# ENHANCING CURVATURE SCALE SPACE FEATURES FOR ROBUST SHAPE CLASSIFICATION

*Stephan Kopf, Thomas Haenselmann, Wolfgang Effelsberg*

Dept. of Computer Science IV, University of Mannheim, Germany
{kopf,haenselmann,effelsberg}@informatik.uni-mannheim.de

## ABSTRACT

The curvature scale space (CSS) technique, which is also part of the MPEG-7 standard is a robust method to describe complex shapes.The central idea is to analyze the curvature of a shape and derive features from inflection points. A major drawback of the CSS method is its poor representation of convex segments: Convex objects cannot be represented at all due to missing inflection points. We have extended the CSS approach to generate feature points for concave *and* convex segments of a shape. This generic approach is applicable to arbitrary objects. In the experimental results, we evaluate as a comprehensive example the automatic recognition of characters in images and videos.

## 1. INTRODUCTION

One of the central topics in computer vision is the recognition of objects in images and videos. The problem can be subdivided into two parts: segmentation and classification. We focus on the classification aspect and present a generic approach that analyzes the outer shape of a segmented object. Many surveys on the recognition of objects in videos based on shape analysis have been published in recent years [1, 2]. One of the reliable and fast shape classification techniques is the curvature scale space (CSS) technique [3] which was one of the features selected to describe objects in the MPEG-7 standard [4]. A severe, and as yet unaddressed, problem with the CSS approach, is the poor representation of convex segments of a shape. We present a new approach based on the CSS technique with which even convex shapes can be classified.

The remainder of this paper is organized as follows: Section 2 describes our new approach to classify shapes. We then present experimental results in Section 3 and conclude with Section 4.

## 2. CLASSIFICATION OF SHAPES

We analyze the outer shape of an object and derive features for classification. The features are based on the curvature scale space (CSS) technique that is presented in this section. A major drawback of the standard CSS approach is its poor representation of the convex segments of a shape. We propose an approach to solve this problem.

### 2.1. Standard CSS Technique

The CSS technique [3] is based on the idea of curve evolution. A CSS image provides a multi-scale representation of the curvature zero crossings of a closed planar shape. A shape is smoothed with a Gauusian kernel and the CSS image shows the zero crossings with respect to their positions on the shape and the width of the Gaussian kernel (or the number of iterations). An example of smoothed shapes and the CSS image is depicted in Figure 1.

During the deformation process, zero crossings merge as the transitions between shape segments of different curvature are equalized. Consequently, after a certain number of iterations, inflection points cease to exist, and the shape of the closed curve becomes convex. Significant shape properties that are visible during a large number of iterations result in high peaks in the CSS image. However, areas with rapidly changing curvatures caused by noise produce only small local maxima. In many cases, the peaks in the CSS image provide a robust and compact representation of a shape.

### 2.2. Extended CSS Features

Certain shapes that differ significantly in their visual appearance nevertheless have similar CSS images. A major drawback of the CSS approach is the inadequate representation of convex segments on the shape. A CSS image represents the position of the inflection points, so concave segments on the shape are required.

We apply the standard CSS approach first to get characteristic feature vectors that classify concave parts of the shape. The general idea is now to create a second shape – we call it *mapped shape* – that will provide additional features for the convex segments of the original shape. The original shape is mapped to a new shape with an inverted curvature. Strong convex segments of the original shape become concave segments of the mapped shape. Significant curvatures in the original shape are still significant in the mapped shape.

*0-7803-9332-5/05/$20.00 ©2005 IEEE*

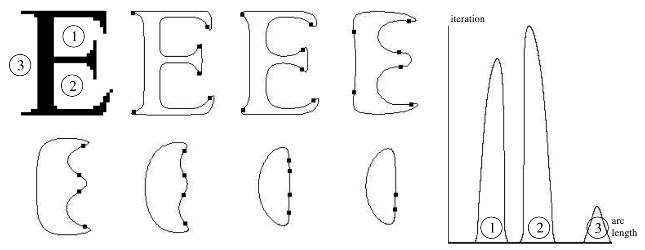

**Fig. 1**. Original character and smoothed shapes with inflection points after 5, 20, 100, 250, 500, 1000 and 1150 iterations. The corresponding CSS image is depicted on the right side. Three major concave segments are labeled.

To create a *mapped shape* [5, 6], we enclose the shape of the character by a circle of radius $R$ and identify the point $P$ of the circle closest to each shape pixel. The shape pixels are mirrored on the tangent of the circle in $P$. Two mapped shapes are depicted in Figure 2. Segments of the shape that have a strong convex curvature are mapped to concave segments. The calculation of the mapped shape is quite fast. Each shape pixel $(x(u), y(u))$ of the closed planar curve is mapped to the new position $(x'(u), y'(u))$. The center of the circle $(M_x, M_y)$ with radius $R$ is the average position of shape pixels.

$$D_{x(u),y(u)} = \sqrt{(M_x - x(u))^2 + (M_y - y(u))^2} \quad (1)$$

$$x'(u) = (x(u) - M_x) \cdot \frac{2 \cdot R - D_{x(u),y(u)}}{D_{x(u),y(u)}} + M_x \quad (2)$$

$$y'(u) = (y(u) - M_y) \cdot \frac{2 \cdot R - D_{x(u),y(u)}}{D_{x(u),y(u)}} + M_y \quad (3)$$

$D_{x(u),y(u)}$ specifies the distance between the center of the circle and the current shape pixel. If the positions of a shape pixel and the center of the circle are the same, a mapping is not possible. In this case, the shape pixel is interpolated from adjacent shape pixels of the mapped shape.

In principle, the mirroring of shapes is not limited to enclosing circles. Although other shapes could be used as well, some difficulties would arise. Angular shapes like rectangles would create discontinuous shapes. Ellipses have the disadvantage that the point $P$ (where the shape pixel is mirrored) is not always unique. E.g., in the case of ellipses that are parallel to the X- and Y-axis, the mirroring is undefined for all points on these axes.

We apply the standard CSS approach to the mapped shape. To indicate the classification of convex segments in the original shape we represent this new CSS image with negative values. In Figure 3 extended CSS images of four characters are depicted. Positive values represent the original CSS images, negative values the CSS images of the mapped shapes.

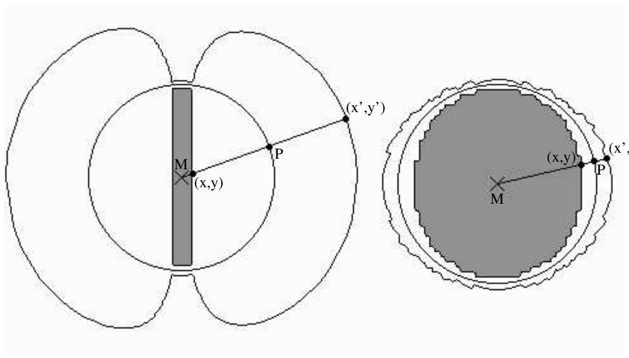

**Fig. 2**. The shapes of two characters (gray color) are "mirrored" at the circle.

The convex characters "I" and "O" cannot be classified with the standard CSS approach, but the dual CSS representations differ significantly.

## 2.3. CSS Matching

For the matching of an unknown object it is sufficient to extract the significant maxima (above a certain noise level). The position on the shape and the value (iteration or Gaussian kernel width) are stored for each peak. These peaks characterize convex regions. The sampled shape pixels are transformed to the mapped (dual) shape, and a second CSS image is created. The mapped feature vectors are stored as negative values. An unknown object is matched by comparing the feature vectors (CSS peaks) to those of the objects that are stored in a database. The summarized Euclidean distances of the height and position of each peak defines the difference between the CSS images. It is not possible to match negative and positive CSS peaks (the concave segments in the original and mapped shape). Details of the matching algorithm of CSS images are published in [7].

## 3. EXPERIMENTAL RESULTS

We have implemented the extended curvatire scale space algorithm and focus in our evaluation on simple shapes (characters), because the classification of complex shapes generally works quite well with the standard curvature scale space approach [3, 5, 7]. Additionally, we illustrate the main concept of the automatic segmentation of superimposed text in images.

## 3.1. Segmentation of Characters

We assume that each text line contains at least several characters. To locate a text region in an image we use the techniques presented by Sato and Smith [8]: The idea is to identify regions with high contrast and sharp edges. If this region suf-

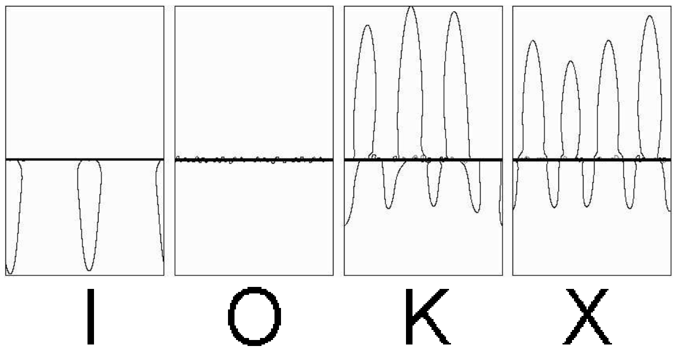

**Fig. 3**. Four examples of extended CSS images are depicted. Positive values represent the original CSS images, negative values the dual images.

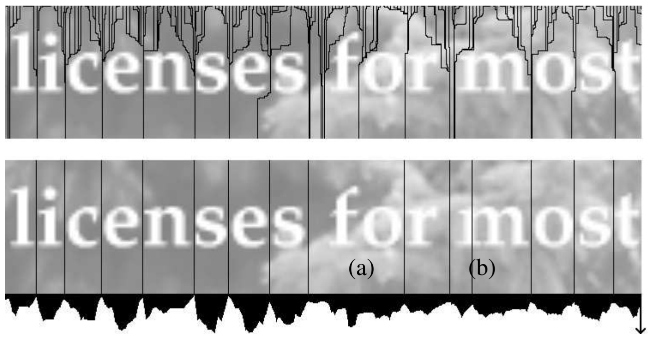

**Fig. 4**. Top: Character separators based on cheapest paths. Bottom: Vertical projection profile with (a) missed separators and (b) split characters.

fices certain constrains like minimum size, the bounding box of this region is classified as text region. Figure 6 displays an example of the detected bounding boxes in an image with a complex background.

Each pixel in each text region is classified as text or background pixel. The distinction is not trivial because the luminance and chrominance values of text pixels of one character can vary significantly. The separation of characters does not work very well with vertical projection profiles that summarize edge values for each column of a text line (see bottom of Figure 4). Many characters are split and separators are missed.

Usually, the contrast between text pixels and background pixels is high, whereas the average difference between adjacent background pixels is much lower. We take advantage of this fact and search a path from the top to the bottom of the text region. Different starting positions in the top row are selected, and the paths with the lowest costs are stored. The costs of a path are defined as the summarized pixel differences between adjacent path pixels. The path with the minimum cost which we call *cheapest path* rarely crosses character pixels and defines a good separator for characters.

We use the *Dijkstra* shortest-path algorithm for graphs to identify the separators. Results of the minimum paths (cheapest paths) are depicted in the top of Figure 4. A major advantage of this approach is the fact that no threshold is required to locate the separators of characters. In the final step of the segmentation, we use a region-growing algorithm to classify pixels as text or background. An example of the final segmentation of characters is depicted in Figure 6.

### 3.2. Classification of Characters

We have implemented a simple pattern matching approach and used a commercial OCR software that is part of a scanner software to evaluate the recognition rates. The CSS peaks of characters of four fonts are stored in a database. Figure 5 depicts sample characters of different fonts.

We have analyzed the recognition results for images and videos. Twenty images with complex backgrounds and ten video segments from different genres with a total length of 20 minutes were selected. A reliable segmentation is quite challenging. We define that a character is correctly segmented if it is not split or merged with other characters. We have analyzed the quality of the segmentation of characters by comparing projection profiles and the optimum path approach. The results in Table 1 (top) indicate that the optimum path algorithm is much more reliable (errors rates drop from 17.4 to 9.2 percent).

| Segmentation errors | Projection Profile | Optimum Path |
|---|---|---|
| Characters split | 9.9 % | 3.8 % |
| Characters merged | 7.5 % | 5.4 % |
| Correctly separated characters | 82.6 % | 90.8 % |

| Recognition results | Images | Video sequences |
|---|---|---|
| Number of characters | 2986 | 1211 |
| Pattern matching | 64.5 % | 72.1 % |
| Standard CSS | 62.7 % | 73.1 % |
| Extended CSS | 70.8 % | 77.2 % |
| Commercial OCR | 70.5 % | 72.4 % |

**Table 1**. Top: Reliability of segmentation based on projection profiles and optimum path. Bottom: Aggregated recognition results.

To calculate the overall recognition rate of characters we analyzed all segmented objects (even artifacts and merged or split characters). Both CSS methods rejected many poor segmented characters. Table 1 (bottom) lists the recognition results for images and video sequences. The results in videos are usually much better due to the additional preprocessing of the frames.

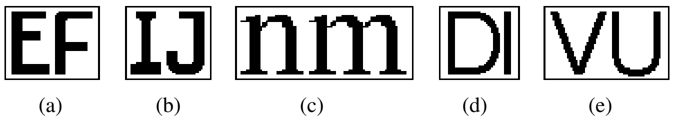

**Fig. 5**. Two examples of the European license plate font (a,b) illustrate the large minimum distance of the pattern matching. The distance is very low in other fonts (c). The standard CSS approach cannot characterize convex characters (d) and even some characters have very similar CSS peaks in the extended approach (e).

The commercial OCR system could not recognize any characters in the original images, so we used the segmented binary images for classification. A direct comparison of the recognition rates is still not possible due to the dictionary lookup in the commercial system. The quality of the segmentation is higher in video sequences, but the commercial OCR systems cannot benefit that much. We assume that the dictionary lookup is less efficient with text in videos.

To sum up, the classification results of shapes with only a few concave segments are very poor with the standard CSS approach. The extension using mapped shapes increases the classification rates significantly and enables the recognition of convex shapes. Figure 6 depicts the major segmentation steps in an image with a complex background. The image includes characters with different fonts and sizes.

## 4. CONCLUSION AND OUTLOOK

We have analyzed the curvature scale space technique and presented major deficiencies of the standard approach. Although it works quite well to describe complex shapes like leaves or an automatically segmented person, the disadvantage of this approach becomes more obvious with the analysis of simpler shapes: the poor representation of convex segments of a shape.

Our extension of the CSS method classifies concave and convex segments of a shape and proves to be very powerful for the recognition of complex *and* simple shapes. We have evaluated as a comprehensive example the automatic recognition of characters in images and videos. As future work, we plan to evaluate the recognition results of the extended CSS method for deformable non-text objects.

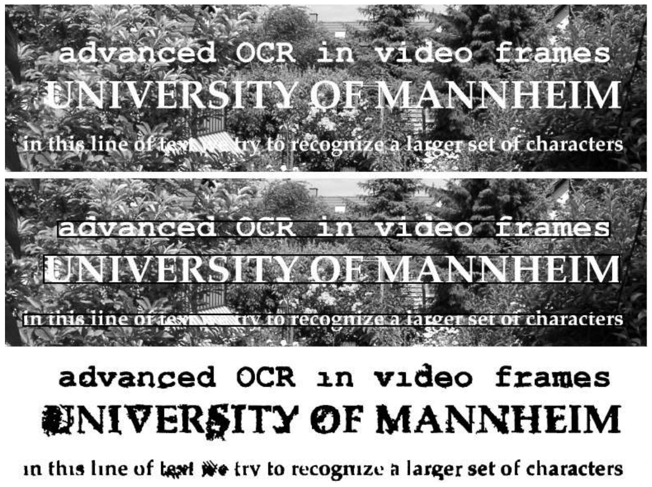
advanced OCR in video frames
UNIVERSITY OF MANNHEIM
in this line of text we try to recognize a larger set of characters

**Fig. 6**. Original image (top), automatically detected text regions (center) and segmented text (bottom)

## 5. REFERENCES

[1] Luciano da Fontoura Costa and Roberto Marcondes Cesar, Jr., *Shape Analysis and Classification*, CRC Press, Boca Raton, FL, September 2000.

[2] Sven Loncaric, “A survey of shape analysis techniques,” in *Pattern Recognition*, August 1998, vol. 31(8), pp. 983–1001.

[3] Farzin Mokhtarian, “Silhouette-based isolated object recognition through curvature scale space,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995, vol. 17(5), pp. 539–544.

[4] Farzin Mokhtarian and Miroslaw Bober, *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization (Computational Imaging and Vision, 25)*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003.

[5] Stephan Kopf, Thomas Haenselmann, and Wolfgang Effelsberg, “Shape-based posture and gesture recognition in videos,” in *Electronic Imaging*. January 2005, vol. 5682, pp. 114–124, IS&T, SPIE.

[6] Stephan Kopf, Thomas Haenselmann, and Wolfgang Effelsberg, “Robust character recognition in low-resolution images and videos,” Tech. Rep. TR-05-002, Dept. of Computer Science, University of Mannheim, Mannheim, Germany, 2005.

[7] Stephan Richter, Gerald Kühne, and Oliver Schuster, “Contour-based classification of video objects,” in *Proceedings of IS&T/SPIE conference on Storage and Retrieval for Media Databases*, January 2001, vol. 4315, pp. 608–618.

[8] Toshio Sato, Takeo Kanade, Ellen K. Hughes, and Michael A. Smith, “Video OCR for digital news archives,” in *IEEE International Workshop on Content-Based Access of Image and Video Databases (CAIVD)*, 1998, pp. 52–60.