machines

MDPI

Article

A Cost-Effective Person-Following System for Assistive Unmanned Vehicles with Deep Learning at the Edge

Anna Boschi [1,2,*], Francesco Salvetti [1,2,3], Vittorio Mazzia [1,2,3] and Marcello Chiaberge [1,2]

1 Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy; francesco.salvetti@polito.it (F.S.); vittorio.mazzia@polito.it (V.M.); marcello.chiaberge@polito.it (M.C.)
2 PIC4SeR, Politecnico di Torino Interdepartmental Centre for Service Robotics, 10129 Turin, Italy
3 SmartData@PoliTo, Big Data and Data Science Laboratory, 10129 Turin, Italy
* Correspondence: anna.boschi@polito.it

Received: 13 July 2020; Accepted: 25 August 2020; Published: 28 August 2020

Abstract: The vital statistics of the last century highlight a sharp increment of the average age of the world population with a consequent growth of the number of older people. Service robotics applications have the potentiality to provide systems and tools to support the autonomous and self-sufficient older adults in their houses in everyday life, thereby avoiding the task of monitoring them with third parties. In this context, we propose a cost-effective modular solution to detect and follow a person in an indoor, domestic environment. We exploited the latest advancements in deep learning optimization techniques, and we compared different neural network accelerators to provide a robust and flexible person-following system at the edge. Our proposed cost-effective and power-efficient solution is fully-integrable with pre-existing navigation stacks and creates the foundations for the development of fully-autonomous and self-contained service robotics applications.

Keywords: person-following; robotics; deep learning; edge AI

1. Introduction

Person-following is a well-known problem in robotic autonomous navigation that consists of the ability to detect and follow a target person with a mobile platform. This task can be achieved with a variety of sensing and moving systems and has fundamental roles in a variety of applications in domestic, industrial, underwater and aerial scenarios [1]. Due to the sharp increment of life expectancy in the last century, the world population has seen a progressive increase in the number of older people [2]. This trend offers an excellent opportunity for developing new service robotics applications to provide continuous assistance to autonomous elders in everyday life. These robotic platforms should be able to identify the target person and follow him to offer their support. Person-following assumes, thus, a vital role, as a technology necessary to enable a variety of different applications. In this context, it is essential to develop a system that focuses on robustness to different domestic scenarios and efficiency to be implemented on low-power devices, without the need for external computing devices. Moreover, the ability to run a person-following algorithm entirely onboard makes the system less prone to security and privacy issues, avoiding unnecessary transmission of sensitive information, such as domestic camera streams.

Generally speaking, a person-following system is composed of a sensing device, a detection algorithm able to provide an estimate of the target position and a following algorithm to control the robot's movements. Indoor robotic platforms use a variety of perception devices, divided into exteroceptive, such as cameras, LiDARs and ultrasonic sensors, and proprioceptive, such as inertial measurement units (IMUs), gyroscopes, accelerometers and encoders. Different solutions to the detection problem can be found in the literature, depending on the used sensors, on the application scenario and the type of approach [1]. Recent developments in deep learning techniques [3] for computer vision gave a significant boost to the ability to efficiently extract meaning from visual information and inspired several solutions for the person-following problem.

Inspired by these approaches, we employed the popular deep learning object detection algorithm YOLOv3-tiny [4], suitably re-trained for the specific task, to detect the target person from an RGB-D frame and compute his location with respect to the robot reference frame. The extracted information was then fed to an efficient control algorithm that generated the suitable linear and angular commands for the robot actuators to achieve person-following. Moreover, we tested the proposed approach on several embedded platforms designed explicitly for the edge AI, that consists of deploying artificial intelligence algorithms on low-power devices. We compared the obtained results with a particular focus on the trade-off between performance and power consumption. The overall solution proposed represents a cost-effective, low-power pipeline for the person-following problem that can be easily employed at the edge as a primary component in complex service robotics tasks.

2. Related Works

Related literature is organized as follows. Firstly, several methods for person-following are analyzed, with a focus on the sensing devices used and on the strategies used to detect the target. Then, deep learning techniques for object detection are briefly discussed, with attention paid to recent developments in edge AI implementations.

2.1. Person Following

The task of recognizing and localizing a person to be followed by a robotic platform has been widely discussed in the literature since the nineties. Islam et al. [1] reviewed and categorized a large number of works focused on achieving person-following in a variety of conditions, such as ground, underwater and aerial scenarios. For what concerns terrestrial applications, important classifications of the different methods are based on the kinds of devices used to sense the environment and on the strategy used to detect the target person.

Most ground applications use a simple unicycle model that controls the robot 2D motion in polar coordinates, with a linear velocity on the xy plane and angular velocity about the z-axis [5]. The chosen detection system should, therefore, be able to find the target position and distance from the robot. Several systems use laser range finders (LRF) measures that directly provide a set of distances, which are clustered and interpreted to extract relevant features. People are localized usually by means of leg [6–11] or torso identification [12–14]. However, these methods mainly rely on static features extracted from 2D point clouds that frequently lead to a poor detection quality. Visual sensors are much more informative since they allow one to sense the entire body of the target, but simple RGB cameras are not enough since a distance measure is also needed. The two main categories of visual sensors able to catch depth information are stereo and RGB-D cameras. Several works [15–20] use the first approach to approximate the distance information by triangulation methods applied on two or more RGB views of the same scene. However, the most used visual sensors for person detection are RGB-D cameras [21–32] that are able to get both RGB images and depth maps by exploiting infrared light. Several methods employ sensor fusion techniques to merge information from different kinds of sensing systems. For example, Alvarez et al. [33] used both images to detect the human torso and lasers to track the legs, Susperregi et al. [34] used an RGB-D camera, lasers and a thermal sensor; and Wang et al. [35] used a monocular camera with an ultrasonic sensor. Hu et al. [36] adopt eda human walking model using a combination of RGB-D data, LRF leg tracking and robot odometry and a sonar sensor for obstacle avoidance during navigation. Koide et al. [14] used LRF data to detect people in the scene, and then cameras to identify them and extract relevant features. Cosgun et al. [8], on the other hand, manually selected the target from an RGB-D view of the environment, and then tracked it with LRF leg identification. Merging data from multiple sensors allows one to increase detection accuracy, but with high increases in the system's complexity and costs. Furthermore, the presence of multiple sources of data requires hardware with high computational power to enable real-time processing. Since our focus was on developing an embedded, cost-effective, low-power system, we selected a low-cost RGB-D camera as the only sensing device.

Focusing on vision-based methods, different strategies can be adopted to detect the person in the environment. Mi et al. [26], Ren et al. [25] and Chi et al. [29] all adopted the Microsoft Kinect SDK that directly provides skeleton position. Satake et al. [16,17] used manually designed templates to extract relevant features and find the target location. Munaro et al. [23], Brookshire [15] and Basso et al. [22], instead, adopted histograms of the oriented gradients (HOG) method for human detection originally proposed by Dalal et al. [37]. More recently, machine learning techniques have been used to solve the person-following task. Chen et al. [19] used an online AdaBoost classifier initialized on a manually-selected bounding box of the target person. Chen [31], instead, used an upper-body detector based on an SVM to get the human position and extract relevant features used during the tracking phase. More recently, deep learning models have been employed to further boost detection accuracy. Chen et al. [18] proposed a CNN-based classifier trained on a manually-selected target with an online learning procedure. Masuzawa et al. [28] adopted the YOLO method [38] to identify the person due to its high results in both precision and recall rates. Wang et al. [20] also employed YOLO as a person detector, but only to predict the initial position of the target, since they are not able to run the algorithm in real-time due to hardware limitations. Jiang et al. [30] jointly used a DCNN-based detector and a PN classifier based on random forests to enhance person localization and tracking. Finally, Yang et al. [32] used a DNN to identify a bounding box image to be scored against the pre-registered user image.

Exactly as in [20,28], we purpose a person-following approach based on the YOLO network, but our methodology is different from theirs. We use a newer and smaller version of YOLO (YOLOv3-tiny), and the re-training and the optimization of the network involve:

- Eliminating the tracking part and relating an additional filter thanks to the continuous detection of the target, so reducing the computational complexity of the solution;
- Running the detection at the edge, so it can be easily realized on the neural board accelerator, without adding an expensive onboard computer (low-cost).

2.2. *Deep Learning for Real-Time Object Detection*

Object detection is a field of computer vision that deals with localizing and labeling objects inside an image. Before the recent huge developments in deep learning techniques, object detection was classically performed with machine learning methods such as the cascade classifier based on Haar-like features [39] or coupled with feature extraction algorithms like the histograms of oriented gradients (HOG) [37,40]. With the recent developments in deep learning, considerable improvements in both the accuracy and efficiency of object detection algorithms have been achieved. Current techniques are split into region proposal methods and single-shot detectors. The former firstly identifies areas inside the image that most likely contain objects of interest, abd then feeds them to a second stage that predicts label and bounding box dimensions. In this category, we find algorithms such as R-CNN [41], fast R-CNN [42] and faster R-CNN [43]. Single-shot detectors, on the other hand, treat the detection task as a regression problem and directly perform both localization and labeling with a single stage. This method makes them generally faster than region proposal techniques, but with slightly less accuracy. The most known single shot detectors are SSD [44] and YOLO [38], with its evolutions YOLOv2 [45], YOLOv3 [4] and YOLOv4 [46]. Lightweight versions of these methods, such as YOLOv3-tiny, have been specifically developed to be implemented in low-power real-time systems and therefore are most suitable for service robotics applications.

Recently, several advancements have been made in edge AI, where deep neural networks are deployed on low-power real-time embedded systems [47]. This field of research has principally flourished thanks to the release of hardware platforms specifically designed to accelerate deep neural network inferences. NVIDIA released boards with onboard GPUs, such as Jetson TX2, AGX Xavier and Nano. Intel produced two generations of USB hardware accelerators called Neural Computing Stick (NCS), and recently Google released its own Coral board and USB accelerator, able to boost inference performance using the Tensor Processing Unit (TPU) chips. In the literature can be found several works which apply optimization techniques to object detection algorithms to deploy them on embedded devices with hardware acceleration [48–52].

In our work, we fine-tuned a pre-trained YOLOv3-tiny network for the person detection task, and we propose a cost-effective person-following system that can generate suitable velocity commands for the robotic actuators, based on RGB-D images. The proposed methodology was extensively tested with several edge AI devices in order to compare performance and power consumption for the different possible configurations. Finally, a potential implementation of the proposed system was integrated and tested with a standard robotic platform.

The rest of the paper is organized as follows. Section 3 presents the dataset used in the re-training and the hardware setup. Section 4 discusses the proposed methodology with an extensive description of the detection mechanism and the control algorithm. Finally, Section 5 presents the experimental discussion, the performance comparison on the considered hardware platforms and the final complete implementation.

3. Materials and Data

The network adopted was pre-trained with the COCO dataset, which contains 80 classes of objects with their respectively bounding box and marks. Subsequently, a technique named *transfer learning* [53] was adopted to realize the re-training and the fine-tuning of the network using a smaller dataset composed by the *person* class only. In this way, a custom version of the network YOLOv3-tiny was produced, optimizing it for accurate and efficient detection. That network was tested and compared with the original model, producing some metric evaluation results. The deep learning network was evaluated on different edge AI devices by assessing the performance of each of them in terms of inference speed and power consumption. Finally, a specific hardware solution was selected to assemble a robotic platform and test it in a real environment.

3.1. Data Description

The images of people used to create the person dataset were extracted from the OIDv4 [54] dataset, which are divided into training, validation and testing. During the re-training phase, we used 6001 images, divided into the training set, 5401, and the test set, 600. The dimension of the images was imposed to be equal to 416×416 during training, in order to be more coherent with the pre-trained input dimension of the original network and the native resolution, 480×640, of the depth camera.

3.2. Hardware Description

The main request to fulfill is the necessity of a real-time response in each step that makes up this robotic application: person detection algorithm, data elaboration, control of the robot and the navigation into the indoor environment. All these operations must be as instantaneous as possible to avoid the loss of the person to follow—extremely probable in case the person moves away from the robot view.

For what concerns the embedded implementation of the neural network, different platforms were evaluated and are shown in Figure 1 and summarized in Table 1: a Raspberry Pi 3 B+ with Intel NCS, a Raspberry Pi 3 B+ with Movidius NCS2, a Coral USB Accelerator, an NVIDIA Jetson AGX Xavier developer kit and an NVIDIA Jetson Nano.

The Neural Computing Sticks (NCS) are USB dedicated hardware accelerators specifically used to perform deep neural network inferences. Both the first and the second generations of the NCS have been tested: the first has a Myriad 2 Vision Processing Unit (VPU), while the second has Myriad X

VPU and reaches eight times the performance of the previous version. These two components request a USB 3.0 or 2.0 interface so that they can be easily used with cheap single-board computers such as a Raspberry board.

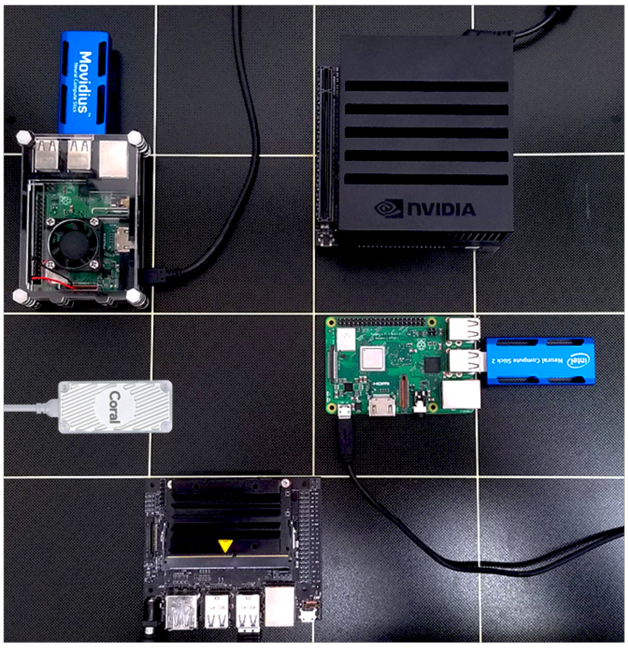


Figure 1. The analyzed embedded devices to deploy deep neural networks at the edge. **Top left**: Raspberry Pi 3 B+ with Intel NCS. **Top right**: NVIDIA Jetson AGX Xavier. **Center left**: Coral USB Accelerator. **Center right**: Raspberry Pi 3 B+ with Movidius NCS2. **Bottom**: NVIDIA Jetson Nano.

The Coral USB Accelerator, released in 2019, is an on-board edge TPU coprocessor able to reach high-performance machine learning inference, with a limited power cost, for TensorFlow Lite models. The board can work at different clock frequencies: maximum or reduced. These frequency types are one the twice of the other, so using the maximum frequency there is an increase of the inference speed with a consequent increase of the power consumption.

The NVIDIA Jetson AGX Xavier, released in 2018, is a System-On-Module able to guarantee high performance and power efficiency. The board contains DRAM, CPU, PMIC, flash memory storage and a dedicated GPU for hardware acceleration, so it has been specifically realized to perform rapidly different neural network operations. The kit is also supplied with several software libraries as NVIDIA JetPack, DeepStream SDKs, CUDA, cuDNN, and TensorRT. It is possible to set different power mode configurations also selecting the number of CPU cores utilized: 10 W (2 cores), 15 W (4 cores), 30 W (2, 4, 6 or 8 cores).

The NVIDIA Jetson Nano is a lightweight, powerful computer explicitly designed for AI in order to run multiple neural networks in parallel for image elaboration. The board mounts a 128-core NVIDIA Maxwell GPU, a Quad-Core ARM Cortex-A57 MPCore CPU and a 4 GB LPDDR4 memory and reaches the peak performance of 472 GFLOPs. It can work in two power modes: at 5 W or 10 W without the support of Tensor cores during the inference acceleration.

The complete hardware selected for testing in the test environment is an upgrade of the TurtleBot3 Waffle Pi from ROBOTIS (https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/), a standard robotic platform supported by ROS and extremely used by developers. This robot uses as on-board PC the Jetson Xavier developer kit (https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit), and it has been equipped with an additional camera sensor, an Intel RealSense Depth camera D435i (https://www.intelrealsense.com/depth-camera-d435i/). This low-cost depth camera, ideal for navigation or object recognition applications, is composed of an RGB module and two infrared cameras separated by a wide IR projector.

Table 1. The main specifications of the embedded HW for edge AI taken into account for the experimentation. The price indicated for each device is referred to the commercial value at the time of the publication.

	Intel NCS	**Intel Movidius NCS2**	**Coral USB Accelerator**	**NVIDIA Jetson AGX Xavier Developer Kit**	**NVIDIA Jetson Nano**
AI performance	100 GFLOPs (FP32)	150 GFLOPs (FP32)	4 TOPs (INT8)	32 TOPs (FP32)	472 GFLOPs (FP32)
HW accelerator	Myriad 2 VPU	Myriad X VPU	Google Edge TPU coprocessor	512-core NVIDIA Volta GPU with 64 Tensor Cores and 2x NVDLA Engines	128-core NVIDIA Maxwell GPU
CPU	N.A.	N.A.	N.A.	8-core NVIDIA Carmel Arm v8.2 64-bit CPU 8MB L2 + 4MB L3	Quad-core ARM Cortex-A57 MPCore processor
Memory	4 GB LPDDR3	4 GB LPDDR3	N.A.	32 GB 256-bit LPDDR4x 136.5 GB/s	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	N.A.	N.A.	N.A.	32 GB eMMC 5.1	Micro SD card slot or 16 GB eMMC 5.1 flash
Power	1 W	1.5 W	1 W	10/15/30 W	5/10 W
Size	73 × 26 mm	73 × 26 mm	65 × 30 mm	100 × 87 mm	70 × 45 mm
Weight	18 g	19 g	20 g	280 g	140 g
Price	$70	$74	$60	$700	$99

4. Proposed Methodology

The goal of this research was to develop an autonomous, real-time, person-following assistive system able to promote the aging-in-place of independent elderly people.

The workflow of our solution is exposed here. Firstly, the detection and localization of the person in the environment are realized, using the RGB-D information from the camera and a neural network specifically designed for fast and accurate real-time object detection, YOLOv3-tiny. Successively, the information about the position of the person with respect to the robot reference frame is used to create a tailored control algorithm, using a linear trend for the linear velocities and a parabolic trend for the angular velocities. In this way, the robot can follow the person while remaining at a certain safe distance from him, thereby avoiding both hitting and losing the target. The pseudo-code of the overall algorithm is reported in the Algorithm 1.

The proposed algorithm, in the practical implementation, is integrated with the open-source Robot Operating System (ROS) (https://www.ros.org/) to set the control to the actuators of the robot. The result is an autonomous, cost-effective person-following system with deep learning at the edge, easily integrable in different unmanned vehicles.

Algorithm 1: Person-following using an RGB-D camera and YOLOv3-tiny network.

```
1: Inputs:
     $D_{h \times w}$: Depth Matrix obtained from the camera
     $x_{bb}$: Horizontal coordinate of the center of the bounding box
     $y_{bb}$: Vertical coordinate of the center of the bounding box
     $x_c$: Horizontal coordinate of the center of the camera frame
     $N_{detection}$: Number of the detections provided by the network
2: Initialize:
     $F_{moreperson} \leftarrow 0$
     $V_{linear_{(0)}} \leftarrow 0$
     $V_{angular_{(0)}} \leftarrow 0$
     $T_{time} \leftarrow 0$
     $i \leftarrow 1$
3: while True do
4:     if ($N_{detection}$ == 0) then
5:         $V_{linear_{(i)}} \leftarrow V_{linear_{(i-1)}}$
6:         $V_{angular_{(i)}} \leftarrow V_{angular_{(i-1)}}$
7:         if $T_{time} > t$ then
8:             reset: $T_{time}$
9:             $V_{linear_{(i)}} \leftarrow 0$
10:            $V_{angular_{(i)}} \leftarrow 0$
11:        end if
12:    else if ($N_{detection}$ == 1) then
13:        $dx \leftarrow (x_c - x_{bb})$
14:        $depth \leftarrow D[y_{bb}, x_{bb}]$
15:        $V_{linear_{(i)}} \leftarrow linearvelocity(depth)$
16:        $V_{angular_{(i)}} \leftarrow angularvelocity(dx)$
17:        reset: $T_{time}, N_{detection}$
18:    else
19:        if $T_{time} > 1$
20:            reset: $T_{time}$
21:            $V_{linear_{(i)}} \leftarrow 0$
22:            $V_{angular_{(i)}} \leftarrow 0$
23:            $F_{moreperson} \leftarrow 1$
24:        end if
25:        reset: $N_{detection}$
26:    end if
27:    $VelocityController(V_{linear_{(i)}}, V_{angular_{(i)}})$
28:    if ($F_{moreperson}$ == 1) then
29:        stop the algorithm for a prefixed time
30:        reset: $F_{moreperson}, T_{time}$
31:    end if
32:    $i++$
33:    acquire next Inputs
34:end while
```

4.1. Person Localization

By using a re-trained version of YOLOv3-tiny for object detection and the RGB-D camera chosen for this application, it is possible to detect and localize a person in space interactively. In fact, once the network is optimized, the precision and recall values allow one to have a continuous detection of the target without the use of a tracker algorithm or additional filter to support the control implementation. That means the use of the network is sufficient to realize real-time person-following, while reducing the computational cost and other power consumption. These considerations are supported both from the average precision AP_{50} obtained from the re-trained network, and the specific use case taking into account: a self-sufficient older person in his or her home environment. As the target to follow is an elderly person, its moving velocity is reduced so the tracker is superfluous and consequently the control is smooth, and this is also supported thanks to the reduced speed of the robot.

Object detection is an important area of research, interested in the processing of images and videos to detect and recognize objects. You only look once (YOLO) [4,38,45] is the object detection method commonly used in the real-time processing image applications. This model, based on a feed-forward convolutional neural network, is considered an evolution of the single-shot-multibox detector (SSD) concept with the idea of predicting both the bounding boxes and the class detection probability simply analyzing the image once. Its architecture is based on a single neural network trained end-to-end to increase the accuracy and to reduce the predictions of false positives on the background.

The operations done by the network can be divided into four steps:

1. The input image is processed with a grid cell as a reference frame.
2. Each grid cell generates bounding boxes and predicts their confidence rate. The confidence rate depends on the accuracy of the network during the detection.
3. Each grid cell has a probability score for each class. The number of classes depends on the dataset used during the training process of the network.
4. The total number of bounding boxes is minimized by setting a minimum confidence rate and using the non-maximum suppression (NMS) algorithm to obtain the final predictions that can be used to generate the final output: an input image with the bounding boxes over the detected objects with the reference classes and the accuracy percentages.

During the evolution of the YOLO architecture, incremental improvements can be recorded in the different versions developed, starting from YOLOv2, which includes many features to increase the performance, until reaching YOLOv3 and YOLOv4, the last two versions of the model, in which there are notable improvements in the capability of the network to detect objects. In our proposed methodology, we suggest a re-trained and optimized version of YOLOv3-tiny, which is the lightweight version of YOLOv3 with a reduced number of trainable parameters. In Table 2 is the structure of the modified architecture of YOLOv3-tiny for only the class person.

Table 2. YOLOv3-tiny architecture designed to work with only class person and an input size of 416 × 416. The network after training is further optimized to be deployed onboard the robotic platform.

Layer	**Type**	**Size/Stride**	**Filters**	**Output**
0	*Convolution*	3 × 3/1	16	416 × 416 × 16
1	*MaxPooling*	2 × 2/2		208 × 208 × 16
2	*Convolution*	3 × 3/1	32	208 × 208 × 32
3	*MaxPooling*	2 × 2/2		104 × 104 × 32
4	*Convolution*	3 × 3/1	64	104 × 104 × 64
5	*MaxPooling*	2 × 2/2		52 × 52 × 64
6	*Convolution*	3 × 3/1	128	52 × 52 × 128
7	*MaxPooling*	2 × 2/2		26 × 26 × 128
8	*Convolution*	3 × 3/1	256	26 × 26 × 256
9	*MaxPooling*	2 × 2/2		13 × 13 × 256
10	*Convolution*	3 × 3/1	512	13 × 13 × 512
11	*MaxPooling*	2 × 2/1		13 × 13 × 512
12	*Convolution*	3 × 3/1	1024	13 × 13 × 1024
13	*Convolution*	1 × 1/1	256	13 × 13 × 256
14	*Convolution*	3 × 3/1	512	13 × 13 × 512
15	*Convolution*	1 × 1/1	255	13 × 13 × 18
16	*YOLO*			
17	*Route 13*			
18	*Convolution*	1 × 1/1	128	13 × 13 × 128
19	*Up-sampling*	2 × 2/1		26 × 26 × 128
20	*Route 19 8*			
21	*Convolution*	3 × 3/1	256	26 × 26 × 256
22	*Convolution*	1 × 1/1	255	26 × 26 × 18
23	*YOLO*			

4.1.1. Person Detection and Localization Implementation

The RGB camera frames are the input data that feed the re-trained YOLOv3-tiny network for the class person. As already introduced, the full input image is treated in functional regions represented as a grid of cells. In each region, bounding boxes are weighted by the predicted probabilities, and the predictions are the result of single network evaluation.

In order to localize the position of the person in the video frame, it is sufficient to use the bounding box information provided by the network. As it is depicted in Figure 2, the bounding box of the detected person directly provides the coordinates of the angle, x, y, with respect to the R0 reference frame.

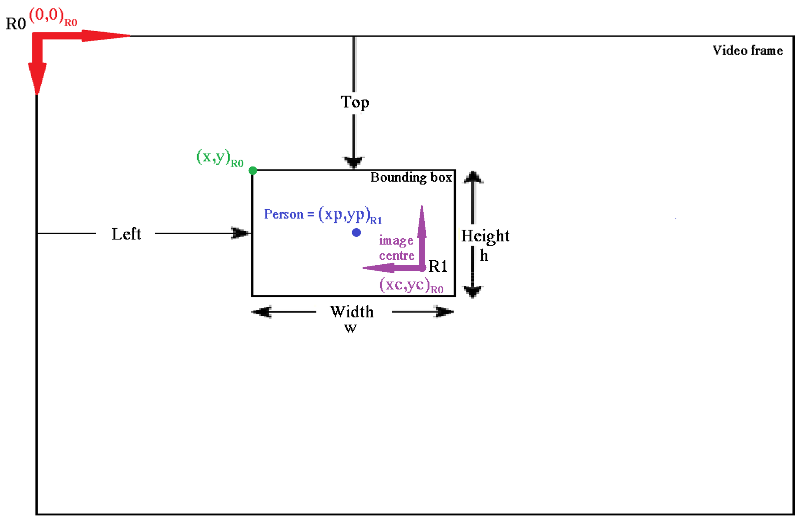


Figure 2. Video frame structure. It represents the reference frame (RF) transformation adopted to compute the pixel coordinates of the center of the bounding box detecting the person with respect to a new RF (R1), the one used during the calculation of the angular velocity control signals for the robot.

By using the information of the center, with respect to R0, it is possible to calculate the coordinates of the midpoint of the person detected with respect to the new reference frame R1 (x_p, y_p):

$$x_p = x_c - \left(x + \frac{w}{2}\right) \tag{1}$$

$$y_p = y_c - \left(y + \frac{h}{2}\right) \tag{2}$$

where x_c and y_c are respectively 319 and 239 pixels due to the image resolution, 640 $\times$ 480, of the camera taken as reference in this study.

The x_p and y_p coordinates are necessary to locate the person in the 2D space and x_p; in particular, is fundamental to develop the angular control algorithm, to adjust the rotation of the robot.

Once known, with the pixel coordinates of the center of the bounding box detecting the person, it is possible to obtain the distance between the robot and the person by merely extracting this information from the corresponding pixel of the depth camera matrix. The dimension of that matrix is equal to the resolution of the acquired RGB frame, and for each pixel position, there is a value in millimeters representing the distance of the camera from what it sees. Since the limits of the depth computation, of the camera taken as reference in this study, are 0.105 m and 8 m, the values of the depth matrix range from 0 to 8000.

The depth value extracted represents the missing coordinate to localize a person in the 3D environment. The z coordinate is strictly necessary to realize the linear velocity control algorithm, which is able to regulate the forward or backward movement of the robot.

4.1.2. Detection Situation Rules

Three possible detection situations are taken into account by controlling the output value, $N_{detection}$, of the network:

1. *Nothing detected:* If nothing is detected, the robot stops. Differently, suppose the robot loses the person it is following. In that case, it continues to move in the direction of the last detection with the previous velocity commands for a pre-imposed time t. After that, if nothing is detected again, the robot stops.
2. *One person detected:* The robot follows the movement of the person while remaining at a certain safe distance from him.
3. *More than one person simultaneously detected:* The robot stops for a prefixed time and then it restarts the normal detection operation. There could be many other solutions to implement, for example, a person tracking algorithm to follow one of the people detected [55–57]. In particular, the presence of a tracking algorithm or an additional filter will have to be considered if the use case is changed—for example, in case this approach will be used in an office environment.

However, for this specific application, it has been decided to block the robot directly because it has been assumed that the person using it should be self-sufficient, living in the house alone. Therefore, if the person receives visits, it would be unpleasant and unnecessary to have a robot following him inside the house.

4.2. Person-Following Control Algorithm

The linear and the angular velocity are regulated using different functions, so, in order to obtain a correct control algorithm, it is necessary to combine them simultaneously.

4.2.1. Angular Velocity Control

The angular velocity is proportional to the horizontal coordinate, x_p, of the center of the bounding box detecting the person, computed with respect to the reference frame located in the center of the frame of the camera. It has a parabolic trend in order to make the movements of the robot smoother and more natural. By considering this reference frame, the dx value is positive if the person is on the left side of the frame or negative if it is on the right side. Figure 3 shows a graphical representation of how the dx value is obtained.

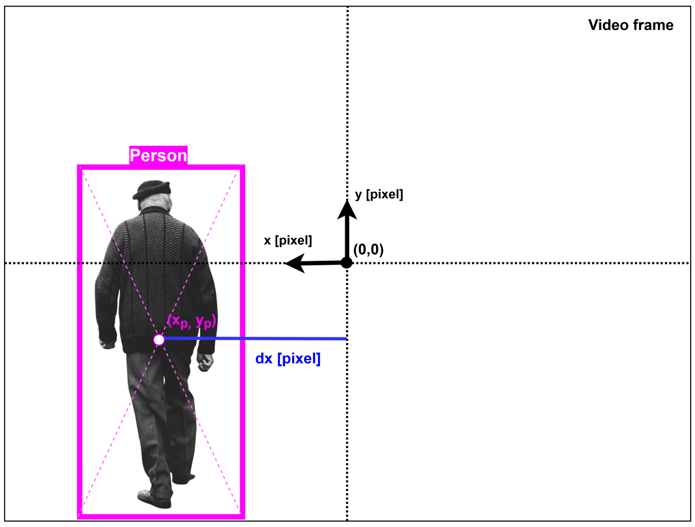


Figure 3. The RGB frame of the camera showing how the dx value, adopted in the computation of the angular velocity, is obtained. This value is equal to the horizontal coordinate of the center of the bounding box detecting the person in the frame.

The controller function generated has the dx value, measured in pixel, as input, and gives as output the angular velocity, $v_{angular\theta}$, according to the following formula:

$$v_{angular\theta} = \begin{cases} \frac{max_{vel} \cdot dx^2}{320^2}, & \text{if } dx \geq 0 \\ \frac{min_{vel} \cdot dx^2}{320^2}, & \text{if } dx < 0 \end{cases} \quad (3)$$

The terms max_{vel} and min_{vel}, which are equal and opposite values, are the upper and the lower limits of the angular velocity of the robotic platform. The number 320 pixels stands for the maximum number of pixels for each side (left and right) of the video frame because the resolution of the image received from our camera is 640×480.

4.2.2. Linear Velocity Control

The linear velocity depends linearly on the distance between the robot and the person detected $depth_m$ measured in meters. In particular, it is in function of the value obtained from the depth matrix of the camera in the center point of the bounding box detecting the person. This control can be represented by a linear trend divided into three regions. In the first region, the distance is superior compared to the set upper limit $m_{vel_{upperlimit}}$, so here the robot moves straight on following a linear proportional trend until it reaches its maximum speed saturating to that value. In the second region, there is a stop condition; in fact, the robot is at the safety distance from the person and remains there to avoid losing the person. The zero value is also assigned when the distance is 0 m, in order to avoid a particular case in the code. In fact, the depth value obtained from our camera has a limit of 0.105 m, so nothing should be detected at a lower distance. The final region is between the limit of the camera: 0.105 m and the distance lower-limit $m_{vel_{lowerlimit}}$. In this condition, the robot goes back following a linear proportional trend until it reaches its maximum negative speed saturating to that value. Here is reported the formula responsible for the linear velocity $v_{linearx}$:

$$v_{linearx} = \begin{cases} depth_m \cdot m1 + q1, & \text{if } depth_m > m_{vel_{upperlimit}} \\ 0, & \text{if } m_{vel_{lowelimit}} < depth_m \leq m_{vel_{upperlimit}} \text{ or } depth_m = 0 \\ depth_m \cdot m2 + q2, & \text{if } depth_m \leq m_{vel_{lowelimit}} \end{cases} \quad (4)$$

The values $m1$, $q1$, $m2$ and $q2$ were found using the equation of the straight line passing through two points.

5. Experimental Discussion and Results

In this section, we firstly discuss some technical details of the re-training procedure of the tiny version of YOLOv3. Then experimental evaluations are discussed for both the model and its deployment on the selected embedded devices. Finally, we present our platform implementation that represents one of the possible practical configurations to realize the person-following solution presented with this work.

5.1. Person Detector Training and Optimization

In order to obtain a lightweight and efficient network for the detection of the target, we modified the original model to classify and localize the class person only. Using OIDv4 [54], we collected a set $\mathbb{X}$ of 6001 training samples, reserving 600 of them for testing. Making use of transfer learning [58], we started our training from a pre-trained backbone, from layer 0 to 15 in Table 2. That greatly speeds up the training, drastically reducing the number of samples required to achieve a high level of accuracy. We trained for 20 epochs with a linear learning rate decay and an initial value of $\eta = 0.0001$. We adopted momentum optimization [59] with $\beta = 0.9$ and a batch size of 32. The training procedure lasted approximately one hour on a workstation with an NVIDIA RTX 2080 Ti and 64 GB of DDR4 SDRAM.

It is possible to observe the effectiveness of the re-training procedure from Table 3. The re-trained version of YOLOv3-tiny gains more than 30% of average precision (AP) at 0.5 of intersection over unit (IOU). Moreover, the resulting single-class network is 23% faster, in terms of inference latency, than the multi-class counterpart. That is due to the reduced number of features maps in the final detection section of the network, from layer 15 to 23 of Table 2.

Table 3. Average precision at 0.5 IOU for person class before and after re-training. It is clear how transfer learning is so effective at improving the performance of the tiny version of the YOLOv3 model, increasing the metric score by more than 30%.

Network	AP_{50}	**Gain**
YOLOv3-tiny	19.21 %	
YOLOv3-tiny$_{person}$	49.30 %	30.09 %

Finally, we optimized the resulting re-trained model with two different libraries: TensorRT and TensorFlow Lite (https://www.tensorflow.org/lite). Optimization is a fundamental process and aims at reducing latency, inference cost, accelerator compatibility, memory and storage footprint. That is mainly achieved with two distinct techniques: model pruning and quantization. The first one simplifies the topological structure, removing unnecessary parts of the architecture, or favors a more sparse model introducing zeros to the parameter tensors. On the other hand, quantization reduces the precision of the numbers used to represent model parameters from float32 to float16 up to int8. That can be accomplished after the training procedure (post-training quantization) or during the training procedure (quantization-aware training), adding fake quantization nodes inside the network and making it robust to quantization noise. Indeed, optimizations can potentially result in changes in model accuracy, and so any operation must be carefully evaluated.

In order not to affect the accuracy of our YOLOv3-tiny implementation, we applied basic pruning optimizations with the TensoRT library, removing unnecessary operations and setting to zero irrelevant weights. Indeed, person detection is a critical step in our solution, and it requires the maintenance of a certain level of performance. Nevertheless, in order to test also the performance of the custom TPU ASIC of the Coral Accelerator, we produced a full integer model with TensorFlow-Lite optimizer to be compatible with the hardware of the device. When only applying model pruning we obtained an insignificant accuracy loss; with 8-bit precision the model loses 22% of its original AP_{50}. Indeed, darker scenes, with partially occluded and small targets, are not precisely detected anymore. However, latency and inference costs are significantly reduced using this extreme optimization procedure.

5.2. Inference with Edge AI Accelerators

After the training and optimization procedures, the re-trained model was deployed on the different edge device configurations presented in Section 3. We tested the performance in terms of absorbed power and frame rate in order to outline different hardware solutions for our proposed cost-effective person-following system. A single-board computer, Raspberry Pi 3B+, was used for all configurations that require a host device.

Firstly, we measured the power consumption of the different solutions at an idle condition, and then we executed the model for approximately five minutes to reach steady-state behavior. We directly measured the current absorbed from the power source, thereby obtaining the power consumption of the entire system.

Since the Jetson boards allow the user to select different working power conditions, we tested all of them. The results are presented in Table 4. The second version of the Intel Movidius Neural Stick achieves a higher frame rate with less power consumption. However, either Jetson Nano running modes reach higher performance at the expense of higher current absorption. On the other hand, Jetson AGX Xavier achieves a much higher frame rate on all running modes, but with other levels of power consumption. Finally, full integer quantization greatly reduces the latency of the model running at more than 30 fps with only 7 W. However, as previously stated, the accuracy loss in this last case could compromise the correct functioning of the entire system in certain types of application.

Table 4. Comparison between different devices' power consumption levels and performances achieved with the re-trained and modified version of YOLOv3-tiny. The fps * achieved with the Coral Accelerator are obtained with an int8 weights precision.

Device	**Mode**	V_{al} $[V]$	I_{mean} $[A]$	**P** $[W]$	**fps**
Raspberry Pi 3B+	IDLE	5	0.61	3.075	N/A
RP3 + Neural Stick 1	RUNNING	5	1.2	6	4
RP3 + Neural Stick 2	RUNNING	5	1.12	5.6	5
Jetson Nano	IDLE 10W	5	0.32	1.6	N/A
	RUNNING 10W	5	1.96	9.8	9
	RUNNNING 5W	5	1.4	7	6
Jetson AGX Xavier	IDLE 30W	19	0.35	6.65	N/A
	RUNNING 30W	19	0.91	17.29	30+
	RUNNING 15W	19	0.82	15.58	28
	RUNNING 10W	19	0.62	11.78	15
RP3 + Coral Accelerator	MAX	5	1.40	7	30+ *

5.3. Platform Implementation

We tested the proposed cost-effective person-following system in a real environment with the configuration presented in Table 5; the specifics about its hardware components have been already introduced in Section 3.2. Figure 4 shows the assembled robot adopted for this application.

Table 5. Hardware configuration adopted for practical simulations. Jetson AGX Xavier and Coral USB Accelerator are used to run the model onboard.

HW Materials	
Robotic Platform	TurtleBot3 Waffle Pi
Edge AI Device	NVIDIA Jetson AGX Xavier
RGB-D camera	Intel RealSense Depth Camera D435i

The tests, performed in a real environment, show robot behavior consistent with expectations. The person detection algorithm is high-speed and reached a high level of performance. The network improvement, obtained from the re-training, is considerable (30.09%), and this implies optimal real-time results and perfect control of the movements of the robot that follows the person.

By testing the network on the Jetson Xavier board, we have obtained a high frame rate (30+ fps at the maximum power of the board), as presented in the Table 4. These results affect the velocity of the detection algorithm that runs in real-time, and consequently the frequency of the overall control system, running on the Jetson Xavier, which ranges between 18 and 27 Hz.

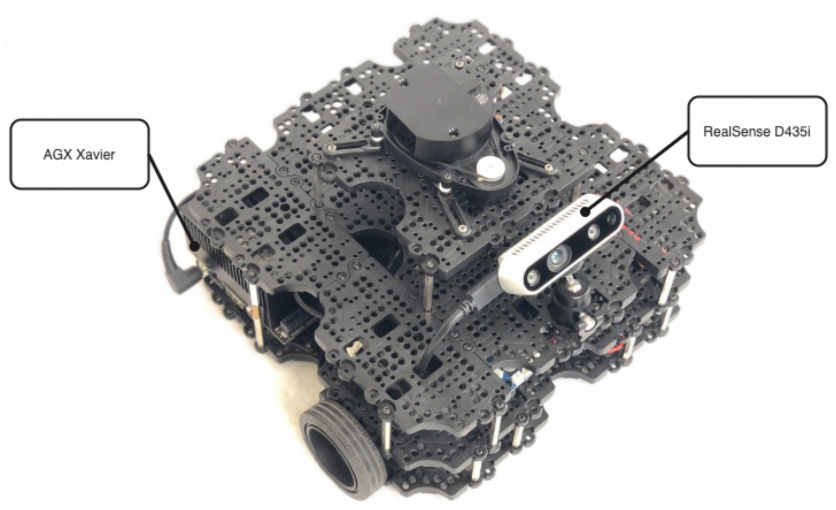


Figure 4. TurtleBot3 Waffle platform with NVIDIA Jetson AGX Xavier and Intel RealSense Depth camera D435i onboard.

During the experiments, conducted in the test environment, we have noticed that using all the outputs of the network to modify the control velocity could be counterproductive because it causes a continuous variation of the inputs of the robot control and consequently a not regular movement of the robot. Differently, imposing the updating of the output of the robot command velocity each 2 Hz, the tests give optimal results: the robot follows the target always in real-time, but with a notable increase in the smoothness of its movement. It is essential to underline that the implementation of the control adopted has been explicitly designed for the use case taken into account, considering an elderly person as the target and the speed limits of the considered robotic platform: ± 0.26 m/s for linear velocity and ± 1.8 rad/s for angular velocity. These limits are reasonable for the considered indoor application since walking sessions are usually short and performed with very limited speed and frequent pauses. However, the control rule can be easily adapted to more performing prototypes if a higher linear speed limit is needed.

During the test phase, the presented control of the robot has been perfected, in particular, considering the characteristics of the platform:

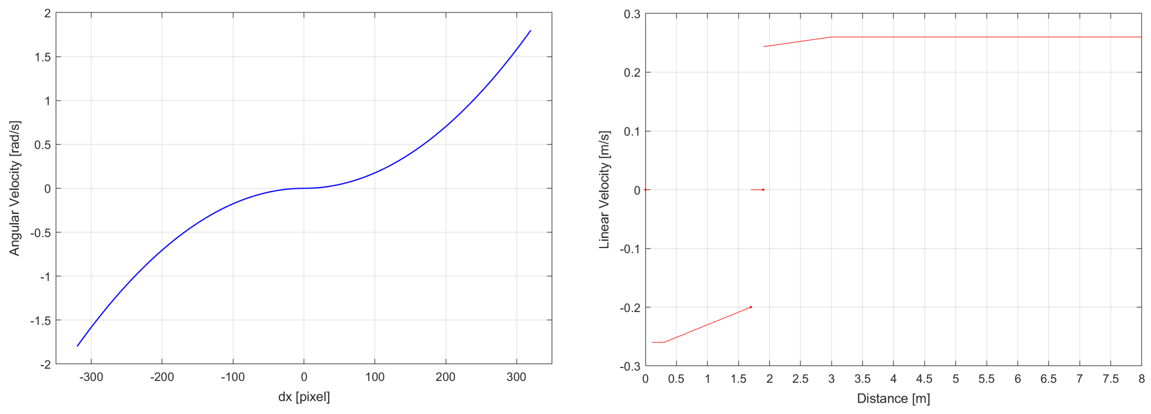
- The optimal distance limits able to define the three areas of the linear velocity control are defined as $m_{vel_{lowelimit}}$ = 1.7 m and $m_{vel_{upperlimit}}$ = 1.9 m. In the range between these two values the robot is in the safe distance zone, so it can only rotate because the linear velocity stays at zero, in order to avoid the generation of any dangerous situations for the target person.
- The best linear increment is computed during both the forward and backward movements of the robot.

Thus, the final obtained values of $m1$, $q1$, $m2$ and $q2$, presented in Section 4, for our platform implementation, are reported in Table 6. Moreover, in Figure 5a the developed angular velocity and linear velocity behaviors are represented while taking into account max_{vel} and min_{vel} of the robotic platform.

It is important to underline that the initial linear velocity control has been designed with different slopes and without singularities. However, during the test phase, trouble has been highlighted: the robot in the restart moving phase proceeded so slowly that it was unable to follow the target without losing it correctly. That is the reason for the introduction of the step singularities, visible on Figure 5b, that, thanks to the small linear velocity of both the robot and the target (the older person), allow one to have a balanced movement and not jerky.

Table 6. The values $m1$, $q1$, $m2$, $q2$ identified, after a test phase, as the best choice of the linear control algorithm, taking into account the robot adopted in our case study, are here reported.

Straight Line	Points
1°: $(m1, q1)$	P_1 (1 m, 0.23 m/s)
	P_2 (3 m, 0.26 m/s)
2°: $(m2, q2)$	P_1 (1 m, −0.23 m/s)
	P_2 (0.3 m, −0.26 m/s)



(**a**) (**b**)

Figure 5. In (**a**) is the angular velocity control plot, computed considering 640 × 480 as the resolution of the camera frame and the limit value of the angular velocity ±1.8 rad/s. Instead, in (**b**) is shown the linear velocity control function, computed considering the depth camera range (from 0.105 m to 8 m), the limit value of the linear velocity of the robotic platform ±0.26 m/s and the safety distance respected inside the interval between 1.7 and 1.9 m.

The overall system has been tested in several environments with different light conditions and target velocities verifying the correctness and completeness of system functionality. The final result meets the demands of an accurate real-time application; the robot moves in safety, consistent with the movements of the person, limiting the chances of losing the target to chase.

We can conclude that we have not realized a simple object tracker, but a person-following method that focuses on cost-effectiveness, since it cuts unnecessary computations to have a low-cost, functional system. Besides, the detection is realized at the edge, so the network is optimized to run on neural accelerators. In this way, it is not necessary to have an expensive computer onboard the robot, which would imply both the increase of the price and the consumption of additional power. The reduction in computational cost and power consumption let us use different types of hardware, presented in Table 1, associated with their respective performances, reported in Table 4.

6. Conclusions

We proposed a cost-effective person-following system for self-sufficient older adults assistance that exploits latest advancements in deep learning optimization techniques and edge AI devices to bring inference directly on the robotic platform with high performance and limited power consumption. We tested different embedded device configurations, and we presented a possible practical implementation to realize the suggested system. The discussed solution is easily replaceable and fully-integrable in pre-existing navigation stacks. Future research may integrate the person-following method with concrete applications and monitoring tools for self-sufficient older adults.

Author Contributions: Conceptualization, M.C.; methodology, A.B.; software, A.B. and V.M.; validation, A.B. and V.M.; data curation, A.B., F.S. and V.M.; writing—original draft, A.B., F.S. and V.M.; writing—review and editing, A.B., F.S. and V.M.; project administration, M.C.; funding acquisition, M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work has been developed with the contributions of the Politecnico di Torino Interdepartmental Centre for Service Robotics PIC4SeR (https://pic4ser.polito.it) and SmartData@Polito (https://smartdata.polito.it).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Islam, M.J.; Hong, J.; Sattar, J. Person-following by autonomous robots: A categorical overview. *Int. J. Robot. Res.* **2019**, *38*, 1581–1618. [CrossRef]
2. *World Population Ageing 2019 (ST/ESA/SER.A/444)*; Population Division, Department of Economic and Social Affairs, United Nations: New York, NY, USA, 2020.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
4. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
5. Pucci, D.; Marchetti, L.; Morin, P. Nonlinear control of unicycle-like robots for person following. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3406–3411.
6. Chung, W.; Kim, H.; Yoo, Y.; Moon, C.B.; Park, J. The detection and following of human legs through inductive approaches for a mobile robot with a single laser range finder. *IEEE Trans. Ind. Electron.* **2011**, *59*, 3156–3166. [CrossRef]
7. Morales Saiki, L.Y.; Satake, S.; Huq, R.; Glas, D.; Kanda, T.; Hagita, N. How do people walk side-by-side? Using a computational model of human behavior for a social robot. In Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, Boston, MA, USA, 5–8 March 2012; pp. 301–308.
8. Cosgun, A.; Florencio, D.A.; Christensen, H.I. Autonomous person following for telepresence robots. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 4335–4342.
9. Leigh, A.; Pineau, J.; Olmedo, N.; Zhang, H. Person tracking and following with 2d laser scanners. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 726–733.
10. Adiwahono, A.H.; Saputra, V.B.; Ng, K.P.; Gao, W.; Ren, Q.; Tan, B.H.; Chang, T. Human tracking and following in dynamic environment for service robots. In Proceedings of the TENCON 2017–2017 IEEE Region 10 Conference, Penang, Malaysia, 5–8 November 2017; pp. 3068–3073.
11. Cen, M.; Huang, Y.; Zhong, X.; Peng, X.; Zou, C. Real-time Obstacle Avoidance and Person Following Based on Adaptive Window Approach. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019; pp. 64–69.
12. Jung, E.J.; Yi, B.J.; Yuta, S. Control algorithms for a mobile robot tracking a human in front. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 2411–2416.
13. Cai, J.; Matsumaru, T. Human detecting and following mobile robot using a laser range sensor. *J. Robot. Mechatron.* **2014**, *26*, 718–734. [CrossRef]
14. Koide, K.; Miura, J. Identification of a specific person using color, height, and gait features for a person following robot. *Robot Auton. Syst.* **2016**, *84*, 76–87. [CrossRef]
15. Brookshire, J. Person following using histograms of oriented gradients. *Int. J. Soc. Robot.* **2010**, *2*, 137–146. [CrossRef]
16. Satake, J.; Chiba, M.; Miura, J. A SIFT-based person identification using a distance-dependent appearance model for a person following robot. In Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, China, 11–14 December 2012; pp. 962–967.

17. Satake, J.; Chiba, M.; Miura, J. Visual person identification using a distance-dependent appearance model for a person following robot. *Int. J. Autom. Comput.* **2013**, *10*, 438–446. [CrossRef]
18. Chen, B.X.; Sahdev, R.; Tsotsos, J.K. Integrating stereo vision with a CNN tracker for a person-following robot. In *International Conference on Computer Vision Systems*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 300–313.
19. Chen, B.X.; Sahdev, R.; Tsotsos, J.K. Person following robot using selected online ada-boosting with stereo camera. In Proceedings of the 2017 14th Conference on Computer and Robot Vision (CRV), Edmonton, AB, Canada, 16–19 May 2017; pp. 48–55.
20. Wang, X.; Zhang, L.; Wang, D.; Hu, X. Person detection, tracking and following using stereo camera. In Proceedings of the Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Qingdao, China, 14–17 October 2017; p. 106150D.
21. Doisy, G.; Jevtic, A.; Lucet, E.; Edan, Y. Adaptive person-following algorithm based on depth images and mapping. In Proceedings of the IROS Workshop on Robot Motion Planning, Vilamoura, Portugal, 7–12 October 2012.
22. Basso, F.; Munaro, M.; Michieletto, S.; Pagello, E.; Menegatti, E. Fast and robust multi-people tracking from RGB-D data for a mobile robot. In *Intelligent Autonomous Systems 12*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 265–276.
23. Munaro, M.; Basso, F.; Michieletto, S.; Pagello, E.; Menegatti, E. A software architecture for RGB-D people tracking based on ROS framework for a mobile robot. In *Frontiers of Intelligent Autonomous Systems*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 53–68.
24. Do, M.Q.; Lin, C.H. Embedded human-following mobile-robot with an RGB-D camera. In Proceedings of the 2015 14th IAPR International Conference on Machine Vision Applications (MVA), Tokyo, Japan, 18–22 May 2015; pp. 555–558.
25. Ren, Q.; Zhao, Q.; Qi, H.; Li, L. Real-time target tracking system for person-following robot. In Proceedings of the 2016 35th Chinese Control Conference (CCC), Chengdu, China, 27–29 July 2016; pp. 6160–6165.
26. Mi, W.; Wang, X.; Ren, P.; Hou, C. A system for an anticipative front human following robot. In Proceedings of the International Conference on Artificial Intelligence and Robotics and the International Conference on Automation, Control and Robotics Engineering, Kitakyushu, Japan, 12–15 July 2016; pp. 1–6.
27. Gupta, M.; Kumar, S.; Behera, L.; Subramanian, V.K. A novel vision-based tracking algorithm for a human-following mobile robot. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *47*, 1415–1427. [CrossRef]
28. Masuzawa, H.; Miura, J.; Oishi, S. Development of a mobile robot for harvest support in greenhouse horticulture—Person following and mapping. In Proceedings of the 2017 IEEE/SICE International Symposium on System Integration (SII), Taipei, Taiwan, 11–14 December 2017; pp. 541–546.
29. Chi, W.; Wang, J.; Meng, M.Q.H. A gait recognition method for human following in service robots. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1429–1440. [CrossRef]
30. Jiang, S.; Yao, W.; Hong, Z.; Li, L.; Su, C.; Kuc, T.Y. A classification-lock tracking strategy allowing a person-following robot to operate in a complicated indoor environment. *Sensors* **2018**, *18*, 3903. [CrossRef] [PubMed]
31. Chen, E. "FOLO": A Vision-Based Human-Following Robot. In Proceedings of the 2018 3rd International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE 2018), Dalian, China, 12–13 May 2018; Atlantis Press: Beijing, China, 2018.
32. Yang, C.A.; Song, K.T. Control Design for Robotic Human-Following and Obstacle Avoidance Using an RGB-D Camera. In Proceedings of the 2019 19th International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, 15–18 October 2019; pp. 934–939.
33. Alvarez-Santos, V.; Pardo, X.M.; Iglesias, R.; Canedo-Rodriguez, A.; Regueiro, C.V. Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot. *Robot. Auton. Syst.* **2012**, *60*, 1021–1036. [CrossRef]
34. Susperregi, L.; Martínez-Otzeta, J.M.; Ansuategui, A.; Ibarguren, A.; Sierra, B. RGB-D, laser and thermal sensor fusion for people following in a mobile robot. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 271. [CrossRef]
35. Wang, M.; Su, D.; Shi, L.; Liu, Y.; Miro, J.V. Real-time 3D human tracking for mobile robots with multisensors. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5081–5087.

36. Hu, J.S.; Wang, J.J.; Ho, D.M. Design of sensing system and anticipative behavior for human following of mobile robots. *IEEE Trans. Ind. Electron.* **2013**, *61*, 1916–1927. [CrossRef]
37. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; pp. 886–893.
38. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
39. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, USA, 8–14 December 2001; pp.511–518.
40. Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
41. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
42. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
43. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of twenty-ninth Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
44. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. *Lect. Notes Comput. Sci.* **2016**, 21–37. [CrossRef]
45. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
46. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
47. Mittal, S. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. *J. Syst. Archit.* **2019**, *97*, 428–442. [CrossRef]
48. Xu, X.; Amaro, J.; Caulfield, S.; Falcao, G.; Moloney, D. Classify 3D voxel based point-cloud using convolutional neural network on a neural compute stick. In Proceedings of the 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 37–43.
49. Kang, D.; Kang, D.; Kang, J.; Yoo, S.; Ha, S. Joint optimization of speed, accuracy, and energy for embedded image recognition systems. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 715–720.
50. Cao, S.; Liu, Y.; Lasang, P.; Shen, S. Detecting the objects on the road using modular lightweight network. *arXiv* **2018**, arXiv:1811.06641.
51. Yang, T.; Ren, Q.; Zhang, F.; Xie, B.; Ren, H.; Li, J.; Zhang, Y. Hybrid camera array-based uav auto-landing on moving ugv in gps-denied environment. *Remote Sens.* **2018**, *10*, 1829. [CrossRef]
52. Mazzia, V.; Khaliq, A.; Salvetti, F.; Chiaberge, M. Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application. *IEEE Access* **2020**, *8*, 9102–9114. [CrossRef]
53. Long, M.; Zhu, H.; Wang, J.; Jordan, M.I. Deep Transfer Learning with Joint Adaptation Networks. *arXiv* **2016**, arXiv:1605.06636.
54. Kuznetsova, A.; Rom, H.; Alldrin, N.; Uijlings, J.; Krasin, I.; Pont-Tuset, J.; Kamali, S.; Popov, S.; Malloci, M.; Duerig, T.; et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv* **2018**, arXiv:1811.00982.
55. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.

56. Chen, L.; Ai, H.; Zhuang, Z.; Shang, C. Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification. In Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 23–27 July 2018; pp. 1–6.
57. Mitzel, D.; Leibe, B. Real-time multi-person tracking with detector assisted structure propagation. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 974–981.
58. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 270–279.
59. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **1964**, *4*, 1–17. [CrossRef]