Finding Stable Extremal Region Boundaries*

Hayko Riemenschneider, Michael Donoser, and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{*hayko,donoser,bischof*}@*icg.tugraz.at*

Abstract

This paper introduces a novel boundary detection framework finding the most stable region boundaries in grayscale images. In contrast to common detection algorithms as Canny, which only analyze local discontinuities in image brightness our method also integrates mid-level information by analyzing regions that support the local gradient magnitudes. We build a component tree where every node contains a single connected region obtained from thresholding the input image and edges define the spatial relations between the nodes. Then connected nodes at different levels in the tree are compared by a simplified chamfer matching method. Matching regions have boundaries that stay similar over several image intensities, and thus contribute as stable edges to the final result. Since the component tree can be calculated in linear time and chamfer matching between nodes in the component tree is reduced to an analysis of the distance transformation, results are obtained in a very efficient manner. Furthermore, the proposed detection algorithm automatically labels all identified boundaries during calculation avoiding the required post-processing of connecting and labeling edge responses for usage in a detection framework. We compare our method against standard Canny and the Berkeley edge detector on the ETHZ shape classes and the Weizmann horses dataset, which both demonstrate better performance in reducing clutter.

1 Introduction

Edge detection is one of the most intensively studied problems in computer vision. It is used in many applications like object detection, stereo matching, segmentation or tracking. Extracting edges mainly has the purpose of reducing the data amount while retaining the important information about the image content.

In general, edge detection methods can be divided into unsupervised approaches which simply analyze local intensity differences and the recently popular group of supervised methods, which try to learn the appearance of edges, e. g. for specific tasks [5] or in a more general way for natural images [12, 13]. Of course, learning based methods provide improved results, nevertheless these methods have the main drawback of high computation time, for example in the range of 12 seconds [5] and 90 seconds [25]. Also for new specific tasks, they require ground truth labeling to learn a new detector. Therefore, purely local edge detectors are still of high interest because of their computational simplicity.

*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.

Although edge detection has many different applications in computer vision, in this paper we mainly focus on the application of detecting objects in cluttered images. Recently many different shape cue based detection frameworks [6, 19, 21] were proposed achieving state-of-the-art results on reference datasets outperforming appearance based methods.

All these methods still rely on a post-processed Canny edge [2] or Berkeley detection [13] result as input. Post-processing is necessary because the most object detection frameworks require labeled lists of connected edges in images, which is mostly done by analyzing T-junctions and building multiple branches of the obtained edges, or even complex contour grouping [7].

In this paper we introduce a novel boundary detection framework which outperforms local detectors as for example the Canny detector by additionally analyzing mid-level cues, i. e. regions that support the local gradient magnitude are analyzed to extract the most stable edges. We are able to remove a lot of noise in the edge images, preserving the important edges for detection. Furthermore, our method allows labeling of all obtained edges during calculation. Therefore, no post-processing is required and the results can be directly used in standard object detection frameworks.

The outline of the paper is as follows. Section 2 gives an overview of the related works for Canny and Berkeley edge detection. Section 3 describes our novel boundary detection method in detail. Our method returns a list of labeled, connected edges that can be passed to any detection framework. In Section 4 we demonstrate on the well-known ETHZ shape classes dataset, that our proposed detection method outperforms other methods in the scope of object detection. 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 [2]. It provides a simple and fast technique with the goals of many detections, accurate localization and single response per edge. The method itself is split in multiple stages. First, a Gaussian blurring is applied to reduce the noise of individual pixels. Second, a Gaussian derivative response is calculated in two directions and their magnitudes. These individual gradients are used to determine the maximum response of four directions, i. e. 0, 45, 90, 135 degree. The gradient image undergoes a non-maxima suppression to filter out weaker responses. An important aspect here is the thresholding of gradient responses, which is done by hysteresis thresholding in Canny. Hysteresis uses two thresholds, where gradients below the low threshold are not marked, and those above the high threshold are marked as edges. The gradient values which fall between the thresholds are only marked as an edge, if they are connected to an edge of strong gradients. In this work, we investigate the standard behavior of edge detection methods. For this purpose the Canny edge detection is run with standard thresholds, which are automatically calculated based on the gradient information. The high threshold is set to a value so that more than 70% of the pixels are not edges. In terms of strong responses, this results to edges for only the top 30% of the gradients. The low threshold is set to 40% of the high threshold. This is the standard method of automatic thresholding for Canny as it is implemented in Matlab.

The Berkeley edge detector [13] is a supervised algorithm usually tuned to natural scenes. The method works by analyzing changes in image intensity, oriented energy, brightness gradient, color gradient, texture gradient and localized texture gradient. These image features are then combined to improve detection of natural boundaries. For the combination process supervised learning is applied which uses a ground truth dataset given by multiple human segmentations. In an experiment with humans

they asked them to draw boundaries of equally important "things" in the images. The results are accurate object boundaries. This ground truth was used in a logistic regression model to learn an optimal combination of the image features. The evaluation benchmark for their experiments is called the *Berkeley segmentation benchmark* and is publicly available[1]. It will be the basis for our experiments in Section 4 and is described in more detail there.

3 Extremal Region Boundary Analysis

Our boundary detection method is based on analyzing a specific data structure named the component tree, which can be build for any connected graph with node values coming from a totally ordered set. For example any grayscale image fulfills this requirement. Nevertheless, also color images can be analyzed if an ordering of the color pixels is provided, for example as shown by Forssén [8]. Section 3.1 describes how the component tree is built in an efficient manner. For detection of the edges, we then analyze the tree by comparing the outer contour of regions at different levels in the component tree by 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 stable ones as our boundary detection result. In Section 3.2 this is explained in detail.

3.1 Component Tree

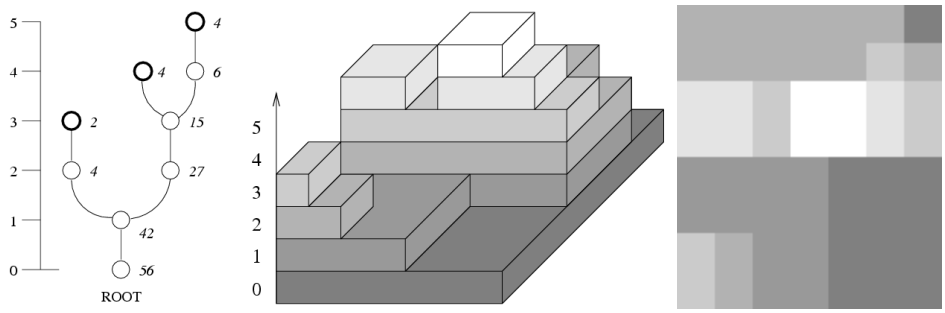
For detecting stable boundaries in the image we analyze connected regions of binary threshold results, which Matas et al. denoted as extremal regions. Therefore we build a hierarchical data structure named component tree for the input grayscale image.

The component tree, originally introduced in statistics [9, 24] for classification and clustering, was redefined by Jones [10] 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"*. Jones further defines it as a *"set of nodes connected by a set of edges"* [10]. A component tree is an acyclic directed graph with nodes corresponding to pixels, or later extremal regions, and edges defining the relationships between their intensity levels. The root node connects all components merged into a complete representation of the input image. At any chosen intensity level of the hierarchy, the component tree contains the extremal regions detected at this level, see Figure 1 for an illustration of the component tree as a graph, a layered hierarchy and the underlying image information. The component tree inherently encodes the structure of the image by inclusion of a lower intensity level in the next intensity level.

A component tree allows to quickly access extremal regions and to store the necessary meta-information in its nodes. There exist several approaches [4, 11, 15, 16, 17, 20] to build component trees. The approach by Najman and Couprie [16] delivers the best runtime complexity to maintain even in worst-case a very efficient algorithm. The most cited reference by Salembier et al. [20] proposes an algorithm determined to be of quadratic runtime for general cases [17]. However, for reduced ranges such as grayscale images, the algorithm would be twice as fast as Najman and Couprie.

Najman and Couprie [16] describe an algorithm to build the component tree in quasi-linear time. The term *quasi* is an amortization of the costs required for the union / find problem, as described by Tarjan [23]. In detail the worst-case complexity is $O(N \times \alpha(N))$ with $\alpha(N)$ being the inverse Ackermann

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/



(a) Graph (b) Layers (c) Image

Figure 1: Illustration of a component tree: a) Graph of connected region nodes with the intensity levels on the axis and the region sizes next to nodes. The root contains all pixels of the image. Corresponding hierarchical layers as shown in [10]. The rightmost part c) shows the corresponding image which is represented by the component tree. The component tree inherently encodes the structure of the image, for example by separating the lighter areas by splits in the tree structure.

function, i. e. practically linear. Thus, the only other dependency is the sort process. For our approach a CountingSort [3] delivers the required linear complexity for sorting the image data.

The high-level idea of [16] is to scan the nodes and their neighbors, which are initially just the pixels of an image. This is done in decreasing order of their intensity to group them whether they are of the same level or establish a hierarchical organization in partial trees. Najman and Couprie make use of two disjoint sets, one to build the partial trees and another to maintain the set of equivalent nodes. They define *canonical nodes* to be the dominant nodes chosen as representatives in a set of equivalent nodes or in a partial tree. The concept of *union by rank*, allowing the union of two sets to return a *canonical node*, is designed to stop complete degeneration of the tree. The second technique known as *path compression* creates a reference list of each node's root. This allows maintaining a low cost when finding and accessing the respective root nodes from any node within a partial tree without navigating up to the root. In these terms, every canonical node and its connected pixels represent an extremal region.

Thus, for every node we can retrieve the pixels for the extremal region, its intensity level, the location of the region and its hierarchy in relation to other extremal regions. The component tree is also the basis for the calculation of Maximally Stable Extremal Regions (MSERs) [14]. Recently Nistér and Stewénius [18] even showed that calculating MSERs is possible in linear time. The distance transform only operates in a local window around the region contour and thus also only requires linear time.

3.2 Stable Extremal Regions Boundaries

We now use the calculated component tree described in Section 3.1 for detecting and labeling boundaries in the input image. The underlying idea is quite similar to the detection of Maximally Stable Extremal Regions (MSER) as proposed by Matas et al. [14]. MSER detection also uses the component tree as data structure 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.

In our case we also estimate a stability value per node, but this time analyzing a boundary stability between the regions. Thus, in contrast to MSER detection which compares region sizes, we measure

how similar the outer contour 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 denote a region R_i^g as the extremal region in the component tree at the threshold g and unique identifier i.

We define the stability value $\Psi(R_i^g)$ of a region R_i^g by comparing its neighboring region $R_k^{g-\Delta}$, which is the extremal regions upwards the component tree. Starting from region R_i^g we move along the tree until a region with gray value $g - \Delta$ is found. Let $\mathcal{C}_j$ and $\mathcal{C}_k$ be the corresponding region boundaries for the current region and its neighbor, respectively.

To measure the contour similarity and to identify similar parts between the regions, we apply a simplified version of the chamfer matching. Since regions within the tree can only grow and always are fixed at the same location, chamfer matching is reduced to an analysis of the distance transformation values. Here we describe the most simple approach of standard chamfer matching, but please note that more sophisticated methods, such as the rotational chamfer matching of Shotton et al. [22], or in general any partial shape matching method can be used in this step.

Thus, as a first step we calculate the distance transformation DT_j for the binary map containing the boundary pixels of $\mathcal{C}_j$. This distance transformation DT_j allows to find partial matches between the boundaries $\mathcal{C}_j$ and $\mathcal{C}_k$ by finding connected chains $\overline{\mathcal{C}}_k \subset \mathcal{C}_k$ fulfilling

$$\overline{\mathcal{C}}_k \subset \mathcal{C}_k \rightarrow DT_j\left(\overline{\mathcal{C}}_k\right) \leq \theta, \tag{1}$$

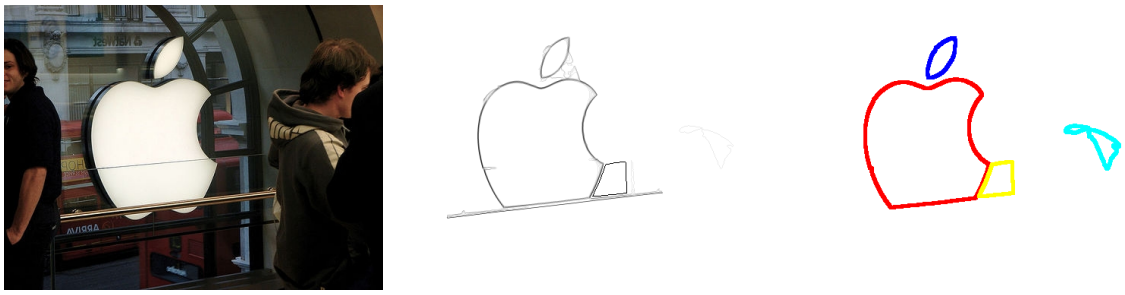
where θ is a maximum boundary distance parameter. Then, for a region R_i^g the stability value $\Psi(R_i^g)$ is set to the average chamfer distance of the matched boundary pixels $\overline{\mathcal{C}}_k$ by

$$\Psi(R_i^g) \ = \frac{1}{N}\sum_{n=1}^{N} DT_j\left(\overline{\mathcal{C}}_k\right), \tag{2}$$

where N is the number of matched pixel. In this way we get a similarity measure $\Psi(R_i^g)$ and matched connected boundary chains for every comparison, and thus every extremal region. The idea is to integrate mid-level information by analyzing regions and matching boundaries that support the local gradient magnitudes. The parameter θ is set to ten and controls how much change in the contour is allowed. The parameter Δ is two and describes how much contrast or intensity levels in terms of the component tree hierarchy are considered.

After calculation of the stability values for every node in the component tree, we detect the most stable ones within the tree and return the corresponding boundary chains as detection result. Since every boundary can be uniquely assigned to a node in the tree, a unique ID can be assigned to every boundary detection. Therefore, cumbersome post-processing to connect and clean edges as for example required for the Canny or the Berkeley edge detector is not necessary.

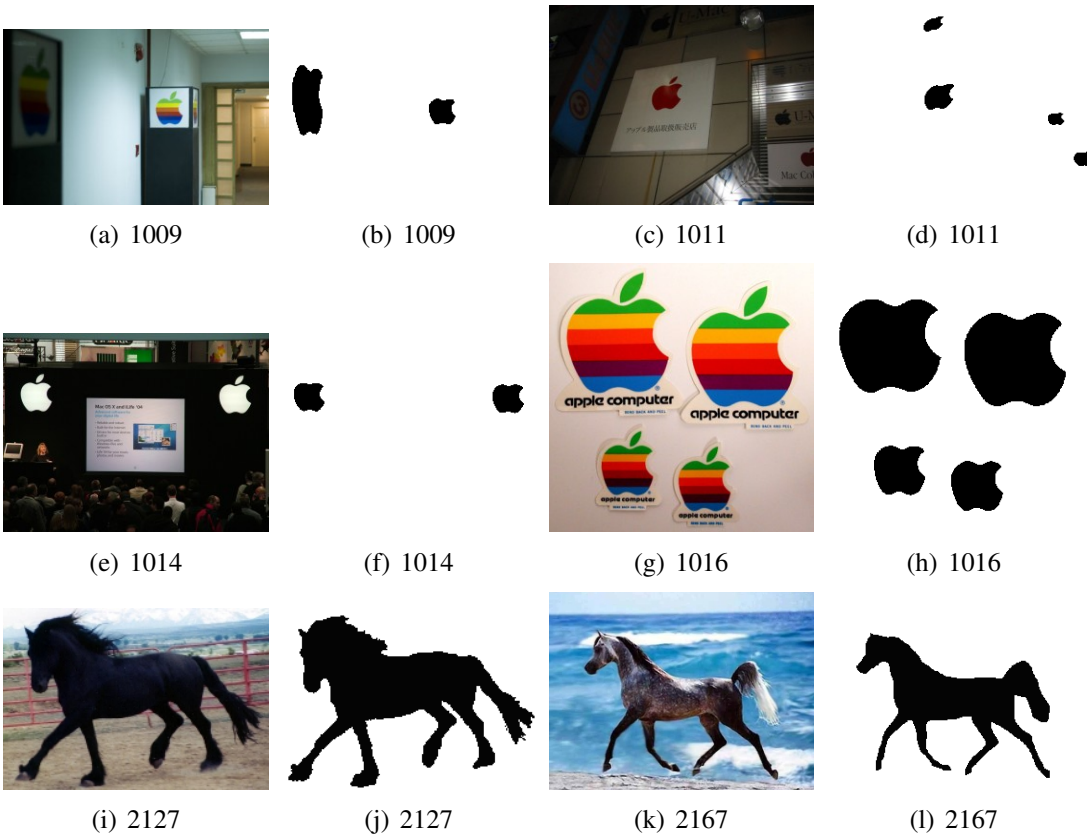
Please note, that the boundary detection results are quite different from simply returning the boundaries of an MSER detection result. First, since we find similar outer contours between extremal regions at different intensity levels, the returned boundaries need not be closed and can represent only part of region's outer contour. Furthermore, since our analysis focuses on whether local parts of the boundary are stable and not only the region sizes. The MSERs for the same image can differ significantly.



(a) Input (b) Score response (c) Our boundary labels

Figure 2: Example of the detection process. For an input image a) a component tree is built for all intensity levels. Outer contour matching between the extremal regions returns a score for each boundary match, which summed up creates b) the score response. The final result c) are the boundary detections and their unique labels.

Refer to Figure 2 for an illustration of the steps when processing an input image and the retrieved gradient and label results. The component tree is build for the intensity levels of the image. The chamfer matching returns a score for every boundary fragment of the extremal regions. The sum over all these scores is returned as a gradient map. However, the final results are our boundary detections and their unique labels, as shown in the right-most part of the figure.



(a) 1009 (b) 1009 (c) 1011 (d) 1011

(e) 1014 (f) 1014 (g) 1016 (h) 1016

(i) 2127 (j) 2127 (k) 2167 (l) 2167

Figure 3: Overview of the ETHZ shape classes [7] (here only apple logos) and their respective ground truth and some horses from the Weizmann dataset [1]. These ground truths are used in the evaluation of edge detection results.

#	P	R	F	ETHZ apple logo	#	P	R	F	Weizmann horses
1	1.00	1.00	1.00	Human segmentation	1	1.00	1.00	1.00	Human segmentation
2	**0.15**	**0.61**	**0.25**	**Our detector**	2	0.39	0.72	0.51	BEL Horse edges [5]
3	0.08	0.95	0.15	Berkeley edges [13]	**3**	**0.32**	**0.45**	**0.37**	**Our detector**
4	0.02	0.99	0.05	Canny edges [2]	4	0.14	0.94	0.25	Canny edges [2]

Table 1: Comparison over the overall F-measure performance of each algorithm for the two datasets ETHZ shape class *apple logo* [7] and Weizmann horses [1]. Please note, that the benchmark is very strict given that only boundaries of the *apple logo* and horses are marked in the human ground truth segmentation. Our boundary detector provides an encouraging improvement over the Berkeley and Canny edge responses.

4 Experiments and Discussion

The focus of the experimental evaluation lies in demonstrating the reduced noise and high retrieval of valuable stable edges. For this we employ the Berkeley segmentation benchmark, which allows to measure the quality of edge maps compared to human ground truth data. In this work we use the ETHZ shape classes [7] and Weizmann horses [1], and a set of figure / ground segmentations as ground truth. For the ETHZ dataset, we created the figure / ground segmentation, whereas for the Weizmann horses it was provided. In both cases the segmentations are object specific and based on the actual image data. Thus, the results are very accurate given the specific object, see Figure 3 for a small subset of ground truth examples.

For evaluation of the results respective to this new defined ground truth data, the edge detection results are compared to the ground truth in the same manner as in the Berkeley segmentation benchmark. This is the pixelwise comparison between two binary images. The result of each edge detection process is a set of binary edges in the image. The Berkeley benchmark prefers softmaps, which encode all responses of the edge detector and then determines the best threshold. However, this is not the focus of our evaluation. We wish to show that given the standard thresholds provided by an edge detector (without any manual tuning), our boundary detector provides exacter edges and far less noise than for example a Canny or Berkeley edge detection.

We use precision and recall as measures for evaluation. 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. Further, we calculate the F-measure as a final comparison value defined as weighted harmonic mean of precision and recall:

$$P = \frac{TP_d}{TP_d + FP_d} \quad and \quad R = \frac{TP_d}{TP_d + FN_d} \quad and \quad F = \frac{2 * P * R}{P + R} \tag{3}$$

where TP_d are all true boundary pixels detected and the denominators $(TP_d + FP_d)$ and $(TP_d + FN_d)$ are all boundary pixels given by the algorithm and all boundary pixels defined in the ground truth data, respectively. The Berkeley benchmark provides precision and recall curves for each threshold of the image. Since all edge responses tested here are binary, we retrieve a single value per image. Using binary responses we can focus on the actual edges returned by an edge detector, rather than all the noisy unimportant responses. We convert our boundary detection labels also to binary edge masks.

Table 1 shows the result of the experiments. For comparison we tested the automatic threshold for a Canny edge detection, all edges of the Berkeley edge detector, which is provided for the ETHZ

dataset, and our detection results. A similar procedure was followed for the Weizmann horse dataset, here we give results for provided edge response from Boosted Edge Learning (BEL) for horses, again standard Canny edge response and finally our boundary detector.

The improvement of our boundary detector are encouraging for both dataset (20% gain for ETHZ and 12% gain for the Weizmann over standard Canny). For the learned detectors, we achieve better results than the Berkeley detector (10%) and worse results against the BEL results specifically trained to horses while our approach does not use any prior shape information.

The advantages of our algorithm are clear when looking at Figure 4. Our boundary detector produces far less noise when detecting edges. Only boundaries are returned which are stable due to their support from larger regions. This removes a lot of the cluttered edges which are present in other edge detection algorithms. Occasionally some edges are missing, which are present in the highly dense responses of Canny, which reflects in the lower recall than other detection methods. 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 strong edge response is seen as a false positive. This is less severe in the case of the Weizmann horses, since the horses are the only dominant object in the images. There exists a lot more background information with strong edges for the ETHZ shape classes.

The second advantage is the implicit labeling returned by our approach. In contrast to the edge responses from Canny or Berkeley, our boundaries are connected and uniquely labeled. This provides a major benefit. First, no post-processing is required to dissect edges from the cluttered responses and split at T-junctions. Second, no additional linking into lists is required. And most valuable complex perceptual grouping is not needed. Our boundary detector inherently provides grouped and connected boundaries, which would otherwise require costly and sometimes destructive post-processing[2].

Of course, in the future we have to evaluate in more detail how boundary detection influences a subsequent object detection step. It would be interesting to see if the removal of noisy edges only reduces computation time or if even detection quality can be improved.

5 Conclusion and Outlook

This paper introduced a novel boundary detection algorithm, based on the idea of analyzing extremal regions within a component tree and measuring their boundary stability by a shape comparison. The benefit of this detector is the integration of context information in terms of support by connected regions. Experimental evaluations using the Berkeley segmentation benchmark on different datasets proved that our detector demonstrates significant improvement over standard approaches like Canny edge detection. It accurately removes noise and clutter and delivers a cleaner, more precise and still rich edge response. Future work will focus on the integration of other partial shape matching methods for evaluation of the stability. We will test our detector response in different detection frameworks and investigate the benefit when integrating color similar to Forssén [8]. Another idea is to learn the appearance for certain classes as new input to improve the detection results. Finally, an interesting idea is to evaluate the repeatability of our stable boundary detector for use as interest points.

[2]Splitting an edge response at a T-junction leaves three or more edges only as parts of the original edge behind. Their complete contour is lost unless costly merging of all combinations or perceptual grouping is performed

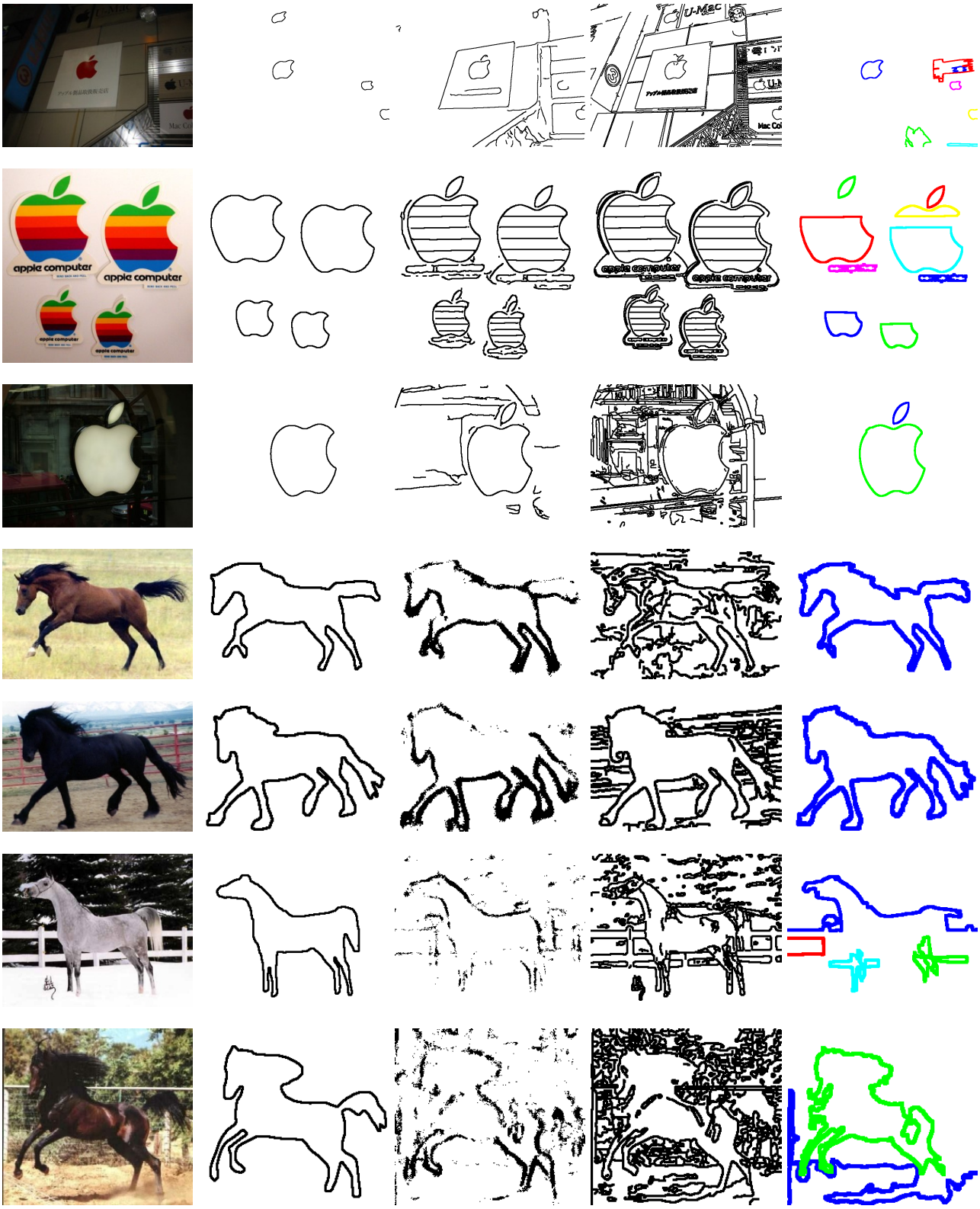


Figure 4: Results for the ETHZ shape class *apple logo* [7] and Weizmann horses [1] shown from left to right by the 1) input images, 2) the human ground truth, 3) the Berkeley [13] and 4) Canny edge [2] responses, and 5) our boundary detection labels. These results show the improvement of our boundary detector over the standard edge responses of other detection algorithms. Our detector returns far less noise and still precise stable edges, and additionally the boundary responses are already connected and uniquely labeled. This removes the requirement for expensive post-processing and grouping required for other methods. Please note, the edges are thickened for better visual appearance and our boundary labels are best viewed in color.

References

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

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

[3] T. Cormen, Ch. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, June 1990.

[4] M. Couprie, L. Najman, and G. Bertrand. Quasi-Linear Algorithms for the Topological Watershed. *Journal of Mathematical Imaging and Vision (JMIV)*, 22(2–3):231–249, January 2005.

[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, Washington, DC, USA, 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, January 2008.

[7] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object Detection by Contour Segment Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3, pages 14–28, 2006.

[8] P. Forssén. Maximally Stable Colour Regions for Recognition and Matching. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, USA, June 2007. IEEE Computer Society, IEEE.

[9] J. Hartigan. Statistical theory in clustering. *Journal of Classification (JOC)*, 2:63–76, 1985.

[10] R. Jones. Component trees for image filtering and segmentation. In *IEEE Workshop on Nonlinear Signal and Image Processing (NSIP)*, September 1997.

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

[12] 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, January 2003.

[13] 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.

[14] 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.

[15] V. Mosorov and T. M. Kowalski. The development of component tree structure for grayscale image segmentation. In *Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, pages 252–253, February 2002.

[16] 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.

[17] L. Najman and M. Couprie. Building the Component Tree in Quasi-Linear Time. *IEEE Transactions on Image Processing (TIP)*, 15(11):3531–3539, November 2006.

[18] D. Nistér and H. Stewénius. Linear Time Maximally Stable Extremal Regions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 183–196, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] 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.

[20] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing (TIP)*, 7(4):555–570, April 1998.

[21] 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.

[22] 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.

[23] R. Tarjan. Efficiency of a Good But Not Linear Set Union Algorithm. *Journal of the ACM (JACM)*, 22(2):215–225, April 1975.

[24] D. Wishart. Mode analysis: A generalization of the nearest neighbor which reduces chaining effects. *A. J. Cole, editor, Numerical Taxonomy*, 2:282–319, 1985.

[25] 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.