Linked Edges as Stable Region Boundaries[*]

Michael Donoser, Hayko Riemenschneider and Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology, Austria
(donoser, hayko, bischof)@icg.tugraz.at

Abstract

Many of the recently popular shape based category recognition methods require stable, connected and labeled edges as input. This paper introduces a novel method to find the most stable region boundaries in grayscale images for this purpose. In contrast to common edge detection algorithms as Canny, which only analyze local discontinuities in image brightness, our method integrates mid-level information by analyzing regions that support the local gradient magnitudes. We use a component tree where every node contains a single connected region obtained from thresholding the gradient magnitude image. Edges in the tree are defined by an inclusion relationship between nested regions in different levels of the tree. Region boundaries which are similar in shape (i. e. have a low chamfer distance) across several levels of the tree are included in the final result. Since the component tree can be calculated in quasi-linear time and chamfer matching between nodes in the component tree is reduced to analysis of the distance transformation, results are obtained in an efficient manner. The proposed detection algorithm labels all identified edges during calculation, thus avoiding the cumbersome post-processing of connecting and labeling edge responses. We evaluate our method on two reference data sets and demonstrate improved performance for shape prototype based localization of objects in images.

1. Introduction

Object category recognition is a challenging and well investigated topic. Despite the tremendous progress in this field it is still not clear what the main features are that help us to group objects into categories. Recently, shape based recognition frameworks became quite popular, since for a wide variety of object categories shape matters more than local appearance. Shape is in general a quite powerful feature since an object contour is invariant to extreme lighting conditions and large variations in texture or color and additionally allows delineating accurate object outlines.

Different authors like [11, 6, 17, 18] achieved state-of-the-art results outperforming appearance based approaches for several object categories. All these methods implicitly require stable, connected and labeled edges in the image as underlying representation, which has the main purpose of reducing the data amount while retaining the important information about the image content. Most approaches rely on a post-processed Canny [3] or Berkeley edge detection [12] result, which are considered as state of the art in this field. Post-processing is necessary because most detection frameworks require labeled lists of connected edges in images, which are obtained by analyzing T-junctions, by building multiple edge branches or even by forming complex contour groups [7].

In general, edge detection is one of the most intensively studied problems in computer vision and has many other applications like stereo matching, segmentation, 3D reconstruction or tracking. Methods can be divided into approaches which simply analyze local intensity differences and the recently popular group of learning based methods, which focus on learning the appearance of edges, e. g. for specific tasks [5] or in a more general way for natural images [10, 12]. Of course, learning based edge detection methods provide improved results making use of training images. However, these methods have the main drawback of high computation time, for example in the range of 12 seconds [5] and 90 seconds [20]. Even a highly optimized implementation for GPUs using 30 cores still requires about 2 seconds per image [4]. Furthermore, they require ground truth to learn application specific detectors. Therefore, local edge detectors are still of high interest because of their computational simplicity.

In this paper we introduce a novel edge detection method, designed for the application of object detection, which extends purely local edge detectors by additionally analyzing mid-level cues, i. e. regions that support the lo-

[*]This work was supported by the Austrian Research Promotion Agency (FFG) project FIT-IT CityFit (815971/14472-GLE/ROD) and the Austrian Science Fund (FWF) under the doctoral program Confluence of Vision and Graphics W1209.

cal gradient magnitude are analyzed to extract the most stable edges. Detection is based on analyzing a hierarchical data structure denoted as component tree where connected regions, which are separated by high gradient magnitudes along their boundaries, are linked in a tree structure. Finding the most stable edges in the tree, by considering shape similarity of the region contours, removes noise and clutter and preserves the important edges for detection. Furthermore, our method automatically labels all obtained edges during calculation. Therefore, no further post-processing is required and the results can be directly passed to any of the available shape based object localization methods.

The outline of the paper is as follows. Section 2 gives an overview on related work concerning edge detectors and outlines similarities of our method to recent state of the art. Section 3 describes our novel edge detection method in detail. In Section 4 we demonstrate on two well known object recognition data sets that the proposed method reduces clutter and maintains the most important edge responses. We furthermore outline a potential application in the area of object localization by using our edge responses in an oriented chamfer matching step. Finally, Section 5 discusses our results and gives an outlook on future work in this area.

2. Related work

The most commonly used edge detection method is Canny [3] because of its simplicity, effectiveness and high efficiency. Canny detects edges in the image by several consecutive steps. First, Gaussian blurring is applied to reduce the noise of individual pixels. Second, 2D Gaussian derivative responses are calculated to detect regions of sharp intensity changes. The magnitude and the orientation of the gradient at each pixel are calculated. The gradient image undergoes a non-maxima suppression to filter out weaker responses. Finally, a connected component analysis based on a hysteresis thresholding heuristic returns the set of final edges. The main problem of Canny is its sensitivity to the manually chosen thresholds.

The Berkeley edge detector [12] is a supervised algorithm usually tuned to natural scenes and is considered as state of the art in this field. It is based on analyzing several image features like raw and oriented energy and texture gradients. These image features are then combined to improve edge detection of natural images. For feature combination, supervised learning is used which exploits a ground truth data set built from multiple human segmentations, where humans were asked to draw boundaries of equally important things in the training images. This ground truth is used in a logistic regression model to learn an optimal combination of the image features and to tune the detector to natural images.

Our method exploits a hierarchical data structure denoted as component tree to obtain stable edge detection results. Building and analyzing such a hierarchy of nested regions for edge detection is not a novel idea. The two most related methods were proposed by Najman and Schmitt [15] and Arbalaez [1].

In [15] the dynamics of watershed contours were used to define edge maps. The method is based on identifying watersheds by a standard flooding process. Starting from local minima in the gradient magnitudes, the image is flooded and when flooded regions merge, the watershed height defines the saliency of the local contour. In contrast to our approach, edge saliency is defined as the minimal contrast between two merged regions along their common boundary and stability analysis of the obtained edges is neglected.

Our detector is in a similar spirit to the boundary extraction method proposed by Arbalaez [1]. Arbalaez also builds a hierarchical data structure using a stratification index and outlines its equivalence to an ultra-metric distance between regions. The data structure is denoted as ultrametric contour map, where each level provides a segmentation of the image at various levels of detail. They exploit the concept of strong causality to provide a saliency value for each segment contour which allows to additionally return boundary extraction results. Contrary to our approach, color information is exploited in terms of a contrast function and a color uniformity measure per segment. Furthermore an image pre-segmentation by means of a Voronoi tessellation is required which increases computational complexity.

3. Component tree edge detection

Our novel edge detection and labeling method is based on three consecutive steps. First, similar to most edge detection algorithms we apply some basic pre-processing steps which are outlined in Section 3.1. The output of this first step is a normalized gradient magnitude image. The second step is to build a hierarchical data structure named component tree which is a unique representation of the obtained gradient magnitude image. The definition of the component tree and its efficient calculation is presented in Section 3.2. Finally, for detecting the most stable edges, we analyze the component tree by comparing the shape of regions at different levels using a simple partial shape matching method. In such a way we are able to measure the boundary stability of every region and to return the most stable ones in an automatically labeled manner as our final edge detection result which is explained in Section 3.3.

3.1. Pre-processing

As a first step we obtain a gradient magnitude image which is used as input to the component tree analysis outlined in Section 3.2. Please note, that we can use any gradient map for our stable edge detection method as e. g. the gradient responses obtained by the Berkeley detector [12]. But due to its simplicity and much lower computational complexity we stick to simple Gaussian derivatives.

In general image derivatives emphasize high frequencies and thus amplify noise. Therefore, it is required to smooth the image with a low-pass filter prior to gradient calculation using a circularly symmetric filter, since edge responses should be independent of orientation. We use the same sequence of pre-processing steps as Canny. We first convolve the image with a 2D Gaussian to remove noise and then apply a first order 2D Gaussian derivative filter. Since Gaussian filtering is separable we can use two 1D convolutions in order to achieve the effect of convolving with a 2D Gaussian. As a result we obtain gradient magnitudes and orientations per pixel. We neglect orientation information in the following steps and simply use the obtained gradient magnitude map I. The component tree as described in the next section requires pixel values coming from a totally ordered set as input. Therefore, we further have to normalize the magnitudes and to scale them to an integer range.

3.2. Component tree

For detection and labeling of stable edges in an image we build a unique data structure denoted as component tree for the obtained gradient magnitude image. The component tree was originally introduced in statistics for classification and clustering and was redefined by Jones [9] for the field of image analysis as a *"representation of a gray-level image that contains information about each image component and the links that exist between components at sequential gray-levels in the image"*.

The component tree can be built for any vertex-weighted graph defined by a triplet $G = (V, E, I)$. V is a finite set of vertices, i. e. the set of pixels from our image with V being a subset of $\mathbb{Z}^2$. Therefore a vertex $x \in V$ in our case is defined by its two coordinates (x_1, x_2). E is the set of edges defining the neighborhood relation of the graph, where Γ is a map from a $x \in V$ to a subset of V, such that for every pixel $x \in V$, $\Gamma(x) = \{y \in V \mid (x, y) \in E\}$. If a pixel $y \in \Gamma(x)$, we say that y is a neighbor of x. In our image setting, the neighborhood relation is defined by the standard 4 pixel neighborhood as $E = \{(x, y) \in V \times V, |x_1 - y_1| + |x_2 - y_2| = 1\}$. I is a map from V to D, where D is any finite set allowing to order all points, *e.g.* a finite subset of integers. In our case, $D = \{0, 1, \ldots, 255\}$ and the mapping I is defined by the normalized gradient magnitudes that are scaled up to an unsigned 8-bit integer range. The magnitude values I are inverted so that a low value represents a high gradient value.

A cross section of the vertex weighted graph at a level t is defined as $I_t = \{x \in V \mid I(x) \geq t\}$. Each possible cross section I_t is a standard non-weighted graph defined by a tuple $G_t = (V, E)$, where E is a binary relation on V being anti-reflexive $(x, x) \notin E$ and symmetric $(x, y) \in E \Leftrightarrow (y, x) \in E$. E for a cross section I_t connects neighboring pixels (x, y) only if $I(x) \geq t$ and $I(y) \geq t$.

We further define linkage between two vertices x and y, if there is a path $P = (p_0, p_1, \ldots, p_N)$ with $p_i \in V$ between x and y so that for every pixel along the path $(p_i, p_{i+1}) \in E$ is valid. We denote a subset $X \subseteq V$ as connected, if any $x, y \in X$ are linked. A connected component C is defined as a maximal subset $C \subseteq V$, where it is not possible to add another pixel $z \in V$ which is linked to any $x \in C$ of the component.

These definitions now allow to define the component tree. We first find the minimum level $t_{min} = min\{I(x) \mid x \in V\}$ and the maximum level $t_{max} = max\{I(x) \mid x \in V\}$. We then calculate all cross sections I_t of the vertex-weighted graph in the range of t_{min} to t_{max} and find in each cross section I_t the connected components C^t as defined above to build the component tree structure from the top to the bottom. The obtained connected components C^t can be organized in a hierarchical data structure denoted as component tree because they are nested and their inclusion relationship is defined by $\forall x \in C^t, \exists y \in C^{t+1} : x = y$. Therefore, region size can only decrease from the top to the bottom of the component tree and the root node contains all components merged into a complete representation of the input image. Please note, that in contrast to the hierarchical structure defined in [1] the localization of the contours is not preserved through different levels of the tree, because we start from a single pixel instead of a super pixel representation.

The component tree can be implemented in many efficient ways. We use the method proposed by Najman and Couprie [14] since it has the lowest computational complexity. Their algorithm runs in quasi-linear time, where the term *quasi* is an amortization of the costs required for the union-find problem. Thus, the worst-case complexity is $O(N \times \alpha(N))$ with $\alpha(N)$ being the inverse Ackermann function, i. e. practically linear. Please note, that the component tree is also the basis for the calculation of Maximally Stable Extremal Regions (MSERs) [13]. Recently Nistér and Stewénius [16] even showed that calculating MSERs is possible in linear time returning an ellipse instead of a pixel representation for each region.

To sum up, the component tree is a unique representation encoding the gradient magnitude structure of the input image as it is illustrated in Figure 1. As can be seen regions within the cross sections of the component tree correspond to connected areas in the image separated by high gradient magnitude values along their boundaries, related to the concept of watershed segmentation. Every node of the tree contains data about the corresponding outer region boundaries, the tree level and the hierarchical relations to other component tree nodes. The shape of some regions stays the same over several levels of the tree, which is the criterion

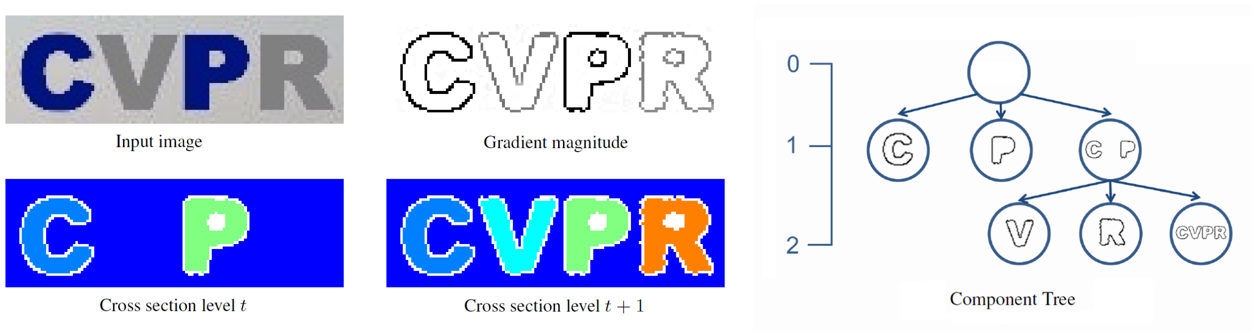

Figure 1. Illustration of component tree representation. First row on the left shows the input image and the obtained gradient magnitude response map (dark values equal high gradient magnitudes). Second row shows label results after thresholding the magnitude image at two different levels. Regions correspond to connected areas separated by high gradient magnitude values along their boundaries. Cross sections are related by inclusion relationship from left to right, i. e. regions can only become smaller. This region nesting defines the hierarchical data structure denoted as component tree shown on the right. Outer boundaries of these regions are analyzed concerning shape similarity over several levels of the tree to detect stable edges.

that we use to detect stable edges as it is outlined in the next section. Furthermore, since we only want to detect stable edges that are supported by a large adjacent region, we remove all regions below a fixed threshold Ω from the tree.

3.3. Stable component tree region boundaries

We now use the component tree structure, built as described in Section 3.2, for detecting and immediately labeling edges in the input image. The underlying idea is quite similar to the concept of Maximally Stable Extremal Region (MSER) detection as proposed by Matas *et al.* [13]. MSER detection builds the component tree directly on the intensities of the image and returns the most stable nodes (extremal regions) of the tree as interest region detection results. The stability is estimated by comparing region sizes between nodes in adjacent levels of the tree.

We follow the same principle but focus on analysis of the boundary stability between the regions. In contrast to MSER detection which compares region sizes, we measure how similar the shape of the regions is, i. e. if parts of the region boundary remain more or less the same over several levels of the component tree.

We define the stability value $\Psi(C_i^t)$ of a connected component C_i^t at a component tree cross section I_t at level t by comparing its shape to a region $C_j^{t-\Delta}$, which is the connected component linked to C_i^t by moving along the hierarchical structure up to a level $t-\Delta$ (see Figure 1), where Δ is a stability parameter of the method. Increasing the value Δ yields stricter stability constraints, i. e. region boundaries have to be stable over more levels of the tree, which leads to a reduced number of detected edges.

To measure the shape similarity between C_i^t and $C_j^{t-\Delta}$ and to identify similar contour parts between the regions, we apply a simplified version of chamfer matching. Since regions within the tree can only grow from level t to level $t-\Delta$ and always are fixed at the same location, chamfer matching is reduced to an analysis of the distance transformation values. We apply the basic approach for chamfer matching, but please note that more sophisticated methods as e. g. chamfer matching variations additionally considering orientation or in general any partial shape matching method can be used in this step.

Let $\mathcal{B}_i$ and $\mathcal{B}_j$ be the sequence of outer boundary pixels for the current region C_i^t and its linked neighbor $C_j^{t-\Delta}$. As a first step we calculate the distance transformation DT_i on the binary image solely containing the boundary pixels of $\mathcal{B}_i$, which assigns to each pixel the minimum distance to a boundary point. This distance transformation DT_i enables to find partial matches between the boundaries $\mathcal{C}_i$ and $\mathcal{C}_j$ by finding connected boundary fragments $\overline{\mathcal{C}}_j \subseteq \mathcal{C}_j$ fulfilling

$$\overline{\mathcal{C}}_j \subseteq \mathcal{C}_j \rightarrow \forall x \in \overline{\mathcal{C}}_j \; : \; DT_i\left(x\right) \leq \Phi, \tag{1}$$

where Φ is a maximum boundary distance parameter. Finally, for the region C_i^t we set the corresponding stability value $\Psi(C_i^t)$ to the average chamfer distance of the matched boundary pixels $\overline{\mathcal{C}}_j$ by

$$\Psi(C_i^t) \;= \frac{1}{N}\sum_{n=1}^{N} DT_i\left(x_n\right) \;\; where \;\; x_n \in \overline{\mathcal{C}}_j \tag{2}$$

and N is the number of matched pixel. In such a way we get a stability score $\Psi(C_i^t)$ and matched connected boundary fragments for every region in an efficient manner, since we only have to look up corresponding distance transformation values.

After calculating the stability values $\Psi(C_i^t)$ for every node in the component tree, we detect the most stable ones similar as in MSER detection as described in [13] and return the corresponding matched boundary fragments as detection result. We further assign the level in the component tree each edge was detected as saliency value, which measures the importance of the edge, since detection at a level t means that all gradient magnitude values of the edge have to be bigger or equal than t. Furthermore, since every detected edge is linked to a specific node in the component tree, a unique edge ID can be assigned during component tree analysis. Therefore, cumbersome post-processing to connect and clean edges is not required and the final output of our method is a labeled set of connected edges.

Please note, that edge detection results are quite different from simply returning the boundaries of an MSER detection result. First of all we use the gradient magnitude image instead of the intensity image as underlying representation. Second, we have a different stability criterion analyzing the stability of the shape of the region contours instead of region size stability. Finally, since we identify parts of the region contours that are similar, the returned edges need not be closed.

4. Experiments

The focus of experimental evaluation lies on demonstrating the reduced noise and high retrieval of valuable stable edges. In Section 4.1 we use the Berkeley benchmark for experimental evaluation on two well known object recognition data sets. Since our proposed method returns well-connected and stable edges, it is also well suited for shape prototype matching methods, where a binary prototype is shifted all over the image and similarities to the provided prototype are calculated. Therefore, in Section 4.2 we use our edges in an oriented chamfer matching method and illustrate results in comparison to Canny and Berkeley edge detection methods.

Our method mainly has four parameters that are fixed for all experiments. The minimum considered region size Ω is 400. The stability parameter Δ is 5 and the shape similarity parameter Φ is 10. Furthermore, after obtaining the result we remove all edges with a length below 70 pixels, which easily can be done because each edge can be directly accessed by its unique ID.

4.1. Berkeley evaluation framework

As a first experiment we employ the Berkeley segmentation benchmark, which allows to measure the quality of edge maps compared to human ground truth data. We use two well known object recognition data sets: the ETHZ shape classes [7] and the Weizmann horses [2]. Ground truth for both data sets is a set of figure/ground segmentations. For the ETHZ data set, we created the figure/ground segmentations whereas for the Weizmann horses they are provided. In both cases the segmentations highlight the shape of the objects that should be detected in the test image. For evaluation of the results respective to the defined ground truth data, different edge detection results are compared to the ground truth in the same manner as in the Berkeley segmentation benchmark.

We use precision and recall to measure the quality of the obtained edge responses. Precision in our case is the probability that a detected boundary pixel is a true boundary pixel. Recall is the probability that a true boundary pixel is detected by the algorithm. We calculate the F-measure as a final comparison value defined as weighted harmonic mean of precision and recall. The Berkeley benchmark provides exactly these precision and recall curves for each threshold of the image. Since object recognition systems often require binary edge detection results as input (e. g. later weighted with the average gradient magnitude), we evaluate the pure binary responses. We select all edges with a gradient magnitude above zero and retrieve a single F-measure per test image. By analyzing binary responses we can focus on the actual edges returned by an edge detector, rather than on the noisy, unimportant responses.

Table 1 summarizes results of this experiment, outlining the calculated precision, recall and F-measures. We compare results for the ETHZ shape classes and the Weizmann horses obtained by the Canny edge detector using default parameters, by the Berkeley edge detector and by our proposed method. The improvements achieved by our stable edge detector are encouraging for both data sets, yielding an average F-measure improvement over all analyzed classes of 18% compared to standard Canny and of 4% compared to the learned Berkeley detector. We are thus able to match the quality of the detection results of a supervised method and the speed of a standard Canny method.

The improvements achieved by our edge detector in these experiments demonstrate how edges become stable when they exhibit region support. The applelogo, bottle, swan and horse classes all contain strong region support for the boundaries. On the other hand, giraffes due to their strong texture deliver less stable region support, which in turn results in a lower recall for our obtained edges.

The advantages of our algorithm are clear when looking at Figure 2, which shows some examples of our stable edge responses for the ETHZ classes. Our edge detector produces far less noise since only stable edges are returned which have a strong support from adjacent regions. This removes a lot of the cluttered edges which are present in other edge responses. Occasionally some edges are missing, which e. g. are present in the highly dense responses of Canny. This is reflected in the lower recall of our method, nevertheless our precision is consistently higher. In general, the precision values are very low since our evaluation strategy is very strict. Only boundaries of the respective object class are marked as true boundaries, and thus any other edge response is seen as a false positive. This is less severe in the case of the Weizmann horses, since there horses

Class	Canny			Berkeley			Our detector		
	P	R	F	P	R	F	P	R	F
ETHZ applelogos	0.02	0.99	0.05	0.08	0.95	0.15	0.12	0.90	**0.21**
ETHZ bottles	0.06	0.99	0.11	0.16	0.95	0.28	0.17	0.84	**0.29**
ETHZ giraffes	0.10	0.99	0.10	0.20	0.90	**0.32**	0.16	0.69	0.26
ETHZ mugs	0.08	0.98	0.15	0.19	0.94	**0.32**	0.18	0.86	0.30
ETHZ swans	0.05	0.98	0.10	0.15	0.94	0.27	0.24	0.82	**0.38**
Weizmann horses	0.14	0.94	0.25	0.18	0.94	0.30	0.33	0.53	**0.41**
Class Average	0.08	0.98	0.13	0.16	0.94	0.27	0.20	0.77	**0.31**

Table 1. Comparison of the overall precision, recall and F-measure performance of each algorithm for the two data sets ETHZ shape classes [7] and Weizmann horses [2]. Please note, that the benchmark is very strict given that only boundaries of objects-of-interest are marked in the human ground truth segmentation (resulting in a low precision value). Our edge detector provides an encouraging improvement over the Canny (18%) and the learned Berkeley edge responses (4%).

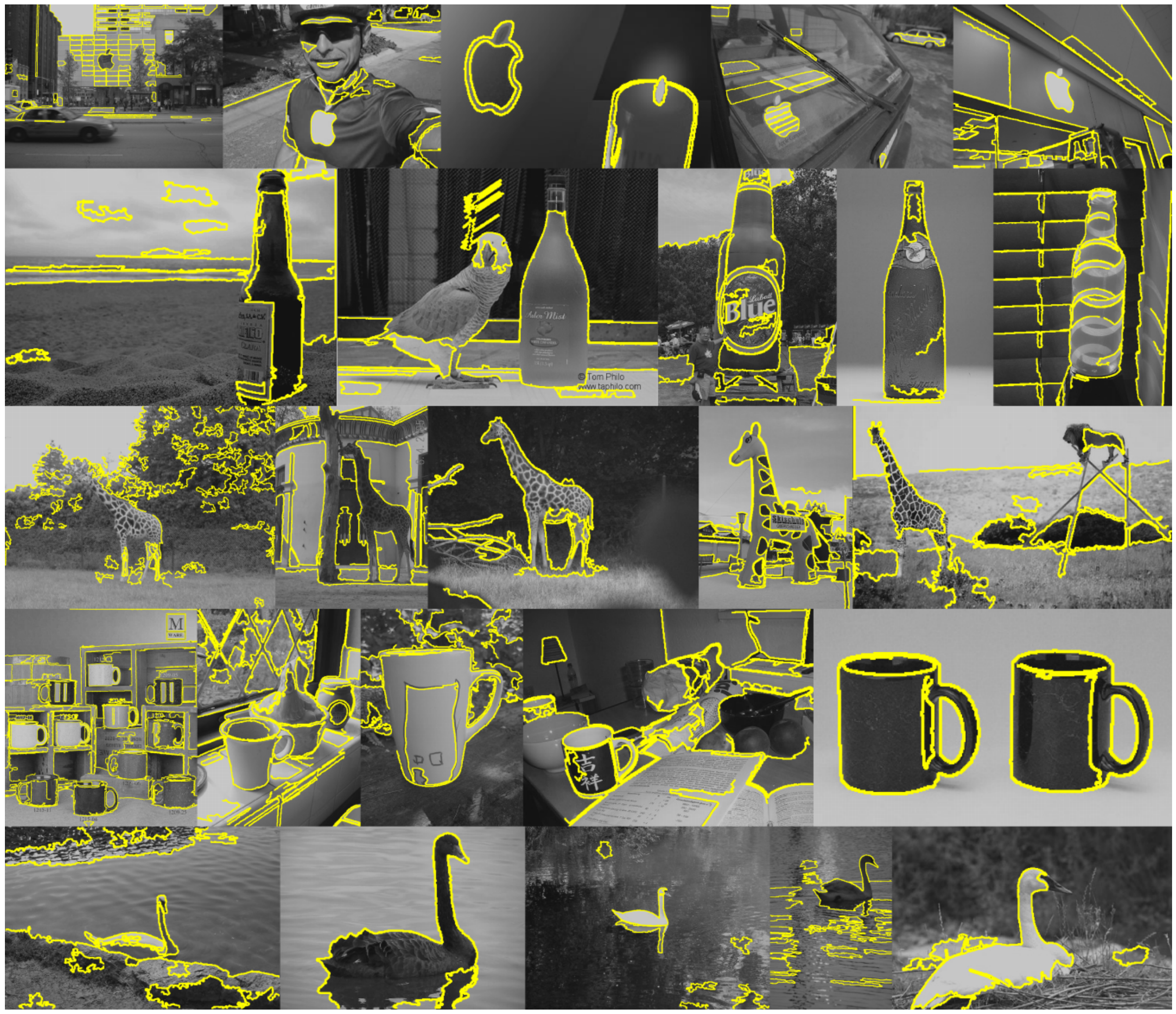

Figure 2. Edge detection result of our proposed stable region boundary extraction method on images of ETHZ object detection data set. Edge responses for all five classes are shown.

are the only dominant object in the images. In comparison, the images of the ETHZ data set contain much more background information including strong edges. The second advantage of our method is the implicit labeling returned by our approach. In contrast to the edge responses from Canny or Berkeley, our edges are connected and uniquely labeled. This provides another benefit, since no post-processing is required to dissect edges from the cluttered responses, to split them at T-junctions and to finally link them into lists. Our edge detector directly provides these results, which would otherwise require costly and sometimes destructive post-processing.

4.2. Oriented chamfer matching

Some of the state-of-the-art recognition methods [19, 8, 18] implicitly rely on local shape prototype matching scores to localize objects in an image. The most prominent method in this area is chamfer matching, which has several extensions improving results, for example by additionally considering orientation. Chamfer matching strongly relies on well connected and undisturbed edges, where clutter has a strong adverse effect on the results. Therefore, improving chamfer matching performance is one potential application area for our novel edge detector.

We evaluate the detection performance on several image types for detecting categories like humans, cars or windows. The corresponding shape prototypes are manually drawn or chosen as mean shape when ground truth was available. As matching method we use oriented chamfer matching [18]. For comparison, we additionally evaluated on Canny and Berkeley edge responses. Figure 3 illustrates detected edges and voting maps returned by oriented chamfer matching for the three methods using a car prototype. As can be seen our detector returns far less noise while retaining the most important edges for localization, which is illustrated by the much cleaner vote maps. Figure 4 shows further chamfer matching results for localizing humans and windows. Again, the significantly reduced clutter and noise in our edge detection results yields much cleaner vote maps.

5. Conclusion and outlook

This paper introduced a novel edge detection algorithm for the purpose of shape based object localization. It is based on the idea of analyzing a hierarchical data structure denoted as component tree containing connected regions which are separated by high gradient values at different magnitudes. We detect the most stable edges within the component tree, which are returned in a labeled manner preventing cumbersome post-processing required by other methods. The benefit of our detector is the integration of context information in terms of support by connected regions. Experimental evaluations using the Berkeley segmentation benchmark on two well known object recognition data sets demonstrated the applicability of our method. It accurately removes noise and clutter and delivers cleaner, more precise and still rich edge responses. Future work will focus on evaluating our detector in different object localization frameworks and on investigating the benefit when integrating color information.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Proceedings of Conference on Computer Vision (CVPR)*, 2006.

[2] E. Borenstein and S. Ullman. Learning to segment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3, pages 315–328, 2004.

[3] J. Canny. A computational approach to edge detection. *IEEE Transations Pattern Analysis Machine Intelligence (PAMI)*, 8(6):679–698, 1986.

[4] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2009.

[5] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971, 2006.

[6] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transations Pattern Analysis Machine Intelligence (PAMI)*, 30(1):36–51, 2008.

[7] V. Ferrari, T. Tuytelaars, and L. V. Gool. Object detection by contour segment networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3, pages 14–28, 2006.

[8] D. Gavrila. Pedestrian detection from a moving vehicle. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2000.

[9] R. Jones. Connected filtering and segmentation using component trees. *Journal of Computer Vision and Image Understanding (CVIU)*, 75(3):215–228, 1999.

[10] S. Konishi, A. Yuille, J. Coughlan, and S. Zhu. Statistical edge detection: learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25:57–74, 2003.

[11] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[12] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(5):530–549, 2004.

[13] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 1, pages 384–393, 2002.

[14] L. Najman and M. Couprie. Quasi-linear algorithm for the component tree. In L. Latecki, D. Mount, and A. Wu, editors, *IS&T/SPIE Symposium on Electronic Imaging, Vision Geometry XII*, volume 5300, pages 98–107, 2004.

[15] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18, 1996.

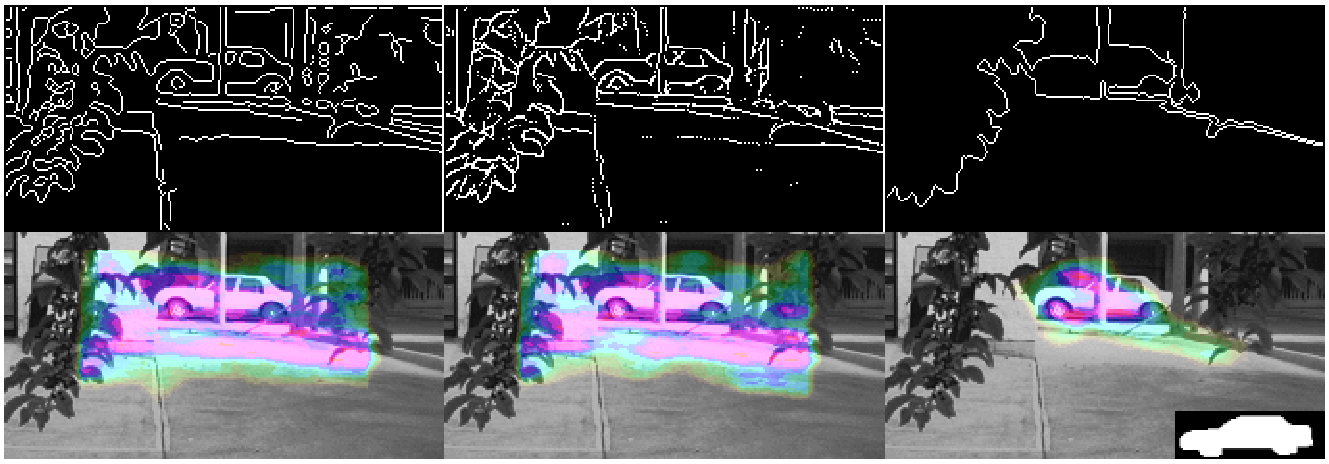

Figure 3. First row shows edge detection results from Canny, Berkeley and our proposed method. Second row shows corresponding vote maps when oriented chamfer matching is applied on edge responses using a car shape prototype (shown on the lower, right side of the figure). As can be seen our approach yields much cleaner vote maps. Please note, that due to the natural image specific training of the Berkeley edge detector, it does not perform well for such image types. Best viewed in color.

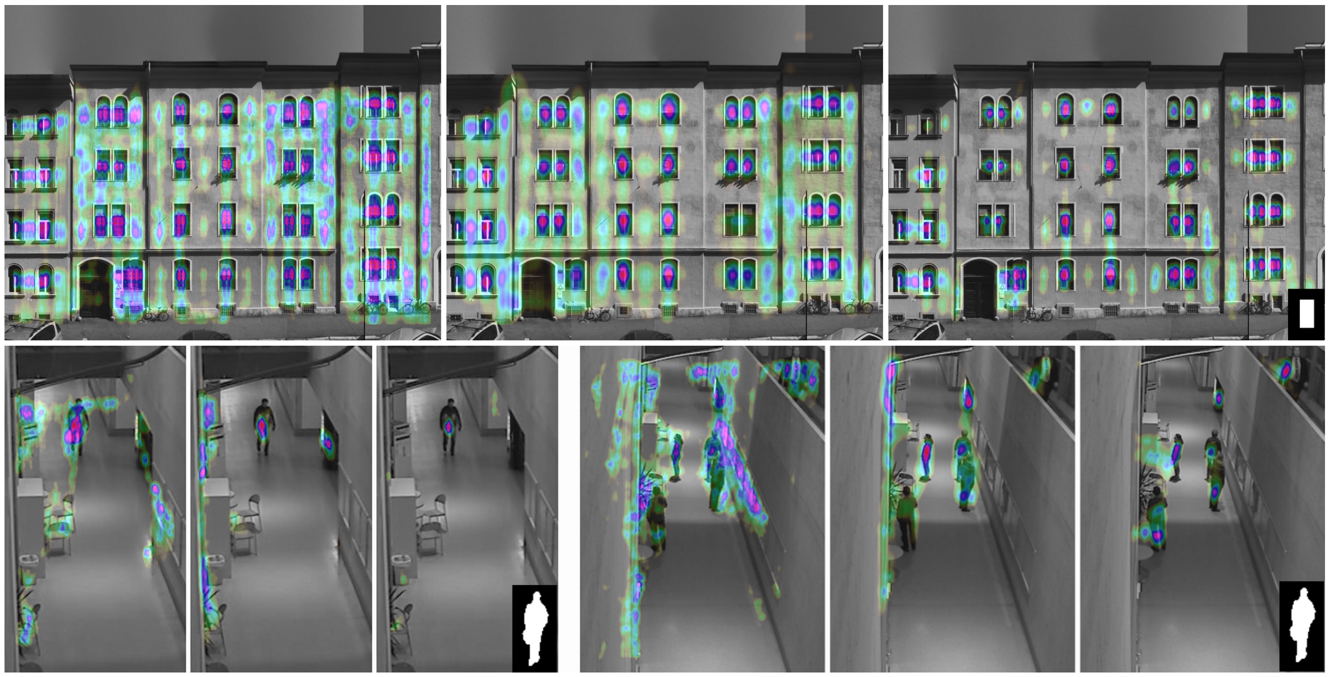

Figure 4. Vote maps returned by oriented chamfer matching using a window (first row) and human (second row) shape prototype (shown on the right side of the figure). First result is always from Canny, second from Berkeley and the third from our proposed edge detector. Best viewed in color.

[16] D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 183–196, 2008.

[17] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, pages 575–588, 2006.

[18] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 503–510, 2005.

[19] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(7):1270–1281, 2008.

[20] S. Zheng, Z. Tu, and A. Yuille. Detecting object boundaries using low-, mid-, and high-level information. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.