Filtered Channel Features for Pedestrian Detection

Shanshan Zhang Rodrigo Benenson Bernt Schiele

Max Planck Institute for Informatics
Saarbrücken, Germany

firstname.lastname@mpi-inf.mpg.de

Abstract

This paper starts from the observation that multiple top performing pedestrian detectors can be modelled by using an intermediate layer filtering low-level features in combination with a boosted decision forest. Based on this observation we propose a unifying framework and experimentally explore different filter families. We report extensive results enabling a systematic analysis.

Using filtered channel features we obtain top performance on the challenging Caltech and KITTI datasets, while using only HOG+LUV as low-level features. When adding optical flow features we further improve detection quality and report the best known results on the Caltech dataset, reaching 93% recall at 1 FPPI.

1. Introduction

Pedestrian detection is an active research area, with 1000+ papers published in the last decade[1], and well established benchmark datasets [9, 13]. It is considered a canonical case of object detection, and has served as playground to explore ideas that might be effective for generic object detection.

Although many different ideas have been explored, and detection quality has been steadily improving [2], arguably it is still unclear what are the key ingredients for good pedestrian detection; e.g. it remains unclear how effective parts, components, and features learning are for this task.

Current top performing pedestrian detection methods all point to an intermediate layer (such as max-pooling or filtering) between the low-level feature maps and the classification layer [42, 45, 29, 25]. In this paper we explore the simplest of such intermediary: a linear transformation implemented as convolution with a filter bank. We propose a framework for filtered channel features (see figure 1) that unifies multiple top performing methods [8, 1, 45, 25], and that enables a systematic exploration of different filter banks. Our experiments show that, with the proper filter bank, filtered channel features reach top detection quality.

[1]Papers from 2004 to 2014 with "pedestrian detection" in the title, according to Google Scholar.

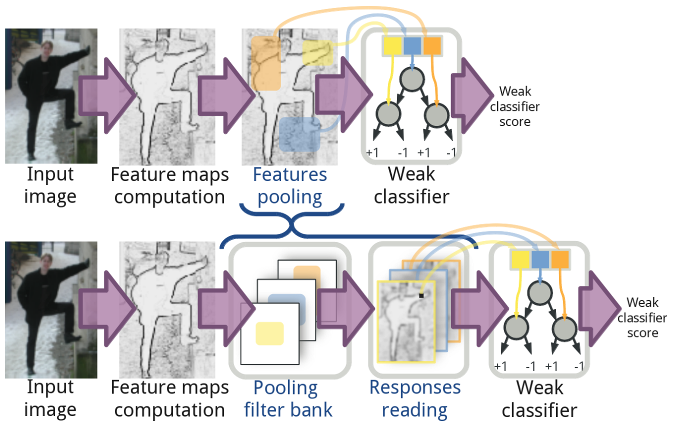


Figure 1: Filtered feature channels illustration, for a single weak classifier reading over a single feature channel. Integral channel features detectors pool features via sums over rectangular regions [8, 1]. We can equivalently rewrite this operation as convolution with a filter bank followed by single pixel reads (see §2). We aim to answer: *What is the effect of selecting different filter banks?*

It has been shown that using extra information at test time (such as context, stereo images, optical flow, etc.) can boost detection quality. In this paper we focus on the "core" sliding window algorithm using solely HOG+LUV features (i.e. oriented gradient magnitude and colour features). We consider context information and optical flow as add-ons, included in the experiments section for the sake of completeness and comparison with existing methods. Using only HOG+LUV features we already reach top performance on the challenging Caltech and KITTI datasets, matching results using optical flow and significantly more features (such as LBP and covariance [42, 29]).

1.1. Related work

Recent survey papers discuss the diverse set of ideas explored for pedestrian detection [10, 14, 9, 2]. The most recent survey [2] indicates that the classifier choice (e.g. linear/non-linear SVM versus decision forest) is not a clear differentiator regarding quality; rather the features used seem more important.

Creativity regarding different types of features has not been lacking. **HOG)** The classic HOG descriptor is based on local image differences (plus pooling and normalization steps), and has been used directly [5], as input for a deformable parts model [11], or as features to be boosted [20, 26]. The integral channel features detector [8, 1] uses a simpler HOG variant with sum pooling and no normalizations. Many extensions of HOG have been proposed (e.g. [17, 11, 6, 34]). **LBP)** Instead of using the magnitude of local pixel differences, LBP uses the difference sign only as signal [41, 42, 29]. **Colour)** Although the appearance of pedestrians is diverse, the background and skin areas do exhibit a colour bias. Colour has shown to be an effective feature for pedestrian detection and hence multiple colour spaces have been explored (both hand-crafted and learned) [8, 18, 19, 23]. **Local structure)** Instead of simple pixel values, some approaches try to encode a larger local structure based on colour similarities (soft-cue) [40, 15], segmentation methods (hard-decision) [27, 32, 36], or by estimating local boundaries [21]. **Covariance)** Another popular way to encode richer information is to compute the covariance amongst features (commonly colour, gradient, and oriented gradient) [38, 29]. **Etc.)** Other features include bag-of-words over colour, HOG, or LBP features [4]; learning sparse dictionary encoders [33]; and training features via a convolutional neural network [35] ([37, 16] appeared while preparing this manuscript). Additional features specific for stereo depth or optical flow have been proposed, however we consider these beyond the focus of this paper. For our flow experiments we will use difference of frames from weakly stabilized videos (`SDt`) [30].

All the feature types listed above can be used in the integral channel features detector framework [8]. This family of detectors is an extension of the old ideas from Viola&Jones [39]. Sums of rectangular regions are used as input to decision trees trained via Adaboost. Both the regions to pool from and the thresholds in the decision trees are selected during training. The crucial difference from the pioneer work [39] is that the sums are done over feature channels other than simple image luminance.

Current top performing pedestrian detection methods (dominating INRIA [5], Caltech [9] and KITTI datasets [13]) are all extensions of the basic integral channel features detector (named `ChnFtrs` in [8], which uses only HOG+LUV features). `SquaresChnFtrs` [2], `InformedHaar` [45], and `LDCF` [25], are discussed in detail in section 2.2. `Katamari` exploits context and optical flow for improved performance. `SpatialPooling(+)` [29] adds max-pooling on top of sum-pooling, and uses additional features such as covariance, LBP, and optical flow. Similarly, `Regionlets` [42] also uses extended features and max-pooling, together with stronger weak classifiers and training a cascade of classifiers. Out of these, `Regionlets` is the only method that has also shown good performance on general classes datasets such as Pascal VOC and ImageNet.

In this paper we will show that vanilla HOG+LUV features have not yet saturated, and that, when properly used, they can reach top performance for pedestrian detection.

1.2. Contributions

- We point out the link between `ACF` [7], `(Squares)ChnFtrs` [8, 1, 2], `InformedHaar` [45], and `LDCF` [25]. See section 2.
- We provide extensive experiments to enable a systematic analysis of the filtered integral channels, covering aspects not explored by related work. We report the summary of 65+ trained models ($\sim$10 days of single machine computation). See sections 4, 5 and 7.
- We show that top detection performance can be reached on Caltech and KITTI using HOG+LUV features only. We additionally report the best known results on Caltech. See section 7.

2. Filtered channel features

Before entering the experimental section, let us describe our general architecture. Methods such as `ChnFtrs` [8], `SquaresChnFtrs` [1, 2] and `ACF` [7] all use the basic architecture depicted in figure 1 (top part, best viewed in colours). The input image is transformed into a set of feature channels (also called feature maps), the feature vector is constructed by sum-pooling over a (large) set of rectangular regions. This feature vector is fed into a decision forest learned via Adaboost. The split nodes in the trees are a simple comparison between a feature value and a learned threshold. Commonly only a subset of the feature vector is used by the learned decision forest. Adaboost serves both for feature selection and for learning the thresholds in the split nodes. For more details on this basic architecture, please consult [8, 1].

A key observation, illustrated in figure 1 (bottom), is that such sum-pooling can be re-written as convolution with a filter bank (one filter per rectangular shape) followed by reading a single value of the convolution's response map. This "filter + pick" view generalizes the integral channel features [8] detectors by allowing to use any filter bank (instead of only rectangular shapes). We name this generalization "filtered channel features detectors".

In our framework, `ACF` [7] has a single filter in its bank, corresponding to a uniform 4×4 pixels pooling region. `ChnFtrs` [8] was a very large (tens of thousands) filter bank comprised of random rectangular shapes. `SquaresChnFtrs` [1, 2], on the other hand, has only 16 filters, each with a square-shaped uniform pooling region of different sizes. See figure 2a for an illustration of the `SquaresChnFtrs` filters, the upper-left filter corresponds to `ACF`'s one.

The `InformedHaar` [45] method can also be seen as a filtered channel features detector, where the filter bank (and read locations) are based on a human shape template (thus the "informed" naming). `LDCF` [25] is also a particular instance of this framework, where the filter bank consists of PCA bases of patches from the training dataset. In sections 4 and 5 we provide experiments revisiting some of the design decisions of these methods.

Note that all the methods mentioned above (and in the majority of experiments below) use only HOG+LUV feature channels[2] (10 channels total). Using linear filters and decision trees on top of these does not allow to reconstruct the decision functions obtained when using LBP or covariance features (used by [42, 29]). Compared to `SpatialPooling` [29] and `Regionlets` [42] the main differences are that we use simpler features, do pooling only via filtering (instead of mixed mean and max-pooling), use simpler weak classifiers (short decision trees), and vanilla discrete Adaboost. We consider the approach considered here mainly orthogonal to the ideas in [42, 29].

2.1. Evaluation protocol

For our experiments we use the Caltech [9, 2] and KITTI datasets [13]. The popular INRIA dataset is considered too small and too close to saturation to provide interesting results. All Caltech results are evaluated using the provided toolbox, and summarised by log-average miss-rate (MR, lower is better) in the $\left[10^{-2}, 10^{0}\right]$ FPPI range for the "reasonable" setup. KITTI results are evaluated via the online evaluation portal, and summarised as average precision (AP, higher is better) for the "moderate" setup.

Caltech10x The raw Caltech dataset consists of videos (acquired at 30 Hz) with every frame annotated. The standard training and evaluation considers one out of each 30 frames (1 631 pedestrians over 4 250 frames in training, 1 014 pedestrians over 4 024 frames in testing).
In our experiments of section 5 we will also consider a $10\times$ increased training set where every 3rd frame is used (linear growth in pedestrians and images). We name this extended training set "Caltech10x". `LDCF` [25] uses a similar extended set for training its model (every 4th frame).

Flow Methods using optical flow do not only use additional neighbour frames during training ($1 \leftrightarrow 4$ depending on the method), but they also do so at test time. Because they have access to additional information at test time, we consider them as a separate group in our results section.

Validation set In order to explore the design space of our pedestrian detector we setup a Caltech validation set by splitting the six training videos into five for training and one for testing (one of the splits suggested in [9]). Most of our experiments use this validation setup. We also report (a posteriori) our key results on the standard test set for comparison to the state of the art.
For the KITTI experiments we also validate some design choices (such as search range and number of scales) before submission on the evaluation server. There we use a $^2/_3 + ^1/_3$ validation setup.

[2]We use "raw" HOG, without any clamping, cell normalization, block normalization, or dimensionality reduction.

2.2. Baselines

`ACF` Our experiments are based on the open source release of `ACF` [7]. Our first baseline is vanilla `ACF` re-trained on the standard Caltech set (*not* Caltech10x), all parameter details are described in section 2.3, and kept identical across experiments unless explicitly stated. On the Caltech test set it obtains 32.6% MR (50.2% MR on validation set). Note that this baseline already improves over more than 50 previously published methods [2] on this dataset. There is also a large gap between `ACF-Ours` (32.6% MR) and the original number from `ACF-Caltech` (44.2% MR [7]). This improvement is mainly due to the change towards a larger model size (from 32×64 pixels in [7] to 60×120 here).

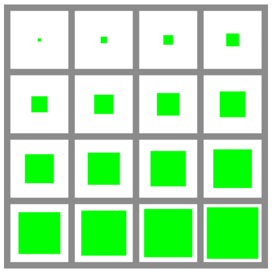
`InformedHaar` Our second baseline is a reimplementation of `InformedHaar` [45]. Here again we observe an important gain from using a larger model size (same change as for `ACF`). While the original `InformedHaar` paper reports 34.6% MR, `InformedHaar-Ours` reaches 27.0% MR on the Caltech test set (39.3% MR on validation set).

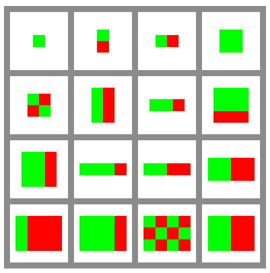
For both our baselines we use exactly the same training set as the original papers. Note that the `InformedHaar-Ours` baseline (27.0% MR) is right away the best known result for a method trained on the standard Caltech training set. In section 3 we will discuss our reimplementation of `LDCF` [25].

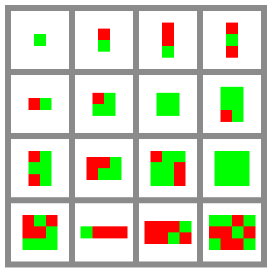
2.3. Model parameters

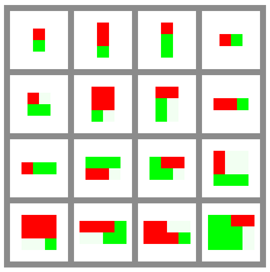
Unless otherwise specified we train all our models using the following parameters. Feature channels are HOG+LUV only. The final classifier includes 4096 level-2 decision trees (L2, 3 stumps per tree), trained via vanilla discrete Adaboost. Each tree is built by doing exhaustive greedy search for each node (no randomization). The model has size 60×120 pixels, and is built via four rounds of hard negative mining (starting from a model with 32 trees, and then 512, 1024, 2048, 4096 trees). Each round adds 10 000 additional negatives to the training set. The sliding window stride is 6 pixels (both during hard negative mining and at test time).

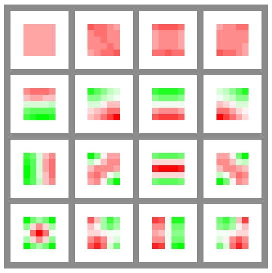
Compared to the default `ACF` parameters, we use a bigger model, more trees, more negative samples, and more boosting rounds. But we do use the same code-base and the same training set.


(a) `SquaresChntrs` filters


(b) `Checkerboards` filters


(c) `RandomFilters`


(d) `InformedFilters`


(e) `LDCF8` filters

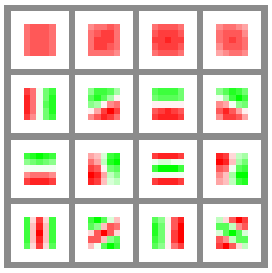

(f) `PcaForeground` filters

Figure 2: Illustration of the different filter banks considered. Except for `SquaresChntrs` filters, only a random subset of the full filter bank is shown. {🟥 Red, □ White, 🟩 Green} indicate $\{-1, 0, +1\}$.

Starting from section 5 we will consider results using Caltech10x. There, better performance is reached when using level-4 decision trees (L4), and Realboost [12] instead of discrete Adaboost. All other parameters are left unchanged.

3. Filter bank families

Given the general architecture and the baselines described in section 2, we now proceed to explore different types of filter banks. Some of them are designed using prior knowledge and they do not change when applied across datasets, others exploit data-driven techniques for learning their filters. Sections 4 and 5 will compare their detection quality.

`InformedFilters` Starting from the `InformedHaar` [45] baseline we use the same "informed" filters but let free the positions where they are applied (instead of fixed in `InformedHaar`); these are selected during the boosting learning. Our initial experiments show that removing the position constraint has a small (positive) effect. Additionally we observe that the original `InformedHaar` filters do not include simple square pooling regions (à la `SquaresChnFtrs`), we thus add these too. We end up with 212 filters in total, to be applied over each of the 10 feature channels. This is equivalent to training decision trees over 2120 (non filtered) channel features.
As illustrated in figure 2d the `InformedFilters` have different sizes, from 1×1 to 4×3 cells (1 cell $=$ 6×6 pixels), and each cell takes a value in $\{-1, 0, +1\}$. These filters are applied with a step size of 6 pixels. For a model of 60×120 pixels this results in 200 features per channel, $2\,120 \cdot 200 = 424\,000$ features in total[3]. In practice considering border effects (large filters are not applied on the border of the model to avoid reading outside it) we end up with $\sim$300 000 features. When training 4 096 level-2 decision trees, at most $4\,096 \cdot 3 = 12\,288$ features will be used, that is $\sim$3% of the total. In this scenario (and all others considered in this paper) Adaboost has a strong role of feature selection.

`Checkerboards` As seen in section 2.2 `InformedHaar` is a strong baseline. It is however unclear how much the "informed" design of the filters is effective compared to other possible choices. `Checkerboards` is a naïve set of filters that covers the same sizes (in number of cells) as `InformedHaar/InformedFilters` and for each size defines (see figure 2b): a uniform square, all horizontal and vertical gradient detectors (±1 values), and all possible checkerboard patterns. These configurations are comparable to `InformedFilters` but do not use the human shape as prior.
The total number of filters is a direct function of the maximum size selected. For up to 4×4 cells we end up with 61 filters, up to 4×3 cells 39 filters, up to 3×3 cells 25 filters, and up to 2×2 cells 7 filters.

`RandomFilters` Our next step towards removing a hand-crafted design is simply using random filters (see figure 2c). Given a desired number of filters and a maximum filter size (in cells), we sample the filter size with uniform distribution, and set its cell values to ±1 with uniform probability. We also experimented with values $\{-1, 0, +1\}$ and observed a (small) quality decrease compared to the binary option).

The design of the filters considered above completely ignores the available training data. In the following, we consider additional filters learned from data.

[3]"Feature channel" refers to the output of the first transformation in figure 1 bottom. "Filters" are the convolutional operators applied to the feature channels. And "features" are entries in the response maps of all filters applied over all channels. A subset of these features are the input to the learned decision forest.

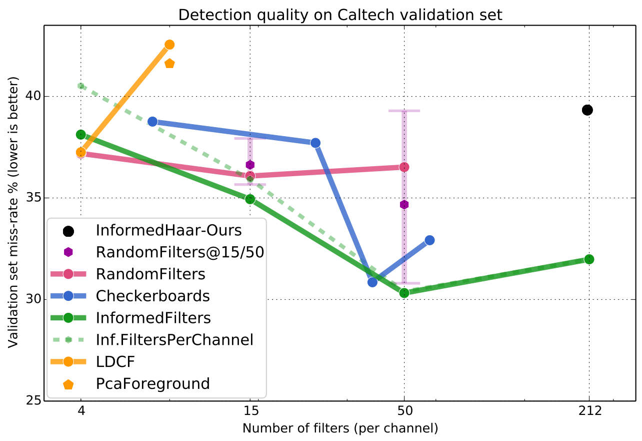


Figure 3: Detection quality (log-average miss-rate MR, lower is better) versus number of filters used. All models trained and tested on the Caltech validation set (see §4).

LDCF [25] The work on PCANet [3] showed that applying arbitrary non-linearities on top of PCA projections of image patches can be surprisingly effective for image classification. Following this intuition LDCF [25] uses learned PCA eigenvectors as filters (see figure 2e).
We present a re-implementation of [25] based on ACF's [7] source code. We follow the original description as closely as possible. We use the same top 4 filters of 10×10 pixels, selected per feature channel based on their eigenvalues (40 filters total). We do change some parameters to be consistent amongst all experiments, see sections 2.3 and 5. The main changes are the training set (we use Caltech10x, sampled every 3 frames, instead of every 4 frames in [25]), and the model size (60×120 pixels instead of 32×64). As seen in section 7, our implementation (LDCF-Ours) clearly improves over the previously published numbers [25], showing the potential of the method.
For comparison with PcaForeground we also consider training LDCF8 where the top 8 filters are selected per channel (80 filters total).

PcaForeground In LDCF the filters are learned using all of the training data available. In practice this means that the learned filters will be dominated by background information, and will have minimal information about the pedestrians. Put differently, learning filters from all the data assumes that the decision boundary is defined by a single distribution (like in Linear Discriminant Analysis [24]), while we might want to define it based on the relation between the background distribution and the foreground distribution (like Fisher's Discriminant Analysis [24]). In PcaForeground we train 8 filters per feature channel, 4 learned from background image patches, and 4 learned from patches extracted over pedestrians (see figure 2f). Compared to LDCF8 the obtained filters are similar but not identical, all other parameters are kept identical.

Other than via PcaForeground/LDCF8, it is not clear how to further increase the number of filters used in LDCF. Past 8 filters per channel, the eigenvalues decrease to negligible values and the eigenvectors become essentially random (similar to RandomFilters).

To keep the filtered channel features setup close to InformedHaar, the filters are applied with a step of 6 pixels. However, to stay close to the original LDCF, the LDCF/PcaForeground filters are evaluated every 2 pixels. Although (for example) LDCF8 uses only ∼10% of the number of filters per channel compared to Checkerboards4x4, due to the step size increase, the obtained feature vector size is ∼40%.

4. How many filters?

Given a fixed set of channel features, a larger filter bank provides a richer view over the data compared to a smaller one. With enough training data one would expect larger filter banks to perform best. We want thus to analyze the trade-off between number of filters and detection quality, as well as which filter bank family performs best.

Figure 3 presents the results of our initial experiments on the Caltech validation set. It shows detection quality versus number of filters per channel. This figure densely summarizes ∼30 trained models.

InformedFilters The first aspect to notice is that there is a meaningful gap between InformedHaar-Ours and InformedFilters despite having a similar number of filters (209 versus 212). This validates the importance of letting Adaboost choose the pooling locations instead of hand-crafting them. Keep in mind that InformedHaar-Ours is a top performing baseline (see §2.2).
Secondly, we observe that (for the fixed training data available) ∼50 filters is better than ∼200. Below 50 filters the performance degrades for all methods (as expected).
To change the number of filters in InformedFilters we train a full model (212 filters), pick the N most frequently used filters (selected from node splitting in the decision forest), and use these to train the desired reduced model. We can select the most frequent filters across channels or per channel (marked as Inf.FiltersPerChannel). We observe that per channel selection is slightly worse than across channels, thus we stick to the latter.
Using the most frequently used filters for selection is clearly a crude strategy since frequent usage does not guarantee discriminative power, and it ignores relation amongst filters. We find this strategy good enough to convey the main points of this work.

Checkerboards also reaches best results in the ∼50 filters region. Here the number of filters is varied by chan-

Training	Method	L2	L3	L4	L5
Caltech	ACF	50.2	*42.1*	48.8	48.7
Caltech10x		52.6	49.9	44.9	*41.3*
Caltech	Checker-boards	32.9	30.4	*28.0*	31.5
Caltech10x		37.0	31.6	*24.7*	*24.7*

Table 1: Effect of the training volume and decision tree depth (Ln) over the detection quality (average miss-rate on validation set, lower is better), for ACF-Ours and Checkerboards variant with (61) filters of 4×4 cells. We observe a similar trend for other filter banks.

ging the maximum filter size (in number of cells). Regarding the lowest miss-rate there is no large gap between the "informed" filters and this naïve baseline.

RandomFilters The hexagonal dots and their deviation bars indicate the mean, maximum and minimum miss-rate obtained out of five random runs. When using a larger number of filters (50) we observe a lower (better) mean but a larger variance compared to when using fewer filters (15). Here again the gap between the best random run and the best result of other methods is not large.
Given a set of five models, we select the N most frequently used filters and train new reduced models; these are shown in the RandomFilters line. Overall the random filters are surprisingly close to the other filter families. This indicates that expanding the feature channels via filtering is the key step for improving detection quality, while selecting the "perfect" filters is a secondary concern.

LDCF/PcaForeground In contrast to the other filter bank families, LDCF under-performs when increasing the number of filters (from 4 to 8) while using the standard Caltech training set (consistent with the observations in [25]). PcaForeground improves marginally over LDCF8.

Takeaways From figure 3 we observe two overall trends. First, the more filters the merrier, with ~50 filters as sweet spot for Caltech training data. Second, there is no flagrant difference between the different filter types.

5. Additional training data

One caveat of the previous experiments is that as we increase the number of filters used, so does the number of features Adaboost must pick from. Since we increased the model capacity (compared to ACF which uses a single filter), we consider using the Caltech10x dataset (§2.1) to verify that our models are not starving for data. Similar to the experiments in [25], we also reconsider the decision tree depth, since additional training data enables bigger models.

Results for two representative methods are collected in table 1. First we observe that already with the original training data, deeper trees do provide significant improvement over level-2 (which was selected when tuning over INRIA data [8, 1]). Second, we notice that increasing the training data volume does provide the expected improvement only when the decision trees are deep enough. For our following experiments we choose to use level-4 decision trees (L4) as a good balance between increased detection quality and reasonable training times.

Aspect	MR	ΔMR
ACF-Ours	50.8	-
+ filters	32.9	+17.9
+ L4	28.0	+4.9
+ Caltech10x	24.7	+3.3
+ Realboost	24.4	+0.3
Checkerboards4x4	24.4	+26.4

Table 2: Ingredients to build our strong detectors (using Checkerboards4x4 in this example, 61 filters). Validation set log-average miss-rate (MR).

Realboost Although previous papers on ChnFtrs detectors reported that different boosting variants all obtain equal results on this task [8, 1], the recent [25] indicated that Realboost has an edge over discrete Adaboost when additional training data is used. We observe the same behaviour in our Caltech10x setup.

As summarized in table 2 using filtered channels, deeper trees, additional training data, and Realboost does provide a significant detection quality boost. For the rest of the paper our models trained on Caltech10x all use level-4 trees and RealBoost, instead of level-2 and discrete Adaboost for the Caltech1x models.

Timing When using Caltech data ACF takes about one hour for training and one for testing. Checkerboards-4x4 takes about 4 and 2 hours respectively. When using Caltech10x the training times for these methods augment to 2 and 29 hours, respectively. The training time does not increase proportionally with the training data volume because the hard negative mining reads a variable amount of images to attain the desired quota of negative samples. This amount increases when a detector has less false positive mistakes.

5.1. Validation set experiments

Based on the results in table 2 we proceed to evaluate on Caltech10x the most promising configurations (filter type and number) from section 4. The results over the Caltech validation set are collected in table 3. We observe a clear overall gain from increasing the training data.

Interestingly with enough RandomFilters we can outperform the strong performance of LDCF-Ours. We also notice that the naïve Checkerboards outperforms the manual design of InformedFilters.

Filters type	# filters	Caltech MR	Caltech10x MR	ΔMR
ACF-Ours	1	50.2	39.8	10.4
LDCF-Ours	4	37.3	34.1	3.2
LDCF8	8	42.6	30.7	11.9
PcaForeground	8	41.6	28.6	13.0
RandomFilters	50	36.5	28.2	8.3
InformedFilters	50	30.3	26.6	3.7
Checkerboards	39	30.9	25.9	5.0
Checkerboards	61	32.9	24.4	8.5

Table 3: Effect of increasing the training set for different methods, quality measured on Caltech validation set (MR: log-average miss-rate).

6. Add-ons

Before presenting the final test set results of our "core" method (section 7), we review some "add-ons" based on the suggestions from [2]. For the sake of evaluating complementarity, comparison with existing methods, and reporting the best possible detection quality, we consider extending our detector with context and optical flow information.

Context Context is modelled via the 2Ped re-scoring method of [28]. It is a post-processing step that merges our detection scores with the results of a two person DPM [11] trained on the INRIA dataset (with extended annotations). In [28] the authors reported an improvement of ∼5 pp (percent points) on the Caltech set, across different methods. In [2] an improvement of 2.8 pp is reported over their strong detector (SquaresChnFtrs+DCT+SDt 25.2% MR). In our experiments however we obtain a gain inferior to 0.5 pp. We have also investigated fusing the 2Ped detection results via a different, more principled, fusion method [43]. We observe consistent results: as the strength of the starting point increases, the gain from 2Ped decreases. When reaching our Checkerboards results, all gains have evaporated. We believe that the 2Ped approach is a promising one, but our experiments indicate that the used DPM template is simply too weak in comparison to our filtered channels.

Optical flow Optical flow is fed to our detector as an additional set of 2 channels (not filtered). We use the implementation from SDt [30] which uses differences of weakly stabilized video frames. On Caltech, the authors of [30] reported a ∼7 pp gain over ACF (44.2% MR), while [2] reported a ∼5 pp percent points improvement over their strong baseline (SquaresChnFtrs+DCT+2Ped 27.4% MR). When using +SDt our results are directly comparable to Katamari [2] and SpatialPooling+ [29] which both use optical flow too.
Using our stronger Checkerboards results SDt provides a 1.4 pp gain. Here again we observe an erosion as the starting point improves (for confirmation, reproduced

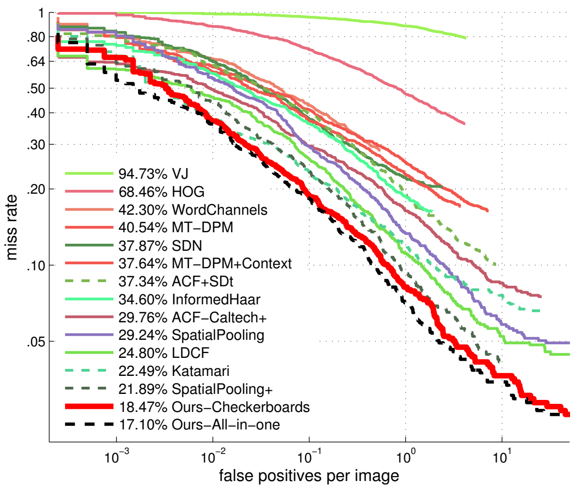


Figure 4: Some of the top quality detection methods on the Caltech test set.

the ACF+SDt results [30], 43.9% → 33.9% MR). We name our Checkerboards+SDt detector All-in-one.

Our filtered channel features results are strong enough to erode existing context and flow features. Although these remain complementary cues, more sophisticated ways of extracting this information will be required to further progress in detection quality.

It should be noted that despite our best efforts we could not reproduce the results from neither 2Ped [28] nor SDt [30] on the KITTI dataset (in spite of its apparent similarity to Caltech). Effective methods for context and optical flow across datasets have yet to be shown. Our main contribution remains on the core detector (only HOG+LUV features over local sliding window pixels in a single frame).

7. Test set results

Having done our exploration of the parameters space on the validation set, we now evaluate the most promising methods on the Caltech and KITTI test sets.

Caltech test set Figures 5 and 4 present our key results on the Caltech test set. For proper comparison, only methods using the same training set should be compared (see [2, figure 3] for a similar table comparing 50+ previous methods). We include for comparison the baselines mentioned in section 2.2, Roerei [1] the best known method trained without any Caltech images, MT-DPM [44] the best known method based on DPM, and SDN [22] the best known method using convolutional neural networks. We also include the top performers Katamari [2] and SpatialPooling+ [29]. We mark as "Caltech$N\times$" both the Caltech10x training set and the one used in LDCF [25] (see section 5).

KITTI test set Figure 6 presents the results on the KITTI test set ("moderate" setup), together with all other reported

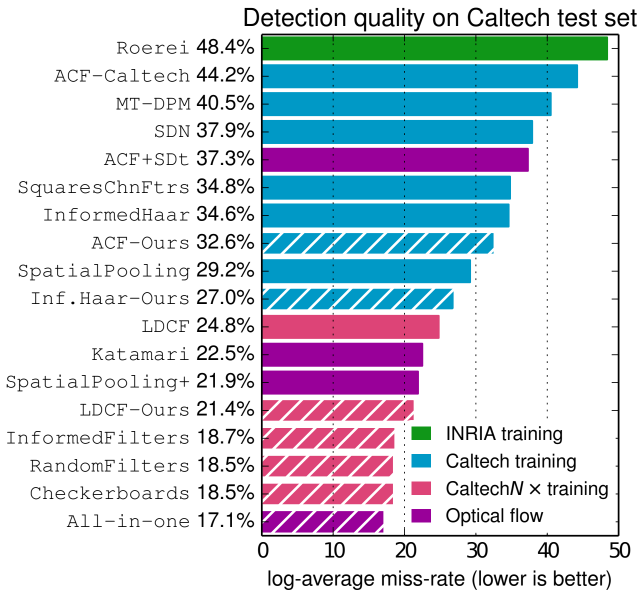


Figure 5: Some of the top quality detection methods for Caltech test set (see text), and our results (highlighted with white hatch). Methods using optical flow are trained on original Caltech except our `All-in-one` which uses Caltech10x. Caltech$N\times$ indicates Caltech10x for all methods but the original `LDCF` (see section 2.1).

methods using only monocular image content (no stereo or LIDAR data). The KITTI evaluation server only recently has started receiving submissions (14 for this task, 11 in the last year), and thus is less prone to dataset over-fitting.
We train our model on the KITTI training set using almost identical parameters as for Caltech. The only change is a subtle pre-processing step in the HOG+LUV computation. On KITTI the input image is smoothed (radius 1 pixel) before the feature channels are computed, while on Caltech we do not. This subtle change provided a $\sim$4 pp (percent points) improvement on the KITTI validation set.

7.1. Analysis

With a $\sim$10 pp (percent points) gap between `ACF/InformedHaar` and `ACF/InformedHaar-Ours` (see figure 5), the results of our baselines show the importance of proper validation of training parameters (large enough model size and negative samples). `InformedHaar-Ours` is the best reported result trained with Caltech1x.

When considering methods trained on Caltech10x, we obtain a clear gap with the previous best results (`LDCF` 24.8% MR $\rightarrow$ `Checkerboards` 18.5% MR). Using our architecture and an adequate number of filters one can obtain strong results using only HOG+LUV features. The amongst the options we considered the filter type seems not critical, in our experiments `Checkerboards4x3` reaches the best performance given the available training data. `RandomFilters` reaches the same result, but requires

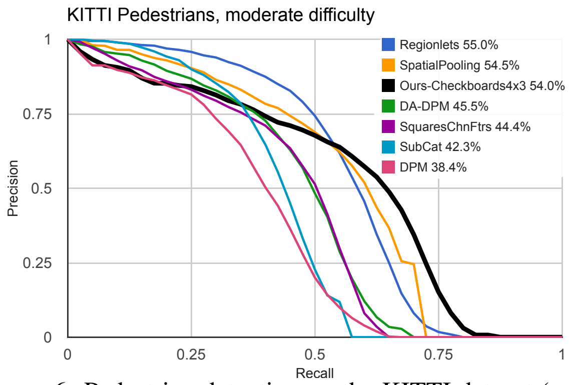


Figure 6: Pedestrian detection on the KITTI dataset (using images only).

training and merging multiple models.

Our results cut by half miss-rate of the best known `convnet` for pedestrian detection (`SDN [22]`), which in principle could learn similar low-level features and their filtering.

When adding optical flow we further push the state of the art and reach 17.1% MR, a comfortable $\sim$5 pp improvement over the previous best optical flow method (`SpatialPooling+`). This is the best reported result on this challenging dataset.

The results on the KITTI dataset confirm the strength of our approach, reaching 54.0% AP, just 1 pp below the best known result on this dataset. Competing methods (`Regionlets` [42] and `SpatialPooling` [29]) both use HOG and additional LBP and covariance features, as well as an intermediate max-pooling step. Adding these remains a possibility for our system. Our results also improve over methods using LIDAR+Image, such as `Fusion-DPM` [31] (46.7% AP, not included in figure 6 for clarity).

8. Conclusion

Through this paper we have shown that the seemingly disconnected methods `ACF`, `(Squares)ChnFtrs`, `InformedHaar`, and `LDCF` can be all put under the filtered channel features detectors umbrella. We have systematically explored different filter banks for such architecture and shown that they provide means for important improvements for pedestrian detection. Our results indicate that HOG+LUV features have not yet saturated, and that competitive results (over Caltech and KITTI datasets) can be obtained using only them. When optical flow information is added we set the new state of art for the Caltech dataset, reaching 17.1% MR (93% recall at 1 false positive per image).

In future work we plan to explore how the insights of this work can be exploited into a more general detection architecture such as convolutional neural networks.

References

[1] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *CVPR*, 2013. 1, 2, 6, 7

[2] R. Benenson, M. Omran, J. Hosang, , and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV, CVRSUAD workshop*, 2014. 1, 2, 3, 7

[3] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? In *arXiv*, 2014. 5

[4] A. D. Costea and S. Nedevschi. Word channel based multiscale pedestrian detection without image resizing and using only one classifier. In *CVPR*, June 2014. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[6] L. Dang, B. Bui, P. D. Vo, T. N. Tran, and B. H. Le. Improved hog descriptors. In *KSE*, 2011. 2

[7] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *PAMI*, 2014. 2, 3, 5

[8] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. 1, 2, 6

[9] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *TPAMI*, 2011. 1, 2, 3

[10] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *PAMI*, 2009. 1

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 2, 7

[12] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 2000. 4

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 2, 3

[14] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *PAMI*, 2010. 1

[15] Y. Goto, Y. Yamauchi, and H. Fujiyoshi. Cs-hog: Color similarity-based hog. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision*, 2013. 2

[16] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, 2015. 2

[17] C. Hou, H. Ai, and S. Lao. Multiview pedestrian detection based on vector boosting. In *ACCV*. 2007. 2

[18] F. Khan, R. Anwer, J. van de Weijer, A. Bagdanov, M. Vanrell, and A. Lopez. Color attributes for object detection. In *CVPR*, 2012. 2

[19] R. Khan, J. Van de Weijer, F. S. Khan, D. Muselet, C. Ducottet, and C. Barat. Discriminative color descriptors. In *CVPR*, 2013. 2

[20] I. Laptev. Improving object detection with boosted histograms. *Image and Vision Computing*, 2009. 2

[21] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 2

[22] P. Luo, Y. Tian, X. Wang, and X. Tang. Switchable deep network for pedestrian detection. In *CVPR*, 2014. 7, 8

[23] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014. 2

[24] K. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 5

[25] W. Nam, P. Dollár, and J. H. Han. Local decorrelation for improved detection. In *NIPS*, 2014. 1, 2, 3, 5, 6, 7

[26] W. Nam, B. Han, and J. Han. Improving object localization using macrofeature layout selection. In *ICCV, Visual Surveillance Workshop*, 2011. 2

[27] P. Ott and M. Everingham. Implicit color segmentation features for pedestrian and object detection. In *CVPR*, 2009. 2

[28] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. In *CVPR*, 2013. 7

[29] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *ECCV*, 2014. 1, 2, 3, 7, 8

[30] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring weak stabilization for motion feature extraction. In *CVPR*, 2013. 2, 7

[31] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining RGB and dense LIDAR data. In *IROS*, 2014. 8

[32] D. Ramanan. Using segmentation to verify object hypotheses. In *CVPR*, 2007. 2

[33] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. In *CVPR*, 2013. 2

[34] A. Satpathy, X. Jiang, and H.-L. Eng. Human detection by quadratic classification on subspace of extended histogram of gradients. *IEEE Transactions on Image Processing*, 2014. 2

[35] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013. 2

[36] Y. Socarras, D. Vazquez, A. Lopez, D. Geronimo, and T. Gevers. Improving hog with image segmentation: Application to human detection. In *Advanced Concepts for Intelligent Vision Systems*. 2012. 2

[37] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *CVPR*, 2015. 2

[38] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *PAMI*, 2008. 2

[39] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 2005. 2

[40] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR*, 2010. 2

[41] X. Wang, X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, 2009. 2

[42] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*. IEEE, 2013. 1, 2, 3, 8

[43] P. Xu, F. Davoine, and T. Denoeux. Evidential combination of pedestrian detectors. In *BMVC*, 2014. 7

[44] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li. Robust multi-resolution pedestrian detection in traffic scenes. In *CVPR*, 2013. 7

[45] S. Zhang, C. Bauckhage, and A. B. Cremers. Informed haar-like features improve pedestrian detection. In *CVPR*, 2014. 1, 2, 3, 4