Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection

Joseph J. Lim
Massachusetts Inst. of Technology
lim@csail.mit.edu

C. Lawrence Zitnick
Microsoft Research
larryz@microsoft.com

Piotr Dollár
Microsoft Research
pdollar@microsoft.com

Abstract

*We propose a novel approach to both learning and detecting local contour-based representations for mid-level features. Our features, called **sketch tokens**, are learned using supervised mid-level information in the form of hand drawn contours in images. Patches of human generated contours are clustered to form sketch token classes and a random forest classifier is used for efficient detection in novel images. We demonstrate our approach on both top-down and bottom-up tasks. We show state-of-the-art results on the top-down task of contour detection while being over 200× faster than competing methods. We also achieve large improvements in detection accuracy for the bottom-up tasks of pedestrian and object detection as measured on INRIA [5] and PASCAL [10], respectively. These gains are due to the complementary information provided by sketch tokens to low-level features such as gradient histograms.*

1. Introduction

For visual recognition, mid-level features provide a bridge between low-level pixel-based information and high-level concepts, such as object and scene level information. Effective mid-level representations abstract low-level pixel information useful for later classification while being robust to irrelevant and noisy signals. Mid-level features may be hand-designed [5, 17] or learned with [16, 33] or without supervision [15, 40]. They serve as the foundation of both bottom-up processing, e.g. object detection, and top-down tasks, such as contour classification [14] or pixel-level segmentation [37] from object class information.

The use of edge information was a popular early approach to designing mid-level features [19]. This was motivated in part by the fact that humans can easily interpret line drawings and sketches [9]. Early edge detectors [3] were used to find more complex shapes such as junctions [22], straight lines and curves [8], and were applied to object recognition [34], structure from motion [32], tracking [31], and 3D shape recovery [25].

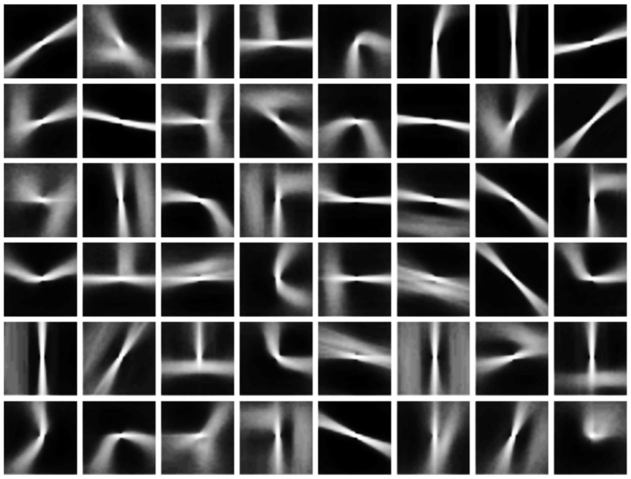


Figure 1. Examples of sketch tokens learned from hand drawn sketches represented using their mean contour structure. Notice the variety and richness of the sketch tokens.

In this work, we propose a novel approach to both learning and detecting local edge-based mid-level features, and demonstrate their effectiveness for both bottom-up and top-down tasks. Our features, called *sketch tokens*, capture local edge structure. The classes of sketch tokens range from standard shapes such as straight lines and junctions to richer structures such as curves and sets of parallel lines (Fig. 1).

Given the vast number of potential local edge structures, we must select an informative subset to represent by the sketch tokens. We propose a novel approach to defining token classes using *supervised mid-level* information, unlike previous approaches that use hand-defined classes [39], high-level supervision [16], or unsupervised information [15]. The supervised mid-level information is obtained from human labeled edges in natural images [1]. This data is both easy to collect and has the potential to generalize, since it is not object-class specific [21, 23]. Patches centered on contours are extracted from the hand drawn sketches and clustered to form a set of token classes. This results in a diverse, representative set of sketch tokens. We typically utilize a few hundred tokens, which captures a majority of the commonly occurring edge structures (Fig. 1).

Our goal is to efficiently predict the occurrence of sketch tokens given an input color image. We propose a data driven approach that classifies each image patch with a token label given a collection of low-level features including oriented gradient channels [7], color channels, and self-similarity channels [26]. The token class assignments resulting from clustering the patches of hand drawn contours provide our ground truth labels for training. We solve this large multi-class problem using random decision forests [2, 4]. The result is an efficient approach that can compute per-pixel token labelings in about *one second* per image.

We demonstrate the advantages of our mid-level sketch tokens on both top-down and bottom-up tasks. We show top-down results using our mid-level features to locate pixel-level contours. Using standard datasets [1], we show state-of-the-art results, while boosting efficiency over 200 fold. We explore the two bottom-up tasks of pedestrian detection and object detection. Results show a large reduction in error rate over previous state-of-the-art techniques on the INRIA pedestrian dataset [5]. We additionally show our mid-level features provide complementary information to the Histogram of Oriented Gradients descriptor [5] on the challenging PASCAL object detection dataset [10].

1.1. Related Work

Many of the most commonly used gradient and edge-based features are hand designed, such as SIFT [17] or HOG [5]. While numerous detection approaches [5, 11, 7] utilize such features directly, other papers learn edge-based features [27, 38, 21] or class-specific edges [23, 18] using object level supervision. Representation based on regions [13] are also popular. An alternative approach is to learn representations directly from pixels via deep networks, either without supervision [15, 40] or using object level supervision [16]. Learned features in these approaches resemble edge filters in early layers and more complex structures in deeper layers [40]. Our approach to learning a contour-based representation differs in that we inject *mid-level* supervision to learn a *universal* set of sketch tokens.

Data driven edge detection has also been popular. Dollár *et al*. [6] cast edge detection as a binary classification problem and used human labeled edges to train a binary patch edge classifier. This was extended by [41] who used contextual and shape information to refine the edge maps. [39] applied a similar approach to detect 17 unique local edge structures including edges, junctions and corners. Instead of training local detectors directly, Ren and Bo [24] learn a patch representation through sparse coding and measure local gradients of the resulting codes. In contrast, our work aims to learn a richer local representation that captures considerably more mid-level image information.

Algorithmically, our use of decision forests to densely classify image patches is inspired by Shotton *et al*.'s work on texton forests [29, 30]. The key difference is that we are learning sketch tokens as opposed to a semantic segmentation. While we utilize fairly standard low-level features [7, 26] and a well known classification algorithm [2, 4], the novelty of our approach is in the definition and use of sketch tokens to effectively encode local image structure.

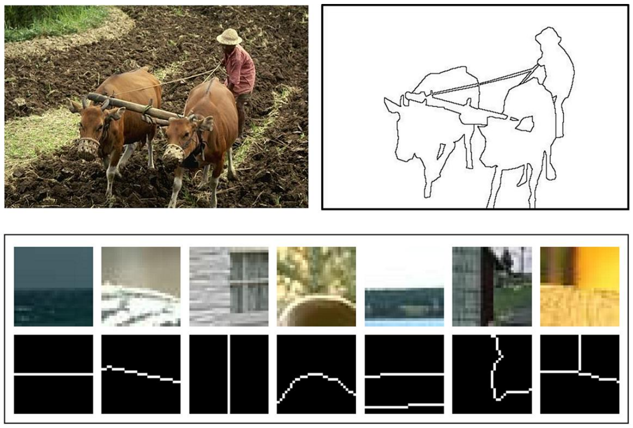


Figure 2. (Top) Example image and corresponding hand drawn sketch. (Bottom) Example image patches and their corresponding hand drawn contours.

2. Sketch Tokens

In this section we describe our approach to learning token classes from hand drawn sketches and for detecting the tokens in novel images. We show how to utilize these tokens for contour and object detection in the following sections. We begin by describing how we define our token classes.

2.1. Defining sketch token classes

Our goal is to define a set of token classes that represent the wide variety of local edge structures that may exist in an image. These include straight lines, t-junctions, y-junctions, corners, curves, parallel lines, etc. We propose an approach for discovering these classes using human-generated image sketches [1], see Figure 2.

Let us assume we have a set of images I with a corresponding set of binary images S representing the hand drawn contours from the sketches. The sketches are generated by asking human subjects to "Divide each image into pieces, where each piece represents a distinguished thing in the image." Further details on the generation of the dataset can be found in [20].

We define the set of sketch token classes by clustering patches s extracted from the binary images S. Each patch s_j extracted from image S_i has a fixed size of 35 × 35 pixels in our experiments. Furthermore, only patches that contain a labeled contour at the center pixel are used (there are approximately two million such patches in the training set of [20]). To provide invariance to slight shifts in edge placement, Daisy descriptors [36] are computed on the binary contour labels contained in s_j. Clustering is performed

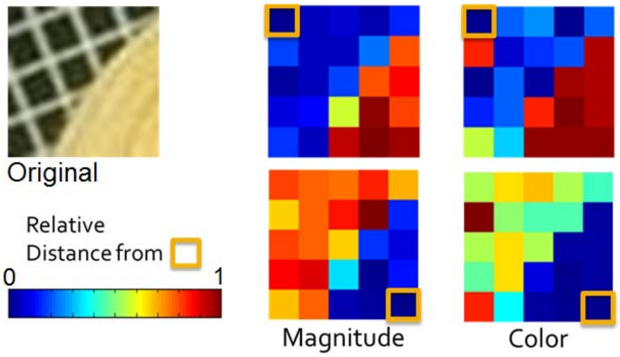


Figure 3. Illustration of the self-similarity features: The $L1$ distance $\sum_k |f_{ijk}|$ from the anchor cell (yellow box) to the other 5×5 cells are shown for color and gradient magnitude channels. The original patch is shown to the left.

on the descriptors using the K-means algorithm. We use $k = 150$ clusters in our experiments except where otherwise noted. Example cluster means are illustrated in Figure 1. Notice the variety of the sketch tokens, ranging from straight lines to more complex structures.

2.2. Detecting sketch tokens

Given a set of sketch token classes, we wish to detect their occurrence in color images. We detect the token classes with a learned classifier. As input, features are computed from color patches x extracted from the training images I. Ground truth class labels are supplied by the clustering results described above if the patch is centered on a contour in the hand drawn sketches S, otherwise the patch is assigned to the background or "no contour" class. The input features extracted from the color image patches x used by the classifier are described next.

2.2.1 Feature extraction

For feature extraction, we use an approach inspired by Dollár *et al.* [7] and compute multiple feature channels per image where each channel has the same size as the input image and captures a different facet of information. Two types of features are then employed: features directly indexing into the channels and self-similarity features.

Our channels are composed of color, gradient, and oriented gradient information in a patch x_i extracted from a color image. Three color channels are computed using the CIE-LUV color space. We compute several normalized gradient channels that vary in orientation and scale [7, 5, 17]. Three gradient magnitude channels are computed with varying amounts of blur (we use Gaussian blurs with $\sigma = 0$, 1.5 and 5 pixels). Additionally, the gradient magnitude channels at $\sigma = 0$ and $\sigma = 1.5$ are split based on orientation to create four additional channels each for a total of eight oriented magnitude channels. Finally all channels are post-blurred with a $\sigma = 1$ Gaussian. Pixels in the resulting channels serve as the first type of feature for our classifier.

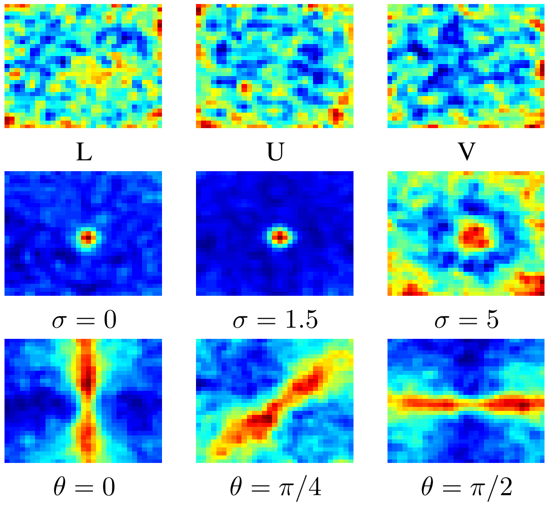


Figure 4. Frequency of example features being selected by the random forest: (first row) color channels, (second row) gradient magnitude channels, (third row) selected orientation channels.

The second type of feature used by our method is based on self-similarity [26]. It is well known that contours not only occur at intensity or color edges, but also at texture boundaries. The self-similarity features capture the portions of an image patch that contain similar textures based on color or gradient information. We compute texture information on a $m \times m$ grid over the patch, with $m = 5$ yielding 7×7 cells for 35×35 patches. For channel k and grid cells i and j, we define the self-similarity feature f_{ijk} as:

$$f_{ijk} = s_{jk} - s_{ik}, \tag{1}$$

where s_{jk} is the sum of grid cell j in channel k. An illustration of self-similarity features is shown in Fig. 3.

Since $f_{ijk} = -f_{jik}$ and $f_{iik} = 0$, the total number of self similarity features per channel is given by $\binom{m \cdot m}{2}$. For $m = 5$ this yields 300 features per channel. For computational efficiency, the sums s_{jk} over an entire image can be computed efficiently by convolving each channel by a box filter of width equal to the cell size.

In summary, we utilize 3 color channels, 3 gradient magnitude channels, and 8 oriented gradient channels for a total of 14 channels. For a 35×35 patch this gives $35 \cdot 35 \cdot 14 = 17150$ channel features and $300 \cdot 14 = 4200$ self-similarity features, yielding a 21350 dimensional feature vector (the learned model will use only a small subset of these features). Computing the channels given a 640×480 input image takes only a fraction of a second using optimized code from Dollár *et al.* [7] available online[1].

[1] http://vision.ucsd.edu/~pdollar/toolbox/doc

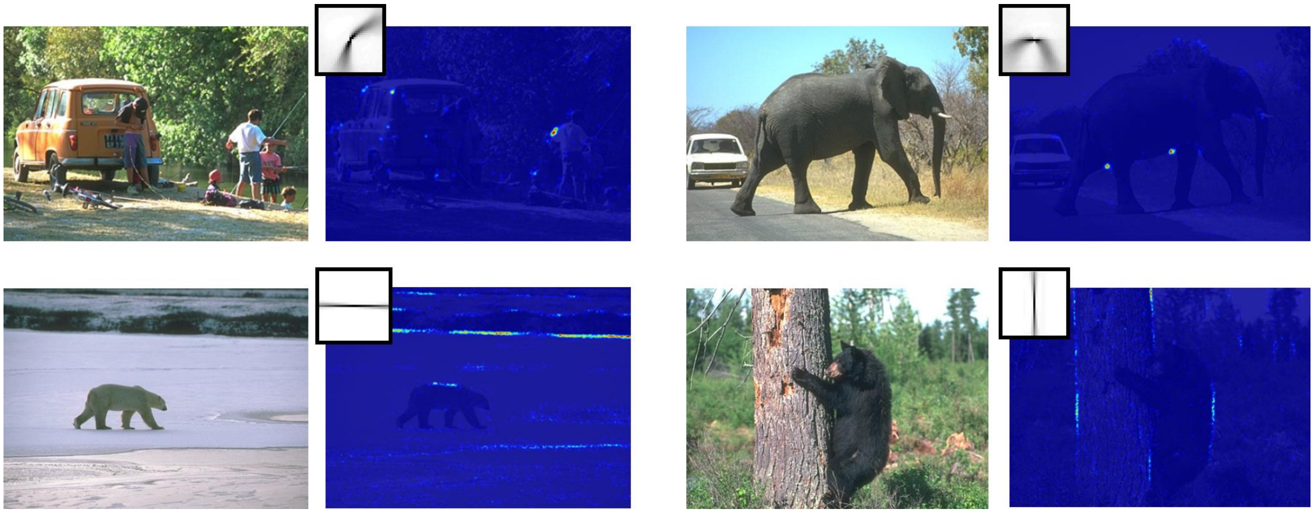


Figure 5. Illustration of the sketch token responses for four tokens. Notice the high selectivity of each sketch token (best viewed in color.)

2.2.2 Classification

Two considerations must be taken into account when choosing a classifier for labeling sketch tokens in image patches. First, every pixel in the image must be labeled, so the classifier must be efficient. Second, the number of potential classes for each patch ranges in the hundreds. In this work we use a random forest classifier, since it is an efficient method for multi-class problems [2, 4, 28].

A random forest is a collection of decision trees whose results are averaged to produce a final result. We randomly sample 150,000 contour patches (1000 per token class) and 160,000 "no contour" patches (800 per training image) for training each tree. The Gini impurity measure is used to select a feature and decision boundary for each branch node from a randomly selected subset of $\sqrt{F}$ of F possible features. The leaf nodes contain the probabilities of belonging to each class and are typically quite sparse. We use a collection of 25 trees trained until every leaf node is pure or contains fewer than 5 examples. The median depth of the tree is 20 although some branches are substantially deeper.

Training each tree takes ~30 minutes and trees are trained in parallel. All model, feature and random forest parameters were set using the BSDS validation set [1].

To gain understanding into which features the decision trees use, we plot the frequency of feature use in Figure 4. Notice the heavy use of the image gradients in the center of the patch based on gradient scale. Gradient orientations are also used in an intuitive manner. We show an illustration of the probabilities for several different sketch tokens in Figure 5. Notice the high selectivity of the different sketch tokens.

3. Contour detection

We now describe our approach to detecting contours using a top-down approach. Sketch tokens provide an estimate of the local edge structure in a patch. However, contour detection only requires the binary labeling of pixel contours. We show that computing mid-level sketch tokens provides accurate and efficient predictions of low-level contours.

Our random forest classifier predicts the probability that an image patch belongs to each token class or the negative set. Since each token has a contour located at its center, we can compute the probability of a contour at the center pixel using the sum of token probabilities. If t_{ij} is the probability of patch x_i belonging to token j, and t_{i0} is the probability of belonging to the "no contour" class, the estimated probability of the patch's center containing a contour is:

$$e_i = \sum_j t_{ij} = 1 - t_{i0}. \tag{2}$$

Once the probability of a contour has been computed at each pixel, a standard non-maximal suppression scheme may be applied to find the peak response of a contour [3].

3.1. Contour detection results

We test our contour detector on the popular Berkeley Segmentation Dataset and Benchmark (BSDS500) [20, 1]. The BSDS500 dataset contains 200 training, 100 validation, and 200 testing images. In Figure 6 and Table 1, we compare our contour detection method against competing methods using standard evaluation metrics [1]. Our method achieves state-of-the-art results among all local methods, and achieves nearly the accuracy of global approaches. Additionally, our contour detector shows improved recall and precision at both ends of the precision-recall curve in Figure 6. Qualitative comparisons are shown in Figure 10.

Our detector processes a 480 × 320 image in under 1 second. This is over 200× more efficient than approaches with similar accuracy, see Table 1. Most computation is spent in the sketch token detection stage, which can be parallelized.

Method	ODS	OIS	AP	Speed
Human	.80	.80	-	-
Canny	.60	.64	.58	1/15 s
Felz-Hutt [12]	.61	.64	.56	1/10 s
gPb (local) [1]	.71	.74	.65	60 s
SCG (local) [24]	.72	.74	.75	100 s
Sketch tokens	**.73**	**.75**	**.78**	**1 s**
gPb (global) [1]	.73	.76	.73	240 s
SCG (global) [24]	.74	.76	.77	280 s

Table 1. Contour detection result on BSDS500: We achieve state-of-the-art results among all local methods. The methods shown in the last two rows perform complex global reasoning given local edge responses, resulting in slightly better performance. However, our approach is 240-280x faster than these global approaches.

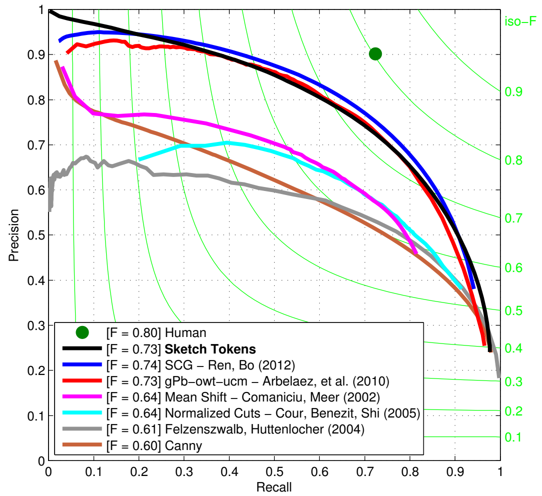


Figure 6. Precision/Recall curve for contour detection. Note that while our method achieves similar F-measure (ODS) to gPb-owt-ucm [1] and SCG [24], we achieve improved results with both low and high recall. This results in high average precision (AP) scores.

Using just $k = 1$ clusters, which is equivalent to classifying pixels as edge versus non-edge, reduced performance considerably from ODS=.73 to ODS=0.68. This is comparable to the results of [6] who used a similar binary classification approach. Using $k = 8$ was sufficient to achieve reasonable edge detection results (ODS=.72).

4. Object detection

We demonstrate our mid-level sketch token features on two object recognition datasets; the INRIA pedestrian dataset [5] and the PASCAL 2007 object recognition dataset [10]. Results on both are described next.

4.1. INRIA pedestrian

For pedestrian detection we use an improved implementation of Dollár *et al.* [7] that utilizes multiple image channels (*e.g.* color, gradient magnitude and oriented gradients) as features for a boosted detector (we utilize similar channels for computing tokens). In addition to standard channels, we add channels corresponding to our sketch token probability maps (computed at twice the resolution of the original images). Unlike traditional channels that capture low-level information, our channels represent the more complex edge structures that may exist in the scene.

channels	# channels	miss rate
LUV	3	72.7%
M+O	7	20.7%
LUV+M+O	10	17.2%
ST	151	19.5%
ST+LUV	154	16.5%
ST+LUV+M+O	**161**	**14.7%**

Table 2. Accuracy of [7] combined with sketch tokens on INRIA [5] with varying choice of channels. The addition of our new mid-level sketch token features gives a significant boost to accuracy.

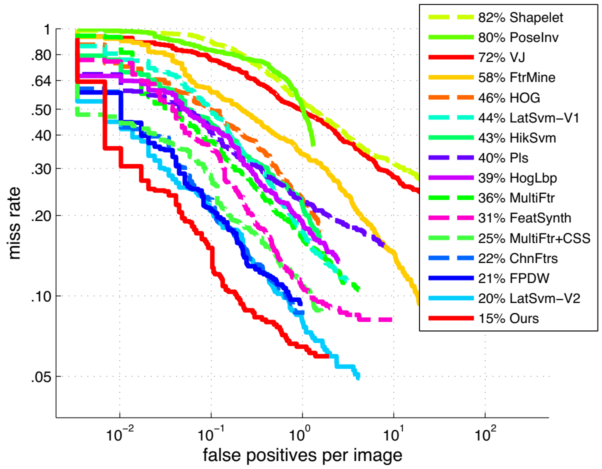


Figure 7. Mean log-average miss rate on the INRIA pedestrian dataset: notice the considerable improvement over previous techniques using our approach. At a 90% detection rate, we achieve a 10× reduction in FPPI over the previous state-of-the-art.

Results are shown in Table 2 and Figure 7. The baseline approach of [7] uses 10 channel features (LUV+M+O) and achieves a log-average miss rate (MR) of 17.2%. Our approach using 150 sketch tokens plus the negative "no contour" feature achieves a MR of 19.5%. Combining sketch tokens and the 10 low-level features achieves 14.7%; a large reduction in error over the baseline approach of [7].

The MR vs. false positive per image rate for numerous state-of-the-art methods is shown in Figure 7. The plot shows large improvement over the best previously published results, while older approaches such as HOG [5] and VJ [35] perform significantly worse.

An interesting question is how the number of sketch tokens affects detection results. In Figure 8, we show results using various numbers of token classes. Notice that using increasing numbers of tokens leads to improved accuracy.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	moto	person	plant	sheep	sofa	train	tv
HOG	19.7	43.9	2.2	4.8	13.4	36.6	40.2	5.4	10.9	15.7	7.5	2.1	41.9	30.9	23.9	3.4	9.3	14.8	26.9	32.4
ST	17.8	41.1	4.8	5.7	11.1	31.9	33.8	5.1	10.8	16.1	7.4	3.1	32.9	27.0	20.9	4.6	8.6	10.4	18.9	26.3
ST+HOG	**21.9**	**48.5**	**6.3**	**6.4**	**14.6**	**41.5**	**43.3**	**6.1**	**15.7**	**19.2**	**14.2**	**3.8**	**46.1**	**34.5**	**30.9**	**8.1**	**15.3**	**18.9**	**30.3**	**36.6**

Table 3. PASCAL 2007 results for linear SVMs: Sketch tokens+HOG outperforms HOG on all classes by 3.8 AP on average.

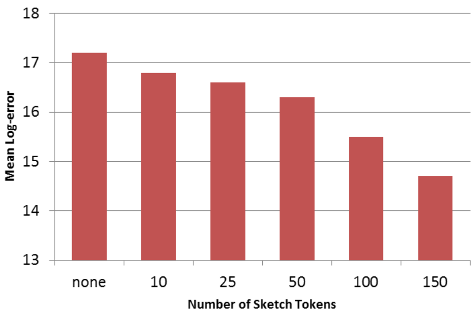


Figure 8. Mean log-error rates on the INRIA dataset using different numbers of sketch tokens. Notice that capturing a large number of varying edge structures leads to a large increase in performance.

4.2. PASCAL VOC 2007

Our final set of results use the PASCAL VOC 2007 dataset [10]. The dataset contains real world images with 20 labeled object categories such as people, dogs, chairs, etc. We perform experiments with the deformable parts model (DPM) of Felzenszwalb *et al*. [11], a widely used state-of-the-art approach for object detection. DPMs use Histogram of Oriented Gradients (HOG) [5] as input features to a latent linear SVM. We propose adding our sketch tokens to the HOG features for training the DPMs.

Unlike the boosting technique we used for pedestrian detection [7], linear SVMs are highly sensitive to how the features are normalized. For normalization we use a technique similar to that used by the HOG descriptor [5]. The sketch token features are divided into a grid with cells of size 4×4 pixels. Each cell is normalized based on its 8 neighbors in a 3×3 grid using the R-HOG technique [5]. Specifically, for each cell we normalize its values four times using sets of 2×2 cells, truncate the four normalized values at 0.2 and average them together. Finally, the features values are scaled to have similar ranges to that of the HOG descriptor.

Results are shown in Tables 3 and 4 for DPMs using the root node only and the full part model, respectively. While individually HOG features outperform sketch tokens, in nearly all cases top Average Precision (AP) scores are achieved with a combination of HOG and sketch tokens. This demonstrates that sketch tokens may provide a valuable complementary source of information over the standard HOG descriptor. We visualize the weights learned by the SVM classifiers for different objects in Figure 9. Notice the object structures, and the use of non-trivial tokens.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	moto	person	plant	sheep	sofa	train	tv
HOG	**27.9**	56.5	1.9	6.2	21.2	48.2	**52.7**	**7.6**	17.7	21.2	14.7	3.0	55.4	42.9	**33.9**	6.0	11.9	21.7	43.2	37.7
ST+HOG	23.8	**58.2**	**10.5**	**8.5**	**27.1**	**50.4**	52.0	7.3	**19.2**	**22.8**	**18.1**	**8.0**	**55.9**	**44.8**	32.4	**13.3**	**15.9**	**22.8**	**46.2**	**44.9**

Table 4. PASCAL 2007 results for DPMs: On average Sketch Tokens+HOG outperformed HOG by 2.5 AP.

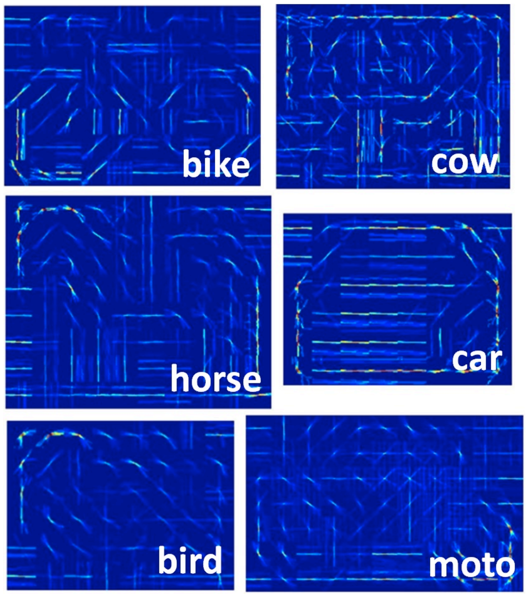
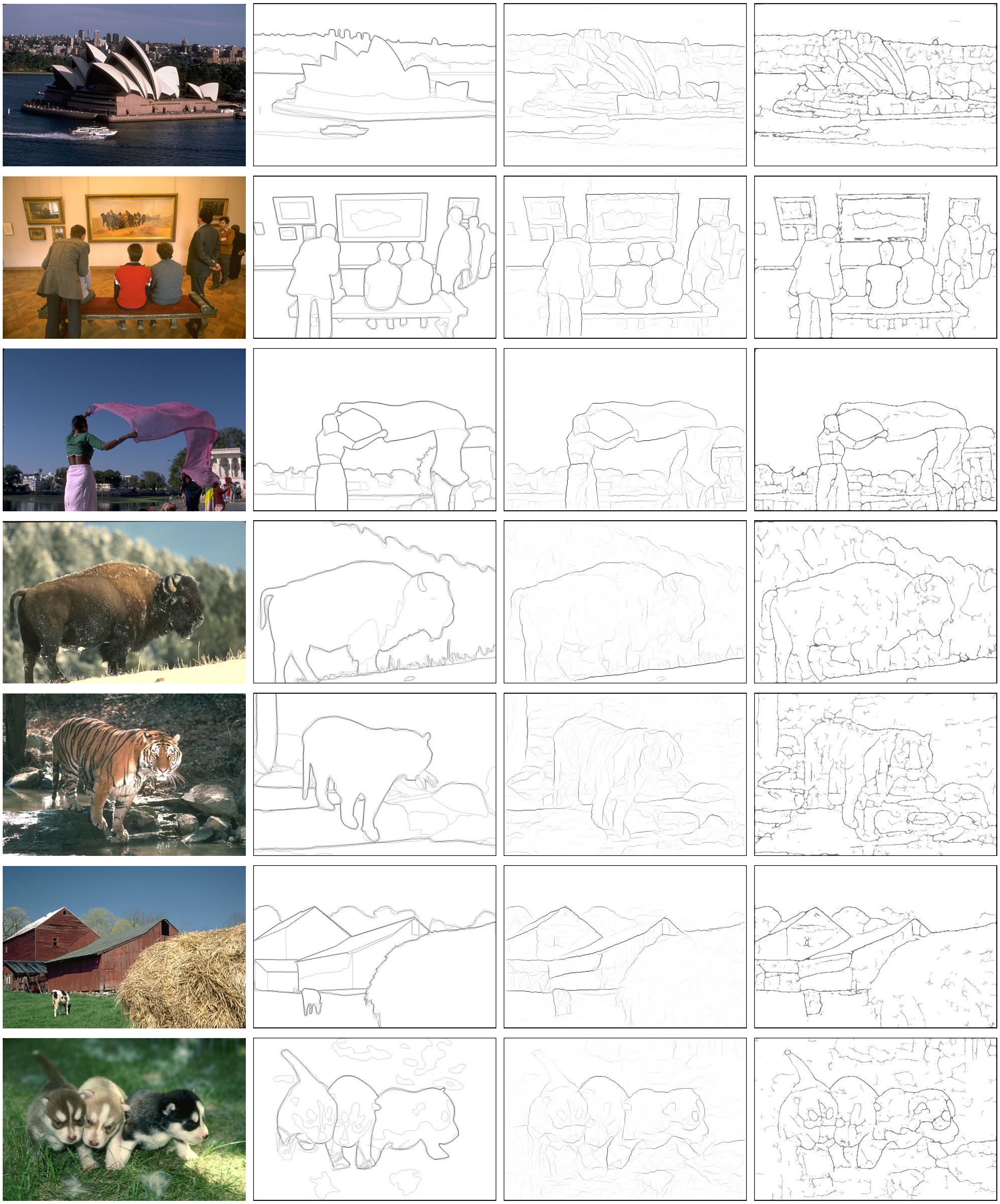


Figure 9. Sketch token weight visualization: We visualize the top 5 sketch tokens multiplied by the learned weight for each cell. Notice the many sketch tokens with rich edge structures that are used.

5. Discussion

Sketch tokens provide a novel approach to feature learning. Unlike many previous methods for generating mid-level features [16, 33, 15, 40], we use supervised mid-level information. Gathering mid-level information from human subjects is a difficult task since such information may not be accessible. Discovering new sets of observable mid-level information that may be used for feature learning is an interesting and open question. Perhaps figure/ground or texture attributes may prove equally informative.

Sketch tokens provide a rich source of information for a variety of tasks. We've explored several in this paper, but other tasks may also benefit. For instance, better contour detection may be useful for many image editing tasks. Tokens tailored to individual categories may also improve detection by adapting to the category's specific characteristics.



(a) Original image (b) Ground truth (c) SCG [24] (d) Sketch Tokens

Figure 10. Examples of contour detection on the BSDS500 [1]. For Sketch Tokens we define edge strength according to Equation 2 and apply smoothing and standard non-maximal suppression to obtain peak edge responses [3]. Note how our method captures finer details such as the structure of Sydney Opera House on the 1st row and human legs on the 2nd row.

In conclusion, we described a new mid-level feature called sketch tokens. The tokens are learned from images with ground truth contours and are fast to compute. Using the learned tokens we demonstrated state-of-the-art results on several datasets ranging from contour to object detection.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33, 2011. 1, 2, 4, 5, 7

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001. 2, 4

[3] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, November 1986. 1, 4, 7

[4] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, February 2012. 2, 4

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2, 3, 5, 6

[6] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 2, 5

[7] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. 2, 3, 5, 6

[8] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, Jan. 1972. 1

[9] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graphics*, 31(4), 2012. 1

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010. 1, 2, 5, 6

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9):1627–1645, 2010. 2, 6

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 5

[13] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *CVPR*, 2009. 2

[14] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 1

[15] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Comp.*, 18(7), 2006. 1, 2, 6

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-prop. network. In *NIPS*, 1990. 1, 2, 6

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 3

[18] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008. 2

[19] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167):187–217, 1980. 1

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 2, 4

[21] A. Opelt and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, 2006. 1, 2

[22] L. Parida, D. Geiger, and R. Hummel. Junctions: detection, classification, and reconstruction. *PAMI*, 20(7), 1998. 1

[23] M. Prasad, A. Zisserman, A. Fitzgibbon, M. Kumar, and P. Torr. Learning class-specific edges for object detection and segmentation. *CVGIP*, 2006. 1, 2

[24] X. Ren and B. Liefeng. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 2, 5, 7

[25] L. G. Roberts. Machine perception of 3d solids. In J. T. T. et al., editor, *Optical and Electro-optical Information Processing*, pages 159–197. MIT Press, 1965. 1

[26] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007. 2, 3

[27] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *ICCV*, 2005. 2

[28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011. 4

[29] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 2

[30] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009. 2

[31] J. Sullivan and S. Carlsson. Recognizing and tracking human action. *ECCV*, 2002. 1

[32] C. Taylor and D. Kriegman. Structure and motion from line segments in multiple images. *PAMI*, 17(11), 1995. 1

[33] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recogn. using classemes. In *ECCV*, 2010. 1, 6

[34] S. Ullman and R. Basri. Recognition by linear combinations of models. *PAMI*, 13(10), 1991. 1

[35] P. A. Viola and M. J. Jones. Robust real-time face det. *IJCV*, 57(2):137–154, 2004. 5

[36] S. A. J. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, 2009. 2

[37] J. Winn and N. Jojic. LOCUS: Learning object classes with unsupervised segmentation. In *ICCV*, 2005. 1

[38] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet part det. *IJCV*, 75(2):247–266, 2007. 2

[39] T. Wu, G. Xia, and S. Zhu. Compositional boosting for computing hierarchical image structures. In *CVPR*, 2007. 1, 2

[40] M. Zeiler, G. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 1, 2, 6

[41] S. Zheng, Z. Tu, and A. Yuille. Detecting object boundaries using low-, mid-, and high-level information. In *CVPR*, 2007. 2