Deep 3D Perception of People and Their Mobility Aids

Marina Kollmitz*, Andreas Eitel, Andres Vasquez, Wolfram Burgard

University of Freiburg, Department of Computer Science, Georges-Köhler-Allee 80, D-79110 Freiburg, Germany

Abstract

Robots operating in populated environments, such as hospitals, office environments or airports, encounter a large variety of people with some of them having an advanced need for cautious interaction because of their advanced age or motion impairments. To provide appropriate assistance and support robot helpers require the ability to recognize people and their potential requirements. In this article, we present a people detection framework that distinguishes people according to the mobility aids they use. Our framework uses a deep convolutional neural network for detecting people in image data. For human-aware robots it is necessary to know where people are in a 3D world reference frame instead of only locating them in a 2D image, therefore we add a 3D centroid regression output to the network to predict the Cartesian position of people. We further use a probabilistic class, position and velocity tracker to account for false detections and occlusions. Our framework comes in two variants: The depth only variant targets high privacy demands, while the RGB only framework provides improved detection performance for non-critical applications. Both variants do not require additional geometric information about the environment. We demonstrate our approach using a dedicated dataset acquired with the support of a mobile robotic platform. The dataset contains five classes: pedestrian, person in wheelchair, pedestrian pushing a person in a wheelchair, person using crutches and person using a walking frame. Our framework achieves an mAP of 0.87 for RGB and 0.79 for depth images at a detection distance threshold of 0.5 m on our dataset, with a runtime of 53 ms per image. The annotated dataset is publicly available and our framework is made open source as a ROS people detector.

Keywords:
mobile robot, service robotics, convolutional neural networks, object detection, object tracking, people detection, multi-class detection

Acknowledgments

This work has been partially supported by the German Federal Ministry of Education and Research (BMBF), contract number 01IS15044B-NaRKo.

1. Introduction

Robot helper systems aim to assist people in everyday tasks. In health care applications, robots can help nurses lifting patients into and out of their hospital beds [1] and support the treatment of elderly people with dementia [2]. Mobile robotic assistants can further be employed for rehabilitation [3], carry out delivery tasks in hospitals [4, 5, 6, 7, 8], accompany and assist healthcare professionals [9] and support elderly people [10, 11].

The desired assistance and guidance offered by robot helper systems is often subjective and depends on many factors, like the physical condition of the interaction partner. People with walking aids tend to walk slower than people without motion impairments, thus it is desirable that robots adapt their velocity when guiding people with walking aids [12]. In addition, knowledge of motion impairments can help robots to better anticipate and predict the motion of pedestrians [13], which is crucial for socially compliant navigation [14, 15, 16]. The use of walking aids further entails certain limitations, such as the inability to use stairs, and special requirements for door and doorway clearances, elevators or bathroom facilities [17]. A robot assistant should be able to perceive and reason about these special requirements and adapt its behavior accordingly.

In this paper, we address the main challenges for perceiving people according to their mobility aids with deep neural network-based detection and probabilistic position and class estimation. Our system is suitable for either depth or RGB camera input and estimates the class, 3D position, velocity and the tracked motion path of people. It has three main components: deep learning-based 3D object detection, 3D position and velocity tracking, and probabilistic class estimation. The object detection module takes depth or RGB images as input and outputs 2D image bounding boxes and 3D centroids for each box, as depicted in Fig. 1. The probabilistic position and class estimation modules resolve occlusions and provide a probability distribution over the mobility aids classes for each detection, taking the previous observations into account, as visualized in Fig. 2. Our approach is designed for mobile robots and relies on odometry information for tracking.

*Corresponding author.
Email addresses: kollmitz@informatik.uni-freiburg.de (Marina Kollmitz), eitel@informatik.uni-freiburg.de (Andreas Eitel), joseandresv@gmail.com (Andres Vasquez), burgard@informatik.uni-freiburg.de (Wolfram Burgard)

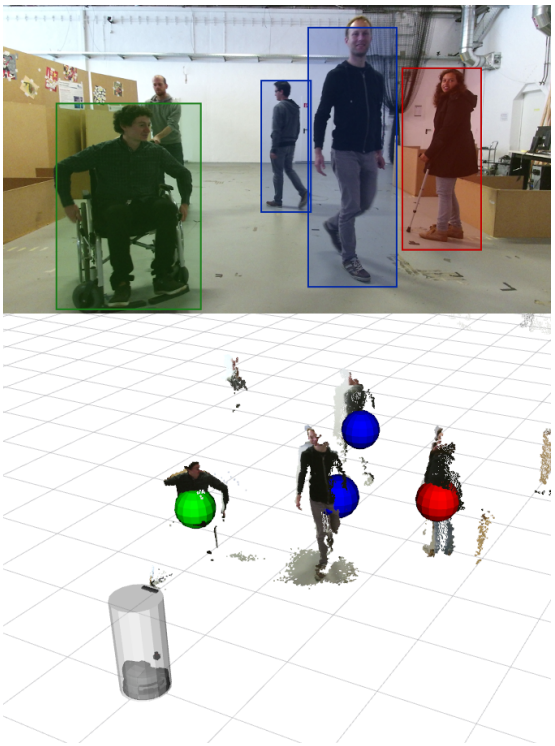

Figure 1: Our perception framework detects people in 2D images and categorizes them according to the mobility aids they use, as shown on the top. It further estimates the 3D centroid of each person from the 2D images, visualized at the bottom as colored spheres. The displayed point cloud only serves as a reference and is not used by our approach.

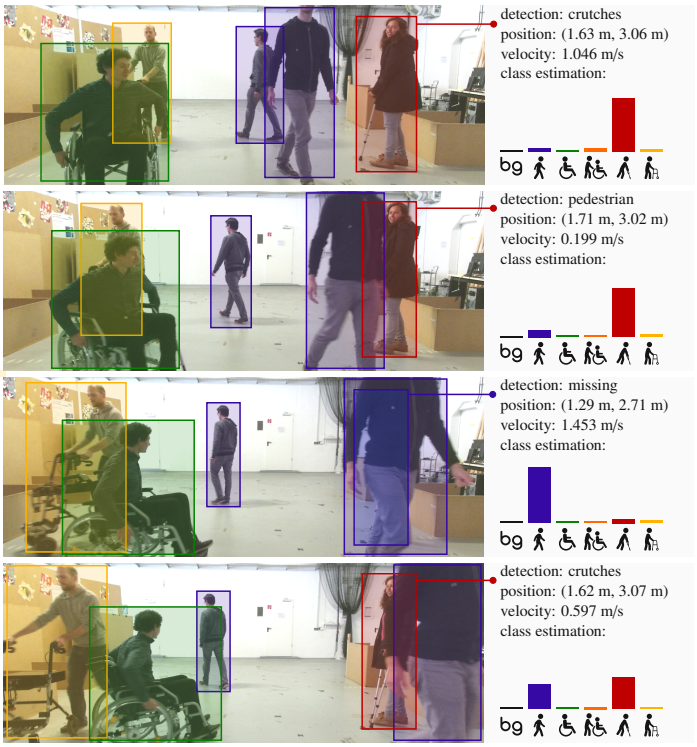

Figure 2: We track people over time to resolve occlusions and false detections. The right box shows the estimated positions, velocities and probabilities over the categories for one person.

This paper extends our previous paper [18], in which we generated object proposals from clustering points clouds generated from depth images. The proposals were then evaluated by a Fast R-CNN [19]. In [18], we obtained the 3D centroids from the mean depth of the proposal cluster points as well as the 2D image bounding boxes. In contrast, this paper builds upon the Faster R-CNN [20] framework, a recent object detection method for 2D bounding box prediction that does not rely on precomputed proposals, but identifies promising image regions itself. We extend the original Faster R-CNN with an additional network output which estimates the 3D centroid depth of each bounding box. This way, we can still detect and categorize people in the 3D space and at the same time profit from the superior object detection performance of Faster R-CNN over Fast R-CNN on 2D images [20]. Our previous approach was limited to the range of the depth sensor, because the proposal generation relied on it. People that are visible in the RGB but not in the depth images could not be detected. Our new system overcomes these limitations and is thus flexible with respect to the employed sensor modality: RGB or depth. Our system further incorporates an adapted version of the probabilistic position, velocity and class estimation module originally presented in [18]. It combines an extended Kalman filter to track the position and velocity of each person and a hidden Markov model to estimate the class of each filter.

In this paper, we further present our extended Mobility Aids Dataset with over 17,000 annotated RGB-D images at 960x540 pixel resolution, originally introduced in our previous work [18]. We collected the dataset with a Kinect 2 camera on a mobile platform in the facilities of the Faculty of Engineering of the University of Freiburg and in a hospital in Frankfurt. We extend the original dataset with 3D centroid depth labels for each annotated 2D bounding box. Further, we include odometry measurements of our robot. The dataset as well as a ROS package of our perception system are publicly available at `http://mobility-aids.informatik.uni-freiburg.de`. The webpage also contains a video of the perception results obtained using our approach.

We evaluate the presented approach on our extended dataset to demonstrate the object detection as well as 3D centroid regression performance. We further analyze the performance of the individual components of our system and their contributions to the overall system. Finally, we test our approach on our mobile robot for a guidance scenario to show that it can be successfully deployed in the physical world and that it enables robots to provide appropriate assistance to people according to their physical impairments.

2. Related Work

This section presents related work for people detection and tracking with mobile platforms, deep learning-based object detection and datasets for people detection.

People Detection and Tracking for Mobile Platforms

Considering the large amount of previous work in the people perception domain we will narrow our discussion on approaches specifically designed for mobile platforms. Hereby, we distinguish the approaches based on the employed sensor modalities, e.g., LiDAR sensors, stereo cameras, and RGB-D, depth or monocular cameras.

In the context of urban driving LiDAR sensors provide a reliable data source for tracking moving objects [21, 22], also in combination with vision data [23]. Several methods include depth information from stereo cameras, which are less precise than LiDAR sensors and therefore follow different tracking pipelines usually in combination with vision-based face or people detectors [24, 25, 26]. Ess et al. [27] present a multi person tracking and detection system that uses stereo cameras mounted on a mobile platform, integrating visual odometry, depth from stereo estimation and pedestrian detection for improved perception. In comparison to the mentioned approaches, we focus on the multi-class aspect of the perception problem that goes beyond classic people detection and therefore requires rich features that only learning-based methods can provide. In addition, besides tracking 3D position and velocity of people we also track the class over time. Ošep et al. [28] combine 2D object detectors and 3D information obtained from stereo cameras for 3D tracking of objects in urban street scenes based on a 2D-3D Kalman filter approach. In [29] they extend their approach to enable tracking of generic unknown objects besides known objects which are classified using Faster R-CNN [20]. In both works the authors focus on traffic scenes and assume that there exists a pretrained object detector for the corresponding classes. In this work we implement a complete detection and tracking system trained on our own dataset and able to run on our mobile robot. Further, we offer an off-the-shelf detection and tracking solution by making the source code publicly available, similar to other frameworks that leverage several existing people detection and tracking methods for mobile robots [30, 31, 32].

For indoor applications RGB-D sensors can provide very reliable depth estimates for close and mid-range distances, which is usually sufficient to cover the typical planning workspace range of a navigation robot. Several people detection and tracking approaches for mobile robots are designed for usage with an RGB-D sensor device. Spinello and Arras [33] present a HOG-based detector, which they also employ in their people tracking framework in combination with a multi-hypothesis Kalman filter [34]. Most prior methods employ manual vision features in combination with 3D point cloud features for dealing with depth data [35, 36], while we use state-of-the-art learning-based features. Similar to others our tracking procedure requires knowledge about the robot position in a world reference frame [37, 32]. In own prior work we show that learned features from deep neural networks improve people detection performance in RGB-D data when compared to manually designed features [38]. Guerry et al. [39] show further detection improvements when leveraging recent state-of-the-art object detectors for the same task. Zimmermann et al. [40] estimate 3D poses of humans from RGB-D data for robotics scenarios.

In our previous approach [18], we perceive people according to their mobility aids in depth images. Our region proposal generator processes the depth information and we evaluate the proposals using Fast R-CNN. The region proposal generator provides the 3D centroid information for each detection. In this work, since Faster R-CNN generates image region proposals itself, we extend it to regress the depth of the centroid pixel in a 2D bounding box using solely image information. Similarly, Chen et al. [41] perform 3D object detection by sampling 3D bounding box proposals from monocular images before feeding those into Fast R-CNN. In contrast to their method we directly regress the depth of the centroid for each bounding box prediction of our Faster R-CNN network. Therefore, the inference time of our method is faster compared to the 1.8s required for sampling proposals reported by Chen et al. [41], which is not fast enough for real robot applications.

The approaches mentioned above consider one person class, while we perform multi-class people detection by destinguishing people according to the mobility aids they use. Other recent work on multi-class people recognition and detection applied to mobile robot scenarios include a LiDAR-based wheelchair/walker detector [42] and a human gender recognition approach [43]. To the best of our knowledge there exists no prior work that presents multi-class people detection on images, applied to service robot scenarios.

Deep Learning for Object Detection

Our work builds upon state-of-the-art object detection methods, specifically region-based convolutional neural networks such as Fast R-CNN [19] and Faster R-CNN [20]. The original Fast R-CNN method is not fast enough for real-time applications, mainly due to slow region proposal algorithms. In the conference version of this paper we dramatically speed up Fast R-CNN inference time by feeding it with bounding box proposals from our very fast depth-based region proposal method [18]. In this paper we show that Faster R-CNN in combination with a smaller backend network achieves higher detection rates at fast inference speed.

Datasets

This paper presents our extended Mobility Aids Dataset which, in addition to 2D image bounding box and multi-class person labels, includes 3D centroid ground truth. The dataset enables the training of multi-class people detection methods for indoor scenarios, e.g., hospitals and public buildings. There exist a few other datasets that contain multi-class labels for people in the context of human attribute recognition [44, 45], but these computer vision datasets do not represent the data and sensors used in mobile robot applications. Most similar to ours is the dataset by Sudowe et al. [46], who recorded video sequences of people from a moving camera for the task of human attribute recognition and the dataset by Linder et al. [43], designed for gender recognition in RGB-D data. In own prior work we present the InOutDoor People Dataset for detection in RGB-D data [38], but it does not contain multiple person classes or 3D centroid labels.

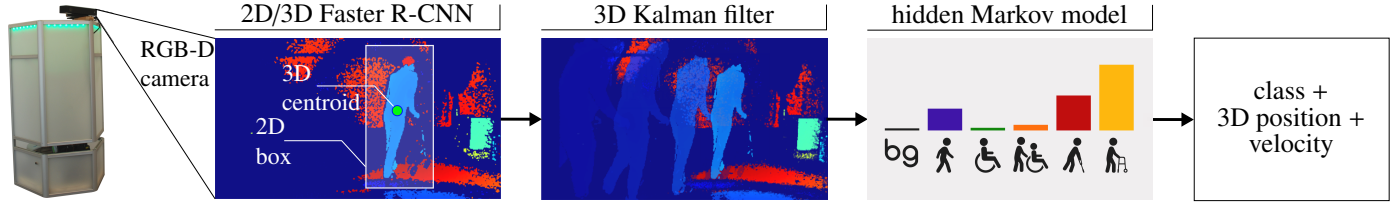

Figure 3: Our perception system operates on 2D images (RGB or depth). We input the images into a Faster R-CNN network, which regresses 2D bounding boxes and 3D centroids of people. We track the resulting detections with a Kalman filter for 3D position and velocity estimation. Further, we employ a hidden Markov model for filtering the class predictions over time.

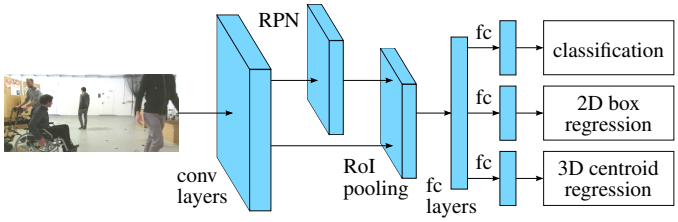

Figure 4: Faster R-CNN architecture with Region Proposal Network (RPN). For each proposal we predict class, 2D bounding box and the 3D centroid of the bounding box.

3. People Perception Framework

Our perception framework processes 2D images, either RGB or depth, and estimates the class as well as the position and velocity of people in a world reference frame. It consists of three modules, as depicted in Fig. 3. Sec. 3.1 introduces the detection stage of our framework, the probabilistic position, class and velocity estimation stages are described in Sec. 3.2.

3.1. People Detection with Faster R-CNN

Our object detection module is based on Faster R-CNN [20], an object detection network which runs in an end-to-end manner from image input to object detection output. We use the open source Detectron implementation [47] of Faster R-CNN. Faster R-CNN uses a region proposal network (RPN) to generate regions of interest (RoIs) in the image, followed by a region-based convolutional neural network to classify the regions of interest and to regress the bounding box of a region. Both network modules share their convolutional layers, resulting in a compact and fast end-to-end system. Furthermore, the resulting network shares convolutions across region proposals, which means that the convolutional feature maps for the input image are computed only once and then used to evaluate each proposal with a regions of interest pooling procedure. Our mobility aids detection extends the original Faster R-CNN by an additional output which estimates the depth value of the 2D image bounding box centroid. We further adapt the loss function for training the network to incorporate the additional network output. The proposal generation, classification and 2D bounding box regression functionalities are identical to the original Faster R-CNN. This section describes the resulting network and specifies the training parameters.

The input of our mobility aids detector is a three channel image, either RGB or color-encoded depth, in the following referred to as DepthJet [48]. In its original form, Faster R-CNN outputs the four 2D bounding box coordinates and the detection scores for each object category. For robotics, however, knowledge about the locations of people in the image plane is often not sufficient. We need to know where people are in the world coordinate frame to be able to interact with them. To this end, we add an additional output to the network – the 3D centroid regression as depicted in Fig. 4. For each proposal, this output estimates the Cartesian distance z_{cam} between the camera and the person's 3D centroid, along the z-coordinate of the camera which points into the image. The 3D centroid of a person with respect to the camera can be computed using the intrinsic camera calibration as

$$x_{\text{cam}} = \frac{x_{\text{im}} - c_x}{f_x} z_{\text{cam}} \tag{1}$$

$$y_{\text{cam}} = \frac{y_{\text{im}} - c_y}{f_y} z_{\text{cam}}, \tag{2}$$

with the optical center $\left(c_x, c_y\right)$ and the focal lengths $\left(f_x, f_y\right)$, both in pixels. The centroid of the person in the image plane $(x_{\text{im}}, y_{\text{im}})$ is the center of the regressed 2D bounding box. The 3D centroid regression layer is added after the last fully connected layer of the network and outputs a single real-valued number for each proposal, using linear neuron activations.

During training, the network jointly optimizes a multi-task loss that contains the object classification and bounding box regression loss, the region proposal loss and the 3D centroid regression loss. Similar to the bounding box regression loss in the original Faster R-CNN, we use the robust loss function $L_{1,s}$ (smooth L1)[19] for the 3D centroid regression loss

$$L_{z,\text{cam}} = \lambda_{z,\text{cam}} \sum_i p^*_{z,\text{cam}} L_{1,s}\left(z_{\text{cam}} - z^*_{\text{cam}}\right) \tag{3}$$

with

$$L_{1,s}(a) = \begin{cases} 0.5a^2 & \text{if } |a| < 1 \\ |a| - 0.5 & \text{otherwise.} \end{cases} \tag{4}$$

The ground truth z-distance between the camera and person i is z^*_{cam}. We use the ground truth label $p^*_{z,\text{cam}}$ to deactivate the 3D centroid regression loss for proposals without centroid depth labels, because the z-distance ground truth is only available for

proposals with non-background object labels. Furthermore, we obtained the centroid depth labels from the depth image of our RGB-D camera, which has a maximum range of ≈8 m. For proposals outside that range, we deactivate the 3D centroid regression loss. Please refer to Sec. 4.1 for the 3D centroid labeling procedure. The weight factor $\lambda_{z,\text{cam}}$ balances the depth regression performance with the image detection output. We found that a value of 10, which is similar to the 2D bounding box regression weights, produces satisfactory depth regression results without significantly diminishing the image detection performance.

We trained our people detection with stochastic gradient descent with a momentum of 0.9 and a weight decay of 0.0001 on a per-image batch size of 256. We used the 2000 best scored RPN proposals for training and a learning rate of 0.0025, which is reduced with a factor of 0.1 after 30.000 and again after 40.000 iterations. We adopt the warm up scheme presented in [49] and reduce the learning rate for the first 500 iterations with a linearly decreasing factor, starting from 1/3 and vanishing to 0.

We train three different network architectures, each with 60.000 iterations: the ResNet-50 [50] architecture as a strong but complex backbone and VGG-CNN-M [51] and GoogLeNet-xxs [38] as faster, more lightweight networks. All networks were pretrained on ImageNet.

At test time, we pass the input image through the Faster R-CNN network and apply class-wise non-maximum suppression to the output bounding boxes. We choose the detections above a class threshold as positive detections and pass them to the probabilistic people tracker presented in the following.

3.2. Probabilistic Position, Velocity and Class Estimation

The probabilistic people tracker is adapted from our previous work [18]. Previously, we transformed the image detections into the world reference frame before processing them in the tracking module. In contrast, in our current approach the transformation is part of the sensor model, which requires linearization and the use of an extended Kalman filter. This leads to a more accurate estimate of the sensor noise, since its covariance matrix is now measured in the native sensor coordinates, the image axes and the centroid depth, and thus is mostly independent of the relative position between the detected person and the camera.

The detection stage provides a set of coordinates for each detected person of the form $(x_{\text{im}}, y_{\text{im}}, z_{\text{cam}}, c')$, where c' is the perceived class. Our position, velocity and class estimator computes the belief $\text{Bel}(\mathbf{x})$ of the person state $\mathbf{x} = (x_w, y_w, z_w, \dot{x}_w, \dot{y}_w, c)^T$, where (x_w, y_w, z_w) denote the person's position and $(\dot{x}_w, \dot{y}_w)$ his or her velocity on the ground plane, both in a fixed world coordinate frame. The class of the person is denoted by c and includes pedestrian, person in wheelchair, pedestrian pushing a person in a wheelchair, person using crutches, person using a walking frame and background. The estimator combines two modules: An extended Kalman filter (EKF)-based multi-tracking module and a hidden Markov model (HMM)-based class estimation module. The estimator manages one track for each perceived person which performs both EKF and HMM filtering.

The EKF state is five-dimensional, $\mathbf{x}_K = (x_w, y_w, z_w, \dot{x}_w, \dot{y}_w)^T$, and uses a constant velocity motion model with the state transition matrix

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5}$$

where Δt is the filter time interval. For the process noise, we adopt the piecewise white noise model as described by Labbe [52] and assume the system dynamics f are disturbed by constant piecewise noise $\mathbf{w}$:

$$f(\mathbf{x}_K) = \mathbf{F}\mathbf{x}_K + \mathbf{w}\mathbf{\Gamma} \tag{6}$$

We model $\mathbf{w}$ by accelerations a_x and a_y and a height error σ_z, which are independent:

$$\mathbf{w} = \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & \sigma_z \end{pmatrix} \tag{7}$$

The noise gain $\mathbf{\Gamma}$ describes how the noise $\mathbf{w}$ propagates into the state space and is modeled by

$$\mathbf{\Gamma} = \begin{pmatrix} \Delta t^2/2 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 \\ 0 & 0 & 1 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \end{pmatrix} \tag{8}$$

The process noise covariance $\mathbf{Q}$ is then calculated as

$$\mathbf{Q} = \mathbf{\Gamma}\mathbf{w}\mathbf{w}^T\mathbf{\Gamma}^T. \tag{9}$$

For the correction step, we integrate the observations $\mathbf{z} = (x_{\text{im}}, y_{\text{im}}, z_{\text{cam}})^T$ from the detection stage. We first need to design the measurement function

$$h(\mathbf{x}_K) = \mathbf{z}. \tag{10}$$

It maps the state to the measurement and, since the state is given in a fixed world coordinate frame, requires to transform the state first to the camera and secondly to the image frame. The transformation between the camera and the fixed world coordinate system is determined by

$${}^c\mathbf{T}_w = ({}^w\mathbf{T}_r \cdot {}^r\mathbf{T}_c)^{-1}, \tag{11}$$

where ${}^w\mathbf{T}_r$ denotes the transformation between the world coordinate frame and the robot base, which we obtain from odometry, and ${}^r\mathbf{T}_c$ is the known transformation between the robot base and the camera. For the first step, we extract the state position and transform it into the camera frame:

$$\begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \\ 1 \end{pmatrix} = {}^c\mathbf{T}_w \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{12}$$

Table 1: Default threshold values of the estimation module.

threshold	value
max. Euclidean distance $d_{\max}$	1.0 m
max. Mahalanobis distance $\delta_{\max}$	7.815
max. position uncertainty $\sigma_{xy,\max}$	4.0 m

As a second step, we need to project x_{cam} and y_{cam} into the image plane, using the intrinsic camera calibration and following Eq. 1 and Eq. 2, by

$$x_{\text{im}} = \frac{x_{\text{cam}}}{z_{\text{cam}}} f_x + c_x \tag{13}$$

$$y_{\text{im}} = \frac{y_{\text{cam}}}{z_{\text{cam}}} f_y + c_y. \tag{14}$$

The measurement function is non-linear with respect to the state $\mathbf{x}_K$, which means that we need to employ an extended Kalman filter and use the Jacobian $\mathbf{H}$ of h in the correction step. We obtain the observation noise $\mathbf{R}$ experimentally by analyzing the Faster R-CNN detections $\mathbf{z}$ and the corresponding ground truth image bounding box and centroid depth labels $\mathbf{z}^*$ of people in the test dataset 1 (Sec. 4.1). To this end, we compile a data matrix $\mathbf{A}$ with N columns

$$a_{*i} = \mathbf{z}_i - \mathbf{z}_i^*, \tag{15}$$

one for each detection-label pair in the dataset, and specify the observation noise as the covariance of $\mathbf{A}$.

For the data association between tracks and observations, we consider the pairwise Mahalanobis distances

$$\delta_{ij}^2(t) = \mathbf{v}_{ij}^T(t)\hat{\mathbf{S}}_{ij}^{-1}(t)\mathbf{v}_{ij}(t) \tag{16}$$

$$\text{with} \quad \mathbf{v}_{ij}(t) = \mathbf{z}_i(t) - h(\hat{\mathbf{x}}_{K,j}(t)) \tag{17}$$

$$\text{and} \quad \hat{\mathbf{S}}_{ij}(t) = \mathbf{H}_j(t)\hat{\mathbf{P}}_j(t)\mathbf{H}_j^T(t) + \mathbf{R}, \tag{18}$$

where $\hat{\mathbf{x}}_K(t)$ and $\hat{\mathbf{P}}(t)$ are the predicted state mean and covariance at time t and the observation and track indices are i and j. We only pair observations if the Mahalanobis distance is below a fixed threshold $\delta_{\max}$. Furthermore, we enforce a maximum Euclidean distance $d_{\max}$ between detections and position hypotheses. If for one observation both thresholds are satisfied for multiple tracks, we pair it to the one with the highest probability density value

$$p_{ij} = \frac{1}{\sqrt{(2\pi)^3 \det \hat{\mathbf{S}}_{ij}(t)}} \exp\left(-\frac{1}{2}\mathbf{v}_{ij}^T(t)\hat{\mathbf{S}}_{ij}^{-1}(t)\mathbf{v}_{ij}(t)\right). \tag{19}$$

If multiple observations are paired with one track at time t, we update it successively with all paired observations. If an observation was not paired, a new track is initialized. Finally, if there is no observation for a track, we perform a prediction without observation update.

Each track also has one HMM associated to it for estimating the class c of the tracked person, according to

$$p(c_t \mid c'_{1:t}) = \eta p(c'_t \mid c_t) \sum_{c_{t-1}} p(c_t \mid c_{t-1}) p(c_{t-1} \mid c'_{1:t-1}). \tag{20}$$

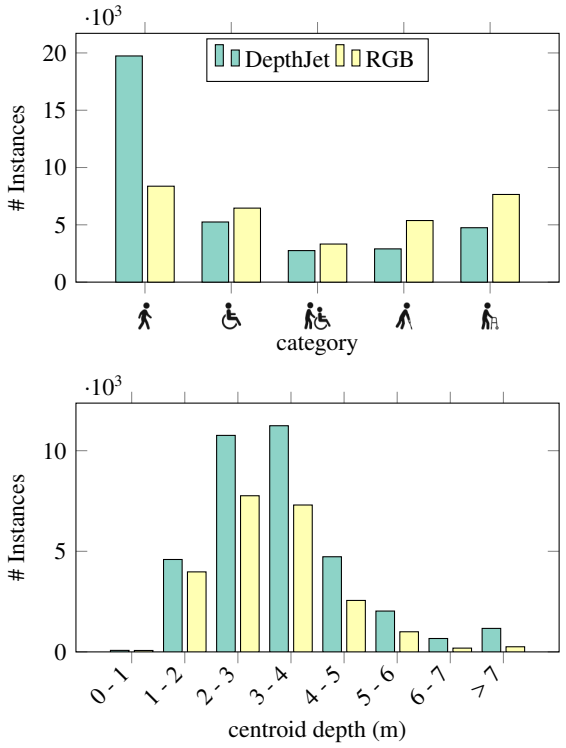

Figure 5: Number of instances per class (top) and per 3D centroid depth (bottom) in the Mobility Aids Dataset.

It models the probability of the tracked person to belong to a given class c_t, given the past observations $c_{1:t}$. Here, c_t is hidden, since we only get observations c'_t for it. The measurement model $p(c'_t \mid c_t)$ connects the hidden with the observed variable for time step t. The HMM further assumes that the class c_t can randomly change from one time step to the next, represented by the transition model $p(c_t \mid c_{t-1})$. In a hospital, a possible transition could be person with crutches → pedestrian → person in wheelchair for a patient who has just finished physiotherapy and hands over his crutches to return to his wheelchair. We initialize the HMM with a uniform prior $p(c_1)$ over all categories.

The softmax output of deep neural network classifiers like Faster R-CNN could be interpreted as $p(c_t \mid c'_t)$. However, training with one-hot encoded labels results in very peaky distributions and over-confident estimates. Therefore, we will not employ the network scores directly for filtering. Instead, we analyze the detections and labels for the test dataset 1 to determine the underlying probability distributions statistically. The amount of detections for one class given a certain label determines the measurement model $p(c'_t \mid c_t)$ of the HMM. It corresponds to the normalized confusion matrix of our classifier. The transition model $p(c_t \mid c_{t-1})$ is given by the amount of transitions from one class to the other with respect to the total number of transitions. Due to the limited amount of examples in the validation set, we might not observe all class transitions and confusions, even if they are possible. Therefore, we assign small probabilities to all unobserved but possible transitions and false detections, using a Dirichlet prior.

The data association for the HMM is given by the EKF. If

Table 2: Object detection performance in terms of average precision (AP, in %), detection only, on the Mobility Aids Dataset. We show the 2D image APs and the APs at a detection distance threshold of 0.5 m.

													mAP	
	Method	Network	0.5 m	2D	0.5 m	2D	0.5 m	2D	0.5 m	2D	0.5 m	2D	0.5 m	2D
DepthJet	Fast R-CNN	G-xxs [18]	75.8	81.2	94.8	96.0	49.6	51.8	76.3	76.6	63.7	64.8	**72.04**	**74.10**
	Faster R-CNN centroid	R-50	80.6	84.5	92.5	96.6	77.4	84.9	85.6	88.3	67.8	68.6	**80.78**	**84.61**
		VGG-M	77.3	81.5	93.5	95.4	78.8	81.0	80.8	81.5	66.5	67.7	**79.38**	**81.41**
		G-xxs	77.0	82.8	93.6	94.8	67.3	74.7	84.3	84.7	67.9	69.2	**78.01**	**81.23**
RGB	Fast R-CNN	G-xxs [18]	68.1	74.9	91.0	92.0	76.1	72.4	71.5	74.2	80.1	82.2	**77.37**	**79.16**
	Faster R-CNN centroid	R-50	84.3	93.5	95.5	98.8	93.9	95.0	95.0	96.8	89.4	98.5	**91.62**	**96.52**
		VGG-M	73.1	87.2	92.4	98.4	94.8	98.3	88.2	91.3	88.2	98.3	**87.35**	**94.71**
		G-xxs	74.6	89.0	91.9	98.1	90.4	96.3	88.4	93.4	94.7	97.3	**87.98**	**94.84**

there is no observation c_t for a track at time step t, we treat it as a background detection in the HMM. The position, velocity and class estimator removes tracks with a standard deviation in position above a threshold $\sigma_{xy,\max}$. Furthermore, tracks where the background class is dominant are deleted. Tab. 1 summarizes the threshold values we used during the experiments.

4. Experiments

In this section we introduce our dataset and evaluate the performance of the Faster R-CNN detection module and our entire perception framework. Additionally, we present a real-robot scenario where the robot uses our perception pipeline to give special assistance in a person guidance task.

4.1. Mobility Aids Dataset

We annotated the RGB and DepthJet images in out Mobility Aids Dataset on a 2D bounding box level with additional 3D centroid depths for each bounding box to train and evaluate our approach. Images were collected in the facilities of the Faculty of Engineering of the University of Freiburg and in a hospital in Frankfurt using a mobile Festo Robotino robot, equipped with a Kinect 2 camera mounted 1 m above the ground and capturing images at 15 frames per second. The robot was controlled by a notebook computer running ROS (Robot Operating System). The 3D centroid labeling procedure is described in Appendix A.

The dataset is subdivided into subsets for training, validation, and testing. We use two test sets to evaluate the performance of the approach. Test set 1 contains 4317 frames and we use it to evaluate the detection methods. We use test set 2 for testing our method in combination with our probabilistic position, velocity and class estimation module. It contains a total of 1801 frames merged from 4 video sequences and in contrast to test set 1 the ground truth labels consider also occluded objects. Note that we use test set 1 as a validation set to tune the parameters of the HMM and EKF (Sec. 3.2). Fig. 5 shows the number of instances for each class and for different centroid depth intervals contained in the Mobility Aids Dataset, for DepthJet and RGB. Note that some images inside the hospital were only recorded in DepthJet because of privacy concerns. Furthermore, some people visible in the RGB images are not visible in the DepthJet images, because of the limited depth range of the camera.

4.2. Object Detection Performance

In this section we evaluate and compare the object detection performance of the Faster R-CNN module in combination with our 3D centroid regression separately, without the tracking module. We compare our detection module to our previous work [18], which was also designed and trained for detecting people according to their mobility aids. One major difference between this approach and our previous work is the estimation of the 3D centroids for the detections. Previously, we used clustered point clouds obtained from the depth images to generate detection proposals, which were then processed by Fast R-CNN. We determined the centroid depth as the mean depth of all proposal cluster points. In contrast, in our Faster R-CNN-based framework we regress the 3D centroid in the network. We further compare different backbone architectures: ResNet-50 [50] (in the following denoted by R-50), VGG-CNN-M [51] (VGG-M) and GoogLeNet-xxs [38] (G-xxs), which we also used in our previous approach.

We use the standard mean average precision (mAP) metric for evaluation. To calculate the mAP, we pair each detection to the ground truth example with the highest 2D bounding box intersection over union above a threshold of 0.5. The detections are paired with decreasing confidence and each label can only be paired to one detection. To include the 3D centroid regression performance in the metric, we additionally only count detections for which the absolute difference between the estimated and true centroid depth is below a threshold and vary this threshold during evaluation. Detections with depth errors above the threshold are regarded as false negatives. Since we cannot determine the depth errors for detections and labels that could not be paired, they are regarded as false examples irrespective of the depth error threshold. Note that some examples do not have depth labels (see Appendix A). Detections paired to those labels are ignored and do not contribute to the metric.

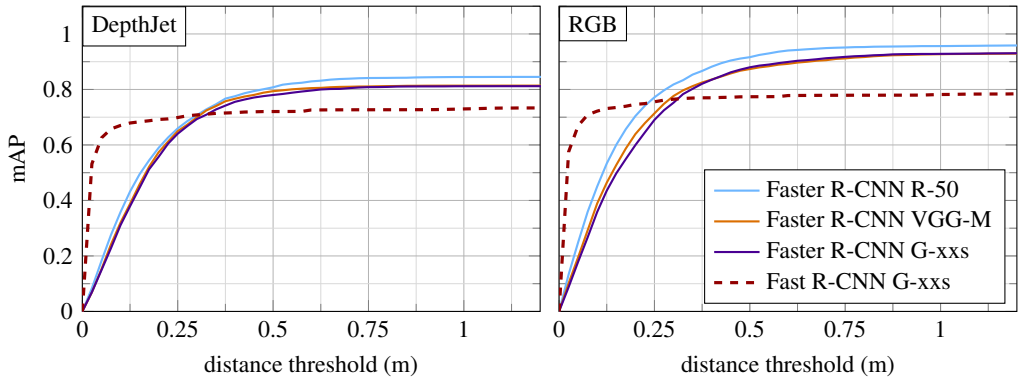

Figure 6: Mean average precision comparison of Fast- and Faster R-CNN with different backbone architectures as a function of the detection distance threshold.

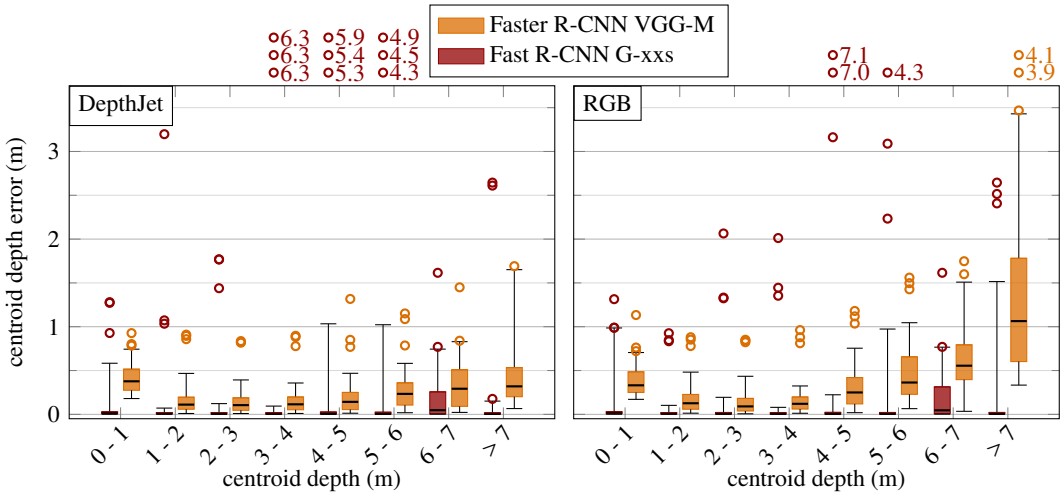

Figure 7: Variation of the centroid depth error with respect to the distance of people to the camera. The centroid depth error is the absolute distance between true and regressed centroid depth. The boxes show the interquartile range with the median, while the whiskers mark the 5th and 95th percentile. For visibility, we only mark the three greatest outliers at each interval with circles and put them above the plot area with the error value next to them if they exceed the range of the plot.

However, unpaired labels without depth information still count as false negatives. Due to the ignored detections, the 2D image mAP is different from the depth error mAP, even as the threshold approaches infinity.

Tab. 2 compares the depth error APs for the Mobility Aids Dataset at a threshold of 0.5 m as well as the 2D image APs, ignoring the depth labels. Note that we also update prior results presented in [18] for GoogLeNet-xxs with Fast R-CNN by applying minor modifications during test time (we allow multiple detections of different classes per segment, which improves mAP). Consistent with the results in [20], Faster R-CNN clearly outperforms Fast R-CNN for the 2D image mAP metric. In addition, the depth error mAPs at 0.5 m for our approach are higher than for Fast-RCNN for all backbone architectures. Fig. 6 shows the evolution of the mAP as a function of the depth error threshold for the different methods. For all backbones architectures, our approach surpasses the performance of our prior work after a threshold of 0.3 m. Our previous work can estimate the centroid depths very precisely, as illustrated by the steep ascent of the mAP curve at low distance thresholds. The mAP curve, however, ends at a much lower level compared to our approach, which shows that it misses a lot of examples that our approach can detect. The differences are especially prominent for the RGB detection results, which is explained by the limited depth range of the camera. Fig. 7 compares the centroid regression performance in more detail. It shows the absolute distance between the true and estimated centroid depth at different depth intervals. Note that we only evaluate the depth regression error for positive detection-label pairs. It is again visible that Fast R-CNN yields in general lower centroid depth errors. However, it also produces a few strong outliers, likely because some of the proposals were generated from depth information not belonging to the detected persons. Our approach produces fewer strong outliers, but the depth regression errors are overall higher. We can also see that our approach yields better centroid depth regression performance for the DepthJet dataset, which is expected since the DepthJet images include the color-encoded depth information. For both image modalities, the medium range between 1 m and 5 m has the lowest depth errors. This range corresponds to possible human-robot interactions within a short time window. For the greater distances, the error increases. Especially on RGB images we need

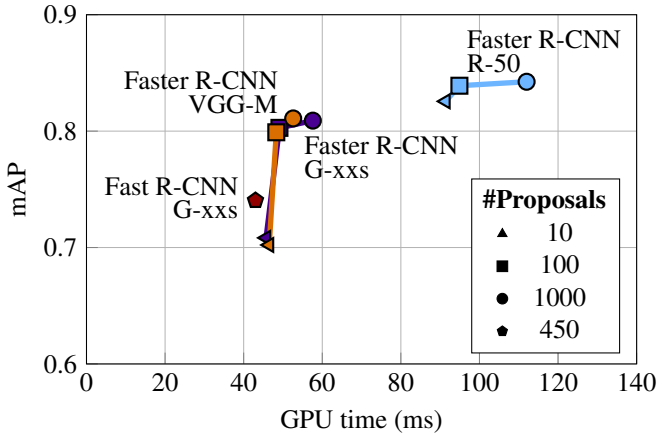

Figure 8: Runtime vs. mAP comparison for Fast- and Faster R-CNN with different backbone architectures, for the DepthJet dataset.

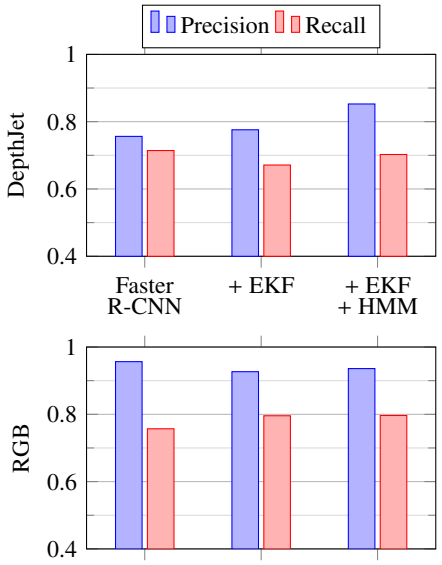

Figure 9: Object detection performance evolution with respect to stand-alone Faster R-CNN, VGG-M backbone.

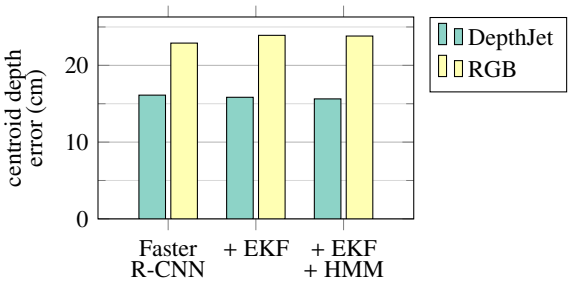

Figure 10: Depth regression performance evolution with respect to stand-alone Faster R-CNN, VGG-M backbone.

to expect larger depth errors with increasing distance. Future work should further investigate how the 3D object detection performance for the close and far range can be improved. Nevertheless, the results show that our work presents a great improvement with regards to our previous work, because it can perceive people that were previously missed entirely. Applications that do not require centimeter accuracy can profit greatly from our new approach. Fig. 11 shows qualitative detection results for RGB and DepthJet.

We further compare the runtime and mAP performance of the methods, see Fig. 8. To this end, we vary the number of top scoring proposals evaluated by Faster R-CNN, between 10, 100 and 1000 proposals per image. With Fast R-CNN we used on average 450 proposals per image [18]. Faster R-CNN with the VGG-M network architecture provides the best tradeoff between runtime and performance. With 100 proposals for each image, the forward pass takes 48 ms and the mAP is 0.80. Measurements are obtained using a computer with 12-Core CPU and a GeForce GTX TITAN X with 12GB of memory.

4.3. Framework Detection Performance

We evaluate our complete framework on our test set 2 to assess the contribution of the different framework modules for the overall detection performance. We compare the performance in terms of precision and recall rather than mAP, because the tracker processes thresholded detections. To be comparable to Tab. 2, we choose the detection thresholds from the mean average precision points of test set 1. Furthermore, we evaluate the mean absolute distance between the predicted and true 3D centroid depth. Fig. 9 shows the precision and recall scores of the framework stages for RGB and DepthJet. Test set 2 is especially challenging because of occlusions, which explains that the recall scores are lower than expected from the test set 1 evaluation. In the DepthJet case, the EKF stage decreases the recall, while the precision only improves slightly. The addition of the HMM finally boosts the precision by ten percent points compared to detection only and recovers the recall back to the detection level. For the RGB case, the tracking module improves recall by almost four percent points, while the precision is slightly decreased by two percent points. The detector is already very strong, so that filtering over time can resolve occlusions and thus improves recall, but filter effects like errors in the assumptions of constant velocity or wrong data associations impact the precision. Fig. 10 summarizes the centroid depth regression error for the detection stages, which is not greatly influenced by the tracking stages. All in all, the experiments confirm that the tracking module has positive effects on the detection performance, even for the already strong RGB detector.

4.4. Person Guidance Scenario

To show the applicability of our framework to a real-world service robot task, we test our system in a person guidance scenario. The task of the robot is to guide visitors to the professor's office in our lab, building 80 at the Faculty of Engineering of the University of Freiburg. The professor's office is located at the first floor, opposite the staircase at the main entrance. An elevator is available at the other side of the corridor.

This experiment is an extension of the one presented in our previous paper [18]. Previously, the robot selected the desti-

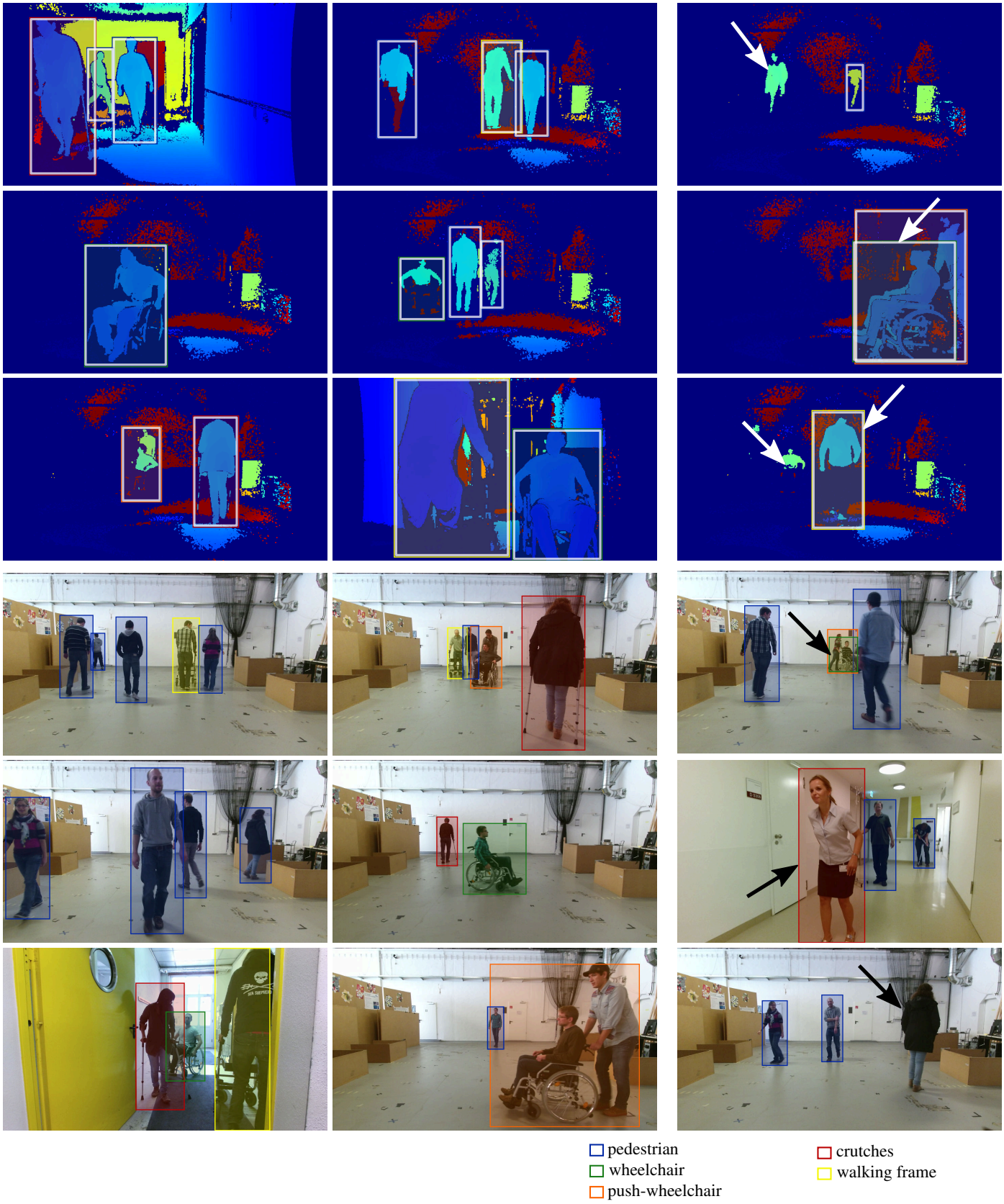

Figure 11: Qualitative object detection results obtained using the Faster R-CNN network with VGG-M backbone, for DepthJet and RGB. Left: positive examples. Right: cases of failure with missed or multiple detections and wrong classifications.

Figure 12: We use our framework to provide assistance in a person guidance experiment. The task of the robot is to guide all pedestrians to the nearest staircase (left image) and all people with mobility aids to the elevator (right image).

nations based on the perceived mobility aids categories of the visitors and traveled at a lower velocity when guiding people to the elevator, but it navigated to the destination without further considering the followers. In this experiment, the robot tracks the followers while navigating and constantly adapts its velocity to them.

Our robot uses a laptop computer with an 8-Core CPU and a GeForce GTX 1080 with 8GB of memory. We use our RGB network with the probabilistic class and velocity estimation module to process the color images of the Kinect 2 camera mounted on the robot at approx. 15 frames per second. We do not use the depth data of the Kinect 2 for this experiment. To ensure that the followers remain in the field of view of the robot, we point the camera to the back of the robot. For the navigation parts we employ the ROS navigation stack [53] in combination with a laser rangefinder pointing to the front of the robot for localization and obstacle avoidance. We use the global_planner package from the ROS navigation stack to generate the global navigation paths. For the local path planning, we adopt the omni_path_follower ROS package [54], which generates velocities for omnidirectional robots to closely follow a navigation path.

We select the initial waiting pose of the robot in the hallway of the ground floor as well as two goal poses, see Fig. 12. At the waiting pose, the robot is facing the wall while the camera is pointing backwards into the hallway. The robot observes an area of interest 3 m in front of the camera and within ±20° from its center. Once it detects a person in this area with an absolute velocity of less than 0.25 m s^{-1}, it starts to navigate to one of the two goals. For pedestrians without perceived motion impairments, it navigates to the goal by the stairs; people with mobility aids are guided to the elevator. The robot uses predefined speech commands to ask the visitors to follow it and inform them how to proceed to the professor's office once the navigation goal is reached. Upon reaching the destination, the robot returns to the waiting pose and waits for the next visitor.

To make sure that the visitors can follow the robot, it keeps track of them during navigation. To this end, the robot retains the track associated to the follower until it reaches its goal. During navigation, the robot always turns the camera towards this person by rotating the base with a rotational velocity of

$$\omega = k_\omega \arctan(\frac{y_r}{-x_r}) \tag{21}$$

during path following, where x_r and y_r denote the position of the person in the robot coordinate frame and k_ω is a proportional gain. The robot coordinate system is at ground level with the x-axis pointing forward and the y-axis pointing to the left. Note that since our robot is omnidirectional, it can follow a path at arbitrary orientations. This enables it to perform a base rotation for keeping the follower in view while navigating to the goal.

The perceived mobility aids give some indication of the preferred velocity of the follower, since people with mobility aids tend to move slower than people without motion impairments [12]. Therefore, our robot chooses an initial guidance velocity of $v_0 = 0.2\,\mathrm{m\,s^{-1}}$ when guiding a person with walking aids, compared to $v_0 = 0.4\,\mathrm{m\,s^{-1}}$ when guiding a pedestrian. However, the motion capabilities likely vary greatly from person to person. Therefore, our robot adjusts its guidance velocity to the person at each time step t to

$$v_t = v_{t-1} + k_v\,(d - \mid x_r \mid)\,. \tag{22}$$

Here, d is a fixed target distance between the robot and the person and k_v is a proportional gain. We additionally restrict the guidance velocity to $0\,\mathrm{m\,s^{-1}} \leq v_t \leq 0.5\,\mathrm{m\,s^{-1}}$. The target distance d is determined as the distance at which the person approached the robot in the waiting area, plus a small offset of +0.2 m to account for the robot driving ahead. If the track of the follower is removed because the background class is dominant or the position uncertainty is too high, the robot considers a person as lost. When it looses the follower, the robot turns towards the path and travels to its destination with the initial guidance velocity v_0.

We tested twenty guidance runs with different people from our lab, four for each of the mobility aids categories: pedes-

trian, person with crutches, person in wheelchair, person with walking frame and person pushing another person in a wheelchair. We marked the waiting area between 1.5 m and 3 m in front of the camera with tape on the floor and asked the participants to approach the robot at the waiting area and follow it once it gives the speech command. Furthermore, we asked the test subjects to keep a distance to the robot roughly as indicated by the tape during the entire experiment to make sure they stay in the field of view of the camera. Additionally, we told the participants that the robot would adjust its velocity to them, so they could walk as fast as they like.

In all of the runs, the robot perceived the correct mobility aids category and successfully navigated the follower to the right destination. Furthermore, the robot successfully kept track of its follower until it reached the goal in seventeen runs. It lost track of the person in one run with a pedestrian and in two runs where a person was pushing another person in a wheelchair. In these runs, the people came too close to the camera and were therefore not detected for multiple frames. A different camera with a wider field of view could be used to solve this problem. Furthermore, the robot could try to find the follower again, maybe taking visual features into account. This is, however, out of the scope of this paper and remains for future work. The tracking module estimated the correct class of the follower in 92.9 % percent of all frames, over all runs. The mean guidance velocity of the robot was $0.38\,\mathrm{m\,s^{-1}}$ with a standard deviation of $0.09\,\mathrm{m\,s^{-1}}$. Here, the robot moved at an average speed of $0.36\,\mathrm{m\,s^{-1}}$ when guiding people with walking aids and at $0.42\,\mathrm{m\,s^{-1}}$ when guiding pedestrians. However, these velocities are not very meaningful since all test subjects were young and healthy. They were physically able to keep up with the robot, therefore they likely co-adapted to its velocity. Many of the test subjects, however, tried different velocities during the experiment and also stopped to test the robot behavior. Some test subjects reported that the robot took too long to adapt its velocity and only started moving again after a stop when they came very close. More sophisticated approaches to person guidance could be used to generate a more natural and prompt guidance behavior, but exceed the scope of this paper.

The experiment confirms that our approach can be successfully applied on a moving robot in an authentic environment. Further, it shows how our approach can be used to give appropriate, individual assistance to people, according to their needs.

5. Conclusion

We proposed a perception system to detect and distinguish people according to the mobility aids they use, based on a deep neural network and supported by tracking and class estimation modules. Our approach shows a significant increase in object detection performance, compared to a Fast R-CNN baseline with depth-based proposal generation. We added a 3D centroid regression output to our network which enables us to estimate the 3D centroids of people from image data only and without additional geometric information. In our person guidance experiment we showed that our detection pipeline enables robots to provide individual assistance to people with advanced needs. In the future, we will further examine how the additional information provided by our framework can improve the behavior of robots in populated environments, for example during autonomous navigation.

References

[1] T. L. Chen, C. C. Kemp, Lead Me by the Hand: Evaluation of a Direct Physical Interface for Nursing Assistant Robots, in: Proc. of the ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI), 2010.

[2] K. Wada, T. Shibata, T. Musha, S. Kimura, Robot Therapy for Elders Affected by Dementia, IEEE Engineering in Medicine and Biology Magazine 27 (4) (2008) 53 – 60.

[3] J. Eriksson, M. J. Mataric, C. J. Winstein, Hands-off Assistive Robotics for Post-Stroke Arm Rehabilitation, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2005.

[4] J. M. Evans, HelpMate: An Autonomous Mobile Robot Courier for Hospitals, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 1994.

[5] M. Y. Shieh, J. C. Hsieh, C. P. Cheng, Design of An Intelligent Hospital Service Robot and Its Applications, in: Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics (SMC), 2004.

[6] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, K. Yoshida, Developing a mobile robot for transport applications in the hospital domain, Robotics and Autonomous Systems 58 (7) (2010) 889 – 899.

[7] F. Capezio, F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, P. Vernazza, T. Vernazza, R. Zaccaria, Mobile Robots in Hospital Environments: an Installation Case Study, in: Proc. of the Eur. Conf. on Mobile Robots (ECMR), 2011.

[8] J. Bačík, F. Ďurovský, M. Biroš, K. Kyslan, D. Perduková, S. Padmanaban, Pathfinder – Development of Automated Guided Vehicle for Hospital Logistics, IEEE Access 5 (2017) 26892–26900.

[9] R. Tasaki, M. Kitazaki, J. Miura, K. Terashima, Prototype Design of Medical Round Supporting Robot "Terapio", in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2015.

[10] B. Graf, M. Hans, R. D. Schraft, Care-O-bot II – Development of a Next Generation Robotic Home Assistant, Autonomous Robots 16 (2) (2004) 193 – 205.

[11] M. E. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramkrishnan, S. Engberg, J. Matthews, J. Dunbar-Jacob, C. McCarthy, S. Thrun, M. Montemerlo, J. Pineau, N. Roy, Pearl: A Mobile Robotic Assistant for the Elderly, in: Proc. of the AAAI Workshop on Automation as Caregiver, 2002.

[12] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, V. Verma, Experiences with a Mobile Robotic Guide for the Elderly, in: Proc. of the Nat. Conf. on Artificial Intelligence (AAAI), 2002.

[13] J. B. Fernando, T. Tanigawa, E. Naito, K. Yamagami, J. Ozawa, Collision Avoidance Path Planning for Hospital Robot with Consideration of Disabled Person's Movement Characteristic, in: Proc. of the IEEE Glob. Conf. on Consumer Electronics (GCCE), 2012.

[14] M. Svenstrup, T. Bak, H. J. Andersen, Trajectory Planning for Robots in Dynamic Human Environments, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.

[15] M. Kuderer, H. Kretzschmar, C. Sprunk, W. Burgard, Feature-Based Prediction of Trajectories for Socially Compliant Navigation, in: Proc. of Robotics: Science and Systems (RSS), 2012.

[16] M. Kollmitz, K. Hsiao, J. Gaa, W. Burgard, Time Dependent Planning on a Layered Social Cost Map for Human-Aware Robot Navigation, in: Proc. of the Eur. Conf. on Mobile Robots (ECMR), 2015.

[17] E. Steinfeld, S. Schroeder, M. Bishop, Accessible Buildings for People with Walking and Reaching Limitations, Tech. Rep., U.S. Department of Housing and Urban Development, 1979.

[18] A. Vasquez, M. Kollmitz, A. Eitel, W. Burgard, Deep Detection of People and their Mobility Aids for a Hospital Robot, in: Proc. of the Eur. Conf. on Mobile Robots (ECMR), 2017.

[19] R. Girshick, Fast R-CNN, in: Proc. of the Int. Conf. on Computer Vision (ICCV), 2015.

[20] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, in: Proc. of the Conf. on Neural Information Processing Systems (NIPS), 2015.

[21] A. Petrovskaya, S. Thrun, Model based vehicle detection and tracking for autonomous urban driving, Autonomous Robots 26 (2-3) (2009) 123 – 139.
[22] A. Dewan, T. Caselitz, G. D. Tipaldi, W. Burgard, Motion-based Detection and Tracking in 3D LiDAR Scans, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2016.
[23] H. Cho, Y.-W. Seo, B. V. Kumar, R. R. Rajkumar, A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2014.
[24] T. Darrell, G. Gordon, M. Harville, J. Woodfill, Integrated Person Tracking Using Stereo, Color, and Pattern Detection, Int. Journal of Computer Vision (IJCV) 37 (2) (2000) 175 – 185.
[25] R. Muñoz-Salinas, E. Aguirre, M. García-Silvente, People Detection and Tracking using Stereo Vision and Color, Image and Vision Computing 25 (6) (2007) 995 – 1007.
[26] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, L. H. Matthies, A Fast Stereo-based System for Detecting and Tracking Pedestrians from a Moving Vehicle, Int. Journal of Robotics Research (IJRR) 28 (11-12) (2009) 1466 – 1485.
[27] A. Ess, B. Leibe, K. Schindler, L. Van Gool, Robust Multiperson Tracking from a Mobile Platform, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 31 (10) (2009) 1831 – 1846.
[28] A. Ošep, W. Mehner, M. Mathias, B. Leibe, Combined Image- and World-Space Tracking in Traffic Scenes, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2017.
[29] A. Ošep, W. Mehner, P. Voigtlaender, B. Leibe, Track, then Decide: Category-Agnostic Vision-based Multi-Object Tracking, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2018.
[30] T. Linder, S. Breuers, B. Leibe, K. O. Arras, On Multi-Modal People Tracking from Mobile Platforms in Very Crowded and Dynamic Environments, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2016.
[31] C. Dondrup, N. Bellotto, F. Jovan, M. Hanheide, Real-Time Multisensor People Tracking for Human-Robot Spatial Interaction, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2015.
[32] O. H. Jafari, D. Mitzel, B. Leibe, Real-Time RGB-D based People Detection and Tracking for Mobile Robots and Head-Worn Cameras, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2014.
[33] L. Spinello, K. O. Arras, People Detection in RGB-D Data, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2011.
[34] M. Luber, L. Spinello, K. O. Arras, People Tracking in RGB-D Data With On-line Boosted Target Models, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2011.
[35] H. Zhang, C. Reardon, L. E. Parker, Real-Time Multiple Human Perception with Color-Depth Cameras on a Mobile Robot, IEEE Transactions on Cybernetics 43 (5) (2013) 1429 – 1441.
[36] J. Liu, Y. Liu, G. Zhang, P. Zhu, Y. Q. Chen, Detecting and tracking people in real time with RGB-D camera, Pattern Recognition Letters 53 (2015) 16 – 23.
[37] M. Munaro, E. Menegatti, Fast RGB-D people tracking for service robots, Autonomous Robots 37 (3) (2014) 227 – 242.
[38] O. Mees, A. Eitel, W. Burgard, Choosing Smartly: Adaptive Multimodal Fusion for Object Detection in Changing Environments, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2016.
[39] J. Guerry, B. Le Saux, D. Filliat, "Look At This One" Detection sharing between modality-independent classifiers for robotic discovery of people, in: Proc. of the Eur. Conf. on Mobile Robots (ECMR), 2017.
[40] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard, T. Brox, 3D Human Pose Estimation in RGB-D Images for Robotic Task Learning, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2018.
[41] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, R. Urtasun, Monocular 3D Object Detection for Autonomous Driving, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.
[42] L. Beyer, A. Hermans, B. Leibe, DROW: Real-Time Deep Learning-Based Wheelchair Detection in 2D Range Data, IEEE Robotics and Automation Letters 2 (2) (2017) 585 – 592.
[43] T. Linder, S. Wehner, K. O. Arras, Real-Time Full-Body Human Gender Recognition in (RGB)-D Data, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2015.
[44] G. Sharma, F. Jurie, Learning discriminative spatial representation for image classification, in: Proc. of the Brit. Machine Vision Conf. (BMVC), 2011.
[45] L. Bourdev, S. Maji, J. Malik, Describing People: A Poselet-Based Approach to Attribute Classification, in: Proc. of the Int. Conf. on Computer Vision (ICCV), 2011.
[46] P. Sudowe, H. Spitzer, B. Leibe, Person Attribute Recognition with a Jointly-trained Holistic CNN Model, in: Proc. of the IEEE Int. Conf. on Computer Vision Workshop (ICCVW), 2015.
[47] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, K. He, Detectron - GitHub, URL https://github.com/facebookresearch/detectron, 2018.
[48] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, W. Burgard, Multimodal Deep Learning for Robust RGB-D Object Recognition, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2015.
[49] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, ArXiv e-prints: 1706.02677, 2017.
[50] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.
[51] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the Devil in the Details: Delving Deep into Convolutional Nets, in: Proc. of the Brit. Machine Vision Conf. (BMVC), 2014.
[52] R. R. Labbe, Kalman and Bayesian Filters in Python - GitHub, URL https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2015.
[53] E. Marder-Eppstein, Navigation - ROS Wiki, URL http://wiki.ros.org/navigation, 2009.
[54] M. Kollmitz, omni_path_follower - GitHub, URL https://github.com/marinaKollmitz/omni_path_follower, 2017.

Appendix A. 3D Centroid Labeling

We label the 3D centroid depths from the depth images of the dataset after annotating the 2D bounding boxes. For the centroid depth labeling, we only consider the DepthJet labels, because they definitely include depth information describing the person. We convert each depth image to a point cloud and remove the ground plane. Then we extract the part of the point cloud which belongs to the 2D bounding box. Finally, we cluster the extracted part and calculate the 3D centroid of each cluster as the mean of all points belonging to the cluster.

Now we need to decide which cluster best represents the person. For both test sets, an experimenter selected the most representative clusters by hand. For the training split, we used a simple heuristic to choose the best cluster from the following criteria:

- cluster size
- distance between cluster center and bounding box center in the image
- depth of cluster centroid.

The cluster size and distance ensure that the heuristic selects a dominant cluster, the cluster depth criterion favors clusters closer to the camera to avoid choosing parts of the background. For all clusters in a bounding box we calculate a score repre-

senting each criterion:

$$s_{\text{size},i} = \frac{n_{\text{cl},i}}{\sum_i n_{\text{cl},i}} \tag{A.1}$$

$$s_{xy,i} = \frac{\sqrt{(x_{\text{cl},i} - x_{\text{im}})^2 + (y_{\text{cl},i} - y_{\text{im}})^2}}{\sqrt{w_{\text{box}}^2 + h_{\text{box}}^2}} \tag{A.2}$$

$$s_{z,i} = 1 - \frac{z_{\text{cl},i}}{z_{\text{max}}} \tag{A.3}$$

The number of points in each cluster i is n_{cl}, $(x_{\text{cl}}, y_{\text{cl}})$ are the image coordinates of the cluster center and w_{box} and h_{box} are the bounding box width and height. The maximum range of the Kinect is denoted by z_{max} and z_{cl} is the centroid depth of the cluster. All scores range between 0 and 1, and we choose the ground truth centroid depth of the 2D bounding box as z_{cl} of the cluster with the maximum overall score

$$s_i = s_{\text{size,i}} \cdot s_{xy,i} \cdot s_{z,i}. \tag{A.4}$$

We evaluated our heuristic on 500 manually labeled bounding boxes where an experimenter selected the most representative depth clusters. The heuristic selects the same cluster in 97.8 % of the cases. For the remaining 2.2 %, the mean absolute depth error between the clusters was 1.78 m. This means that our heuristic is very reliable, but outliers are possible. After labeling the 3D centroid depths for the DepthJet dataset, we transfer them to the RGB bounding boxes. For labels which are present in RGB but not in DepthJet, e.g. due to the limited depth range of the camera, we deactivate the depth label by setting $p^*_{z,\text{cam}} = 0$, see Eq.3.