Affine-Invariant Local Descriptors and Neighborhood Statistics for Texture Recognition

Svetlana Lazebnik
Beckman Institute
University of Illinois, Urbana, USA
slazebni@uiuc.edu

Cordelia Schmid
Inria Rhône-Alpes
Montbonnot, France
Cordelia.Schmid@inrialpes.fr

Jean Ponce
Beckman Institute
University of Illinois, Urbana, USA
ponce@cs.uiuc.edu

Abstract

This paper presents a framework for texture recognition based on local affine-invariant descriptors and their spatial layout. At modeling time, a generative model of local descriptors is learned from sample images using the EM algorithm. The EM framework allows the incorporation of unsegmented multi-texture images into the training set. The second modeling step consists of gathering co-occurrence statistics of neighboring descriptors. At recognition time, initial probabilities computed from the generative model are refined using a relaxation step that incorporates co-occurrence statistics. Performance is evaluated on images of an indoor scene and pictures of wild animals.

1 Introduction

Texture representations that are invariant to a wide range of geometric and photometric transformations are desirable for many applications, including wide-baseline matching [9, 13, 15], texture-based retrieval in image databases [12, 14], segmentation of natural scenes [7], recognition of materials [16], and recognition of semantic texture categories, e.g., natural vs. man-made [3]. In this paper, we investigate a texture representation that is invariant to any geometric transformations that can be locally approximated by an affine model, from perspective distortions to non-rigid deformations.

Recently, several affine-invariant region detectors have been developed for the applications of wide-baseline matching, indexing, and retrieval [9, 15]. As demonstrated in our earlier work [4], such detectors can also make effective texture analysis tools. In this paper, we use a texture representation based on a sparse set of affine-invariant regions to perform retrieval and segmentation of multi-texture images. This task is more challenging than the recognition of single-texture images: instead of comparing distributions of local features gathered over a large field, we are forced to classify each local feature individually. Since it is not always possible to unambiguously classify a small image region, we must augment the local representation with a description of the spatial relationship between neighoring regions. The systems developed by Malik et al. [7] and Schmid [14] are examples of this two-layer architecture, with intensity-based descriptors at the first level and histograms of texton distributions at the second.

This paper describes a conceptually similar two-stage approach to texture modeling. The first stage consists in estimating the distribution of local intensity descriptors. Unlike most existing methods, which use fixed-size windows to compute these descriptors, ours employs *shape selection*: the area over which the descriptors are computed is determined automatically using an *affine adaptation* process [5]. We represent the distribution of descriptors in each class by a Gaussian mixture model where each component corresponds to a "sub-class". This generative model is used to assign the most likely sub-class label to each region extracted from a training image. At the second stage of the modeling process, co-occurrence statistics of different sub-class labels are computed over neighborhoods adaptively defined using the affine shape of local regions. Test images (which may contain multiple textures) are also processed in two stages. First, the generative model is used to assign initial probability estimates of sub-class membership to all feature vectors. These estimates are then refined using a relaxation step that incorporates co-occurrence statistics.

The most basic form of the modeling process is *fully supervised*, i.e., the training data contains only single-texture images. However, we show in Section 2.2 that a weaker form of supervision is possible: the training data may include unsegmented multi-texture images. In Section 3, we evaluate the proposed texture representation on two data sets. The first set consists of photographs of textured surfaces taken from different viewpoints and featuring significant scale changes and perspective distortions. The second set consists of images of animals whose appearance can be adequately modeled by texture-based methods.

2 Modeling Textures

2.1 Feature Extraction

At the feature extraction stage, our implementation uses an affine-adapted Laplacian blob detector based on the scale and shape selection framework developed by Lindeberg et al. [5, 6]. The detector begins by finding the locations in scale space where a normalized Laplacian measure attains a local maximum. Informally, the spatial coordinates of the maximum define the center of a circular "blob", and the scale at which the maximum is achieved becomes the *characteristic scale* of the blob. The second stage applies an *affine adaptation* process based on the second-moment matrix. The regions found by the detector are ellipses defined by $(\mathbf{p} - \mathbf{p}_0)^T M(\mathbf{p} - \mathbf{p}_0) \le 1$, where $\mathbf{p}_0$ is the center of the ellipse, and M is a 2×2 symmetric *local shape matrix* (see [5, 9] for details). We can *normalize* the patch defined by M by applying to it any transformation that maps the ellipse onto a unit circle. It can be shown that if two image patches are initially related by an affine transformation, then the respective normalized patches are related by a rotation [5, 9]. We eliminate this ambiguity by representing each normalized patch by a rotationally invariant descriptor.

The descriptors used in this work are *intensity domain spin images* [4] inspired by the spin images used by Johnson and Hebert [2] for matching range data. An intensity domain spin image is a two-dimensional histogram of brightness values in an affine-normalized patch. The two dimensions of the histogram are d, the distance from the center of the normalized patch, and i, the intensity value. Thus the "slice" of the spin image corresponding to a fixed d is simply the histogram of the intensity values of pixels located at a distance d from the center. In this work, the size of spin images is 10×10. Before using spin images as input to the density estimation process described in the next section, we rescale them to have a constant norm and "flatten" them into 100-dimensional feature vectors denoted $\mathbf{x}$ below.

2.2 Density Estimation

In the supervised framework, the training data consists of single-texture sample images from classes with labels C_ℓ, $\ell = 1, \dots, L$. The class-conditional densities $p(\mathbf{x}|C_\ell)$ can be estimated using all the feature vectors extracted from the images belonging to class C_ℓ. We model class-conditional densities as $p(\mathbf{x}|C_\ell) = \sum_{m=1}^{M} p(\mathbf{x}|c_{\ell m})\, p(c_{\ell m})$, where the components $c_{\ell m}$, $m = 1, \dots, M$, are thought of as *sub-classes*. Each $p(\mathbf{x}|c_{\ell m})$ is assumed to be a Gaussian with mean $\mu_{\ell m}$ and covariance matrix $\Sigma_{\ell m}$. The EM algorithm is used to estimate the parameters of the mixture model, namely the means $\mu_{\ell m}$, covariances $\Sigma_{\ell m}$, and mixing weights $p(c_{\ell m})$. EM is initialized with the output of the K-means algorithm. In this work, we use the same number of mixture components for each class ($M = 15$ and $M = 10$, respectively, for the experiments reported in Sections 3.1 and 3.2). We limit the number of free parameters and control numerical behavior by using spherical Gaussians with covariance matrices of the form $\Sigma_{\ell m} = \sigma_{\ell m}^2 I$.

The EM framework provides a natural way of incorporating unsegmented multi-texture images into the training set. Our approach is inspired by the work of Nigam et al. [10], who have proposed techniques for using unlabeled training data in text classification. Suppose we are given a multi-texture image annotated with the set $\mathcal{L}$ of class indices that it contains—that is, each feature vector $\mathbf{x}$ extracted from this image has a *label set* of the form $C_\mathcal{L} = \{C_\ell | \ell \in \mathcal{L}\}$. To accommodate label sets, the density estimation framework needs to be modified: instead of partitioning the training data into subsets belonging to each class and separately estimating L mixture models with M components each, we now use all the data simultaneously to estimate a single mixture model with $L \times M$ components. The estimation process must start by selecting some initial values for model parameters. During the *expectation* or E-step, we use the parameters to compute probabilistic sub-class membership weights given the feature vectors $\mathbf{x}$ and the label sets $C_\mathcal{L}$: $p(c_{\ell m}|\mathbf{x}, C_\mathcal{L}) \propto p(\mathbf{x}|c_{\ell m})\, p(c_{\ell m}|C_\mathcal{L})$, where $p(c_{\ell m}|C_\mathcal{L}) = 0$ for all $\ell \notin \mathcal{L}$ and $\sum_{\ell \in \mathcal{L}} \sum_{m=1}^{M} p(c_{\ell m}|C_\mathcal{L}) = 1$. During the *maximization* or M-step, we use the computed weights to re-estimate the parameters by maximizing the expected likelihood of the data in the standard fashion [1].

Overall, the incorporation of incompletely labeled data requires only a slight modification of the EM algorithm used for estimating class-conditional densities. However, this modification is of great utility, since the task of segmenting training examples by hand becomes an odious chore even for moderately-sized data sets. In situations where it is difficult to obtain large amounts of fully labeled examples, training on incompletely labeled or unlabeled data helps to improve classification performance [10].

In the subsequent experiments, we exercise the EM framework in two different ways. The data set of Section 3.1 contains both single- and multi-texture training images, which are used respectively to initialize and refine the parameters of the generative model. The data set of Section 3.2 consists entirely of unsegmented multi-texture images.

2.3 Neighborhood Statistics

This section describes the second layer of our representation, which accumulates information about the distribution of pairs of sub-class labels in neighboring regions. After the density estimation step, each region in the training image is assigned the sub-class label that maximizes the posterior probability $p(c_{\ell m}|\mathbf{x}, C_\mathcal{L})$. Next, we need a method for computing the neighborhood of a region centered at location $\mathbf{p}_0$ and having local shape matrix M. The simplest approach is to define the neighborhood as the set of all points $\mathbf{p}$ such that $(\mathbf{p} - \mathbf{p}_0)^T M(\mathbf{p} - \mathbf{p}_0) \le \alpha$ for some constant α. However, in practice this definition produces poor results: points with small ellipses get too few neighbors, and points with large ellipses get too many. A better approach is to "grow" the ellipse by adding a constant absolute amount (15 pixels in the implementation) to the major and minor axes, and to let the neighborhood consist of all points that fall inside this enlarged ellipse. In this way, the size and shape of the neighborhood still depends on the affine shape of the region, but the neighborhood structure is more balanced.

Once we have defined the neighborhood structure, we can think of the image as a directed graph with arcs emanating from the center of each region to other centers within its neighborhood. The existence of an arc from a region with sub-class label c to another region with label c' is a joint event (c, c') (note that the order is important since the neighborhood relation is not symmetric). We find the relative frequencies $p(c, c')$ for all pairs (c, c'), and also compute the marginals $\hat{p}(c) = \sum_{c'} p(c, c')$ and $\check{p}(c') = \sum_{c} p(c, c')$. Finally, we compute the values

$$r(c, c') = \frac{p(c, c') - \hat{p}(c)\,\check{p}(c')}{\left[\left(\hat{p}(c) - \hat{p}^2(c)\right)\left(\check{p}(c') - \check{p}^2(c')\right)\right]^{\frac{1}{2}}}$$

representing the correlations between the events that the labels c and c', respectively, belong to the source and destination nodes of the same arc. The values of $r(c, c')$ must lie between -1 and 1; negative (resp. positive) values indicate that c and c' rarely (resp. frequently) co-occur as labels at endpoints of the same edge.

In our experiments, we have found that the values of $r(c, c')$ are reliable only when c and c' are sub-classes of the same class. Part of the difficulty in estimating correlations across classes is the lack of data in the training set. Even if the set contains multi-texture images, only a few arcs actually fall across texture boundaries. Unless the number of texture classes is very small, it is quite difficult to create a training set that would include samples of every possible boundary. Thus, whenever c and c' belong to different classes, we set $r(c, c')$ to a constant negative value that serves as a "smoothness constraint" in the relaxation algorithm described in the next section (we use values between -0.5 and -1, all of which tend to produce similar results).

2.4 Relaxation

We have implemented the classic relaxation algorithm of Rosenfeld et al. [11]. The initial estimate of the probability that the ith region has label c, denoted $p_i^{(0)}(c)$, is obtained from the learned mixture model as the posterior $p(c|\mathbf{x}_i)$. Note that since we run relaxation on unlabeled test data, these probabilities must be computed for all $L \times M$ sub-class labels corresponding to all possible classes. At each iteration, new estimates $p_i^{(t+1)}(c)$ are obtained by updating the current probabilities $p_i^{(t)}(c)$ using the equation

$$p_i^{(t+1)}(c) = \frac{p_i^{(t)}(c)\left[1 + q_i^{(t)}(c)\right]}{\sum_c p_i^{(t)}(c)\left[1 + q_i^{(t)}(c)\right]} \, ,$$

$$q_i^{(t)}(c) = \sum_j w_{ij} \left[\sum_{c'} r(c, c')\, p_j^{(t)}(c')\right] . \qquad (1)$$

The scalars w_{ij} are weights that indicate how much influence region j exerts on region i. We treat w_{ij} as a binary indicator variable that is nonzero if and only if the jth region belongs to the ith neighborhood. The weights are required to be normalized so that $\sum_j w_{ij} = 1$ [11].

The update equation (1) can be justified in qualitative terms as follows. Note that $p_j^{(t)}(c')$ has no practical effect on $p_i^{(t)}(c)$ when the ith and jth regions are not neighbors, when c and c' are uncorrelated, or when the probability $p_j^{(t)}(c')$ is low. However, the effect is significant when the jth region belongs to the ith neighborhood and the value of $p_j^{(t)}(c')$ is high. The correlation $r(c, c')$ expresses how "compatible" the labels c and c' are at nearby locations. Thus, $p_i^{(t)}(c)$ is increased (resp. decreased) by the largest amount when $r(c, c')$ has a large positive (resp. negative) value. Overall, the probabilities of different sub-class labels at neighboring locations reinforce each other in an intuitively satisfying fashion. Even though the iteration of (1) has no convergence guarantees, we have found it to behave well on our data. To obtain the results of Sections 3.1 and 3.2, we run relaxation for 200 iterations.

2.5 Classification and Retrieval

Individual regions are classified by assigning them to the class that maximizes $p_i(C_\ell) = \sum_{m=1}^{M} p_i(c_{\ell m})$. To perform classification and retrieval at the image level, we need to define a "global" score for each class. In the experiments of the next section, the score for class C_ℓ is computed by summing the probability of C_ℓ over all N regions found in the image: $\sum_{i=1}^{N} \sum_{m=1}^{M} p_i(c_{\ell m})$, where the $p_i(c_{\ell m})$ are the probability estimates following relaxation. Classification of single-texture images is carried out by assigning the image to the class with the highest score, and retrieval for a given texture model proceeds from highest scores to lowest.

3 Experimental Results

3.1 The Indoor Scene

Our first data set contains seven different textures present in a single indoor scene (Figure 1). To test the invariance of our representation, we have gathered images over a wide range of viewpoints and scales. The data set is partitioned as follows: 10 single-texture training images of each class; 10 single-texture validation images of each class; 13 two-texture training images; and 45 multi-texture test images.

Table 1 shows classification results for the single-texture validation images following training on single-texture images only. The columns labeled "image" show the fraction of images classified correctly using the score described in Section 2.5. As can be seen from the first column, successful classification at the image level does not require relaxation: good results are achieved in most cases by using the probabilities output by the generative model. Interestingly, for class T6 (marble), the classification rate actually drops as an artifact of relaxation. When the right class has relatively low initial probabilities, the self-reinforcing nature of relaxation often serves to diminish these probabilities further. The columns labeled "region", which show the fraction of all *individual regions* in the validation images that were correctly classified based on the probabilities $p_i(C_\ell)$,

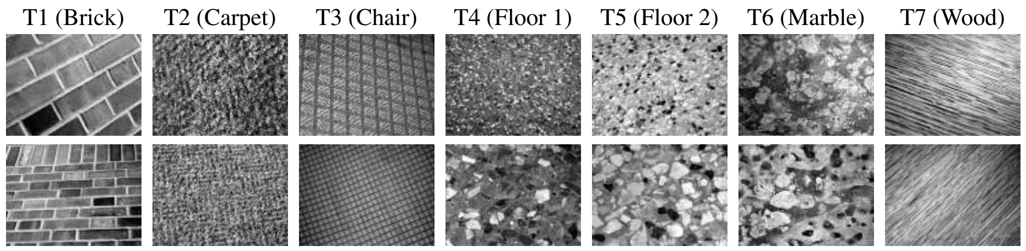


Figure 1: Samples of the texture classes used in the experiments of Section 3.1.

	Before relaxation		After relaxation	
Class	image	region	image	region
T1	1.00	0.61	1.00	0.97
T2	1.00	0.58	1.00	0.99
T3	0.90	0.70	0.90	0.85
T4	1.00	0.61	1.00	0.99
T5	1.00	0.45	1.00	0.95
T6	0.90	0.29	0.80	0.67
T7	0.60	0.41	0.70	0.73

Table 1: Classification rates for single-texture images.

are much more indicative of the impact of relaxation: for all seven classes, classification rates improve dramatically.

Next, we evaluate the performance of the system for retrieval of images containing a given texture. Figure 2 shows the results in the form of ROC curves that plot the positive detection rate (the number of correct images retrieved over the total number of correct images) against the false detection rate (the number of false positives over the total number of negatives in the data set). The top row shows results obtained after fully supervised training using single-texture images only, as described in Section 2.2. The bottom row shows the results obtained after re-estimating the generative model following the incorporation of 13 two-texture images into the training set. Following relaxation, a modest improvement in performance is achieved for most of the classes. A more significant improvement could probably be achieved by using more multi-texture training samples [10].

For the majority of test images, our system succeeds in providing an accurate segmentation of the image into regions of different texture. Part (a) of Figure 3 shows a typical example of the difference made by relaxation in the assignment of class labels to individual regions. Part (b) shows more examples where the relaxation was successful. Note in particular the top example of part (b), where the perceptually similar classes T4 and T5 are unambiguously separated. Part (c) of Figure 3 shows two examples of segmentation failure. In the bottom example, classes T2 (carpet) and T3 (chair) are confused, which can be partly explained by the fact that the scales at which the two textures appear in this image are not well represented in the training set. Overall, we have found the relaxation process to be sensitive to initialization, in the sense that poor initial probability estimates lead to artifacts in the final assignment.

3.2 Animals

Our second data set consists of unsegmented images of three kinds of animals: cheetahs, giraffes, and zebras. The training set contains 10 images from each class, and the test set contains 20 images from each class, plus 20 "negative" images not containing instances of the target animals. To account for the lack of segmentation, we introduce an additional "background" class, and each training image is labeled as containing the appropriate animal and the background. To initialize EM on this data, we randomly assign each feature vector either to the appropriate animal class, or to the background. The ROC curves for each class are shown in Figure 4, and segmentation results are shown in Figure 5. Overall, our system appears to have learned very good models for cheetahs and zebras, but not for giraffes.

We conjecture that several factors account for the weakness of the giraffe model. Some of the blame can be placed on the early stage of feature extraction. Namely, the Laplacian-based affine region detector is not well adapted to the giraffe texture whose blobs have a relatively complex shape. At the learning stage, the system also appears to be "distracted" by background features, such as sky and trees, that occur more commonly in training samples of giraffes than of the other animals. In the bottom image of Figure 5, "giraffe-ness" is associated with some parts of the background, as opposed to the animals themselves. The artificial "background" class is simply too inhomogeneous to be successfully represented in the mixture framework. A principled solution to this problem would involve partitioning the background into a set of natural classes (e.g., grass, trees, water, rocks, etc.) and building larger training sets that would include these classes in different combinations.

Overall, our results (though somewhat uneven) are promising. Unlike many other methods suitable for modeling natural textures, ours does not require negative examples. The EM framework shows surprising aptitude for automatically separating positive areas of the image from negative ones, without the need for specially designed significance scores such as the ones used by Schmid [14].

4 Discussion and Future Work

The texture representation method proposed in this paper offers several important advantages over other methods proposed in recent literature [3, 7, 14]. The use of an interest

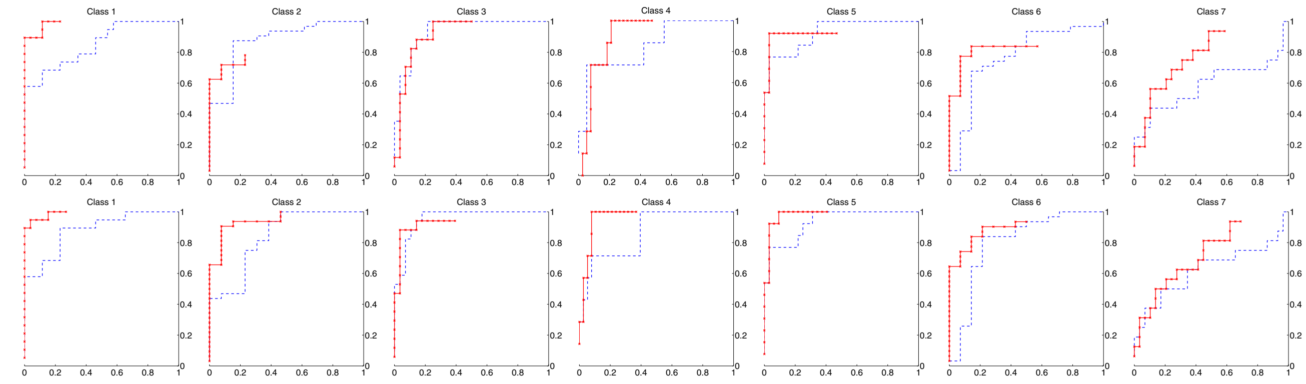
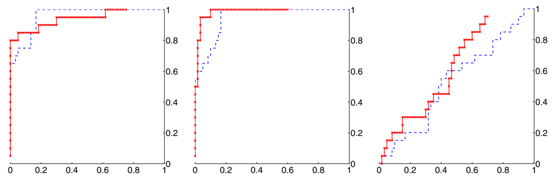


Figure 2: ROC curves for retrieval in the test set of 45 multi-texture images. The dashed (resp. solid) line represents performance before (resp. after) relaxation. Top row: single-texture training images only, bottom row: single-texture and two-texture training images.



Cheetah Zebra Giraffe

Figure 4: ROC curves for the animal dataset. The dashed (resp. solid) line represents performance before (resp. after) relaxation.

point detector leads to a sparse representation that selects the most perceptually salient regions in an image, while the shape selection process provides affine invariance. Another important advantage of shape selection is the adaptive determination of both levels of image structure: the window size over which local descriptors are computed, and the neighborhood relationship between adjacent windows.

In the future, we will pursue several directions for the improvement of our system. We have found that the performance of relaxation is sensitive to the quality of the initial probability estimates; therefore, we need to obtain the best estimates possible. To this end, we plan to investigate the effectiveness of discriminative models, e.g. neural networks, that output confidence values interpretable as probabilities of class membership. Relaxation can also be made more effective by the use of stronger geometric neighborhood relations that take into account affine shape while preserving the maximum amount of invariance. Finally, we plan to extend our work to modeling complex texture categories found in natural imagery, e.g., cities, forests, and oceans.

Acknowledgments. This research was partially funded by the UIUC Campus Research Board, the National Science Foundation grants IRI-990709 and IIS-0308087, the European Project LAVA (IST-2001-34405), and by a UIUC-CNRS collaboration agreement.

References

[1] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] A. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes", *IEEE Trans. PAMI*, 21(5), 1999, pp. 433-449.

[3] S. Kumar and M. Hebert, "Man-Made Structure Detection in Natural Images Using a Causal Multiscale Random Field", *Proc. CVPR*, 2003, vol. 1, pp. 119-126.

[4] S. Lazebnik, C. Schmid, and J. Ponce, "A Sparse Texture Representation Using Affine-Invariant Regions", *Proc. CVPR*, 2003, vol. 2, pp. 319-324.

[5] T. Lindeberg and J. Gårding, "Shape-Adapted Smoothing in Estimation of 3-D Depth Cues from Affine Distortions of Local 2-D Brightness Structure", *Image and Vision Computing*, 15, 1997, pp. 415-434.

[6] T. Lindeberg, "Feature Detection with Automatic Scale Selection", *IJCV* 30(2), 1998, pp. 77-116.

[7] J. Malik, S. Belongie, T. Leung and J. Shi, "Contour and Texture Analysis for Image Segmentation", *IJCV* 43(1), 2001, pp. 7-27.

[8] K. Mikolajczyk and C. Schmid, "Indexing Based on Scale Invariant Interest Points", *Proc. ICCV*, 2001, pp. 525-531.

[9] K. Mikolajczyk and C. Schmid, "An Affine Invariant Interest Point Detector", *Proc. ECCV*, 2002, vol. 1, pp. 128-142.

[10] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM", *Machine Learning* 39 (2/3), 2000, pp. 103-134.

[11] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene Labeling by Relaxation Operations", *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6), 1976, pp. 420-433.

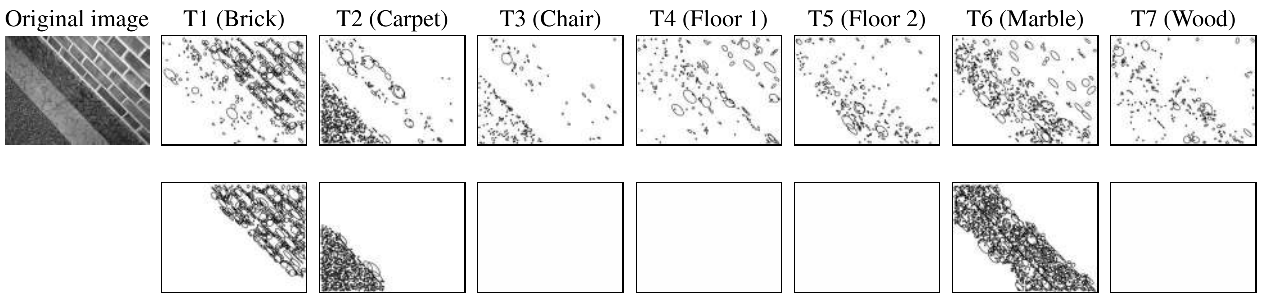
[12] Y. Rubner and C. Tomasi, "Texture-Based Image Retrieval Without Segmentation", *Proc. ICCV*, 1999, pp. 1018-1024.

[13] F. Schaffalitzky and A. Zisserman, "Viewpoint Invariant Texture Matching and Wide Baseline Stereo", *Proc. ICCV*, 2001, vol. 2, pp. 636-643.

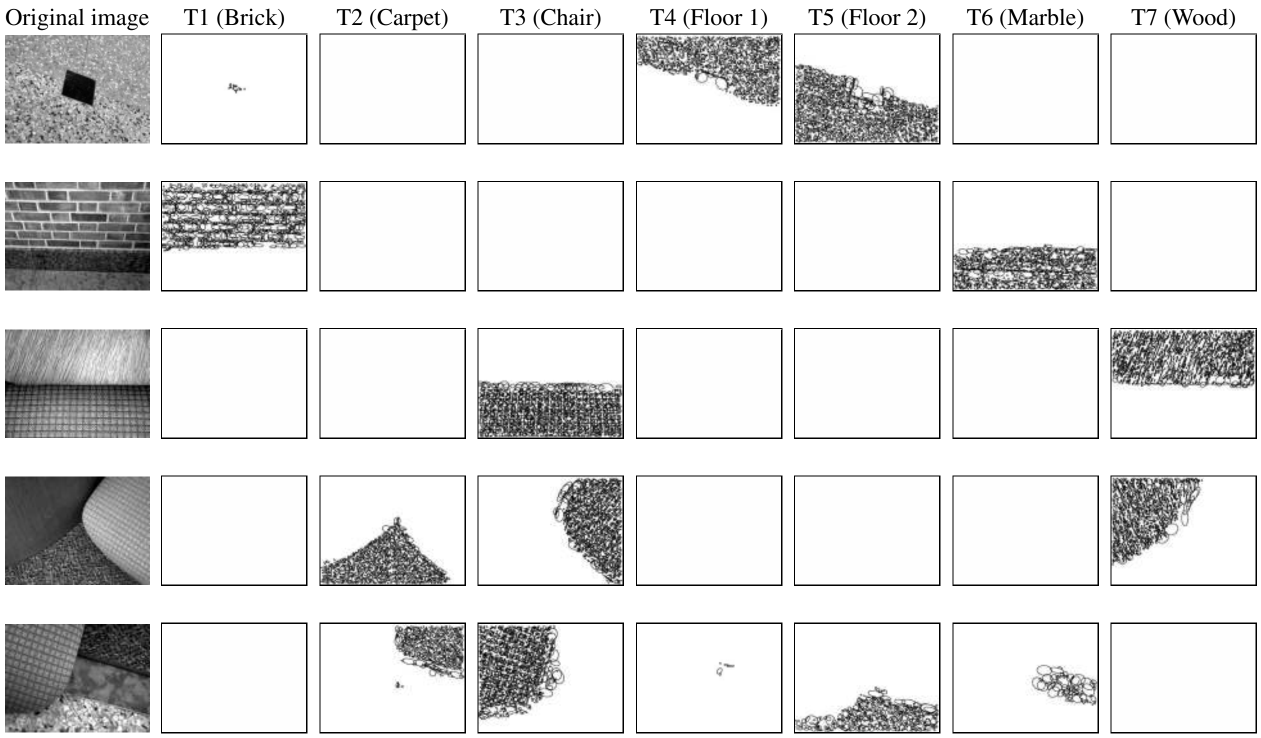
[14] C. Schmid, "Constructing Models for Content-Based Image Retrieval", *Proc. CVPR*, 2001, vol. 2, pp. 39-45.

[15] T. Tuytelaars and L. Van Gool, "Matching Widely Separated Views based on Affinely Invariant Neighbourhoods", submitted to *IJCV*, 2001.

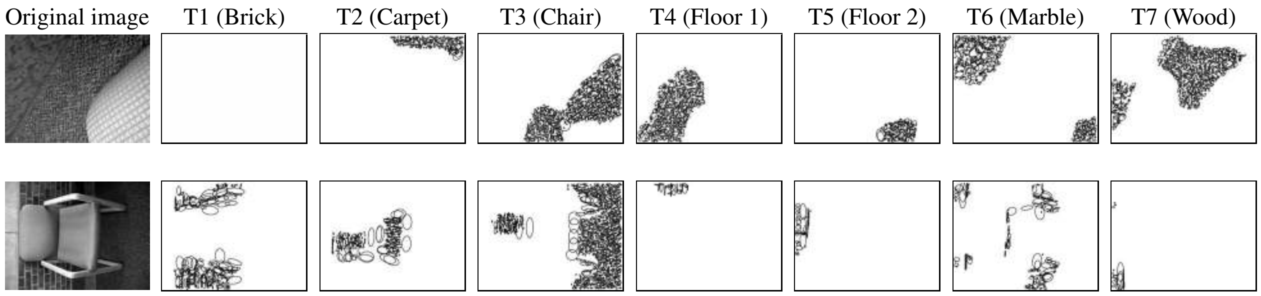
[16] M. Varma and A. Zisserman, "Classifying Images of Materials: Achieving Viewpoint and Illumination Independence", *Proc. ECCV*, 2002, vol. 3, pp. 255-271.



(a) Initial labeling of regions (top) vs. the final labeling following relaxation (bottom).



(b) Successful segmentation examples.



(c) Unsuccessful segmentation examples.

Figure 3: Segmentation results.

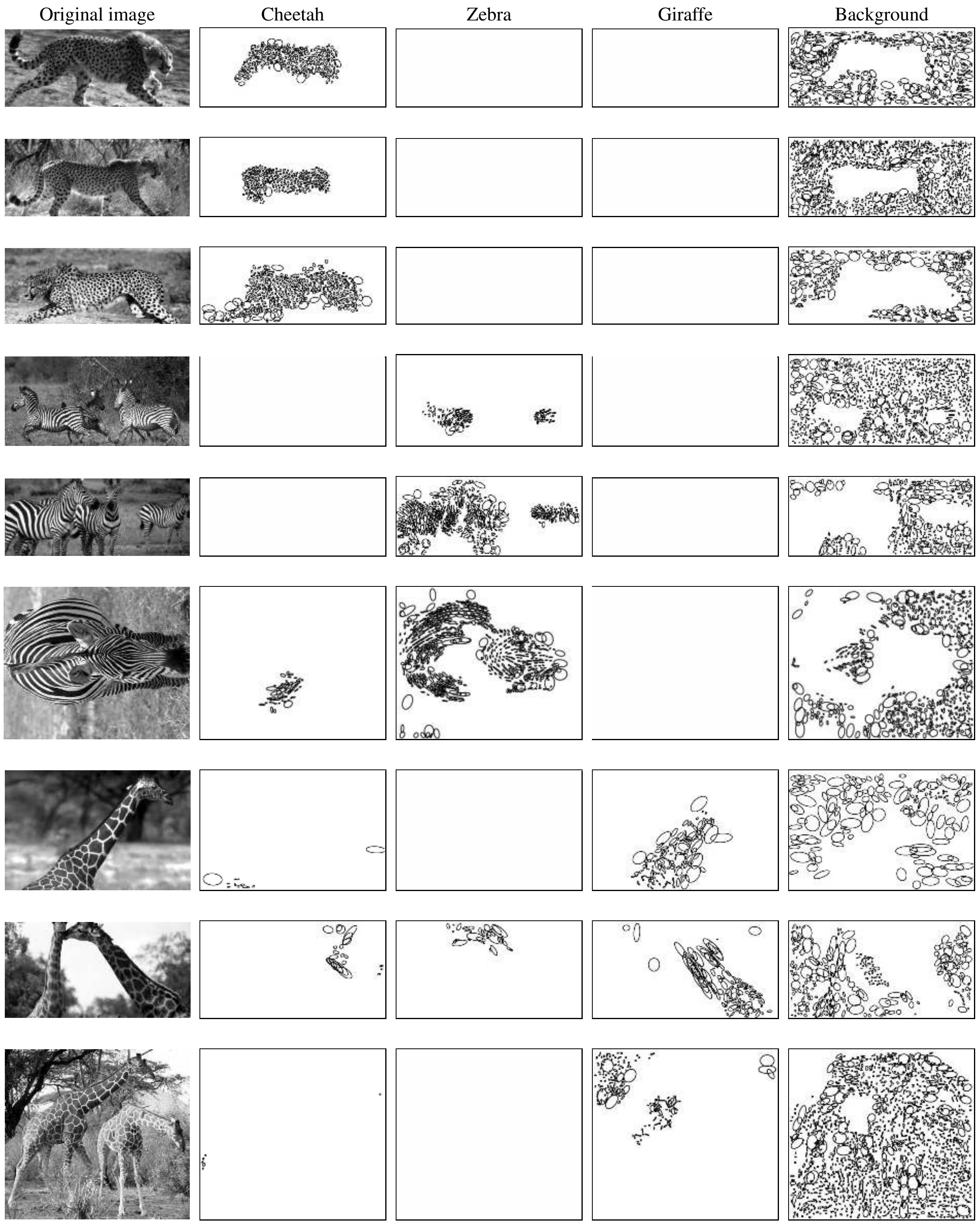


Figure 5: Segmentation on the animal dataset.