Handling Occlusions with Franken-classifiers

Markus Mathias [1] Rodrigo Benenson [2] Radu Timofte [1,3] Luc Van Gool [1,3]

[1] ESAT-PSI/VISICS, iMinds
KU Leuven, Belgium

[2] MPI Informatics
Saarbrücken, Germany

[3] D-ITET/CVL
ETH Zürich, Switzerland

Abstract

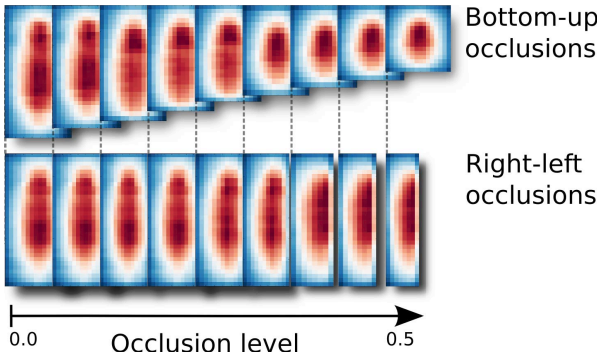
Detecting partially occluded pedestrians is challenging. A common practice to maximize detection quality is to train a set of occlusion-specific classifiers, each for a certain amount and type of occlusion. Since training classifiers is expensive, only a handful are typically trained. We show that by using many occlusion-specific classifiers, we outperform previous approaches on three pedestrian datasets; INRIA, ETH, and Caltech USA. We present a new approach to train such classifiers. By reusing computations among different training stages, 16 occlusion-specific classifiers can be trained at only one tenth the cost of one full training. We show that also test time cost grows sub-linearly.

1. Introduction

The reliable detection of pedestrians is important for applications like surveillance, autonomous robotic navigation, or automotive safety. While the detection quality has constantly improved over recent years, state-of-the-art methods struggle to detect pedestrians that are far away (small in the image), in unusual poses, or occluded [11]. Occlusion[1] is legion. In street scenes about 70 % of all pedestrians appear occluded in at least one frame [7]. Yet, the best performing pedestrian detectors do not handle occlusions explicitly.

A common practice to maximize the detection of occluded objects is to train a set of occlusion-specific classifiers, one classifier for each type (e.g. occlusion from the left) and for each level of occlusion. Since training is costly, only a limited number (3~5) of such classifiers tend to be trained. As the datasets and number of object classes grow, the need for efficient training is rising.

We introduce the idea of spatially biasing feature selection during the training of boosted classifiers. Starting from one biased classifier trained for full-body detection, we reuse training time operations to efficiently build a set of occlusion specific classifiers. The gain in computation time (one order of magnitude) enables to train classifiers for all amounts of occlusions exhaustively, at a fraction of the cost for training a standard detector. Our occlusion classifiers reach 97 % of the performance of a brute-force approach, while requiring only 8 % of the training time. At test time, feature sharing among the occlusion-specific classifiers yields a sub-linear growth in cost. Using our exhaustive set of classifiers provides better performance than using only a sparse set of individually trained occlusion-specific classifiers. We report top performance on the INRIA, ETH and Caltech pedestrian datasets.



(a) Training many occlusion specific classifiers is costly

(b) Result of our combined occlusion-specific classifiers

Figure 1: Motivation: to handle frequently occurring occlusions, we train many occlusion specific classifiers.

[1] Here, we do not distinguish between truncations and occlusions.

1.1. Related work

Object and pedestrian detection in particular have received significant attention over the last years. Despite this, relatively few authors have addressed the issue of occlusion handling. Overall, two common approaches exist:

Training multiple classifiers Statistics on pedestrian occlusion show that a few occlusion types (from the bottom, right, and left) cover more than $95\,\%$ of cases [7]. Thus, it has been proposed to train a small set of classifiers, each one for a specific occlusion [19]. At test time, occlusions are detected (e.g. using a depth map), and their appropriate classifiers are used. This provides the best detection quality for each case, but costs more training. Therefore, in practice only a few detectors are trained. We address this issue.

Occlusion as latent variable An arguably more principled approach is to model occlusion as a latent variable, to be estimated at test time.

In [3] structured regression is used to estimate object bounding boxes at test time, thus handling some occlusions. This model however does not exploit spatial information (bag-of-words models), hence is not competitive.

Another approach is to divide template models into blocks or parts and infer visibility of each block at test time. Occlusion inference is done using depth information [8] or by optimizing the observation likelihood [20, 17, 15, 9, 10].

When marking blocks as occluded these models loose discriminative power since less information is extracted, the usable part of the feature vector shrinks. In contrast, when training a classifier for each occlusion type and level the feature extraction focuses on the visible area, thus enabling improved detection. We expect that, independent of object class, a detector trained for a specific occlusion will surpass "cutting down" a detector that assumes full visibility (see figure 2a).

More recently it was proposed to train specific detectors for pairs of occluding and occluded objects [13]. Although this approach provides good results for pairs of pedestrians, it is unclear how it would scale up for different pairs of classes. These approaches already count training and testing cost in days and hours, while we are in the range of hours and minutes. In this paper we target the general case where the occluded object is independent of the occluding one (wall, image border, etc.).

Hoiem et al. [11] underlined the importance of handling occlusions and the difficulty of properly evaluating it. We work around the evaluation issues in sections 5 and 6.

1.2. Contributions

Our contributions are:

1. We present, for the first time, experiments quantifying the performance of the Integral Channel Features detector (`ChnFtrs`) [6] in the presence of various amounts of occlusion (§5).

2. We propose an effective method to train an exhaustive set of occlusion-specific detectors, at a fraction ($\sim$ 10%) of the cost of a brute-force approach that would train each classifier from scratch (§4.3).

3. We show that at test time, when using the trained set of occlusion-specific detectors, the computation cost grows only sub-linearly ($\sim 20\%$ of the brute-force approach in our setup) (§6.2).

4. When using our exhaustive set of occlusion-specific detectors (named "Franken-classifiers"), we improve the state of the art of pedestrian detection on three challenging datasets (§6).

To the best of our knowledge this is the first work addressing the issue of computation cost of training many occlusion-specific classifiers. In this work we focus on handling occlusions from the bottom, left and right. However, the approach is generic and could be used for any type of occlusions (such as "bottom-left corner", or "occluded torso").

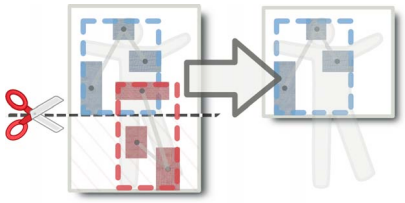
First we briefly describe the `ChnFtrs` detector (section 2), and its poor performance under occlusion (section 3). We then improve it by adding a spatial bias (section 4.1). In sections 4.2 and 4.3 we leverage this idea to efficiently build a set of occlusion-specific classifiers. We evaluate the proposed methods, both when assuming that occlusion boundaries are known (section 5), and when they are unknown (section 6). We conclude in section 7.

2. Integral channel features classifier

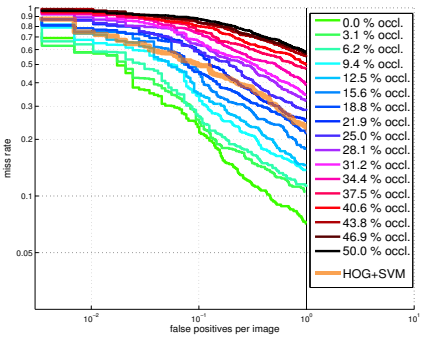
As a base classifier we use our implementation of the Integral Channel Features (`ChnFtrs`) detector [6], similar in spirit to the work of Viola and Jones [16] (building upon the open source implementation of [1]). In recent publications this approach was shown to provide state-of-the-art quality [7, 2], speed [1], and a significant improvement over the traditional HOG+linear SVM detector [4]. The key difference is that `ChnFtrs` selects the distribution of image features to maximize discriminative power, instead of using a hand-designed grid like HOG+SVM.

The `ChnFtrs` detector is based on discrete Adaboost, using depth-2 decision trees as weak classifiers. The nodes of the trees are decision stumps, defined by a rectangular region in one of 10 image channels (6 quantized orientations, 1 gradient magnitude and 3 LUV color channels) together with a threshold over the sum of values over the region.

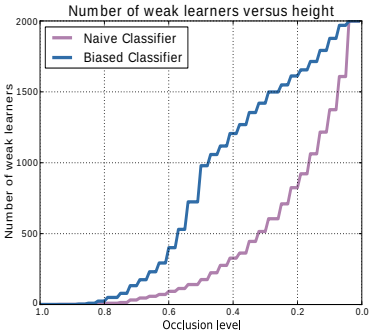
At each iteration of the training procedure a weak classifier must be built. The trees are built in a greedy fashion, learning one node at a time. For each node in the weak classifier, a set of candidate nodes is built using a predefined set of regions and exhaustively searching the optimal



(a) Each classifier is composed of a set of weak classifiers (dashed boxes), each weak classifier reads three regions of the image. When cutting a classifier, we eliminate all weak classifiers that read pixels in the occluded area



(b) Bottom occlusion with Naive approach: performance for different occlusion levels



(c) Comparison of the number of features remaining for the occluded classifiers

Figure 2: Simply "cutting" a full-body detector results in a significant quality drop, due to a exponential loss in the number of weak classifiers.

threshold values. The node that results in the smallest classification error is selected (algorithm 1 in the supplementary material). The errors are estimated using weighted samples, as traditionally done in Adaboost. For more details on this procedure, consult [6]. All our models are composed of 2 000 weak classifiers and are trained using two bootstrapping stages. All parameters are given in appendix A.

3. Classifiers for different occlusion levels

In this paper we consider the most frequent types of pedestrian occlusions: occlusions from the bottom and right/left. For each type we build a set of occlusion-specific classifiers in the range of $0\,\%$ to $50\,\%$ occlusion[2], see figure 1a. For our model of 16×32 pixels this corresponds to a maximum number of 8 right/left and 16 bottom-up occlusion-specific classifiers ($+1$ full-body classifier).

In the following sections we focus on bottom occlusions, but all conclusions are equally valid for right/left occlusions, as shown in the results of section 5.

3.1. Naive approach

The simplest approach to construct a set of occlusion-specific classifiers is to train one full-body classifier, and then "cut it" for each occlusion level: removing all weak classifiers with nodes whose rectangular regions overlap the occluded area (see figure 2a). The cuts are performed instantaneously and therefore do not add to the total training time.

In figure 2b we show the detection quality for each of these "naive" classifiers (see section 5 for evaluation method). It can be observed that quality drops drastically as occlusion increases (miss-rate is in log scale). For reference, we also show the results of HOG+SVM [4].

[2]In our experiments, past $50\,\%$ the detection quality becomes very low.

The performance drop can be explained by the number of weak classifiers left for a given level of occlusion. The quality of the detector is correlated with the number of weak classifiers. In figure 2c we present the number of weak classifiers as a function of the level of occlusion. It can be observed that already at 20% occlusion more than 50% percent of the weak classifiers have been lost (see corresponding illustration 3a).

Scrutinising the learned models shows that the regions used by weak classifiers are well distributed across the model. The exponential drop in weak classifiers indicates that most span a large part of the object height, *i.e.* many of them observe both the head and feet of the pedestrians.

In the next sections we explore alternatives to improve over the unsatisfactory quality of the naive approach.

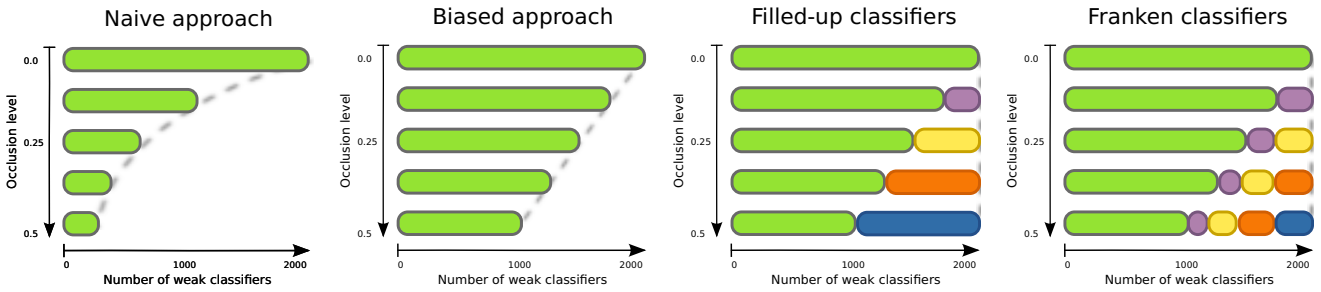
3.2. Brute-force approach

The best possible results can be obtained with a brute-force approach. For each occlusion level a new classifier is trained from scratch, restricted to only use the visible part. By construction, each occlusion-specific classifier will have selected the best weak classifiers for the task. This set puts an upper bound on the achievable quality (see figure 4).

Given the setup of section 2, training each classifier takes roughly 1 hour. Training the 17 classifiers (1 full-body + 16 occlusion levels) takes more than 18 hours. When training for multiple scales [1, 2], multiple classes, or for larger datasets, training can easily take *multiple days*. This is the **core problem** addressed in this paper.

4. Fast training of occlusion-specific classifiers

We propose a fast training method for occlusion-specific classifiers. To explain it we proceed in three stages, biased (§4.1), filled-up (§4.2), and finally, our fastest method,



(a) Naive approach (b) Biased approach (c) Filled-up classifiers (d) Franken-classifiers

Figure 3: Losses in the number of weak classifiers lead to losses in classification quality. A naive approach degrades rapidly in the presence of occlusion (figure 3a). We describe three approaches to cope with this problem (sections 4.1, 4.2 and 4.3).

Franken-classifiers (§4.3).

Some may consider training time of secondary importance. However both industry and research operate with limited time budgets. It is known that the quality of a method or implementation improves by "testing many ideas". Given a fixed time budget, we believe that being to "test out" 10 times more ideas is a significant gain.

4.1. Biased Classifier

The quality of the naive approach is too poor and the brute-force approach too slow to train. To cope with these issues, we propose to bias the classifier training towards a distribution of weak classifiers more suitable for generating ("cutting") occlusion-specific classifiers. This will result in the weak classifiers being more concentrated in the non-occluded areas than when training a non-constrained classifier (as in section 3.1).

The **key insight** of this work, is that it is possible to change the spatial distribution of the regions selected by the weak classifiers, without a significant quality drop[3].

At each iteration, the training algorithm picks the candidate node with the lowest error. As the number of possible nodes is very large and the discriminativity of each node low, the best ranking nodes typically have a performance very similar to the best selected node. In the next iteration Adaboost selects the best node with respect to the previous weak classifiers, thus selecting a slightly worse weak classifier in one stage does not necessarily imply that the final classifier will have worse performance.

To handle bottom occlusions we want to bias our weak classifiers upwards. A single parameter β is used to trade-off the weak classifiers position bias versus the quality of the resulting detector.

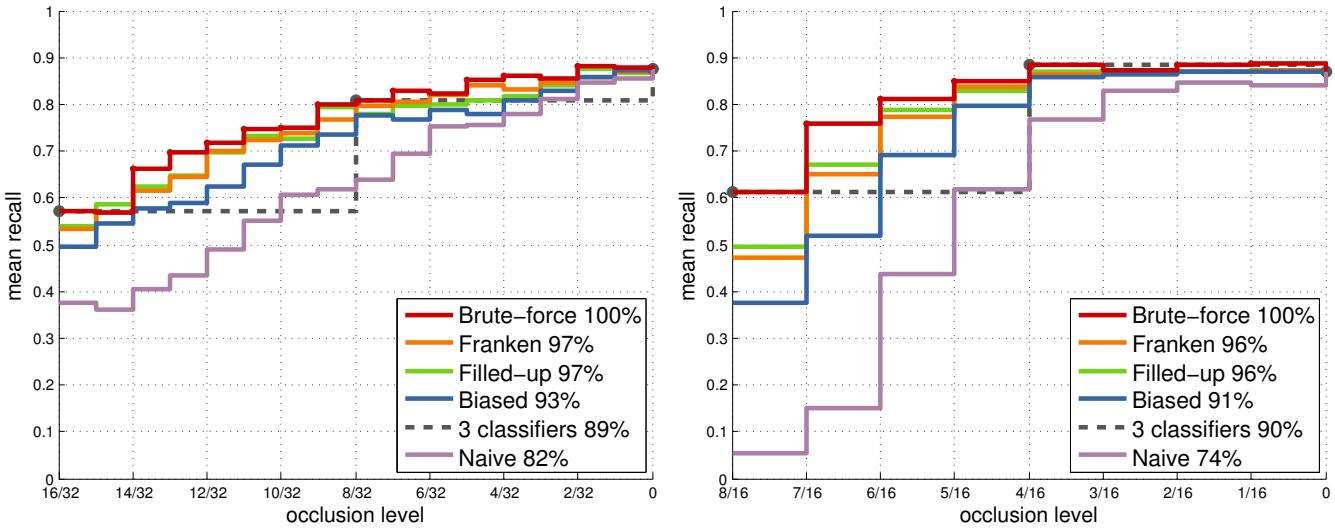
In rough term, at each iteration, β defines the set of nodes considered "top ranking" as the nodes with error inferior to $(1 + \beta) \cdot \text{min\_error}$. If a top ranking node is available in the upper $50\,\%$, it is selected, otherwise the uppermost top ranking node is selected. Given the root node of the decision tree, the child nodes are selected in a similar manner but using the root node bottom as boundary instead of $50\,\%$ of the model height. For more details please consult the supplementary material. In all our experiments we use $\beta = 0.05$. Our analysis show that β is not very sensitive, values in the range $[0.01,\ 0.08]$ provide similar results.

The learned biased classifier will be cut in a similar manner as the naive approach. Adaboost learns a linear combination of weak classifiers; its learning procedure is sensitive to the ordering of the selected weak classifiers. When removing weak classifiers based on a geometric criterion, they will be removed at arbitrary positions in the original classifier sequence. The previously learnt weights of the linear combination will then be sub-optimal. To improve the quality of the remaining strong classifier, we reset the weights by applying the Adaboost algorithm over the remaining weak classifiers sequence.

Figure 2c shows the obtained node distribution for the biased classifier (versus the naive approach). Where the weak classifiers are unconstrained, the biased classifier shows the same exponential behaviour as the naive classifier. Between 0% and 50% occlusion level, the curve has a roughly linear behaviour (see also illustration 3b). More weak classifiers lead to better quality.

Costs and benefits Training a biased classifier has essentially the same cost as the normal approach. Training time is dominated by the construction of the node sets, not by the node selection itself. Using the bias, many more weak classifiers remain at each occlusion level (up to $4\times$ more), this significantly boosts classification accuracy. Cutting, revisiting and resetting the weights of the weak classifiers has a negligible cost in comparison to the overall training.

[3]Less than 0.2 percent points on mean average miss-rate, well within the variance of the training method. See related figure in supplementary material.



(a) Bottom occlusion

(b) Right occlusion

Figure 4: Comparison of the different approaches to handle occlusion. Vertical axis indicates mean recall over 0.1 to 1 false positives per image, and thus is an indicator of quality.

4.2. Filled-up classifiers

The quality of the `ChnFtrs` detector depends on the number of weak classifiers. Although the biased classifier presented in the previous section presents a significant improvement over the naive approach, the number of weak classifiers still falls as the occlusion level increases. To further improve the situation we propose to train a single biased classifier, "cut it" for each occlusion level, and then extend each occlusion-specific classifier by training additional weak classifiers until reaching the same number of weak classifiers as in the full-body detector. We call these new classifiers "filled-up", see figure 3c. Although having the same number of weak classifiers does not equate to reaching an equal quality, we use this measure as a proxy.

Similar to the biased case, after cutting a classifier the weights of the remaining weak classifiers need to be reset. By doing so, we also obtain new weights for the data samples which are then used to initialize the next boosting round. After resetting the weights, the classifier is then extended using standard Adaboost training until we reach the desired amount of weak classifiers.

Costs and benefits The `ChnFtrs` classifier training is done in three stages (see appendix A). For the filled-up classifiers, only the last training stage is extended (first two rounds are a fix cost). Creating the set of candidate nodes dominates the time of this last stage, thus the cost of filling-up the classifiers is roughly linear to the number of added weak classifiers.
Starting from the full-body classifier, we need to build 16 additional occlusion-specific classifiers. A brute-force approach would require training $16 \times 2\,000 = 32\,000$ weak classifiers. Using the filled-up approach, experiments show that we only need to add about $\sim 10\,000$ weak classifiers, *i.e.* only one third of the brute-force cost. As we will show in section 5, the filled-up classifiers reach $97\,\%$ of the quality of the brute-force approach.

4.3. Franken-classifiers

The filled-up approach boosts quality, but still requires to train a significant number of weak classifiers. We can further decrease the training time by generating the occlusion-specific classifiers in a recursive way (see figure 3d).

Similar to the filled-up classifiers, we start from the full-body biased classifier and remove weak classifiers to generate the first occlusion classifier (least occluded). The additional weak classifiers are learned without spatial bias. Given the full classifier for the first occlusion level, we proceed to cut it using the second occlusion level. The second occlusion level is then filled-up. This process is repeated until the last occlusion level is reached. Because of the recursive training, the classifier for the last (and most drastic) occlusion level will potentially have weak classifiers originating from all previous occlusion levels (see figure 3d). We name this compound classifier the "Franken-classifier", paying tribute to Dr. Frankenstein [12] who would have appreciated the idea of "building a classifier out of many pieces".

Costs and benefits Compared to the filled-up classifier we further reduce the number of weak classifiers to be trained. Instead of $16 \times 2\,000 = 32\,000$ weak classi-

Type	Number of classifiers	Quality	Relative training time	Training time in minutes
Brute-force	17	100%	100%	1 088
5 classifiers	5	94%	29%	320
3 classifiers	3	89%	18%	192
Franken	1+16	**97%**	**8%**	64+28
Filled-up	1+16	97%	11%	64+54
Biased	1+16	93%	6%	64
Naive	1+16	82%	6%	64

Table 1: Comparison between the proposed approaches.

fiers for the brute-force approach, or $10\,000$ for the filled-up case; our experiments show that we only need to add about $\sim 6\,000$ weak classifiers to the last training stage. We now need less than one fifth of the brute-force training cost. In our experiments, the Franken-classifiers reach the same quality as the filled-up classifiers.

5. Independent Franken-classifiers evaluation

In the previous sections we presented different methods to obtain occlusion classifiers. All descriptions so far have referred to occlusions in the lower part of the pedestrian. For occlusions from the right we use exactly the same procedure and parameters. The occlusion-specific classifiers from the left are obtained by mirroring the right classifiers.

5.1. Evaluation method

We first evaluate our occlusion-specific classifier independently and show their performance for each occlusion type and level. Evaluating occlusion cases is a delicate matter [11]. Most pedestrian datasets contain only few annotated occluded pedestrians, for instance, Caltech USA [7] has only 100 pedestrians in the "partially occluded" range. To work around this issue, we propose to use the larger set of non-occluded pedestrians. Each occlusion-specific classifier is evaluated using a sliding-window over the *whole image*. Each test window is classified using only pixels that are located in the non-occluded area. *E.g.* the classifier for pedestrians occluded by $50\,\%$ from the bottom, only contains features located in the upper part of the test window. For these evaluations we use the INRIA dataset [4].

With this approach we have more testing data, we evaluate the occlusion-specific detectors in a setup more similar to their test-time usage (FPPI vs FPPW, see [7, figure 10]), and the results for each occlusion level are comparable since all occlusion-specific classifiers are evaluated over the same pedestrians.

In section 6 we discuss an evaluation that uses all the occlusion-specific detectors jointly.

5.2. Classifier quality

To compare the different classifiers, we average the miss-rate in the interval from 0.1 to 1 false positives per image. Figure 4 summarizes the result of all 76[4] trained classifiers over 124[5] evaluations on the INRIA test set. Each single evaluation curve in figure 2b is represented as one step in figure 4. For a given level of occlusion we use the closest classifier not overlapping with the occlusion. Percentages after the labels indicate the ratio of the area under the curve compared to the brute-force approach. The method "`3 classifiers`" refers to using brute-force classifiers at 0%, 25% and 50% occlusion.

It can be observed that our proposed methods significantly improve over the naive approach, and reach comparable quality to brute-force.

5.3. Training time computational cost

Table 1 relates the quality of the occlusion classifiers to the measured training time (wall time). As mentioned in section 2, we use three training stages (two bootstrapping rounds). The biased approach is applied to all stages, while filling-up and creating the Franken-classifiers is only done in the last stage. Due to this, the wall time is not directly proportional to the weak classifiers count. For the brute-force approach each classifier is trained independently.

Given our results, the biased classifiers should be preferred over the naive approach, as the quality for all occlusion levels is much better, while the training time remains the same. The Franken-classifiers provide even higher quality with very low training time. Training 3 or 5 brute-force classifiers takes more time than the proposed approaches, while still having lower quality.

5.4. Test time computational cost

The individual Franken-classifiers can be used in a straight forward fashion on the image borders to detect truncated pedestrians. When searching pedestrians over an image with 640×480 pixels over 55 scales ($0.4\times$ to $5\times$), around 5 million candidate detection windows will be evaluated. Handling up to 50% occlusions on the left, right and bottom borders generates only 9% additional candidate windows. Importantly, using 17 models or 3 models, corresponds to exactly the same amount of window evaluations, and thus to the exact same evaluation cost (assuming all models have the same number of weak classifiers). In this setting, the number of models affects the quality and the memory usage, but not the test time. The same applies

[4] Trainings for bottom occlusions: 17 (brute-force) + 1 (biased) + 32 (filled-up and Franken-classifiers); respectively, from the right: 9 + 1 + 16

[5] Evaluations for occlusions from the bottom: 16×5 (brute-force, filled-up, Franken, biased, and naive) + 1 (shared evaluation for brute-force and naive) + 1 (shared for biased, filled-up and Franken); and respectively, occlusions from the right: (8×5) + 1 + 1

for any scenario in which occlusions (of unknown objects) can be detected, e.g. using depth sensors, optical flow, or tracking information.

6. Joint Franken-classifiers evaluation

In the previous section we evaluated our Franken-classifiers in the scenario where occlusions are known, but the presence of a pedestrian on such occlusion boundaries is unknown. In this section we consider detecting pedestrians without knowing a priori where the occlusions occur. To do so, we evaluate all our Franken-classifiers everywhere on the image, and then merge the detections. As full-body classifier we use the biased classifier for the bottom occlusion. The classifier scores are roughly calibrated by normalizing the maximum achievable detection score.

As a *proof of concept* we propose merging the different classifier results by non-maximum suppression over bounding boxes. This approach is surely suboptimal, using a more sophisticated approach such as [18] would further increase the detection results.

Merging detections Our merging approach is based on two principles: "detectors with higher occlusion levels have worse quality", and "the Franken-classifiers should complement the full-body detections". To account for the first principle, we reweight the detections by the cube of 1 - occlusion level. To account for the second principle we do non-maxima suppression in two stages. Since occlusion detectors will trigger on fully visible pedestrians, for each zero occlusion detection we remove all overlapping detections, and increase its score by adding up the score of the overlapping bounding boxes. In a second stage we do non-maxima suppression between all occluded detections, and join the two sets to obtain our final detection bounding boxes.

6.1. Detection quality

In this section we use as base classifier the `SquaresChnFtrs` [2]. It is identical to `ChnFtrs`, but instead of a random set of rectangles it uses all possible squares as feature pool. Additionally, to reflect the lower confidence in detection of smaller scale pedestrians, we linearly penalise the score of detections smaller than the dataset's median bounding box height. All images are pre-processed as in [2].

We run the evaluations using the toolbox provided in [7]. To reduce visual clutter we have included only top performing methods. In figure 5 we show the joint detection performance of our 33 Franken-classifiers and use as comparison baseline the unbiased `SquaresChnFtrs` classifier.

Using the Franken-classifiers the detection quality improves on all dataset, reaching top quality (for single scale models that do not consider context or multiple frames). On Caltech we reach the best reported results for partial and heavily occluded pedestrians. In the supplementary material we include the additional Caltech occlusion ranges, and show that 33 classifiers improves over using only 7.

Figure 1b shows one example of our joint detection on the ETH dataset.

6.2. Test time computational cost

Using the same setup as this paper, the maximal reported detection speed for a single detector is 50 Hz [1, 5]. Running all $8 \times 2 + 16 + 1 = 33$ independently would increase the testing time by a factor of $33\times$.

Similar to how we can train many occlusion models in a fraction of the time of a full model, we can also evaluate our 33 classifiers much faster than independently. By design our training procedure generate models that share many features (figure 3d). These shared features need to be calculated only once per candidate detection window. In our joint experiment, the total amount of unique weak classifiers to evaluate sums up to $\sim$ 14 000, this is significantly less than $2\,000 \times 33 = 66\,000$. Exploiting the shared features, enables a $\sim 5\times$ speed-up with respect to a naive brute force evaluation. Sharing features between models to reduce test time was previously explored in work such as [14]. There the sharing needs to be added into the training procedure, while our Franken-classifiers share features by design.

Given our training procedure, all models in each occlusion type share at least ~1000 features (number of remaining features at $50\,\%$ occlusion level, figure 2c). When using a soft cascade over `ChnFtrs` [5, 1], in average as few as $\sim$ 20 weak learners are evaluated per detection window. After sorting the common features upfront, each occlusion type shares the same initial ~ 1000 features, which enables the soft cascade to directly drop sets of classifiers; further speeding up the evaluation with respect to the brute force case (soft cascades usually bring a $10\times$ speedup).

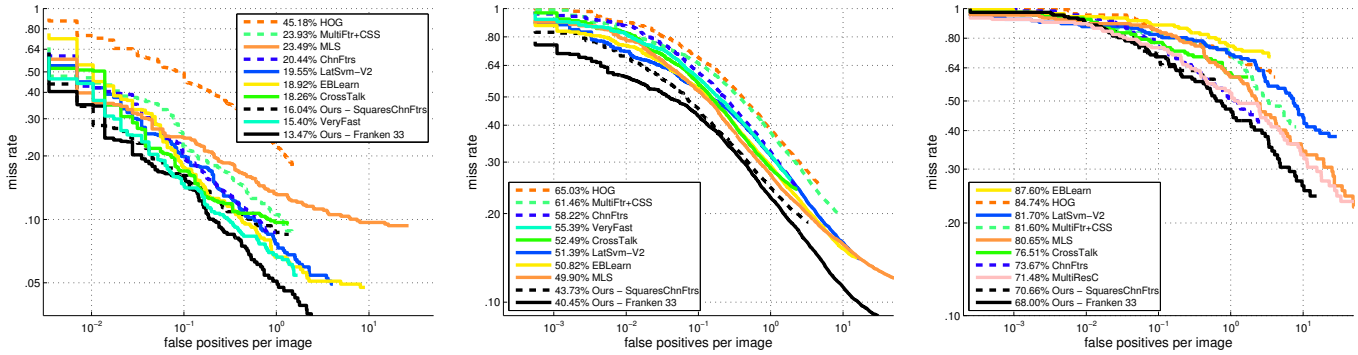
We estimate that a speed aware implementation, using the methods of [1] and [5] should reach comfortably 5 Hz or more. Such implementation is left for future work.

7. Conclusion

To the best of our knowledge this is the first work that investigates the behaviour of the `ChnFtrs` detector under occlusion. We have shown that a naive approach to handle occlusions provides poor quality, and that occlusion-specific classifiers can perform significantly better.

We proposed a new approach that results in sub-linear cost for both training and testing occlusion-specific classifiers. A proof of concept usage of the Franken-classifiers shows that we can reach top quality detection on challenging pedestrian datasets. We expect more sophisticated fusion methods will further improve results.

Acknowledgements: Work partially supported by the Toyota Motor Corporation, and the IWT project PARIS.



(a) INRIA (b) ETH reasonable (c) Caltech USA partially occluded subset

Figure 5: Improved detection quality when using occlusion-specific classifiers.

References

[1] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *CVPR*, 2012.

[2] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *CVPR*, 2013.

[3] M. Blaschko and C. Lampert. Learning to Localize Objects with Structured Output Regression. In *ECCV*, 2008.

[4] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[5] P. Dollár, R. Appel, and W. Kienzle. Crosstalk Cascades for Frame-Rate Pedestrian Detection. In *ECCV*, 2012.

[6] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *BMVC*, 2009.

[7] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian Detection: An Evaluation of the State of the Art. *TPAMI*, 2011.

[8] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *CVPR*, 2010.

[9] T. Gao, B. Packer, and D. Koller. A Segmentation-aware Object Detection Model with Occlusion Handling. In *CVPR*, 2011.

[10] R. Girshick, P. Felzenszwalb, and D. McAllester. Object Detection with Grammar Models. In *NIPS*, 2011.

[11] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing Error in Object Detectors. In *ECCV*, 2012.

[12] M. W. Shelley. *Frankenstein; or, the modern Prometheus*. 1818.

[13] S. Tang, M. Andriluka, and B. Schiele. Detection and Tracking of Occluded People. In *BMVC*, 2012.

[14] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient bossting procedures for multiclass object detection. In *CVPR*, 2004.

[15] A. Vedaldi and A. Zisserman. Structured Output Regression for Detection with Partial Truncation. In *NIPS*, 2009.

[16] P. Viola and M. Jones. Robust Real-Time Face Detection. In *IJCV*, 2004.

[17] X. Wang, X. Han, and S. Yan. HOG-LBP human detector with partial occlusion handling. In *ICCV*, 2009.

[18] P. Wohlhart, M. Donoser, P. Roth, and H. Bischof. Detecting Partially Occluded Objects with an Implicit Shape Model Random Field. In *ACCV*, 2012.

[19] C. Wojek, S. Walk, S. Roth, and B. Schiele. Monocular 3D scene understanding with explicit occlusion reasoning. In *CVPR*, 2011.

[20] B. Wu and R. Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. In *ICCV*, 2005.

A. Training parameters

Unless otherwise specified we use the parameters that provided the best classification results in the original paper [6]. The nodes are constructed using a pool of 30 000 candidate regions. The full classifier consists of 2 000 weak classifiers. We train in 3 stages; the first stage randomly samples 5 000 negative samples, the second and third stage use bootstrapping to add 5 000 additional hard negatives each. To be faster and memory efficient we shrink the feature channels by a factor 4 (see [6, addendum]). The model window is of size 64×128 pixels, after shrinking it has size 16×32 pixels. Training time is measured on a desktop machine with an Intel Core i7 870 CPU and a Nvidia GeForce GTX 590 GPU.