Unsupervised Learning of Invariant Features Using Video

David Stavens and Sebastian Thrun
Stanford Artificial Intelligence Laboratory
Computer Science Department
Stanford, CA 94305-9010
{stavens,thrun}@robotics.stanford.edu

Abstract

We present an algorithm that learns invariant features from real data in an entirely unsupervised fashion. The principal benefit of our method is that it can be applied without human intervention to a particular application or data set, learning the specific invariances necessary for excellent feature performance on that data. Our algorithm relies on the ability to track image patches over time using optical flow. With the wide availability of high frame rate video (eg: on the web, from a robot), good tracking is straightforward to achieve. The algorithm then optimizes feature parameters such that patches corresponding to the same physical location have feature descriptors that are as similar as possible while simultaneously maximizing the distinctness of descriptors for different locations. Thus, our method captures data or application specific invariances yet does not require any manual supervision. We apply our algorithm to learn domain-optimized versions of SIFT and HOG. SIFT and HOG features are excellent and widely used. However, they are general and by definition not tailored to a specific domain. Our domain-optimized versions offer a substantial performance increase for classification and correspondence tasks we consider. Furthermore, we show that the features our method learns are near the optimal that would be achieved by directly optimizing the test set performance of a classifier. Finally, we demonstrate that the learning often allows fewer features to be used for some tasks, which has the potential to dramatically improve computational concerns for very large data sets.

1. Introduction

Many feature functions have been proposed in the literature. However, they are usually not tailored to a specific domain. While some of these features achieve excellent results across a broad range of applications, their performance for particular applications can be improved because they do not capture domain-specific invariances. Optimizing a feature function for a particular domain is challenging because most features have numerous parameters that effect their invariances, such as the standard deviation of Gaussians used for smoothing and the number of bins in histograms used for orientation calculation. These parameters can be set entirely by hand, using hand-labeled data and learning, or by mathematical processes such as warping an image patch while optimizing the feature value to be invariant. However, we believe those approaches have limitations. A completely manual approach or one involving hand-labeling can be costly or time consuming and may not produce optimal results. Patch warping and related methods tend to be synthetic, not capturing all the invariances in the data.

We present an algorithm that uses unsupervised machine learning to optimize feature invariances using video. Because our algorithm is entirely unsupervised and computationally efficient, it can handle huge amounts of data. Furthermore, because it is data driven, the invariances it learns are accurate for the specific application domain or data set. Our method has two key steps.

First, our method performs correspondence between successive video frames. For this step we use basic Harris corner features and the Lucas-Kanade pyramidal optical flow algorithm. This approach is well-known as a method for correspondence on video and works well despite its simplicity because high frame rate limits motion between frames. A small patch of pixels around each Harris corner feature is extracted and saved, resulting in many sets of patches. Each set contains multiple views of approximately the same location in the environment. The number of sets is bounded only by video length and availability of corner features to track.

The second step optimizes the higher-level features. Here, "higher-level" implies a feature of greater sophistication and complexity than Harris corners. Because SIFT and HOG are widely used in the literature, we focus on them in this paper. However, any type of feature may be used. It is well-established that a feature function should compute the same value when applied to the same real-world object

1

or type of object but be distinct for different objects. To reach this goal, we optimize feature parameters using an efficient, effective search procedure and scoring function. Our method requires no hand-labeling and can be completely automated for any task involving video.

To show that our optimized features capture the appropriate domain-specific invariances, we demonstrate that parameters optimized for one video stream outperform, on that stream, those optimized for another and vice-versa. In particular, we look at the performance of optimized versions of SIFT and HOG on two tasks: car detection in an urban environment and correspondence for 3D reconstruction of surgery-related imagery. We show that the version of SIFT optimized for car detection, on a car detection task, outperforms the version of SIFT optimized for the surgical task. We also show the converse is true; the surgical SIFT features outperform the car features on the surgical task. When we compare our parameters to the original highly-tuned parameters presented by Lowe, we find our performance is better for the car detection task and essentially the same for the correspondence task. We present similar results for HOG features. Furthermore, we analyze the effect of optimization on the rotational invariance property of SIFT.

In addition, the parameters our method learns are near the optimal achieved by directly optimizing the test set performance of a classifier. That is, iteratively, we train the classifier on a training set, evaluate its performance on the test set, and change feature parameters to improve test set performance. We find that our method, despite being entirely unsupervised, produces parameters near this optimal. Finally, we show that optimized parameters can allow fewer features to be used for comparable application performance. This has the potential to improve computational issues associated with very large data sets or high resolution images.

Our method does have limitations. For example, in highly repetitive environments or video with loops, the same object may be encountered multiple times and thus recorded in distinct patch sets. In such cases, optimizing for distinctness between patch sets could be sub-optimal. However, our experimental results indicate that for typical, non-pathological cases performance is excellent.

2. Related Work

Our work shares some themes with [27, 28]. [27, 28] extract patches from different views of an accurately reconstructed 3D scene. These patches are used for optimizing parameters for a class of features. However, there are key differences. First, this previous work learns parameters for particular feature functions but does not distinguish between parameters for *different applications*. We show that ideal parameters are highly application dependent even for a specific feature function. Our method of finding parameters handles this case natively. Second, [27, 28] rely on accu-

rate 3D reconstruction whereas our method requires only video of standard frame rate. We feel that video is more easily acquired in practice across a wide variety of applications and tasks. In contrast, an accurate 3D reconstruction can be challenging to achieve. Third, in [27, 28], the patches selected for optimization tend to correspond with interest points selected by the SIFT algorithm. This is due to the nature of their algorithm. Our method works well with any Harris corner feature – any location in the image with sufficient texture to avoid the aperture problem. Thus, our method can consider a much broader class of patches. Finally, we explicitly compare our learned parameters to a well-defined optimal, establishing a global sense of performance. See [28] for a recent literature review of the intersection of features and learning. Other work on features and learning includes [1, 18, 11, 12, 6, 26, 29].

In addition, several other aspects of this paper are rooted in the literature. Self-taught learning [19] has appeared previously as applied to continuous streams of data [23, 7]. The Harris features we use are due to [10] and the enhancements in [22]. We use Lucas-Kanade optical flow from [16] with the pyramidal modifications from [4]. SIFT is due to [15, 14] and HOG features are presented in [8]. We use the SIFT implementation from Lowe [13] and the HOG implementation from [4]. We focus on HOG and SIFT due to their wide use in the literature including [31, 9, 24, 30, 20] and [21, 5, 2, 17, 3], respectively.

3. Generating Sets of Patches

First, our algorithm generates sets of patches. Each patch in a set corresponds to a different view of approximately the same physical location. We generate these patch sets by tracking simple features frame-to-frame in standard frame rate (30 fps) video. Due to the relatively small amount of motion between frames, it is not difficult to achieve good correspondence. We use Harris corner features [10] as modified by Shi and Tomasi [22]. In the interest of making our results fully reproducible, we use the standard implementation found in the OpenCV Library [4]. Our algorithm tracks points using the well-known Lucas and Kanade optical flow method [16] modified with image pyramids, once again due to the implementation publicly available in [4].

An example of the corner-detection and tracking process is shown in Figure 1. The images are real data we use in the experiments (Section 5). The left-most image is the first in four consecutive frames that progress to the right. In the first frame, the center of each yellow box is a corner detected by [22, 4]. These corners are tracked to the four subsequent frames using the Lucas-Kanade pyramidal optical flow method [16, 4]. The green arrows show the motion of each corner feature into the next frame. This is not a prediction. The algorithm uses the subsequent image to perform the calculation. The pixels in each yellow box are extracted

Figure 1. An example of corner feature tracking with optical flow. The video sequence moves frame-to-frame from left to right. The center of each yellow box in the left-most (first) frame is a corner detected by [22, 4]. The green arrows show the motion of these corners into the next frame as determined with optical flow [16, 4]. This is not a prediction. The flow algorithm uses the next frame in the sequence for this calculation. The yellow boxes contain the patches extracted and saved for later optimization. The yellow path tails in the subsequent frames depict patch movement as estimated by the optical flow. In total, four patch sets are produced, each with four patches. They are shown in Figure 2. Our algorithm produces additional patches for these frames. We do not show them for a simpler illustration.

Figure 2. The extracted patch sets from Figure 1. The correspondence is not perfect, but satisfactory for good optimization and application-specific feature performance.
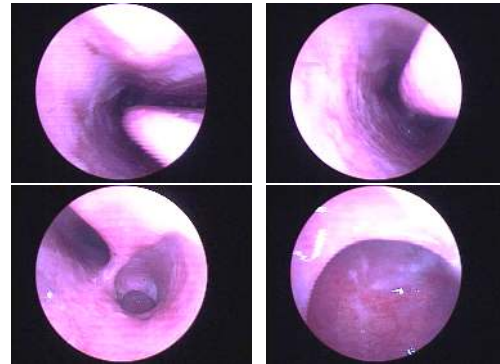

Figure 3. Four non-consecutive images from the surgical data set. The images are non-consecutive because the camera moves slowly, making the actual motion between frames small.

and saved for the optimization step. The yellow path tails show patch movement from previous frames as estimated by the flow algorithm. In this example, four patch sets of four patches each are produced. They appear in Figure 2.

Actually, our algorithm finds many more corners on these frames and is also tracking corners found in frames prior to the left-most frame. We omit these in the figure for a simpler illustration. In practice, on each frame, our algorithm continues tracking corners from previous frames and also calculates a new set of corners to begin tracking. Tracking of a particular corner terminates when the flow algorithm can no longer confidently solve for its new position. We also perform outlier rejection. Tracks are discarded if they are dramatically different from the average for the current frame. While this could potentially discard interesting data, we find it eliminates the majority of optical flow tracking errors, resulting in cleaner data for optimization.

Our algorithm uses corner features and optical flow. However any features and correspondence algorithm could be used. Corner features work with virtually any textured image patch. We argue that texture is a necessary condition for any algorithm seeking multiple views of the same area. Thus, our method is especially complete in the types of patches it uses. We feel this makes the optimization step particularly general, really capturing all possible im-

age patch types and variations. Furthermore, corners and optical flow are very fast to compute. Therefore our method could be used online to adapt features in real-time.

Corner features are not enough for many vision applications such as object recognition and correspondence for 3D reconstruction in the absence of dense data. This motivates the next step of our method where higher-level features are calculated and optimized for the tracked patches.

4. Optimizing Higher-Level Features

We define a "higher-level" feature to be anything more sophisticated than the basic corner features we use in Section 3. In our experiments, we use SIFT and HOG. However, the optimization process is identical for any higher-level feature. Let p be any single patch generated in Section 3. Then $p \in P \in \vec{P}$ where P is the set of all patches for a single environment location and $\vec{P}$ is the set of all patch sets. Since we do not optimize on pixels, we use p as the

feature vector associated with patch p. Let $\vec{\theta}$ be the set of parameters used by the feature function. Our optimization problem is then:

$$\arg_{\vec{\theta}} \sum_{\forall P \in \vec{P}} (\gamma \min \sum_{\forall p,q \in P} \frac{1}{|p|} \sum_{i=0}^{|p|} |p_i - q_i| \quad (1)$$

$$+ \max \sum_{\forall p \in P, \forall q \in (\forall (Q \in \vec{P}) \neq P)} \frac{1}{|p|} \sum_{i=0}^{|p|} |p_i - q_i|).$$

So, we are optimizing the parameters of $\vec{\theta}$. The $\min$ part of the optimization is minimizing the distance between the feature vectors generated for all patches within *one* set, all patches corresponding to the same physical location. The $\max$ part of the optimization is maximizing the distance between the feature vectors in *one* set and those in all other sets. Finally, the outer summation adds over all patch sets, turning our optimization for the *one* set into an optimization for all. That is, we are making all the patches that are actually the same as close as possible in feature value and all the patches that are different as different as possible in feature value. γ is a parameter trading off the extent to which we would rather make the same patches match as opposed to making differing patches not match. Simplifying without loss of generality we have:

$$\arg\min_{\vec{\theta}} \sum_{\forall P \in \vec{P}} (\gamma \sum_{\forall p,q \in P} \frac{1}{|p|} \sum_{i=0}^{|p|} |p_i - q_i| \quad (2)$$

$$- \sum_{\forall p \in P, \forall q \in (\forall (Q \in \vec{P}) \neq P)} \frac{1}{|p|} \sum_{i=0}^{|p|} |p_i - q_i|).$$

Since $|\vec{P}|$ is potentially enormous, in practice we replace $\forall (Q \in \vec{P}) \neq P$ with a small number of samples. In fact, we find experimentally that one is sufficient for good performance. Even with this approximation, the optimization is still challenging computationally. Thus we also assume the elements of $\vec{\theta}$ are conditionally independent. This allows us to perform coordinate descent in parameter space. However, we find experimentally that the space is fraught with local minima. Thus a more reliable method is to simply search the space. A search is tractable in this case because a feature can be generated rapidly (in about .0025 seconds) and the well-known "good" range of many parameters is not enormous, even at small discretization [13].

5. Experimental Analysis

To prove our algorithm learns application/data-specific invariances, we demonstrate two properties experimentally. First, we show our algorithm learns feature parameters that are distinct for different sets of data, *and* the parameters learned for one data set outperform, on that data set, parameters learned for a different data set and vice-versa. This establishes the idea that optimal feature invariances are application/data dependent and that our algorithm successfully adapts parameters to take advantage of this property. Second, we show that the parameters our algorithm learns are near the best for the application/data, despite the totally unsupervised nature of our algorithm.

5.1. Data Sets

We use two data sets for experimental analysis, Google Map's Street View and a surgical data set from inside a pig sinus that emulates many human medical tasks. Examples of data from Street View are seen in Figure 1 and non-consecutive examples of the surgical data can be seen in Figure 3. The surgical frames in the figure are non-consecutive because the camera moves slowly, limiting motion between frames. However, we use all frames for the optical flow tracking. Thus, for the surgical case, sets can have many patches. That is, because the camera does not move quickly, locations stay within view for many frames, and a large number of views/patches are saved. The data sets are very different in color, lighting, motion, texture, and many other factors.

5.2. Experimental Protocol

For each data set, patches were extracted as described in Section 3. Then for each data set and each feature type, optimization was performed as presented in Section 4. We used both HOG and SIFT features. For SIFT features, for full reproducibility, we used Lowe's own implementation. For HOG features, we used the publicly available implementation from [4]. All parameters from these implementations (8 for HOG, 11 for SIFT) were used for the optimization, except parameters varying descriptor length. An in-depth motivation of these parameters requires describing the entire HOG and SIFT algorithms and thus is beyond the scope of this paper. Note all steps up to this point are totally automated and unsupervised. We then used these optimized features in experimental applications to analyze their efficacy. We took care that the patches used for the optimization process were not used in either the training or testing sets. We used color images for HOG and grayscale for SIFT.

All of our experiments are framed as classification tasks. Of course, we used a distinct training set and test set. Reported accuracy is an average over 10 trials where the composition of training and testing sets were varied randomly. Error bars are standard deviations. For each image, our classifier samples features at evenly spaced locations on a grid as in [25]. For SIFT features, 3 scales were used. Each of these features was concatenated together in one (large!) vector describing the image. A standard SVM with a linear kernel was trained and tested in the standard fashion. This
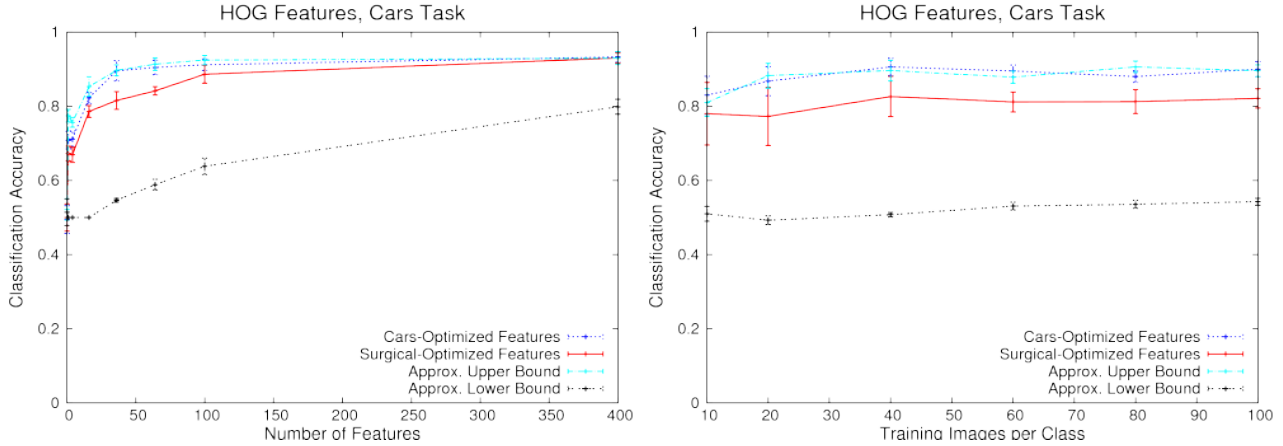
Figure 4. HOG feature optimization for the cars task. The features optimized for the cars task (blue line) outperform those optimized for the surgical task (red line). This verifies that our algorithm captures application-dependent feature invariances. The approximate theoretical best (light blue line) is only slightly better than our method. The approximate theoretical worst is reasonable for a 2-class problem.
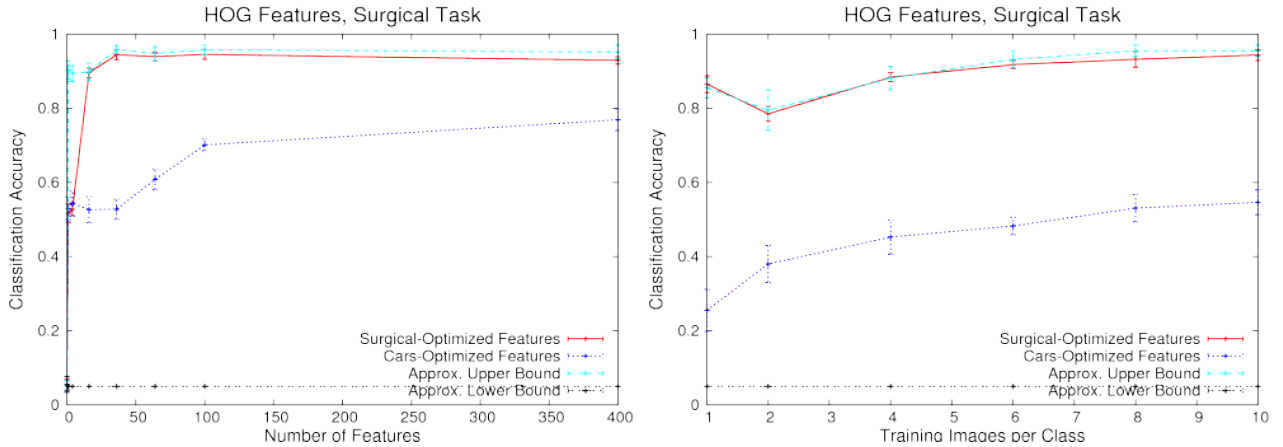

Figure 5. HOG feature optimization for the surgical task. The features optimized for the surgical task (red line) outperform those optimized for the cars task (blue line). This verifies our claim that our algorithm captures application-dependent feature invariances. The approximate theoretical best (light blue) is only slightly better than our method. The approximate theoretical worst is reasonable for a 20-class problem. Because this is a correspondence task on patches, reasonable performance for small numbers of features or images is not surprising.

is a simple classification scheme, not the most recent presented in the literature. Our goal, however, is to emphasize the quality of our features, not the learning algorithm. For full reproducibility, we used the SVM implementation publicly available in [4]. The vertical axis shows classification accuracy, the fraction of all images in the test set classified correctly. We use two different horizontal axes: the resolution of the grid (for SIFT, multiplied by scales) and the number of training examples per class. The number of features is important as it allows us to illustrate the efficacy of our features. For plots specifying the number of training examples, the resolution of the grid was 6 x 6.

For the Street View data set, we consider car detection. To establish ground truth, we manually labeled cars

with bounding boxes. To generate negative examples, we randomly cut bounding boxes from the image. We spot-checked the negative examples to verify cars were not present. For all plots varying the number of features, training and testing each occurred on 100 examples per class.

For the surgical data set, we consider patch correspondence for 3D reconstruction. We do not perform 3D reconstruction itself. To evaluate correspondence, we randomly selected 20 patch sets (20 P's from $\vec{P}$, see Section 4). For plots varying the number of features, we used 20 patches from each of these sets. The 20 were split evenly for training and testing. Thus our classifier performs correspondence as a 20-way classification problem. The classification of each patch in the test set was deemed its correspondence.

Intuitively, we expect correspondence to perform reasonably at finding a match even if only one point has been seen before. This explains why our plots for the surgical data show good performance even with one training example.

For SIFT features, we also plot the original/default parameters used in Lowe's implementation. These provide a basis of comparison. Additionally, for HOG features, we evaluated the optimality of our method. In particular, we searched all parameterizations, iteratively training and testing the classifier for each. We kept the parameterization with the best test set performance for a static number of features and training examples. Then, with that parameterization, we evaluated performance for all numbers of features and training examples as described above. As a check, we also retained the lowest test set performance. We present these bounds only for HOG features because the search is computationally intensive, and HOG features compute significantly faster in our implementation. To make computation tractable, we used the conditional independence assumption described in Section 4 for this search as well. Thus it is an approximate theoretical best. When searching, we averaged the performance of a parameter set over many trials. To save computation, we used fewer trials for the lower bound making it less accurate.

5.3. Experimental Results

Results for HOG features appear in Figures 4 and 5. Car detection results are in Figure 4. Note the difference in performance between the car-optimized features (blue line) and the surgical-optimized features (red-line). The car-optimized features offer significantly better performance, particularly with fewer features. These results establish that optimal feature parameters and invariances are application dependent and that our algorithm can learn these invariances in an unsupervised fashion. The performance of our method nears the theoretical best (light blue). At one data point, the theoretical best dips below the performance of our features. This may be due to the approximate manner in which we calculate the theoretical best parameters or that we only calculate them for one vertical "slice" of the plot, then use those parameters for the remainder of the horizontal axis trials. The theoretical worst performance, barely above chance for small numbers of features, seems reasonable for a 2-class problem. Figure 5 provides analogous results for the surgical task. Here, the features optimized for the surgical task (red line) outperform the car-optimized features (blue line) with analogous conclusions. Since this is a 20-class problem, a theoretical worst case of .05 seems reasonable. Because the surgical task is a correspondence task on small image patches, it is not surprising we achieve good performance with a small number of features and examples.

Results for the SIFT features appear in Figures 7 and 8

Figure 6. Image examples from the cars task that were misclassified with the surgical-SIFT but not the car-SIFT features.

and are analogous. As with HOG, the features optimized for the car task outperform, on the car task, the features optimized for the surgical task. Similarly, the features optimized for the surgical task outperform, on that task, the features optimized for cars. Note that, in all plots that vary the number of features, fewer of the task-specific features are needed for a particular performance level. This may be useful for algorithms on large data sets. Interestingly, our surgical-specific parameters for the surgical correspondence task produce results nearly identical to Lowe's. However, for the cars task, our parameters are significantly better. Lowe's original intent was to produce features for matching/correspondence. Evidently his extensive tuning well-optimized them for that task. Finally, Figure 6 shows selected images from the cars task that were classified correctly with the car-optimized SIFT but not the surgical-optimized SIFT features.

5.4. SIFT Rotational Invariance

For our applications and data sets, features should *not* be rotationally invariant. However SIFT features are naturally rotationally invariant. SIFT features achieve rotational invariance by histogramming gradients, selecting the most frequent bin as the orientation, and then extracting the local descriptor relative to that orientation. Our algorithm can turn off this functionality. Specifically, the number of bins in the histogram is a parameter we optimize. Rotational invariance is disabled if the number of bins becomes one. We also include an additional parameter that acts as a binary switch for rotational invariance. If the parameter is off, the orientation is set to a constant value, eliminating the invariance. (We need the binary parameter because we set a lower limit on the number of bins to make the interpolation step well-defined.) For both the cars and the surgical tasks, our optimization decreases the bins to their minimum, then flips the binary parameter off.

To demonstrate that rotational invariance is in fact disabled, consider SIFT feature matching using the surgical data set. We selected every 50th image in the set, omitting numbers 0 and 50 because they are too near the start, leaving 18 images in total. We rotated each image by 10 degree intervals. Rotation occurred about the center of the
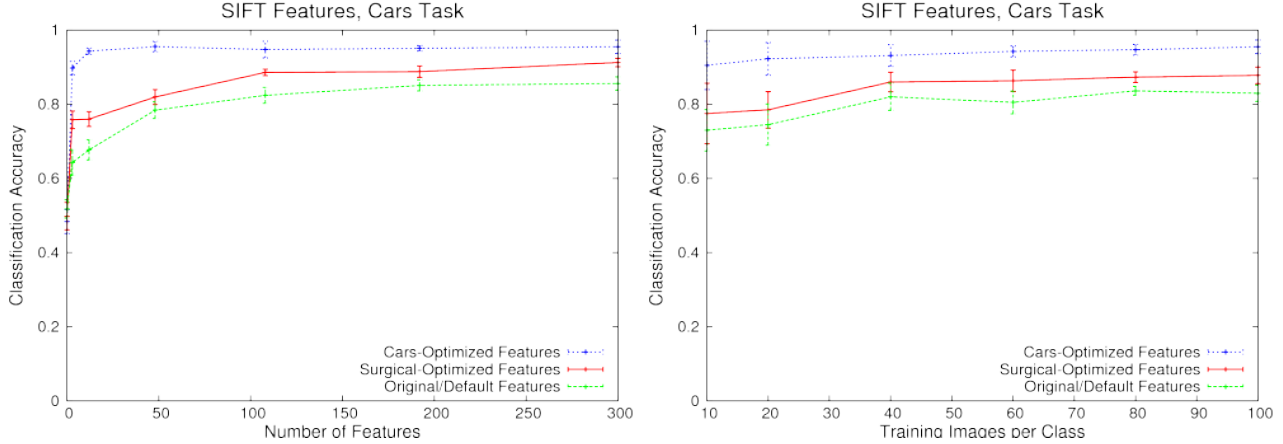
Figure 7. SIFT feature optimization for the cars task. The features optimized for the cars task (blue line) outperform those optimized for the surgical task (red line) and the parameters provided by Lowe (green line). This verifies our claim that our algorithm captures application-dependent feature invariances.
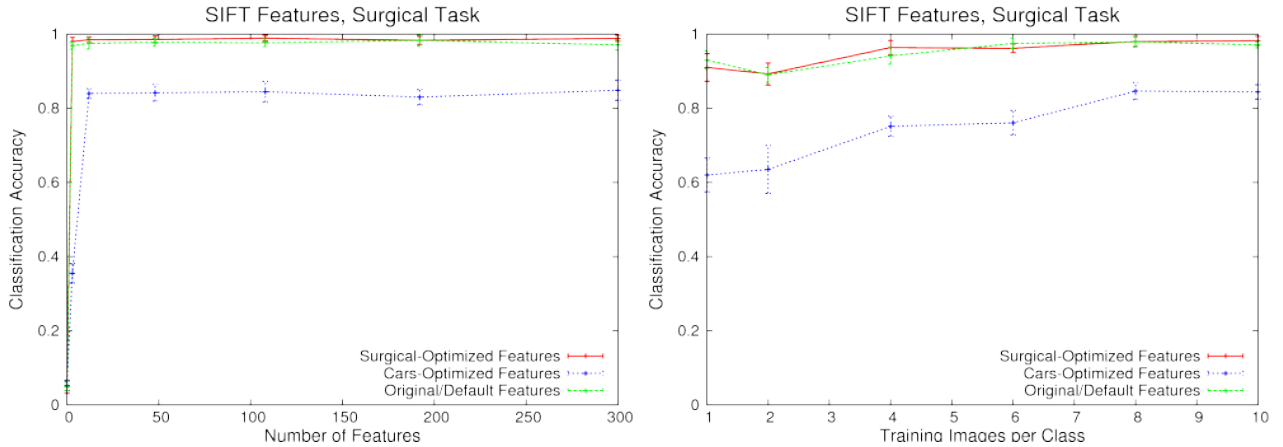

Figure 8. SIFT feature optimization for the surgical task. The features optimized for the surgical task (red line) outperform those optimized for the cars task (blue line). This verifies our claim that our algorithm captures application-dependent feature invariances. Intuitively, we expect correspondence to work well even with a single known point. This justifies the good performance presented even with one training example. The performance of our surgical-optimized features is nearly identical to Lowe's (green-line). Lowe's extensive optimization was geared toward matching/correspondence problems.

circle, not the image (see Figure 3). We computed a single SIFT feature in the center of each. For each image, we scored the difference between the descriptor at 0 degrees of rotation and after X degrees of rotation using the function:

$$\sqrt{\frac{1}{|f|} \sum_{i=0}^{|f|} (f_{angle_i} - f_{0_i})^2}. \tag{3}$$

That is, a single score is the square-root of the normalized sum of squared differences of each element of two SIFT feature vectors. The vectors are from a rotation of $angle$ and zero degrees, respectively. Figure 9 shows averages and standard deviations over 18 trials as a function of an-

gle. Clearly, our optimization has removed rotational invariance from the surgical-optimized SIFT features. This is the correct behavior given the invariances in our data. An analogous plot could be produced for car-optimized SIFT.

Rotational invariance is *not* the only invariance optimized for SIFT. There are many other parameters being optimized as discussed in Section 5.2. For example, the standard deviation of the Gaussian used for smoothing is important. Our optimization disables rotational invariance for both of the application-specific SIFT features. Thus, it does not account for the performance difference between them. It may when compared to Lowe's parameters which have rotational invariance.
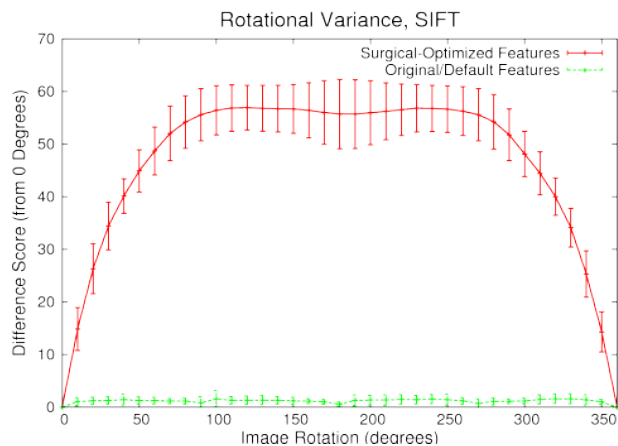
Figure 9. The effect of rotation on our surgical-optimized SIFT features. Our optimization removed rotational invariance. This is correct given the application-specific invariances.

6. Discussion

We present an algorithm that learns optimized features in an unsupervised fashion. Our features capture application or data-specific invariances, improving performance. We focus on optimizing SIFT and HOG features due to their widespread use in the literature. However our method will work for any feature function with parameters. For HOG features, we show our optimized versions are near the best possible for both the surgical and car tasks. For SIFT, our parameters for the surgical task produce performance very near the extensively researched parameters from Lowe. For the cars task, our SIFT parameters offer substantially better performance. Due to the widespread use of SIFT and HOG, we feel our method has the potential for immediate impact.

There are numerous opportunities for future work. Because our method is entirely unsupervised, we plan to explore how it scales to enormous data sets. For example, we could use every video on the web (or a very large subset) to learn optimized but general-purpose features. Rather than optimizing, we could learn a whole new class of features directly from pixels. We could also pursue additional applications. For example, numerous robotic systems would benefit from real-time feature optimization. It is also possible our core idea of unsupervised patch-extraction and optimization could be applied to other tasks in vision, such as camera calibration. Finally, we have alluded to the intersection of optimized features and feature compression and could explore it in more depth.

Acknowledgments

References

[1] M. Aharon, M. Elad, and A. M. Bruckstein. The K-SVD algorithm. In *Proc. of SPARSE*, 2005.

[2] T. Barfoot. Online visual motion estimation using fastslam with sift features. In *Proc. of the 2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[3] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. On the use of sift features for face authentication. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2007.

[4] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[5] M. Brown and D. G. Lowe. Recognising panoramas. In *Proc. of the Ninth IEEE Int. Conf. on Computer Vision (ICCV)*, 2003.

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[7] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[10] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of the 4th Alvey Vision Conf.*, pages 147–151, 1988.

[11] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[12] H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proc. of Neural Information Processing Systems (NIPS)*, 2009.

[13] D. Lowe. http://people.cs.ubc.ca/ lowe/keypoints/.

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[15] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the Seventh Int. Conf. on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[16] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 1981 DARPA Imaging Understanding Workshop*, pages 121–130, 1981.

[17] J. Luo, Y. Ma, E. Takikawa, S. Lao, M. Kawade, and B.-L. Lu. Person-specific sift features for face recognition. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[18] J. Mairal, M. Elad, and G. Sapiro. Sparse learned representations for image restoration. In *Proc. of the 4th World Conf. of the Int. Assoc. for Statistical Computing (IASC)*, 2008.

[19] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. of the 24th Int. Conf. on Machine Learning (ICML)*, 2007.

[20] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[21] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2001.

[22] J. Shi and C. Tomasi. Good features to track. In *Proc. of the 9th IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[23] D. Stavens and S. Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proc. of the Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2006.

[24] D. Tran and D. A. Forsyth. Configuration estimates improve pedestrian finding. In *Proc. of Neural Information Processing Systems (NIPS)*, 2007.

[25] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2007.

[26] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2008.

[27] S. Winder and M. Brown. Learning local image descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[28] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[29] L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770.

[30] W. Zhang, G. Zelinsky, and D. Samaras. Real-time accurate object detection using multiple resolutions. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2007.

[31] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.