

Visualizing Object Detection Features

by

Carl Vondrick

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

© Massachusetts Institute of Technology 2013. All rights reserved.

Author
.....

Department of Electrical Engineering and Computer Science
May 15, 2013

Certified by
.....

Antonio Torralba
Associate Professor
Thesis Supervisor

Accepted by
.....

Leslie A. Kolodziejksi
Chairman, Department Committee on Graduate Theses

Dedicated to my sister

Visualizing Object Detection Features

by

Carl Vondrick

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We introduce algorithms to visualize feature spaces used by object detectors. The tools in this paper allow a human to put on ‘HOG goggles’ and perceive the visual world as a HOG based object detector sees it. We found that these visualizations allow us to analyze object detection systems in new ways and gain new insight into the detector’s failures. For example, when we visualize high scoring false alarms, we discovered that, although they are clearly wrong in image space, they do look deceptively similar to true positives in feature space. This result suggests that many of these false alarms are caused by our choice of feature space, and indicates that creating a better learning algorithm or building bigger datasets is unlikely to correct these errors. By visualizing feature spaces, we can gain a more intuitive understanding of our detection systems.

Thesis Supervisor: Antonio Torralba

Title: Associate Professor

Acknowledgments

The ideas and tools in this thesis would not have been possible without the patient guidance from many people, whom I hope to acknowledge here.

Most of these algorithms and experiments grew out from countless conversations with my collaborators and lab mates, and for this I am indebted. Many thanks to Agata Lapedriza, Andrew Owens, Hamed Pirsiavash, Jenny Yuen, Jianxiong Xiao, Joseph Lim, and Zoya Gavrilov. I give special thanks to Aditya Khosla and Tomasz Malisiewicz for going the extra mile to help me refine methods, design experiments, and execute evaluation.

I thank my advisor, Antonio Torralba, for providing a never ending stream of wisdom and ideas that has proved crucial when brainstorming experiments and interpreting results. His constant enthusiasm and encouragement makes working with him a great pleasure.

I wish to give thanks to Deva Ramanan for being an inspiring mentor who introduced me to the field of computer vision and taught me how to conduct scientific research.

Finally, I owe great thanks to my friends for their laughter and my family for their endless support.

Funding for this work was provided by a NSF Graduate Research Fellowship, a Google research award, ONR MURI N000141010933 and NSF Career Award No. 0747120.

Contents

1	Introduction	15
1.1	Contributions	19
1.2	Thesis Overview	19
2	Related Work	21
2.1	Feature Inversion and Visualization	21
2.2	Diagnosing Object Detection Systems	22
3	Feature Visualization Algorithms	23
3.1	Algorithm A: Exemplar LDA (ELDA)	24
3.2	Algorithm B: Ridge Regression	25
3.3	Algorithm C: Direct Optimization	26
3.4	Algorithm D: Paired Dictionary Learning	26
4	Evaluation of Visualizations	31
4.1	Qualitative Results	31
4.1.1	SIFT Comparison	31
4.1.2	Dimensionality	33
4.1.3	Color Inversions	33
4.2	Quantitative Benchmarks	33
4.2.1	Reconstruction Error	35
4.2.2	Visualization Benchmark	35

5 Understanding Object Detectors	43
5.1 HOG Goggles	43
5.2 Human+HOG Detectors	46
5.3 Tweaking HOG	48
5.3.1 Normalization	49
5.3.2 Texture	49
5.4 Interpolation in HOG Space	49
5.5 Visualizing Models	51
5.5.1 Model Weight Visualization	51
5.5.2 Super Objects	54
5.6 Choice of Features	54
6 Conclusion	57

List of Figures

1-1	Example of our visualization.	16
1-2	A curious false detection	17
1-3	Visualizing the curious false detection	17
1-4	Visualizations of false positives	18
3-1	Feature inversion by Exemplar LDA	24
3-2	Visualization of paired dictionary	27
3-3	Paired dictionary learning graphic	28
4-1	Qualitative results of feature inversions	32
4-2	Comparison of HOG inversions to SIFT inversions	34
4-3	Feature inversions and dimensionality	35
4-4	Color inversion results on outdoor images	36
4-5	Color inversions results on indoor images	37
4-6	Color inversions learned dictionaries	37
4-7	Visualization benchmark confusion matrices	40
5-1	HOG Goggles	44
5-2	Visualizing false positives	45
5-3	Human+HOG Detector User Interface	46
5-4	Human+HOG detector performance	47
5-5	Visualizing HOG normalization	50
5-6	Visualizing HOG texture	51
5-7	Linear interpolation in HOG space	52

5-8	Visualizing learned models	53
5-9	Visualizing super objects	54
5-10	Visualizing HSC vs HOG	55

List of Tables

4.1	Measuring reconstruction error	38
4.2	Measuring visualization performance	39

Chapter 1

Introduction

“The real voyage of discovery consists not in seeking new landscapes but in having new eyes.” — Marcel Proust

One of the fundamental problems in computer vision is to build a system that automatically recognizes objects in images. The rise of object recognition systems would enable many particularly exciting applications, such as robots that clean and cook in our homes, smart cameras that automatically respond to crises, or cars that drive themselves. Unfortunately, although humans are able to perceive the visual world without difficulty, building an automatic vision system has proved challenging.

While there are many possible approaches to object recognition, the dominant paradigm today uses machine learning methods to learn visual appearance models from a large database [7, 14, 26, 35, 2, 15, 11, 22]. One of the most crucial components in this paradigm is the feature space for representing an image, and the choice of features often has the most impact on the final performance [29]. Consequently, there has been significant work focused on creating better features for object recognition [28, 7, 24, 1, 4, 38, 31, 9].

This thesis introduces the tools to visualize feature spaces.¹ Since most feature spaces are too high dimensional for humans to directly inspect, we present algorithms to invert feature descriptors back to a natural image. We found that these inversions

¹A preliminary version of this work appeared in [36].

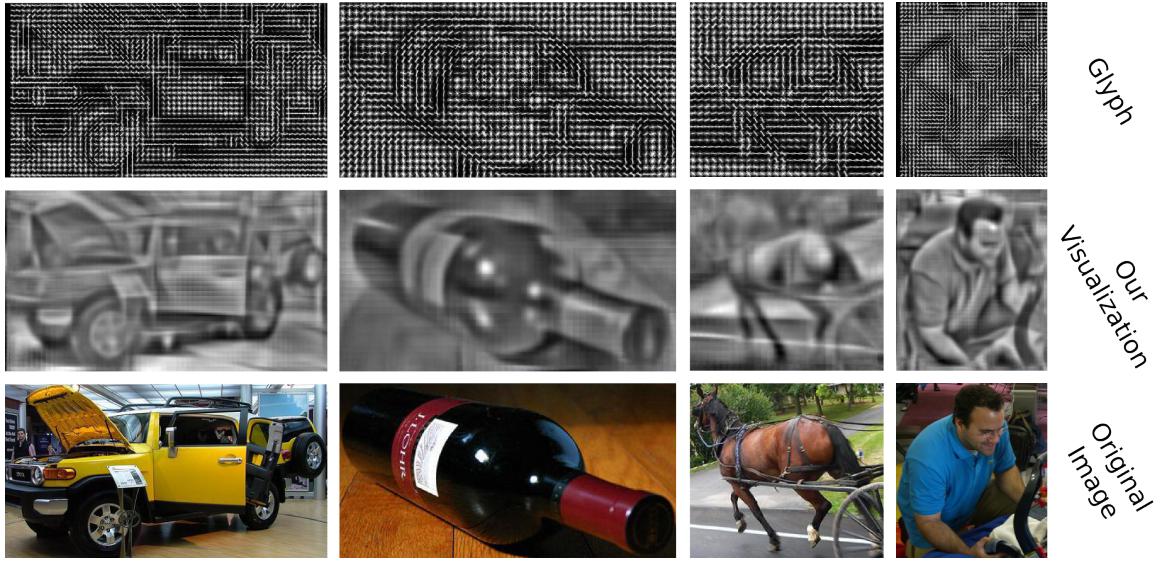


Figure 1-1: Example of our visualization.

provide an accurate and intuitive visualization of feature descriptors commonly used in object detection. See Figure 1-1 for an example of the visualization.

We discovered that these visualizations allow us to inspect object detection systems in new ways and obtain insights into a detector’s failures. Consider Figure 1-2, which shows a high scoring detection from an object detector with HOG features [7] and a linear SVM [6] with deformable parts [14] trained on PASCAL [12]. Despite the field’s progress, *why* do our detectors still think that sea water looks like a car?

Our visualizations offer an explanation. Figure 1-3 shows the output from our visualization on the features for the false car detection. This visualization reveals that, while there are clearly no cars in the original image, there is a car hiding in the HOG descriptor. HOG features see a slightly different visual world than what humans see, and by visualizing this space, we can gain a more intuitive understanding of our object detectors.

Figure 1-4 inverts more top detections for a few categories. Can you guess which are false alarms? Take a minute to study the figure since the next sentence might ruin the surprise. Although every visualization looks like a true positive, all of these detections are actually false alarms. Consequently, we can conclude that, even with a better learning algorithm or more data, these false alarms will likely persist. In other



Figure 1-2: An image from PASCAL and a high scoring car detection from DPM [14]. Why did the detector fail?

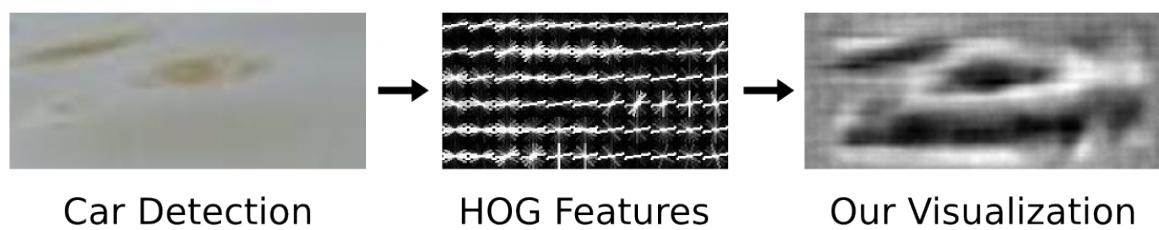
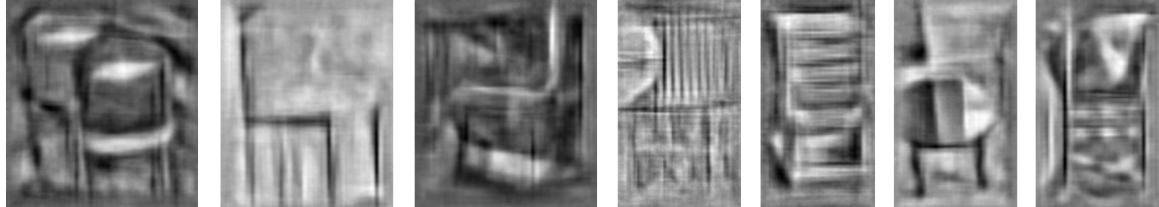


Figure 1-3: We show the crop for the false car detection from Figure 1-2. On the right, we show our visualization of the HOG features for the same patch. Our visualization reveals that this false alarm actually looks like a car in HOG space.



(a) Person



(b) Chair



(c) Car

Figure 1-4: We visualize some high scoring detections from the deformable parts model [14] for person (top left), chair (top right), and car (bottom). Can you guess which are false alarms? Take a minute to study this figure, then see Figure 5-2 for the corresponding RGB patches.

words, the features are to blame.

We expect feature visualizations can be a powerful tool for understanding object detection systems and advancing research in computer vision. The tools in this thesis allow humans to put on “feature space glasses” to perceive the visual world as a computer sees it. Our hope is that these visualization tools will allow scientists and researchers to gain a more intuitive understanding of the features that we use everyday to ultimately advance the state-of-the-art in object detection systems.

1.1 Contributions

The contributions in this thesis revolve around developing and using feature space visualizations for advancing research in object detection:

1. The primary contribution of this thesis is the presentation of algorithms for visualizing features used in object detection. To this end, we present four algorithms to invert object detection features back to natural images. Each algorithm has different trade-offs: some are fast, some are non-parametric, and others are more accurate. All of our algorithms are simple to understand, and they are general so they can be used to visualize any feature.
2. In order to compare our visualization quality, we further propose two quantitative benchmarks to evaluate the performance of feature inversion algorithms. Our first metric is automatic and uses normalized cross correlation to measure how well an algorithm reconstruct relative pixel values. Our second metric uses a large human study to see how well people can recover high level semantic information from our visualizations. As we will show, our visualizations are significantly more accurate than existing methods under both of these metrics.
3. The final contribution of this thesis is demonstrating that visualizations are useful for inspecting the behavior of object detection systems and analyzing failures. We present a variety of experiments using our HOG inverse, such as diagnosing a detector’s false positives, generating high scoring “super objects” for an object detector, examining different feature hyperparameters, and visualizing an object detector’s learned decision boundary. Our visualizations reveal that the features are to blame for many object detection failures.

1.2 Thesis Overview

The remainder of this thesis describes, evaluates, and applies our visualization algorithms in detail. Chapter 2 briefly reviews related work in feature inversion and

diagnosing object detection errors. Chapter 3 describes four algorithms to invert features. Chapter 4 evaluates all our algorithms on HOG features using both an automatic benchmark as well as a large human study. Chapter 5 then uses our visualization algorithms to explain some of the unusual behaviors of object detection systems. Chapter 6 finally offers concluding remarks.

Chapter 2

Related Work

In this chapter, we briefly review related work. This thesis is most closely related to work in feature inversion and a recent line of papers that attempt to diagnose object detection systems.

2.1 Feature Inversion and Visualization

Our visualization algorithms extend an actively growing body of work in feature inversion. Torralba and Oliva, in early work [34], described a simple iterative procedure to recover images only given gist descriptors [28]. Weinzaepfel et al. [39] were the first to reconstruct an image given its keypoint SIFT descriptors [24]. Their approach obtains compelling reconstructions using a nearest neighbor based approach on a massive database. d’Angelo et al. [8] then developed an algorithm to reconstruct images given only LBP features [4, 1]. Their method analytically solves for the inverse image and does not require a dataset. In a related vein of work, Hariharan et al. [16] further describe a method to invert descriptors back to a contour image.

While [39, 8, 34] do a good job at reconstructing images from SIFT, LBP, and gist features, our visualization algorithms have several advantages. Firstly, while existing methods are designed for specific features, our visualization algorithms we propose are feature independent. Since we cast feature inversion as a machine learning problem, our algorithms can be used to visualize any feature. In this thesis, we focus on features

for object detection, the most popular of which is HOG. Secondly, our algorithms are fast: our best algorithm can invert features in under a second on a desktop computer, enabling interactive visualization. Finally, to our knowledge, this thesis is the first to invert HOG.

2.2 Diagnosing Object Detection Systems

The application of our visualizations complement a recent line of papers that provide tools to diagnose object recognition systems, which we briefly mention here. Parikh and Zitnick [30, 29] introduced a new paradigm for human debugging of object detectors, an idea that we adopt in our experiments. Hoiem et al. [18] performed a large study analyzing the errors that object detectors make. Divvala et al. [10] analyze part-based detectors to determine which components of object detection have the most impact on performance. Tatu et al. [32] explored the set of images that generate identical HOG descriptors. Liu and Wang [23] designed algorithms to highlight which image regions contribute the most to a classifier’s confidence. Zhu et al. [41] try to determine whether we have reached Bayes risk for HOG. The tools in this thesis enable an alternative mode to analyze object detectors through visualizations. By putting on ‘HOG glasses’ and visualizing the world according to the features, we are able to gain a better understanding of the failures and behaviors of our object detection systems.

Chapter 3

Feature Visualization Algorithms

We pose the feature visualization problem as one of feature inversion, i.e. recovering the natural image that generated a feature vector. Let $x \in \mathbb{R}^D$ be an image and $y = \phi(x)$ be the corresponding feature descriptor. Since $\phi(\cdot)$ is a many-to-one function, no analytic inverse exists. Hence, we seek an image x that, when we compute features on it, closely matches the original descriptor y :

$$\phi^{-1}(y) = \operatorname{argmin}_{x \in \mathbb{R}^D} \|\phi(x) - y\|_2^2 \quad (3.1)$$

Optimizing Equation 3.1 is challenging. Although Equation 3.1 is not convex, we tried gradient-descent strategies by numerically evaluating the derivative in image space with Newton's method for HOG features. Unfortunately, we observed poor results, likely because HOG is both highly sensitive to noise and Equation 3.1 has frequent local minima.

In the rest of this chapter, we present four algorithms for inverting features. We focus on HOG, although our algorithms are general and can be applied to any feature. We begin by describing our simplest algorithm that uses an exemplar object detector to invert features. We then describe a parametric algorithm based off linear regression that learns a mapping between features and natural images. We next present an algorithm that searches over a large space of candidate images to minimize the reconstruction error. Finally, we introduce our main feature inversion that learns a

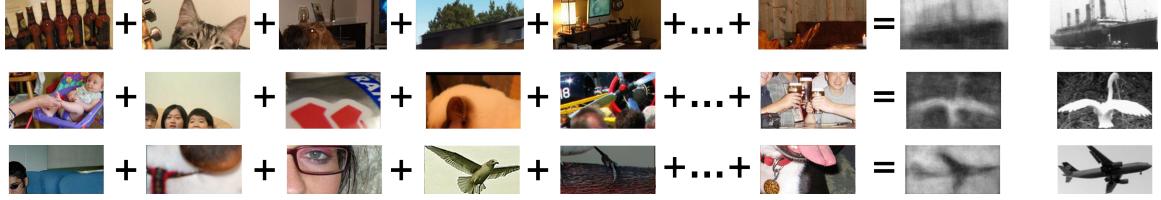


Figure 3-1: Inverting HOG features using exemplar LDA. We train an exemplar LDA model on the HOG descriptor we wish to invert and apply it to a large database. The left hand side of the above equation are the top detections, while the right hand side shows the average of the top 100. Even though all top detections are semantically meaningless, their average is close to the original image, shown on the right. Notice that all the top detections share structure with the original, e.g., the top left bottles create the smoke stack for the ship, and the middle right hands compose the wings for the bird.

pair of dictionaries that enable regression between features and natural images.

3.1 Algorithm A: Exemplar LDA (ELDA)

Consider the top detections for the exemplar object detector [17, 26] for a few images shown in Fig.3-1. Although all top detections are false positives, notice that each detection captures some statistics about the query. Even though the detections are wrong, if we squint, we can see parts of the original object appear in each detection.

We use this simple observation to produce our first inversion algorithm. Suppose we wish to invert HOG feature y . We first train an exemplar LDA detector [17] for this query:

$$w = \Sigma^{-1}(y - \mu) \quad (3.2)$$

We then score w against every sliding window on a large database. The HOG inverse is then simply the average of the top K detections in RGB space:

$$\phi_A^{-1}(y) = \frac{1}{K} \sum_{i=1}^K z_i \quad (3.3)$$

where z_i is a top detection.

This method, although simple, produces surprisingly accurate reconstructions, even when the database does not contain the category of the HOG template. We note that this method may be subject to dataset bias issues [33]. We also point out that a similar nearest neighbor based technique is used in brain research to visualize what a person might be seeing [27].

3.2 Algorithm B: Ridge Regression

Unfortunately, running an object detector across a large database is computationally expensive. In this section, we present a fast, parametric inversion algorithm.

Let $X \in \mathbb{R}^D$ be a random variable representing a gray scale image and $Y \in \mathbb{R}^d$ be a random variable of its corresponding HOG point. We define these random variables to be normally distributed on a $D + d$ -variate Gaussian $P(X, Y) \sim \mathcal{N}(\mu, \Sigma)$ with parameters:

$$\mu = [\mu_X \ \mu_Y] \quad \text{and} \quad \Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{XY}^T & \Sigma_{YY} \end{bmatrix} \quad (3.4)$$

In order to invert a HOG feature y , we calculate the most likely image from the conditional Gaussian distribution $P(X|Y = y)$:

$$\phi_B^{-1}(y) = \underset{x \in \mathbb{R}^D}{\operatorname{argmax}} P(X = x|Y = y) \quad (3.5)$$

It is well known that Gaussians have a closed form conditional mode:

$$\phi_B^{-1}(y) = \Sigma_{XY}\Sigma_{YY}^{-1}(y - \mu_Y) + \mu_X \quad (3.6)$$

Under this inversion algorithm, any HOG point can be inverted by a single matrix multiplication, allowing for inversion in under a second.

We estimate μ and Σ on a large database. In practice, Σ is not positive definite; we add a small uniform prior (i.e., $\hat{\Sigma} = \Sigma + \lambda I$) so Σ can be inverted. Since we wish to invert any HOG point, we assume that $P(X, Y)$ is stationary [17], allowing

us to efficiently learn the covariance across massive datasets. We invert an arbitrary dimensional HOG point by marginalizing out unused dimensions.

We found that ridge regression yields blurred inversions. Intuitively, since HOG is invariant to shifts up to its bin size, there are many images that map to the same HOG point. Ridge regression is reporting the statistically most likely image, which is the average over all shifts. This causes ridge regression to only recover the low frequencies of the original image.

3.3 Algorithm C: Direct Optimization

We now provide a baseline that attempts to find images that, when we compute HOG on it, sufficiently match the original descriptor. In order to do this efficiently, we only consider images that span a natural image basis. Let $U \in \mathbb{R}^{D \times K}$ be the natural image basis. We found using the first K eigenvectors of $\Sigma_{XX} \in \mathbb{R}^{D \times D}$ worked well for this basis. Any image $x \in \mathbb{R}^D$ can be encoded by coefficients $\rho \in \mathbb{R}^K$ in this basis: $x = U\rho$. We wish to minimize:

$$\begin{aligned} \phi_C^{-1}(y) &= U\rho^* \\ \text{where } \rho^* &= \underset{\rho \in \mathbb{R}^K}{\operatorname{argmin}} \|\phi(U\rho) - y\|_2^2 \end{aligned} \tag{3.7}$$

Empirically we found success optimizing Equation 3.7 using coordinate descent on ρ with random restarts. We use an over-complete basis corresponding to sparse Gabor-like filters for U . We compute the eigenvectors of Σ_{XX} across different scales and translate smaller eigenvectors to form U .

3.4 Algorithm D: Paired Dictionary Learning

Direct optimization obtains highly accurate results, but since optimization requires computing HOG features on a large number of candidate images, convergence is slow. In our final algorithm, we propose a fast approximation.

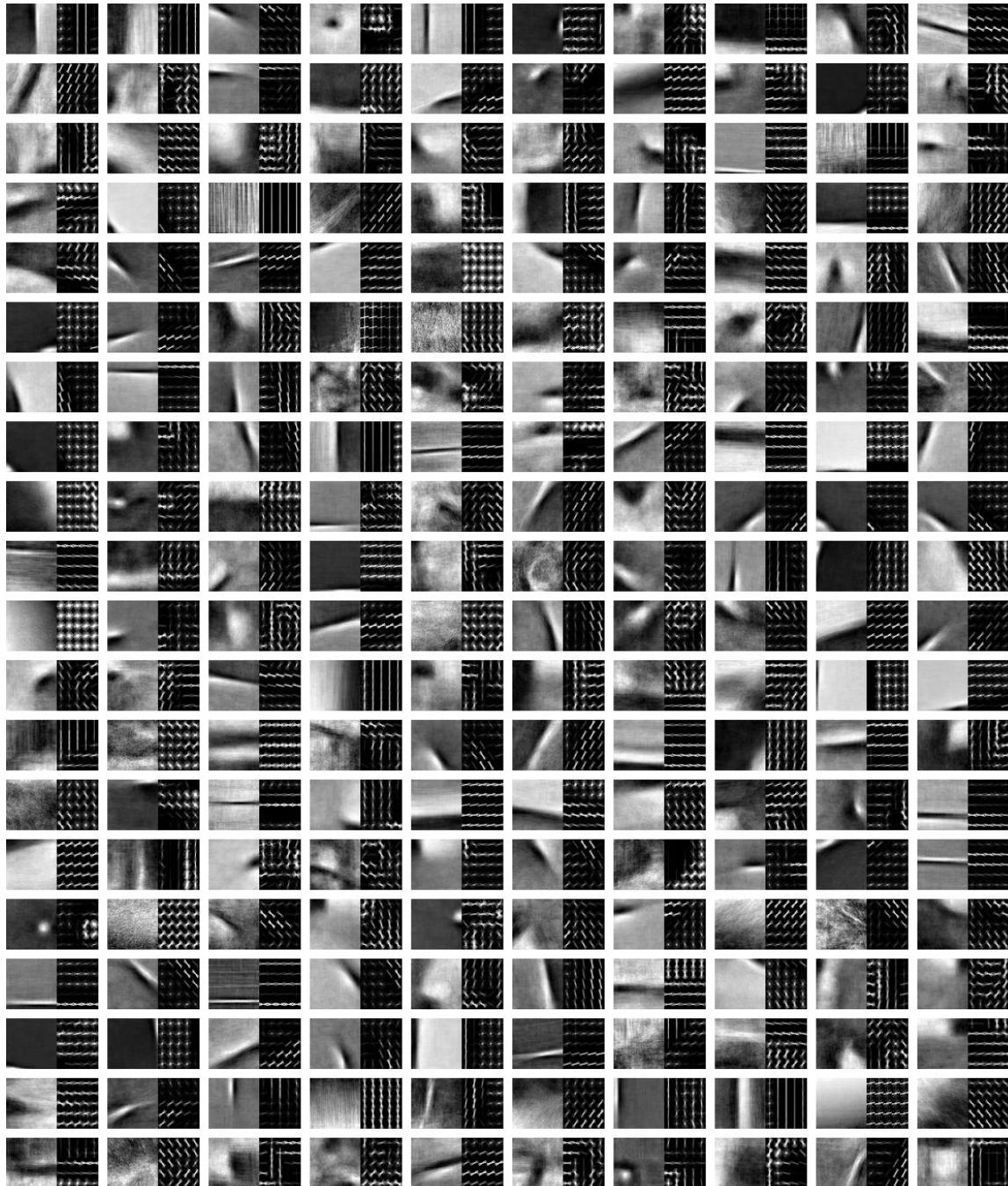


Figure 3-2: Some pairs of dictionaries for U and V . The left of every pair is the gray scale dictionary element and the right is the positive components elements in the HOG dictionary. Notice that the gray patches are correlated with the HOG patches.

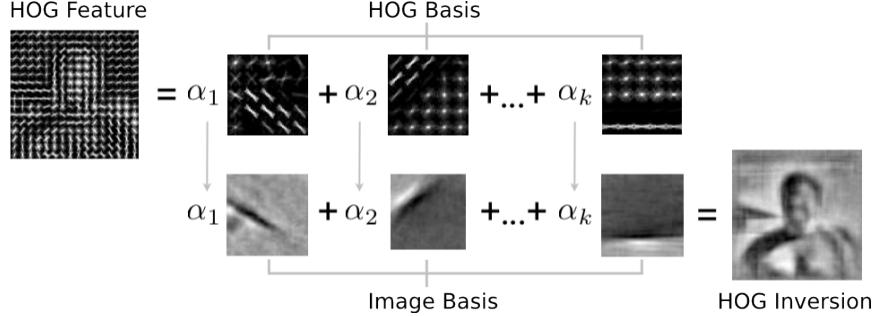


Figure 3-3: Inverting HOG using paired dictionary learning. We first project the HOG vector on to a HOG basis. By jointly learning a coupled basis of HOG features and natural images, we then transfer the coefficients to the image basis to recover the natural image.

Let $x \in \mathbb{R}^D$ be an image and $y \in \mathbb{R}^d$ be its HOG descriptor. Suppose we write x and y in terms of bases $U \in \mathbb{R}^{D \times K}$ and $V \in \mathbb{R}^{d \times K}$ respectively, but with shared coefficients $\alpha \in \mathbb{R}^K$:

$$x = U\alpha \quad \text{and} \quad y = V\alpha \quad (3.8)$$

The key observation is that inversion can be obtained by first projecting the HOG features y onto the HOG basis V , then projecting α into the natural image basis U :

$$\begin{aligned} \phi_D^{-1}(y) &= U\alpha^* \\ \text{where } \alpha^* &= \underset{\alpha \in \mathbb{R}^K}{\operatorname{argmin}} \|V\alpha - y\|_2^2 \quad \text{s.t. } \|\alpha\|_1 \leq \lambda \end{aligned} \quad (3.9)$$

See Figure 3-3 for a graphical representation of the paired dictionaries. Since efficient solvers for Equation 3.9 exist [25, 21], we can invert features in under two seconds on a 4 core CPU.

Paired dictionaries require finding appropriate bases U and V such that Equation 3.8 holds. To do this, we solve a paired dictionary learning problem, inspired by recent super resolution sparse coding work [40, 37]:

$$\begin{aligned} \underset{U, V, \alpha}{\operatorname{argmin}} \sum_{i=1}^N & (\|x_i - U\alpha_i\|_2^2 + \|\phi(x_i) - V\alpha_i\|_2^2) \\ \text{s.t. } & \|\alpha_i\|_1 \leq \lambda \forall i, \|U\|_2^2 \leq \gamma_1, \|V\|_2^2 \leq \gamma_2 \end{aligned} \quad (3.10)$$

After a few algebraic manipulations, the above objective simplifies to a standard sparse coding and dictionary learning problem with concatenated dictionaries, which we optimize using SPAMS [25]. Optimization typically took a few hours on medium sized problems. We estimate U and V with a dictionary size $K \approx 10^3$ and training samples $N \approx 10^6$ from a large database. See Figure 3-2 for a visualization of the learned dictionary pairs.

Unfortunately, the paired dictionary learning formulation suffers on problems of nontrivial scale. In practice, we only learn dictionaries for 5×5 HOG templates. In order to invert a $w \times h$ HOG template y , we invert every 5×5 subpatch inside y and average overlapping patches in the final reconstruction. We found that this approximation works well in practice. We hope to alleviate this concern in the future with convolutional sparse coding [20, 3].

We found that the paired dictionary still obtains reasonable performance if, instead of learning the dictionaries with Equation 3.10, we use randomly samples from the training set as the dictionaries. While there is a noticeable drop in performance, the results are still reasonable, an observation that reinforces recent findings that learning dictionaries may not be crucial for performance in many vision tasks [5]. We note that random samples allow the paired dictionary to be trained in real time with only a modest drop in performance. Nonetheless, learning still provides some benefit, and so report results with the learned dictionaries.

Chapter 4

Evaluation of Visualizations

In this chapter, we evaluate our four inversion algorithms using both qualitative and quantitative measures. We use PASCAL VOC 2011 [12] as our dataset and we invert patches corresponding to objects. Any algorithm that required training could only access the training set. During evaluation, only images from the validation set are examined. The database for exemplar LDA excluded the category of the patch we were inverting to reduce the effect of biases.

4.1 Qualitative Results

We show our inversions in Figure 4-1 for a few object categories. Exemplar LDA and ridge regression tend to produce blurred visualizations. Direct optimization recovers high frequency details at the expense of extra noise. Paired dictionary learning produces the best visualization for HOG descriptors. By learning a sparse dictionary over the visual world and the correlation between HOG and natural images, paired dictionary learning recovered high frequencies without introducing significant noise.

4.1.1 SIFT Comparison

We compare our HOG inversions against SIFT reconstructions on the INRIA Holidays dataset [19]. Figure 4-2 shows a qualitative comparison between paired dictionary

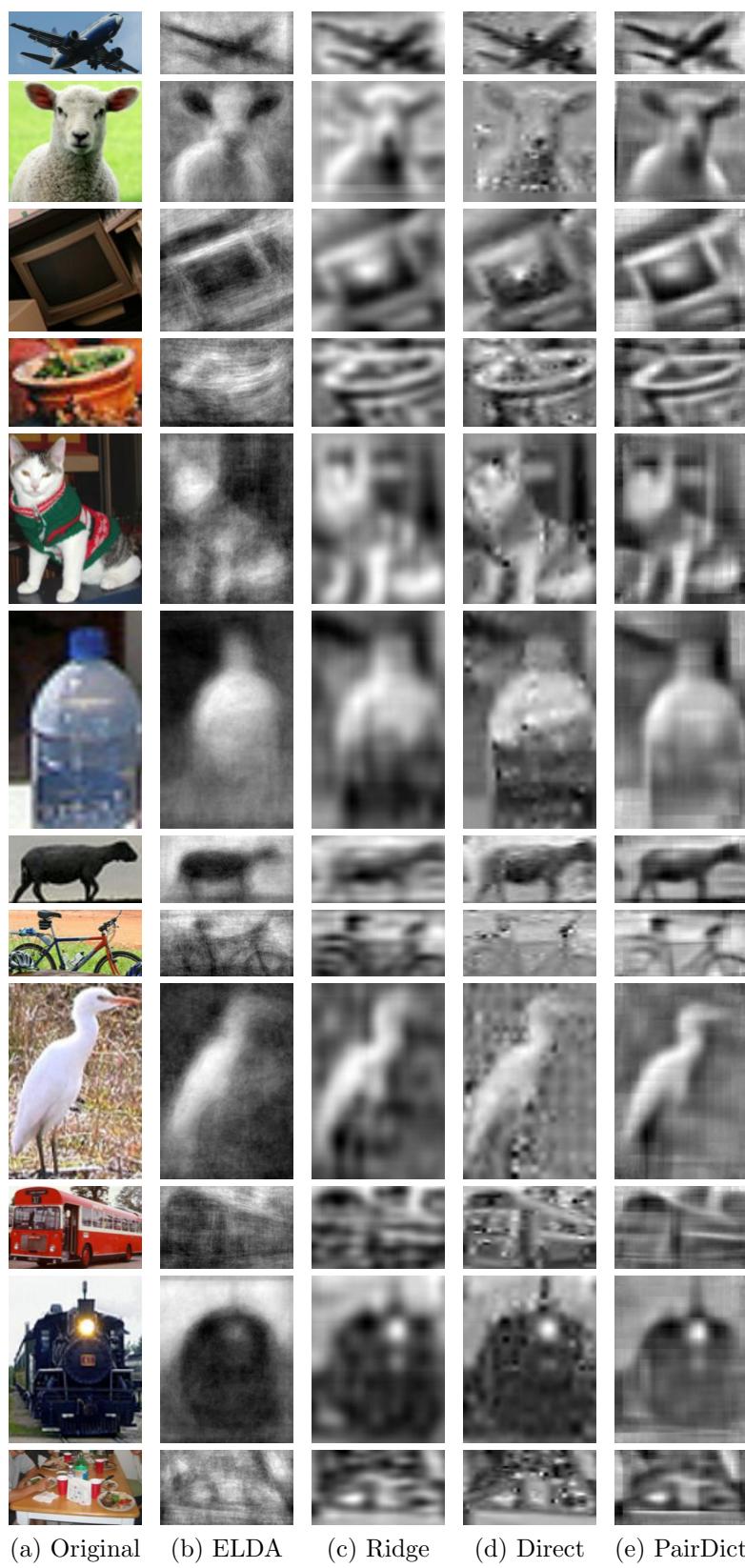


Figure 4-1: We show the results for all four of our inversion algorithms on held out image patches on similar dimensions common for object detection.

learning and Weinzaepfel et al. [39]. Notice that HOG inversion is more blurred than key point SIFT since HOG is histogram based.

4.1.2 Dimensionality

HOG inversions are sensitive to the dimensionality of their templates. For medium (10×10) to large templates (40×40), we obtain reasonable performance. But, for small templates (5×5) the inversion is blurred. Figure 4-3 shows examples as the HOG descriptor dimensionality changes.

4.1.3 Color Inversions

We discovered that the paired dictionary is able to recover color from HOG descriptors. Figure 4-4 shows the result of training a paired dictionary to estimate RGB images instead of grayscale images. While the paired dictionary assigns arbitrary colors to man-made objects and in-door scenes (see Figure 4-5), it frequently colors natural objects correctly, such as grass or the sky, likely because those categories are strongly correlated to HOG descriptors. We focus on grayscale visualizations in this thesis because we found those to be more intuitive for humans to understand.

4.2 Quantative Benchmarks

In the remainder of this chapter, we evaluate our algorithms under two benchmarks: first, an inversion metric that measures how well our inversions reconstruct the original images, and second, a visualization challenge conducted on Amazon Mechanical Turk designed to determine how well people can infer the original category from the inverse. The first experiment measures the algorithm’s reconstruction error, while the second experiment analyzes the recovery of high level semantics.

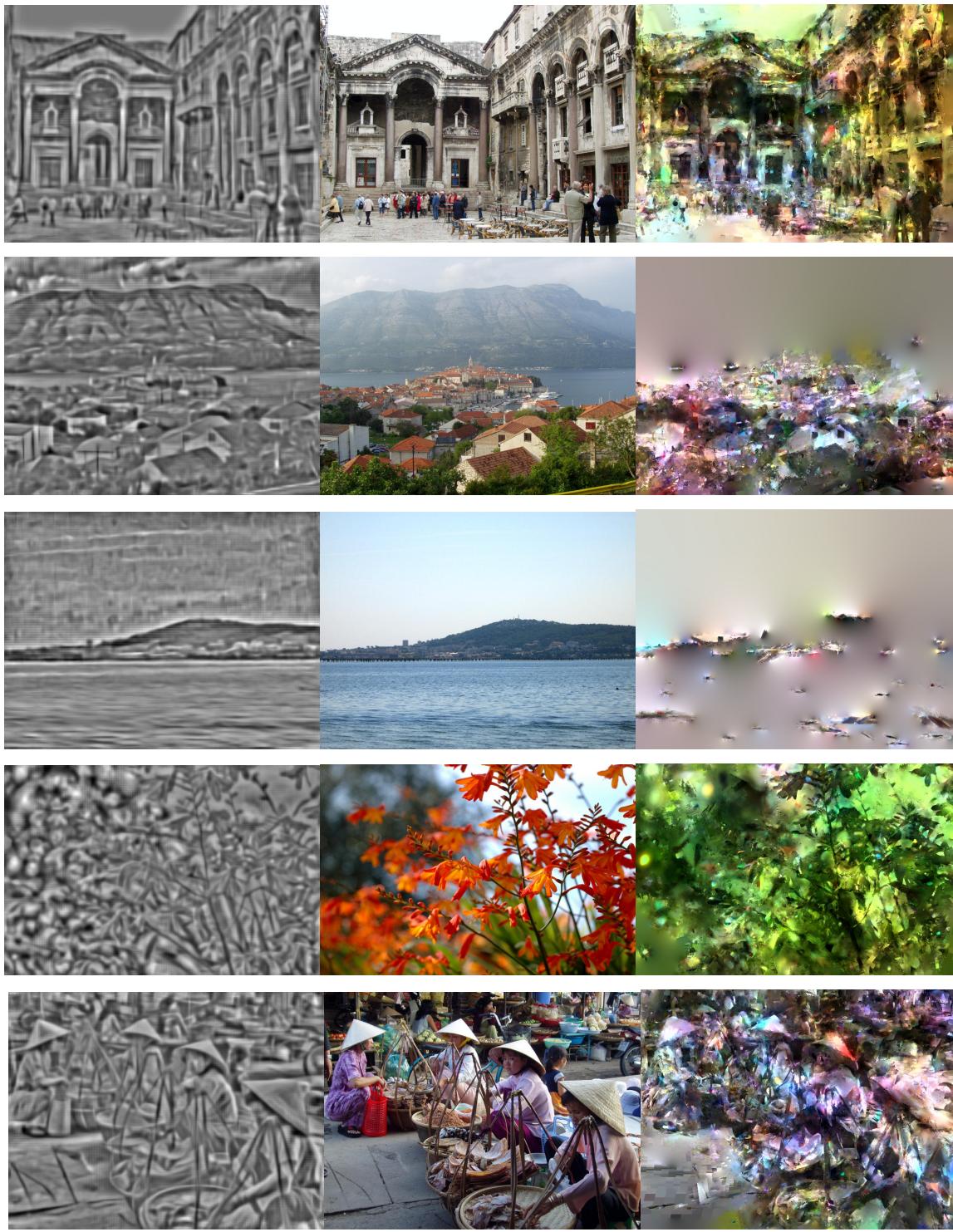


Figure 4-2: We compare our paired dictionary learning approach on HOG with the algorithm of [39] on SIFT. Since HOG is invariant to color, we are only able to recover a grayscale image. Furthermore, our blurred inversion shows that HOG is a more coarse descriptor than keypoint SIFT.

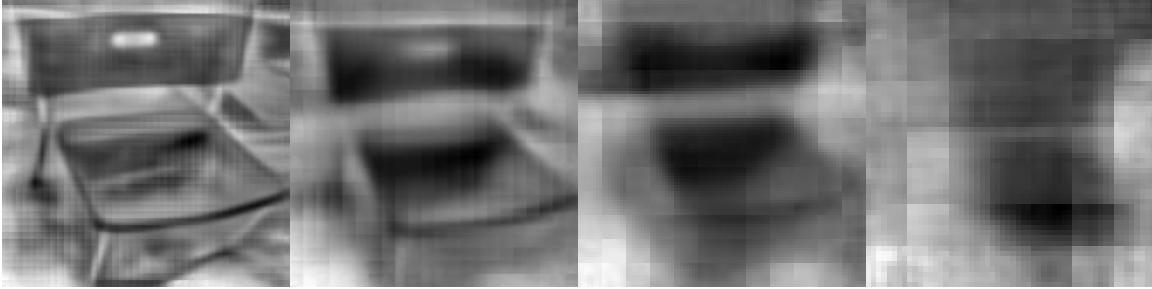


Figure 4-3: Our inversion algorithms are sensitive to the HOG template size. Larger templates are easier to invert since they are less invariant. We show how performance degrades as the template becomes smaller. Dimensions in HOG space shown: 40×40 , 20×20 , 10×10 , and 5×5 .

4.2.1 Reconstruction Error

We consider the inversion performance of our algorithm: given a HOG feature y , how well does our inverse $\phi^{-1}(y)$ reconstruct the original pixels x for each algorithm? Since HOG is invariant up to a constant shift and scale, we score each inversion against the original image with normalized cross correlation. Our results are shown in Table 4.1. Overall, exemplar LDA does the best at pixel level reconstruction.

4.2.2 Visualization Benchmark

While the inversion benchmark evaluates how well the inversions reconstruct the original image, it does not capture the high level content of the inverse: is the inverse of a sheep still a sheep? To evaluate this, we conducted a study on Amazon Mechanical Turk. We sampled 2,000 windows corresponding to objects in PASCAL VOC 2011. We then showed participants an inversion from one of our algorithms and asked users to classify it into one of the 20 categories. Each window was shown to three different users. Users were required to pass a training course and qualification exam before participating in order to guarantee users understood the task. Users could optionally select that they were not confident in their answer. We also compared our algorithms against the standard black-and-white HOG glyph popularized by [7].

Our results in Table 4.2 show that paired dictionary learning and direct optimization provide the best visualization of HOG descriptors for humans. Ridge regression

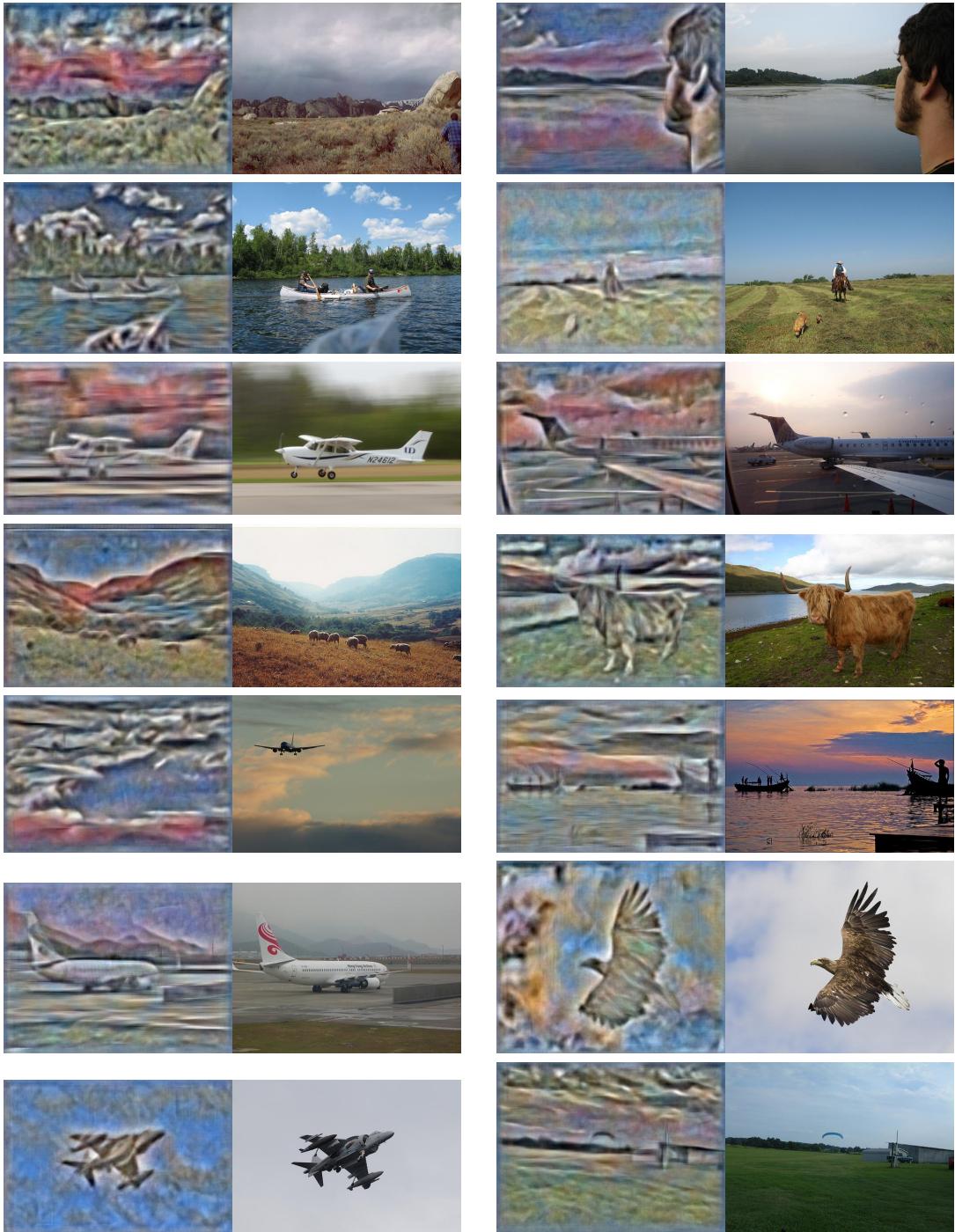


Figure 4-4: Color inversions of PASCAL images on outdoor scenes. Left is our inverse and right is the original image.

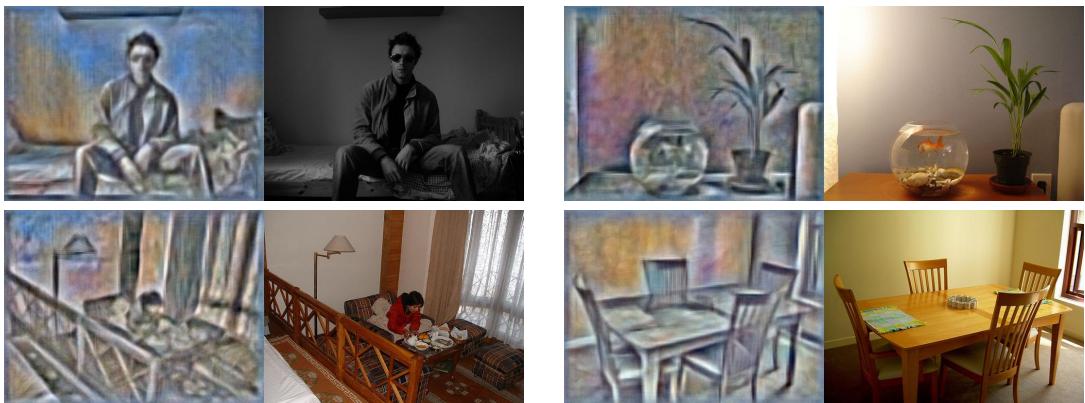


Figure 4-5: Color inversions of PASCAL images on indoor scenes. Left is our inverse and right is the original image.

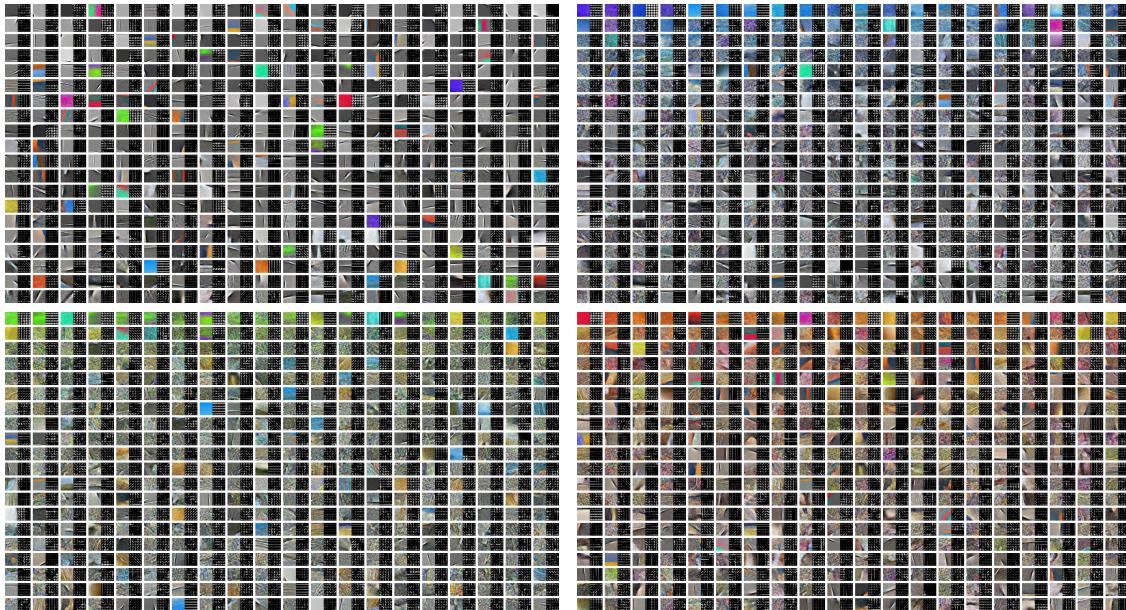


Figure 4-6: Visualization of the learned paired dictionary for inverting HOG to RGB images.

Category	ELDA	Ridge	Direct	PairDict
aeroplane	0.634	0.633	0.596	0.609
bicycle	0.452	0.577	0.513	0.561
bird	0.680	0.650	0.618	0.638
boat	0.697	0.678	0.631	0.629
bottle	0.697	0.683	0.660	0.671
bus	0.627	0.632	0.587	0.585
car	0.668	0.677	0.652	0.639
cat	0.749	0.712	0.687	0.705
chair	0.660	0.621	0.604	0.617
cow	0.720	0.663	0.632	0.650
table	0.656	0.617	0.582	0.614
dog	0.717	0.676	0.638	0.667
horse	0.686	0.633	0.586	0.635
motorbike	0.573	0.617	0.549	0.592
person	0.696	0.667	0.646	0.646
pottedplant	0.674	0.679	0.629	0.649
sheep	0.743	0.731	0.692	0.695
sofa	0.691	0.657	0.633	0.657
train	0.697	0.684	0.634	0.645
tvmonitor	0.711	0.640	0.638	0.629
Mean	0.671	0.656	0.620	0.637

Table 4.1: We evaluate the performance of our inversion algorithm by comparing the inverse to the ground truth image using the mean normalized cross correlation. Higher is better; a score of 1 is perfect. In general, exemplar LDA does slightly better at reconstructing the original pixels.

Category	ELDA	Ridge	Direct	PairDict	Glyph	Expert
aeroplane	0.433	0.391	0.568	0.645	0.297	0.333
bicycle	0.327	0.127	0.362	0.307	0.405	0.438
bird	0.364	0.263	0.378	0.372	0.193	0.059
boat	0.292	0.182	0.255	0.329	0.119	0.352
bottle	0.269	0.282	0.283	0.446	0.312	0.222
bus	0.473	0.395	0.541	0.549	0.122	0.118
car	0.397	0.457	0.617	0.585	0.359	0.389
cat	0.219	0.178	0.381	0.199	0.139	0.286
chair	0.099	0.239	0.223	0.386	0.119	0.167
cow	0.133	0.103	0.230	0.197	0.072	0.214
table	0.152	0.064	0.162	0.237	0.071	0.125
dog	0.222	0.316	0.351	0.343	0.107	0.150
horse	0.260	0.290	0.354	0.446	0.144	0.150
motorbike	0.221	0.232	0.396	0.224	0.298	0.350
person	0.458	0.546	0.502	0.676	0.301	0.375
pottedplant	0.112	0.109	0.203	0.091	0.080	0.136
sheep	0.227	0.194	0.368	0.253	0.041	0.000
sofa	0.138	0.100	0.162	0.293	0.104	0.000
train	0.311	0.244	0.316	0.404	0.173	0.133
tvmonitor	0.537	0.439	0.449	0.682	0.354	0.666
Mean	0.282	0.258	0.355	0.383	0.191	0.233

Table 4.2: We evaluate visualization performance across twenty PASCAL VOC categories by asking Mechanical Turk workers to classify our inversions. Numbers are percent classified correctly; higher is better. Chance is 0.05. Glyph refers to the standard black-and-white HOG diagram popularized by [7]. Paired dictionary learning provides the best visualizations for humans. Interestingly, the glyph is best for bicycles.

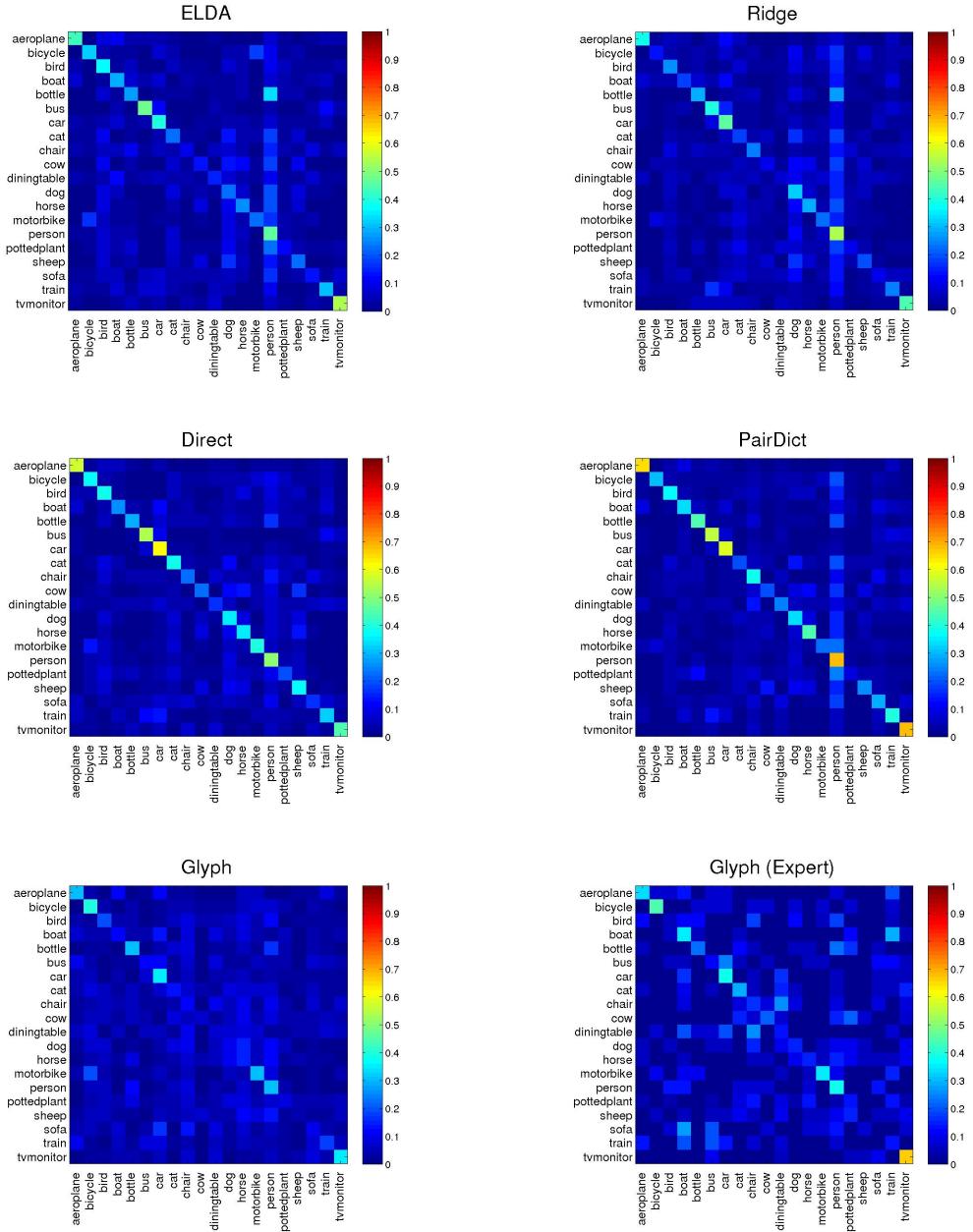


Figure 4-7: We show the confusion matrices for each of our four algorithms as well as the standard HOG black-and-white glyph visualization. The vertical axis is the ground truth category and the horizontal axis is the predicted category.

and exemplar LDA performs better than the glyph, but they suffer from blurred inversions. Human performance on the HOG glyph is generally poor, and participants were even the slowest at completing that study. Interestingly, the glyph does the best job at visualizing bicycles, likely due to their unique circular gradients. Overall, our results suggest that visualizing HOG with the glyph is misleading, and using richer diagrams is useful for interpreting HOG vectors.

There is strong correlation with the accuracy of humans classifying the HOG inversions with the performance of HOG based object detectors. We found human classification accuracy on inversions and the state-of-the-art object detection AP scores from [13] are correlated with a Spearman’s rank correlation coefficient of 0.77. This result suggests that humans can predict the performance of object detectors by only looking at HOG visualizations.

Figure 4-7 shows the classification confusion matrix for all algorithms. Participants tended to make the same mistakes that object detectors make. Notice that bottles are often confused with people, motorbikes with bicycles, and animals with other animals. Users incorrectly showed a strong prior that the inversions were for people, evidenced by a bright vertical bar in the confusion matrix.

Chapter 5

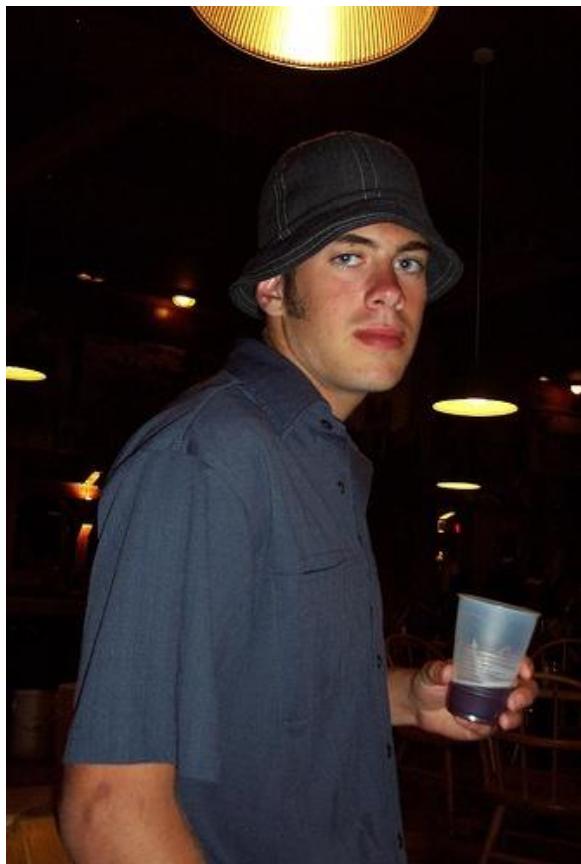
Understanding Object Detectors

We have so far presented four algorithms to visualize object detection features. We evaluated the visualizations with a large human study, and we found that paired dictionary learning provides the most intuitive visualization of HOG features. In this section, we will use this visualization to inspect the behavior of object detection systems.

5.1 HOG Goggles

Our visualizations reveal that the world that features see is slightly different from the world that the human eye perceives. Figure 5-1a shows a normal photograph of a man standing in a dark room, but Figure 5-1b shows how HOG features see the same man. Since HOG is invariant to illumination changes and amplifies gradients, the background of the scene, normally invisible to the human eye, materializes in our visualization.

In order to understand how this clutter affects object detection, we visualized the features of some of the top false alarms from the Felzenszwalb et al. object detection system [14] when applied to the PASCAL VOC 2007 test set. Figure 1-4 shows our visualizations of the features of the top false alarms. Notice how the false alarms look very similar to true positives. This result suggests that these failures are due to limitations of HOG, and consequently, even if we develop better learning algorithms



(a) Human Vision



(b) HOG Vision

Figure 5-1: HOG inversion reveals the world that object detectors see. The left shows a man standing in a dark room. If we compute HOG on this image and invert it, the previously dark scene behind the man emerges. Notice the wall structure, the lamp post, and the chair in the bottom right hand corner.



(a) Person



(b) Chair



(c) Car

Figure 5-2: We show the original RGB patches that correspond to the visualizations from Figure 1-4. We print the original patches on a separate page to highlight how the inverses of false positives look like true positives. We recommend comparing this figure side-by-side with Figure 1-4. Top left: person detections, top right: chair detections, bottom: car detections.

or use larger datasets, these will false alarms will likely persist.

Figure 5-2 shows the corresponding RGB image patches for the false positives discussed above. Notice how when we view these detections in image space, all of the false alarms are difficult to explain. Why do chair detectors fire on buses, or people detectors on cherries? Instead, by visualizing the detections in feature space, we discovered that the learning algorithm actually made reasonable failures since the features are deceptively similar to true positives.

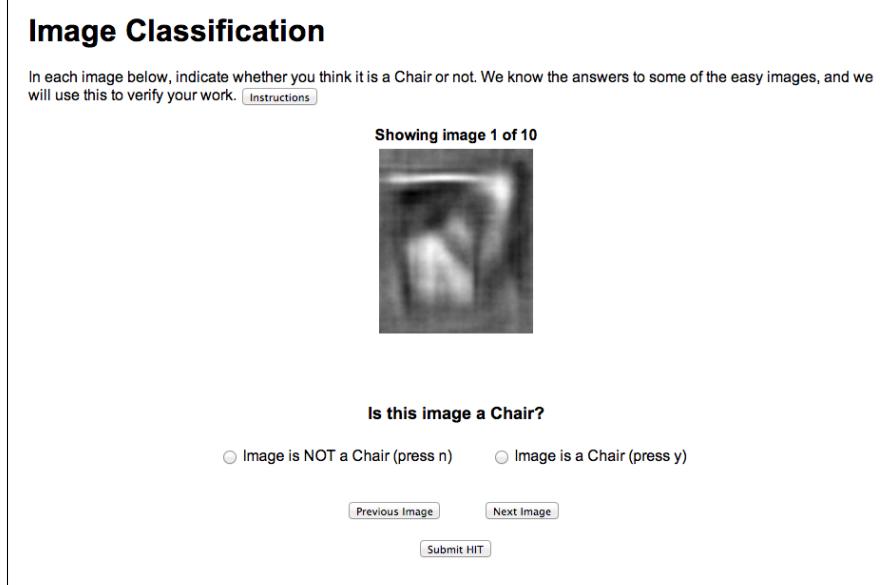


Figure 5-3: The Amazon Mechanical Turk interface for the Human+HOG detector.

5.2 Human+HOG Detectors

Although HOG features are designed for machines, how well do humans see in HOG space? If we could quantify human vision on the HOG feature space, we could get insights into the performance of HOG with a perfect learning algorithm (people). Inspired by Parikh and Zitnick’s methodology [30, 29], we conducted a large human study where we had Amazon Mechanical Turk workers act as sliding window HOG based object detectors.

We built an online interface for humans (see Figure 5-3) to look at HOG visualizations of window patches at the same resolution as DPM. We instructed workers to either classify a HOG visualization as a positive example or a negative example for a category. By averaging over multiple people (we used 25 people per window), we obtain a real value score for a HOG patch. To build our dataset, we sampled top detections¹ from DPM on the PASCAL VOC 2007 dataset for a few categories. Our dataset consisted of around 5,000 windows per category and around 20% were true positives.

¹Note that recall will now go to 1 in these experiments because we only consider windows that DPM detects. Consequently, this experiment can only give us relative orderings of detectors. Unfortunately, computing full precision-recall curves is cost prohibitive.

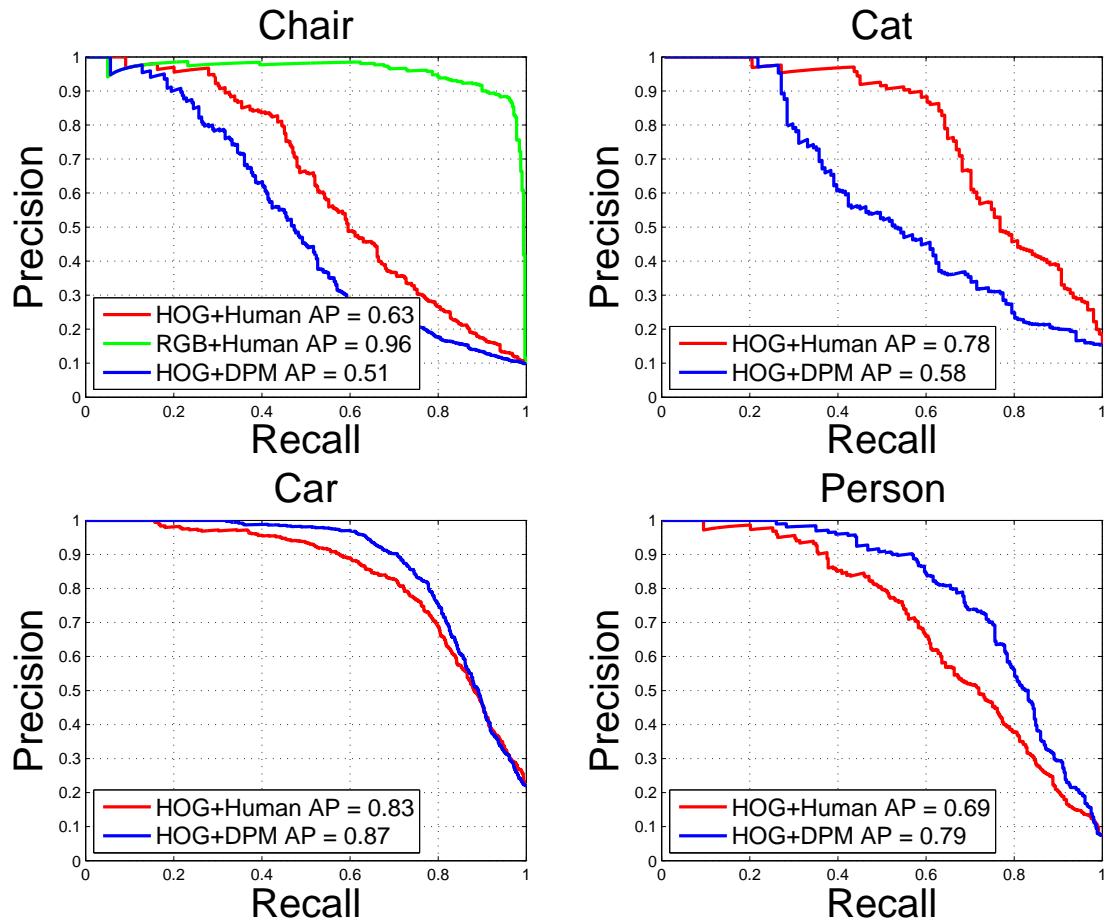


Figure 5-4: By instructing multiple human subjects to classify the visualizations, we show performance results with an ideal learning algorithm (i.e., humans) on the HOG feature space. Please see text for details.

Figure 5-4 shows precision recall curves for the Human+HOG based object detector. In most cases, human subjects classifying HOG visualizations were able to rank sliding windows with either the same accuracy or better than DPM. Humans tied DPM for recognizing cars, suggesting that performance may be saturated for car detection on HOG. Humans were slightly superior to DPM for chairs, although performance might be nearing saturation soon. There appears to be the most potential for improvement for detecting cats with HOG. Subjects performed slightly worse than DPM for detecting people, but we believe this is the case because humans tend to be good at finding other people in abstract drawings.

We then repeated the same experiment as above on chairs except we instructed users to classify the original RGB patch instead of the HOG visualization. As expected, humans achieved near perfect accuracy at detecting chairs with RGB sliding windows. The performance gap between the Human+HOG detector and Human+RGB detector demonstrates the amount of information that HOG features discard.

Our experiments suggest that there is still some performance left to be squeezed out of HOG. However, DPM is likely operating very close to the performance limit of HOG. Since humans are the ideal learning agent and they still had trouble detecting objects in HOG space, HOG may be too lossy of a descriptor for high performance object detection. If we wish to significantly advance the state-of-the-art in recognition, we suspect focusing effort on building better features that capture finer details as well as higher level information will lead to substantial performance improvements in object detection.

5.3 Tweaking HOG

In this section, we visualize a few tweaked variants of HOG. We show how HOG’s normalization step affects the feature, and we offer a new visualization of the texture features inside the HOG descriptor.

5.3.1 Normalization

A crucial step in computing HOG is normalizing each bin with its neighbors. Figure 5-5 shows inversions with and without normalization. While no normalization makes the inversions less noisy, they are also no longer invariant to lighting. This visualization confirms that the normalization step makes HOG robust to lighting changes.

5.3.2 Texture

A common implementation of HOG adds a texture based feature to each cell, an idea popularized by [14]. While the HOG glyph does not visualize this texture feature, our inversion offers one of the first visualizations of these cells. Figure 5-6 shows results where we invert from only the texture dimensions. While the inversions are predictably degraded, there is still significant information in the texture dimensions. Notably, the texture features primarily capture sharper gradients.

5.4 Interpolation in HOG Space

Since object detection is computationally expensive, most state-of-the-art object detectors today depend on linear classifiers. Figure 5-7 analyzes whether recognition is linear separable in HOG space by inverting the midpoint between two positive examples. Not surprisingly, our results show that frequently the midpoint no longer resembles the positive class. Since linear classifiers assume that the midpoint of any positive example is also a positive, this result indicates that perfect car detection is not possible with a single linear separator in HOG space. Car detection may be solvable with view based mixture components, motivating much recent work in increasing model complexity [26, 14].

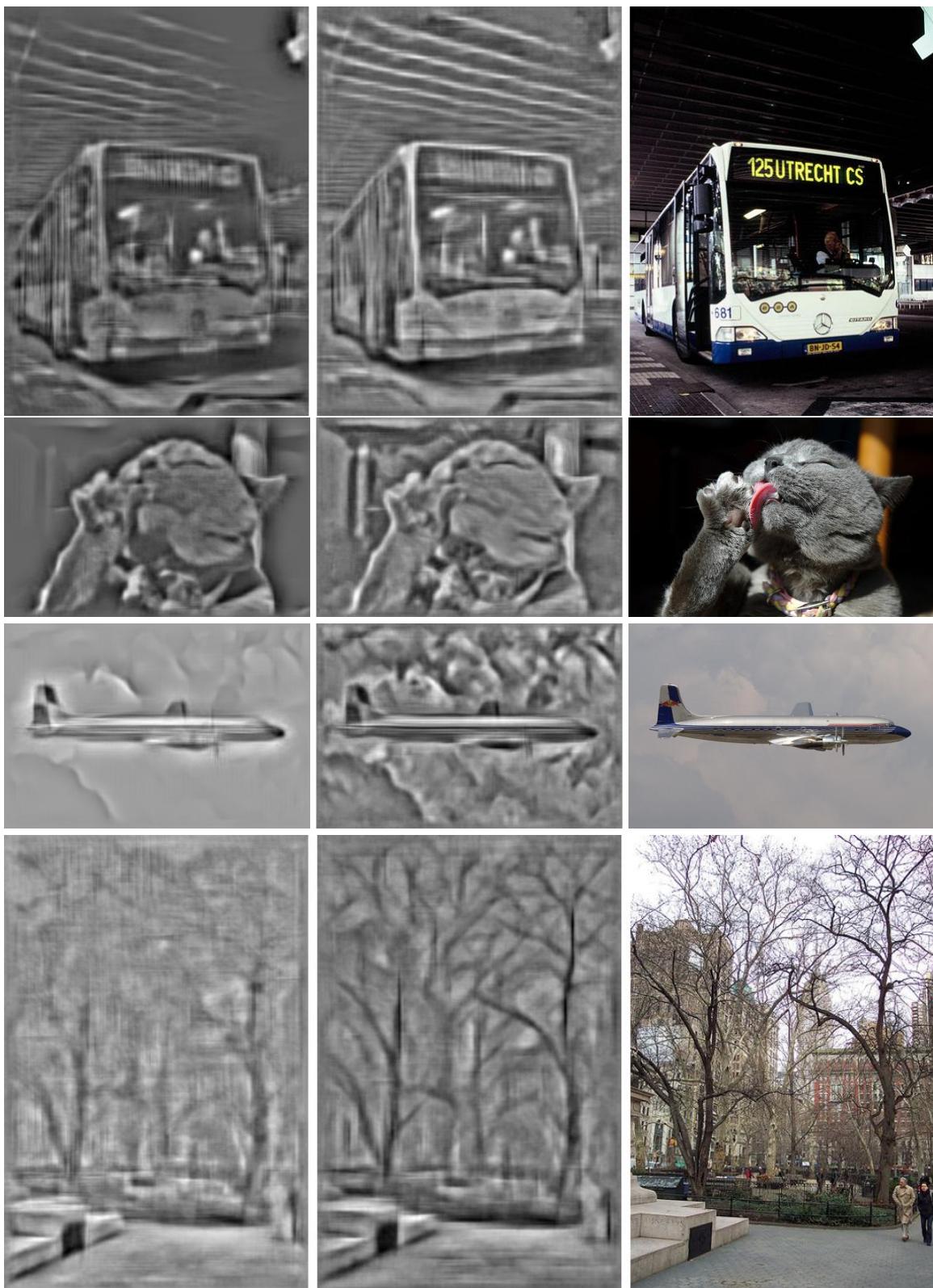


Figure 5-5: We compare HOG with and without normalization. Left: no normalization. Middle: with normalization. Right: original image. Notice how normalization increases HOG's robustness to lighting (rows 1 and 2) at the expense of extra noise (rows 3 and 4).



Figure 5-6: We invert only the texture features inside HOG and compare to the full HOG reconstruction. Left: texture only visualization. Middle: full HOG visualization. Right: original image. Even though the texture features are very low dimensional (4 dimensions per cell), there is still significant information stored inside.

5.5 Visualizing Models

Although our focus in this thesis to visualize feature descriptors, our algorithms are also able to visualize learned object models. In this section, we visualize a few models from popular object detectors.

5.5.1 Model Weight Visualization

We found our algorithms are also useful for visualizing the learned models of an object detector. Figure 5-8 visualizes the root templates and the parts from [14] by inverting the positive components of the learned weights. These visualizations provide hints on which gradients the learning found discriminating. Notice the detailed structure that emerges from our visualization that is not apparent in the HOG glyph. In most cases, one can recognize the category of detector by only looking at the visualizations.

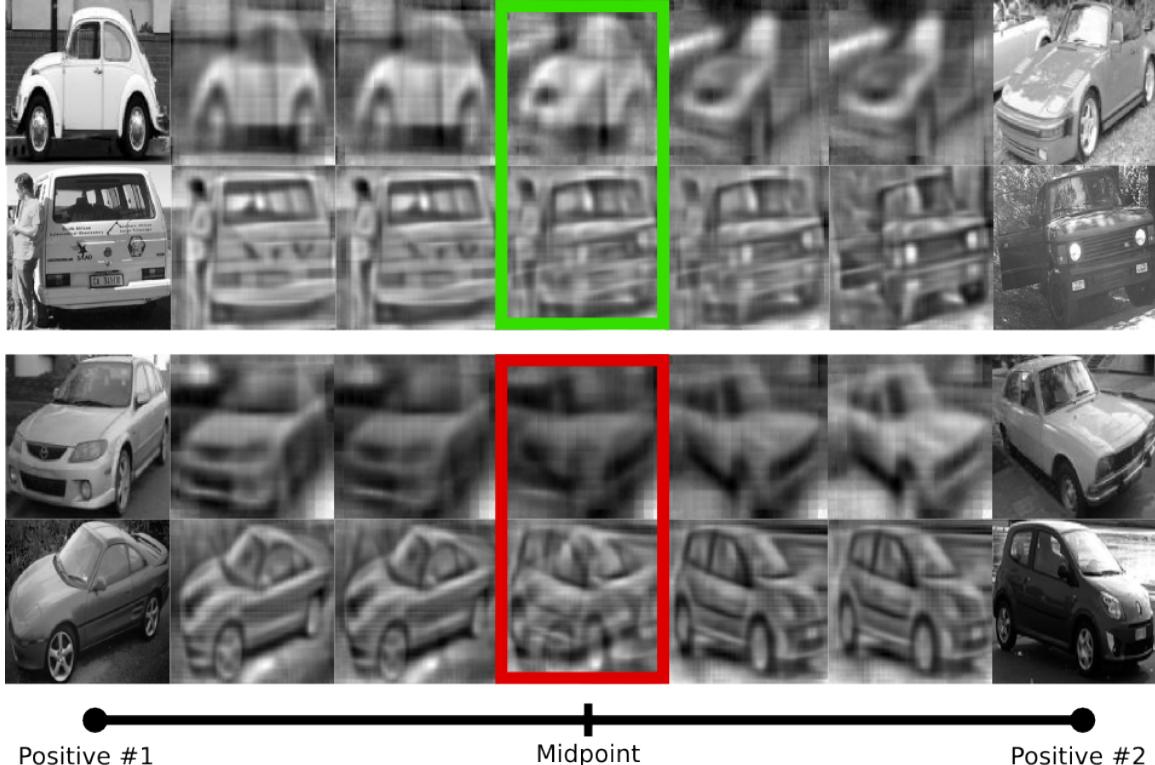


Figure 5-7: We linearly interpolate between examples in HOG space and invert its path. First two rows: occasionally, the interpolation of two examples is still in the positive class even under extreme viewpoint change. Last two rows: frequently, however, the midpoint is no longer the positive. This confirms that a single linear separator in HOG space is insufficient for perfect object detection.

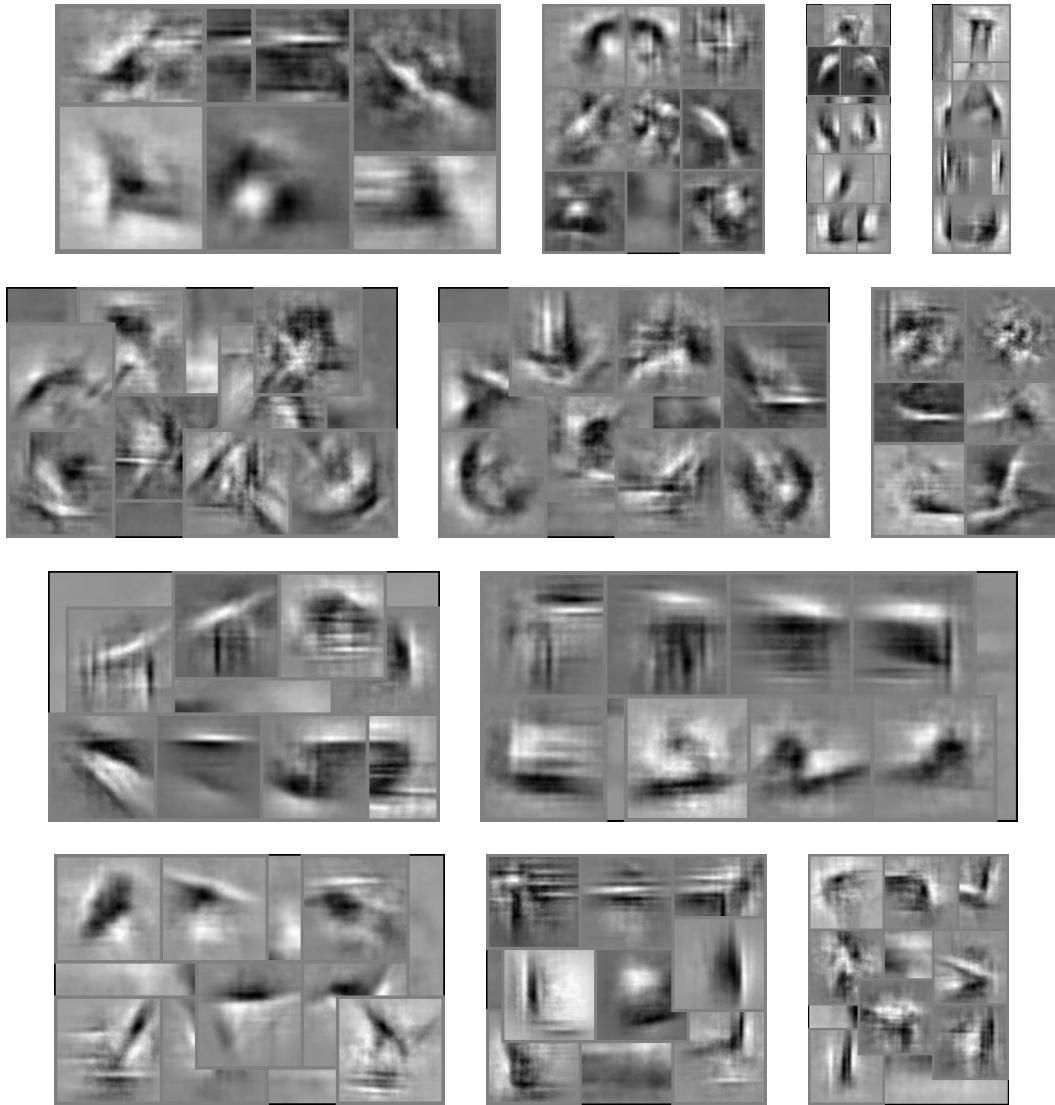


Figure 5-8: We visualize a few deformable parts models trained with [14]. Notice the structure that emerges with our visualization. First row: car, person, bottle, bicycle, motorbike, potted plant. Second row: train, bus, horse, television, chair. For the right most visualizations, we also included the HOG glyph. Our visualizations tend to reveal more detail than the glyph.

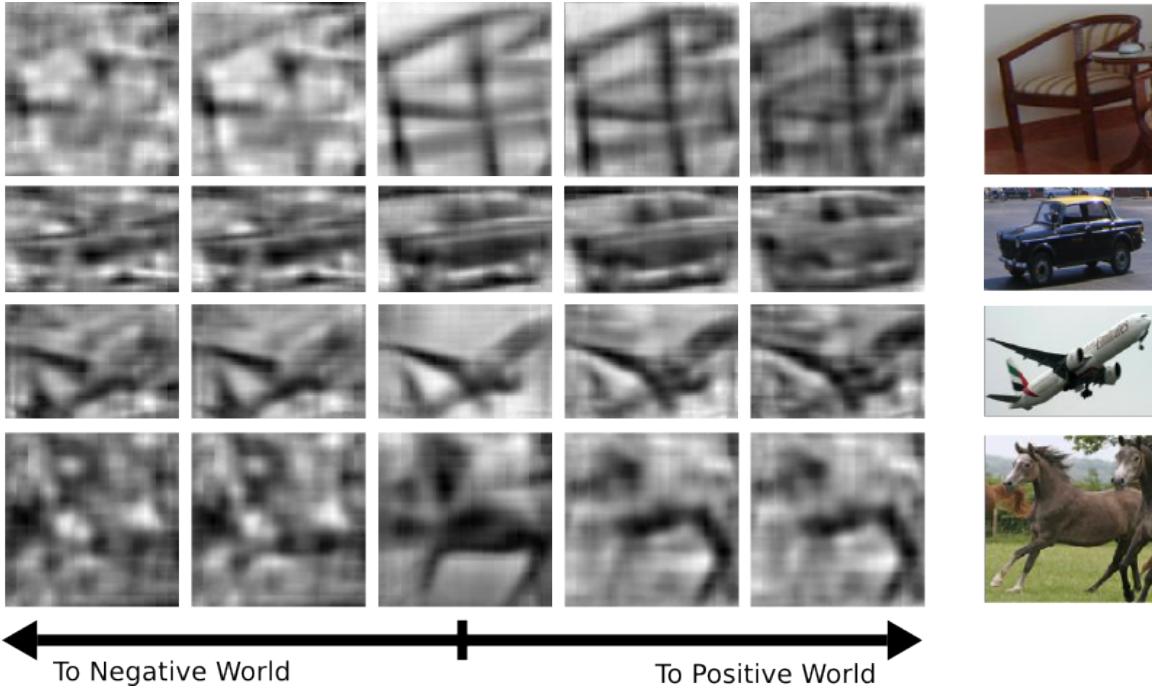


Figure 5-9: We train single component, linear SVM object detectors with HOG for a variety of categories and translate in HOG space orthogonal to the decision hyperplane. Moving towards the right is making the object more positive and to the left is making it more negative. The full color image on the right is the original image. Moving towards the positive world causes the discriminative gradients of the example to increase, and moving to the negative world causes the example to become more like background noise.

5.5.2 Super Objects

In Figure 5-9, we examine how the appearance of objects change as we make an object “more positive” or “more negative.” We move perpendicularly to the class decision boundary in HOG space. As the object becomes more and more positive, the key gradients become more pronounced, but if the object is downgraded towards the negative world, the object starts looking like noise.

5.6 Choice of Features

While HOG is the most popular feature for object detection today, it is not the only one. In a recent paper, Ren and Ramanan showed that using a Histogram of Sparse Codes (HSC) [31] in place of HOG can improve performance on object detection

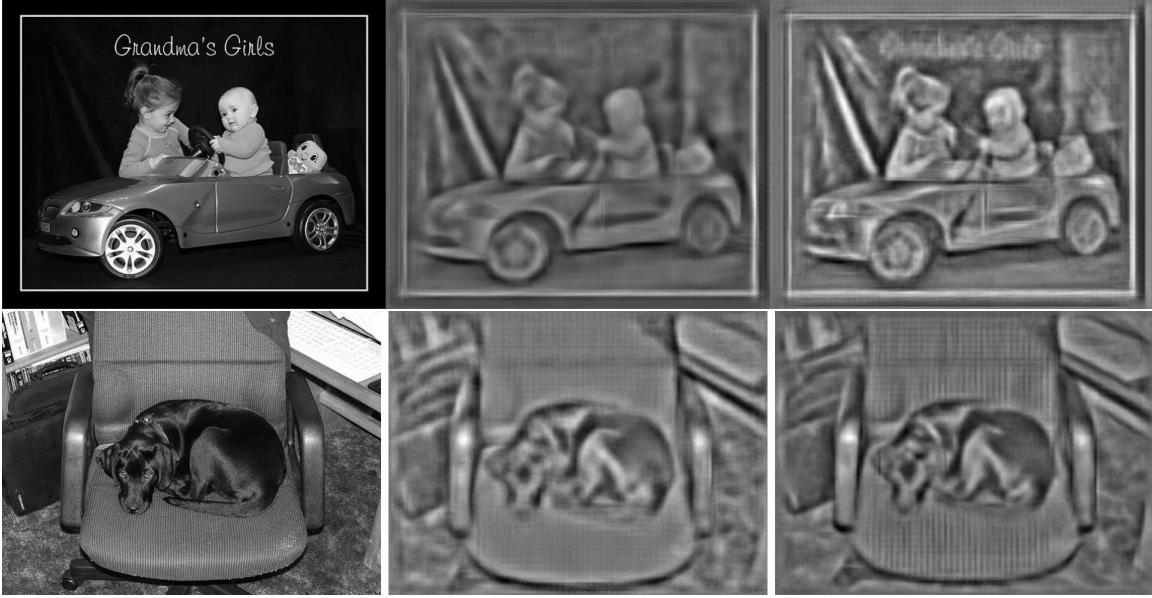


Figure 5-10: We visualize HSC and HOG features using a paired dictionary. Best viewed on the computer. Left: original image, middle: HSC visualization, right: HOG visualization. Notice how HSC discards image artifacts added during post processing (“Grandma’s Girls” is missing) and appears to blur high frequencies (stripes on the chair).

benchmarks. In this section, we compare visualizations of HSC with visualizations of HOG. To do this, we trained our paired dictionary to invert HSC features instead of HOG.

Figure 5-10 shows visualizations of HSC features on images from PASCAL VOC. In general, HSC visualizations are very similar to HOG visualizations, but they do reveal that HSC captures slightly different information than HOG. Firstly, HSC often discards image artifacts that are added during post-processing, such as timestamps or in-painted text, while HOG is highly sensitive to it. We hypothesize this is the case because the HSC feature is learned from natural images, and post-processing text is unnatural. Secondly, HSC tends to blur high frequencies, which we believe happens because the basis set is often small. Finally, HSC tends to capture less noise than HOG. This appears to arise from the lack of a normalization step in HSC, so insignificant gradients are not magnified.

Chapter 6

Conclusion

This thesis has presented four algorithms to visualize object detection features. Each of our algorithms have a variety of trade-offs: some are fast, some are non-parametric, some have better pixel reconstructions, and others have superior recovery of high-level semantics. We evaluated our algorithms with a large user study on Amazon Mechanical Turk and our results demonstrate that visualizing HOG with our algorithms provide both a more accurate and intuitive visualization for humans than the standard black-and-white HOG glyph. We then used these visualizations to examine the false alarms from a state-of-the-art object detector, and our experiments show that while many false alarms are clearly wrong in image espace, they are still reasonable failures since their features are deceptively similar to true positives. Our visualizations allow us to conclude that the features are to blame for the failures.

We believe visualizations can be a powerful tool for understanding object detection systems and advancing research in computer vision. The tools in this thesis allow a scientist to carefully inspect our feature spaces and perceive the world as an object detector sees it. Since object detection researchers analyze HOG glyphs everyday and nearly every recent object detection paper includes HOG visualizations, we hope more intuitive visualizations will lead to insights that advance the state-of-the-art in computer vision.

Bibliography

- [1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *CVPR*, 2012.
- [2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009.
- [3] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding.
- [4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *ECCV*, 2010.
- [5] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011.
- [6] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [8] E. d’Angelo, A. Alahi, and P. Vandergheynst. Beyond bits: Reconstructing images from local binary descriptors. *ICPR*, 2012.
- [9] M. Dikmen, D. Hoiem, and T. S. Huang. A data driven method for feature transformation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3314–3321. IEEE, 2012.
- [10] S.K. Divvala, A.A. Efros, and M. Hebert. How important are deformable parts in the deformable parts model? *Technical Report*, 2012.
- [11] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. *BMVC*, 2010.
- [12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.
- [13] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

- [14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.
- [15] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2188–2202, 2011.
- [16] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.
- [17] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. *ECCV*, 2012.
- [18] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. *ECCV*, 2012.
- [19] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *ECCV*, 2008.
- [20] K. Kavukcuoglu, P. Sermanet, Y.L. Boureau, K. Gregor, M. Mathieu, and Y. Le-Cun. Learning convolutional feature hierarchies for visual recognition. *NIPS*, 2010.
- [21] H. Lee, A. Battle, R. Raina, and A.Y. Ng. Efficient sparse coding algorithms. *NIPS*, 2007.
- [22] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T Freeman. Sift flow: dense correspondence across different scenes. In *Computer Vision–ECCV 2008*, pages 28–42. Springer, 2008.
- [23] L. Liu and L. Wang. What has my classifier learned? visualizing the classification rules of bag-of-feature model by support region detection. In *CVPR*, 2012.
- [24] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.
- [25] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009.
- [26] T. Malisiewicz, A. Gupta, and A.A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.
- [27] S. Nishimoto, A.T. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J.L. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 2011.
- [28] A. Oliva, A. Torralba, et al. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 2006.

- [29] D. Parikh and C L. Zitnick. The role of features, algorithms and data in visual recognition. In *CVPR*, 2010.
- [30] D. Parikh and C.L. Zitnick. Human-debugging of machines. In *Workshop on Computational Social Science and the Wisdom of Crowds, NIPS*, 2011.
- [31] X. Ren and D. Ramanan. Histograms of sparse codes for object detection.
- [32] A. Tatu, F. Lauze, M. Nielsen, and B. Kimia. Exploring the representation capabilities of the hog descriptor. In *ICCV Workshops*, 2011.
- [33] A. Torralba and A.A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011.
- [34] A. Torralba and A. Oliva. Depth estimation from image structure. *PAMI*, 2002.
- [35] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.
- [36] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Inverting and visualizing features for object detection. *arXiv preprint arXiv:1212.2278*, 2012.
- [37] S. Wang, L. Zhang, Y. Liang, and Q. Pan. Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis. In *CVPR*, 2012.
- [38] X. Wang, T.X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, 2009.
- [39] P. Weinzaepfel, H. Jégou, and P. Pérez. Reconstructing an image from its local descriptors. In *CVPR*, 2011.
- [40] J. Yang, J. Wright, T.S. Huang, and Y. Ma. Image super-resolution via sparse representation. *Transactions on Image Processing*, 2010.
- [41] X. Zhu, C. Vondrick, D. Ramanan, and C.C. Fowlkes. Do we need more training data or better models for object detection? *BMVC*, 2012.