Task Specific Local Region Matching

Boris Babenko, Piotr Dollár and Serge Belongie
Department of Computer Science and Engineering
University of California, San Diego
{bbabenko,pdollar,sjb}@cs.ucsd.edu

Abstract

Many problems in computer vision require the knowledge of potential point correspondences between two images. The usual approach for automatically determining correspondences begins by comparing small neighborhoods of high saliency in both images. Since speed is of the essence, most current approaches for local region matching involve the computation of a feature vector that is invariant to various geometric and photometric transformations, followed by fast distance computations using standard vector norms. These algorithms include many parameters, and choosing an algorithm and setting its parameters for a given problem is more an art than a science. Furthermore, although invariance of the resulting feature space is in general desirable, there is necessarily a tradeoff between invariance and descriptiveness for any given task. In this paper we pose local region matching as a classification problem, and use powerful machine learning techniques to train a classifier that selects features from a much larger pool. Our algorithm can be trained on specific domains or tasks, and performs better than the state of the art in such cases. Since our method is an application of boosting, we refer to it as ***Boo****sted Region* ***M****atching (BOOM).*

1. Introduction

Local region matching is a tool for discovering potential correspondences between two images, where the regions are centered on detected interest points. These correspondences can then be used in various computer vision problems such as pose estimation and wide baseline matching [17, 31, 28]. In the realm of object and scene recognition, local region matching has many benefits over global matching because it is more robust to occlusions and viewpoint changes [29]. Although local region matching has also been used for broad category classification (*e.g.* [12, 11, 24]), in this paper we limit ourselves to instances where correspondences are not ambiguous.

The standard paradigm for local region matching has two steps. The first step is interest point detection, which usually consists of locating the maxima of the response of some interest/saliency function. More recent detection algorithms are able to return the canonical scale and orientation of the region, in addition to the location (for a recent survey, see [26]).

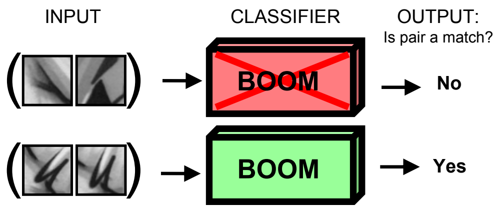
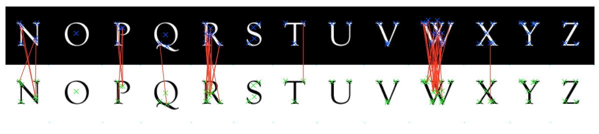


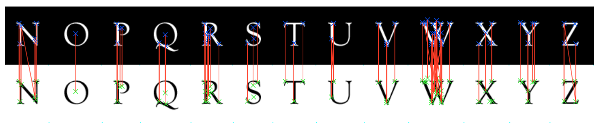
Figure 1. Framing the correspondence problem as binary classification. Rather than designing a new local region descriptors for a specific task, we train a classifier using labeled examples of matching regions. The classifier takes as input a pair of image patches, one extracted from each image, for which we would like to know whether they are in correspondence.

The second step is to find correspondences between the detected points. A descriptor is usually computed for each point, using information about the image patch centered on that point. These descriptors are designed to be invariant to various geometric and photometric transformations. Nearest neighbors in descriptor space are then putative correspondences. Although a lot of work has gone into designing descriptors that are robust to image transformations, there is necessarily a tradeoff between invariance and descriptiveness. For example, if you wanted to match features between images of a ‘6’ and a ‘9’, complete rotational invariance could result in incorrect matches. A second drawback is that although current algorithms are designed for general settings, when the domain becomes more specialized, the feature space must be tuned or completely re-engineered.

An illustration of this can be seen in Fig. 2 where we used the popular SIFT [22] descriptor to find matches between contrast reversed letter pairs. Because the SIFT descriptor is not engineered to handle this type of invariance, it performs poorly in this task. On the other hand, BOOM is able to learn the invariance from a small number of training images, and performs much better. Clearly, one could easily tweak the SIFT algorithm to make it work in this particular case. However, we argue that with more challenging data it



(a) SIFT



(b) BOOM

Figure 2. An illustrative example: a set of contrast reversed letter pairs is shown. Each image is 85×70 pixels. Each $\times$ is an interest point, and the red lines connect matching points. Since the letters are aligned, non-vertical lines are incorrect matches. (a) We used the SIFT descriptor to find correspondences between each letter pair, by thresholding L_2 distance between descriptors. (b) We trained BOOM on the first 13 letters of the alphabet, and used it detect the correspondences as shown. Although in this simple example it would be easy to tweak SIFT to raise its performance, doing so for a more challenging domain is not trivial.

is easier and more intuitive to provide training data than to alter the descriptors.

In this paper, we introduce a local region matching algorithm that can be trained for specific applications. Given labeled examples of correspondences, it can learn the types of invariance needed for that particular task. We divorce the detection and description steps, and concentrate only on the latter; any reasonable detector can be plugged into our system. First, we pose the correspondence problem as a simple binary classification task. We then design a classifier that computes simple features of a pair of patches, and outputs $+1$ or -1, depending on whether the two patches are a correspondence or not, respectively (see Fig. 1 for an illustration). The classifier can be trained with either human labeled or synthesized data. The learning framework we choose is the AdaBoost algorithm, which has been shown to be effective and efficient for object detection [32]. Hence, we refer to our method as **Boo**sted Region **M**atching (BOOM). We apply our algorithm to images in various domains, and compare our method to the SIFT descriptor, which we consider to be a good representative of general purpose descriptors. As a baseline, we also include the performance of the raw pixel values as a descriptor (we refer to this descriptor as PIXEL).

2. Previous Work

The literature on invariant image region detection and description methods is quite rich, and there are advances made every year. For a recent survey of point detectors please refer to [26], and for a recent survey of descriptors please refer to [25]. The latter finds the SIFT [22] descriptor to be the best in many tasks, and this algorithm has become widely used in many computer vision applications.

Several recent works use machine learning methods to train a saliency classifier on human labeled data. In [9] an edge detector is trained using images that have been labeled by humans. Similarly, in [20], a saliency detector is trained using human gaze data. Other methods, such as [15] and [11], construct class specific saliency detectors.

Finally, there have also been a few works that pose the problem of point matching as a classification or optimization problem. In [23] an optimal linear combination of simple distance functions is computed for the task of nearest neighbor search for patches of an image. Another method, described in [21], poses point matching as a multiclass classification problem, where each point on a training image of an object is considered a class. Extra examples of each class are synthesized by perturbing the original image. Though this method gives good results and is ideal for the case where one particular object is of interest, it is unable to generalize to unseen objects or scenes. In [30], an algorithm called BoostPro is used to find a set of projections in SIFT feature space that are explicitly sensitive to similarity pairs that have been labeled and provided as training data. In [34], a discriminative similarity function is learned for the purpose of accurate motion estimation. Similarly, in [18] a similarity function is learned for faces. Though our method is similar, we address the general problem of correspondence, rather than a specific application. Finally, the approach in [33] uses learning to find a an optimal set of parameters for a generalized SIFT descriptor by maximizing the area under the ROC curve for the training data. The learning aspect of this approach is similar in spirit to ours, though its generality is limited by the structure of the SIFT-based model.

3. Boosted region matching

In this section we describe how to solve the correspondence problem using binary classification, and give the specifics of the learning framework and features we use.

3.1. Local region matching as binary classification

For our purposes, we define an interest point detector as some function $\mathcal{F}(I) = (l_1, l_2 ... l_n)$, where I is an image and l_i is is the i^{th} interest point. The information for each interest point could be as minimal as 2D location $l_i = (x_i, y_i)$, or as rich as $l_i = (x_i, y_i, \theta_i, \sigma_i)$, where θ_i is the dominant orientation, and σ_i is the dominant scale of the local neighborhood centered around the point. In this sense, there is a gray line between detection and description since dominant orientation and scale could be considered as part of the descriptor.

Given a set of interest points in two images, we look at every possible pair of points, deciding whether each pair is a putative correspondence. Note that we do not enforce a 1-to-1 mapping, nor do we constrain the number of points in the images to be the same. Using the information returned by a detector, we crop out two small patches of a fixed size around those points. Patches can also be cropped at the correct orientation and scale (if this information is available from the detector), and then transformed to a fixed size. Given two patches (p_L, p_R) we train a classifier such that:

$$h(p_L, p_R) = \begin{cases} +1 & \text{if } p_L \text{ and } p_R \text{ are in correspondence} \\ -1 & \text{otherwise} \end{cases}$$

The training data is then a set of patch pairs and their labels. We compute a set of features for the patch pairs, and train a statistical classifier over these features. The test data has a natural imbalance towards negative pairs, since most of the possible point pairs in two images are not in correspondence. This imbalance also arises in object detection, where a classifier is applied to every patch in an image, and positive patches are rare (*e.g.* face detection). We use a learning framework that is commonly used for object detection, and is designed to deal with this issue.

3.2. Features

The AdaBoost algorithm [14] paired with simple Haar type features has been used effectively for various object detection and recognition tasks. In particular, Viola & Jones [32] introduced a method to efficiently compute such features and created a cascaded classifier for real-time face detection. Our approach extends this work to the local region matching domain.

Unlike the object detection problem, our features must be computed over a pair of patches, rather than a single patch. We extend the standard Haar features to fit our task. First, we discuss a set of patch features that operate on a single image patch, and then describe how to combine a pair of patch features into a scalar pair feature.

3.3. Patch features

The patch features come in two types: sum and histogram. First, we define an image channel as $g(I)$, where I is either the grayscale image or one of the RGB color channels, and g is any function that takes an image as input and returns an image of the same size. Examples of channels include gradient magnitude, response map of an edge detector, or the RGB channels themselves (see Fig. 4 for a plot of which image channels are chosen by our system for different experiments). Let p be an array of channels computed over some patch, indexed by c. We define $sum(p[c], r_i)$ to be the sum of pixel values of $p[c]$ inside rectangle r_i. A sum-type patch feature with $|r|$ rectangles is then defined as:

$$S(p) = \frac{\sum_{i=1}^{|r|} w_i sum(p[c], r_i)}{\sum_{i=1}^{|r|} |w_i| area(r_i)}$$

The denominator is a normalization term to ensure that S is between -1 and 1. Each sum-type feature is therefore parameterized by a channel index c, a set of rectangles r, and a set of real-valued weights w for the rectangles. Similar features were used in [32] over grayscale images, and in [9] over various image channels. In our experiments we use the following image channels for sum-type features: R,G,B color channels (when available), brightness, gradient magnitude, $\cos$ and $\sin$ of the gradient angle.

The histogram-type features are inspired by work in [35], [8] and [22], that use histograms of oriented gradients (HOG). To construct such a histogram, gradient magnitude values are dropped into bins according to the gradient orientation. To generalize this notion, we define $hist(p[c], p[c'], r_i)$ to be a histogram of pixel values of $p[c]$ inside rectangle r_i, binned by the pixel values of $p[c']$ (in the case of HOG, $p[c]$ is gradient magnitude, and $p[c']$ is gradient orientation). We define a histogram-type patch feature as:

$$H(p) = \sum_{i=1}^{|r|} w_i hist(p[c], p[c'], r_i)$$

In our experiments we use the HOG channel pair, as well as histograms of hue (c and c' both index into the hue channel).

Both the sum and $hist$ functions can be efficiently computed via the integral image trick, as was introduced in [32] for sums and [27] for histograms. The integral image is computed once for each image, and the sum of any rectangle in the image can subsequently be computed with only 4 memory accesses.

3.4. Pair features

Our pair features are divided into two types as well. As would be expected, each pair feature in our system is composed of two patch features (for sake of clarity, we refer to these as the left and right patch features). Thus, a sum-type pair feature is composed of two sum-type patch features $S_L(p)$ and $S_R(p)$, and is computed as follows:

$$f(p_L, p_R) = |\alpha S_L(p_L)^k - \beta S_R(p_R)^k|$$

This feature is a simple generalization of the L_k norm. For example, if the coefficients α and β are 1, then $k = 1$ corresponds to the L_1 norm and $k = 2$ to the squared L_2 norm. Setting α or β to unequal values has the effect of altering the relative normalization of the two patch features.

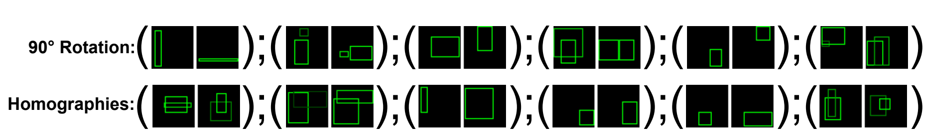
Similarly, a histogram-type pair feature is composed of two histogram-type patch features $H_L(p)$ and $H_R(p)$. We compute this feature as follows:

$$f(p_L, p_R) = \|H_L(p_L) - H_R(p_R)\|_2$$

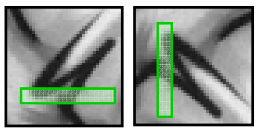
While other histogram metrics could be used, we chose L_2 distance for its simplicity.

3.5. Learning framework

The above features are plugged into an AdaBoost cascaded classifier, similar to what is described in [32]. At each iteration the AdaBoost algorithm greedily chooses a weak classifier with the minimum error. When the weak



(a)



(b)

Figure 3. (a) This figure shows a visualization of the features that were chosen by BOOM when trained on 90° rotations and on a range of homographies (see Section 4.2). Brighter rectangles have higher weight. To make this figure less crowded, we chose not to show the image channel or type associated with each feature. The features chosen for 90° rotations are intuitive – the left patch features are approximately 90° rotations of the right ones. For general homographies the features are more complex, but still display some intuitive structure. (b) A pair feature is shown overlayed with a pair of training image patches. For a positive patch pair the absolute difference or L_2 distance will be close to 0, since the rectangles cover the same pixels.

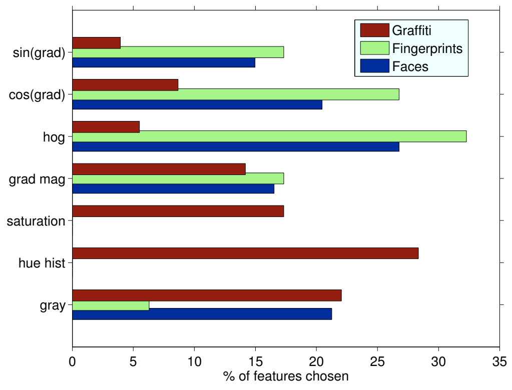


Figure 4. Image channels associated with the features that were chosen by BOOM for the three main experiments in Section 4. Notice that the fingerprints and face images were grayscale, and thus the hue histogram and saturation features were not chosen. Also, note that for the graffiti experiment the system does not choose as many features that rely on gradient angle. This is because we do not include any kind of orientation normalization, so for patch pairs that exhibit rotation the gradient directions do not have as much discriminative power.

classifiers each depend on only one feature, boosting inadvertently performs feature selection, resulting in an efficient classifier. The cascade is then a decision list, where each node is an AdaBoost classifier. An example is classified positive only if all the nodes in the cascade classify it as positive. This framework works well in situations where there are many more negative examples in the testing data, since most of the negative examples can be weeded out quickly by the first few nodes in the cascade. We refer the reader to [32] for a more detailed discussion.

The only difference between the framework used by Viola & Jones and ours is the choice of the weak classifier. While [32] uses a 1D stump classifier, we found that for our purposes a range classifier worked better. Taking three parameters $\theta_L, \theta_R \in \mathbb{R}, \beta = \{+1, -1\}$ and a feature value x, we define the range classifier as follows:

$$h_{\theta_L,\theta_R,\beta}(x) \;=\; \begin{cases} \beta & \text{if } x > \theta_L \;\&\; x < \theta_R \\ -\beta & \text{otherwise} \end{cases}$$

The algorithm to learn the parameters of this weak classifier from data is straightforward, and involves a search for the optimal thresholds θ_L and θ_R. The intuition behind this choice of weak learner is that for a pair of similar patches, the absolute difference or L_2 distance of the two patch features is close to 0 (see Fig. 3(b) for an example).

For each experiment we generate a large pool of pair features randomly. Due to the number of parameters, the number of possible features is incredibly large, and the pool we generate is only a small subset of all possibilities. To deal with this issue, we perform steepest descent search over the space of (α, β, k) on the top 100 sum-type features in every iteration of AdaBoost, and pick the best resulting feature, as suggested in [10]. This allows AdaBoost to explore an approximation of the entire space of features. Although giving this extra power to the classifier may theoretically result in overfitting, we have found that in our experiments this was not an issue.

Though we could constrain our pair features to be symmetric, making their left and right patch sides identical, we remove this constraint to make the system more flexible. However, we do constrain the left and right features to be parameterized by the same image channel index. An illustration of the flexibility of asymmetric features is shown in Fig. 3, where we ran a simple experiment to train BOOM on 90° rotations, and show some of the features chosen. We also show the features chosen when we trained BOOM on a wide range of homographies (see Section 4 for details). In both cases the chosen features display some intuitive structure.

We view these features as a set of ingredients that are commonly used in designing invariant descriptors. Instead of combining these ingredients by hand to fit some task, we feed them into a supervised learning framework along with labeled examples. Surely the design of these features was not effortless. Nevertheless, this design must only be done once; for each new task, the system learns how to combine the features automatically from labeled data. Though the features are simple and efficient to compute, our experiments show that they are robust enough to handle a wide variety of domains. Finally these features are easy to extend, either by adding more image channels, or by inventing new feature types.

3.6. Efficiency

At run-time, for two images with N and M interest points, respectively, our classifier gets invoked NM times. Though the cascade and integral image tricks significantly reduce the amount of computation needed, we can cut the costs even more if we consider the features themselves. Recall that each pair feature consists of a left and right patch feature. The left patch features get computed for patches from the first image only, and the right patch features get computed for patches from the second image only. Each pair feature is then either a difference or L_2 distance of the values of the left and right patch features. Therefore, rather than computing features for each pair, we simply precompute the left and right patch features for all the patches in the first and second image, respectively. When a pair feature is needed by the classifier, only a difference or L_2 distance is left to compute. Computing the patch features requires computation of different channels of the patch, so it is much more expensive than the L_2 distance, difference, or thresholding steps. Therefore, while the brute force approach results in $\mathcal{O}(MN)$ expensive steps, precomputing the responses cuts this down to $\mathcal{O}(MN)$ cheap steps and $\mathcal{O}(M+N)$ expensive steps.

This makes BOOM similar to standard methods of assigning descriptors to each point, except in our case, the descriptors used on the two images are not symmetric, since the right and left patch features need not be the same. Secondly, instead of using a vector norm to compute similarity, we use a slightly more complicated function which includes thresholding, subtraction and L_2 distance computations (all the parts of the final classifier returned by AdaBoost). Hence, in terms of the computational costs, our method is comparable to standard techniques used for local region matching. Another interesting advantage of our approach is that by controlling the depth of the AdaBoost cascade, the user can directly control the tradeoff between accuracy and computational cost, as is mandated by the application. It is not clear how one would do this with conventional descriptor based methods. In general, the run time for our un-optimized code for a pair of images is of the same order of magnitude as Lowe's SIFT code. For example, a pair of images from the experiment in Section 4.2 take a little over a minute to run. Note that these images have about 3,000 points each, so the classifier gets invoked close to 10^7 times. Training time was under 30 minutes for all experiments.

4. Experiments

In this section we present our results on three different experiments. Our goal was to choose experiments in a wide range of domains where correspondence detection is necessary. In particular we chose the following three data sets: (1) point matching in a planar scene under perspective distortion, which is useful for panorama stitching, wide baseline matching, etc, (2) fingerprint minutia matching, which is used in fingerprint recognition engines, and (3) point matching in faces with non-linear illumination changes, which is useful for face recognition and tracking.

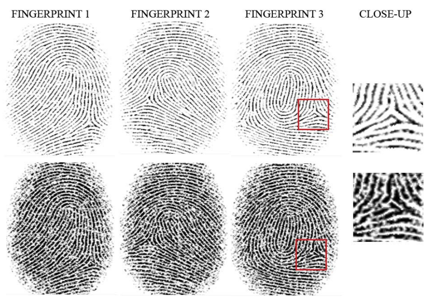


Figure 5. Fingerprints generated using the SFinGe tool. Top row displays a low pressure setting, bottom row display a high pressure setting. The last column shows a close-up, demonstrating the non-linear change in a small window. Each image is 290×242 pixels.

Because we treat the matching problem as classification, we evaluate performance by comparing ROC curves.

4.1. Methodology

In our experiments, we compare the performance of BOOM and the SIFT descriptor[1]. Also, as a baseline, we compute results for the simplest descriptor: the grayscale image patch of a fixed size around the point strung out into a vector (we refer to this method as PIXEL). We realize that more recent algorithms have claimed to outperform SIFT (*e.g.* [6, 19, 25]) and some older techniques are also popular (*e.g.* [4, 5]); we chose SIFT because it is widely used in the community and because we believe it is a good representative of generic local region descriptor algorithms.

As we mentioned before, any interest point detector can be plugged into our system. We decided to keep this constant in our experiments, and used the DoG detector described in [22], which comes built in to Lowe's SIFT code. The DoG detector returns (x, y, θ, σ) coordinates of each point. While SIFT uses all of this information, in our experiments we decided to discard scale and orientation when using the detector with BOOM to demonstrate the robustness of our features. The image channels used by BOOM, and all learning parameters except cascade depth and patch size are kept the same for all experiments. The cascade depth depends on the amount and complexity of data, and must be tuned slightly to avoid overfitting. The patch size we use is roughly 10% of the image diagnoal in each domain. Empirically we have found that the exact patch size does not significantly affect performance. The same patch size is used for PIXEL measurements.

Ground truth correspondences can either be provided manually or automatically. In our experiments, each image pair is related by some known homography H. For each point x_1 in the first image we go through each point x_2 in the second image and label the pair as a "true correspondence" if $\|x_1 - Hx_2\| \leq \rho$, where ρ is roughly 1% of the image size (similar criteria were used in [25]). The ROC curves shown are generated in the following ways:

[1] Made available by David Lowe at [1] .

- for BOOM we sweep through the threshold of the last layer of the cascade.
- for SIFT and PIXEL we threshold the L_2 distance between two points in SIFT feature space or pixel value space, respectively, and sweep through this threshold.

Our intention is to capture the raw performance of the descriptors. Many post-processing tricks exist to further refine or filter the potential correspondences. For example, one could limit the matches to be one-to-one by taking only the best match for each point, or run RANSAC [13] to robustly fit a homography or fundamental matrix to the correspondences. Alternatively, Lowe suggests using the ratio between the top two matches as a gauge for the quality of the match. We stay agnostic to these post-processing techniques, and use only the output of BOOM and L_2 distance between the SIFT and PIXEL descriptors when comparing results. Furthermore, we chose not to compare the performance of our algorithm within an end-to-end system, and instead isolate the matching accuracy. We assume that any end-to-end system that relies on correspondences would benefit from a higher matching accuracy.

4.2. Planar scene under perspective

In the first experiment we use the INRIA set of graffiti images, which are commonly used to evaluate region descriptors [2]. To train BOOM we took one graffiti image and synthesized a range of out-of-plane rotations (shown in Fig. 6(a) and (b)). This data sets also comes with computed homographies relating the planar scenes in the pictures shown in Fig. 7 to the reference image shown in Fig. 6(c). We use this set of images and the known homographies as ground truth for testing. Note that although we train with synthetic data, the testing is done on real images. Furthermore, the same classifier is used on all testing images.

4.3. Fingerprints

Fingerprint minutia matching is a well studied problem that has many applications in industry. The SFinGe tool is a fingerprint synthesis program that can generate realistic fingerprints that was designed to generate large amounts of test data for fingerprint recognition algorithms [7]. The tool allows the user to set various parameters such as geometric transformations, amount of noise and blur, the type of fingerprint (arch, right loop, left loop, whirl, and tented arch). Some examples of what this tool can produce are shown in Fig. 5. For this experiment we generated pairs of fingerprints with two different pressure parameters (low and high pressure)[3]. The change is purely photometric, and the fingerprint pairs are pixel-registered, meaning the homography $H = I$. Though this is convenient for gathering ground

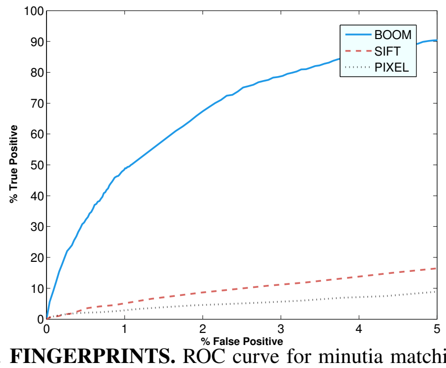


Figure 8. **FINGERPRINTS.** ROC curve for minutia matching of three test fingerprint pairs, which are shown in Fig. 5. BOOM is able to learn the invariance and outperforms both PIXEL and SIFT.

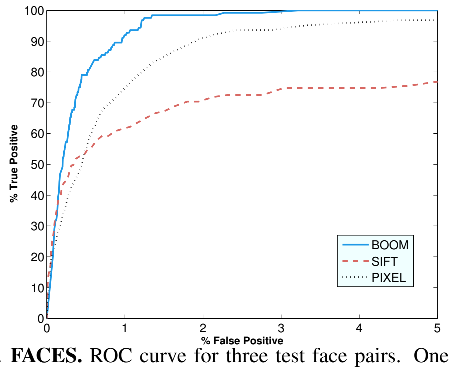


Figure 9. **FACES.** ROC curve for three test face pairs. One of the test pairs is shown in Fig. 10. Surprisingly, PIXEL outperforms the SIFT descriptor; BOOM outperforms both PIXEL and SIFT.

truth correspondences, it is certainly not necessary in the testing phase. Fig. 5 also shows a close-up of two corresponding patches from one of the test pairs. We see that the change in the patches is non-linear. We train BOOM on 12 fingerprint pairs of different types, and test on 3 novel pairs. In Fig. 4 we see that BOOM tends to choose image channels that rely on gradient direction. ROC plots for BOOM, SIFT and PIXEL are shown in Fig. 8. We see that SIFT has very poor performance in this domain because the image transformations are not well modeled by SIFT's design. Even with a relatively small amount of training data BOOM is able to get decent performance on this data set. Note that this performance is not up to par with state of the art finger print matching algorithms. A part of the problem is that the DoG detector that we used for the sake of consistency is not particularly effective in this task. Nevertheless, with minimal effort and simple generic features BOOM is able to outperform a general purpose descriptor.

4.4. Non-linear illumination change

The Yale Face Database B (Yale-B) [16] contains photographs of 10 subjects under different poses and lighting conditions. The photographs were taken with a special rig such that all photographs of each subject were taken within 2 seconds. This ensures that per pose, all photographs of each person are pixel-registered, and the homography $H = I$ as before. For this experiment we chose two dif-

[2]Dataset made available by the Visual Geometry Group at University of Oxfordat [2]

[3]A demo version of the software was kindly provided by the Biometric Systems Lab at University of Bologna.

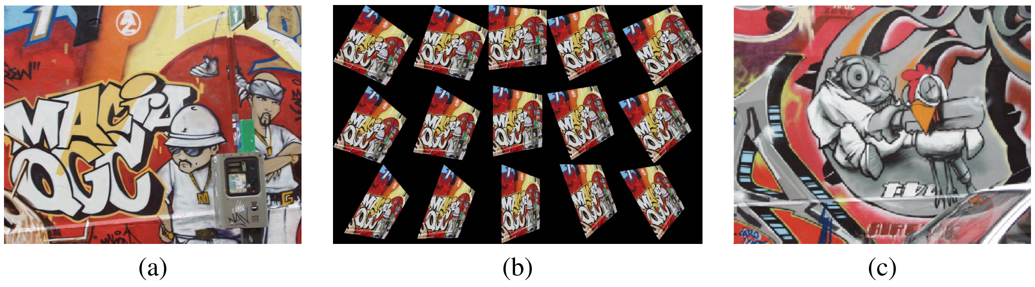


Figure 6. (a) The training reference image. (b) All of the synthesized out-of-plane rotations used in training. (c) The testing reference image. Images are 640×480 pixels. In all cases BOOM outperforms SIFT, and both outperform the PIXEL baseline.

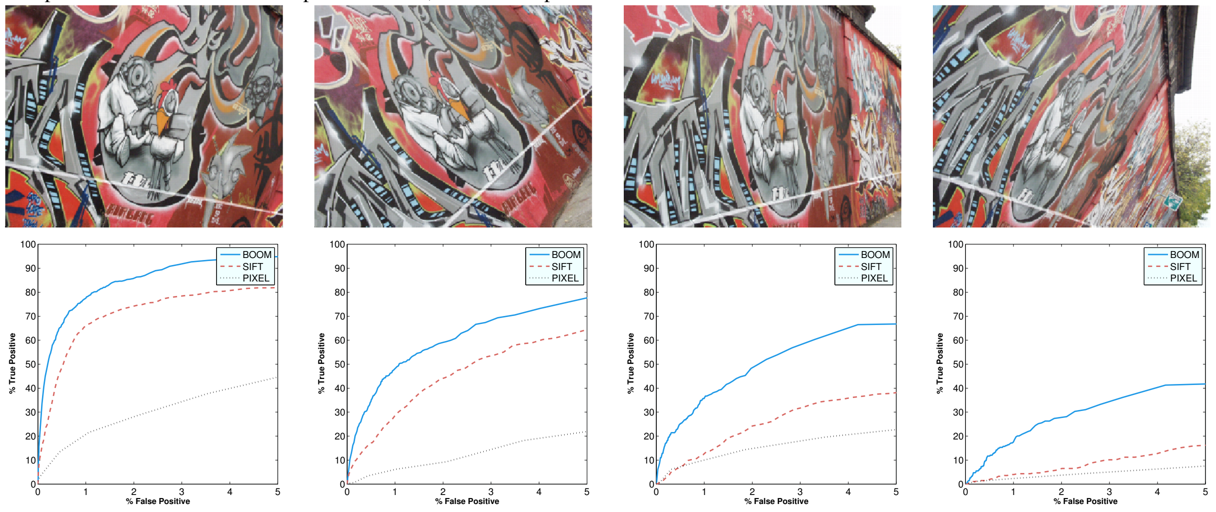


Figure 7. Top row: out-of-plane rotations of the image from Fig. 6(c). Bottom row: ROC curves comparing performance on each out-of-plane rotation. In all cases BOOM outperforms both PIXEL and SIFT.

ferent lighting angles, and three different poses. In order to learn illumination invariance we split the photographs into pairs with different illumination angles. We trained BOOM on 9 of the subjects (27 image pairs), and tested with the remaining subject (3 image pairs, one of which is shown in Fig. 10). ROC plots comparing BOOM, SIFT and PIXEL are shown in Fig. 9. Again, BOOM is able to learn the necessary invariance and achieves superior performance. Surprisingly, PIXEL outperforms SIFT despite its simplicity. Since there is geometric transformation between the images, the raw pixel values are a decent descriptor in this case.

5. Conclusions & Future Work

We have presented a method that uses machine learning techniques to create a classifier that decides if a pair of points in two images is a potential correspondence. While current approaches to solving the correspondence problem involve highly engineered feature sets, and many heuristics, our method learns from labeled examples and uses simple efficient features. Our approach can thus be trained for specific tasks or domains, where a general purpose descriptor would often fail.

The potential drawback in our system is the requirement of training data. However, there are a few factors that alleviate this problem. First, as we saw in Section 4.3, in many specialized domains realistic training data can be synthesized. Furthermore, during manual labeling, just a few correspondences are needed to compute the geometric transformation between two images. Once this transformation is known, many more ground truth correspondences can be inferred. Lastly, there has been an increased interest recently in active learning for vision applications [3], where the system asks a user to label only those examples which it will find useful most informative. This type of method has been shown to significantly reduce the amount of manual labor in building a training set.

6. Acknowledgements

This work was funded by NSF Career Grant #0448615, the Alfred P. Sloan Research Fellowship, NSF IGERT Grant DGE-0333451, and the Calit2/UCSD Summer Undergraduate Research Scholarship Program. Authors would also like to thank Nadav Ben-Haim for insightful discussions.

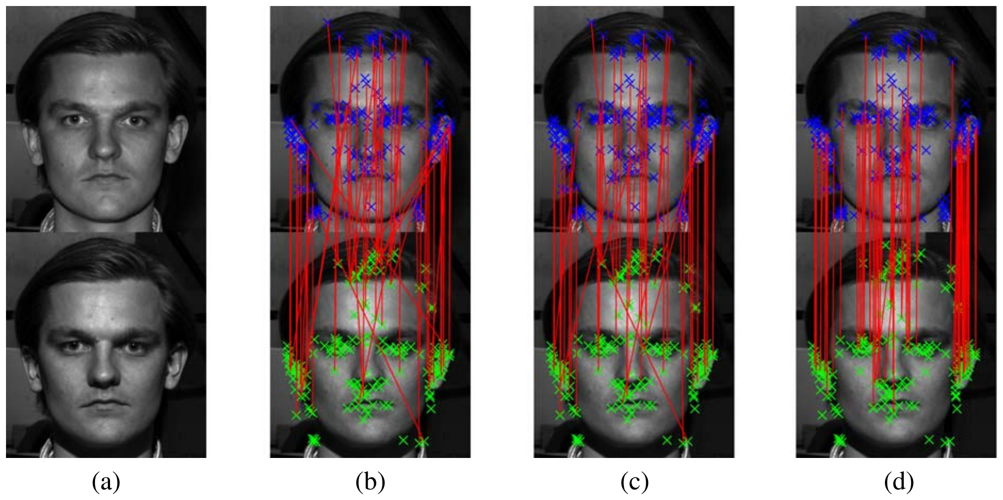


Figure 10. A test face image from the Yale Face Database with two different illuminations is shown in (a). We compute potential correspondences using (b) SIFT; (c) SIFT with a distance ratio threshold post-processing discussed in Section 4; (d) BOOM. Because the faces are aligned, non-vertical lines are incorrect matches. We see that BOOM outperforms both versions of SIFT. Each image is 350 $\times$ 290 pixels.

References

[1] http://www.cs.ubc.ca/ lowe/keypoints/.

[2] http://www.robots.ox.ac.uk/ vgg/research/affine/.

[3] Y. Abramson and Y. Freund. SEmi-automatic VIsual LEarning (SEVILLE): Tutorial on active learning for visual object recognition. *CVPR*, 2005.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching. In *NIPS*, pages 831–837, 2000.

[5] A. Berg and J. Malik. Geometric blur and template matching. In *CVPR*, pages I:607–614, 2001.

[6] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *CVPR*, pages I: 510–517, 2005.

[7] R. Cappelli. SFinGe: an Approach to Synthetic Fingerprint Generation. In *International Workshop on Biometric Technologies*, pages 147–154, June 2004.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[9] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, pages II: 1964–1971, 2006.

[10] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, June 2007.

[11] G. Dorko and C. Schmid. Maximally stable local description for scale selection. In *ECCV*, pages IV: 504–516, 2006.

[12] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages II: 264–271, 2003.

[13] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[14] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[15] D. Gao and N. Vasconcelos. Integrated learning of saliency, complex features, and object detectors from cluttered scenes. In *CVPR*, pages II: 282–287, 2005.

[16] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *PAMI*, 23(6):643–660, 2001.

[17] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000.

[18] M. Jones and P. Viola. Face recognition using boosted local features. *MERL Technical Report TR2003-025*, May 2003.

[19] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *CVPR*, pages II: 506–513, 2004.

[20] W. Kienzle, F. Wichmann, B. Schölkopf, and M. Franz. Learning an interest operator from human eye movements. In *CVPRW*, page 24, 2006.

[21] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28(9):1465–1479, September 2006.

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

[23] S. Mahamud and M. Hebert. The optimal distance measure for object detection. In *CVPR*, pages I: 248–255, 2003.

[24] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, pages I: 26–36, 2006.

[25] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, October 2004.

[26] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, October 2005.

[27] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR*, pages I: 829–836, 2005.

[28] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or 'How do I organize my holiday snaps?'. In *ECCV*, page I: 414, 2002.

[29] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–534, May 1997.

[30] G. Shakhnarovich. Learning task-specific similarity. *PhD Thesis, MIT*, 2006.

[31] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *IJCV*, 59(1):61–85, August 2004.

[32] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, May 2004.

[33] S. Winder and M. Brown. Learning local image descriptors. pages 1–8, 2007.

[34] S. Zhou, B. Georgescu, D. Comaniciu, and J. Shao. Boostmotion: Boosting a discriminative similarity function for motion estimation. In *CVPR*, pages II: 1761–1768, 2006.

[35] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages II: 1491–1498, 2006.