# Exploiting Low-level Image Segmentation for Object Recognition

Volker Roth and Björn Ommer

ETH Zurich, Institute of Computational Science
Universität-Str. 6, CH-8092 Zurich
{vroth,bjoern.ommer}@inf.ethz.ch

**Abstract.** A method for exploiting the information in low-level image segmentations for the purpose of object recognition is presented. The key idea is to use a whole ensemble of segmentations per image, computed on different random samples of image sites. Along the boundaries of those segmentations that are *stable* under the sampling process we extract strings of vectors that contain local image descriptors like shape, texture and intensities. Pairs of such strings are *aligned*, and based on the alignment scores a mixture model is trained which divides the segments in an image into fore- and background. Given such candidate foreground segments, we show that it is possible to build a state-of-the-art object recognition system that exhibits excellent performance on a standard benchmark database. This result shows that despite the inherent problems of low-level image segmentation in poor data conditions, segmentation can indeed be a valuable tool for object recognition in real-world images.

## 1 Introduction

The goal of image segmentation is the detection of meaningful structures from a cluttered scene. Most current segmentation techniques take a bottom-up approach, where local image properties such as feature similarity (brightness, texture, motion etc) are used to detect coherent units. Unfortunately, image segmentation becomes very difficult in poor data conditions like shadows, occlusions and noise. In such situations, the detected coherent units often do not coincide with our perception of objects in a scene.

The automatic detection and recognition of visual objects in images, on the other hand, has been among the prime objectives of computer vision for several decades. Large intra-category variations of appearances and instantiations within object classes turn learning category models into a key challenge. Therefore, common characteristics of an object class have to be captured while offering invariance with respect to variabilities or absence of these features. In principle, segmentation algorithms might help to solve the *object detection* task by partitioning the image into meaningful parts that might serve as the inputs of a classification system. Many papers on image segmentation contain statements of the form *"segmentation is an important preprocessing step for object recognition"*. Due to the above limitations, however, the practical usefulness of low-level segmentation algorithms for the purpose of object recognition is questionable and, indeed, the currently best approaches to object recognition do *not* employ low-level segmentation, see e.g. [1–5].

In order to circumvent the obvious problems of segmentation, it has been proposed to treat segmentation and recognition in an interleaved manner, e.g.[6]. Approaches of this kind typically mix bottom-up strategies with top-down elements based on back-propagating hypotheses about the objects down to the segmentation level. These methods seem to work well, if the initial segmentation is of "reasonable" quality, which is often the case if one considers *moving* objects in videos where the motion information supports the segmentation process. Good performance can also be achieved, if only a small number of classes is considered for which relatively strong initial object hypotheses can be build by including additional side information. For the task of detecting objects in still images, however, the recognition performance of these methods is still rather limited, particularly if there are many potential object classes.

Despite the generally poor quality of bottom-up segmentations in real-world images, we demonstrate that it is possible to exploit low-level segmentations for building a very powerful object recognition system. The key idea is to use not only one segmentation, but a whole *ensemble of segmentations* which often captures at least *parts* of the objects in a scene. Such partial matches of the objects boundaries can be successfully used for discriminating between foreground/background segments.

Given the candidate foreground segments, we show that it is often possible to recognize the object class. We propose two different approaches: the **direct approach** exclusively relies on the low-level segmentations by computing majority votes over all stable segments in an image, whereas the **combined approach** uses the predicted foreground segments as input of a hierarchical classification scheme. The latter learns to group parts of the image foreground segments into a hierarchy of category-specific compositions, and binds them together using a probabilistic shape model to recognize objects in scenes. The foundation for this approach is laid by the principle of *compositionality* [7]: As observed in cognition in general and especially in human vision, complex entities are perceived as compositions of comparably few, simple, and widely usable parts. Objects are represented based on their components and the relations between them. Composition models bridge the semantic gap between low level image features and high level scene categorizations [3, 4] by establishing intermediate hidden layer representations. Our experiments with the *caltech 101* database [8] show that both the direct and the combined approach allow us to build a highly competitive object recognition system.

## 2 Ensembles of Low-level Segmentations

For segmenting the images we use an adapted version of the algorithm proposed in [9] which combines both the ideas of *partitioning* and *feature combination/selection*. The latter aspect turns out to be very important for finding good segmentations, since segment-specific information is often spread over different cues like color and texture. The core of this algorithm consists of a Gaussian mixture model with built-in *relevance detection* which automatically selects important features by maximizing a constrained likelihood criterion. In order to find reasonable settings for the free model parameters, we devise a resampling-based model selection strategy which follows largely [10, 9]. The key idea is to draw *resamples* of the object set, to train the segmentation model on the individual resamples and to compare the resulting solutions. Adapted to our

image segmentation problem, this strategy translates into sampling different *image sites*, inferring a segmentation on the basis of these sites and identifying *stable* segmentations (i.e. those which can be reproduced on many different random samples of image sites). We repeat this procedure for different numbers of mixture modes, and finally we receive a *stability-ranked* list of prototypical segmentations, see [9] for further details.

In addition to selecting these stable segmentations, we also overlay all individual segmentations to compute a *probabilistic boundary map* that encodes for each pixel its probability of being part of a segment boundary, see figure 1 for a schematic overview. Despite the fact that many individual segmentations are often of rather poor quality, the ensemble approach has two important advantages: (i) Within the subgroup of *stable* segmentations we often find relatively good partitions; (ii) the aggregated boundary map typically captures many details of the object in the image. To highlight the latter issue, we have additionally plotted the response of a Canny edge-detector in the right panel of figure 1. Due to the local character of the edge detection process, the Canny edges are much more noisy than the aggregated segment boundaries.

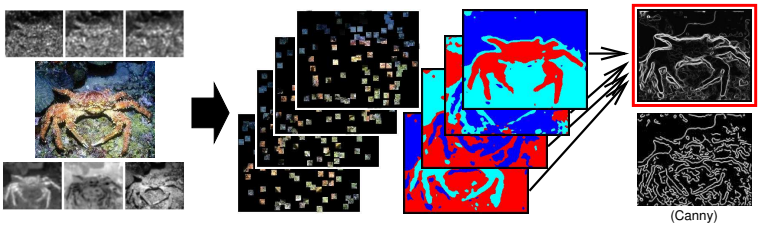


**Fig. 1.** Ensembles of segmentations. Left: input image and extracted features (top: three texture channels, bottom: LUV color channels). Middle: resampled image sites and corresponding segmentations. Top right: probabilistic boundary map found by overlaying all individual segment boundaries. Bottom right: Canny-edges for comparison.

## 3 Foreground/background Learning

In the following we assume that we are given a set of training images with *category labels*. Additional information about the location of the objects, however, is not available. We further assume that there exists a *background* category with images that belong to none of the categories. Such a situation is e.g. given for the popular *caltech 101 dataset* [8] that contains images from 101 categories. For all experiments we used the images from 20 categories in *caltech 101*: *anchor, umbrella, barrel, trilobite, wrench, windsor_chair, tick, stapler, electric_guitar, gramophone, stop_sign, cup, lobster, crayfish, water_lilly, crab, starfish, wild_cat, pyramid, pagoda*. This choice was guided by two criteria: we wanted a subset that is reasonably small ($\approx$ 1000 images) to explore a new method and that is sufficiently difficult to reliably evaluate the performance. The chosen categories are a mixture of artificial and natural object classes and they contain some classes that are very difficult to separate like *lobster* and *crayfish*. From all classes we randomly pick a *training set* of 25 images each. The remaining images are exclusively used for performance evaluation.

Based on ensembles of segmentations, we now introduce a method for identifying foreground segments. This foreground learning takes place in a *pairwise setting*. We first randomly pick two categories. For all training images belonging to these two categories, we consider all segmentations which exceed a certain stability threshold (see section 2) and we extract the boundary of each connected component. On regularly spaced points along these curves, vectors of *local image descriptors* are extracted. Thus, each connected segment is represented as a *string of vectors*. The same procedure is applied to the training images in the background class. Putting all such strings together we obtain a dataset consisting of $n$ boundary strings from two categories and the background class. We then compute *local string alignments* for all pairs of these $n$ strings. The final $(n \times n)$ matrix of alignment scores is transformed into a valid *Mercer kernel*. In order to discriminate between fore- and background segments, we learn a *Gaussian mixture model* with three modes on these data which are represented by the kernel matrix. The estimated membership probabilities in one of the modes are used for identifying foreground segments: those segments that have a high probability for the *correct* image category are treated as foreground areas.

**Boundary extraction and string representation**. After the segmentation process, each pixel in an image has a group label. In a first step, *connected* pixels which share the same group label are extracted. For simplicity, we will refer to such connected groups of pixels as *segments* in the sequel. For each of these segments, we compute a chain-code representation of the segment boundary. We call such a boundary *closed* if the segment is entirely contained in the image, i.e. if it does not hit the image borders. For such closed segments the boundary chain is extended to two full circulations, which guarantees us that the alignment score between two such segments becomes independent of the starting point (note that we use *local* alignments). If a segment is not closed, we start at the image border and continue the chain until the border is hit again. Figure 2 depicts examples of such segment boundaries.

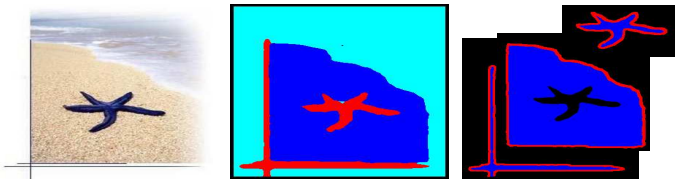

**Fig. 2.** Boundary extraction. Left: original image; middle: most stable segmentation; right: three extracted segments (blue) and their boundaries (red).

On regular intervals along the segment boundaries, we then extract a *vector of image descriptors*. The components of such a vector contain three different descriptor types: a *shape context* histogram, a *texture patch* and a *gray-value patch*. The shape context descriptor [11] consists of a log-polar histogram with 60 bins (10 angles, 6 scales) which is centered at the current position along the boundary. Each bin represents the (weighted) sum of those pixels in the map of aggregated segment boundaries which fall into the bin-specific part of the polar histogram and which are "close" to the segment, i.e. which lie in a close vicinity of the segment, see the green tube around the segment

in figure 3. The texture- and gray-value patches consist of locally averaged values of Gabor filter responses and image intensities respectively. In analogy to the shape context descriptor, a polar grid is used for defining the areas over which the averaging takes place. This polar geometry has the advantage that we can easily incorporate rotation invariance into the alignment process by simply shifting the indices of the descriptors.

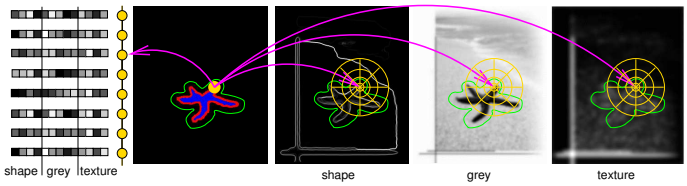


**Fig. 3.** String representation of segments. Left to right: chain of vectors containing local image descriptors (schematic), segment boundary (red) and vicinity around the segment (green), polar histogram of the shape context descriptor (yellow), polar gray-value patch, polar texture patch.

**String alignments**. Having extracted a string for each of the $n$ segments, we then compute the $n \times n$ matrix of pairwise local alignments by way of the Smith-Waterman algorithm [12]. Contrary to the typical setting in which this algorithm is used, in our case we do not have a fixed alphabet of symbols for which a predefined scoring table for aligning pairs of these symbols is available. We rather have "strings" which are ordered collections of vectors with real-valued entries. Instead of looking up the symbol-wise scores in a table, in each step of the algorithm we evaluate a scoring function for two vectors. The components of these vectors consist of 60 bins of a shape context histogram, 60 locally averaged texture measurements and 60 locally averaged gray-values. Thus, a vector is composed of three subvectors $v = (v^{\text{shape}}, v^{\text{text}}, v^{\text{gray}})^\top$

In the experiments below we use a simple aggregation of these three cues that combines $\chi^2$ distances between shape context histograms with correlation scores for texture and intensity patches: the scoring function for two vectors $v_1, v_2$ has the form

$$s(v_1, v_2) = a - b \cdot \left( D_{\chi^2}(v_1^{\text{shape}}, v_2^{\text{shape}}) + D_{cc}(v_1^{\text{text}}, v_2^{\text{text}}) + D_{cc}(v_1^{\text{gray}}, v_2^{\text{gray}}) \right), \quad (1)$$

with the $\chi^2$ distance $D_{\chi^2}(v_1, v_2)$ and the cross correlation distance $D_{cc}(v_1, v_2) = 1 - |cor(v_1, v_2)|$, with $cor(v_1, v_2)$ being the correlation between the vectors $v_1$ and $v_2$. Note that distances are transformed into similarities, so that a high score means that two strings are similar. The constants $a = 1/2, b = 1/3$ were selected empirically.

Since the extracted segments often capture only *parts* of the objects, the alignment scores are divided by the length of the alignment. In order to avoid high scores for very short "random" alignments, we consider such length-normalized alignments as "valid" only if the total alignment length exceed a certain threshold. In our experiments we require that two strings must be aligned at more that 15 consecutive positions, otherwise the score is down-weighted by a factor of ten. For a better geometric interpretation, we have depicted such positions which align to each other in figure 4 below as blue lines.

To further decrease the sensitivity to local segmentation errors, we allow *gaps* in the alignments. Such gaps are penalized by a predefined cost value $g$. In our experiments we

use $g = 0.1$ which means that the current alignment score is decreased by $0.1$ whenever a position in one string is aligned with a gap in the other. For two strings $x, y$ with lengths $l, l'$ the alignment algorithm recursively fills the $(l \times l')$ matrix $F$:

$$F(i,j) = \max\{0, F(i-1,j-1) + s(x_i, y_j), F(i-1,j) - g, F(i,j-1) - g\}. \quad (2)$$

*Backtracking* from the highest value in $F$ yields the optimal alignment, see [12] for details. Recall that the $i$-th position of string $x$ is a shape/texture/intensity-vector, and that $s(\cdot,\cdot)$ denotes the scoring function defined in (1). An example alignment matrix for the categories "wrench" and "windsor_chair" is depicted in the right panel of figure 4, which shows a distinct block structure.

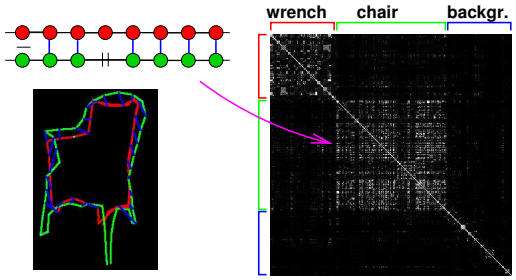


**Fig. 4.** String alignments. Left: alignment of two boundary strings (top: schematic, bottom: 2 segments from the "windsor_chair" category). The blue lines connect aligned vectors. Right: pairwise alignment matrix for segments from the categories "wrench","windsor_chair" and "background".

**Detecting foreground segments**. In the final step in our foreground-detection process a *Gaussian mixture model* is learned for the $n$ segments. These segments are represented in form of a $(n \times n)$ matrix $K$ of pairwise alignment scores. If this matrix would be positive semidefinite we could identify it as a *Mercer kernel* and train a mixture model in the kernel-induced space as proposed e.g. in [13]. It is well known that probabilistic alignment models such as *pair hidden Markov models* produce scores which fulfill the requirements of a valid Mercer kernel. For simplicity, however, we used a deterministic alignment model which might violate the positive-semi-definiteness condition. Moreover, the length-normalization of scores can lead to additional negative eigenvalues of $K$. In practice, however, we observe that there are typically only very few negative eigenvalues which are all of small magnitude. In order to transform it into a valid Mercer kernel, we use the *kernel PCA* idea [14] to find a decomposition $K = V \Lambda V^\top$ with a diagonal matrix $\Lambda$ of eigenvalues. Discarding all negative eigenvalues we form a valid kernel $K' = V_+ \Lambda_+ V_+^\top$.

Based on this kernel matrix $K$ we now learn a Gaussian mixture model with 3 mixture modes. For initialization we label all segments in an image according to its category label, despite the fact that some segments might belong to the background class. During further iterations of the EM algorithm (see [13] for details), we re-estimate these membership probabilities in one of the three classes (two categories + background) for each segment. It is interesting that the selection of foreground segments does not vary

significantly if *different pairs* of categories or if *more than two* categories are selected. Examples of detected foreground segments are depicted in figure 5.

To predict foreground segments in the *test images* we first reduce the size of the training set by extracting from each of the 25 training images per category only the *two highest scoring foreground segments*. Based on the string representations of these $2 \cdot 25 \cdot 20$ segments (2 segments/image, 25 images/category, 20 categories), we compute a new pairwise alignment matrix of size $1000 \times 1000$ which now represents *all* training images. Discarding negative eigenvalues we again arrive at a valid kernel matrix $K' = V_+ \Lambda_+ V_+^\top =: XX^\top$ that allows us to form a vectorial representation of the segments as the rows of the matrix $X$. Based on this data matrix and the corresponding category labels of the training images we learn a probabilistic 20-class kernel classifier. In the experiments we used a multi-class variant of *nonlinear kernel discriminant analysis* described in [15], which allows us to *predict* foreground segments in test images.

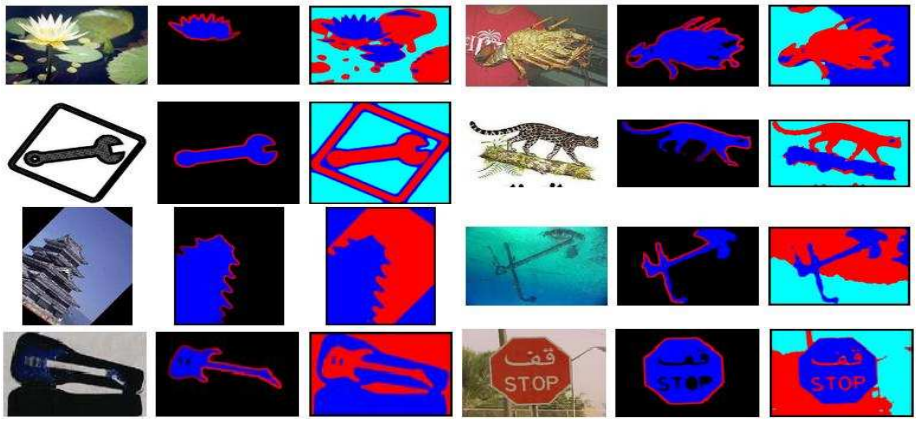


**Fig. 5.** Detecting foreground segments. From left to right in triplets: image, detected foreground segment (the one with the highest probability), corresponding stable segmentation.

## 4 Object Recognition

In order to exploit the results of the foreground identification for the purpose of object recognition, we use the classifier that was learned on the basis of the training images as described above to predict foreground segments in the *test images*. For this purpose we align each segment in a test image with *all* $n = 1000$ training segments. The resulting 1000-dimensional alignment vector is projected onto the set of eigenvectors $V_+$ of $K'$. Appropriate scaling by the eigenvalues (see [14]) yields a vectorial representation $x_t$ of the test segment, for which the classifier predicts a set of membership probabilities in each of the 20 image categories. Segments that can be clearly assigned to one of the categories (i.e. which have a high membership probability) are considered as *hypothetical foreground segments* in a test image.

These hypotheses are now used for predicting the category labels of the test images in two different ways: the **direct approach** computes a weighted majority vote over all

segments in a test image. When assigning each image the most probable category label, the average retrieval rate of the direct approach is $58.3\%$. Among the *two most probable* categories, we find the correct one in $\approx 71\%$, and among the *three most probable* in $\approx 79\%$. Taking into account that the direct approach only uses low-level segmentations and that for roughly 1/4 of all images it seems to be very difficult to find any good segmentations, these retrieval rates are amazingly high. For comparison: our reference implementation [4] (which currently is one of the best methods on the caltech 101 database) achieves an average retrieval rate of $61.8\%$ when trained exclusively on these 20 categories. For analyzing the effect of using many segmentations per image (we used 100 in the experiments), we repeated the whole processing pipeline with only 5 segmentations per image. In this setting, the average retrieval rate drops down to $26\%$ which effectively demonstrates the advantage of using large ensembles.

The **combined approach** uses the boundaries of the predicted foreground segments as input for a *compositionality-based* recognition system which implements a variant of the model in [4]. The segment boundary contours are first split into shorter subcurves before encoding them using localized feature histograms from [3]. Top-down grouping of segment boundaries yields compositions of curves with increased discriminative power compared to their original constituents. The conceptual idea is to group image parts not based on their similarity but based on the familiarity of their composition. Assume for the moment that groupings which are distinctive for categories have already been learned from the training data. The objective of top-down grouping is then to form a hierarchy of compositions by combining those constituents whose composition has highest category posterior. The goal is now to automatically learn and represent models for top-down grouping in the case of large numbers of object classes. We tackle this problem by first estimating category dependent co-occurrence statistics of foreground curve segments in the training images. Using this distribution, the curves are then grouped. The resulting compositions are used to update the previously estimated category dependent grouping statistics and to learn a *global shape model*. This shape model is used for coupling all the established compositions in a final step. Due to limited space we refer the interested reader to [4] for details about the model and its practical implementation in form of a graphical model employing believe propagation.

Our first experiments with this combined approach yielded an average retrieval rate of $62.3\%$ which is at least competitive to the reference model [4] (however, the increase in performance is probably not statistically significant). Figure 6 shows the corresponding *category confusion table*. This result shows that despite the difficulties of low-level segmentation, it is possible to exploit the information contained in ensembles of segmentations for building state-of-the-art recognition systems.

## 5 Discussion

Despite the fact that bottom-up image segmentation is sometimes considered as an important preprocessing step for object recognition, the actual usefulness of such an approach in real-world recognition scenarios is still an ongoing debate. For real-world scenes it is often difficult to find segmentations of sufficiently high quality that would allow to reliably extract object-specific information like shape, color distribution, tex-

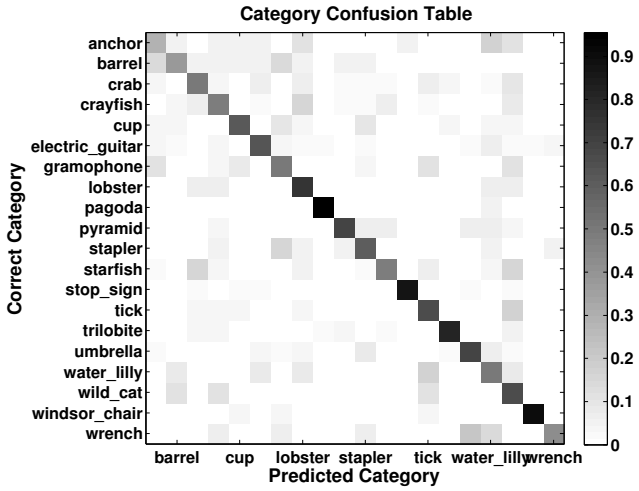


**Fig. 6.** Category confusion table of the compositional model with $62.3\%$ retrieval rate.

ture features etc. It is, thus, not surprising that the currently best object recognition systems do *not* use low-level segmentations.

In this work we show that it is indeed possible to build state-of-the-art recognition systems on the basis of low-level image segmentations. The key idea is to use not only one single segmentation per image, but ensembles of many segmentations that are computed on different random samples of image sites. Although most of the individual segmentations might be of rather poor quality, the combination of many segmentations helps to overcome problems induced by poor data conditions in two ways: (i) analyzing the variation of segmentation solutions under the sampling process, we can identify subsets of *stable* segmentations that in many cases are of much higher quality than single-segmentation solutions. (ii) Aggregating all segment boundaries, we build probabilistic boundary maps. Compared with standard edge-detectors, the aggregated segment boundaries often encode at least parts of the "true" objects in the image.

Segmentations that have been identified as *stable* are represented by local image descriptors along their boundaries. These descriptors encode *shape, intensities* and *texture* in the form of histograms of segment boundaries, gray-value patches and local Gabor filter responses. Based on this string representation, all stable segments are compared utilizing a string alignment algorithm. From the matrix of alignment scores, a Mercer kernel is derived on which a (kernelized) Gaussian mixture model is trained which is used to build hypotheses about foreground segments. The hypothetical foreground segments are then used for recognizing the objects in test images in two different ways: the *direct approach* exclusively relies on the low-level segmentation information by building weighted majority votes over all segments in an image. In the *combined approach*, the segment boundaries serve as inputs for a compositionality-based recognition system which aggregates curves (or parts thereof) to category-specific compositions.

On a 20-category subset of the *caltech 101* database we compare these two approaches with one of the currently best recognition systems which yields a retrieval rate

of 61.8 % on the considered images. We observe that even the direct approach which "naively" works on the segments without building any compositions achieves a very good performance of 58.2% (a "bag-of features" approach on the hypothetical foreground segments yields only 49%). First experiments with the combined approach even slightly outperform the base-line system (although not in a statistically significant way). As more important than the exact retrieval rates, however, we consider the following: (i) Low-level segmentations can indeed be used for building competitive object recognition systems for real-world images. (ii) The use of large ensembles of segmentations is essential (otherwise the performance drops down significantly). (iii) A comparison with the performance of the method from [4] indicates that the segmentation process concentrates relevant image information in few boundary curves and mainly discards non-discriminative image regions. (iv) We believe that both the direct- and the combined approach can be substantially improved by systematically searching for advanced local image descriptors and improved scoring functions in the alignment process.

**Acknowledgments.** This work was supported in part by the Swiss national fund under contract no. 200021-107636.

## References

1. Agarwal, S., Awan, A., Roth, D.: Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Machine Intell. **26**(11) (2004)
2. Leibe, B., Schiele, B.: Scale-invariant object categorization using a scale-adaptive mean-shift search. In: DAGM-Symposium. (2004) 145–153
3. Ommer, B., Buhmann, J.M.: Object categorization by compositional graphical models. In: EMMCVPR. (2005) 235–250
4. Ommer, B., Buhmann, J.M.: Learning compositional categorization models. In: ECCV, Springer (2006) 316–329
5. Berg, A.C., Berg, T.L., Malik, J.: Shape matching and object recognition using low distortion correspondence. In: CVPR. (2005) 26–33
6. Yu, S.X., Gross, R., Shi, J.: Concurrent object recognition and segmentation by graph partitioning. In: NIPS, MIT Press (2002) 1383–1390
7. Geman, S., Potter, D.F., Chi, Z.: *Composition Systems*. Technical report, Division of Applied Mathematics, Brown University, Providence, RI (1998)
8. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: CVPR Workshop GMBV. (2004)
9. Roth, V., Lange, T.: Adaptive feature selection in image segmentation. In: Pattern Recognition–DAGM'04, Springer (2004) 9–17
10. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Computation **16**(6) (2004) 1299 – 1323
11. Belongie, S., Malik, J., Puzicha, J.: Matching shapes. In: ICCV. (2001) 454–463
12. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
13. Roth, V., Steinhage, V.: Nonlinear discriminant analysis using kernel functions. In Solla, S., Leen, T., Müller, K.R., eds.: NIPS 12, MIT Press (1999) 568–574
14. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5) (1998) 1299–1319
15. Roth, V., Tsuda, K.: Pairwise coupling for machine recognition of hand-printed japanese characters. In: CVPR. (2001) 1120–1125