

# Learning Dictionaries of Discriminative Image Patches

Anders Lindbjerg Dahl

abd@imm.dtu.dk, <http://www.imm.dtu.dk/~abd>

Rasmus Larsen

rl@imm.dtu.dk, <http://www.imm.dtu.dk/~rl>

DTU Informatics

Technical University of Denmark

Lyngby, Denmark

---

## Abstract

Remarkable results have been obtained using image models based on image patches, for example sparse generative models for image inpainting, noise reduction and super-resolution, sparse texture segmentation or texton models. In this paper we propose a powerful and yet simple approach for segmentation using dictionaries of image patches with associated label data. The approach is based on ideas from sparse generative image models and texton based texture modeling. The intensity and label dictionaries are learned from training images with associated label information of (a subset) of the pixels based on a modified vector quantization approach. For new images the intensity dictionary is used to encode the image data and the label dictionary is used to build a segmentation of the image. We demonstrate the algorithm on composite and real texture images and show how successful training is possible even for noisy image and low-quality label training data. In our experimental evaluation we achieve state-of-the-art performance for segmentation.

## 1 Introduction

We address the problem of supervised image segmentation. Our approach is based on a set of image patches stored in an intensity dictionary, with associated label information stored in a label dictionary. These dictionaries are learned from training data. Based on the intensity dictionary we encode an unknown image, and use the associated label information for inferring label probabilities to each pixel. Our methods is simple and highly robust to noise as illustrated in the segmentation examples in Figure 1.

The proposed method is based on modeling an image using small image patches, which has similarities to sparse image coding [3, 14] and textons [7, 9, 17]. Sparse methods based on an overcomplete dictionary have become very popular for solving a number of image analysis problems. This group of methods is generative and models an image by linearly combining a set of dictionary atoms. The generative nature is directly applicable in for example image restoration problems like denoising [4, 6], image compression [8], inpainting and texture separation [5], super-resolution [19] and many more [3]. Some of the attractive properties of the sparse methods are their robustness to noise, closeness to data and simplicity in both implementation and interpretation. We aim at adopting these properties in our method, which generally target discriminative problems.

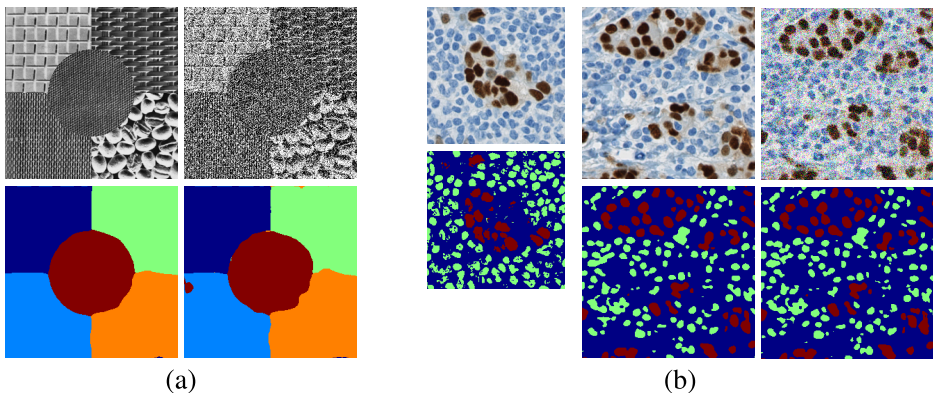


Figure 1: Example of multi class segmentation using patch based classification. (a) combination of five Brodatz textures – *left* original and *right* added 50 % Gaussian noise. (b) histopathological colored tissue sample of cell nuclei – *left* training sample, *center* original test image and *right* added 100 % Gaussian noise (relative to standard deviation).

Sparse methods based on dictionary learning has previously been used for discriminative problems [11, 12, 13, 18]. These methods are based on one dictionary for each class and the residual error as the discriminant. They show that a class specific dictionary can be chosen, which sparsely encode one class well and other classes bad. This requires a dictionary for each class, which is computational costly, but these methods achieve very high performance. Initially the chosen dictionaries were optimized for reconstruction [13, 18], but significant improvements have been shown by optimizing for discriminative power [11, 12]. The procedure that we suggest is different from these methods, by using one dictionary for encoding an image and then inferring label probabilities from the label dictionary. The result is a simple model with only one dictionary for all classes.

## 2 Method

Our method is based on a dictionary of image patches, which we will denote dictionary atoms. To each atom we associate a label atom that we use for building the segmentation image. The procedure for building the dictionary is illustrated in Figure 2 and the segmentation procedure is shown in Figure 3. Basically the segmentation is performed by finding the nearest dictionary atom to an image patch. The associated label atom is used for inferring the label probability to the image region covered by the image patch. Patches are overlapping so several label atoms are added to the same pixel.

A good dictionary is characterized by modeling the image patches well and containing atoms that are unique for a given texture. We use these two properties in constructing our discriminative dictionary.

In our approach we store the image intensity information and the label information in separate dictionaries. We denote the intensity dictionary  $\mathbf{D} \in \mathbb{R}^{hn \times m}$ , which is a matrix with  $m$  atoms – one in each column. A dictionary atom is an image patch concatenated to a vector. The size of the image patch is  $\sqrt{n} \times \sqrt{n}$  and  $h$  is the color depth with for example  $h = 1$  for a grey scale image and  $h = 3$  for a RGB image.  $\mathbf{D}$  is used for modeling an image patch  $\mathbf{x}$  by

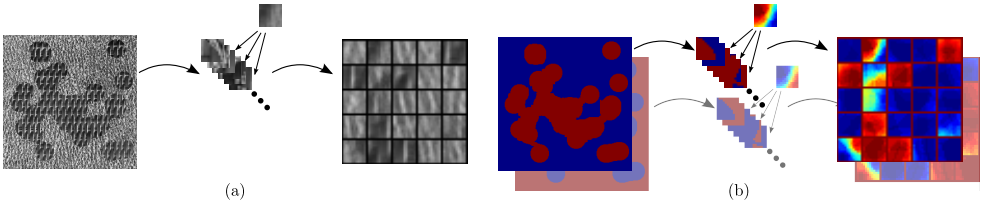


Figure 2: Schematic illustration of the construction of the dictionary. (a) is the intensity dictionary and (b) is the label dictionary. Note that the label dictionary atoms can cover more than one class. A set of training image patches are used for constructing the dictionaries based on the modified vector quantization described in Algorithm 1 and 2, and the segmentation is carried out using Algorithm 3.

choosing the nearest atom in  $\mathbf{d}_j \in \mathbf{D}$

$$\mathbf{d}^* = \min_j \|\mathbf{d}_j - \mathbf{x}\|, \quad (1)$$

where  $j = \{1, \dots, m\}$ . We want to learn  $\mathbf{D}$  from data to model the image textures well. This can be found by choosing  $\mathbf{D}$  such that it minimizes the residual error

$$\mathbf{D} = \arg \min_{\mathbf{D}} \sum_{i=1}^o \|\mathbf{d}_i^* - \mathbf{x}_i\|_2, \quad (2)$$

where  $o$  is the number of training samples. This is a clustering problem, that is known to be NP-hard [10], but a good iterative approximation based on vector quantization can be found with the k-means algorithm.

A dictionary with small residual errors on the image atoms does not necessarily have good discriminative properties. So, simultaneously to modeling intensity data well, we also want the dictionary atoms to be unique for a specific class and hereby have high discriminative power. We associate the intensity dictionary  $\mathbf{D}$  with a label dictionary  $\mathbf{L} \in [0; 1]^{n \times m \times l}$ , where  $l$  is the number of classes. The spatial extension of the label dictionary atoms is the same as the intensity dictionary, so a probability can be inferred on each pixel. Each atom  $\mathbf{d}_j$  in  $\mathbf{D}$  has an associated atom  $\mathbf{l}_j$  in  $\mathbf{L}$ , and a pixel in a label atom contains the probability of all classes, so  $\sum_{k=1}^l \mathbf{l}_k(g) = 1$  where  $g \in \{1, \dots, n\}$  is the pixels in the label atom. Ideally a pixel in a label atom will have probability 1 for one class and 0 for all other classes. An observed label atom can be changed to an ideal atom by changing the label probabilities. We denote the ideal label atom  $\hat{\mathbf{l}}$  and this is found by

$$\hat{\mathbf{l}}_k(g) = \begin{cases} 1 & \text{if } k = \max_k \mathbf{l}_k(g) \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

The discriminative power of the dictionary atoms can be found by the following minimization

$$\mathbf{L} = \arg \min_{\mathbf{L}} \sum_{i=1}^o \sum_{g=1}^n \|\hat{\mathbf{l}}_i - \mathbf{l}_i\|_1, \quad (4)$$

where each atom in  $\mathbf{L}$  is optimized to contain label information for one class in each pixel.

**Algorithm 1** Initialize the dictionary

**Require:** Set of training images ( $\hat{\mathbf{P}}$ ) with associated label images ( $\hat{\mathbf{P}}_l$ ) and empty dictionaries  $\mathbf{D}$  and  $\mathbf{L}$

- 1: Select a random subset of  $\eta$  images patches  $\mathbf{X}'$  of size  $\sqrt{n} \times \sqrt{n} \times h$  and corresponding subset of label patches  $\mathbf{Y}'$  of size  $\sqrt{n} \times \sqrt{n} \times l$
- 2: Set the first patch and label patch as the first dictionary atoms  $\mathbf{d}_1 = \mathbf{x}_1$  and  $\mathbf{l}_1 = \mathbf{y}_1$
- 3: Set number of dictionary atoms  $n_d = 1$

- 4: **for**  $i = 2$  **to**  $\eta$  **do**
- 5:    $w = \min_{j=1}^{n_d} (1 - \frac{1}{n} \sum_{g=1}^n \sum_{k=1}^l \| \mathbf{y}'_{ik} - \mathbf{l}_{jk} \|_1)$
- 6:   **if** ( $w \geq 0$ ) **then**
- 7:     Add  $\mathbf{x}'_i$  to  $\mathbf{D}$  and  $\mathbf{y}'_i$  to  $\mathbf{L}$
- 8:   **else**
- 9:      $n_d = n_d + 1$
- 10:     $\mathbf{d}_{n_d} = \mathbf{x}'_i$  and  $\mathbf{l}_{n_d} = \mathbf{y}'_i$
- 11:   **end if**
- 12: **end for**
- 13: **return**  $\mathbf{D}$  and  $\mathbf{L}$

**Algorithm 2** Building the dictionary

**Require:** Set of images used for initializing the dictionaries ( $\hat{\mathbf{P}}, \hat{\mathbf{P}}_l$ ) and initialized dictionaries ( $\mathbf{D}, \mathbf{L}$ )

- 1: Select the set of all images patches  $\mathbf{X}$  of size  $\sqrt{n} \times \sqrt{n} \times h$  and all corresponding label patches  $\mathbf{Y}$  of size  $\sqrt{n} \times \sqrt{n} \times l$
- 2: Choose empty positive and negative intensity dictionaries  $\mathbf{D}_{pos}, \mathbf{D}_{neg}$  and empty label dictionary  $\bar{\mathbf{L}}$
- 3: Choose number of iterations  $n_{iter}$  and a displacement factor  $\tau$
- 4: **for**  $t = 1$  **to**  $n_{iter}$  **do**
- 5:   Build ideal label dictionary  $\hat{\mathbf{L}}$  (Equation 3)
- 6:   **for**  $i = 1$  **to**  $o$  **do**
- 7:     Find  $\mathbf{d}^*$  (Equation 1) and the corresponding ideal label dictionary atom  $\hat{\mathbf{l}}^*$

- 8:    $w = \min_{j=1}^m (1 - \frac{1}{n} \sum_{g=1}^n \sum_{k=1}^l \| \mathbf{y}_{ik} - \hat{\mathbf{l}}_k^* \|_1)$
- 9:   **if** ( $w \geq 0$ ) **then**
- 10:     Add  $\mathbf{x}_i$  to  $\mathbf{D}_{pos}$
- 11:   **else**
- 12:     Add  $\mathbf{x}_i$  to  $\mathbf{D}_{neg}$
- 13:   **end if**
- 14:   Add  $\mathbf{y}_i$  to  $\bar{\mathbf{L}}$
- 15: **end for**
- 16: Update dictionaries  $\mathbf{D}$ :
- 17: **for**  $j = 1$  **to**  $m$  **do**
- 18:    $\mathbf{d}_j = \mathbf{d}_j + \tau(\mathbf{d}_{pos,j} - \mathbf{d}_{neg,j})$
- 19: **end for**
- 20: Set  $\mathbf{L} = \bar{\mathbf{L}}$
- 21: **end for**
- 22: **return**  $\mathbf{D}$  and  $\mathbf{L}$

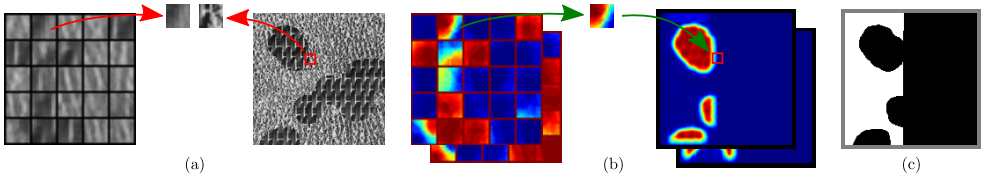


Figure 3: Schematic illustration of the labeling. In (a) an image patch is extracted and the most similar dictionary atom is found. In (b) the corresponding label atom is added to the label probability image in a window covering the same spatial area as in the intensity image. Note that this is a binary segmentation, so the label image has two label dimensions. The binary segmentation (c) is found as the label image with the highest energy (*the black and white part*). The figure illustrates labeling half way through the algorithm.

The coupling of the label and intensity dictionaries requires the optimization to be done for both dictionaries simultaneously. We propose an iterative approach for approximating the solutions of Equations 2 and 4. First we initialize the dictionaries using a subset of the training image patches, as described in Algorithm 1. The procedure ensures that the initial dictionary atoms are unique, such that image patches both with similar intensity and similar label patches belong to the same dictionary atoms. Based on the initialized atoms the dictionary is iteratively optimized based on the procedure described in Algorithm 2.

Algorithm 2 is a modified vector quantization, but the label information is included in building the clusters. The dictionary atoms are iteratively updated to be moved towards image patches that have similar labels as the ideal label atom (Equation 3) and away from dissimilar patches. This is implemented by dividing the image patches into similar and dissimilar groups where each group has a center point in  $\mathbb{R}^{n \times h}$ . These two center points form a vector and we move along this vector towards the center of the similar group. The step towards the similar group is controlled by the parameter  $\tau$ . We found the algorithm to obtain good performance after 20 - 40 iterations. The performance is not sensitive to the choice of  $\tau$  where we found a value between 0.1 - 0.5 to be a good choice, and all experiments are performed using  $\tau = 0.2$ .

To build the label dictionary we employ the label dictionary. The label image is a probability map for the class labels. Given an image  $\mathbf{P} \in \mathbb{R}^{r \times c \times h}$  we construct a label image  $\mathbf{P}_l \in [0; 1]^{r \times c \times l}$  initially with all zeros. We build the label image by pixel wise coding  $\mathbf{P}$  using the dictionary  $\mathbf{D}$ . This is done by visiting all pixels  $\mathbf{P}(i, j)$  where we can extract a  $\sqrt{n} \times \sqrt{n} \times h$  image patch. The nearest dictionary atom  $\mathbf{d}_j$  is found and the corresponding label patch  $\mathbf{l}_j$  is added to the  $\sqrt{n} \times \sqrt{n} \times l$  image window centered at position  $(i, j)$ . The final label image will in each pixel  $\mathbf{P}_l(i, j)$  have the average probability of all labels that covered that pixel. The third dimension of  $\mathbf{P}_l$  encodes probability for each label. Figure 3 illustrates the labeling process and Algorithm 3 describes the steps in the labeling procedure.

### 3 Experiments

We have performed a number of segmentation experiments on composed and natural textures to demonstrate the properties of our model. The experiments show the method's flexibility and robustness to noise, and we compare to similar approaches on a number of image com-

---

#### Algorithm 3 Image labeling

---

<p><b>Require:</b> Image to be labeled (<math>\mathbf{P}</math>) and empty label image (<math>\mathbf{P}_l</math>) and trained dictionaries (<math>\mathbf{D}, \mathbf{L}</math>)</p> <p>1: Select the set of all <math>q</math> images patches <math>\mathbf{X}</math> of size <math>\sqrt{n} \times \sqrt{n} \times h</math> from <math>I</math> and all corresponding label patches <math>\mathbf{Y}</math> of size <math>\sqrt{n} \times \sqrt{n} \times l</math> from <math>\mathbf{P}_l</math></p> <p>2: <math>s_h = \lfloor \sqrt{n}/2 \rfloor</math></p> <p>3: <b>for</b> <math>i = 1 + s_h</math> <b>to</b> <math>r - s_h</math> <b>do</b></p> <p>4:     <b>for</b> <math>j = 1 + s_h</math> <b>to</b> <math>c - s_h</math> <b>do</b></p> <p>5:         Find <math>\mathbf{d}^*(i, j)</math> (Equation 1)</p>	<p>6:     <math>t_r = 0</math></p> <p>7:     <b>for</b> <math>i_r = i - s_h</math> <b>to</b> <math>i + s_h</math> <b>do</b></p> <p>8:         <math>t_r = t_r + 1, t_c = 0</math></p> <p>9:         <b>for</b> <math>j_c = j - s_h</math> <b>to</b> <math>j + s_h</math> <b>do</b></p> <p>10:             <math>t_c = t_c + 1</math></p> <p>11:             <math>I_l(i_r, j_c) = \mathbf{I}^*(t_r, t_c)</math></p> <p>12:             <b>end for</b></p> <p>13:         <b>end for</b></p> <p>14:     <b>end for</b></p> <p>15: <b>end for</b></p> <p>16: <b>return</b> <math>I_l</math></p>
--	---

---

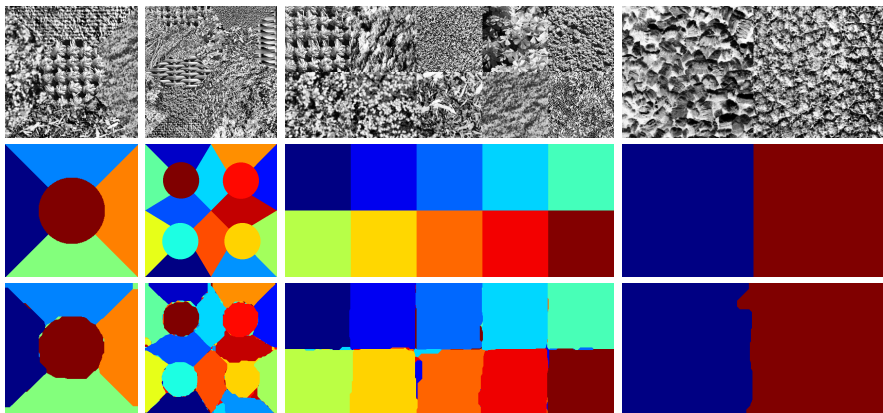


Figure 4: Examples of the test textures used in the segmentation experiment. Top row shows the composed textures, middle row shows the ground truth and bottom row shows the segmentation results. The shown examples are image number #4, #7, #9, and #12, with 5, 16, 10 and 2 texture classes.

positions made from Brodatz textures [10], the VisTex database<sup>1</sup>, and the MeasTex database<sup>2</sup>. The textures and training samples are downloaded from Randen’s homepage<sup>3</sup>. These textures are described in [10] and examples are shown in Figure 4.

**Composed textures** Our texture segmentation experiments are compared to [10, 18] and the results are shown in Table 1. The proposed method performs better than existing methods in more than half of the samples, and follows closely in the rest. On average it performs better than the best performing previous method by [10] and there is not an outlier like texture #2 in [10].

We have performed the segmentation by training a dictionary as described in Section 2. In Table 2 the number of patches for initializing the dictionaries are shown together with the size of the resulting dictionaries. The images are preprocessed by taking first and second order derivatives excluding the original intensity, which results in five dimensions in each pixel ( $\hat{\mathbf{P}} = [\mathbf{P}_x, \mathbf{P}_y, \mathbf{P}_{xx}, \mathbf{P}_{yy}, \mathbf{P}_{xy}]^T$ ). We have used a Gaussian mask on each image patch with a standard deviation of half the patch size (1.5 for the  $3 \times 3$  patches).

From the segmentation procedure we obtain a probability image. These probability images have been postprocessed by smoothing to obtain the final segmentation, which closely follows the procedure described in [10]. The first smoothing is obtained by convolving the probability image using a Gaussian filter with a standard deviation of 8. The second smoothing is based on graph cuts using the method of Kolmogorov and Zabih [9]. Before applying graph cuts we convolve using a Gaussian with standard deviation of 4.5. We use a 8-neighborhood graph and  $\alpha$ -expansion to obtain the final segmentation. To speed up the nearest neighbor search for image and dictionary atoms we use approximate nearest neighbor based the FLANN library by Muja and Lowe [13].

The effect of including the label atoms in the optimization is shown in Table 1. In this

<sup>1</sup><http://vismod.media.mit.edu/vismod/imagery/VisionTexture/>

<sup>2</sup><http://www.texturesynthesis.com/meastex/meastex.html>

<sup>3</sup><http://www.ux.uio.no/~tranden/>



			Label-optimized		k-means	
#			Gauss	GC	Gauss	GC
1	5.50	1.61	2.00	1.52	2.20	<b>1.50</b>
2	7.30	16.42	6.85	4.31	5.49	<b>4.12</b>
3	13.20	4.15	5.95	<b>3.66</b>	6.79	4.39
4	5.60	3.67	3.86	<b>3.60</b>	5.49	4.58
5	10.50	4.32	3.91	3.70	4.95	<b>3.02</b>
6	17.10	<b>9.04</b>	18.24	13.58	22.55	17.77
7	17.20	8.80	10.12	<b>5.66</b>	14.66	9.18
8	18.90	<b>2.24</b>	9.82	6.74	20.90	23.02
9	21.40	<b>2.04</b>	6.85	2.57	16.10	5.24
10	NA	0.17	0.27	0.20	0.19	<b>0.06</b>
11	NA	<b>0.60</b>	1.20	1.09	1.92	1.64
12	NA	<b>0.37</b>	3.34	1.58	1.56	0.77
Avg.	(11.67)	4.45	6.03	<b>4.00</b>	7.48	6.27

Table 1: Texture segmentation performance compared to existing segmentation algorithms. Results are error rates in percentage. The two procedures *Label optimized* is based on the proposed method and *k-means* is based on building the dictionary from k-means clustering with same dictionary sizes as in the label optimized. *Gauss* is segmentation based on Gaussian smoothing and *GC* is segmentation based on graph cuts. Best performance is marked with bold. All patches are  $3 \times 3$  pixels.

experiment the atoms are build based on k-means clustering using the same dictionary sizes as obtained from the label optimized method. In some of the examples with few classes the k-means method outperforms the label-optimized method. But especially for samples with many classes, and on average, the k-means is inferior to the label optimized method.

The size of the atoms influences the performance of the algorithm, and we found small atoms of  $3 \times 3$  pixels superior. In Table 3 we show results with varying atom sizes. We also performed an experiment with varying noise levels. We added 10% and 20% Gaussian noise and the average segmentation errors rose from 6.03% to 6.19% and 15.24%. In this experiment we employed only Gaussian smoothing postprocessing.

**Detailed compositions** The dictionary atoms can contain textures from more than one texture class, which makes them suitable for segmenting detailed structures. This is illustrated

#	Init. size	Dict. size	#	Init. size	Dict. size
1	15,000	5,262	7	50,000	39,535
2	15,000	7,329	8	50,000	31,868
3	15,000	9,644	9	50,000	38,894
4	15,000	10,043	10	15,000	4,448
5	15,000	8,579	11	15,000	3,558
6	50,000	35,769	12	15,000	3,562

Table 2: Size of the dictionaries. *Init. size* is the number of patches used for initializing the dictionary and *Dict. size* is the size of the resulting dictionary.

Patch size (pixels)	$3 \times 3$	$5 \times 5$	$7 \times 7$	$9 \times 9$	$11 \times 11$
Avg.	7.81	8.53	9.89	13.00	16.38

Table 3: Average performance with varying patch size. The dictionaries was initialized with 1/10 of the size of the dictionaries in Table 2 to decrease the computational time.

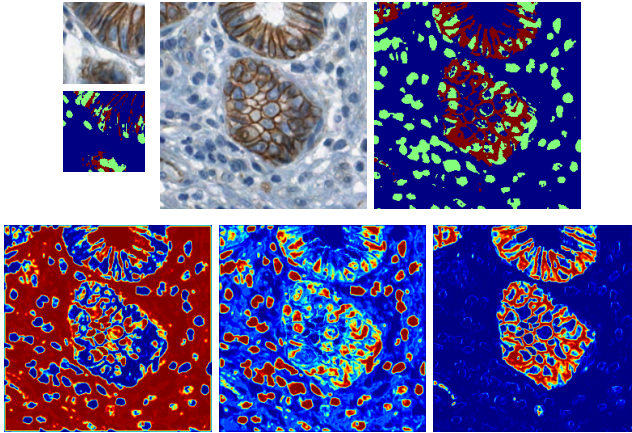


Figure 5: Examples of the segmentation of a histopathological tissue sample. The two small images upper left are training image and training mask. The resulting test image and segmentation is shown upper right. Bottom row shows the image probability maps for background (*left*), blue stained cell nuclei (*center*), and brow stained cytoplasm (*right*).  $4 \times 4$  pixel image patches were used and no smoothing.

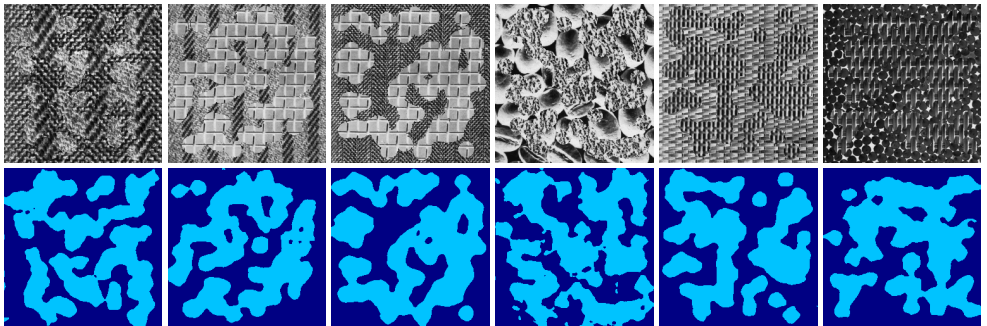


Figure 6: Segmentation examples of detailed composed images. *Top row* test images and *bottom row* segmentation results. Each image is a composition of Brodatz textures. Segmentation error from left to right are 5.48%, 4.31%, 2.97%, 10.15%, 3.28%, and 5.98%. We used  $11 \times 11$  pixel image patches and no smoothing.

in Figure 5 and 6, which shows how the method is capable of precisely segmenting detailed structures.



## 4 Discussion

We have presented a method for segmentation problems that is solved in a simple yet powerful manner. It incorporates the attractive properties of sparse methods including simplicity in training, models that are easy to interpret, and high robustness to noise. It is related to textures by learning local discriminative dictionary atoms from image data. In our experimental examples we focus on segmentation, but we believe that the method could be adapted to a range of discriminative problems.

The proposed method is simple compared to prior work because it contains all texture information in one dictionary. This is different from methods based on learning a dictionary for each class like [10, 12, 13, 14]. Employing an approximate nearest neighbor search makes the encoding complexity  $O(n \log n)$ , and having only one dictionary this search can be done very fast even for large dictionaries. This has allowed us to use much larger dictionaries than for example suggested in [10], where they use dictionaries of 128 atoms. A limitation is the memory usage, because the label dictionary needs atoms for all classes, and building the label image from overlapping atoms also requires some processing power.

We found this procedure to be highly robust to noise, which we expect to be an advantage over methods dependent on the residual encoding error as a discriminant. We expect an increased noise level to dominate the residual and this way reduce the discriminative power of this group of methods. In our future work we plan to experimentally validate this claim. For the test textures we experienced a decline in performance when adding around 15% Gaussian noise, but in other examples we were able to add up to 50% - 100% noise with only little decrease in performance. The method is especially robust for noise added to regular textures.

In this paper we have focused on segmentation, where the method is superior to prior work. Despite the limited number of samples this data contains a large variation in textures and is a challenging segmentation benchmark [10, 14]. Furthermore we have demonstrated the method for segmenting detailed image structures, where the method has the advantage that the dictionary atoms can code for more than one class label. In addition the method provides a probability map, which can be used together with for example a shape prior to give very precise segmentations.

We have also tested a k-means clustering approach, where the dictionary is build as cluster centers in the intensity image. In this approach no label information is included in the optimization. This simple approach works well for textures with few classes, but especially for textures containing many label classes this unsupervised approach is inferior.

The proposed dictionary training approach has nice properties in increasing the segmentation performance, but there is no guarantee for convergence towards a fixed dictionary or any performance expectations. An analysis of convergence properties together with effective data structures for reducing computational cost is the plan for future research.

## 5 Conclusion

We have presented a framework for solving discriminative problems based on learning a generative discriminative dictionary of local image patches employing a modified vector quantization. This dictionary of intensity atoms is coupled with a label dictionary. We learn the dictionaries from training data using an iterative optimization procedure to obtain intensity atoms that model the image well and have high discriminative power. With our

approach we obtain state-of-the-art results. The proposed method is well suited to segment detailed structures, because the dictionary atoms can contain label information for multiple classes. Our procedure is especially robust to noise, making it applicable to a range of problems.

## References

- [1] P. Brodatz. *Textures: a photographic album for artists and designers*. Dover Publications, 1966.
- [2] O. Bryt and M. Elad. Compression of facial images using the K-SVD algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008. ISSN 1047-3203.
- [3] M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010. ISBN 144197010X.
- [4] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006. ISSN 1057-7149.
- [5] M. Elad, J. L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis*, 19(3):340–358, 2005. ISSN 1063-5203.
- [6] A. K. Fletcher, S. Rangan, V. K. Goyal, and K. Ramchandran. Denoising by sparse approximation: Error bounds based on rate-distortion theory. *EURASIP Journal on Applied Signal Processing*, 2006:98, 2006. ISSN 1110-8657.
- [7] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 1981. ISSN 0028-0836.
- [8] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 147–159, 2004. ISSN 0162-8828.
- [9] T. Leung and J. Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.
- [10] M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. *WALCOM: Algorithms and Computation*, pages 274–285, 2009.
- [11] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. (CVPR)*, pages 1–8. IEEE, 2008.
- [12] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised Dictionary Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1033–1040, 2008.

- [13] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. INSTICC Press, 2009.
- [14] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997. ISSN 0042-6989.
- [15] G. Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, 2009. ISSN 0924-9907.
- [16] T. Randen and J. H. Husøy. Filtering for Texture Classification: A Comparative Study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999. ISSN 0162-8828.
- [17] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. (CVPR 2008)*, pages 1–8. IEEE, 2008.
- [18] K. Skretting and J. H. Husøy. Texture classification using sparse frame-based representations. *EURASIP Journal on Applied Signal Processing*, 2006:102, 2006. ISSN 1110-8657.
- [19] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. (CVPR)*. IEEE, 2008.