Efficient Detection for Spatially Local Coding

Sancho McCann and David G. Lowe

University of British Columbia

Abstract. In this paper, we present an efficient detector for the Spatially Local Coding (SLC) object model. SLC is a recent, high performing object classifier that has yet to be applied in a detection (object localization) setting. SLC uses features that jointly code for both appearance and location, making it difficult to apply the existing approaches to efficient detection. We design an approximate Hough transform for the SLC model that uses a cascade of thresholds followed by gradient descent to achieve efficiency as well as accurate localization. We evaluate the resulting detector on the Daimler Monocular Pedestrian dataset.

1 Introduction

Spatially Local Coding (SLC) [1] has been recently proposed as an alternative way of including spatial information for object recognition. By using features that code for both appearance and location, SLC avoids the need to use fixed grids in the spatial pyramid model and uses a simple, whole-image region during the pooling stage. It outperforms modern variants of the spatial pyramid at equivalent model dimensionalities, and achieved better classification performance than all previous single-feature methods when tested on the Caltech 101 and 256 object recognition datasets [1].

Given SLC's high performance as a classifier, it is natural to adopt it for use as a detector to find the location of objects within images of natural scenes.

Our contribution in this exploratory paper is the design and evaluation of an efficient detector for Spatially Local Coding. Given the novel way that SLC includes location information, designing this detector is not a straight-forward application of an existing detector framework. We demonstrate the promise of this detector by evaluating on the Daimler monocular pedestrian dataset [2].

We start with a short review of previous detector work, then review the SLC model in detail, describe the design and optimization of our detector, and finish with evaluation and discussion.

2 Related work

There is a large diversity of approaches to object detection, and we review in this section a small sampling to highlight the range of methods that have shown success.

One family of approaches starts with local feature patches such as SIFT [3] and combines them in bags-of-words [4] or spatial pyramids [5] for classifications of image sub-windows. These have been applied to detection through the use of the spatial pyramid as an exemplar model [6].

Another line of work starts with larger-scale features representing the overall object appearance or parts. Examples here include histograms of oriented gradients (HOG) [7] and the deformable parts model [8].

Any classifier can be used as a detector by treating the detection problem as localized classification: sliding the classifier across the image at different scales and finding maxima of the classification function. However, this can be expensive, especially as one increases the resolution of the search space. More efficient alternatives has been proposed. Efficient sub-window search [9, 10] avoids exhaustive evaluation of every sub-window by using a branch-and-bound search.

Most related to our proposed method is the implicit shape model (ISM) [11], which uses a generalized Hough transform to vote for object centers. ISM's object model is generative, and its probabilistic Hough voting *is* part of the model. In our detector, the Hough voting (see Section 5) is only an approximation to the SLC model.

Higher order information such as object interrelationships [12] and tracking (in the case of video or image sequences) could also be leveraged to aid in detection performance, but these approaches are orthogonal to basic detector design.

Recent work on deep learning [13, 14] has demonstrated excellent classification performance when very large training sets and computational resources are used to learn diverse sets of features for recognition. When such large training sets are unavailable, there continues to be a need for systems that can use existing standard feature sets such as are explored in this paper.

3 Spatially local coding

We now review the Spatially Local Coding (SLC) classifier [1] with a focus on the aspects relevant to detection. We present SLC as a variant within the coding/pooling framework of Boureau *et al.* [15].

Given an image $\mathcal{I}$, let feature extraction be represented as:

$$\Phi(\mathcal{I}) : \mathcal{I} \mapsto \{(\phi_1, x_1, y_1), \ldots (\phi_{n_\mathcal{I}}, x_{n_\mathcal{I}}, y_{n_\mathcal{I}})\},$$

with feature i having a local appearance described by ϕ_i, and centered at (x_i, y_i).

Features are coded through a coding function $g((\phi_i, x_i, y_i))$. In bags-of-words [4] and spatial pyramid [5] approaches, coding functions code only the appearance portion of the descriptor ϕ_i, such that $g((\phi_i, x_i, y_i)) = \hat{g}(\phi_i)$. Any location information is included later, at the pooling stage.

SLC differs from previous coding/pooling methods in that it uses a coding function g that directly handles spatial locality, and uses a single, whole-image pooling region during the pooling stage. Instead of choosing $g(\phi_i, x_i, y_i) =$

$\hat{g}(\phi_i)$, SLC simultaneously codes ϕ_i and the location (x_i, y_i) by using a location-augmented descriptor: $\phi_{\mathbf{i}}^{(\lambda)} = [\phi_{i1}, \phi_{i2}, \ldots, \phi_{id}, \lambda x_i, \lambda y_i]$, where $\lambda \in \mathbb{R}$ is a location weighting factor giving the importance of the location in feature matching.

SLC chooses localized soft-assignment [16] for $\hat{g}$ to map each feature onto codewords. Let $\mathrm{NN}_{(\kappa)}(\phi_i)$ be the set of κ nearest neighbors to ϕ_i in a dictionary $\mathbf{D}^{(\lambda)}$ of location-weighted codewords. Then, the localized soft assignment coding is:

$$\hat{g}(\phi_i^{(\lambda)}) = \mathbf{u}_i = [u_{i1}, u_{i2}, \ldots, u_{ik}] : \tag{1}$$

$$u_{ij} = \frac{\exp(-\beta d(\phi_i^{(\lambda)}, \mathbf{D}_j^{(\lambda)})}{\sum_{a=1}^{k} \exp(-\beta d(\phi_i^{(\lambda)}, \mathbf{D}_a^{(\lambda)}))}$$

$$d(\phi_i^{(\lambda)}, \mathbf{D}_j^{(\lambda)}) = \begin{cases} \|\phi_i^{(\lambda)} - \mathbf{D}_j^{(\lambda)}\|^2 & \text{if } \mathbf{D}_j^{(\lambda)} \in \mathrm{NN}_{(\kappa)}(\phi_i) \\ \infty & \text{otherwise} \end{cases}$$

The final SLC histogram representation $\mathbf{h}$ of a subregion $\mathcal{S}$ is a max-pooling histogram [15, 17] of all $\mathbf{u}_i$ in that region:

$$\mathbf{h}_{\max} = [h_1, h_2, \ldots, h_k] \quad \text{where}$$
$$h_j = \max\{u_{ij} | (x_i, y_i) \in \mathcal{S}\} \tag{2}$$

The resulting histogram is used as the input layer to a linear SVM. [1] showed that a linear SVM obtained higher performance on the SLC model than a histogram intersection kernel SVM.

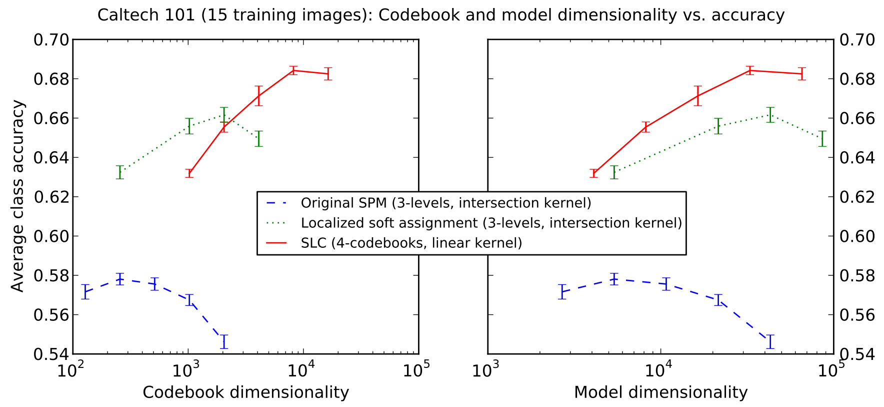


Fig. 1: [1] demonstrated that SLC performed better than the basic spatial pyramid model and better than more recent state-of-the-art refinements in the context of single-image classification (Caltech 101 and 256). Figure reproduced from [1].

In summary, SLC uses location-augmented feature vectors, localized soft-assignment coding, and a single, whole-image max-pooling region. SLC moves the task of maintaining spatial locality into the coding stage, whereas previously, this had been left for the pooling stage.

We avoid the issue of λ selection by use of the multi-level SLC variant as suggested by [1]. Multi-level SLC codes across several dictionaries at once, each with a different λ. Not only does this avoid having to fine-tune a parameter, McCann *et al.* [1] showed that a combination of several dictionaries with different λ gave better classification performance than any of the individual dictionaries alone.

4 Detection methods

This section reviews two methods that seek to avoid having to perform an exhaustive sliding window detection. A 320×240 image "contains more than one billion rectangular sub-images" [9]. As individual evaluations of the SLC classifier are relatively expensive, it is important to avoid unnecessary evaluations.

4.1 Efficient sub-window search

Efficient sub-window search (ESS) was presented by Lampert *et al.* [9, 10] as a way of effectively performing exhaustive search of all image sub-windows in time that is sub-linear relative to the number of possible sub-windows. ESS is a branch-and-bound algorithm that relies on efficiently computing an upper bound for the scores of sets of sub-windows.

The efficiency of the original ESS bound results from a one-time quantization from features into codewords and then accumulation of those codewords' linear SVM weights into integral histograms [9]. This step is not possible with SLC. Without committing to a particular sub-window reference frame, the quantization from a feature into an SLC codeword is not defined, because location relative to the sub-window is a component of the feature.

The alternative, feature-centric efficient sub-window search proposed by Lehmann *et al.* [18, 19] is also inadequate for SLC because the feature-to-codeword quantization still depends on a commitment to a particular detection window as a reference frame. For each possible sub-window, a different feature-to-codeword mapping takes place, rendering the bound of feature-centric ESS expensive to compute. An approximate bound similar to the approximation we make in Section 5 is possible, and we include an implementation with our code, but there are two reasons we decide against this approach. First, as observed by [18], the large number of extracted features results in a computation bottleneck. Second, we have observed that using the approximation from Section 5 when computing the bounds results in traversing many unfruitful paths through the branch-and-bound search space.

4.2 Hough transform

The detection framework we develop builds upon the Generalized Hough transform [11, 19]. While it lacks the theoretical guarantees of the efficient sub-window search, the Hough transform is compatible with approximations to SLC, and can still quickly suggest promising regions of the image over which to focus expensive evaluation of the full SLC model. It handles large numbers of features well (the voting phase scales linearly with the number of features). We are able to greatly speed up detection over the sliding window approach, without sacrificing performance.

We do not use a standard Hough transform. Our approach differs in two ways. First, we perform *max-pooling voting* prior to multiplying those intermediate votes by weights derived from the linear SVM. Second, our initial votes are based on an *approximation* to the SLC model. This is due to the way that SLC coding depends on committment to a particular detection hypothesis. It is not obvious that this approximation will result in useful peaks in the Hough space, and we evaluate the suitability of these peaks in Section 5.2.

After obtaining these preliminary hypotheses from the peaks in Hough space, we apply a cascade of thresholds and refinement to focus the SLC classifier on only the most promising regions. The design and optimization of this pipeline is described next.

5 The detection pipeline

5.1 Approximate SLC Hough transform

We follow the approach of Lehmann *et al.* [19] in considering hypothesis footprints. Each hypothesis stamps out a footprint over which evidence for an object detection is accumulated. However, instead of accumulating votes hypothesis-by-hypothesis as in a sliding window approach, the Hough transform has each feature cast a weighted vote for (or against) hypotheses consistent (or inconsistent) with that feature's occurrence.

The vote weights associated with each codeword are derived from the linear SVM weights. As in Lampert *et al.* [10], we re-write the linear SVM decision function as a sum of per-codeword weights. The SVM decision function is $f(\mathbf{h}) = \beta + \sum_i \alpha_i \langle \mathbf{h}, \mathbf{h}^{(i)} \rangle$, where $\mathbf{h}$ is the SLC histogram being classified, $\mathbf{h}^{(i)}$ are the training histograms, and α_i are the learned per-example SVM weights. We can extract per-codeword weights $w_j = \sum_i \alpha_i \mathbf{h}_j^{(i)}$, and re-write the decision function as:

$$f(\mathbf{h}) = \beta + \sum_j \mathbf{h}_j \cdot w_j \tag{3}$$

and drop the bias term β because only the relative scores matter.

Since SLC is based on max-pooling rather than sum-pooling, we need to perform the Hough transform in two phases. The first phase, where most of the work is done, involves building a max-pooling histogram at each bin in the discretized Hough space.

This phase occurs in a 5D Hough space: (s, a, x, y, c), with s being scale, represented by hypothesis window width, a being aspect ratio, x and y being the hypothesis center, and c being the codeword. See Figure 2 for a visualization.

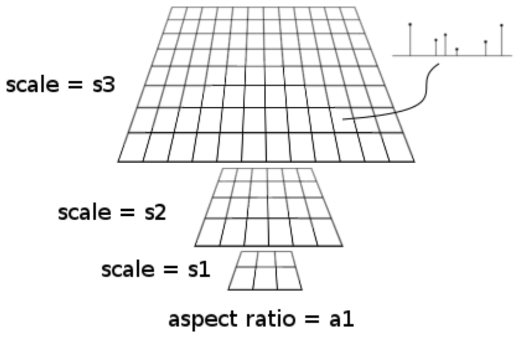


Fig. 2: For each $(a, s) \in \mathcal{A} \times \mathcal{S}$, we build an x, y grid of Hough bins, each of which tracks per-codeword votes. After max-pooling voting is complete, we collapse each histogram into a Hough score for each bin using Equation 3.

To determine the region over which a feature will cast a vote, we use a location distribution for each SLC codeword c: $(\mu^{(c)}, \sigma^2_{(c)})$ (with $\mu = (\mu_x, \mu_y) \in [-0.5, 0.5]^2$). The location distribution is only explicitly used during this approximate Hough transform step and is not part of the final SLC model. It is a means of approximating the region over which a given codeword is likely to contribute to the hypothesis footprint.

SLC quantizes using feature location relative to a reference frame. However, there is no such reference frame when quantizing features prior to voting in Hough space. Thus, we make the following approximation. We quantize using localized soft assignment [16] based solely on appearance information (we drop the λ in Equation 1). This maps a feature to the codewords that it *might* be matched to under SLC. We use this tentative matching based on appearance along with the learned location distributions to determine the bins in which to cast Hough votes.

Given feature center (f_x, f_y), an appearance-only soft-coding that results in non-zero weight for codeword c, and learned location distribution for that codeword, $(\mu^{(c)}, \sigma^2_{(c)})$, we compute for each $(a, s) \in \mathcal{A} \times \mathcal{S}$ the Hough bins that this features should vote in as follows:

$$\hat{\mu}_x = f_x - \mu_x^{(c)} \cdot s; \quad \hat{\mu}_y = f_y - \mu_y^{(c)} \cdot \frac{s}{a} \tag{4}$$

$$\hat{\sigma}_x = \sigma_x^{(c)} \cdot s; \quad \hat{\sigma}_y = \sigma_y^{(c)} \cdot \frac{s}{a} \tag{5}$$

Equations 4 and 5 invert the learned location distribution into Hough space at the appropriate scale (width) and aspect ratio. We update the max-pooling histogram for codeword c in all Hough bins within $3\hat{\sigma}$ of $\hat{\mu}$.

After all votes have been cast, we apply Equation 3 to each bin to turn the histograms into scores.

5.2 Optimizing Hough predictions

In this section, we coarsely optimize parameters of our approximate Hough transform on a subset of the VOC 2012 *car* category. We train on all non-truncated, non-difficult, non-occluded cars in the *train* portion of the training set, and test on all images in the *validation* set that include a car.

Our goal in using the Hough transform and cascade of refinements is to reduce the number of times we need to evaluate using the full model. First, we check that the approximation we use when computing the Hough votes is appropriate. Do the Hough scores reflect roughly the regions of the image that are more likely to contain the object of interest? We run a simple thresholding algorithm (Algorithm 1) for this check. Figure 3 shows that the Hough scores are meaningful. Almost every Hough bin has a score greater than -1, so the recall when thresholding at -1 is effectively the maximum recall we can achieve with a Hough transform at this bin density. We retain almost 100% of that recall and eliminate 60% of the bins from consideration by thresholding at zero. (Recall achieves a maximum of only 86% in this series of experiments due to our choice of search grid resolution and minimum scale. We make the same choices for the sliding window baseline we compare against later in Figure 5.)

Algorithm 1: Thresholding bins

```
Data: 4D Hough map $m$, threshold $t$
results ← {};
for bin $b \in m$ do
    if $b.score >= t$ then
        add b to results ;
    end
end
```

We can do better. Instead of selecting all Hough bins that pass a threshold, what if we select only local *peaks* in Hough space that pass the threshold? Figure 4 shows the result of running Algorithm 2 with a varying threshold. There are on average 2528 local Hough peaks in each image. This results in recall of 81% compared to exhaustive consideration of all candidate windows (86%), as shown at the left extreme of Figure 3). Again, by thresholding at zero, we remove even more windows from consideration without losing further recall.

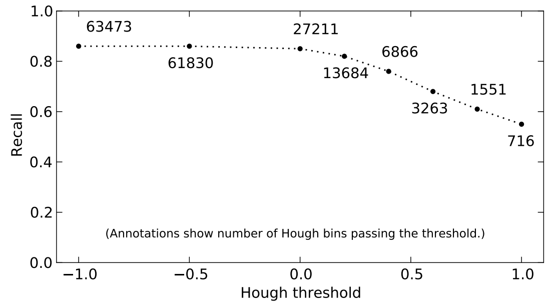


Fig. 3: We are able to maintain high recall while eliminating more than half of the candidate windows by discarding Hough bins receiving negative scores. This experiment runs Algorithm 1 with a varying threshold. (Recall in this experiment only reaches 86% due to our choice of grid resolution and minimum scale.)

Algorithm 2: Thresholding peaks

```
Data: 4D Hough map $m$, threshold $t$
results ← {};
for bin $b \in m$ do
    if $b.score >= t$ and IsLocalMax(b) then
        add b to results ;
    end
end
```

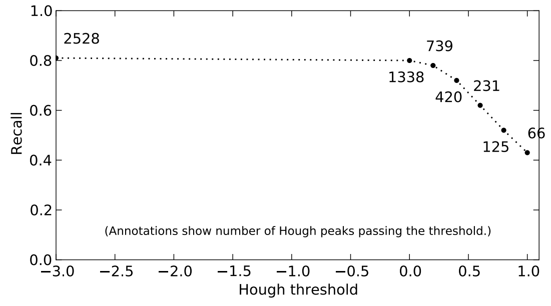


Fig. 4: By focusing only on peaks, we eliminate many candidate locations. Thresholding at zero further reduces the number of candidate windows without reducing recall. This experiment runs Algorithm 2 with a varying threshold.

This is additional evidence that the signal from our Hough transform, even though based on an approximation of our full model, is meaningful. The score well separates candidate regions from background.

Many windows still remain, and many candidates overlap significantly with one-another. We perform non-maximum suppression at this point to produce the final predictions based solely on the Hough scores (Algorithm 3). This results in poor detection performance (13% average precision) compared to the sliding window detector.

Algorithm 3: Predict from thresholded peaks

```
Data: 4D Hough map $m$, threshold $t$
results ← {};
for bin $b \in m$ do
    if $b.score >= t$ and IsLocalMax(b) then
        add b to results ;
    end
end
NonMaximumSuppression(results)
```

The discrepancy can be explained by the Hough scores being only an approximation to the full model. The scores of the candidate windows at this stage are not as accurate as what the full model would provide, and they aren't as precisely localized. Even if we did re-evaluate each of these peaks with the true model (Algorithm 4), the fact that they aren't well-localized means that those scores will still not be as informative, boosting performance to only 19% average precision. Figure 5 (*Hough peaks* and *Re-scored Hough peaks*) shows the performance of these two methods.

Algorithm 4: Predict from re-scored peaks

```
Data: 4D Hough map $m$, threshold $t$
results ← {};
for bin $b \in m$ do
    if $b.score >= t$ and IsLocalMax(b) then
        b.score = EvaluateSLC(b);
        if $b.score >= t$ then
            add b to results ;
        end
    end
end
NonMaximumSuppression(results)
```

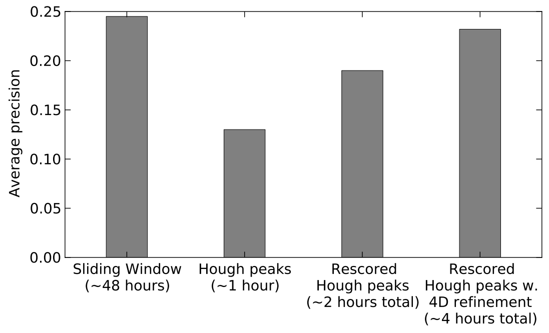


Fig. 5: By re-scoring the Hough peaks, and then refining their locations using gradient descent, we retrieve the performance of the sliding window detector while gaining a large speedup. All methods were subject to non-maximum suppression to eliminate overlapping predictions.

One last step is necessary to nearly recover the performance of the sliding window approach. After re-scoring the Hough peaks, so that we know their score under the true model, we again discard peaks with negative scores, suppress strong overlaps, and finally refine the remaining peaks using a gradient descent procedure [19]. Our gradient descent uses a finite difference approximation of the gradient, followed by line search. We simultaneously refine the (x, y) location, the aspect ratio, and scale until we reach a local maximum.

Algorithm 5: Full detection pipeline

```
Data: 4D Hough map $m$, threshold $t$
results ← {};
for bin $b \in m$ do
    if $b.score >= t$ and IsLocalMax(b) then
        b.score = EvaluateSLC(b);
        if $b.score >= t$ then
            add b to results ;
        end
    end
end
NonMaximumSuppression(results);
for $b \in results$ do
    b ← GradientDescentSLC(b);
end
NonMaximumSuppression(results);
```

As a result, we consider a much finer set of potential candidate windows around the peaks than the sliding window approach does, but only evaluate a small number of windows in total using the full SLC model.

Now that we have largely recovered the performance of the brute force sliding window detector using a much faster approximation, we turn our attention to the design of the training phase.

5.3 Mining hard negative examples

For the above experiments, we used a model that was trained with a single pass through the training set, collecting all non-difficult, non-truncated, non-occluded positive examples and 10x that many negative examples selected randomly from regions of the training data that did not overlap with any positive example.

A technique used by many others in the past [20, 8, 7] is to mine the training data for hard negative examples: negative windows that the classifier erroneously predicts as belonging to the class. Walk *et al.* [20] observe that without hard negative selection, performance is extremely sensitive to the randomness in selecting examples for the negative training set, and that at least 2 re-training rounds are required in order to reach full performance using HOG + linear SVM. We follow this practice by running the detector over the training set, and adding the highest scoring false positives into our negative training set. Figure 6 shows the effect of hard negative mining on the car class. The effect of additional hard negative examples is clear.

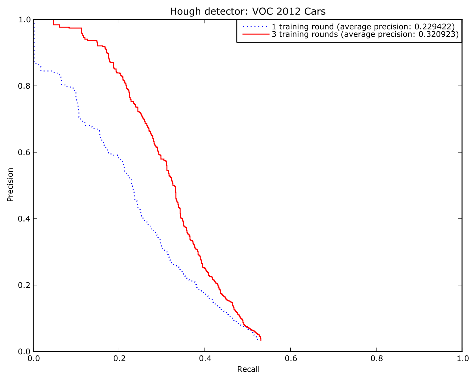


Fig. 6: Three rounds of training with hard negative mining results in a significant improvement in classifier accuracy.

6 Evaluation

We evaluate our detector on the Daimler monocular pedestrian dataset [2]. This dataset presents a challenging real-world problem and has been used to test and compare among other methods. Additionally, its training data is consistant in its cropping, alignment, and aspect ratio, which allows us to focus solely on the localization performance of our search strategy, and not introduce the confounding factors of handling of widely varying aspect ratio or viewpoint.

The Daimler pedestrian training set comprises 15660 pedestrian training examples, each presented in a 48×96 cropped image, and 6744 images containing no pedestrians. We extract multi-scale SIFT and build three SLC dictionaries with $\lambda = \{0.0, 1.5, 3.0\}$. We perform three training rounds. During the first training round, we extract random negative training windows such that the number of negatives is $7\times$ the number of positives (chosen to fit within memory constraints). We train a linear SVM using k-fold cross validation for selection of the hyper-parameters. Before re-training in the next round, we run our detector across all 6744 negative training images to mine for the most difficult (most highly scored) negative examples, and replace the 25% easiest examples from the previous training round with an equivalent number of new difficult examples.

For testing, we followed the evaluation protocol described by Dollár *et al.* [21]. We evaluate against all fully-visible, labeled pedestrians, ignoring bicyclists, motorcyclists, pedestrian groups, and partially visible pedestrians. Detections and failed detections of ignored annotations neither count for or against our detector. As in [21], we standardize the aspect ratio of all ground truth boxes to 0.41 by retaining the annotated height and adjusting the width of the ground truth bounding box to 0.41 times the height. We up-scale the test images by 1.6 to allow detection of the smaller scale pedestrians.

Enzweiler *et al.* [2] initially proposed reporting recall vs. false positives per frame. However, Dollar *et al.* [21] point out that false positives per image is a more useful measure of false positives in this setting. They observe that the difference is in whether one is evaluating the performance of the classifier underlying the detector, or evaluating the performance of the entire detector pipeline as a complete system. "Choices made in converting a binary classifier to a detector, including choices for spatial and scale stride and non-maximal suppression, influence full image performance." [21]. We follow them in reporting miss-rate vs. false positives per image. This is similar to reporting precision vs. recall as in the Visual Object Classes Challenge, but false positives per images is perhaps more important in an automotive pedestrian detection setting. Figure 7 shows our results compared against the single-feature, non-motion results reported by [21].

Our detector outperforms HOG [7], histogram intersection kernel SVM [22], and an early version of the deformable parts model [23] throughout a wide range of false-positive rates. The one method that has an advantage is the latent SVM [8] that learns explicit subparts. The recall of our detector does saturate at higher false positive rates, but Dollár *et al.* identify the region between 10^{-2} and 10^{0} false positives per window as the region most relevant for comparison. This is supported by Hussein *et al.* [24] who also use a score that focuses more on the region of the curve with low false alarms. They say, "this is useful since in many applications we are more interested in the low false alarm rate range". Nevertheless, the saturation of recall in our detector is a point for future investigation. We suspect this is due to some of the true detections not being covered by the initial set of Hough peaks, and our refinement phase is not able to recover from these poor local maxima. There is obvious room for improvement, but we believe these results demonstrate that application of the Spatially Local Coding feature to the detection problem is a fruitful area for future research.

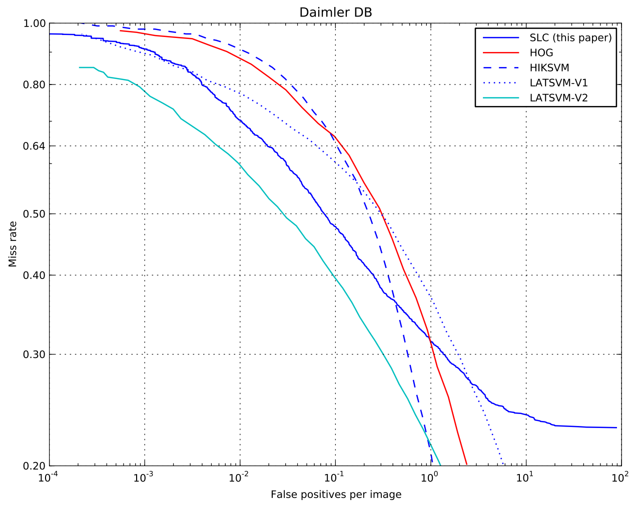
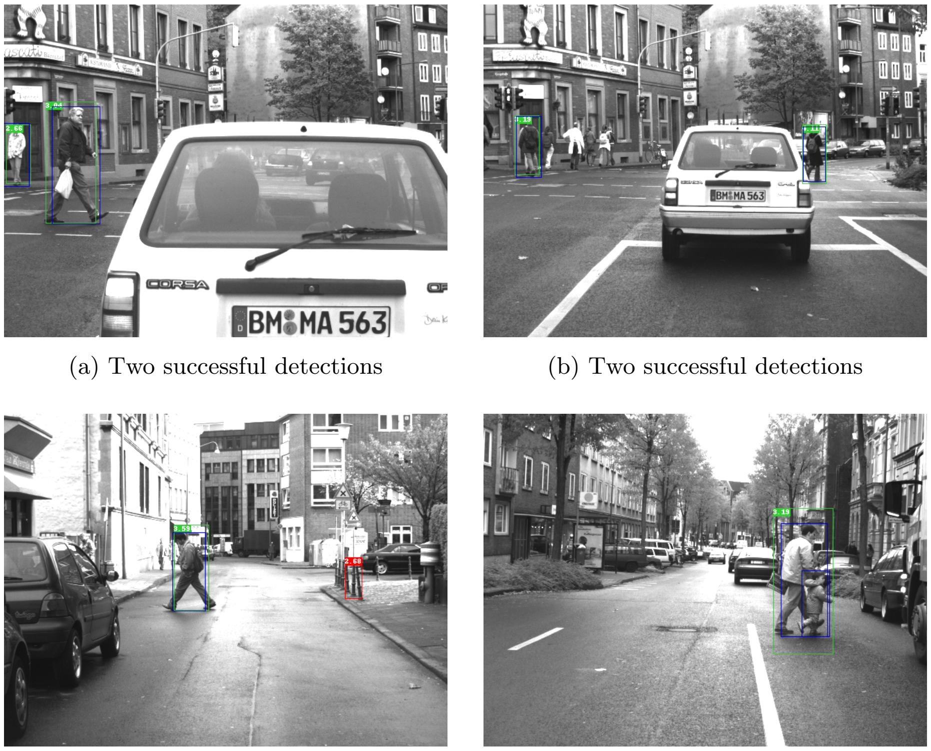


Fig. 7: Results on the Daimler monocular pedestrians dataset. (The data is taken from [21]. We have extracted the performance curves for the single-feature, non-motion methods that they evaluate.)

Figure 8 shows typical output of our detector. Common error cases include reporting false positives on regions containing vertical structures and merging two pedestrians into a single detection.

7 Conclusion

We've engineered a method of localization using Spatially Localized Features through use of a Hough transform approximation, followed by non-maximum suppression and gradient descent refinement. The approach was demonstrated on the widely used Daimler monocular pedestrian dataset, achieving a good level of detection accuracy. We believe this demonstrates that further research towards using SLC in the detection context is warranted.



(a) Two successful detections

(b) Two successful detections

(c) A success, and a false positive

(d) A missed detection (the child)

Fig. 8: Examples of typical detections made by our detector on Daimler monocular pedestrian dataset. We've displayed detections above the threshold associated with the equal error rate. The numbers attached to each detection report the SLC score. True positives are outlined in green. Ground truth annotations are outlined in blue. False positives are outlined in red.

There is considerable scope for further research to extend this approach. One component of the current pipeline needing improvement is in the selection of Hough peaks. Initializing the gradient descent procedure from a broader selection of locations would lead to improved recall. There is also considerable scope for expanding the set of features being used and location weights to achieve higher detection performance.

References

1. McCann, S., Lowe, D.G.: Spatially local coding for object recognition. In: ACCV. (2012) 204–217
2. Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: survey and experiments. PAMI **31** (2009) 2179–95

3. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV **60** (2004) 91–110
4. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV. (2004)
5. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: CVPR. (2006)
6. Chum, O., Zisserman, A.: An exemplar model for learning object classes. In: CVPR. (2007)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
8. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part Based Models. PAMI **32** (2009) 1627–1645
9. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR. (2008)
10. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Efficient subwindow search: a branch and bound framework for object localization. PAMI **31** (2009) 2129–42
11. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. IJCV **77** (2008) 259–289
12. Wohlhart, P., Donoser, M., Roth, P., Bischof, H.: Detecting partially occluded objects with an implicit shape model random field. In: ACCV. (2013)
13. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
14. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. arXiv:1409.4842 [cs.CV] (2014)
15. Boureau, Y.L., Bach, F., LeCun, Y., Ponce, J.: Learning mid-level features for recognition. In: CVPR. (2010)
16. Liu, L., Wang, L., Liu, X.: In Defense of Soft-assignment Coding. In: ICCV. (2011)
17. Yang, J., Yu, K., Gon, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. (2009)
18. Lehmann, A., Van Gool, L., Leibe, B.: Feature-Centric Efficient Subwindow Search. In: CVPR. (2009)
19. Lehmann, A., Leibe, B., van Gool, L.: PRISM: Principled implicit shape model. In: British Machine Vision Conference. (2009)
20. Walk, S., Majer, N., Schindler, K., Schiele, B.: New features and insights for pedestrian detection. In: CVPR. (2010)
21. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: an evaluation of the state of the art. PAMI **34** (2012) 743–61
22. Maji, S., Berg, A., Malik, J.: Classification using Intersection Kernel Support Vector Machines is Efficient. In: CVPR. (2008)
23. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multi-scale, deformable part model. In: CVPR. (2008)
24. Hussein, M., Porikli, F., Davis, L.: A comprehensive evaluation framework and a comparative study for human detectors. IEEE Transactions on Intelligent Transportation Systems **10** (2009)