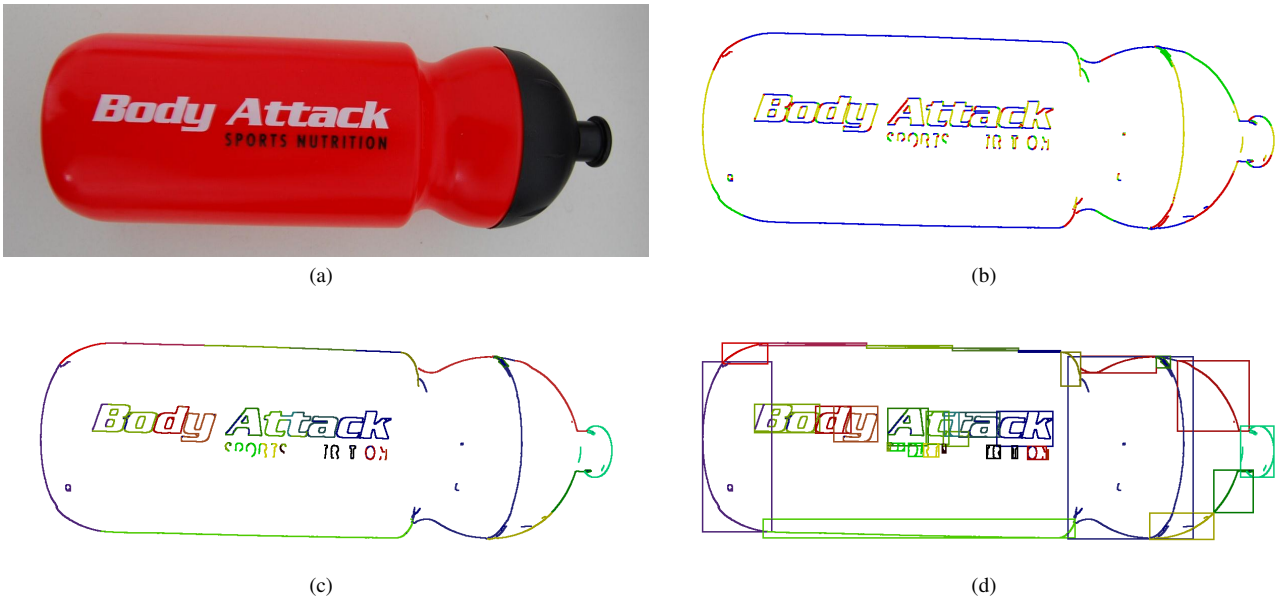
ConGrap - Contour Detection based on Gradient Map of Images

IPCV'11 - The 2011 International Conference on Image Processing, Computer Vision, and Pattern Recognition

Frank Nagl* Konrad Kölzer† Paul Grimm‡ Tobias Bindel§ Stephan Rothe¶

Erfurt University of Applied Sciences



(a) (b) (c) (d)

Figure 1: ConGrap contour detection: (a) Initial image. (b) Gradient Map (blue horizontal, red positive diagonal, yellow vertical and green negative diagonal edges). (c) Resulting contour image. (d) Contours including their bounding boxes.

Abstract

In this paper, we present **ConGrap**, a novel contour detector that uses the Gradient Map and results in closed contours with semantic connections. In contrast to common edge and contour detections, ConGrap not only produces an edge image, but also provides additional information (e.g. the borderline pixel coordinates, the bounding box, etc.) about every contour.

Additionally, the resulting contour image provides closed contours without discontinuities and merges regions with semantic connections together. Consequently, the ConGrap contour image can be seen as an enhanced edge image as well as a kind of segmentation and object recognition.

Keywords: Pattern Recognition, Contour Detection, Edge Detection, Segmentation, Gradient Map

Index Terms: I.4.6 [Image Processing And Computer Vision]: Segmentation—Edge and feature detection

*e-mail: frank.nagl@fh-erfurt.de
†e-mail: konrad.koelzer@fh-erfurt.de
‡e-mail: grimm@fh-erfurt.de
§e-mail: tobias.bindel@fh-erfurt.de
¶e-mail: stephan.rothe@fh-erfurt.de

1 Introduction and Motivation

Object recognition in digital images is one of the most important and challenging tasks of computer vision. The type of objects to be detected can range from abstract geometric objects like lines or circles to complex real objects like human faces or finger prints. The majority of object recognition algorithms involve edge detection as a preprocessing operation to perform a simple structural analysis of a digital image.

However, edge detection only provides a pixel-wise classification of edges, which does not allow conclusion on the shape of complex objects present in the image. A much better representation for the visible shape of objects can be achieved by finding the silhouettes of objects. In contrast to edges, silhouettes already provide an abstract representation of the image contents and can be used to divide the image into foreground objects and background. Moreover, it is an interesting question of computer vision to divide the scene of an image not only in foreground and background, but also in semantic segments.

Therefore, we present **ConGrap**, a novel algorithm that finds closed contours of objects as polygon line paths. This contour detection generates the **Gradient Map** of the image by using a gradient-based edge detection that results in borders of contours by present edges and their orientations.

ConGrap works as a multiple-stage algorithm. Using the edge image and the generated Gradient Map, a contour tracer divides the image in semantic parts. For each edge pixel a three-stage hierarchical analysis of neighbored pixels is performed. In this process

each edge pixel is mapped to a closed contour. Following this step, the borderline is extracted by tracing each contour for detecting its outside pixels. A final postprocessing optimizes the contour image by merging contours with the same semantic relation.

This paper is structured into 5 sections. The next section investigates related edge and contour detection as well as segmentation algorithms. Section 3 describes the requirements and the concept of our novel contour detection for finding closed contours of objects as polygon line paths. The following section 4 presents the implementation of ConGrap as well as produced results and evaluates them. The last section provides a summary and discusses issues about potential future work.

2 State-of-the-Art

Dominant edges in an image are important cues for finding abstract shapes – both for computers and humans. Edges are locations in an image, where the intensity in a small local area along a direction varies strongly. The higher the variation, the more significant the edge appears in the image. Mathematically, edges can be calculated as the first order derivative of the image intensity function.

The most common gradient-based operators for edge detection are Sobel, Prewitt, Roberts, Kirsch and Sobel [4]. They use the first order derivative and differ in usage of direction and size of the pattern for selecting the candidate pixel. This results in thick and not locatable edges.

The Laplacian edge detector also calculates the second order derivative for thinner and exacter edge lines. The most widespread variation is known as Laplacian of Gaussian (LoG) or the Marr-Hildreth operator, which extends the basic approach by noise suppression [4].

Another edge detector for thin and exact edge lines is presented from John Canny [2]. The Canny algorithm is also based on the first-derivative combined with noise reduction. It uses a set of directed filters that are used on different levels of detail and the result is summarized into an edge-map. The purpose of this approach is to reach three goals: Good detection, good localization and one response to edge. The first stage of the algorithm is to convolve the gray scaled image with a Gaussian function - with an appropriate pattern size - to get a smooth image. After that, the first gradient operator is used to find the edge strength. The third stage is called non-maximal suppression and delivers a set of edge points in the form of a binary image. Finally a double thresholding with low and high thresholds, called hysteresis, is applied to the output of stage three and delivers the edges in the image.

All the introduced edge detectors give no information about potential relations between the edge pixels. The detection of a connected region or an object is not possible.

In contrast to edge detection, the Hough-Transformation [11] detects straight lines. Its basic idea is to accumulate all line candidates for all edge pixels. The line parameters slope and intercept define a line candidate and span a two-dimensional parameter space. A 2D accumulator array is used as discretized version of this parameter space to collect votes for individual line candidates. After that, local maxima in the accumulator array represent significant lines in the image. Duda and Hart [6] generalized the hough transformation to detect any kind of shapes (e.g. circles or ellipses) that can be described by a suitable parametrizations. Also they introduced a more efficient parametrization for straight lines by using normal angle and distance from origin as parameter.

In contrast to the gradient based operators, [19] presented an approach to detect edges with anisotropic diffusion. Black et. al. [1] presented an improved version of this by using a new edge-stopping function based on the Tukey's biweight estimator to get sharper boundaries. Also these approaches do not detect connected regions.

Many approaches for automatic edge and contour detection as well as image segmentation have been proposed over the years. In [10] Grigorescu et al. presents a biologically motivated algorithm to improve the results of the Canny edge detection in natural images by distinguish between isolated contours of a shape and edges or lines that are part of a texture. A further development presented in [9] involves the mechanism of the primary visual cortex of primates that influences the perception of groups of edges or lines to suppress the responses in surrounding textures. Joshi and Sivaswamy [12] purpose a similar scheme inspired by the mechanism of surround influence recognized in the primary visual cortex, where locally found responses by a gradient computation are modulated by the responses of the global image. This scheme uses a Sobel edge detector followed by a mask operation for the surrounding influence. Another approach of a biologically motivated multi-resolution contour detection is proposed in [18]. First, the gradient at different resolutions is computed. Next the biologically motivated inhibition step is applied to suppress lines or edges of surrounding textures. Finally, long connected lines are preferred in a contour-oriented binarization algorithm rather than short lines, which mostly belong to a texture. All these approaches result in edge images, giving no information about they belong to each other. Also closed contours are not guarantied.

An isoperimetric graph partitioning method for image segmentation is presented in [7], which produces the same quality segmentation of spectral graph methods, but requires only the solution to a linear system rather than an eigenvector problem.

Another approach based on a fuzzy logic implementation is presented by Losson et al. [14]. The pixel classification is performed by detecting the modes of a spatial-color compactness function, witch classifies the colors by the location in its associated color-space and the location in the image itself. Again, these methods only provide an edge image as result.

A none fuzzy logic classifier was proposed by Martin et al. in [16] to detect boundaries in natural scenes using measurements like changes in brightness, color, and association to textures. The classifier is trained by human labeled images as ground truth and outputs the posterior probability of a boundary at each image location and orientation. In [15] a further development to detect and localize junctions is presented. Also a highly parallelized implementation based on the method of Marin et al. running on commodity graphics processors from Nvidia was proposed in [3]. The disadvantage of this method is the necessity to train the algorithm with ground truth images labeled by humans.

3 Concept

This section presents **ConGrap**, a novel contour detector by using a Gradient Map of the initial image. In contrast to edge and other contour detectors, ConGrap does not not only provide a result image, but also calculates the borderline and the bounding box of every contour. This is similar to segmentation results, because a contour represents a segment of the initial image. Unlike a common

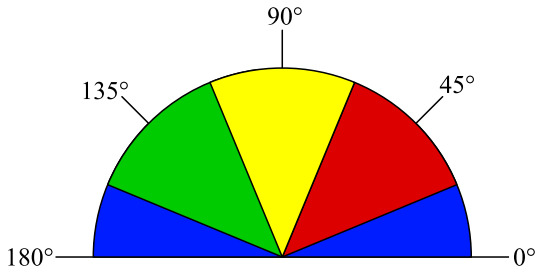
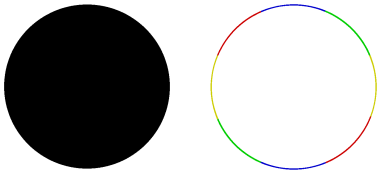


Figure 2: Clamping of edge orientation angle θ into four cases [8].



(a) (b)

Figure 3: Gradient map of a synthetic image: (a) Initial image showing a circle. (b) Corresponding Gradient Map.

segmentation process, the result image of ConGrap also represents an enhanced edge image.

Furthermore, ConGrap works with many different classes of image content (like outdoor and indoor photos, panoramas, photos with large or shallow depth of field, synthetic images, etc.), as long as the underlying Gradient Map contains all significant edges of an image.

The algorithm of ConGrap works in several steps:

1. Gradient-based edge detection,
2. contour tracing,
3. extraction of borderlines, and
4. optimization.

3.1 Gradient-based Edge Detection

Like already mentioned in section 2, a usual edge detection computes the discontinuities of the image intensity function in two – horizontal and vertical – dimensions. Thereby, the derivatives ($f'(x)$ and $f'(y)$) in both directions are used to find the gradient of intensity, which is a measure to identify an edge [5].

Additionally to a gradient image (edge image), the derivatives give information about the edge direction. With the calculated angle θ in eq. 1, the orientation of an edge is known [8].

$$\theta = \begin{cases} 0^\circ & \text{if f'(y) = 0 and f'(x)} \neq 0 \\ 90^\circ & \text{if f'(y) = 0 and f'(x) = 0} \\ \arctan\left(\frac{f'(x)}{f'(y)}\right) & \text{if f'(y)} \neq 0 \text{ and f'(x)} \neq 0 \end{cases} \quad (1)$$

In discrete domain, direct neighbor pixels of a location can only accept four directions. Thus, θ needs to be clamped. Fig. 2 shows the clamp range of θ.

ConCrap stores the clamped orientation θ for every pixel of the initial image. We call this result **Gradient Map**. Fig. 3 shows the visualized Gradient Map, consistent with the clamp range of fig. 2.

3.2 Contour Tracing

The contour tracing mechanism passes through the lines of the edge image. If an edge pixel is detected, which does not belong to a contour yet, the pixel will be the start pixel of a new contour. This ensures that all edge pixels will be part of a contour. Based on this contour pixel cp, a three-stage hierarchical analysis of the pixel neighbors in a specified radius r will be started. It will be checked, if they are

1. edge pixels, when this returns no results then
2. own contour pixels, when this returns no results then
3. other contour pixels.

For each of the three steps, the analysis process searches for the neighbor pixel np that has the highest significance to be part of the contour. This significance is expressed by the match value m. The lowest value of m expresses the highest significance to add np to the contour. Match value m is defined for each neighbor pixel np as the distance between cp and np. Thereby, the distances of both directions will be stored separately in $distX$ and $distY$ (in pixel coordinates). The specified parameter w represents a constant weight to privilege np with the same orientation (θ of the Gradient Map) as cp. This results in the following pseudo code:

```
if (θ(np) == θ(cp))
  m = Max(distX, distY)
else
  m = Max(distX, distY) + w
```

An illustration of calculated match values can be seen in the following sample image showing a red line. Pixel cp belongs to a diagonal red line and is the current pixel to analyze. The parameter w is set to 7.

Image

x	x	x	x	x
x	x	x	x	x
x	x	p	x	x
x	x	x	x	x
x	x	x	x	x

Match Value Matrix

9	9	9	9	2
9	8	8	1	9
9	8	p	8	9
9	8	8	8	9
2	9	9	9	9

Figure 4: Illustration of calculated match values: Initial image with a diagonal red line and the corresponding match value matrix.

The resulting match value matrix shows that the direct neighbor top right has the lowest match value ($m = 1$) and hence the highest significance. Congruously, this neighbor np becomes part of the contour of pixel cp. A line will be drawn into the edge image and connects cp with np. If the analysis process works in step 2 or 3, the contour tracing closes the contour with this pixel. If the the analysis process works in step 1, the analysis starts from the neighbor position again. This recursive process is done until the contour will be closed or no pixels will be detected inside the specified radius r.

3.3 Extraction of Borderlines

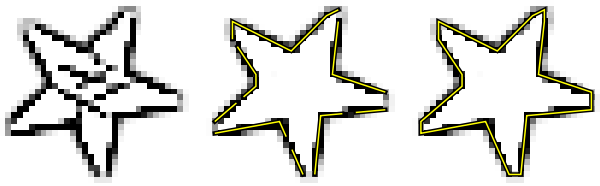
After contour tracing, the edge image is transformed into an image with advanced edges as contours. In the final step, all contours have to be reduced to their borderlines as a closed polygon line path.

Again the (advanced) edge image will be passed through the lines and also through the columns. The first and last contour pixel per line and per column will be kept, all other pixels will be removed. After passing through the image, the contour pixels are divided into four groups: All left, right, upper and lower pixels. The pixels of a group will be connected by a line. Additionally the first pixel of the left group will be connected with first pixel of right group. The same procedure is done with the last pixels. Also the first and last pixels of the upper and lower groups will be connected. This results in a closed contour represented as closed polygon line path. Fig. 5 shows the intermediate steps from an advanced edge image to a closed contour.

3.4 Optimization

ConGrap is configurable with several parameters, like the radius r for tracing neighbor pixels and the weight w to privilege neighbor pixels with same θ at the Gradient Map.

A further parameter p defines a percentage rate for joining contours. The edge image can contain discontinuities in an edge line. When the contour tracer cannot compensate a discontinuity an edge



(a) (b) (c)

Figure 5: Getting a closed contour: (a) All pixels of one contour after analysis algorithm. (b) Detected borderline of the contour including discontinuities inside the borderline. (c) All borderline pixels connected by a polyline to a closed contour.

will be divided into multiple contours. Also an edge image with thick edges results in wrong multiple contours. Parameter p determines how many percent of a smaller contour has to intersect a bigger contour for merging together. Fig. 6(a) shows an edge image, which results in multiple contours after contour tracing (see fig. 6(b)). With $p = 90\%$, the inner contours are merged to the biggest contour (see fig. 6(c)).

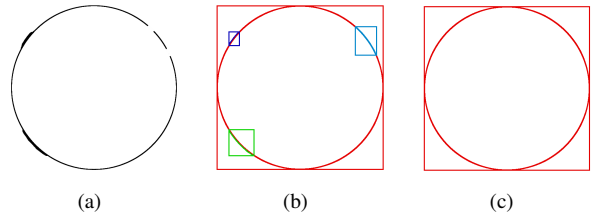
The last parameter for manipulating ConGrap is *mcs*. *mcs* determines the minimum contour size, more precisely the minimum number of pixels belonging to a contour. This parameter avoids, that image noise results in many very small wrong contours.

4 Realization and Results

4.1 Details Of Implementation

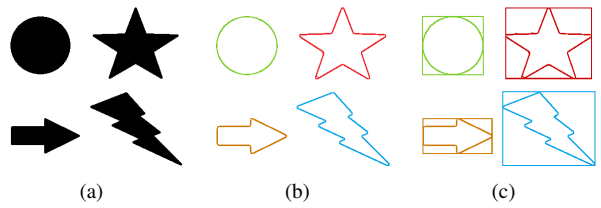
ConGrap is implemented as a component of the open source image and video processing framework SBIP [17]. Like described in section 3, the realization of ConGrap works in four steps. The first step realizes a gradient-based edge detection. Therefore, a modified version of the canny edge detection of the AForge.Net Framework is used [13]. Based on the canny edge implementation, ConGrap generates the Gradient Map. For this purpose, the Sobel edge detection as first step of the canny algorithm is enhanced and stores the edge orientation θ in a temporary image.

To avoid stack overflow problems, the recursive processes of the contour tracer (see section 3.2) are implemented as iterative routines. Also in contrast to our approach in section 3.3, our implementation does not remove non-borderline pixels obligatory. This can be activated by an additional parameter. The reason for this modification is to hold all edge information in the result image. If only the contour borderlines are shown, potential semantic incorrect contours could remove important edge pixels. This would reduce the quality of the resulting contour image as enhanced edge image. The resulting contour image has an 8 bits per pixel format and uses indexed colors / pseudo colors [20]. This results in a colored contour image and the color index is used as contour identifier. Furthermore, ConGrap provides all corresponding information of a contour like a list of the borderline pixel coordinates, the bonding box coordinates and the contour identifier.



(a) (b) (c)

Figure 6: Merging smaller contours, which intersect a bigger contour (contour images with corresponding bounding boxes): (a) Edge image with two too thick edge areas on left semi circle and two discontinuities on right semi circle. (b) Four detected contours after contour tracing. (c) Merging the inner contours with the outer red contour.



(a) (b) (c)

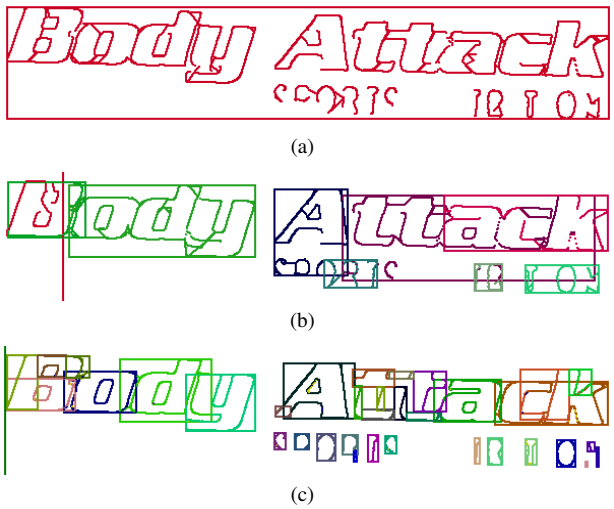
Figure 7: ConGrap contour detection of a synthetic image: (a) Initial image. (b) Result image with four closed contours. (c) Contours with their bounding boxes.

4.2 Results And Evaluation

ConGrap realizes a contour detector that finds closed contours of objects as polygon line paths. Fig. 7 shows the result of a synthetic image. The image contains four objects, which are accurate detected as four contours.

Concluding, ConGrap works well with synthetic images. But the question is, how does ConGrap work with original photos. In fig. 1, a photo of a real object is divided in several contours (see fig. 1(c) and 1(d)). Every contour represents a semantic part of the object. Depending on the configuration of the parameters of ConGrap, semantic errors can be minimized, but the amount of contours will be increased. Fig. 8 shows the results of the lettering of fig. 1(a) with different configurations. The higher the parameters r and p are, the more contours will be summarized. The higher w is, the more insusceptible is ConGrap for semantic errors, caused by crossing edges, but also the more contours will not summarized, caused by over restrictive behavior.

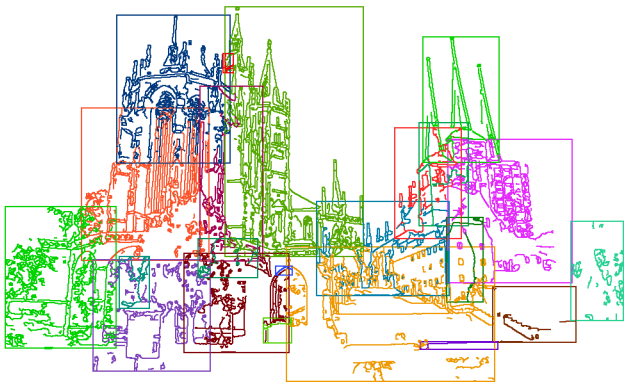
A second example is shown in fig. 9. An outdoor photo of a cathedral consists a very complex image content structure. Thus, a gradient-based edge detection results in many unordered edge pixels. Again, in dependency of its configuration, ConGrap result con-



(a) (b) (c)

Figure 8: ConGrap results with different configurations: (a) Only one contour for all letters ($w = 1$, $r = 3px$, $p = 70\%$, $min = 5px$). (b) 8 contours with many semantic errors ($w = 5$, $r = 10px$, $p = 80\%$, $min = 5px$). (c) Multiple contours, almost one contour per letter, a few semantic errors ($w = 1$, $r = 10px$, $p = 90\%$, $min = 5px$).

(a)



(b)

Figure 9: ConGrap result of an outdoor photo: (a) Initial image, showing the cathedral of Erfurt. (b) ConGraps result image (24 contours) including the bounding box of the contours.

tains many contours with minimized semantic errors versus a low number of contours with higher number of semantic errors. Fig. 9(b) shows a result of ConGrap with 24 contours ($w = 5$, $r = 3px$, $p = 80\%$, $min = 1px$).

4.3 Use Case

Last example in fig. 10 shows a monument, also with complex image content structure. ConGrap divides the image in 719 contours, caused by a configuration for minimized semantic errors ($w = 5$, $r = 3px$, $p = 0\%$ - no summarizing, $min = 5px$). We use these contours for texture analyzing and evaluating, which contours contain to the monument. The extraction of objects from photos (like the monument) can be used for potential occlusion problems of embedded virtual parts in a photo (Augmented Reality) or for image-based 3-D reconstruction of objects.

5 Conclusion and Future Work

In this paper **ConGrap**, a new approach for detecting closed contours with semantic connections was presented. In contrast to common edge and contour detections, ConGrap not only produce an edge image, but also the information, which pixels belong to the borderline of the contour. It is optimized to find silhouettes of objects or semantic connected parts in single images. Therefore, a multiple-stage algorithm was implemented. Based on a gradient-based edge detection, a **Gradient Map** was generated to store the orientation of every edge pixel. In the Implementation, the canny edge detector was used, because of its robust results. Using the edge image and the generated Gradient Map, a contour tracer separated the image in semantic parts and objects. For each edge pixel a three-stage hierarchical analysis of neighbored pixels was performed. In this process each edge pixel was mapped to a closed contour. Afterwards the borderline was extracted by tracing each contour for detecting its outside pixels. A final post-processing optimized the contour image by merging contours with same semantic relation. The results had shown, that ConGrap can work without semantic errors in synthetic images. Also in common photos, the results show minimal semantic errors. But this demands a high number of contours. A smaller number of contours produces a higher number of semantic errors. Related to the resulted image, ConGrap can compensate potential discontinuities in an edge image and produces a enhanced edge image with closed contours.

In current implementation, ConGrap does not consider color information of the image content. More precisely, the hue and saturation will be ignored. In future work, the multiple-stage algorithm of ConGrap will be extended with the evaluation of the object color by converting the photo to the HSL color model. This means, the contour tracer will consider edges, edge orientations and the color of a contour. For gradient-based edge detection, the canny edge detector is used. This detection converts the initial image into gray- scale. The RGB color information get lost. In future work, an alternative edge detection will be used, which processes separately every color channel. This results in three edge images with different gradient values. Based on this, the contour tracer will consider three edge images instead of one, which results in more accurate contours.

Acknowledgements

This work was funded by BMBF (Federal Ministry of Education and Research, project no.: 17N0909) and TAB (Thueringer Aufbaubank , project no.: 2007 FE 9028). Also, we thank Mario Bernsdorf (ZGM Weimar) for his indirect support to this work. Furthermore, special thanks to Mohammad Yudha Kuntjoro (http://www.yudha.de) and Rolf Kruse (Invirt GmbH) for their provided photo material.

References

[1] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. In *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, volume 7, pages 421–32, 1998.

[2] J. Canny. A computational approach to edge detection. *Readings in computer vision: issues, problems, principles, and paradigms*, 184, 1987.

[3] B. Catanzaro, B. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2381–2388. IEEE, 2010.

[4] B. Chanda and M. Dtuta. *Digital Image Processing and Analysis*. Prentice-Hall of India Pvt.Ltd, 2 2004.

[5] B. Chanda and D. D. Majumder. Edge and Line Detection. In *Digital Image Processing and Analysis*, pages 239–277. Prentice-Hall of India Pvt.Ltd, 2004.

[6] R. Duda and P. Hart. Use of the Hough transformation to detect lines and curves in pictures. In *Communications of the ACM*, volume 15, pages 11–15. ACM, 1972.

[7] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(3):469–75, Mar. 2006.

[8] B. Green. Canny Edge Detection Tutorial. `http://www.pages.drexel.edu/~weg22/can_tut.html`, 2009.

[9] C. Grigorescu. Contour and boundary detection improved by surround suppression of texture edges. *Image and Vision Computing*, 22(8):609–622, Aug. 2004.

[10] C. Grigorescu, N. Petkov, and M. Westenberg. Contour detection operators based on surround inhibition. In *Image Processing, 2003.*

ICIP 2003. Proceedings. 2003 International Conference on, volume 3, pages 3–6. IEEE, 2003.

[11] P. V. C. Hough. Method and means for recognizing complex patterns, December 1962. United States Patent 3069654.

[12] G. Joshi and J. Sivaswamy. A simple scheme for contour detection. In *Proc. of the Conference on Computer Vision Theory and Applications*, pages 236–242, 2006.

[13] A. Kirilov. AForge.NET Framework. `http://code.google.com/p/aforge/`, 2011.

[14] O. Losson, C. Botte-Lecocq, and L. Macaire. Fuzzy Mode Enhancement and Detection for Color Image Segmentation. *EURASIP Journal on Image and Video Processing*, 2008:1–19, 2008.

[15] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

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

[17] F. Nagl. SBIP - Shader-Based-Image-Processor. A XNA 4.0-based Framework for real-time capable, graphics board-based Image and Video Processing. `http://code.google.com/p/sbip/`, 2011.

[18] G. Papari, P. Campisi, N. Petkov, and A. Neri. A Biologically Motivated Multiresolution Approach to Contour Detection. *EURASIP Journal on Advances in Signal Processing*, 2007:1–29, 2007.

[19] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 12, pages 629–639, 1990.

[20] C. Poynton. Raster Images - Pseudocolor. In *Digital Video and HDTV Algorithms and Interfaces*, pages 34–40. Morgan Kaufmann, 2002.

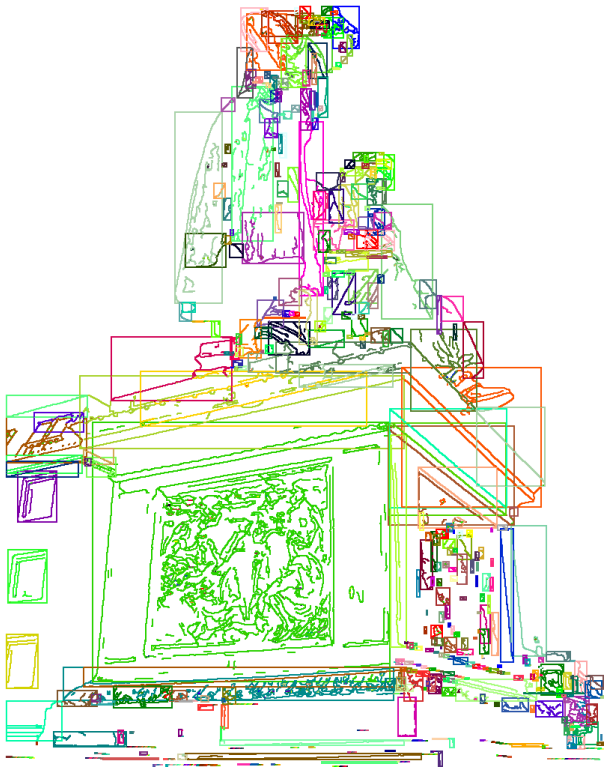


Figure 10: Photo of the the Brothers Grimm monument in Hanau is divided by ConGrap in 719 contours. These contours are used for evaluation, which contour belongs to the monument and which not.