# Learning-Based Symmetry Detection in Natural Images

Stavros Tsogkas and Iasonas Kokkinos

Center for Visual Computing, École Centrale de Paris, FR
Equipe Galen, INRIA Saclay, Ile-de-France, FR
Université Paris-Est, LIGM (UMR CNRS), Center for Visual Computing, Ecole des Ponts ParisTech, FR
{stavros.tsogkas,iasonas.kokkinos}@ecp.fr

**Abstract.** In this work we propose a learning-based approach to symmetry detection in natural images. We focus on ribbon-like structures, i.e. contours marking local and approximate reflection symmetry and make three contributions to improve their detection. First, we create and make publicly available a ground-truth dataset for this task by building on the Berkeley Segmentation Dataset. Second, we extract features representing multiple complementary cues, such as grayscale structure, color, texture, and spectral clustering information. Third, we use supervised learning to *learn* how to combine these cues, and employ MIL to accommodate the unknown scale and orientation of the symmetric structures. We systematically evaluate the performance contribution of each individual component in our pipeline, and demonstrate that overall we consistently improve upon results obtained using existing alternatives.

## 1 Introduction

The importance of low-level feature detection in many high-level vision problems is well-known: good features are often a determining factor for success in tasks such as image segmentation, object recognition and motion estimation. Owning to this, improving low-level feature extraction algorithms has attracted a lot of attention in recent years and employing machine learning methods to tackle this problem has been a particularly fruitful approach. Starting from works on boundary detection [1,2,3], recent works have also used learning-based techniques for corner detection [4] and junction detection [5,6].

In this work we develop a learning-based approach to detect symmetry axes in natural images. By symmetry axes we mean contours lying in the middle of elongated structures, also referred to as *ribbons* or *ridges* in the literature. These contours locally amount to approximate reflective symmetry. Automatically extracting such structures from images can prove useful in numerous contexts. An obvious application is in image segmentation, where skeletons can serve as seeds for watershed segmentation, or to enforce "rigid" pairwise costs for MRFs. Symmetry axes can also be used to propose object part locations [7], serving bottom-up object recognition, and to transfer this knowledge among object

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VII, LNCS 7578, pp. 41–54, 2012.

classes enabling efficient learning of new models [8]. For excellent presentations of the importance and uses of symmetry in vision we defer to [9,10].

Our approach is distinct from the large body of work on silhouette-based symmetry detection in that instead of assuming that we are provided with pre-segmented shapes we aim at directly extracting symmetry axes from the image. This broadens the range of potential applications but also makes the problem more challenging. Several works have studied this problem over the previous decades [11,12,9,13,14] under the names of grayscale symmetry or skeleton, or ridge, valley, crease, or ribbon detection; it is with respect to such works that we compare our own.

Our first contribution is to construct and make publicly available a ground-truth dataset for symmetry detection. As we explain in Sec. 3, we construct our dataset by applying skeletonization on manually selected segments from the Berkeley Segmentation Dataset (BSD300) [15]. We use this dataset for both training and evaluation, and hope that it will prove useful for improving and benchmarking symmetry detection algorithms in the future.

Our second contribution, described in Sec. 4, lies in extracting multiple cues for symmetry detection. We exploit information from grayscale, color and texture cues by constructing histogram-based region discrepancy features in a manner similar to [2] as well as a spectral clustering cue, as in [16]. In particular we extract features by considering interior-exterior region combinations that correspond to symmetry axes at multiple scale and orientations, and score each orientation and scale accordingly.

Our third contribution, described in Sec. 5, consists in using Multiple Instance Learning (MIL) [17,18] during training so as to allow for uncertainty in the scale and orientation at which the features are computed. In particular, our classifier's decision is based on scale- and orientation-sensitive features. MIL allows us to handle these parameters as latent variables during training while during testing a noisy-or [19] combination is used to deliver the symmetry probability map.

Finally, in Sec. 6, we use our ground-truth to systematically evaluate every aspect of our algorithm, such as the contributions of each cue individually. We compare with previously reported methods and demonstrate a systematic improvement in performance.

## 2 Previous Work

Among the multiple notions of symmetry [10] the one that has been most broadly studied in vision is local reflective symmetry, and in particular its computation from binary shapes. This approach can be traced back to the *medial axis transform* (MAT) [20], defined as the locus of centers of maximal inscribed circles in an object. An analogous approach was taken in [21], which introduced a two-dimensional shape representation termed *smoothed local symmetries* (SLS). In recent work [22] skeleton construction is posed as an optimization problem that involves the conflicting goals of contour simplicity and good reconstruction of the original shape; methods of a similar spirit include [23,24,25,26]. Such silhouette-based methods rely on the often unrealistic assumption that boundary curves

are closed and sufficiently smooth and they also suffer from sensitivity to slight perturbations in the original shape. More importantly, they require that a binary image from a prior figure-ground segmentation is available.

Levinshtein et al. [14] present a method for extracting skeletal branches from color images based on a region segmentation process. They use superpixels extracted at different scales to build an adjacency graph and learn the probability that two adjacent superpixels represent medial point approximations of the same symmetric part. Despite its ability to parse an image into symmetric segments, this method produces axes formed by linear segments and therefore cannot handle structures with large curvature variations along the symmetry axis. As shown by our experiments, our approach can successfully handle such cases.

In grayscale images another approach is to treat symmetry axes as ridges and valleys; these are defined as local extrema of some hand-crafted analytic function of the image brightness. For instance in [27,28] the authors rely on concepts from vector analysis to introduce a joint definition for ridges and valleys in grayscale images. In [29] an algorithm is proposed that jointly determines the ridge position and corresponding bilateral distance from the boundary with sub-pixel accuracy. A way to combine multiple eigenvalues of the image Hessian in measures of ridge strength is proposed in [11], which also introduces an approach for ridge detection at multiple scales. Other multiscale schemes are introduced in [30], whereas the authors in [13] compare many of these criteria and discuss other alternatives.

## 3 Ground Truth Construction

Training a binary classifier for symmetry detection in natural images requires a ground-truth dataset of human annotations of symmetry axes. The main technical hurdle has been the definition of *symmetry axes* in natural images, since a number of factors, including shading, deformation, perspective effects and partial occlusion make exact reflective symmetry, defined as e.g. in [31], hard to occur. One option we considered was to use annotation tools such as Amazon's Mechanical Turk, but realized that this will open another "can of worms": the annotations one gets are typically noisy, while describing symmetry to non-experts is a non trivial task. Instead, we opt for an operational definition of symmetry that leverages upon the segmentations available for the Berkeley Segmentation Dataset (BSDS300) [15].

In particular, every image in the BSDS300 dataset is accompanied by 5-7 human segmentations. We combine the information provided by this dataset with a recent skeletonization algorithm [23] to extract skeletons of image segments corresponding to object parts. This way we obtain multiple binary images of segment skeletons which we combine into the final skeleton by taking their union.

Segments corresponding to the image background do not convey information useful for high-level tasks such as object recognition, so it is desirable that they be pruned. Furthermore, we need to account for noisy branches arising from the sensitivity of the skeleton extraction algorithm to minor shape perturbations. We address these issues by creating an interface that allows a human user to

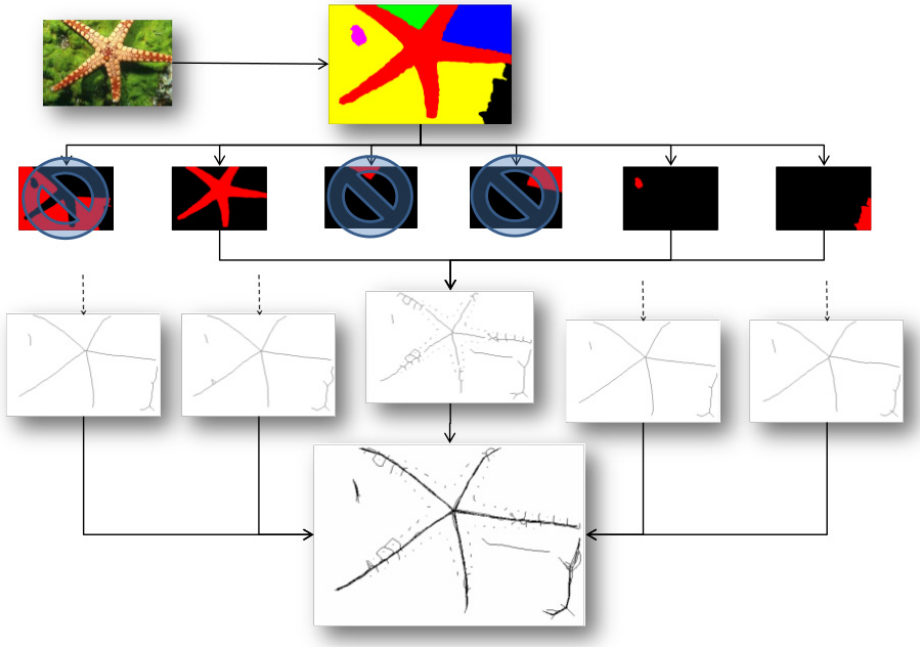


**Fig. 1.** Construction of an image skeleton by composing single-segment skeletons: the user examines one-by-one the segments provided in a human segmentation and rejects the ones that are deemed inappropriate for skeleton extraction. The process is repeated for every human-generated image segmentation and the union of the resulting skeletons forms the symmetry ground-truth.

supervise the construction of the ground truth: given a segmentation of the input image, the human user can examine every segment separately and decide whether it should be included in the overall skeleton map. By using skeletonization on top of human-generated segmentations we ensure that the symmetry axes will correspond to object boundaries. In Fig. 1 we show the partial and final skeletons obtained from the segmentations of an example image. We note here that the above procedure is applied separately for each of the multiple human segmentations available for every image in the BSDS300. We aggregate the multiple skeleton maps obtained this way into a final one by taking their union.

## 4 Feature Extraction

Our feature extraction method is inspired mainly by [2], where the authors use features extracted locally from image patches to determine the existence of a boundary at some orientation. Based on the success of the boundary model presented in that paper we examine whether similar local features can be employed

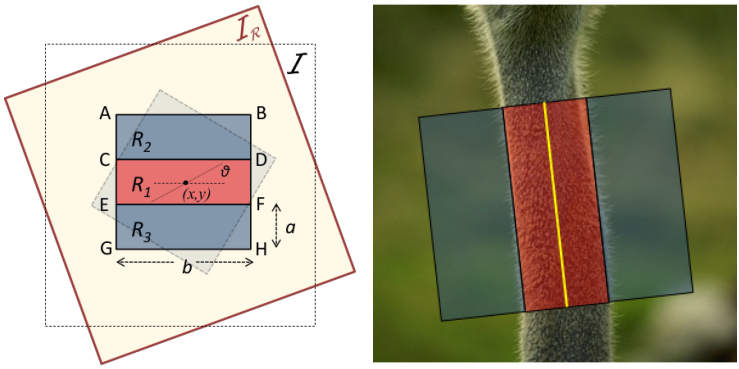



**Fig. 2.** Left: Rectangles used to rapidly calculate sums in integral images. Rotation of the filters on the original integral image $I$ is equivalent to filters aligned with axes $x-y$ in a rotated version of the image, $I_R$. Scale $s$ is equal to $\frac{a}{2}$ (typically $a = 3 \cdot b$). Right: Feature contents of the middle part show high dissimilarity to the contents of the left and right parts. Therefore the cue strength in the vertical diameter (in yellow) is high.

for symmetry detection. One significant difference in our setting is that we need to consider more and also larger sizes for the pixel neighborhoods used for feature extraction. We address this problem by creating a Gaussian pyramid of the original image and performing the feature extraction at multiple scales. To alleviate the additional computational cost of this multi-scale analysis, we replace filtering operations with integral image-based computations [32], and obtain features at various orientations by creating multiple integral images for rotated versions of the original image [33]. In the following, we describe in detail each of the individual features we use.

### 4.1 Histogram-Based Operators and Features Using Integral Images

We consider three adjacent rectangles of side lengths $a$ and $b$, labeled as shown in Fig. 2; the middle rectangle is centered at location $(x, y)$ on the image plane. For any two such rectangles we define a dissimilarity function $H_{i,j}(x, y, \theta, s)$, where indices $i, j \in \{1, 2, 3\}$ indicate which two rectangles are being compared, $\theta$ denotes orientation, and $s = a/2$ denotes scale.

Following [2], to compute $H$ first we create a histogram representation of the empirical distribution of some feature value in the pixels included in each rectangle. Then we use the $\chi^2$-distance function [34] to compare the two histograms, defined for two histograms $g$ and $h$ with $K$ bins as

$$\chi^2(g,h) = \frac{1}{2}\sum_{k=1}^{K}\frac{(g(k)-h(k))^2}{g(k)+h(k)}. \tag{1}$$

Among the many possible distance functions between histograms we selected the $\chi^2$-distance for its relative computational simplicity. Using the above notation the function $H$ becomes

$$H_{i,j}(x, y, \theta, s) = \frac{1}{2} \sum_k \frac{(R_i(k) - R_j(k))^2}{R_i(k) + R_j(k)}, \tag{2}$$

where $R_i, R_j$ are the histograms constructed from regions $i$ and $j$ respectively.

The above procedure is used separately for brightness, color, and texture features. Brightness and color histograms are formed by converting the image to the CIELAB color space and using the pixel values from the brightness channel $L^*$ and the color channels $a^*$ and $b^*$ respectively. For texture we adopt the texton approach described in [2]. We perform this computation at multiple scales for each level of a four-level Gaussian pyramid for a total of thirteen scales. We also use eight different orientations per scale. To speed-up computation we use an integral-image implementation at each scale and orientation, with different orientations being accommodated by forming the integral image after rotating the original image. Using our Matlab implementation the entire feature extraction procedure takes about 50 seconds for a $321 \times 481$ image on a 4-core Intel Xeon computer.

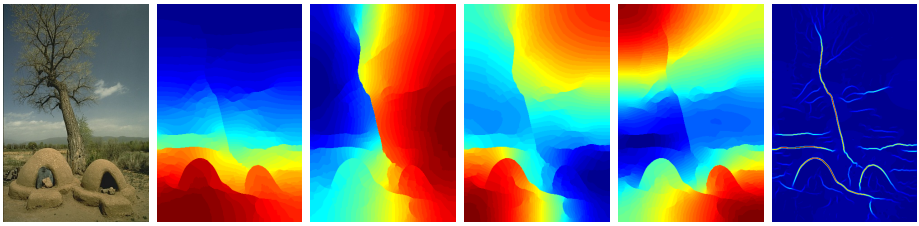
### 4.2 Spectral Clustering Feature

Spectral clustering [35] has been introduced [36] and typically used for image segmentation [37,38]. The basic idea of the algorithm is as follows: we construct a sparse symmetric matrix $\mathbf{W}$ representing pixel affinities, as produced by some spatial cue; then we use the generalized eigenvectors of the graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $D_{ii} = \sum_j W_{ij}$, to create feature vectors for each pixel. These feature vectors are used as inputs to a clustering algorithm such as $k$-means to create the partition of the image into segments.

In [16] the authors observe that these eigenvectors carry useful contour information and convolve the images formed from each eigenvector with Gaussian directional derivative filters at eight orientations to obtain the oriented signals $\nabla_\theta \mathbf{v}_k(x, y), \quad \theta \in \{0, \ldots, \frac{7\pi}{8}\}$. These "spectral" derivatives effectively locate points of significant change in the eigenvector values while overlooking smooth changes that can lead to incorrect breaking up of large uniform regions.

We adopt a similar approach to derive a spectral feature for symmetry. We first train a detector using the histogram features described in the previous section. We use the responses of this first-stage detector to create the affinity matrix $\mathbf{W}$, calculating affinities with a functional similar to the intervening contour cue used in [16] but replacing boundary strength with the output of the first-stage detector. Then we solve for the corresponding generalized eigenvectors and convolve the eigenvector images with the Gaussian directional derivative filters. In the same fashion as in [16], we combine the results into a single *spectral* component for our symmetry detector using

$$\text{spSym}(x, y, \theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} \cdot \nabla_\theta \mathbf{v_k}(x, y), \tag{3}$$



(a) Input image (b) Eigenvector images (c) Spectral cue

**Fig. 3.** Spectral symmetry feature: using as input image (a) we setup a generalized eigenvector problem of the form described in Sec. 4.2. The first four eigenvectors are shown in the center, while (c) shows the magnitude of the spectral symmetry feature.

where $\lambda_k$ is the $k-$th larger eigenvalue. In Fig. 3 we plot the eigenvectors and the spectral component for $n = 5$. The value $n$ of eigenvectors can vary depending on the image content. We found that a value between 30 and 50 gives reasonable results for most images; in our experiments we use $n = 50$. This spectral component can now be used as an additional feature to enhance our detector's performance. A new, *global* symmetry detector is constructed as a linear combination of the histogram-based detector and the spectral feature:

$$\text{gSym}(x, y, \theta, s) = \sum_i \beta_i H_i(x, y, \theta, s) + \beta_s \cdot \text{spSym}(x, y, \theta), \tag{4}$$

where $i$ is used here to index the features described in Sec. 4, $\beta_i$ are their corresponding weights, and $\beta_s$ the weight of the additional spectral feature. The algorithm we use to learn the weights $\beta_i$ and $\beta_s$ is described in Sec. 5. Note that we use the same spectral signal for all scales.

We have observed experimentally that the spectral component acts complementary to the histogram gradient features whose responses are mostly driven by local information. The former extracts only the most salient ridges and its contribution consists in reducing clutter after non-maximum suppression and connecting broken parts of the same symmetry axis.

### 4.3 Feature vector combinations

The process described in the previous sections produces a 13-dimensional feature vector for each pixel: three $L^*$ channel histogram-difference features, one for each pair of rectangles $(H^L_{i,j}, (i,j) \in T$, with $T = \{(1,2),(1,3),(2,3)\})$, nine for the color and texture channels $(H^a_{i,j}, H^b_{i,j}$, and $H^t_{i,j}$ respectively, $(i,j) \in T)$ and the spectral feature. This feature vector is extracted at all orientations and scales except for the spectral component which varies only across orientations. In the case of grayscale images we omit the features corresponding to the $a^*$ and $b^*$ color channels resulting in a 7-dimensional feature vector.

The selection of this feature set can be intuitively justified as follows: we believe that points lying on a symmetry axis at orientation $\theta$ and scale $s$ exhibit high dissimilarity in terms of some of the used cues when compared to their surroundings (see Fig. 4.1). Consequently, for a pixel at location $(x, y)$, $H_{1,3}(x, y, \theta, s)$ and $H_{2,3}(x, y, \theta, s)$ are likely to have high values, while we can expect $H_{2,3}(x, y, \theta, s)$ to have a small value. Different cues and rectangle combinations provide complementary, and potentially also conflicting information; we now proceed to describe how we learn to combine these measurements for symmetry detection.

## 5 Training with Multiple Instance Learning

We use Multiple Instance Learning (MIL) to train our detector as this allows us to treat both scale and orientation as latent variables. In particular, our features are scale- and orientation- dependent. Training our detector in the standard supervised setting would require choosing a scale and orientation combination for every symmetry point. Instead we leave this decision to MIL, which provides a principled, and heuristic-free approach to accommodate the unknown scale and orientation parameters.

In traditional supervised learning we have a training dataset consisting of input-output pairs. In a classification problem the inputs are the *instance examples* $\{\mathbf{x_1}, \mathbf{x_2} \ldots, \mathbf{x_n}\}$ and the outputs are *labels* that denote the class among a set of $K$ possible classes; for these labels we use the notation $\{y_1, y_2 \ldots, y_n\}$. Instance examples $\mathbf{x_i}$ typically lie in $\mathbb{R}^d$, $y_i$ in $\{0, 1\}$, and the goal is to construct a classifier function $h : \mathbb{R}^d \to \{0, 1\}$ that can predict outputs/labels for novel inputs. In the MIL paradigm labels are instead assigned to *sets* of instances called *bags*. The training set consists of the set of bags $\{X_1, X_2 \ldots, X_n\}$ and the bag labels $\{y_1, y_2 \ldots, y_n\}$, where $X_i = \{\mathbf{x_{i1}}, \mathbf{x_{i2}} \ldots, \mathbf{x_{im}}\}$, $\mathbf{x_{ij}} \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ for a binary classification problem. A bag is considered *positive* if it contains at least one positive instance, whereas a bag is considered negative if all its instances are negative.

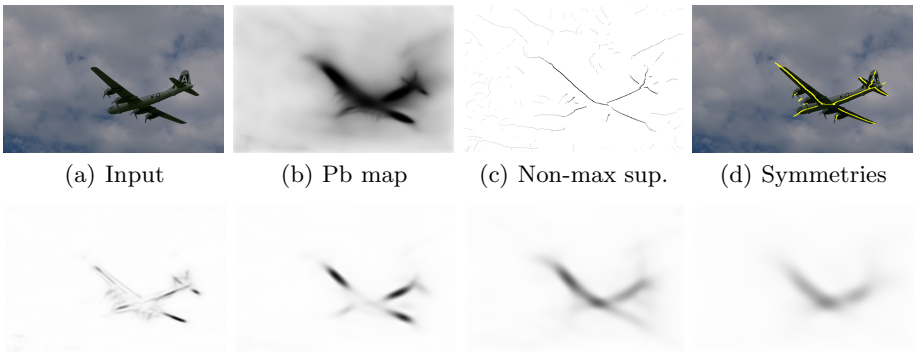
The training criterion is expressed in terms of the probabilities $P_i$ assigned to bags that should acquire label $y_i$. The standard (negative) log-likelihood cost can be written as:

$$C = -\sum_{i}^{N} \{y_i \log(P_i) + (1 - y_i) \log(1 - P_i)\}. \tag{5}$$

The particular twist in MIL is that the bag probabilities are expressed in terms of the instance probabilities. The latter are given by a logistic function of the form $p_{ij} = (1+e^{-(\mathbf{w}^T\mathbf{x})})^{-1}$ while the bag probabilities are obtained by a *noisy-or* combination rule:

$$P_i = 1 - \prod_{j} (1 - p_{ij}). \tag{6}$$

What we present above constitutes a simple setting of MIL training; we refer to [17] for a more thorough presentation of MIL alternatives.



(a) Input (b) Pb map (c) Non-max sup. (d) Symmetries

(e) Probability responses at increasing scales

**Fig. 4.** Top row: processing steps from the initial image to the final symmetry axis map. Bottom row: probability responses before non-maximum suppression at increasing scales from left to right.

In our problem every image pixel represents a bag of features. The instances contained in such a bag are the feature vectors at all orientations and scales. To give a concrete example, in our experiments we use features at 8 orientations and 13 scales for each pixel, for a total of 104 instances in each bag. For the log-likelihood optimization step we use conjugate gradients to compute search directions and quadratic and cubic polynomial approximations to perform line search.

We show the results of the trained symmetry detector in Fig. 4. We estimate the probability-of-symmetry at every pixel and every scale and orientation combination, resulting in a 4-dimensional symmetry map. These probabilities are combined with the noisy-or rule into an aggregate symmetry response. As this can result in a diffuse response, a non-maximum suppression step is used for thinning prior to thresholding.

## 6 Results

We quantitatively validate the performance of our detector using an evaluation protocol employing the precision-recall framework, identical to the one in [2]. The constructed ground-truth data consist of 5-7 different symmetry axis maps per image, resulting from the available segmentations in the BSDS300. We begin by thresholding the detector response and matching the result with each ground-truth map separately. If a detected positive is matched with at least one of the binary maps, it is classified as true positive. On the other hand, pixels that correspond to no ground-truth map are declared false positives. The hit rate is averaged over the number of different ridge maps. Perfect recall is achieved when every ground-truth point corresponds to some point declared positive by our system. We allow correspondence between detected positives and pixels neighboring

to the ground-truth positives to take into account small localization errors in the ground-truth.

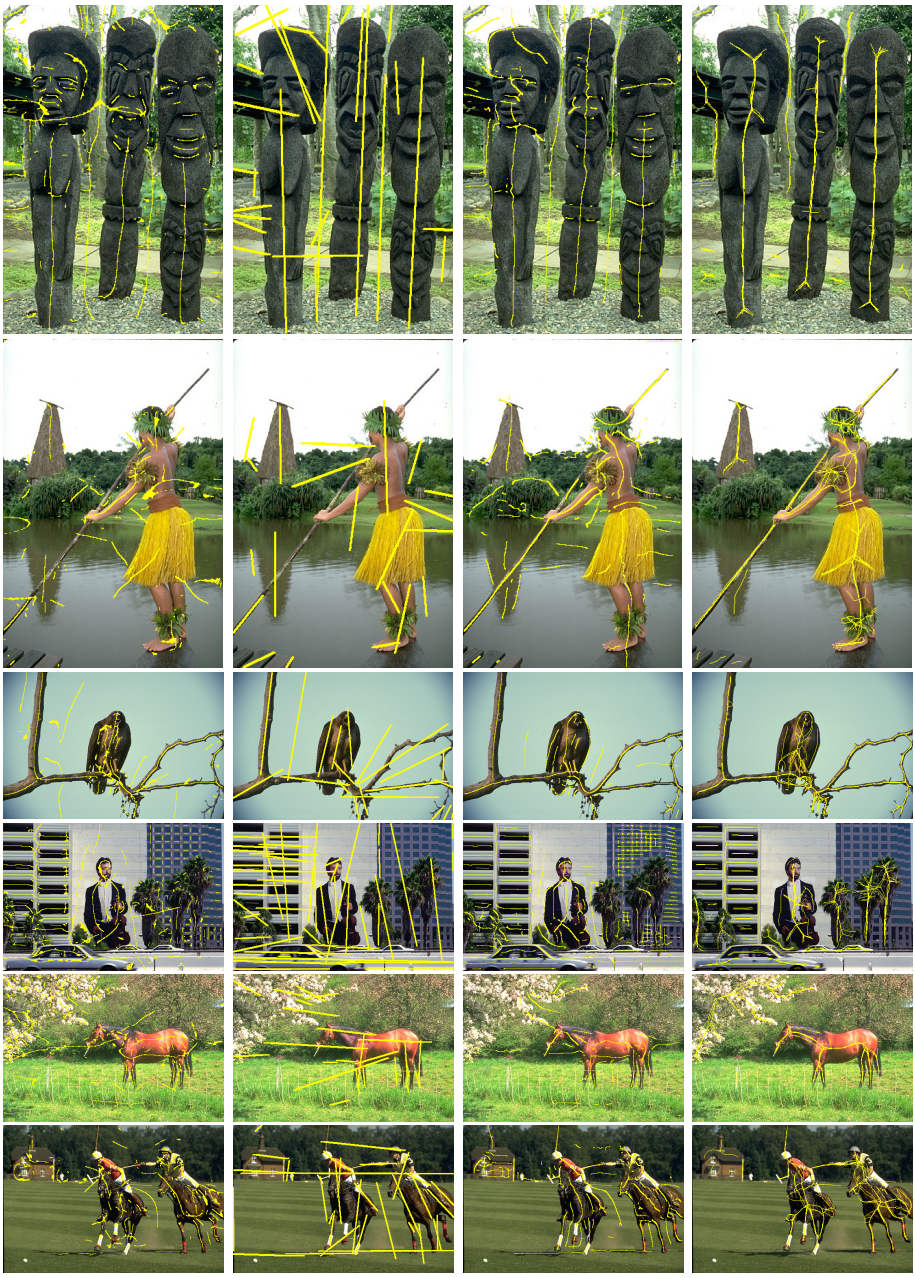
We can assess the hardness of the task at hand by assessing human performance on this task. A proxy for human performance can be obtained by evaluating the skeletons delivered by human segmentations. For this we associate our ground-truth data with an F-measure value by picking sequentially each one of the binary maps and treating it as a thresholded detector output. This is followed by matching with the remaining binary maps for the same image in the same way as described above. The score delivered by this procedure was $F = 0.73$ and the corresponding iso-curve is shown in 5(a); we can interpret this as an upper bound on what we can expect to receive by our machine-generated symmetry maps.

In Fig. 5(a) we show the precision-recall curves associated to our method, obtained by thresholding the detector response at different values. In order to confirm the importance of each separate feature used to train our detector, we compare with the results of detectors trained with a subset of the available features. We also compare its performance against the Lindeberg ridge detector, implemented as in [7], and we see that we attain a steadily better maximum F-measure throughout the precision-recall spectrum.

On the same plot we also compare the performance of our algorithm against the more recent approach of Levinshtein et al. [14], even though they do not deal with exactly the same problem: in their work they learn affinities between adjacent superpixels of the original image and cluster them to form symmetric parts. These parts are then clustered anew in groups belonging to the same object, and the symmetry axes of these regions are extracted as the major axes of appropriately fitted ellipses. Their algorithm effectively comes up with a parsing of the image into regions, while our algorithm stops at a level before that and


(a) Precision-recall curves (b) Lindeberg (c) Levinshtein (d) This work

**Fig. 5.** Quantitative and qualitative comparison of our detector with other methods. In (a) we plot the performance of our detector trained with different feature combinations to illustrate the boost due to color and spectral features. The ground truth F-measure is represented by the red iso-curve. For (b) we use the implementation of [7] and for (c) the available code of [14]. More qualitative results are included in Fig. 6.



(a) Lindeberg [11,7] (b) Levinshtein [14] (c) This work (d) Ground-truth

**Fig. 6.** Qualitative results for all three compared methods and respective ground-truth

provides longer, continuous contours, even with large curvature, as demonstrated in 5(c). Finally, the authors in [14] train their detector on the Weizmann dataset and not in natural images, as we do.

Having mentioned these differences, we believe that it is reasonable to compare our algorithm to theirs since we both aim at indicating symmetric structures in images. Since Levinshtein's algorithm returns binary results we do not include a precision-recall curve in Fig. 5(a). We evaluated its performance on the testing dataset though, treating the binary image as a thresholded probability map and seeking pixel correspondences with the ground truth ridge maps in the same way as for the other methods. This gave an overall F-measure of 0.355.

Apart from the performance difference - which as we mention above, may involve caveats due to the differences in the training set and the particular features being used - we argue that our approach has merit due to relying on local cues, instead of using region detection as a front-end, while also providing continuous contours, which can be subsequently broken and re-grouped at will. As such, our method does not make "early commitments" from which it may be hard to recover post-hoc.

## 7 Conclusions

In this work we have introduced a novel approach to symmetry detection. Our main contribution is the introduction of a machine learning framework that exploits low-level and global, spectral features to train a symmetry detector. We have constructed a symmetry axis ground-truth dataset on top of the Berkeley Segmentation Dataset using human-assisted segment skeletonization. We make this dataset publicly available and hope it will spur further research in symmetry detection from natural images. Finally we use Multiple Instance Learning to accommodate for the unknown scale and orientation of symmetry axes during training, by treating both as latent variables.

We have evaluated our method on our dataset and validated that it outperforms existing works on symmetry detection. We demonstrate the importance of our additional features, namely color and texture, which enhance the more common gray-scale information.

One of the main merits of our approach is its flexibility; our detector can be tailored to a specific application through training over an according ground truth dataset and application-driven features. This can lead to a significant improvement in performance on the task under question, which can range from medical image analysis to body part detection; we intend to explore these directions in future work. On top of that, we can straightforwardly enhance our system by adding more features in the training process.

The most computationally intensive part of our method is the extraction of the histogram and spectral features. In future work we intend to exploit the fact that extraction can be performed independently at different scales and orientations, and port our code to GPU to achieve real-time feature extraction. For the solution of the generalized eigenvector problem there is already a CUDA

implementation presented in [33] which can be used. We also intend to increase the number of features used in training, with the most prominent choice being SIFT descriptors [39], which have already been used in learning approaches for boundary detection [40].

Another potential source of improvement would be the replacement of our current, semi-automated ground-truth construction procedure with the acquisition of a thoroughly human-annotated dataset. We are currently in the process of constructing such a dataset and intend to use it and distribute in future works.

**Acknowledgements.** This work was funded by grant ANR-10-JCJC-0205. We thank the reviewers for their constructive feedback and the authors of [2,16] for making their code available.

## References

1. Konishi, S., Yuille, A.L., Coughlan, J.M., Zhu, S.C.: Statistical Edge Detection: Learning and Evaluating Edge Cues. PAMI (2003)
2. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI (2004)
3. Kokkinos, I.: Boundary Detection Using F-Measure-, Filter- and Feature- ($F^3$) Boost. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 650–663. Springer, Heidelberg (2010)
4. Rosten, E., Drummond, T.W.: Machine Learning for High-Speed Corner Detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
5. Apostoloff, N., Fitzgibbon, A.W.: Learning spatiotemporal t-junctions for occlusion detection. In: CVPR (2005)
6. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: CVPR (2008)
7. Kokkinos, I., Maragos, P., Yuille, A.: Bottom-up & top-down object detection using primal sketch features and graphical models. In: CVPR, vol. 2, pp. 1893–1900. IEEE (2006)
8. Stark, M., Goesele, M., Schiele, B.: A shape-based object class model for knowledge transfer. In: ICCV (2009)
9. Siddiqi, K., Pizer, S.: Medial Representations. Springer (2009)
10. Liu, Y.: Computational symmetry in computer vision and computer graphics. Now publishers Inc. (2009)
11. Lindeberg, T.: Edge detection and ridge detection with automatic scale selection. IJCV (1998)
12. Pizer, S., Burbeck, C., Coggins, J., Fritsch, D., Morse, B.: Object shape before boundary shape: Scale-space medial axes. JMIV (1994)
13. Lôpez, A., Lumbreras, F., Serrat, J., Villanueva, J.: Evaluation of methods for ridge and valley detection. PAMI (1999)
14. Levinshtein, A., Dickinson, S., Sminchisescu, C.: Multiscale symmetric part detection and grouping. In: ICCV (2009)
15. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV (2001)

16. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI (2011)
17. Babenko, B., Dollár, P., Tu, Z., Belongie, S., et al.: Simultaneous learning and alignment: Multi-instance and multi-pose learning (2008)
18. Keeler, J., Rumelhart, D., Leow, W.: Integrated segmentation and recognition of hand-printed numerals. In: NIPS (1990)
19. Viola, P., Platt, J., Zhang, C.: Multiple instance boosting for object detection. In: NIPS (2006)
20. Blum, H., et al.: A transformation for extracting new descriptors of shape. In: Models for the Perception of Speech and Visual Form (1967)
21. Brady, M., Asada, H.: Smoothed local symmetries and their implementation. IJRR (1984)
22. van Eede, M., Macrini, D., Telea, A., Sminchisescu, C., Dickinson, S.: Canonical skeletons for shape matching. In: ICPR (2006)
23. Telea, A., Van Wijk, J.: An augmented fast marching method for computing skeletons and centerlines. In: Eurographics (2002)
24. Telea, A., Sminchisescu, C., Dickinson, S.: Optimal inference for hierarchical skeleton abstraction. In: Pattern Recognition (2004)
25. Demirci, M., Shokoufandeh, A., Dickinson, S.: Skeletal shape abstraction from examples. PAMI (2009)
26. Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S.: Hamilton-jacobi skeletons. IJCV (2002)
27. Haralick, R.: Ridges and valleys on digital images. In: CVGIP (1983)
28. Eberly, D., Gardner, R., Morse, B., Pizer, S., Scharlach, C.: Ridges for image analysis. JMIV (1994)
29. Steger, C.: An unbiased detector of curvilinear structures. PAMI (1998)
30. Pizer, S., Eberly, D., Fritsch, D., Morse, B.: Zoom-invariant vision of figural shape: The mathematics of cores* 1. In: CVIU (1998)
31. Weyl, H.: Symmetry. Princeton University Press (1983)
32. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR (2001)
33. Catanzaro, B., Su, B., Sundaram, N., Lee, Y., Murphy, M., Keutzer, K.: Efficient, high-quality image contour detection. In: ICCV (2009)
34. Rubner, Y., Puzicha, J., Tomasi, C., Buhmann, J.: Empirical evaluation of dissimilarity measures for color and texture. In: CVIU (2001)
35. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS (2002)
36. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI (2000)
37. Cour, T., Benezit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. In: CVPR (2005)
38. Maji, S., Vishnoi, N., Malik, J.: Biased normalized cuts. In: CVPR (2011)
39. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV (2004)
40. Kokkinos, I.: Highly accurate boundary detection and grouping. In: CVPR (2010)