22^{nd} *Computer Vision Winter Workshop*
Nicole M. Artner, Ines Janusch, Walter G. Kropatsch (eds.)
Retz, Austria, February 6–8, 2017

Pedestrian Detection in RGB-D Images from an Elevated Viewpoint

Christian Ertler Horst Possegger Michael Opitz Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology, Austria
christian.ertler@student.tugraz.at, {possegger,michael.opitz,bischof}@icg.tugraz.at

Abstract. *We propose an extension to the state-of-the-art Faster R-CNN detection model for multi-modal pedestrian detection from RGB-D images. The proposed architectures address this problem by fusing convolutional neural network (CNN) representations. We elaborate two architectures, which primarily differ in the position of the fusion inside the model, and further compare several static and parametrized fusion layers. Moreover, we show how recent advances in the area of non-maximum suppression (NMS) can improve the detection results of our models and make them more robust in applications with varying pedestrian densities. Our models are trained and evaluated on a custom dataset comprising images of crosswalk scenes taken from an elevated viewpoint. This viewpoint results in uncommon and highly variable poses of pedestrians, demanding powerful detection models.*

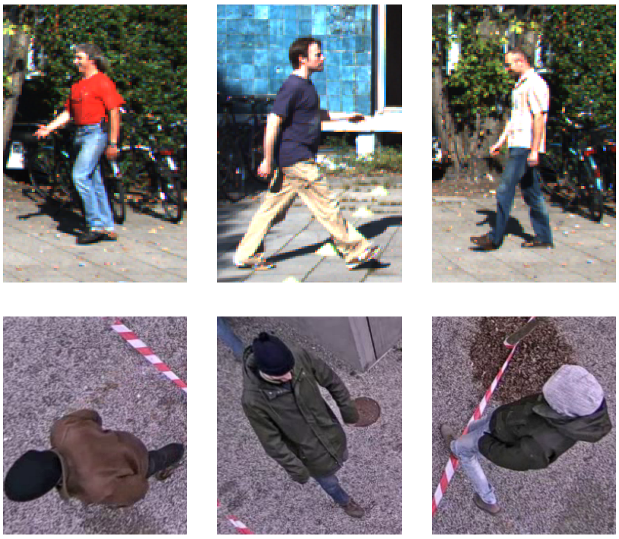


Figure 1: Example images of pedestrians from a classical surveillance viewpoint of the KITTI [8] (top row) and our elevated viewpoint (bottom row).

1. Introduction

Pedestrian detection, as a specialized case of object detection, is a crucial pre-processing step for many applications in computer vision, including visual surveillance, autonomous vehicles, automated video analysis, *etc*. The major research interest in these areas are classical surveillance scenarios, *i.e*. pedestrians are captured from a slightly elevated viewpoint which results in a long-range field-of-view. Thus, classical pedestrian detectors are specifically designed for side-view and front-view pedestrians in an upright pose and are trained on datasets comprising such images [2, 8].

In this work, we address the problem of pedestrian detection from an elevated viewpoint in RGB-D images. In particular, we rely on stereo cameras mounted on top of traffic lights filming downwards to capture the pedestrian wait area below. This overhead viewpoint introduces significant differences in human appearance and pose compared to classical surveillance viewpoints, where the effects of perspective are negligible. Figure 1 illustrates some of these differences. In contrast to traditional long-range field-of-view scenarios, the pose and appearance of a person captured from our high elevation viewpoints heavily depend on the relative position to the camera. For example, the only visible parts of a person standing just beneath the camera are the head and shoulders. People at the border of the field-of-view, on the other hand, appear elongated and rotated about their vertical axis. Due to these transformations, it is not possible to make common assumptions about the location of certain parts inside a person's bounding box (*e.g*. the head is not strictly at the top and the legs are not always at the bottom). Another side effect is the increased variation of the bounding box aspect ratios. The typical assumption of a nearly fixed, upright aspect ratio [2, 6] cannot be made in our overhead scenario, since bounding box shapes range from rectangular (both upright and horizontal) to square.

Since traditional single-template based approaches for pedestrian detection (such as histogram of oriented gradients (HoG) based detectors, *e.g.* [2]) cannot cope with large pose variations, we utilize state-of-the-art CNN based detectors. These detectors are capable of learning strong representations, and thus are capable of detecting objects with large pose variations [9, 22].

Our main contribution is to adapt the Faster R-CNN [22] model for RGB-D images and fine-tune it on custom datasets recorded from our overhead viewpoint. We propose different fusion architectures and fusion layers to fuse the RGB and depth modalities inside the model. We perform a detailed evaluation of our architectures to identify the benefits of various fusion approaches. Our experiments show that integrating depth clearly improves detection performance compared to RGB images only. Further, we integrate an additional CNN for the task of NMS based on work of Hosang *et al.* [13], and show that it helps to make the detector more robust to scenes of varying pedestrian densities.

The remainder of this paper is structured as follows: Section 2 reviews related work, Section 3 describes our detection pipeline, Section 4 discusses our evaluation results. Finally, Section 5 concludes our work.

2. Related Work

Traditionally, object detectors are categorized into holistic HoG based detectors (*e.g.* [2]), deformable part model (DPM) based detectors (*e.g.* [6]), boosting based detectors (*e.g.* [3]), and bag-of-words based detectors (*e.g.* [25]). HoG and DPM based detectors operate in a sliding-window manner over the whole image, whereas bag-of-words based approaches rely on pre-computed object proposals.

With the success of deep learning methods in recent years [16], detectors based on CNNs have been shown to outperform those traditional approaches. A seminal work in this area is R-CNN proposed by Girshick *et al.* [10], which makes use of a classification network by applying it to a set of pre-computed object proposals and feeding the extracted features to class-specific SVMs in order to classify each region independently. Its successors Fast R-CNN [9] and Faster R-CNN [22] further improve over it by sharing computation on single images with region of interest (RoI) pooling and integrating the region proposal generation into the network, respectively. Other CNN based detectors (*e.g.* [18, 21]) operate in a fully-convolutional manner and do not rely on region proposals to generate detection hypotheses.

The particular setup of pedestrian detection from an elevated viewpoint has received only little research interest. Ahmed and Carter [1] propose to project the image of a person to the center of the camera based on the known position before computing HoG features. However, their approach cannot be efficiently adapted to state-of-the-art CNN detectors since their computational efficiency results from sharing computations for all detections within a single image. Other approaches include [19] focusing on efficient HoG computation, and [20] proposing a feature descriptor for head detection in depth-only images. All of these are restricted to a top-view setup where the ground plane is parallel to the camera sensor, which is not the case for our approach.

Some work has been published concerning deep CNNs operating on depth data. For example, Gupta *et al.* [11] show the possibility of fine-tuning an ImageNet network with depth images. They propose the HHA encoding, which represents each depth pixel by three features, namely horizontal disparity, height above ground, and the angle of gravity. Eitel *et al.* [4] achieve similar results with a simpler encoding, where they apply a static colormap to the depth data, *i.e.* simply using a pseudocolor image derived from the depth values. Both reuse freely available RGB networks and fine-tune them with their encoded depth data. Closely related to our depth fusion approach, Liu *et al.* [17] also formulate pedestrian detection as a CNN fusion problem. However, they also constrain their approach to standard surveillance viewpoints as opposed to our highly elevated viewpoint. Additionally, they exploit multispectral data (*i.e.* thermal imagery) in contrast to our depth data.

Most modern object detectors (*e.g.* [6, 9, 10, 22]) rely on a greedy algorithm for NMS. This post-processing step is commonly not involved in the training process and each detection is treated independently. Wan *et al.* [26] propose a CNN training loss which is aware of NMS. However, it still relies on the same greedy NMS algorithm with fixed parameters. Hosang *et al.* [13] propose a CNN architecture capable of learning and performing NMS by itself as replacement for the traditional greedy algorithm. We adopt their NMS approach, extend it for handling sparse detections, and train a similar model for our pedestrian detector.

3. Detection Pipeline

In the following, we briefly summarize the baseline CNN model (Section 3.1) and discuss the proposed extensions in more detail (Section 3.2). Section 3.3 then explains the NMS CNN model.

3.1. Baseline Faster R-CNN Model

Due to the success of deep learning, many CNN based object detectors suitable for our experiments have been published. At the time of writing, the model offering the best trade-off between detection and runtime performance is Faster R-CNN [22]. It has a two-stage architecture which allows to adapt a classification CNN for the task of object detection. Both stages share computation by operating on the same features extracted from the convolutional part of the classification network. The first stage consists of a region proposal network (RPN) that generates a set of proposals based on these convolutional features. In the second stage, a RoI Pooling layer [9] pools the features inside the various proposal boxes to fixed-size (*i.e.* the size of the underlying network's last pooling layer's output) feature maps. These features are then passed through the fully-connected (FC) classification layers to perform classification and bounding box refinement on each subwindow.

We fine-tune the freely available Faster R-CNN model based on the Zeiler & Fergus (ZF) network [27]. Due to overfitting concerns resulting from our small datasets, we omit the last hidden layer (FC7). Instead, we directly connect the first hidden FC layer (FC6) to the output layers.

3.2. Faster R-CNN with Depth Fusion

A straightforward approach to incorporate the depth modality for detection is to adjust the network dimensions such that it can deal with RGB-D images. However, this approach demands a lot of training data and further leads to a significant amount of computational effort. Instead, we follow the original idea of Faster R-CNN and fine-tune existing RGB models. To this end, we explore different approaches to fuse two modalities in a Faster R-CNN model.

Depth Encoding

Fine-tuning an RGB CNN with depth data poses the problem of having a different number of input channels. The networks expect three channels, whereas depth is a single channel. We follow Eitel *et al.* [4] and use a colormap to spread the depth information across the three required channels. However, instead of directly using the depth w.r.t. to the camera, we compute height above ground (HaG) values derived from the dense depth maps. To this end, we automatically estimate the ground plane from the recorded depth information and compute the point-to-plane distance for each point in the depth map. The advantage of HaG over raw depth data lies in lower data variation, since the height of a person does not depend on the position relative to the camera in contrast to horizontal disparity or depth. The resulting HaG values are finally colored using the parula colormap[1]. Figure 3 shows an example of such an input pair (RGB and HaG).

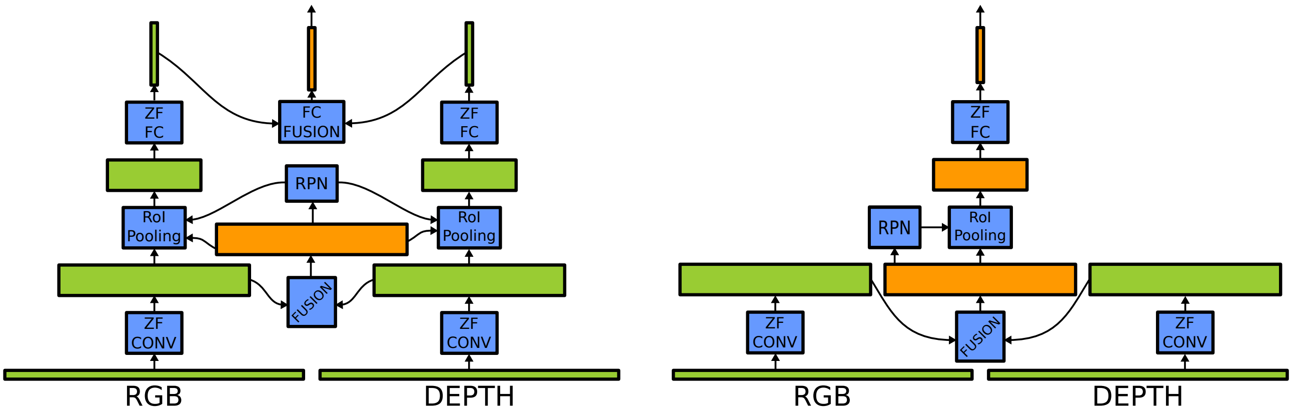
Our preliminary experiments showed that this encoding outperforms the HHA encoding proposed by Gupta *et al.* [11] for our application scenario. Thus, for the following experiments and evaluations we stick to the proposed HaG encoding.

Fusion Architectures

We investigate two architectures for the problem of fusing two Faster R-CNN models operating on different modalities, namely (1) late fusion and (2) mid-layer fusion. A visualization of both fusion architectures can be found in Figure 2.

The two approaches differ primarily in the location of the fusion. As the name suggests, late fusion pulls the modality fusion to the latest possible location in the network (*i.e.* the last hidden layer FC6), resulting in two nearly independent network streams for RGB and HaG data. Note that they are not entirely independent since we also fuse the mid-layer features for the RPN in order to get unified region proposals for both streams. In the mid-layer fusion architecture, on the other hand, fusion occurs solely after the last convolutional layer. These layers are known to extract spatially related semantic representations as opposed to earlier layers or FC layers, which extract low-level features or spatially unrelated high-level features, respectively [27]. This earlier fusion approach significantly reduces the number

[1] MATLAB's default colormap since version R2014b.



(a) Late fusion.

(b) Mid-layer fusion.

Figure 2: Illustration of our two approaches to fuse RGB and depth data in Faster R-CNN: (a) late fusion and (b) mid-layer fusion. Blue boxes correspond to CNN layers (or groups of layers), green boxes to single-modality feature maps, and orange boxes to fused feature maps. The arrows indicate the forward direction of the network.

of model parameters (*i.e.* from 117 M to 45 M) and replaces the two independent network streams by a single one operating on the fused representations.

Fusion Layers

Given two feature maps $\mathbf{M}^{\text{rgb}}, \mathbf{M}^{\text{hag}} \in \mathcal{R}^{H \times W \times C}$, the output of a fusion layer must be another feature map $\mathbf{M}^{\text{fused}} \in \mathcal{R}^{H \times W \times C}$, where H, W, C denote the height, width, and number of channels of the feature maps, respectively. The constraint of producing a feature map with the same dimensionality is necessary since we cannot change the underlying CNN model, because we want to exploit the benefits of transfer learning and preserve features learned from large scale datasets, *e.g.* ImageNet.

We consider six different fusion layers, namely (1) average, (2) sum, (3) max, (4) FC, (5) conv, and (6) inception fusion. The first set of fusion layers performs element-wise, parameterless operations:

(1) takes the average value at every spatial location and channel, *i.e.* $\mathbf{M}^{\text{fused}}_{h,w,c} = \frac{\mathbf{M}^{\text{rgb}}_{h,w,c}}{2} + \frac{\mathbf{M}^{\text{hag}}_{h,w,c}}{2}$.

(2) computes the sum of both features at every spatial location and channel, *i.e.* $\mathbf{M}^{\text{fused}}_{h,w,c} = \mathbf{M}^{\text{rgb}}_{h,w,c} + \mathbf{M}^{\text{hag}}_{h,w,c}$.

(3) takes the maximum value at every spatial location and channel, *i.e.* $\mathbf{M}^{\text{fused}}_{h,w,c} = \max\left(\mathbf{M}^{\text{rgb}}_{h,w,c}, \mathbf{M}^{\text{hag}}_{h,w,c}\right)$.

The second set consists of parametrized operations which need to be optimized during training. We first concatenate the feature maps along the channel dimension to get $\mathbf{M}^{\text{concat}}_{h,w} = \mathbf{M}^{\text{rgb}}_{h,w} \parallel \mathbf{M}^{\text{hag}}_{h,w} \in \mathcal{R}^{H \times W \times 2C}$. The following layers need to perform dimensionality reduction in order to fit $\mathbf{M}^{\text{fused}}$ again:

(4) applies an FC layer to $\mathbf{M}^{\text{concat}}$. Note that we can only use this layer for late fusion albeit not for mid-layer fusion. The inner product performed by this layer treats the input as vector, and therefore destroys the spatial relationship of the features as required by the RPN and RoI pooling layer.

(5) consists of a 1×1 convolutional layer with C filters operating on $\mathbf{M}^{\text{concat}}$.

(6) is inspired by GoogLeNet's [24] inception module. It consists of parallel convolutional layers operating on $\mathbf{M}^{\text{concat}}$ with 1×1, 3×3 and 5×5 convolutions, and another parallel layer with 3×3 max-pooling. The concatenated output features of these layers give the fused feature map $\mathbf{M}^{\text{fused}}_{h,w} = \mathbf{M}^{\text{conv1}}_{h,w} \parallel \mathbf{M}^{\text{conv2}}_{h,w} \parallel \mathbf{M}^{\text{conv3}}_{h,w} \parallel \mathbf{M}^{\text{pool}}_{h,w} \in \mathcal{R}^{H \times W \times C}$.

All fusion layers are followed by rectified linear unit (ReLU) non-linearities.

The spatial relationship between two features is clearly given by the location inside the input feature maps for all fusion layers. This is not necessarily true for the channel dimension. Element-wise fusion layers treat every channel independently, so subsequent

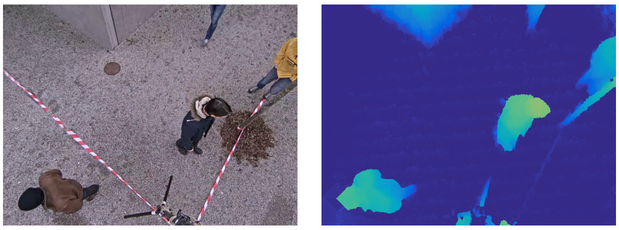


Figure 3: Left: An RGB image from our dataset. Right: The corresponding colored HaG image. The colormap ranges from blue (low, *i.e.* close to the ground plane) to yellow (high).

layers must learn how to extract the optimal relationships. The second set of layers (*i.e.* the parametrized fusion layers), however, are able to directly learn the optimal channel relationships, since they process the entire tensor consisting of all channels at a particular spatial location.

3.3. Non-Maximum Suppression

The original Faster R-CNN [22] model relies on a post-processing step for NMS. They use a greedy algorithm to suppress predicted boxes based on a constant intersection over union (IoU) overlap threshold. This is necessary since the detection model treats every prediction independently from each other, resulting in multiple detections of the same object. However, choosing a constant threshold involves a trade-off between precision and recall. Additionally, the model is heavily tuned to the validation set and thus, the detector may perform worse if the density of objects differs significantly between training and runtime.

To overcome the drawbacks of this static algorithm, we follow Hosang *et al.* [13] and train an additional CNN to replace the greedy NMS algorithm. Their so-called Tnet model does not need additional ground truth labeling. It can be trained using the output of our detection model (*i.e.* the boxes and corresponding detection confidence scores) and the ground truth labels of our dataset.

Architecture

Tnet is a fully-convolutional network operating on a two-dimensional grid, which discretizes the detection boxes' coordinate system. Each box is mapped to the grid based on the coordinates of its center. The input of the network is two-fold, namely a score map and an IoU map. The score map encodes the positions and confidence scores of the detections after applying traditional NMS with several thresholds. Second, the IoU map encodes the overlaps of boxes in a pre-defined neighborhood of size $N \times N$, respectively. The network extracts correlated features per cell by performing $N \times N$ convolutions on the score map and 1×1 convolutions on the IoU map. The concatenated features are then further processed by three consecutive 1×1 convolutional layers, which learn to rescore the input detections in order to produce the final output score map. We refer the interested reader to [13] for a detailed description of the architecture and training process.

Implementation Details

We implement Tnet using Caffe [14] following the description in [13] with a few adaptions to fit our needs. We reduce the number of filters from 512 to 128 per layer and increase the grid downsampling factor from 4 to 6. Both modifications do not harm the performance in our experiments, but result in a decent speedup. We use a broad range $[1.0, 0.8, 0.7, \ldots, 0.1]$ of NMS thresholds for the score map computation to provide as much information as possible to the network, and use $N = 11$ for the neighborhood.

Further, we tweak the loss weighting since — in contrast to Hosang *et al.* — we only have a sparse set of detections and thus, most input values are zero. To this end, we ignore the loss of cells without any detection in the $N \times N$ neighborhood and normalize the weights by the number of enabled cells. This leads to a much more stable convergence in our experiments.

During training, we perform data augmentation by randomly flipping the coordinate system both vertically and horizontally. Additionally, we move each box in x and y-direction by a random number of pixels drawn from a normal distribution with zero mean and a standard deviation of 6.

4. Evaluation and Discussion

Our experiments regarding RGB and HaG fusion are based on the publicly available Python code of Faster R-CNN [22], adapted for multimodal input according to the architectures described in Section 3.2. We initialize all layers, which are present in the original model, with the publicly available pre-trained weights of Faster R-CNN. Note that also the layers belonging to the HaG stream are initialized in this way. All weights of the newly introduced fusion layers are initialized using Xavier initialization [14]. Following [22], we train the models with stochastic gradient descent (SGD) using the approximate joint training method[2]. We train the model for 50k iterations with a learning rate of 10^{-4}. After 20k iterations, the learning rate is lowered by a factor of 10. We train Tnet following the training process in [13] with the Adam solver [15] and MSRA initialization [12] using a learning rate of 10^{-4}. Additionally, we perform gradient clipping such that the L_2 norm of all gradients does not exceed a value of 1000. Momentum is set to 0.9 and weight decay to $5 \cdot 10^{-4}$ for all experiments (*i.e.* Faster R-CNN and Tnet).

4.1. Dataset

Our custom dataset consists of images from video sequences recorded on two different locations using a stereo setup mounted on a telescopic rod or a traffic light pole, respectively, filming downwards to the scene. We use the discrete-continuous approach from Shekhovtsov *et al.* [23] to compute the stereo disparities and calibrate the camera setup using the toolbox of Ferstl *et al.* [7]. We train our models using 447 samples from an uncontrolled recording on a public site, which includes many different people. The validation set consists of 82 samples from the same recording. We ensure that the validation set does not contain samples of people present in the training set. For testing, we use 321 samples from a recording in a controlled environment on a separate location, comprising a different background than the training set. This split is chosen to test the generalization capabilities of the models and to make sure that the models do not overfit to the static background in the training set. Example images of the validation and test set can be found in Figure 5.

4.2. Performance Metrics

We stick to the evaluation metrics of the Pascal VOC challenge [5]. More specifically, we compute the average precision (AP) as the mean interpolated precision at eleven equally spaced recall levels as defined in [5]. A detection will be considered correct (true positive), if its bounding box overlaps more than $50\,\%$ with a ground truth bounding box (*i.e.* IoU > 0.5). Double detections are considered incorrect (false positive).

Model	**AP**	
	mid-layer	**late**
RGB-only	**81.95** (0.35)	
HaG-only	52.05 (3.55)	
sum fusion	88.60 (1.00)	87.55 (0.65)
average fusion	87.00 (0.00)	**87.70** (0.90)
max fusion	**89.89** (0.20)	87.65 (0.75)
conv fusion	86.35 (0.55)	85.60 (1.10)
inception fusion	88.85 (0.85)	—

Table 1: Detection performance of our several fusion architectures and layers compared to the baseline RGB-only model. The AP values are averaged over two trainings with random initialization. The standard deviation is given in parenthesis.

4.3. Experiments with Depth Fusion

To evaluate the influence of the additional depth modality, we compare our several fusion approaches from Section 3.2 to the baseline RGB-only model. The results in Table 1 show that all fusion combinations (*i.e.* mid-layer and late fusion in combination with our several fusion layers) outperform the baseline by ≈ 3.7 to 8.0 AP points. This is a clear indicator that the additional depth modality helps to improve the detection results of a Faster R-CNN model. Further, the values reveal that mid-layer fusion is superior to late fusion (except for the average fusion model). This indicates that the mid-layer representations learned by the network are more eligible for modality fusion than representations learned by later layers. We hypothesize that the mixture of semantic meanings and spatial and visual details in these mid-layer representations provides more complementary features than the high-level representations learned by the last hidden FC layers. Thus, the network is able to gain more information from the combination of both modalities.

4.4. Experiments with NMS

To evaluate the performance of our learned Tnet model from Section 3.3, we compare it to the original greedy NMS algorithm used in Faster R-CNN and also in the experiments from Section 4.3. We train Tnet with the raw detection boxes (*i.e.* before NMS) obtained from our mid-layer fusion model with max

[2] Published as supplemental material to [22]. Our preliminary experiments validated the similar performance and reduced complexity compared to the original alternating training method.

Model	**AP**		
	all	**overlap**	**no overlap**
Tnet	**90.10**	**87.00**	**95.90**
NMS$_{0.9}$	41.20	37.30	49.40
NMS$_{0.8}$	67.80	61.80	76.40
NMS$_{0.7}$	85.60	78.10	93.40
NMS$_{0.6}$	**89.70**	**82.30**	95.40
NMS$_{0.5}$	88.30	81.00	**95.90**
NMS$_{0.4}$	87.10	79.30	95.30
NMS$_{0.3}$	86.30	77.90	95.20
NMS$_{0.2}$	83.30	74.00	95.00
NMS$_{0.1}$	78.20	65.40	94.30

Table 2: Comparison of greedy NMS with several overlap thresholds and our Tnet model. AP values are shown for the entire test set (first column), samples with overlapping ground truth boxes (second column), and samples without overlapping ground truth boxes.

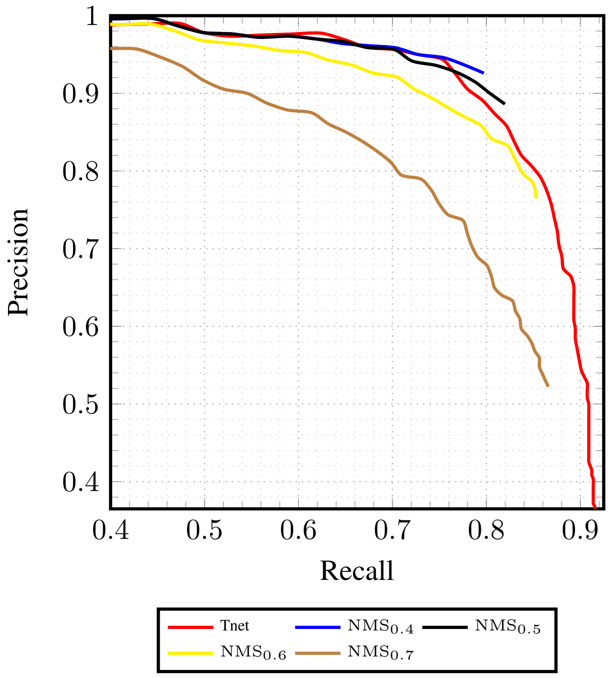


Figure 4: Evaluation of Tnet and the best performing greedy NMS thresholds.

fusion (*i.e.* the model yielding the best performance on our dataset). Of course, the same model is used during testing to obtain the raw detections for Tnet and greedy NMS.

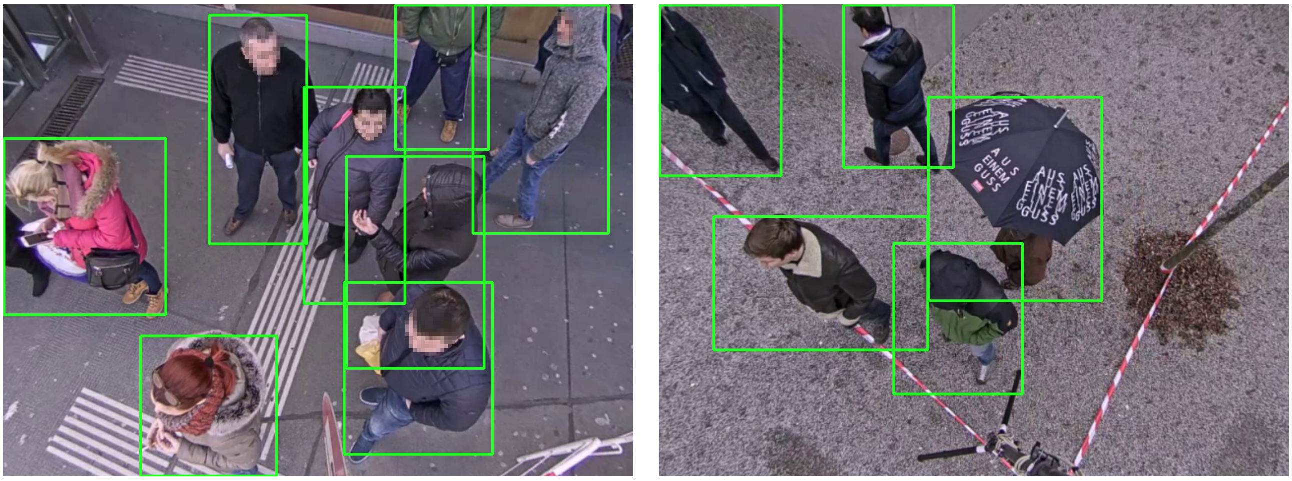
Table 2 summarizes the performance results of Tnet and several greedy NMS thresholds in terms of AP. To emphasize the contribution of Tnet, we split the test set into samples with (1) at least two overlapping ground truth annotations and (2) without any overlapping ground truth annotations and evaluate both splits independently. The overall performance boost of Tnet measured over all samples is negligible. However, this comes mainly from the fact that the non-overlapping samples dominate the test set and Tnet cannot outperform the best NMS threshold in this case. The more interesting and challenging part is the set of overlapping samples, which poses a much harder problem for NMS. Here, Tnet improves AP by 4.7 points compared to the best NMS threshold. These values show that Tnet — in contrast to greedy NMS — is able to adapt to scenes with varying pedestrian densities. Another advantage of our learned model is that it eliminates the need to choose a constant overlap threshold for production. Choosing such a threshold always involves a trade-off between precision and recall, as shown in Figure 4. We see that low greedy NMS thresholds of 0.4 and 0.5 provide high precision, but low recall. Higher thresholds, however, improve recall at the cost of lower precision. Although Tnet cannot strictly outperform all NMS thresholds, it provides a smooth and high enough precision over the entire recall range. Figure 6 shows some results of both greedy NMS and Tnet on our datasets.

4.5. Runtime Performance

The following runtime measurements are performed on a workstation consisting of a 3.20 GHz Intel Core i5 CPU and an NVIDIA GTX 970 with 4 GB memory. Our RGB-only model takes about 67 ms for an image of 800 × 600 pixels. The mid-layer fusion architecture is just slightly slower with 87 ms, and the late fusion takes 119 ms. The original NMS procedure costs about 12 ms. Tnet approximately doubles this runtime with about 28 ms.

5. Conclusion

In this work, we addressed the task of pedestrian detection from an elevated viewpoint in RGB-D images. We proposed two extensions to the state-of-the-art Faster R-CNN detection model by fusing RGB and depth representations at different layers of the model. Several ways of fusing those representations in terms of different fusion layers were presented. Our experiments showed that mid-layer representations are most eligible for fusion. Further,



(a) Validation set.

(b) Test set.

Figure 5: Examples of the validation set and test set. The boxes correspond to the detections of the best performing model (*i.e.* mid-layer fusion architecture with max fusion).

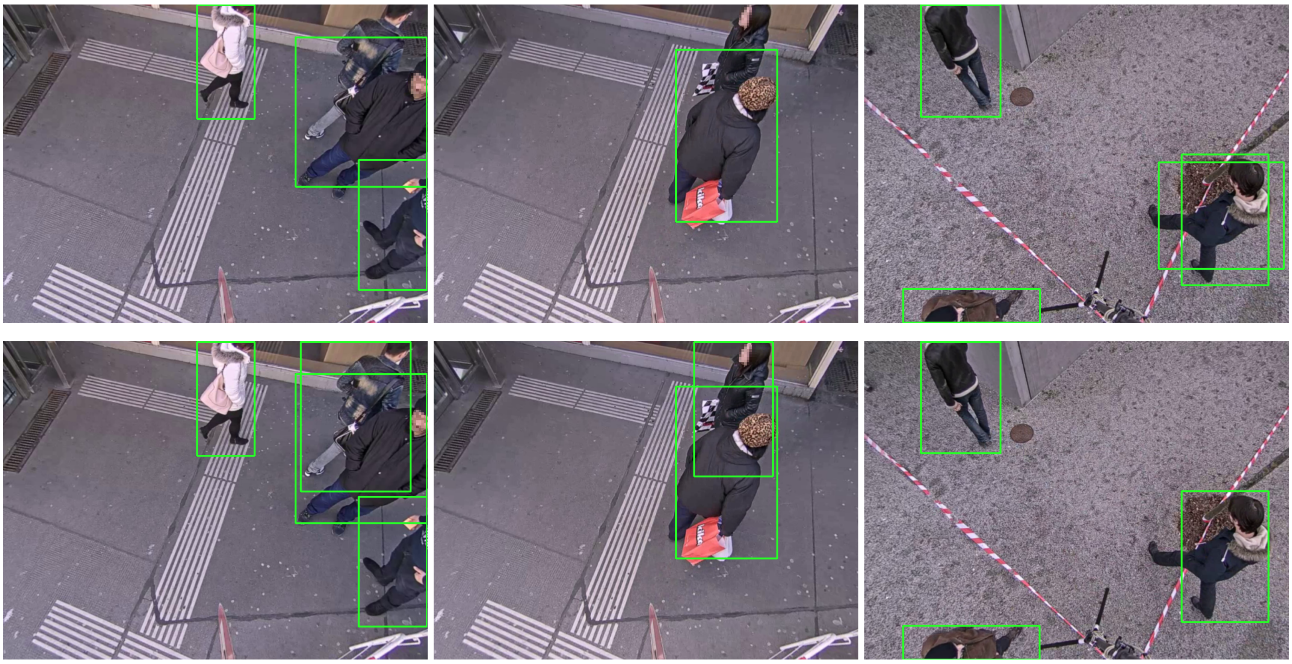


Figure 6: Qualitative comparison of the best performing greedy NMS threshold (first row) and Tnet (second row). Both models operate on the same precision. Selecting a fixed greedy NMS threshold leads to missed detections (first two columns) and multiple detections for a single person (rightmost column), whereas our modified Tnet is able to cope with such scenarios.

we extend Faster R-CNN with a recent learnable approach to NMS, which improves performance especially in crowded situations and applications with varying pedestrian density.

References

[1] I. Ahmed and J. N. Carter. A robust person detector for overhead views. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2012. 2

[2] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 1, 2

[3] P. Dollár, Z. Tu, P. Perona, and S. J. Belongie. Integral Channel Features. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009. 2

[4] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal Deep Learning for Robust RGB-D Object Recognition. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2015. 2, 3

[5] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010. 6

[6] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010. 2

[7] D. Ferstl, C. Reinbacher, G. Riegler, M. Rüther, and H. Bischof. Learning Depth Calibration of Time-of-Flight Cameras. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2015. 6

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013. 1

[9] R. Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 2, 3

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[11] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 2, 3

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 6

[13] J. Hosang, R. Benenson, and B. Schiele. A Convnet for Non-Maximum Suppression. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2016. 2, 5, 6

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014. 5, 6

[15] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 6

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2012. 2

[17] J. Liu, S. Zhang, S. Wang, and D. N. Metaxas. Multispectral Deep Neural Networks for Pedestrian Detection. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 2

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2

[19] Y. Pang, Y. Yuan, X. Li, and J. Pan. Efficient HOG human detection. *Signal Processing*, 91(4):773–781, 2011. 2

[20] M. Rauter. Reliable Human Detection and Tracking in Top-View Depth Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2015. 2, 3, 5, 6

[23] A. Shekhovtsov, C. Reinbacher, G. Graber, and T. Pock. Solving Dense Image Matching in Real-Time using Discrete-Continuous Optimization. In *Proceedings of the Computer Vision Winter Workshop (CVWW)*, 2016. 6

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4

[25] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[26] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a Convolutional Network, Deformable Parts Model and non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[27] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 3