Wagner, C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen. Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes. *Sensors*, 14(3):4312–4341, 2014.

[66] S. Michieletto, F. Stival, F. Castelli, M. Khosravi, A. Landini, S. Ellero, R. Land, N. Boscolo, S. Tonello, B. Varaticeanu, C. Nicolescu, and E. Pagello. Flexicoil: Flexible robotized coils winding for electric machines manufacturing industry. In *ICRA workshop on Industry of the future: Collaborative, Connected, Cognitive*, 2017.

[67] C. Morato, K. N. Kaipa, B. Zhao, and S. K. Gupta. Toward safe human robot collaboration by using multiple kinects based real-time human tracking. *Journal of Computing and Information Science in Engineering*, 14(1):011006, 2014.

[68] M. Munaro, A. Basso, A. Fossati, L. Van Gool, and E. Menegatti. 3d reconstruction of freely moving persons for re-identification with a depth sensor. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4512–4519. IEEE, 2014.

[69] M. Munaro, F. Basso, and E. Menegatti. Tracking people within groups with rgb-d data. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2101–2107. IEEE, 2012.

[70] M. Munaro, F. Basso, and E. Menegatti. Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks. *Robotics and Autonomous Systems*, 75:525–538, 2016.

[71] M. Munaro, F. Basso, S. Michieletto, E. Pagello, and E. Menegatti. A software architecture for rgb-d people tracking based on ros framework for a mobile robot. In *Frontiers of Intelligent Autonomous Systems*, pages 53–68. Springer, 2013.

[72] M. Munaro, A. Fossati, A. Basso, E. Menegatti, and L. Van Gool. One-shot person re-identification with a consumer depth camera. In *Person Re-Identification*, pages 161–181. Springer, 2014.

[73] M. Munaro, S. Ghidoni, D. T. Dizmen, and E. Menegatti. A feature-based approach to people re-identification using skeleton keypoints. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5644–5651. IEEE, 2014.

[74] M. Munaro, A. Horn, R. Illum, J. Burke, and R. B. Rusu. Openptrack: People tracking for heterogeneous networks of color-depth cameras. In *IAS-13 Workshop Proceedings: 1st Intl. Workshop on 3D Robot Perception with Point Cloud Library*, pages 235–247, 2014.

[75] M. Munaro, C. Lewis, D. Chambers, P. Hvass, and E. Menegatti. Rgb-d human detection and tracking for industrial environments. In *Intelligent Autonomous Systems 13*, pages 1655–1668. Springer, 2016.

[76] M. Munaro and E. Menegatti. Fast rgb-d people tracking for service robots. *Autonomous Robots*, 37(3):227–242, 2014.

[77] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[78] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[79] L. Nanni, M. Munaro, S. Ghidoni, E. Menegatti, and S. Brahnam. Ensemble of different approaches for a reliable person re-identification system. *Applied Computing and Informatics*, 12(2):142–153, jul 2016.

[80] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[81] K. Nishi and J. Miura. A head position estimation method for a variety of recumbent positions for a care robot. In *Proceedings of the 6th Int. Conf. on Advanced Mechatronics*, 2015.

[82] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[83] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, pages 41.1–41.12. BMVA, 2015.

[84] N. Pears, Y. Liu, and P. Bunting. *3D imaging, analysis and applications*, volume 3. Springer, 2012.

[85] J. Perry, S. Kellog, S. Vaidya, J.-H. Youn, H. Ali, and H. Sharif. Survey and evaluation of real-time fall detection approaches. In *High-Capacity Optical Networks and Enabling Technologies (HONET), 2009 6th International Symposium on*, pages 158–164. IEEE, 2009.

[86] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016.

[87] G. Pitteri, M. Munaro, and E. Menegatti. Depth-based frontal view generation for pose invariant face recognition with consumer rgb-d sensors. In *International Conference on Intelligent Autonomous Systems*, pages 925–937. Springer, 2016.

[88] M. E. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, S. Engberg, J. T. Matthews, J. Dunbar-Jacob, C. E. McCarthy, et al. Pearl: A mobile robotic assistant for the elderly. In *AAAI workshop on automation as eldercare*, volume 2002, pages 85–91, 2002.

[89] M. Popescu, Y. Li, M. Skubic, and M. Rantz. An acoustic fall detector system that uses sound height information to reduce the false alarm rate. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 4628–4631. IEEE, 2008.

[90] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[91] C. R. Rao. The use and interpretation of principal component analysis in applied research. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 329–358, 1964.

[92] P. Rashidi and A. Mihailidis. A survey on ambient-assisted living tools for older adults. *Biomedical and Health Informatics, IEEE Journal of*, 17(3):579–590, 2013.

[93] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[94] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, July 2017.

[95] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[96] A. Roitberg, A. Perzylo, N. Somani, M. Giuliani, M. Rickert, and A. Knoll. Human activity recognition in the context of industrial human-robot interaction.

In *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, pages 1–10. IEEE, 2014.

[97] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[98] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-m. W. Hwu. Optimization principles and application performance evaluation of a multithreaded gpu using cuda. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82. ACM, 2008.

[99] J. Sanders and E. Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[100] H. Sarbolandi, D. Lefloch, and A. Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 139:1–20, 2015.

[101] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Conference on Computer Vision and Pattern Recognition*, pages 815–823. IEEE, 2015.

[102] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[103] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[104] S. C. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius. Convexity based object partitioning for robot applications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3213–3220. IEEE, 2014.

[105] F. Stival, S. Michieletto, and E. Pagello. How to deploy a wire with a robotic platform: Learning from human visual demonstrations. *Procedia Manufacturing*, 11:224–232, 2017.

[106] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 Classes. In *Conference on Computer Vision and Pattern Recognition*, pages 1891–1898. IEEE, 2014.

[107] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[108] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda. Softassign and em-icp on gpu. In *Networking and Computing (ICNC), 2010 First International Conference on*, pages 179–183. IEEE, 2010.

[109] H. Tian, B. Zhuang, Y. Hua, and A. Cai. Depth inference with convolutional neural network. In *Visual Communications and Image Processing Conference, 2014 IEEE*, pages 169–172. IEEE, 2014.

[110] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Conference on Computer Vision and Pattern Recognition*, pages 71–86. IEEE, 1992.

[111] United Nations. Department of Economic. *World population ageing, 1950-2050*. Number 207. New York: United Nations, 2002.

[112] R. Vezzani, D. Baltieri, and R. Cucchiara. Pathnodes integration of standalone particle filters for people tracking on distributed surveillance systems. In *Image Analysis and Processing–ICIAP 2009*, pages 404–413. Springer, 2009.

[113] M. Volkhardt, F. Schneemann, and H.-M. Gross. Fallen person detection for mobile robots using 3d depth data. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 3573–3578. IEEE, 2013.

[114] C. Wang, Y. Wang, and A. L. Yuille. An approach to pose-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2013.

[115] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1290–1297. IEEE, 2012.

[116] S. Wang, S. Zabir, and B. Leibe. Lying pose recognition for elderly fall detection. *Robotics: Science and Systems VII*, 345, 2012.

[117] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19, 2013.

[118] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4732, June 2016.

[119] A. Williams, D. Ganesan, and A. Hanson. Aging in place: fall detection and localization in a distributed smart camera network. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 892–901. ACM, 2007.

[120] D. Wolf, J. Prankl, and M. Vincze. Fast semantic segmentation of 3d point clouds using a dense crf with learned parameters. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4867–4873. IEEE, 2015.

[121] D. Wolf, J. Prankl, and M. Vincze. Enhancing semantic segmentation for robotics: the power of 3-d entangled forests. *IEEE Robotics and Automation Letters*, 1(1):49–56, 2016.

[122] L. Xiang. libfreenect2 CUDA library. https://github.com/xlz/libfreenect2, 2015. [Online; accessed 2016-02-03].

[123] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013.

[124] A. Yazar, F. Erden, and A. E. Cetin. Multi-sensor ambient assisted living system for fall detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 14)*, pages 1–3. Citeseer, 2014.

[125] K.-Y. Yeung, T.-H. Kwok, and C. C. Wang. Improved skeleton tracking by duplex kinects: A practical approach for real-time applications. *Journal of Computing and Information Science in Engineering*, 13(4):041007, 2013.

[126] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2752–2759, 2013.

[127] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti. Performance evaluation of the 1st and 2nd generation kinect for multimedia applications. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[128] G. Zhang, X. Huang, S. Z. Li, Y. Wang, and X. Wu. Boosting local binary pattern (LBP)-based face recognition. In *Advances in Biometric Person Authentication*, pages 179–186. Springer, 2004.

[129] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, dec 2003.

[130] Y. Zhao, M. Carraro, M. Munaro, and E. Menegatti. Robust multiple object tracking in rgb-d camera networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6625–6632, Sept 2017.

[131] Z. Zivkovic. Wireless smart camera network for real-time human 3d pose reconstruction. *Computer Vision and Image Understanding*, 114(11):1215–1222, 2010.