

Highly Accurate Boundary Detection and Grouping

Iasonas Kokkinos
Laboratoire MAS, Ecole Centrale de Paris
INRIA Saclay, GALEN Group

Abstract

In this work we address boundary detection and boundary grouping. We first pursue a learning-based approach to boundary detection. For this (i) we leverage appearance and context information by extracting descriptors around edgels and use them as features for classification, (ii) we use discriminative dimensionality reduction for efficiency and (iii) we use outlier-resilient boosting to deal with noise in the training set. We then introduce fractional-linear programming to optimize a grouping criterion that is expressed as a cost ratio. Our contributions are systematically evaluated on the Berkeley benchmark.

1. Introduction

A revival of research on boundary detection has been observed during the last years, largely due to the introduction of ground-truth labeled data and the treatment of the problem in a machine learning framework [15, 20]. This has resulted in consistent improvements on these benchmarks [25, 7, 2, 19, 24, 18, 5], while weeding out heuristic and ad-hoc approaches used in the past.

As the performance of boundary detection steadily improves, using boundaries for higher-level tasks such as recognition has received increased interest lately, as shape seems to be the missing piece in the current, appearance-dominated, object recognition research. However, in order to use boundaries for such tasks we need to relate image and object contours, which is hindered by the fragmentation problem, as failures in the front-end detector due to occlusions or shading are inevitable. Therefore a contour grouping stage is essential in order to provide a higher-level module with long and informative contours.

Our work make a contribution towards accurate boundary detection by pushing further the machine learning approach, while also providing a simple and efficient solution to the contour grouping problem. Our first contribution, presented in Section 2, is to extract SIFT descriptors [16] at multiple scales around each candidate edgel to describe the context in which each edgel appears. These descriptors are

discriminatively projected into a lower-dimensional space, where a boosting-based classifier is trained in a manner that is resilient to outliers.

We then turn to the contour grouping problem, where we revisit the idea of optimizing the length-normalized saliency of a contour. The resulting problem is typically solved using challenging discrete optimization algorithms [13, 9, 30], which make contour grouping inaccessible to some extent to the non-expert. In Section 3 we rephrase the optimization problem as a linear-fractional program, which can be transformed into an equivalent linear program. This substantially simpler formulation can be solved efficiently using standard linear programming libraries, thereby minimizing the implementation effort required to use perceptual grouping -we also make our contour grouping code publicly available.

Our contributions are experimentally evaluated on the Berkeley Segmentation Benchmark, demonstrating systematic improvements over most other comparable detectors.

2. Boundary Detection

2.1. Prior Work

The introduction of ground-truth labeled datasets [15, 20] and the phrasing of edge detection as a pattern recognition task has led to systematic improvements based on Boosting [7], topological properties of the image [2], Normalized Cut eigenvectors [19], multiscale processing [24] and sparse dictionaries [18], among others. The current state-of-the-art is held by [3] where the detector of [19] is combined with image segmentation. Moreover, the merits of perceptual grouping were explored in [25, 9, 30], demonstrating that it boosts performance in the high-precision regime by eliminating isolated edge fragments.

2.2. Front-end Boundary Detection

We start by applying a simple boundary detector to the image and keeping all points where its response is above a conservative threshold. These points are then used for a more elaborate evaluation, while the rest of the image is classified as being non-boundary. This reduces the number of processed pixels by typically two orders of magnitude.

To minimize the number of false negatives, we combine the Canny (Gradient-Magnitude) and the PB edge detectors [20]. The first has high recall, i.e. identifies all object boundaries, while the second has generally higher precision at the same level of recall. We therefore keep the response of Canny whenever PB is below a certain threshold and retain the response of PB elsewhere.

2.3. Descriptor Extraction

We classify each candidate edgel based on the distribution of gradients in its vicinity, captured by SIFT descriptors [16]. Apart from being invariant to multiplicative and additive changes in the image, SIFT features are also robust to small displacements. As such, they provide a robust feature vector around each point, capturing the local image context.

We simplify the boundary classification task by extracting SIFT descriptors aligned with the orientation of the initially detected edgels, thereby discarding variation due to orientation. However, we do not determine the scale at which the descriptor is extracted: on the one hand, edges are inherently one-dimensional signals, so their scale cannot be reliably estimated. On the other, [24] empirically validated that edges are detected more reliably based on their persistence at multiple scales. We therefore extract SIFT features at three different scales (2,4 and 8 pixels per SIFT bin), using the complementary information captured at different scales to identify edges. Finally, we exploit color information by extracting descriptors in the color-opponent channels along the lines of [1]. Summing up, we compute 9 SIFT descriptors at each point, using 3 channels at 3 different scales; descriptors are extracted with the code of [28].

2.4. Discriminative Dimensionality Reduction

The front-end processing described above yields a 9×128 -dimensional feature vector which can be used as input to a machine learning algorithm. It is however preferable to project the data onto a low-dimensional space beforehand both for efficiency and to improve classification.

PCA was initially used to compress SIFT features in [14]; more recent works [12, 22] have focused on discriminative dimensionality reduction techniques that preserve the distances between points from different patches in the lower-dimensional space. In our case however the task is classification instead of nearest neighbor search, so we resort to LDA-type techniques that provide low dimensional features that can separate the classes. As the dimensionality of the subspace recovered by LDA for a c -class problem is $c - 1$, in our binary classification case we can only recover a one-dimensional feature, which is too restrictive.

We therefore use the work of Cook and coworkers [6] to recover a linear projection of high-dimensional data that preserves the information required for classification. In spe-

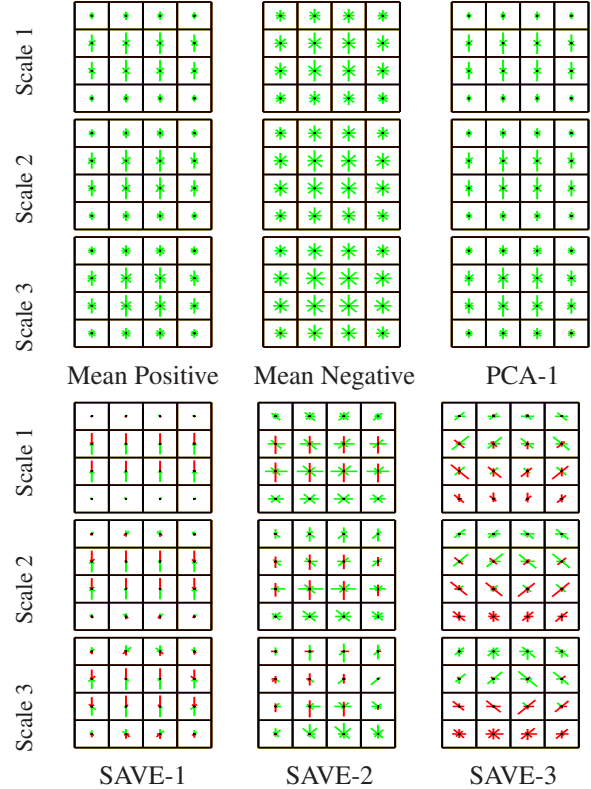


Figure 1. Multi-Scale Descriptors and Dimensionality Reduction. Each column corresponds to a multi-scale SIFT descriptor extracted from three consecutive scales of the gray channel. In the top row we show the mean of the positive and negative classes as well as the first PCA eigenvector for the positive class. In the second row we show the first three projection dimensions computed via SAVE. Red arrows correspond to negative elements.

cific, a $d \times N$ projection matrix B is sought such that:

$$P(y|Bx, x) = P(y|Bx), \quad (1)$$

i.e. conditioned on the d elements of Bx , the class label y is independent of the N -dimensional feature vector x . We experimented with several of the techniques described in [6], and found Spliced Average Variance Estimation (SAVE) to give the best performance. Due to lack of space we describe SAVE only as an algorithm, referring to [6] for details:

We first estimate means and covariances, $\mu_0, \Sigma_0, \mu_1, \Sigma_1$ for the features belonging to the two classes, and μ, Σ for the whole training set. We then compute:

$$\nu = \Sigma^{-1/2}(\mu_1 - \mu_0) \quad \Delta = \Sigma^{-1/2}(\Sigma_1 - \Sigma_0)\Sigma^{-1/2} \quad (2)$$

Subsequently we form the matrix $M = (\Delta; \nu)$ and compute its Singular Value Decomposition: $USV^T = M$. The optimal $d \times N$ dimensional projection matrix B is given by the first d columns of U : $B = U(:, 1:d)^T$.

Even though we use approximately 10^6 positive and negative points, Σ can still be ill-conditioned. For this, after

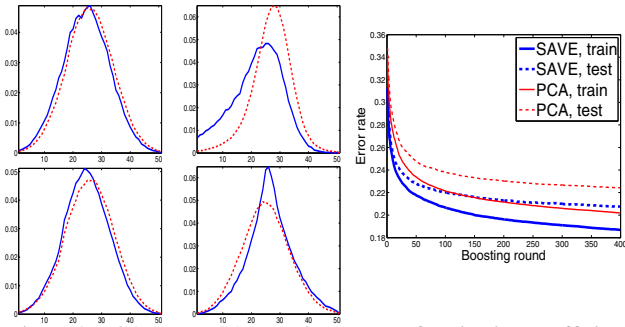


Figure 2. PCA versus SAVE: Histograms of projection coefficients onto the 3rd and 4th basis elements of PCA (left) and SAVE (right) for the positive (blue) and negative (red-dashed) classes: the projection computed by SAVE is more discriminative. The SAVE-based classifier therefore performs better during both training and testing.

SVD and prior to inversion in (2), we update all of Σ 's eigenvalues as: $\lambda_i = \max(\lambda_i, 1/10^{-3}\lambda_1)$, where λ_1 is the maximal eigenvalue.

The first elements of U are shown in Fig. 1. For instance the second vector sums the gradient strengths along the same orientation with the edge and subtracts them in the perpendicular orientation, thereby measuring the ‘purity’ of the edge in its context. As shown in Fig. 2, while being of the same complexity as PCA, SAVE yields better separated classes. Further, unlike LDA, SAVE allows us to control the dimensionality of the projection; we use $d = 100$.

2.5. Classifier Construction

As in [15, 20, 7], we treat the combination of the extracted features into a decision about the presence of an edge as a learning problem, and use the Berkeley benchmark for training and testing. We use Adaboost [10, 11] which can be summarized as follows:

Given: $(x^i, y^i), x^i \in \mathcal{X}, y^i \in \{-1, 1\}, i = 1, \dots, N$

$D_1^i = \frac{1}{N}, \forall i$

for $t = 1$ to T **do**

- (a) Find classifier h_t with smallest weighted error, ϵ_t
- (b) Compute voting strength a_t of h_t :
- (c) Update weights of samples.
- (d) Normalize weights to have unit sum

end for

Output $f(x) = \sum_t a_t h_t(x)$

The algorithm learns at each round t a simple classifier h_t on weighted training data, determines its influence a_t on the final decision, identifies poorly classified samples based on the product of the label y and the weak learner’s output $h(x)$, and increases their importance for the next round.

Weak learners: Gentleboost We use Gentleboost [11], which is a well-behaved variant of [10]. Gentleboost views the class labels as continuous targets, and minimizes their reconstruction error in terms of the weak learner’s outputs. At each round the weak learner and its voting strength are

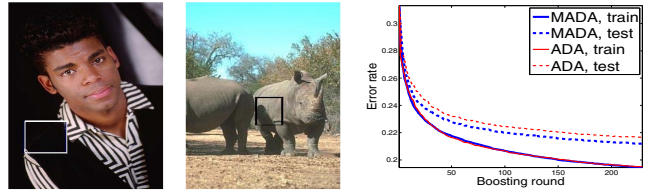


Figure 3. Outliers & Madaboost: The shoulder is labeled as a boundary, while the discontinuity on the rhino’s torso is interpreted as a shadow. For all our classifier can know, these points are outliers. The learning curve shows that Madaboost deals gracefully with outliers and generalizes better than Adaboost.

determined by a weighted least squares fit, using as weights the current distribution on samples. We use level-3 Decision trees; these perform better than decision stumps, which operate on a single feature dimension at a time.

Outliers: Madaboost Outliers harm the generalization performance of Adaboost as their weights increase at each round of boosting. Eventually they ‘hijack’ the training process and lead to overfitting. Two examples of outliers for our problem are shown on the right of Fig. 3: such points are impossible to classify based on low-level cues, and should thus be ignored during training with Adaboost.

For this we use Modified Adaboost (Madaboost) [8] which moderates the weight that outliers have in larger rounds of boosting. Contrary to other robust variants of Adaboost (see e.g. ref.s in [21]), Madaboost is compatible with Gentleboost, as it only modifies the weighting D of the data, leaving the rule for setting a_t intact. Specifically Madaboost keeps track of the weight that each data point would have originally, $D_1^i w$, where $w = \exp(-\sum_{i=1}^t a_t y_t h_t(x^i))$ as in Adaboost, and then thresholds it by its initial weight. This means that step (c) becomes $D_{t+1}^i \propto \min(D_1^i, D_t^i w)$. As shown in Fig. 3, this results in systematically better performance on the test set.

2.6. Comparative Detection Results

In Fig. 4 we compare visually the Berkeley PB detector [20] to our detector; a systematic evaluation follows in Sec. 4. We show the probability of a pixel being a boundary (higher is darker), as estimated from different detectors; we turn the classifier’s output into a probability as in [11]. Our results are more robust to changes in contrast, while the PB detector gives faint responses at low-contrast boundaries, e.g. the seaside or the feet of the horse. We attribute this to SIFT, as it discards multiplicative changes in the image. The last row illustrates the improvements from the grouping algorithm described next: salient boundaries are enhanced, while short and isolated fragments are penalized.

3. Boundary Grouping

We now introduce a simple and efficient approach to the optimization of a normalized saliency criterion used for

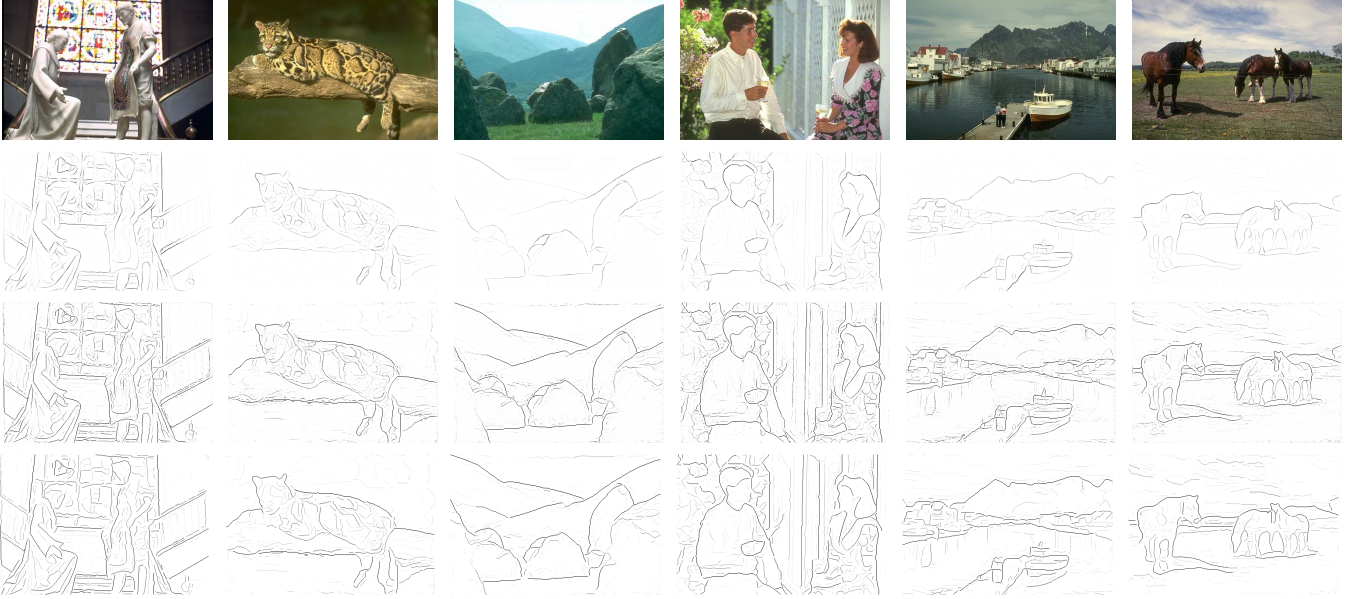


Figure 4. Comparative results: Row 1: Test image from Berkeley Benchmark. Row 2: Boundaries of the Berkeley edge detector of [20]. Row 3: Boundaries from boosting-based detection. Row 4: Boundaries from boosting followed by grouping. Please see on screen.

grouping. We start by describing the problem in a general setting, and then narrow down to a specific formulation that we employ and optimize in our work.

3.1. Cost Ratio Criterion

The criterion that drives our contour detection is a combination of a smoothness term that favors smooth contours and a detector-based measure of local edge strength. Even though our criterion does not formally stem from Bayes' rule, we will use the terms 'prior' and 'probability' to motivate the terms appearing in it.

For the smoothness term we use the Elastica prior [23] on curves, which models the tangent θ of a contour Γ as a Brownian motion process; this gives:

$$P(\Gamma) = \frac{1}{Z} \exp\left(-\int_{\Gamma} (a\kappa^2(s) + b) ds\right), \quad (3)$$

where $\kappa = \frac{d\theta}{ds}$ is the contour's curvature, and b is a parameter penalizing arbitrarily long contours.

The data-based term aggregates the response of the boundary detector over the candidate curve:

$$P(\Gamma; I) = \exp\left(\int_{\Gamma} \log P_B(s) ds\right) \quad (4)$$

or, after discretization, $\prod_{i \in \Gamma} P_B(i)$, where Γ is now a set of image pixels. The product of these two terms yields:

$$P(\Gamma)P(\Gamma; I) \propto \exp\left(\int_{\Gamma} -[\log P_B(s) + a\kappa^2(s) + b] ds\right) \quad (5)$$

Note that this is not an application of Bayes rule, as (4) is not the image likelihood, i.e. $P(\Gamma; I) \neq P(I|\Gamma)$. From

now on we will only work with the term inside the exponential, and view it as a 'cost' that is low for smooth curves and strong boundary responses. We modify this cost by adding a term $c_T(\Gamma)$ depending on the type $T(\Gamma)$ of the curve; this will be small for closed curves, capturing their higher saliency.

As this criterion increases with the length of the curve, it is biased towards short curves; instead we consider its length-averaged version:

$$C(\Gamma) = \frac{\int_{\Gamma} (E(s) + a\kappa^2(s)) ds + c_T(\Gamma)}{\int 1 ds} = \frac{W(\Gamma)}{L(\Gamma)}, \quad (6)$$

where $E(s) = -\log P_B(s)$. In the fraction on the right $W(\Gamma)$ is the 'weight' of the curve (its cost), and $L(\Gamma)$ its length. We have discarded the $b ds$ term in (5) as after normalization it became a constant.

Normalizing with respect to curve length renders the grouping invariant to changes in object scale which is a desirable property; this idea has therefore been broadly used in the grouping literature, starting from the Minimum Ratio Weight Contours of Jermyn and Ishikawa [13] and subsequently in the works of [29, 9] and more recently [26]. Our main contribution in this respect is a simple and efficient algorithm to optimize this criterion that combines the continuous formulation outlined above with an approximation based on straight line segments, as described below.

3.2. Cost Ratio Optimization

Having defined our objective, (6), we now turn to its optimization. In Sec. 3.2.1 we define a graph-based formulation for grouping, that views salient contours as circles in a



Figure 5. Graph construction as in [17, 30]: Left: A bidirected graph is constructed by separately treating edges with different directions. Right: Connections among conjugate nodes help detect open contours as cycles in a graph.

graph. In Sec. 3.2.2 we express the optimized criterion in terms of straight line segments, and then describe how to efficiently optimize it in Sec. 3.3 using a Fractional-Linear Programming formulation.

3.2.1 Graph Topology

A common formulation of perceptual grouping problems is in terms of minimizing a cost defined over a weighted directed graph; the nodes in this graph are simple image structures (line segments in our case), two nodes are connected if they are geometrically compatible, and the connection weights are set to capture the desired properties of a grouping (we use the term ‘connection’ instead of ‘graph edge’ to avoid confusion with image edges). We use a bidirected graph, as in [17], where each line segment is represented with two nodes -one for each direction; we will be referring to such nodes as *conjugate*.

Using this directed graph, closed image contours amount to graph circles, i.e. paths that begin and end at the same node. Moreover, based on a modification of the graph topology suggested in [30] we can also map open image contours to graph circles by introducing connections between each node and its conjugate; we can thereby trace an open contour while traversing a circle in the graph by using the connections between the conjugate nodes at the endpoints. We refer to [30] for details.

3.2.2 Line-based Graph Weights

Having described the graph topology we now turn to transcribing the criterion of (6) on a graph. As this criterion is expressed in terms of curvilinear integrals we can come up with approximations that are efficiently computable.

For this we replace the integrals in (6) by line-based approximations obtained by breaking up a continuous contour into straight edge segments. We thus perform our subsequent optimization over a reduced set of variables: for example, an image from the Berkeley dataset contains around $13 \cdot 10^4$ pixels, but only a few hundred edge segments.

Piecewise Constant Approximation We break up the domain of the curvilinear integral in (6), into the ‘observed’ subdomain, i.e. along the line segments, and an ‘imagined’ subdomain where we have missing data (failures of

the boundary detector). For the imagined subdomain we replace the integrands with lower bounds described below.

Considering a continuous curve that is broken into K straight edge segments, $L_i, i = 1 \dots, K$, we approximate the numerator and denominator of (6) as:

$$\mathcal{W}(\Gamma) = \int_{\Gamma} E(s) + a\kappa^2(s)ds + c_{T(\Gamma)} \quad (7)$$

$$\simeq \sum_{i=1}^K (E_i + a \cdot 0) + \sum_{i=1}^{K-1} (D_{i,i+1}E_{Th} + aG_{i,i+1}) + c_{T(\Gamma)} \quad (8)$$

$$\mathcal{L}(\Gamma) = \int_{\Gamma} 1ds \simeq \sum_{i=1}^K |L_i| + \sum_{i=1}^{K-1} D_{i,i+1} \quad (9)$$

$$L_i = \int_{C_i} 1ds \quad E_i = \int_{L_i} E(s)ds \quad (10)$$

where $D_{i,i+1}$ is the distance between the end-point of segment i and the beginning of segment $i + 1$, while $E_{Th}, G_{i,i+1}$ will be described below.

In going from (7) to (8) we lower bound the terms showing up in the integration as follows: (i) the squared curvature integral along the line segment is set equal to zero, as we assume the curve is straight. (ii) the length of the contour connecting two segments is considered equal to the Euclidean distance between their endpoints. (iii) The integral of the edge-based cost $E(s)$ along the ‘imagined’ contour connecting two segments is lower bounded by the distance between their endpoints $D_{i,i+1}$ times the cost corresponding the threshold value, $E_{th} = -\log(Th)$ used to obtain the boundary contours. We thus assume that the edge strength at the missing points was just below threshold, thereby lower bounding their actual cost. (iv) The value of the Elastica integral $\int \kappa^2(s)ds$ along the -potentially curved- imaginary contour is replaced with the minimum possible cost $G_{i,i+1}$ of a path connecting the two lines. For this we use the analytical, scale-invariant approximation of [27], which closely approximates the minimum of the Elastica integral, after its normalization with respect to scale.

Graph Construction Based on the approximation described above we now build a graph that allows us to express (6) in terms of path costs and lengths.

The graph nodes are formed from line segments obtained by thresholding the boundary detector outputs at a conservative value, and breaking the formed boundaries into straight lines. Each line is represented by two nodes, one for each possible orientation; the indexes of these nodes will be denoted as i, i' for notational convenience. For each pair of graph nodes i, j we consider that an imaginary arc connects the end-point of node i with the start-point of node j ; in practice we limit ourselves to connections that are closer than 15 pixels. For each pair we compute:

$$w_{i,j} = E_i + D_{i,j}E_{Th} + aG_{i,j}, \quad l_{i,j} = L_i + D_{i,j}.$$

Finally, we rewrite $c_{T(\Gamma)}$ as:

$$c_{T(\Gamma)} = c_{Closed} + [T(\Gamma) = O](c_{Open} - c_{Closed}) \quad (11)$$

where $[T(\Gamma) = O]$ is one when Γ is Open. We break up the last term in two, and introduce it as the weight between conjugate nodes, i.e. $w_{i,i'} = (c_{Open} - c_{Closed})/2$, $l_{i,i'} = 0$. As an open contour uses two conjugate nodes to close a circle, it pays an extra cost $(c_{Open} - c_{Closed})$.

The numerator and denominator of the grouping criterion in (6) for any cyclic path traversing nodes (e_1, \dots, e_C, e_1) then become:

$$\mathcal{W}(\Gamma) = \sum_{i=1}^C w_{e_i, e_{i+1}} + c_{T(\Gamma)}, \quad \mathcal{L}(\Gamma) = \sum_{i=1}^C l_{e_i, e_{i+1}} \quad (12)$$

3.3. A Fractional-Linear Programming Approach

Once the graph has been setup in the manner described above, the optimization of the cost can be phrased as a minimum weight ratio cycle problem, i.e. finding a graph cycle that has the minimum cost, when divided by the length of the path. This is a well-studied combinatorial optimization problem, for which discrete optimization solutions exist, extensively covered in [13]. However these solutions require implementing nontrivial algorithms such as finding zero-weight cycles [13], maintaining priority queues and hashing schemes [9] or approximately finding circular paths with largest area [30], among others. This makes them hard for non-experts and impedes their broader adoption.

Here we propose an approach based on fractional-linear programming, whose main advantage is being substantially simpler to implement and understand: its implementation amounts to setting up a linear program requiring only a few lines of Matlab code, which we will distribute.

In our approach we first introduce a set of variables that indicate whether a connection in the graph is used, we then introduce constraints so that the used connections will form a graph circle, and then optimize the criterion (6) subject to these constraints. Since in (12) we express the numerator and denominator of our cost criterion in terms of edge weights/lengths, we can express our cost criterion in terms of the used connections. Using v_k to indicate whether the k -th graph connection is used, the optimization problem writes:

$$\min \frac{\sum_k v_k w_k + c_C}{\sum_j v_j l_j} \quad (13)$$

$$s.t. \quad v_k \geq 0, \quad v_k \leq 1, \quad \forall k \quad (14)$$

$$\sum_k v_k A_{i,k} = 0, \quad \forall i \quad (15)$$

Open Curves

$$v_k - v_{k'} = 0 \quad (16)$$

$$\sum_{k \in K'} v_k = 2 \quad (17)$$

Closed Curves

$$v_k + v_{k'} \leq 1, \quad (18)$$

$$\sum_{k \in K'} v_k = 0 \quad (19)$$

Instead of indexing connections based on the pair of nodes they connect, we now directly index connections for simplicity. The optimized quantity in (13) is the ratio between the edge cost added up along the utilized connections and the length of the curve formed by concatenating the connected line segments. The inequalities in (14) constrain the solution to lie between 0 and 1, so our problem is a linear relaxation of the initial binary optimization problem.

Constraint (15) involves the Adjacency matrix A of the graph, whose entry $A_{i,k}$ is +1 when connection k departs from node i and -1 when it arrives. Enforcing this for all nodes guarantees that the connections will form a circle.

The open-curve constraint (16) forces an open curve to travel back in the same way that it went from one endpoint to another. For this, it assures that for each used connection k its conjugate connection k' will be used, too.

The closed curve constraint (18) allows the use of only one of the conjugate connections. The set K' appearing in the constraints of (17) and (19) consists of the connections among conjugate nodes, i.e. points where a curve turns around. Thereby, (17) constrains each open curve to have two such nodes, one in its middle and another in the end, while (19) prohibits the use of such nodes for closed curves.

As detailed in [4], an optimization problem of this form can be converted into a linear program. In specific, for $y = \frac{x}{e^T x + f}$ and $z = \frac{1}{e^T x + f}$, the following two optimization problems are equivalent:

$$\min \frac{c^T x + d}{e^T x + f}$$

$$Gx < 0$$

$$Ax = b$$

$$\min \quad c^T y + dz$$

$$Gy - hz < 0, \quad z > 0$$

$$Ay - bz = 0, \quad e^T y + fz = 1$$

Having built the matrices for the original fractional programming problem, solving the equivalent problem based on a sparse LP library (LPSOLVE) typically takes a fraction of a second. Examples of groupings recovered in this manner are shown in Fig. 6.

We note that the algorithms of [13, 26] use binary/linear search for the optimal value of λ , and solve a combinatorial problem for each λ ; instead our fractional-linear programming solution furnishes the optimal λ and the corresponding solution in a single shot.

Implementation Details We detect contours greedily, by iteratively solving the fractional-linear program above, and for every edge that participates in a grouping all of its connection weights are set to infinity at the next iteration.

Fractional solutions can occur when the flow along a patch splits into two halves at a certain edge and merges

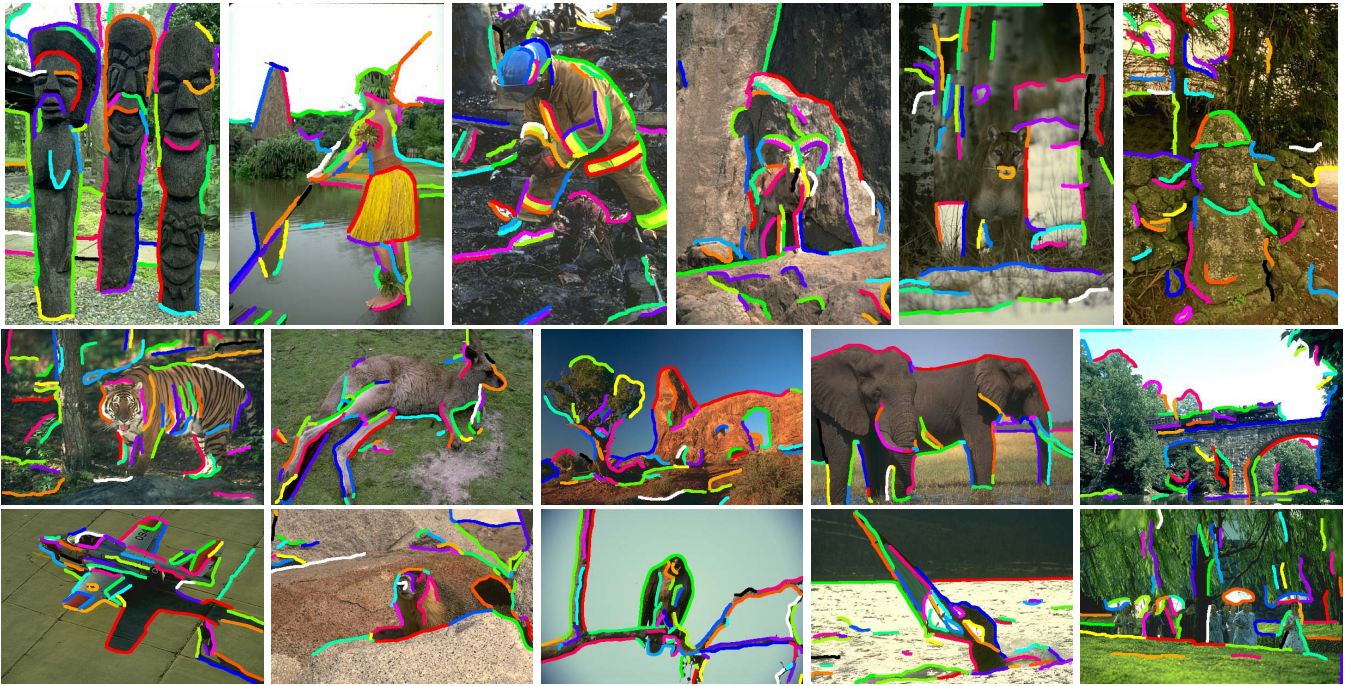


Figure 6. The first (strongest) 50 groupings found by our algorithm. Please see in color on screen.

later on at a subsequent edge. Whenever this happens, we temporarily remove all connections including such edges and solve the optimization problem again. As this can be done rapidly, the additional computational cost is negligible. On average, the optimization takes a fraction of a second per contour, and approximately 20 seconds for an image containing 200-300 hundred contours.

4. Benchmarking Results

We systematically evaluate our approach on the Berkeley Benchmark. The first experiment, shown in Fig. 7 explores the complementarity of the descriptor-based information with the Berkeley edge detector’s output. We train (i) a classifier using the 100-dimensional feature vector obtained by SAVE, (ii) a classifier using the output of the Berkeley Pb detector computed at 4 scales increasing by $\sqrt{2}/2$, and (iii) a classifier that uses both sources of information. From the improved performance of the classifier it is evident that these cues are complementary.

We then compare our classifier that uses both cues to the current state-of-the-art; we observe that our approach has better performance, particularly in the high-recall regime, which is most useful for object recognition.

A method that performs better in the high-precision regime is the gPB detector of [19] (and therefore also [3]; this is expected, as these detectors use global information computed from segmentation. However, when combined with grouping our method yields comparable results also in the high-precision regime, while we expect that combining our method with the ‘spectral gradient’ features used in [19]

can yield a complementary boost in performance.

To assess the merits of grouping, we first explore the gain obtained when compared to the plain, pixel-level boundary detection algorithm. For this at each edge we replace the probability estimate provided by Adaboost with $1/(1 + \exp(-C))$, where C is the normalized cost of the contour with which the edgel was grouped. This was also used to generate the bottom row in Fig. 4.

As is shown in Fig. 7 using grouping has the same impact as that observed also elsewhere in the literature [30, 9], i.e. a boost in the high-precision regime, that may not be however accompanied by an improvement in the high-recall regime. This is intuitive since by grouping we discard small isolated segments but cannot recover pixels that were initially missed by detection; therefore recall does not increase, or can even decrease if small boundaries do not become part of a grouping. Comparing to other state-of-the-art grouping algorithms we observe that (a) we outperform [9], apparently due to the better quality of our boundary detector (otherwise the criteria we are optimizing are similar) (b) our algorithm has comparable performance to that of Zhu and Shi [30] in the high precision regime, but has substantially better performance in the high recall regime.

In the last plot we compare our results to the ones of works that are closest to us, namely the multi-scale approach of Ren [24] and the boosting-based method of Dollar et. al. [7]. Comparing to [7], we attribute the improved performance to our features; the PDT classifier used in [7] is extremely efficient and we would expect it to outperform our simpler boosting algorithm, if combined with our features. But the authors in [7] use thousands of Ga-

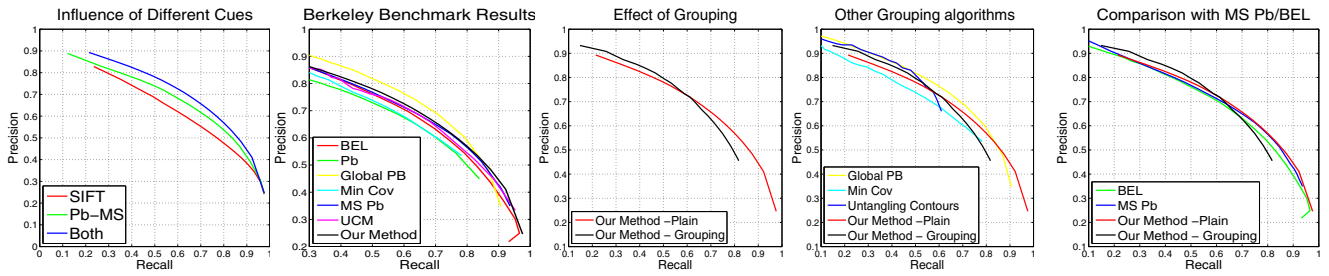


Figure 7. Benchmarking results. Left: Detection. Middle: Grouping. Right: comparison with the most similar works of [24, 7].

bor, Haar, DoG and Edge features, and leave Adaboost to decide which are useful. Instead we use a 100-dimensional feature set by extracting the same descriptor at three scales and three color channels and then reducing its dimensionality. This both performs better and is more efficient.

5. Conclusions

In the first part we introduced a highly accurate learning-based approach to boundary detection, that utilizes appearance descriptors in conjunction with boosting. We then developed a simple perceptual grouping algorithm that relies on fractional-linear programming. We have evaluated the merit of our work on a standard benchmark, demonstrating that each of our contributions can result in a systematic improvement in performance. In future research we intend to exploit the contours delivered by our method for object recognition, as well as to extend the learning method described here to other features apart from boundaries.

References

- [1] A. Abdel-Hakim and A. Faraq. Csfift: A sift descriptor with color invariant characteristics. In *CVPR*, 2006.
- [2] P. Arbelaez. Boundary Extraction in Natural Images Using Ultrametric Contour Maps. In *WPOCV*, 2006.
- [3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009.
- [4] S. Boyd and R. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [5] B. Catanzaro, B. S. amd N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *ICCV*, 2009.
- [6] D. Cook and H. Lee. Dimension reduction in binary response regression. *JASA*, 94, 1999.
- [7] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries. In *CVPR*, 2006.
- [8] C. Domingo and O. Watanabe. Madaboost: A modification of adaboost. In *Proc. COLT*, 2000.
- [9] P. Felzenszwalb and D. McAllester. A Min-Cover Approach for Finding Salient Curves. In *WPOCV*, 2006.
- [10] Y. Freund and R. Schapire. Experiments with a new Boosting Algorithm. In *ICML*, pages 148–156, 1996.
- [11] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Stat.*, 2000.
- [12] G. Hua, M. Brown, and S. Winder. Discriminant Embedding for Local Image Descriptors. In *ICCV*, 2007.
- [13] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *PAMI*, 2001.
- [14] Y. Ke and R. Sukthankar. PCA-SIFT: A Distinctive Representation for Local Image Descriptors. In *CVPR*, 2004.
- [15] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical Edge Detection: Learning and Evaluating Edge Cues. *PAMI*, 25(1), 2003.
- [16] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2), 2004.
- [17] S. Mahamud, L. Williams, K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *PAMI*, 25:433444, 2003.
- [18] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008.
- [19] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using Contours to Detect and Localize Junctions in Natural Images. In *CVPR*, 2008.
- [20] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *PAMI*, 26(5):530–549, 2004.
- [21] R. Meir and G. Ratsch. An introduction to boosting and leveraging. In *Adv. Lec. Mach. Learning*, 2003.
- [22] K. Mikolajczyk and J. Matas. Improving descriptors for tree matching by optimal linear projection. In *ICCV*, 2007.
- [23] D. Mumford. *Elastica and Computer Vision*. In *Algebraic Geometry and its applications*. 1993.
- [24] X. Ren. Multiscale helps boundary detection. In *ECCV*, 2008.
- [25] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using crfs. In *ICCV*, 2005.
- [26] T. Schoenemann, S. Masnou, and D. Cremers. The Elastic Ratio: Introducing Curvature into Ratio-based Globally Optimal Image Segmentation. *PAMI*, 2010.
- [27] E. Sharon, A. Brandt, and R. Basri. Completion energies and scale. *PAMI*, 22, 2000.
- [28] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>, 2008.
- [29] S. Wang, T. Kubota, J. M. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *PAMI*, 2005.
- [30] Q. Zhu, G. Song, and J. Shi. Untangling cycles for contour grouping. In *ICCV*, 2007.