UNIVERSITÄT PADERBORN

FAKULTÄT FÜR ELEKTROTECHNIK INFORMATIK UND MATHEMATIK

# Local Scale-Invariant Contour Features for Object Recognition

Von der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Ing. Markus Hennig

| | |
|---|---|
| Erster Gutachter: | Prof. Dr. Erdal Kayacan |
| Zweiter Gutachter: | Prof. Dr. Reinhold Häb-Umbach |
| Tag der mündlichen Prüfung: | 29.10.2025 |

Paderborn 2025

Diss. EIM-E/390

**Markus Hennig**
*Local Scale-Invariant Contour Features*
*for Object Recognition*
Ph.D. Thesis, October 2025

**Paderborn University**
*GET Lab*
Department of Electrical Engineering and Information Technology
Faculty of Computer Science, Electrical Engineering and Mathematics
Warburger Str. 100
33098 Paderborn, Germany

All content written by the author. The following tools were selectively used to support linguistic clarity and precise technical terminology: DeepL Pro, ChatGPT-4o, Grammarly.

Figures 2.2b and 2.2c reproduced with permission from Elsevier and SAGE Publications.

# Abstract

Object recognition is one of the most important problems in computer vision and can be addressed using various techniques and functional subcomponents, including different preprocessing steps, feature types, and learning and classification schemes, depending on the specific task and operational conditions. Both traditional and deep learning-based methods require stable features that are sufficiently descriptive and distinctive to recognize and distinguish different objects or classes. In this work, two new methods to extract such features are presented: the first method determines local scale-invariant contour features around curvature extrema, while the second method introduces an ambiguity model with edge tracing to extract these features from binary edge images. At the end of this work, it is proposed to integrate the methods into a deep learning-based end-to-end feature detection approach.

Besides technical rationales, the contour features presented in this work are inspired by insights from human vision research. In particular, curvature extrema are the most informative points along contours and highly salient in human vision. A unique aspect of the features is that they are assigned a characteristic scale. While this is a widely adopted approach for appearance-based features, there is no robust methodology for assigning characteristic scales to curvature extrema. The features are extracted using curvature scale-space analysis, which provides a formalized bottom-up framework for this task. Computational experiments demonstrate that the contour features can be reliably detected even under extreme scale changes, noise, and partial occlusion. In the context of this method, box filter approximations are analyzed, and a selected approach is integrated to achieve real-time capability. Furthermore, a new padding method is presented to process open contours.

Binary edge images often include intersections, junctions, and other structures that make it difficult to extract coherent object contours. Nevertheless, such contours are required for extracting the contour features presented in this work and other methods. The ambiguity model presented in this work is designed to describe and resolve such ambiguities. Despite using only four straightforward principles, the model can handle complex structures in binary edge images in an intuitive and effective manner. Compared to existing methods, the model provides the most detailed decomposition of binary edge images into meaningful segments while also reducing redundancy (double reading of edge pixels). It is shown that the method can effectively resolve complex ambiguities in different application examples.

# Zusammenfassung

Die Objekterkennung ist eines der wichtigsten Probleme der digitalen Bildverarbeitung und kann mit verschiedenen Ansätzen und funktionalen Teilkomponenten adressiert werden. Dies umfasst verschiedene Vorverarbeitungsschritte, Merkmalstypen sowie Lern- und Klassifizierungsmethoden, abhängig von der spezifischen Aufgabe und den Einsatzbedingungen. Sowohl traditionelle als auch Deep-Learning-basierte Ansätze erfordern robuste Merkmale, die hinreichend deskriptiv und distinktiv sind, um Objekte oder Klassen zu erkennen und zu unterscheiden. In dieser Arbeit werden zwei neue Methoden zur Extraktion solcher Merkmale vorgestellt: Die erste bestimmt lokale, skalierungsinvariante Konturmerkmale um Krümmungsextrema. Die zweite stellt ein Mehrdeutigkeitsmodell mit Kantenverfolgung dar, um die Merkmale aus binären Kantenbildern zu extrahieren. Abschließend wird vorgeschlagen, die Methoden in einen Deep-Learning-basierten End-to-End-Ansatz zur Merkmalsdetektion zu integrieren.

Neben technischen Überlegungen sind die Konturmerkmale von Erkenntnissen zur menschlichen Wahrnehmung motiviert. Insbesondere weisen Krümmungsextrema den größten Informationsgehalt entlang von Objektkonturen und eine hohe Salienz in der menschlichen Wahrnehmung auf. Eine Besonderheit der Merkmale ist die Zuordnung einer charakteristischen Skalierung. Während dies für ansichtenbasierte Merkmale verbreitet ist, fehlt eine robuste Methodik für Krümmungsextrema. Zur Extraktion wird ein skalenraumtheoretischer Ansatz verwendet, der dazu ein formalisiertes Bottom-Up-Framework bietet. Es wird gezeigt, dass die Konturmerkmale auch bei starken Größenänderungen, Rauschen und teilweiser Überdeckung zuverlässig extrahiert werden können. Außerdem werden Boxfilter-Ansätze analysiert und integriert, um Echtzeitfähigkeit zu erzielen, sowie eine neue Padding-Methode für offene Konturen vorgestellt.

Binäre Kantenbilder enthalten oft Schnittpunkte, Kreuzungen und andere Strukturen, die die Extraktion kohärenter Objektkonturen erschweren. Diese sind jedoch zur Extraktion der Konturmerkmale sowie für andere Methoden erforderlich. Das in dieser Arbeit vorgestellte Mehrdeutigkeitsmodell wurde entwickelt, um solche Mehrdeutigkeiten zu beschreiben und aufzulösen. Obwohl das Modell nur vier einfache Prinzipien verwendet, kann es komplexe Strukturen in binären Kantenbildern auf intuitive und effektive Weise verarbeiten. Im Vergleich zu bestehenden Methoden bietet das Modell die differenzierteste Segmentierung und reduziert gleichzeitig Redundanzen (doppeltes Einlesen von Kantenpixeln). Es wird gezeigt, dass die Methode komplexe Mehrdeutigkeiten in verschiedenen Anwendungsbeispielen effektiv auflösen kann.

# Contents

# 1 Introduction

Object recognition in digital images has been an important research area for decades and is still a central problem in computer vision. It can be addressed using various techniques and functional subcomponents, including different preprocessing steps, feature types, and learning and classification schemes, especially depending on the specific task and operational conditions (see, e.g., the overviews by Amit et al., 2021; Zou et al., 2023; Kaur and W. Singh, 2024). Some of the most important application domains are robotics, autonomous driving, surveillance, and medical image analysis. In robotics, for example, object recognition is required for autonomous navigation and object manipulation, and is therefore an essential component for interactions with the environment (Soori et al., 2023). Similarly, road objects have to be reliably detected in autonomous driving, with specific challenges in accuracy, interpretability, inference time, and sensor data fusion (Zhao et al., 2024). Recognition capabilities have significantly improved with deep learning techniques, especially convolutional neural networks (CNNs). At the core of nearly any object recognition method is a feature extraction stage. The characteristics of the features employed—their type, repeatability, precision, and discriminative capabilities—significantly contribute to the recognition performance. Both traditional and deep learning-based methods require stable features that are sufficiently descriptive and distinctive to recognize and distinguish different objects or classes. While traditional methods use manually engineered features, deep learning-based methods generally learn features automatically.

In this work, a new method to determine local scale-invariant contour features is presented, which fall into the category of manually engineered shape features. Additionally, a new general and intuitive ambiguity model for binary edge images is presented. The model can be used to extract coherent object contours to determine the contour features. While global features generally lead to more compact object representations, local features can better cope with partial occlusion, viewpoint changes, or deformable objects due to their mutual independence and localized nature. The ambiguity model is particularly promising for use in connection with current edge and object contour detection methods, where deep learning-based methods achieve the best results (Jing et al., 2022; D. Yang et al., 2022). While the contour features provide some attractive properties, particularly being robust, meaningful, and interpretable, their extraction requires several preprocessing steps. Here is another connection to modern deep learning-based methods: local features can serve as anchors for supervised training in end-to-end feature detection approaches (e.g., Ma et al., 2021). To substitute the preprocessing steps, it is proposed to use the methods developed in this work to generate training data for a deep learning-based end-to-end feature detection approach.

Object contours are interesting to analyze because they provide highly descriptive yet compact object representations and therefore have a long history in object recognition methods (Andreopoulos and Tsotsos, 2013). In digital images, object contours can be detected even in case of challenging illumination conditions and complex object textures, especially when using learning instead of traditional gradient-based edge detection methods (e.g., Pu et al., 2022). In addition to technical rationales, the presented contour features are motivated by insights from human vision research, where the role of shape information is extensively studied (e.g., M. Singh, 2015; Bracci and Op de Beeck, 2023). As pointed out in (Baker et al., 2018) based on established literature on the subject (see references therein and Section 2.1.1), shape is generally the most important cue for recognizing objects in human vision. In particular, objects can be accurately recognized even when other visual dimensions are removed. In this context, the focus of this work is on the role of curvature extrema of 2D object contours, which are used as keypoints to determine the local contour features.

## 1.1 Objectives and Contributions

The main objective of this work is to develop a method for extracting local scale-invariant contour features at curvature extrema to complement appearance-based features in a modular manner. While it is common to assign characteristic scales to local appearance-based features, there is no robust methodology for assigning characteristic scales to local contour features. Yet, it is well-known that shape information—which is encoded in contours—is generally the most important cue for recognizing objects in human vision, and that curvature extrema are the most informative points along object contours. The following two methods are parameter-free and could therefore be used in a wide range of applications.

The contour features are extracted using curvature scale-space (CSS) analysis, which provides a formalized bottom-up framework for this task. While most existing methods based on this framework are global, the method presented in this work is a local approach that assigns characteristic scales to curvature extrema. Furthermore, it supports both closed and open contours. The key idea in this work is to identify and distinguish local and global characteristics in the signature functions of curvature extrema within the CSS representation—an aspect that has not been explored in the literature. To process open contours, their evolution should approximate that of the original closed contour when computing the CSS representation. To this end, a new padding method is presented. Since the convolutions required to compute the CSS representation are computationally expensive, a box filter approximation is integrated to enable real-time capability. Such approximations have not been analyzed in the context of CSS computation.

The method developed in this work to extract the contour features requires traced object contours without ambiguities—where ambiguities refer to structures where the path of a contour is not clearly defined, such as at intersections or junctions. To handle such cases, a new ambiguity model is presented, which traces edge pixels in an ordered sequence and is specifically designed to describe and resolve such ambiguities. Compared to existing methods, the model provides the most detailed decomposition of binary edge images into meaningful segments while also reducing redundancy (double reading of edge pixels). Despite using only four straightforward principles, the model can handle complex structures in binary edge images in an intuitive and effective manner.

## 1.2 Thesis Structure

In Chapter 2, background information on object recognition in the human visual system and artificial vision systems is provided in relation to the methods developed in this work. In Chapter 3, the developed ambiguity model for binary edge images is presented, and a detailed comparison with existing methods is provided. In Chapter 4, the local scale-invariant contour features are introduced, based on a detailed analysis of curvature extrema signature functions. In Chapter 5, the main achievements are summarized and a potential integration of the developed methods into a deep learning-based end-to-end feature detection approach is outlined.

# 2 Background and Motivation

This chapter provides background information on object recognition in the human visual system and artificial vision systems in relation to the methods developed in this work. Section 2.1 focuses on the human visual system, while Section 2.2 focuses on artificial vision systems.

Human visual system: In Section 2.1.1, the relevance of contours, shape, and other features is discussed, which serve as building blocks for object recognition in a hierarchical processing scheme. In Section 2.1.2, the role of curvature extrema is described, which are the most informative points along object contours. In Section 2.1.3, the importance of processing visual information at multiple scales is outlined.

Artificial vision systems: In Section 2.2.1, local and global features are characterized from a general perspective, followed by a description of different types of local image features. In Section 2.2.2, the general relevance and properties of local invariant features are discussed. In Section 2.2.3, the concept of characteristic scales is reviewed, which are employed for invariant feature description. In Section 2.2.4, important local feature detection methods such as SIFT, SURF, and ORB are described. Finally, region- and edge-based image segmentation methods are discussed in Sections 2.2.5 and 2.2.6, where the results of these methods can be processed using the methods developed in this work.

Preliminary considerations on the aspects discussed in this chapter have been published in (Hennig and Mertsching, 2016).

## 2.1 Human Vision and Object Recognition

Motivated by the capabilities of human vision, object recognition methods in computer vision are often inspired by insights into the mechanisms of the human visual system (Poggio and Ullman, 2013). This is also the case for the contour features presented in this work. Such mechanisms are extensively explored in different disciplines, including cognitive psychology, neuroscience, and computer vision, with dissolving boundaries between the fields (DiCarlo et al., 2012). The numerous works reflected in that review, as well as more recent reviews (e.g., X. Yang et al., 2022; Bracci and Op de Beeck, 2023), show that while there is no universal model for human vision, there are profound insights into its complex and complementary subprocesses.

Two foundational object recognition frameworks developed in cognitive psychology are the Recognition-by-Components model by Biederman (1987), where objects are represented as structural models based on 3D geometric shape primitives ("geons"), and the view-based model by Bülthoff and Edelman (1992), where objects are represented as sets of 2D views. While structural information is definitely involved, the dominant mechanisms for object recognition are view-based (Tarr and Bülthoff, 1998; Andreopoulos and Tsotsos, 2013). As the contour features presented in this work are extracted from images, they are also more in line with the view-based model. As human vision accomplishes a range of tasks beyond object recognition (e.g., active object observation, object tracking, obstacle avoidance) and is influenced by pre-cuings, top-down knowledge, context, etc., it is often broken down into the problem of *core object recognition*, the ability to rapidly recognize isolated objects at the category level despite variations in appearance (DiCarlo et al., 2012; Wichmann and Geirhos, 2023). The field of neuroscience provides insights into the corresponding processes in the brain through the analysis of neuronal activity, using methods like Functional Magnetic Resonance Imaging (fMRI) (e.g., Ayzenberg and Behrmann, 2022b, and the references therein). For example, it is well known that visual information is processed in two main neural pathways: the ventral pathway (Lateral geniculate nucleus (LGN), V1, V2, V4, Inferior temporal cortex (IT)) for object recognition ("what"), and the dorsal pathway for locating and manipulating objects ("where/how"). Core object recognition is essentially solved by fast feedforward computations in the ventral pathway (DiCarlo et al., 2012). Another important finding is that visual information for object recognition is processed in a hierarchical manner, starting with basic features in V1, such as edges of different orientations, contrasts, spatial frequencies, and colors, and then gradually reaching object representations in the IT. In line with that, the contour features presented in this work are also intended to be used as intermediate elements in hierarchical object recognition schemes. The field of computer vision provides additional insights: the idea is that if a model works well in this field, then it could also explain processes involved in human vision. For example, one of the research questions currently investigated in various studies is whether Deep neural networks (DNNs) are adequate models of human core object recognition. While promising, they do not seem to be completely sufficient (Wichmann and Geirhos, 2023).

To further motivate the development of the contour features, some more specific aspects of human object recognition are outlined in the following sections: the role of contours, which represent the outline of objects and provide shape information; the role of curvature extrema, which highlight significant points along contours; and the importance of scale, which corresponds to different levels of abstraction.

### 2.1.1 Contours, Shape, and Other Features

As explained by DiCarlo et al. (2012), every retinal image of an object is almost unique due to varying object positions relative to the observer, changing lighting conditions, object deformations, and diverse visual contexts. Nonetheless, objects can be reliably recognized and distinguished, which therefore requires identity-preserving features. Particularly important in this regard are object contours, as their structural and relational properties in images are relatively invariant to changes in position, scale, and rotation. Furthermore, object contours represent the outline (boundaries) of objects and therefore provide fundamental information about their shape and structure. As pointed out by M. Singh (2015), they also indicate some physically significant phenomena, corresponding to concentrated regions of information. It is therefore not surprising that contours have already been one of the main elements in early computational models of vision (e.g., Marr, 1982).

As outlined above, the human visual system processes information in a hierarchical manner. Regarding object contours, the process starts with edge detection in V1 (with major breakthroughs in understanding this process by Hubel and Wiesel, summarized in their retrospective, 2004). An inherent strength of edges is their robustness due to their relative invariance to changing lighting conditions, textures, and colors. The detected edges are then integrated into contours and further processed in higher visual areas like the IT (Loffler, 2008). As summarized by M. Singh (2015), the process is based on parts and their spatial relationships, where the parts are segmented according to systematic and predictable rules. These parts often correspond to psychologically meaningful subunits of objects (e.g., head, leg, branch), which are important for object recognition and other tasks.

There are many additional processes that are an integral part of human vision, including segmentation, perceptual grouping, interpolation, extrapolation, and attention (e.g., Dickinson and Pizlo, 2013; Wolfe et al., 2024). Furthermore, shape representations in human vision are not only contour-based but also region-based (M. Singh, 2015). However, the intention of the remainder of this section is to motivate the relevance of contours and shape from a more general perspective.

Consider Figure 2.1: Although the patterns have different colors, textures, and substructures, with no specific relation to the shapes, and the physical scales (sizes) of the

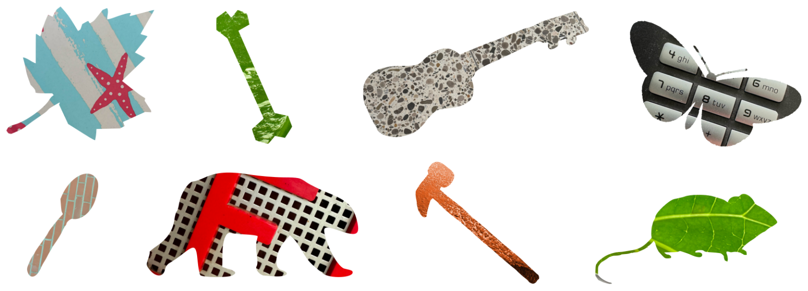


Fig. 2.1: Shapes from different datasets filled with arbitrary patterns. Despite the patterns and varying physical scales, the objects can be clearly recognized, which shows the importance of shape over other features. Idea based on (Baker et al., 2018).

objects are quite different, the objects can still be clearly recognized. While this shows the importance of shape over other features, the recognition capabilities of human vision based on shapes and contours have also been systematically investigated in numerous studies. For example, Biederman and Ju (1988) conducted an experiment where participants had to identify objects presented in two different formats: professionally photographed full-color images and simplified line drawings. The overall mean reaction times and error rates were nearly identical for both types of stimuli. Based on that, they concluded that while "color, brightness, and texture can be instrumental in defining edges and can provide cues for visual search, they play only a secondary role in the real-time recognition of an intact object when its edges can be readily extracted." In another study by Cole et al. (2009), participants used a computer-aided system to assign the orientation of normals at various positions on the surfaces of 3D objects. The objects were presented in six different styles: one shaded image and five different types of line drawings. In this case, the authors concluded that line drawings are almost as accurate as shaded images in depicting the 3D shape of objects, although not all line drawing styles are equally effective. In a study conducted by Walther et al. (2011), fMRI data was collected while participants "viewed photographs and line drawings of beaches, city streets, forests, highways, mountains, and offices." In this case, the authors were able to decode the scene categories from the fMRI data for the line drawings with the same accuracy as for the color photographs. The authors concluded that the information used in specific brain regions to distinguish scene categories is similar for line drawings and color photographs. Further studies investigating the importance of contours and shape in human vision are discussed in (Ayzenberg and Behrmann, 2022a). An important conclusion from that and other works is that shape representations of objects in the ventral pathway "may be [...] described as a basis set of local image features," which is in line with the contour features presented in this work, as they are also local features.

While shape is important, there are several other important features for object recognition in human vision. As summarized in (Ge et al., 2022), the most important features besides shape are color and texture, and their roles in human vision have also been extensively studied in various works. This is reasonable, as these features are essential to distinguish certain objects. An example is also given in that work: To distinguish a zebra from a horse, texture appears to be more important than shape, as their outlines are quite similar. On the other hand, to distinguish a safari car with a zebra pattern from an actual zebra, shape appears to be more important. Various studies on the role of texture in human vision are discussed in (Pasupathy et al., 2019). Similar examples can be found for color: For instance, to distinguish lemons and oranges viewed in cross-section, color appears to be more important than shape and texture. Various studies on the role of color in human vision are discussed in (Bramão et al., 2011). In summary, effective object recognition requires the integration of different feature types. In computer vision, local image features are often appearance-based (texture-based) and associated with a characteristic scale (cf. Section 2.2.3). The objective for the contour features presented in this work is therefore to assign them a characteristic scale so that they can be integrated with appearance-based features in a modular fashion.

### 2.1.2 Contour Curvature Extrema

The contour features presented in this work are located at curvature extrema of object contours, motivated by different considerations. As summarized in (M. Singh, 2015), curvature extrema are the most informative points along contours and also play a significant role in human vision. Furthermore, they can be reliably detected and localized, and their information content makes them more distinctive than other points, which are important properties of local image features in computer vision, as discussed in Section 2.2.2.

One of the first works emphasizing the role of curvature extrema in human vision was presented by Attneave (1954). In one of his experiments, participants were instructed to resemble the shape of given object contours using 10 dots and then to indicate their positions along the contours. The result was that “most of these points [were] taken from regions where the contour is most different from a straight line,” corresponding to curvature extrema. Based on this observation, the author concluded that common objects can be efficiently represented by connecting curvature extrema with straight lines. To demonstrate that, he provided a drawing now famously known as Attneave’s Cat, which is shown in Figure 2.2a. Furthermore, the author pointed out the direct relation of such questions to information theory (details below). To verify the results, the experiment with the dots was later repeated in a very similar form by Norman et al. (2001). While Attneave did not describe the exact construction method of his test contours, Norman et al. used the shadows (outlines) of natural objects (sweet potatoes) and indeed obtained similar results. An example is shown in Figure 2.2b. The lines represent histograms, where the bar lengths correspond to the number of participants who placed points at the respective positions. In a large-scale study by De Winter and Wagemans (2008), participants were asked to mark an arbitrary number of salient points along the contours of everyday objects. Also in this case, the authors found that these points "are usually very close to strong curvature extrema."

Another aspect discussed in (M. Singh, 2015) is that in the case of closed contours, curvature extrema can be assigned a distinct sign: positive for convex segments (curvature maxima) and negative for concave segments (curvature minima). In the case of open contours, the distinction between inside and outside is unclear so that the sign is ambiguous. Curvature minima are particularly interesting because they are important reference points for part segmentation, i.e., they are often located at the transitions between individual object parts. For example, curvature minima separate the fingers of a hand, the branches from the trunk of a tree, or the fins of a fish. An example for the latter is shown in Figure 2.2c. In the corresponding study by De Winter and Wagemans (2006), participants were instructed to segment the outlines of everyday objects into salient or important parts using segmentation lines. As summarized in the study, the corresponding segmentation principles in human vision have been formalized in the literature as specific rules, such as the minima rule and short-cut rule. The contour features presented in this work are also assigned the sign of curvature to obtain a richer description and to potentially consider such principles in specific applications.

As already implied above and elaborated in (Feldman and M. Singh, 2005), it is consistent with information theory that curvature extrema are the most informative points along contours. From this perspective, the information encoded along contours increases with larger turning angles and, as a result, with higher curvature (the turning angle is the discrete counterpart of curvature). The connection is straightforward: from a probabilistic point of view, a contour is most likely to continue in the direction of the turning angle, with monotonically decreasing probability as the deviation from that direction increases. In other words, larger deviations are more surprising and therefore carry more information. If the probability that a turning angle $\alpha$ is observed is denoted $P(\alpha)$, then the information content (or surprise) $H$ associated with $\alpha$ can be computed using the following standard equation from Shannon (Feldman and M. Singh, 2005):

$$H(\alpha) = -\ln(P(\alpha))\,. \tag{2.1}$$

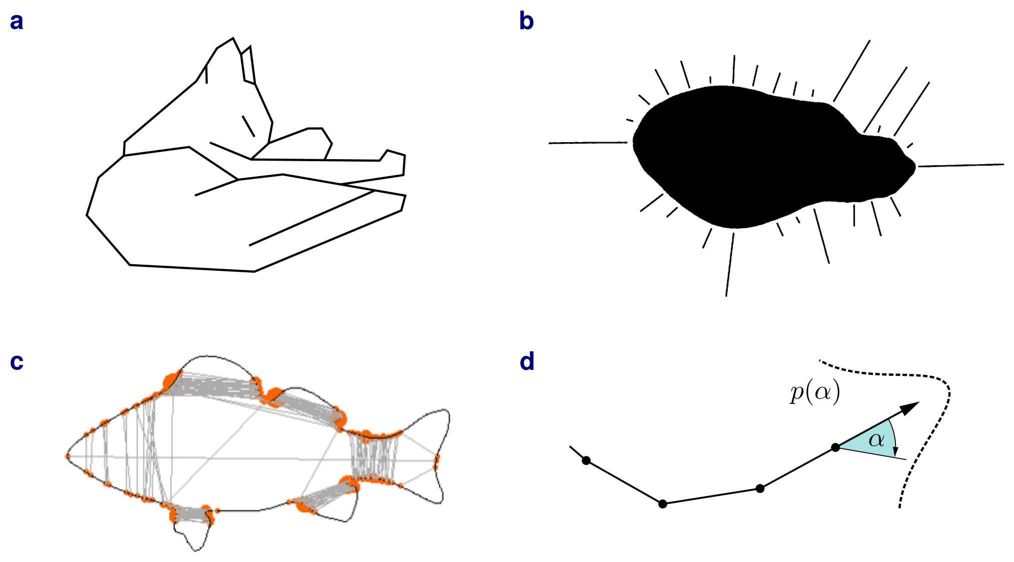


**Fig. 2.2:** Examples illustrating the importance of contour curvature extrema in human vision. **a** Attneave's cat, redrawn based on (Attneave, 1954). **b** Histogram representing positions where participants placed points to reproduce the given shape, reproduced from (Norman et al., 2001). **c** Lines drawn by participants to segment the given contour, reproduced from (De Winter and Wagemans, 2006). **d** Conceptual visualization of a probabilistic contour continuation model, idea based on (M. Singh, 2015).

As also described in that work and supported by experimental findings with participants, the underlying continuous distribution $p(\alpha)$ describing the expectations about how a contour continues can be modeled as a von Mises distribution centered at $\alpha = 0$ (straight continuation). A conceptual visualization is shown in Figure 2.2d. The von Mises distribution is essentially the circular counterpart of the Gaussian normal distribution, with its support over the angles $(-\pi, \pi)$ instead of $(-\infty, \infty)$. Note that these observations align with the principle of good continuity, a key concept in perceptual grouping, for example in Gestalt theory (Rock and Palmer, 1990). In this work, this principle is used to resolve ambiguities in binary edge images, as discussed in Section 3.3.6.

### 2.1.3 Importance of Scale

As already outlined, effective object recognition requires identity-preserving features, which is particularly important in case of scale changes. As discussed in greater detail in connection with scale-space theory in Section 4.1, the problem is twofold: First, real-world objects are built based on structures of different scales, and second, the size of an object in the image plane changes with the distance to the observer (zooming excluded). Nonetheless, structures at different scales can be observed simultaneously.

For example, from an appropriate distance to a tree, one can see both the treetop and individual leaves simultaneously. As a result, as Kuijper (2002) puts it (p. 19), “the eye and the system behind it [are] capable of working multi-scale.”

In fact, it has been found that the retina of the human eye is already structured to process visual information at multiple scales by using receptive fields of different sizes (Field and Chichilnisky, 2007; Lindeberg, 2021). The basic principle is that circular center-surround regions of photoreceptors (rods and cones) converge onto single ganglion cells through intermediate bipolar cells, so that information is collected over regions of different sizes in the visual field. Another important observation is that the spatial structure of receptive field responses can be modeled based on Gaussian derivative operators, particularly Difference of Gaussians (DoG) (Young, 1987), corresponding to Gaussian blurring combined with second-order derivatives. In comparison to first-order derivatives, second-order derivatives are isotropic. Furthermore, first-order derivatives are nonzero along intensity ramps, whereas second-order derivatives are only nonzero at the beginnings and ends of intensity ramps, making them more effective for enhancing edges and the overall sharpness (cf. Gonzalez and Woods, 2018). As discussed in Section 2.2.4, DoG operators are also used in computer vision to determine local image features. Regarding the contour features presented in this work, they are assigned scale information in the form of characteristic scales.

## 2.2 Local Image Features and Object Recognition

In traditional object recognition methods (as opposed to deep learning-based methods), one can generally distinguish between local and global features and corresponding object representations, which often provide complementary information (Grauman and Leibe, 2011; X. Li et al., 2013; Zou et al., 2023). Global representations are single descriptions of entire images or larger image regions, typically based on global statistical characteristics of appearance. Common representations are color or gray-level histograms and the gray-level co-occurrence matrix (cf. Khaldi et al., 2019). Similarly, features like Zernike moments, Fourier descriptors, and Hu moments can be used to describe the overall shape of an object or image region (Kurnianggoro et al., 2018).

Global features provide compact object representations that are particularly efficient for category-level object recognition. In contrast, object representations based on local features are more detailed, as individual part characteristics are explicitly described. As a result, local features are particularly effective for the recognition of specific objects (e.g., identifying a particular car, rather than just the general category *car*, cf. Grauman and Leibe, 2011). Furthermore, local features can better cope with partial occlusion, viewpoint changes, and deformations due to their mutual independence and localized nature. On the other hand, finding correspondences between individual features is generally more complex (feature matching, details below). However, such correspondences are required for tasks such as aligning images or tracking the motion of individual object parts (e.g., Gauglitz et al., 2011). Another characteristic difference is that when searching for an object in an image, global representations generally require search windows with many different sizes and aspect ratios (cf. Uijlings et al., 2013), whereas local feature detectors directly identify specific points or regions of interest. Concerning deep learning techniques, especially CNNs, their hierarchical structure aggregates small features into higher-level representations (Khan et al., 2020) and can therefore be seen as an approach that integrates both local and global features. One disadvantage of CNNs is that the specific type of features used and the integration process are difficult to control (cf. discussion in Section 5.1).

As summarized by Tuytelaars and Mikolajczyk (2008), one can distinguish three broad application areas for local features. First, specific local features can have a particular semantic meaning in certain applications; for example, specific image structures could indicate manufacturing defects in an inspection task. Second, local features can be used as anchor points to find well-localized correspondences between image structures for tasks such as tracking scene elements, camera calibration, or 3D reconstruction. Third, local features can be used to represent images for tasks such as object or scene recognition without additional segmentation steps. It is not always important what the local features actually represent, as long as they work for the specific application (such as being stable enough for tracking). The contour features presented in this work are not bound to a specific application. As already outlined, they are intended to be integrated with appearance-based features in a modular fashion.

### 2.2.1 Types of Local Image Features

Over the past decades, a wide range of methods has been developed to detect and describe local image features (Y. Li et al., 2015; Zou et al., 2023). Apart from edge detection and region segmentation methods (which are briefly discussed in Sections 2.2.5 and 2.2.6, respectively), one can generally distinguish between corner and blob detection methods. These methods can be further categorized depending on their general methodology, as illustrated in Figure 2.3. The methods listed at the bottom have been selected as examples because they are widely used in the literature, to illustrate important properties of different methods, and to set the contour features presented in this work in further context. Such contour features are typically considered corner features,

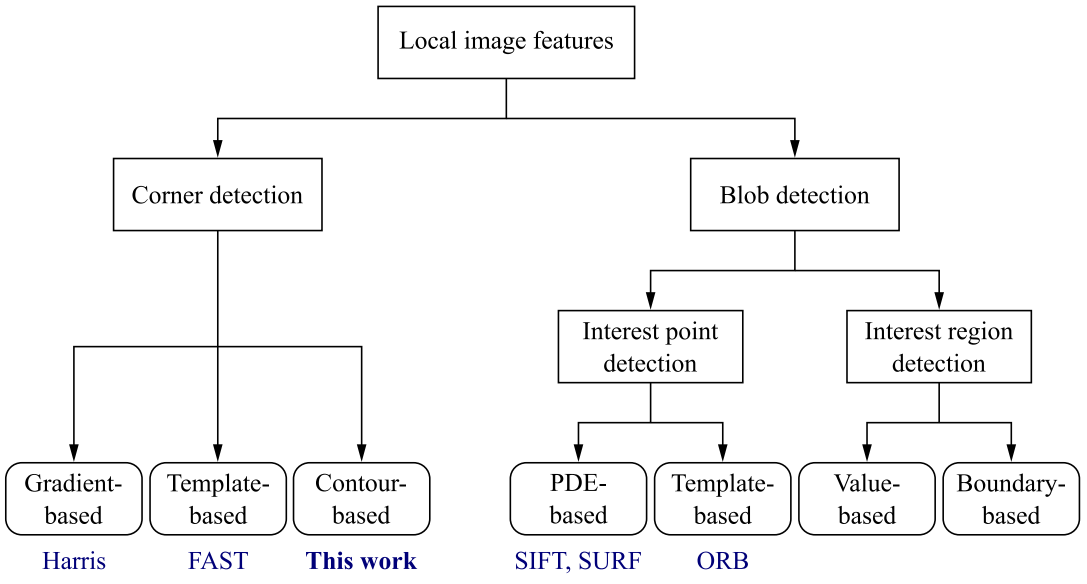


**Fig. 2.3:** Categorization of local image features, adapted based on (Y. Li et al., 2015). Acronyms and details are described in the text. The methods listed at the bottom are only some examples that are widely used in the literature.

although this does not imply that the corresponding curvature extrema must be particularly sharp. Rather, it indicates that their curvature is sufficiently high relative to their surroundings. In this perspective, corners can be defined as points with two different and significant gradient directions in gray-value images, or as points along contours where the curvature reaches a local minimum or maximum. Blobs can be defined as regions with a regular shape in which the pixels have consistent characteristics, such as in circular center-surround regions, as shown in Figure 2.5.

Feature detection methods often work on gray-value images, such as the Harris corner detector, while others first require the determination of contours, such as the method presented in this work. Gray-value images are used because they simplify the data, reduce computational complexity, and are generally more robust than color information. The Harris corner detector computes a matrix from the weighted sum of the structure matrix (second-moment matrix) over small windows of the image. The corner response is determined based on the eigenvalues of the resulting matrix, and a point is considered a corner if the response exceeds a certain threshold (Harris and Stephens, 1988). An example image with Harris corners is shown in Figure 2.8. The other methods are described in Section 2.2.4.

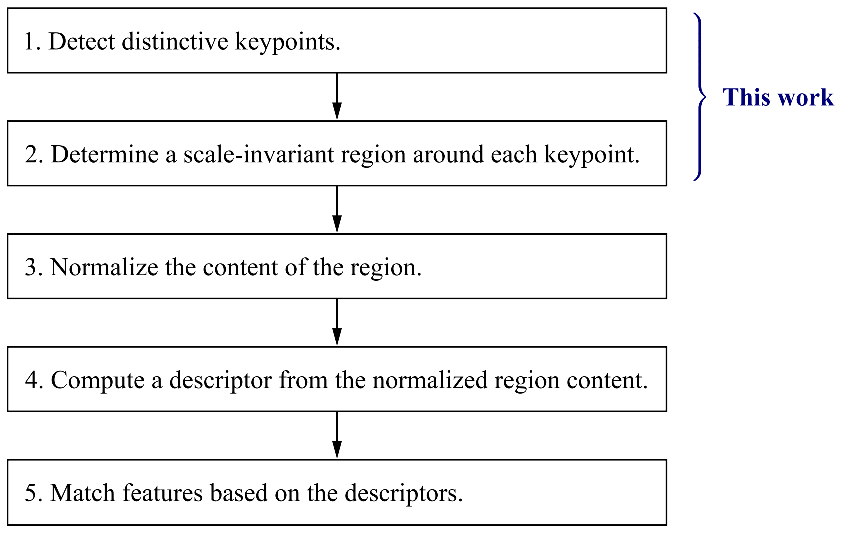

1. Detect distinctive keypoints.
2. Determine a scale-invariant region around each keypoint.
3. Normalize the content of the region.
4. Compute a descriptor from the normalized region content.
5. Match features based on the descriptors.
This work

**Fig. 2.4:** Standard procedure when working with local invariant features, based on the description in (Grauman and Leibe, 2011). This work focuses on the first two steps.

### 2.2.2 Local Invariant Image Features

As summarized in (Grauman and Leibe, 2011), an important step towards robust object recognition methods has been the development of local *invariant* features, particularly Scale-Invariant Feature Transform (SIFT; Lowe, 1999; 2004). The objective of using such features is to detect and represent local image structures in a way that is invariant to transformations typically encountered in practice, in particular rotation, scale changes, and affine transformations. However, achieving perfect invariance is usually not possible due to noise, discretization artifacts, blur, and other factors; the objective is therefore to achieve sufficient robustness. The following properties are particularly important for effective features (Grauman and Leibe, 2011):

- The feature extraction should be repeatable and precise (well-localized) so that the same features are extracted from different images of the same object.
- The features should be distinctive so that identical features can be matched and different features can be distinguished from each other.

As noted by Tuytelaars and Mikolajczyk (2008), the number of features should also be large enough to effectively represent objects or scenes. Furthermore, their density should reflect the information content in an image. These principles align with the contour features presented in this work, as curvature extrema are well-localized and the most informative points along contours (cf. Section 2.1.2).

The standard procedure when working with local invariant features is shown in Figure 2.4. In general, a local feature is defined by two main properties, which are also the focus of this work: its position $(x, y)$ in the image plane and a scale-invariant region around that position (steps 1 and 2). This region is then used to compute its description, typically in the form of a feature vector (steps 3 and 4). The key is that this region automatically adapts to scale changes so that the same image structures can still be identified and subsequently normalized—typically to the same size and a consistent orientation—to compute a rotation- and scale-invariant description. Achieving this automatic adaptation is a particular challenge, especially for the contour features presented in this work, as the image transformations are usually unknown and the features have to be determined in a bottom-up manner. This is why both SIFT and also the contour features presented in this work use a scale-space approach. In both methods, keypoints are detected as scale-space extrema, and the region around each keypoint is defined by a characteristic scale $\hat{\sigma}$. Further details on characteristic scales and SIFT are described in Sections 2.2.3 and 2.2.4, respectively. Note that the procedure can be extended to affine transformations by estimating affine shape parameters (by fitting elliptical regions) around detected features (Grauman and Leibe, 2011). However, this extension is also computationally more expensive.

The last step is to find correspondences between the same features in different images based on their descriptors (feature matching, step 5). The basic procedure typically consists of a matching and a verification stage. In the matching stage, candidate matches are identified as pairs with the smallest Euclidean distance between their descriptors, provided the distance is below a certain threshold. In the verification stage, false matches are removed. The simplest matching strategy is to compare each descriptor with all other candidate descriptors (brute-force matching). However, such a strategy can be computationally expensive, especially considering that feature vectors are often high-dimensional and when searching in larger image databases. For example, SIFT feature vectors are in $\mathbb{R}^{128}$, and the relatively small example image with $233 \times 189$ pixels in (Lowe, 2004) already contains 536 stable SIFT keypoints. Thus, the linear search is often replaced by more efficient strategies, such as tree or hashing-based algorithms. The verification stage is necessary because when working with real-world images, there are usually false matches due to repeating or similar image structures, i.e., the distance alone is not sufficient. Common strategies to avoid false matches include considering the ratio of the distance to the next best match, as close similar matches are often less reliable, or using methods like RANSAC (random sample consensus) to verify the geometric consistency. For further details and strategies, refer to (Q. Huang et al., 2024).

Besides direct feature matching, which is particularly effective for recognizing specific objects, local features can also be used for object detection, scene classification, and other category-level tasks. In such tasks, the focus is on the presence of similar features rather than their exact position or one-to-one correspondence. A common method in this regard is Bag-of-Features, where images are represented as histograms of feature frequencies, computed from clusters of local features identified using a set of training images (Csurka et al., 2004; Bay et al., 2008).

### 2.2.3 Characteristic Scales

An important property of local image features is whether they are represented as single points only or with additional scale information. As already outlined, the characteristic scales define the region around each keypoint used to compute its descriptor in an invariant manner. The term *characteristic* is used because it indicates the scale at which a particular feature is most prominent within a scale-space representation of an image (Lindeberg, 2014). In principle, scale refers to the standard deviation $\sigma$ of the filters used to compute different levels in the scale-space representation (details below).

An illustrative example is shown in Figure 2.5: each input image (left column) contains a prominent blob of a different size in the center. The Laplacian of Gaussian (LoG) function used to generate the filter masks is given as follows:

$$\mathrm{LoG}(x, y; \sigma) = \nabla^2 g(x, y; \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \tag{2.2}$$

where $\nabla^2$ denotes the Laplace operator, and the Gaussian $g$ is given in Equation (2.5). Let $(x_c, y_c)$ denote the center of the respective input image $f(x, y)$. The corresponding result $R$ is then computed by applying the LoG filter:

$$R(x_c, y_c; \sigma) = f(x, y) * \sigma^2\,\mathrm{LoG}(x, y; \sigma)\Big|_{(x_c, y_c)}, \tag{2.3}$$

where $*$ denotes discrete spatial convolution. The additional scaling factor of $\sigma^2$ is required to compensate for the effect of different filter sizes on the response magnitude and therefore to achieve true scale invariance (Lindeberg, 1994b; Lowe, 2004). The characteristic scale $\hat{\sigma}$ corresponds to the position of the maximum of this *signature function*. In this example, the filter response is computed directly at the image center to illustrate the principle, as this is the known position of the blobs. In practice, however, the response is computed at every position in the input image, and the positions with high responses are identified as keypoint candidates. As the filter responses show

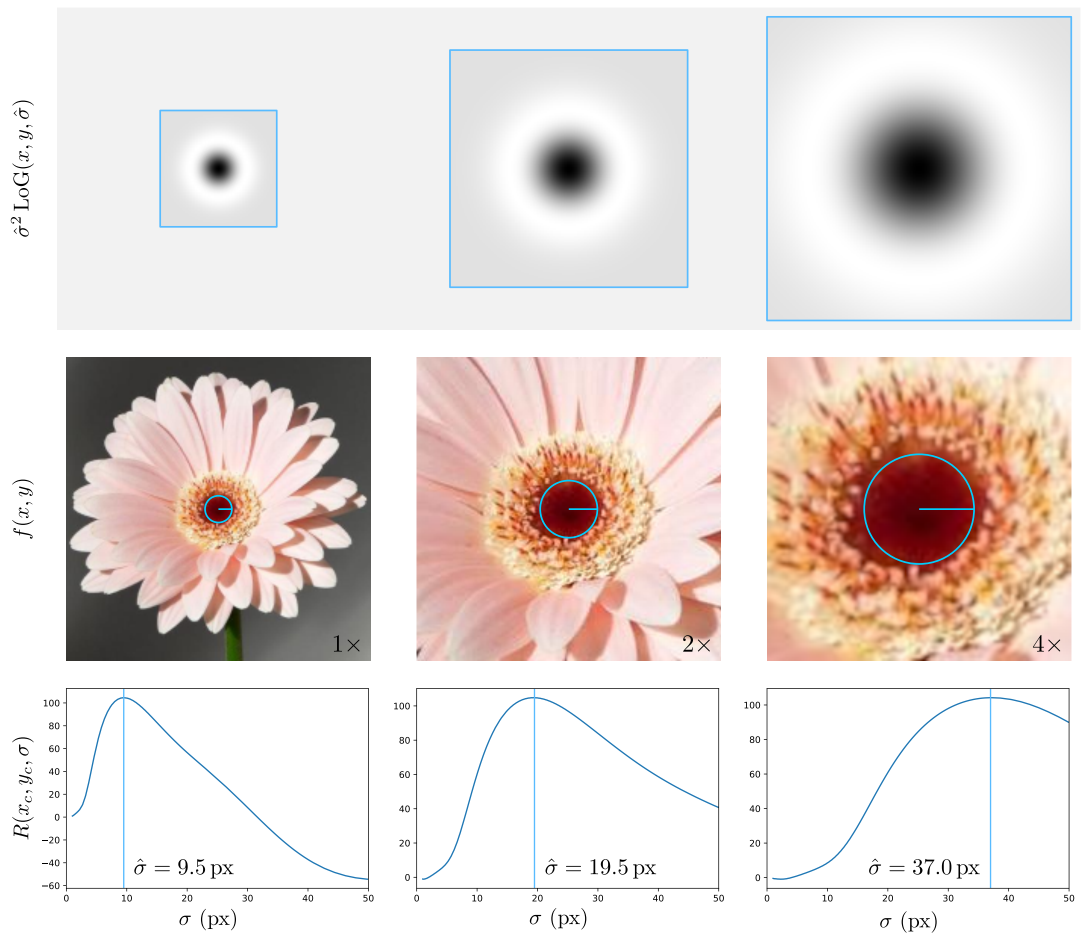


**Fig. 2.5:** Example of assigning characteristic scales to blob features. Top row: filters corresponding to the detected characteristic scales $\hat{\sigma}$ (normalized for display). Middle row: input images with overlaid regions defined by these scales (orientations not computed, lines for visual reference). Bottom row: filter responses (signature functions) at the image centers with maximum values indicated. Input image from the 102 Category Flower Dataset (Nilsback and Zisserman, 2008), then artificially scaled.

(bottom row), the signature functions and characteristic scales automatically adapt to scale changes so that the same image structures can be identified in a bottom-up manner without prior knowledge of these changes. For further details about the underlying scale-space framework, refer to Section 4.1.

The slight variations from the ideal values of the characteristic scales $\hat{\sigma}$ in the bottom row of Figure 2.5 are a result of the artificial scaling, the discrete increment of $\sigma$, and the discrete nature of the image data. In the middle row, the resulting regions indicated by the overlaid circles have a radius of $\hat{r} = \hat{\sigma}$, which is also used in the Open Source Computer Vision Library (OpenCV) implementation of SIFT (although other factors then 1.0 are possible). The characteristic scales of the contour features presented in this work are also based on the analysis of signature functions over scale, but the process is more complex than simply searching for the position of one distinct extremum (see Section 4.3.5). This could also be a reason why assigning characteristic scales is more common for appearance-based features than for contour features.

The top row in Figure 2.5 also shows the LoG filters for the respective characteristic scales. As the filtering process represents a correlation operation, these are the scales at which the filters are (mathematically) most similar to the blobs. However, it is important to note that candidate features do not need to perfectly match the symmetric form of the shown blobs. The filter response is high for any image structures which resemble the form of the LoG filter to a certain extent. Furthermore, it is common to not only check for maxima of the response $R$, but also for minima to equally consider dark-on-light and light-on-dark image structures.

The LoG filter is often used in the literature as it provides the properties for effective features described in Section 2.2.2: The same features can be extracted in a rotation-invariant manner due to the isotropic nature of the filter, the extraction is scale-invariant, and the specific form of the filter leads to well-localized features. Furthermore, as high filter responses require systematic changes in the image structure, computing distinctive descriptors is facilitated (homogeneous regions do not provide distinctive cues). The information content of an image is also represented, as keypoint candidates require a certain level of contrast—typically found in center-surround regions—which often correspond to salient structures in real-world images.

### 2.2.4 SIFT, SURF, and Other Features

This section describes the keypoint detection process and the assignment of characteristic scales in SIFT, Speeded-Up Robust Features (SURF; Bay et al., 2008), and other methods (addressing steps 1 and 2 in Figure 2.4). SIFT is used as a reference method in this work as it has established the principal steps outlined in Section 2.2.2 for working with local invariant features and remains one of the most influential methods in the field of feature detection and description. Many subsequent methods with similar principal steps have been derived from SIFT, with a particular focus on efficiency, and it has even been adapted using deep learning techniques (Yi et al., 2016; Ma et al., 2021; Tsourounis et al., 2022).

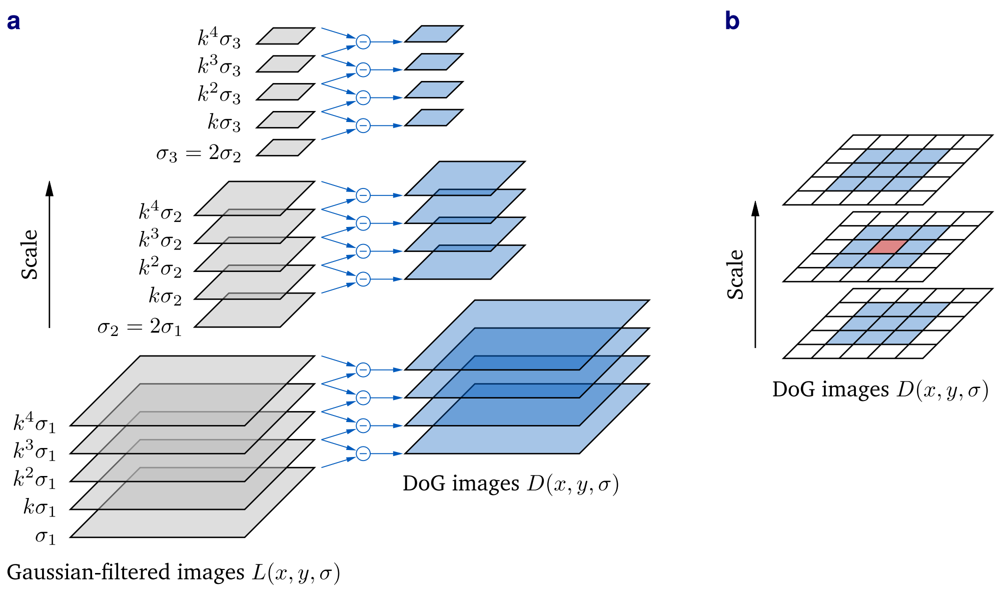


**Fig. 2.6:** Scale-space construction in SIFT with three octaves, adapted from (Gonzalez and Woods, 2018). **a** Computation of DoG images based on Gaussian-filtered images. **b** Detection of extrema (minima or maxima) in the DoG images by comparing the current pixel (highlighted) with its 26 neighbors.

### 2.2.4.1 Scale-Invariant Feature Transform (SIFT)

Instead of directly using LoG filters at different scales as described in Section 2.2.3, SIFT approximates the filter results using DoG images. These images are straightforward to compute, and the Gaussian-filtered images are also used to compute the descriptors in that method. Furthermore, rather than maintaining a constant image size and linearly increasing the scale, the scale-space is constructed using an image pyramid approach with octaves (where each octave corresponds to a doubling of $\sigma$). In this approach, the image size is decreased in each octave to improve the computational efficiency (details below). This is an advantage given the high computational complexity of the filtering process: For an $M \times N$ image and an $m \times n$ filter, the complexity at a single scale is $\mathcal{O}(MNmn)$, and the process is repeated across multiple scales.

To compute the Gaussian-filtered images $L(x, y; \sigma)$ in Figure 2.6a, the input image $f(x, y)$ is convolved with Gaussians $g(x, y; \sigma)$:

$$L(x, y; \sigma) = g(x, y; \sigma) * f(x, y)\,, \tag{2.4}$$

where the standard deviation $\sigma$ (scale parameter) of the Gaussian is systematically increased, and

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right) . \tag{2.5}$$

The standard deviation is set to $\sigma, k\sigma, k^2\sigma, k^3\sigma, \ldots$, where $k = 2^{1/s}$, and $s$ is the number of DoG images per octave used to search for extrema (also called intervals per octave). This procedure leads to equally spaced scales between these DoG images across the octaves. The DoG images are obtained by computing the difference of two Gaussian-filtered images with nearby scales separated by the factor $k$ (Lowe, 2004):

$$D(x, y; \sigma) = L(x, y; k\sigma) - L(x, y; \sigma) . \tag{2.6}$$

Each octave corresponds to a doubling of the scale parameter $\sigma$, and the number of octaves and DoG images are parameters that depend on the size of the input image. Since each pixel in the DoG images is compared to its 26 neighbors, including the adjacent images above and below, as shown in Figure 2.6b, and the first and last DoG images in an octave do not have images above or below, $s = 2$ in this example. The principal relationship between the Gaussian filter and the LoG filter is given as follows (Lowe, 2004):

$$g(x, y; k\sigma) - g(x, y; \sigma) \approx (k - 1)\sigma^2 \, \mathrm{LoG}(x, y; \sigma) . \tag{2.7}$$

As already outlined in conjunction with Equation (2.3), the factor $\sigma^2$ is required for true scale invariance. The factor $(k - 1)$ is constant for all scales so that it does not affect the extrema locations.

In summary, there are $s + 2$ DoG images and $s + 3$ Gaussian-filtered images per octave. A pixel is selected as a keypoint candidate if its value is larger or smaller than the values of its 26 neighbors. The corresponding characteristic scale is then set to the value of $\sigma$ in the DoG image $D(x, y; \sigma)$, where the extremum has been found.

As outlined in (Gonzalez and Woods, 2018), the *first* image of the *second* octave is formed by downsampling the original image by skipping every other row and column, and then applying Gaussian filtering (smoothing) with a standard deviation of $\sigma_1$. This corresponds to smoothing the original image using $\sigma_2 = 2\sigma_1$. Note that the scale $\sigma_2$ of the first image in the second octave refers to the scale *represented* by that image, but the actual filtering is still performed using $\sigma_1$. The remaining octaves are computed in the same manner.

The procedure described is combined with additional steps to enhance the robustness of the keypoints. To obtain the input image for the first octave, the original image is smoothed with a Gaussian with $\sigma = 0.5$ and then doubled in size by linear interpolation.

As reported in (Lowe, 2004), this preprocessing improves the number of stable keypoints. Furthermore, the initial keypoints detected in the DoG images are not necessarily accurate with respect to their position and scale $(x, y; \sigma)$ due to the discrete nature of the data. The accuracy of a keypoint can be improved using a quadratic Taylor expansion of $D(x, y; \sigma)$ at the respective position. Furthermore, keypoints with low contrast are unstable and typically not distinctive, so they should be removed. These points have small absolute values in the DoG images and are removed if their values are below a certain threshold. Keypoints along edges are not well-localized (since points along an edge appear very similar) and should therefore be removed as well. These points can be identified by analyzing the principal curvature at keypoint positions, which is high in one direction (along the edge) and low in the perpendicular direction.

For the subsequent steps of computing a descriptor from the region around each keypoint defined by its characteristic scale, refer to (Lowe, 2004) and (Gonzalez and Woods, 2018). In summary, the region is normalized to a fixed size and divided into smaller subregions, where orientation histograms are computed and concatenated into a feature vector in $\mathbb{R}^{128}$.

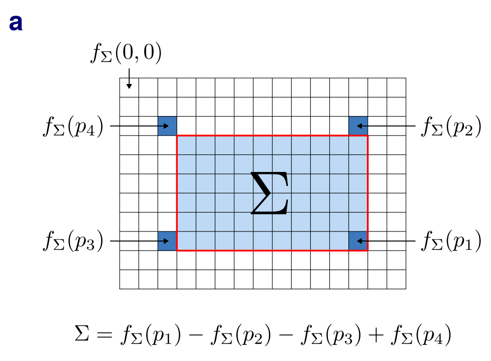
### 2.2.4.2 Speeded-Up Robust Features (SURF)

The SURF method introduced in (Bay et al., 2008) detects blob and corner features and uses an efficient scale-space representation based on the Hessian matrix (Hessian detector), which is another common feature detection method besides using the LoG filter. The method is discussed here because its use of box filters has been adapted to extract the contour features presented in this work and to provide a more comprehensive overview of widely-used feature detectors.

The Hessian matrix $\mathbf{H}(x, y; \sigma)$ at a point $(x, y)$ in an image at scale $\sigma$ is defined as:

$$\mathbf{H}(x, y; \sigma) = \begin{bmatrix} L_{xx}(x, y; \sigma) & L_{xy}(x, y; \sigma) \\ L_{xy}(x, y; \sigma) & L_{yy}(x, y; \sigma) \end{bmatrix}, \tag{2.8}$$

where $L_{xx}$, $L_{yy}$, and $L_{xy}$ are the convolutions of the image with second-order Gaussian derivatives, corresponding to Equation (2.4) when applying the convolution derivative theorem. Keypoint candidates are identified based on the determinant of $\mathbf{H}$, which has high values at points with strong curvature in multiple directions, corresponding to local intensity maxima (peaks) or minima (valleys). Similar to SIFT, SURF uses octaves in its scale-space construction, but increases the filter size instead of reducing the image size. Non-maximum suppression is again applied based on the 26 neighbors of each pixel, as shown in Figure 2.6b, but using the determinant of $\mathbf{H}$ instead of DoG images.



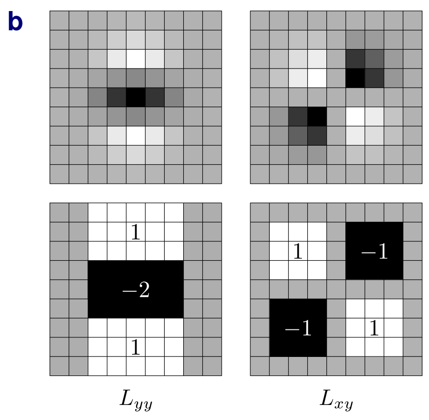


**Fig. 2.7:** Components for an efficient scale-space construction in SURF, adapted from (Bay et al., 2008). **a** Integral image, where the sum of intensities over a rectangular area can be computed using only three arithmetic operations. **b** Examples of Gaussian second-order partial derivatives (top row) and their box filter approximations (bottom row) used to compute $L_{yy}$ and $L_{xy}$, respectively.

As already noted, the method is of particular interest here due to its use of box filters for an efficient scale-space construction. The computational speedup is achieved by using box filters in conjunction with integral images. The integral image $f_\Sigma(x,y)$ stores the sum of the intensity values within the rectangle between the origin and the point $p = (x,y)$ of the input image (Bay et al., 2008):

$$f_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} f(x,y) \tag{2.9}$$

Based on the integral image, the sum of the intensities over a rectangular area—which is required for box filters—can be computed using only two subtraction and one addition operation (three arithmetic operations), as shown in Figure 2.7a. As a result, the filtering process is independent of the filter size in terms of the number of arithmetic operations and therefore much faster than regular convolution, where all filter coefficients and intensity values have to be multiplied and summed. Examples of Gaussian second-order partial derivatives and their box filter approximations used in SURF are shown in Figure 2.7a, where $L_{xx}$ is constructed in the same manner. The size of the filters is increased for higher scales while ensuring that their size remains odd.

For the subsequent steps of computing a descriptor from the region around each keypoint, refer to (Bay et al., 2008). In summary, the normalized region is divided into smaller subregions, where Haar wavelet responses are computed and concatenated into a feature vector in $\mathbb{R}^{64}$.

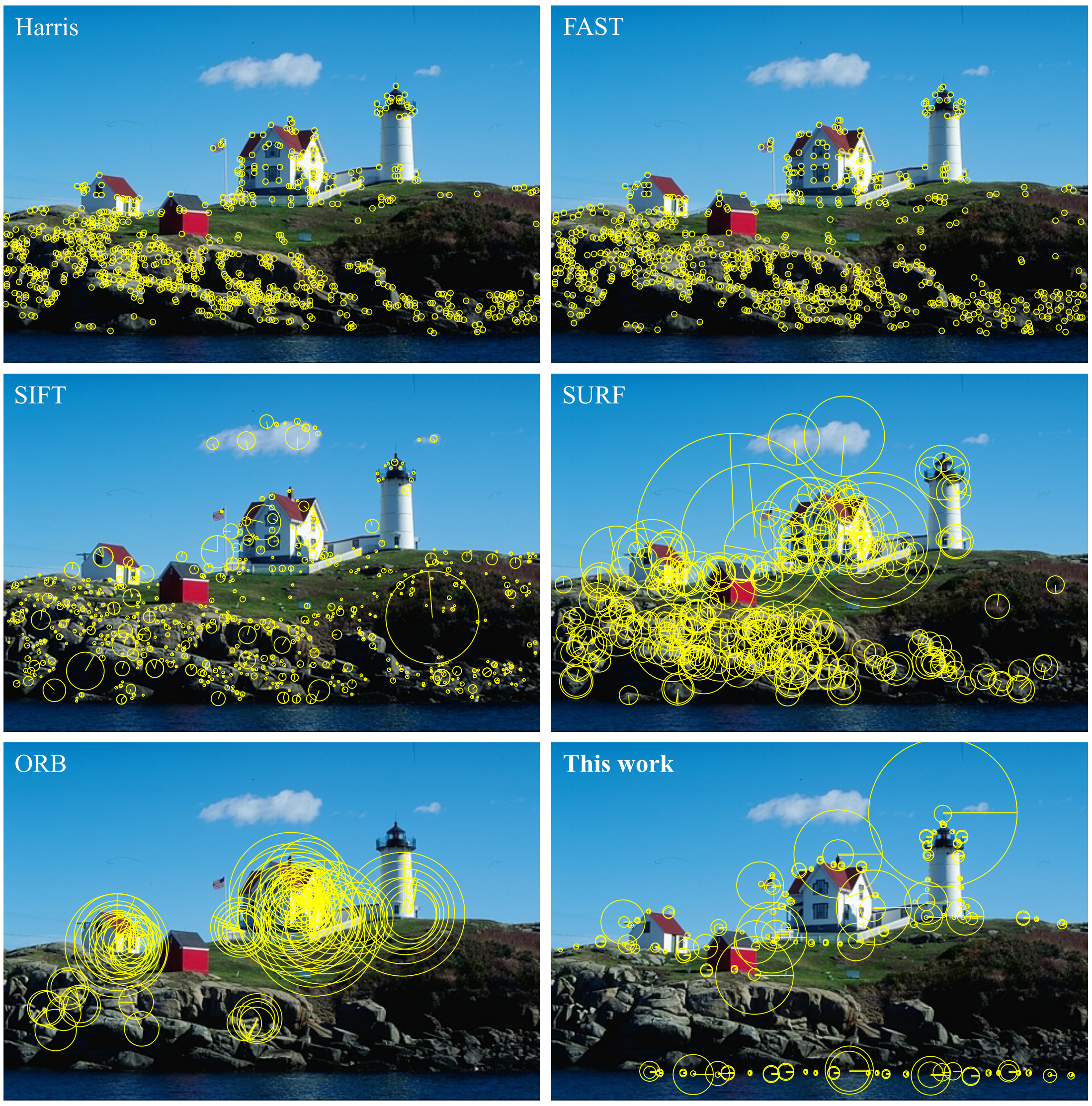


**Fig. 2.8:** Results for different local feature detection methods and the method developed in this work (orientations not computed) on image 228076 from the BSDS dataset. All results (except for this work) were computed using OpenCV with standard parameters, with additional non-maximum suppression for Harris and retaining only the 100 strongest keypoints for ORB. The results for the method developed in this work were computed based on manual segmentation data provided with the dataset (cf. Figure 2.9b).

#### 2.2.4.3 Oriented FAST and Rotated BRIEF (ORB)

Another widely used feature detection and description method, introduced in (Rublee et al., 2011), is ORB. This method is discussed here as an example of using binary feature descriptors, a widely used approach for achieving high computational efficiency.

A comparative evaluation of binary features can be found in (Heinly et al., 2012). As the name suggests, ORB combines and extends two other methods: the FAST feature detector (Features from Accelerated Segment Test; Rosten and Drummond, 2006), extended with orientation information, and the BRIEF feature descriptor (Binary Robust Independent Elementary Features; Calonder et al., 2010), extended so that the descriptor is computed relative to the orientation from the detector to obtain a rotation-invariant description.

In its original form, FAST is a corner detector that identifies candidate keypoints by comparing the intensity of each pixel with the intensities of its surrounding pixels in a circular pattern. As this pattern is fixed, ORB is categorized as a template-based method in Figure 2.3. Although FAST is a corner detector, ORB is categorized as a blob detector in that figure (according to Y. Li et al., 2015) because it employs an image pyramid to detect the FAST features at multiple scales. Within the pyramid, fine-scale image structures are merged into larger structures so that regions with blob-like structures can be identified across scales. The orientation of the detected keypoints is computed based on intensity-based image moments. Based on that, the rotation-invariant descriptor is computed by concatenating the results of specific pairwise pixel comparisons into a feature vector in $\mathbb{B}^{256}$.

### 2.2.5 Region-Based Image Segmentation

The method developed in this work to extract the contour features requires traced object contours without ambiguities (where ambiguities are points where the path of a contour is not clearly defined, such as at intersections or junctions; and where the contour points are ordered; see Chapter 3 for details). One option to obtain such contours is to trace the outlines of region-based image segmentations, with examples shown in Figure 2.9. For an overview of both traditional and more recent deep learning-based methods, refer to the surveys by Minaee et al. (2022) and Yu et al. (2023). Many traditional segmentation methods use clustering techniques to group similar pixels or regions based on features such as color, texture, intensity, and spatial proximity. However, among the numerous image segmentation methods developed in the literature, deep learning-based methods often achieve the highest accuracy rates on popular benchmarks (Minaee et al., 2022). In region-based image segmentation, one can generally distinguish between semantic and instance segmentation. The objective of semantic segmentation is to label object categories (e.g., a group of people as a single region), whereas the objective of instance segmentation is to label individual objects (e.g., each individual person as a separate region). As the contour features presented in this work are local features designed to describe local salient image structures, instance segmentation regions—or panoptic

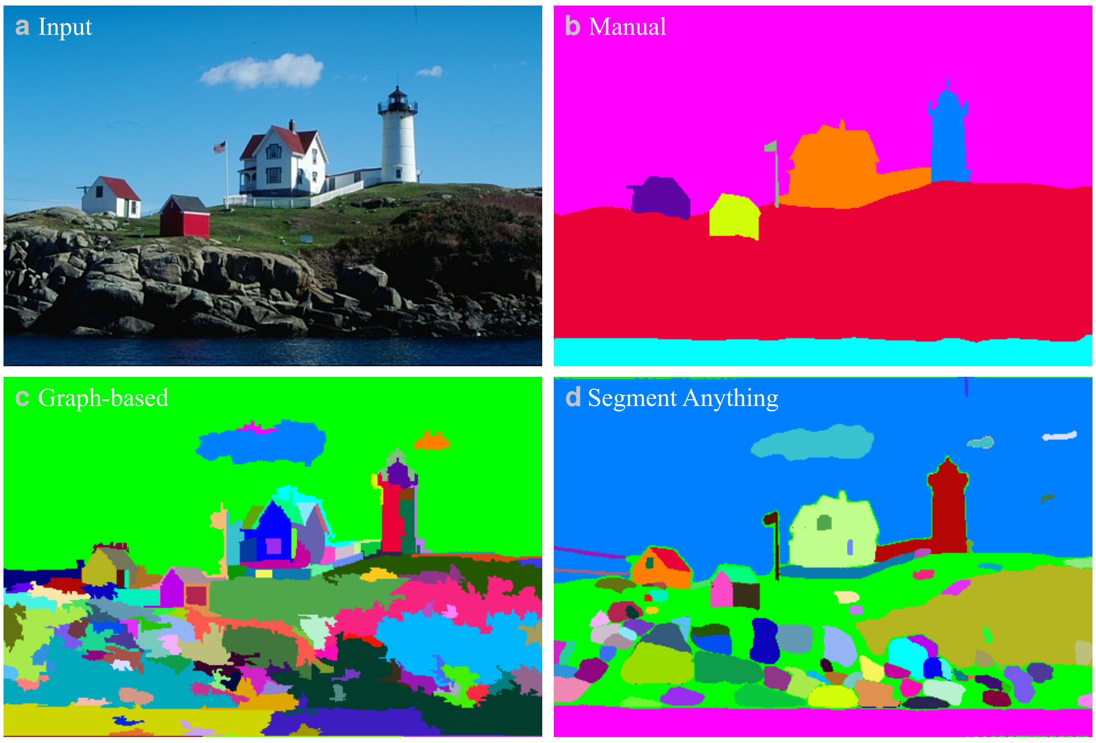


**Fig. 2.9:** Results for different segmentation methods on image 228076 from the BSDS. The region contours can be traced and used to extract the contour features presented in this work. **a** Input image. **b** Manual segmentation by a single participant provided with the dataset. **c** Graph-Based segmentation (Felzenszwalb and Huttenlocher, 2004), computed using OpenCV. **d** Segment Anything (Kirillov et al., 2023), computed using the implementation provided by the authors.

segmentation regions, which combine both—are better suited to obtain accurate and meaningful object contours.

Deep learning-based methods require training data in the form of manually labeled pixel-level masks, and a number of such datasets are available (refer to Section 4 in Minaee et al., 2022, for an overview). Some of the most popular datasets in this context are the the Microsoft Common Objects in Context dataset (MS COCO; Lin et al., 2014), the PASCAL Visual Object Classes dataset (PASCAL VOC; Everingham et al., 2010), and the Berkeley Segmentation Dataset (BSDS; Arbeláez et al., 2011). Throughout this work, BSDS specifically refers to the BSDS500 version, not the earlier BSDS300 version. While MS COCO provides instance masks for individual objects, PASCAL VOC only provides semantic segmentation data. In the BSDS, regions are annotated without differentiating between instances of the same object category or assigning semantic labels to regions. In this case, subjects were instructed to "divide each image into pieces, where each piece represents a distinguished thing in the image." An example annotation by a single participant is shown in Figure 2.9b. Each image has been annotated by around five subjects to capture different perceptions of what are meaningful entities.

The result of the graph-based method, as shown in Figure 2.9c, is a popular traditional approach that employs graph-theoretic principles for image segmentation. As can be seen, the method is not capable of capturing the sea in the lower part of the image or the rock structures in a meaningful manner. However, structures like the houses are clearly identified and can probably be merged to also consider larger-scale structures. A popular method in this regard is Selective Search (Uijlings et al., 2013), where the output of the algorithm is used to generate region proposals—rectangles (bounding boxes)—which can then be used as input regions for deep learning-based classification methods. This also underlines the importance of considering different scales when analyzing images, as discussed in greater detail in connection with scale-space theory in Section 4.1. The advantage of using deep learning-based segmentation methods can be clearly seen in Figure 2.9d, where the sea and even the rock structures are captured in a meaningful manner. Segment Anything is one of the most accurate and versatile segmentation methods, capable of providing high-quality instance segmentation masks alongside other segmentation tasks, such as semantic and interactive segmentation (Kirillov et al., 2023). In Section 5.2, it is proposed to use deep learning-based region segmentations as one option to generate training data for extracting the contour features presented in this work directly from real images using an end-to-end deep learning-based approach.

### 2.2.6 Edge Images and Edge Detection

Another option to obtain traced object contours for extracting contour features with the method developed in this work is to retrieve them from edge images, with examples shown in Figure 2.10. For an overview of both traditional and more recent deep learning-based edge detection methods, refer to the surveys by Jing et al. (2022) and D. Yang et al. (2022). Following the survey by D. Yang et al., one can generally distinguish between traditional and learning-based edge (and contour) detection methods, with further categorization depending on the specific methodology, such as classical learning-based and deep learning-based methods. Furthermore, just as with region-based segmentation methods in Section 2.2.5, deep learning-based methods generally achieve the highest accuracy rates on popular benchmarks and also require appropriate training data. A common approach is to derive such training data from the outlines of region segmentations, with an example of the resulting edges shown in Figure 2.10b.

One of the most popular and still widely used traditional methods is the Canny edge detector (Canny, 1986). The algorithm is based on three objectives: low error rate,

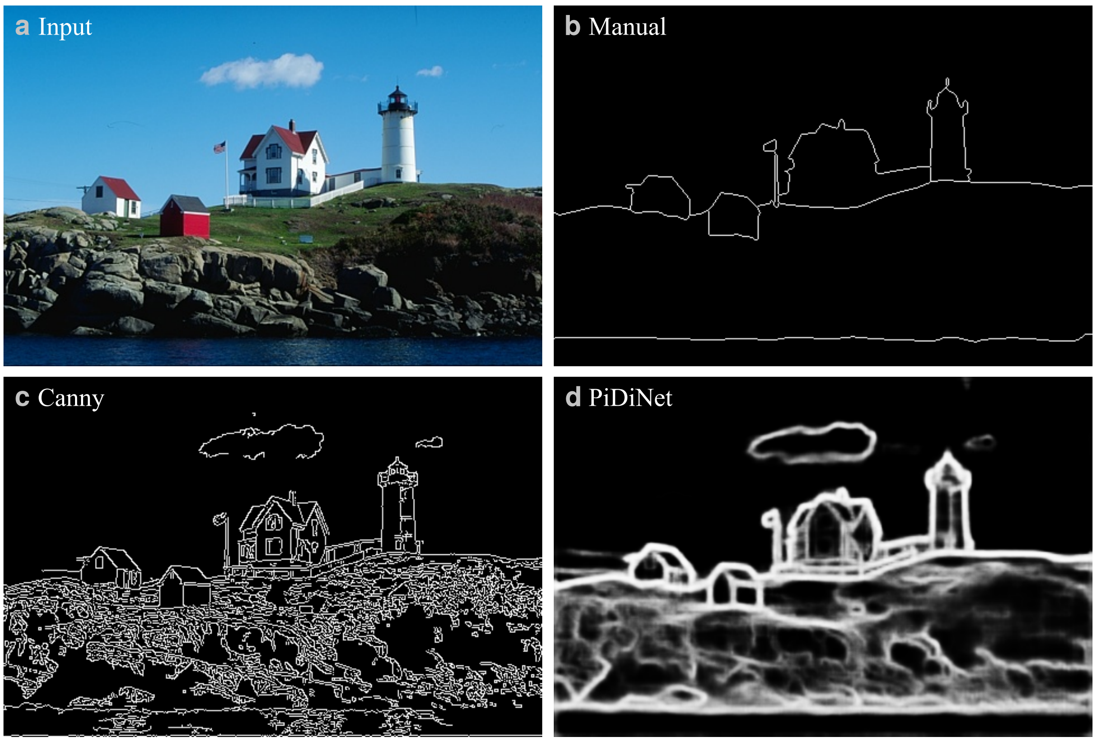


**Fig. 2.10:** Results for different edge detection methods on image 228076 from the BSDS. The edges can be further processed and traced to extract the contour features presented in this work. **a** Input image. **b** Manual segmentation by a single participant provided with the dataset, derived from the outlines of region segmentations. **c** Canny edge detector (Canny, 1986), computed using OpenCV. **d** Pixel Difference Network (PiDiNet; Su et al., 2023), computed using the implementation provided by the authors. Further binary edge images are shown in Figure 3.7.

well-localized edge points, and single edge point responses to obtain one-pixel wide edges. After an initial smoothing step to suppress noise, the edge detection process employs gradient computation, for example using Sobel masks, and is therefore based on first-order derivatives. This step produces an edge map similar to Figure 2.10d, where the gray values represent the edge strength. However, to obtain distinct object contours and the binary edge image shown in Figure 2.10c, the edge map has to be post-processed. This includes non-maximum suppression, double (hysteresis) thresholding, and connectivity analysis to link edges. Although other strategies are possible, this post-processing "has become a standard procedure for many edge and object contour detection methods" (D. Yang et al., 2022).

An important advantage of learning-based edge detection methods compared to differentiation-based methods is that they can suppress edges caused by textures. In other words, they can distinguish between meaningful object boundaries and irrelevant texture edges. Compare Figure 2.9c and 2.9d: It can clearly be seen that the gradient-based Canny method detects many edges in textured regions like the sea and the rock structures, whereas the deep learning-based PiDiNet detects the strongest edges at the outer boundaries of these structures. Learning-based edge detection methods are specifically trained to identify edges and contours, which enables them to capture finer details than region-based segmentation methods. While region-based segmentation relies on categories or labels, edge detection methods can identify edges without relying on these attributes, as their training data includes only edges and no specific semantic information.

Motivated by the increasing capabilities of deep learning-based edge detection methods, a new method to trace edges in binary edge images has been developed as part of this work. As shown in Figure 2.10, object contours cannot be directly attributed to a single object. Instead, the edges exhibit ambiguities in the form of intersections, junctions, and other structures. Furthermore, it is also ambiguous which entities should be considered meaningful object structures—for example, whether the island in the figure should be treated as one structure or distinguished into grass and rock structures. The method developed in this work can describe and resolve such ambiguities in an intuitive and flexible manner. This method is presented in Chapter 3. In Section 5.2, it is proposed to combine this method with deep learning-based edge detection as another option (besides region segmentations) to generate training data for extracting the contour features presented in this work directly from real images using an end-to-end deep learning-based approach.

# 3 General Ambiguity Model for Binary Edge Images

As outlined in Section 2.2.6, binary edge images often include intersections, junctions, and other structures that make it difficult to extract coherent object contours. Nevertheless, such contours are essential for tasks such as object recognition and for extracting the contour features presented in this work. This chapter introduces the general ambiguity model for binary edge images developed in this work, which is designed to describe and resolve such ambiguities. The model is combined with edge tracing, where edges are *ordered* sequences of connected pixels. From a broader perspective, the objective is to provide a versatile preprocessing method for tasks such as figure-ground segmentation, object recognition, topological analysis, and other applications. An open source C++ implementation of the method has been published on GitHub.

Significant parts of this chapter, including all figures, are reproduced from (Hennig, Leineke, and Mertsching, 2024), with all writing and major conceptual contributions attributed to the author of this thesis.

After a brief introduction in Section 3.1, the most common methods for processing binary edge images are characterized in Section 3.2 in relation to the method developed in this work. In Section 3.3, the ambiguity model is introduced, including a detailed description of the modeling principles, a pseudocode implementation with three core subfunctions, a proof of correctness, and results for various examples and postprocessing steps. In Section 3.4, the method from this work is compared against other methods in terms of component decomposition, redundancy, runtime, and other aspects. The chapter concludes with a summary in Section 3.5.

## 3.1 Introduction

Coherent object contours in the form of traced edges (i.e., ordered sequences of connected edge pixels) are required for many standard shape description methods such as Fourier descriptors, chain codes, signature functions, and curvature scale-space (CSS) analysis (Mokhtarian and Bober, 2003; L. Wang et al., 2018). The contour features

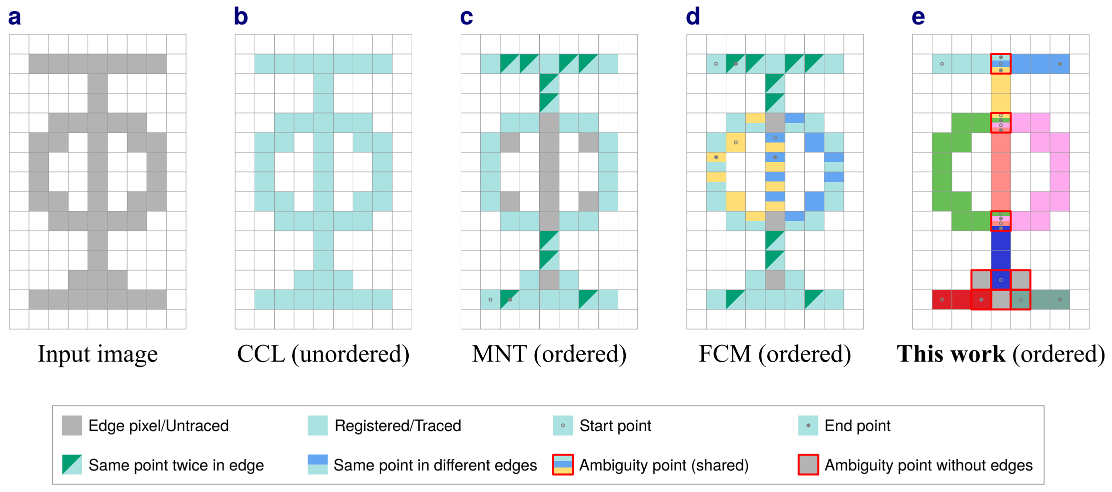


**Fig. 3.1:** Comparison of different standard methods for processing binary edge images with the method developed in this work, where different colors represent different edges. **a** Input image. **b** Connected Component Labeling (CCL; cf. L. He et al., 2017). **c** Moore-Neighbor Tracing (MNT; Ghuneim, 2000). **d** Find Contours Method (FCM, as named in OpenCV; Suzuki et al., 1985). **e** This work, which provides direct access to ambiguities, registers all pixels, and avoids redundancies.

presented in this work are also derived from CSS analysis, as described in Chapter 4. Another potential application is the generation of object bounding box proposals, such as in (Zitnick and Dollár, 2014). Additionally, the method could be used in applications where explicit edge connection information—as effectively described by the ambiguity model—is of interest, such as determining cell-cell contact phenotypes similar to the work in (Brezovjakova et al., 2019) or describing vessel crossings similar to the work in (G. Wang et al., 2023).

Figure 3.1 shows the results of common methods for processing binary edge images compared to the method developed in this work. The methods are described in Section 3.2. In summary, the method from this work registers *all* pixels without redundancy (no double reading per edge), provides a structured decomposition into meaningful edges, and direct access to ambiguities. In comparison, CCL does not provide any order, while MNT and FCM miss some inner pixels, including those in the vertical line running through the circle. Furthermore, these methods do not directly model ambiguities of edge paths.

## 3.2 Related Work

One of the most common methods for processing binary edge images is CCL, which simply labels connected pixels. MNT is a standard contour tracing method and, in this work, serves as a representative for many similar methods with slight modifications. A comprehensive comparison can be found in (Seo et al., 2016). Besides MNT, the authors review existing contour tracing methods, such as the Square Tracing Algorithm, Radial Sweep, and Theo Pavlidis' algorithm. FCM is another popular tracing method that also detects hierarchies between edges (parents and children). However, none of the methods provide an ambiguity model. Since MNT and FCM are region contour tracers, they often trace edge pixels more than once per edge, leading to redundancies. In contrast, the method developed in this work traces each edge pixel only once per edge. Some inner pixels are not traced at all in MNT and FCM.

Compared to existing works, the method developed in this work provides direct access to ambiguities and connected traced edges in the image plane, relating them without complex multi-stage processes or graph representations. The method *describes* binary edge images and provides an intermediate representation independent of any specific edge detection method or application-specific objectives and metrics. The work in (Heng and Ngan, 2001) focuses on identifying specific simple edge configurations for splitting edges, while (Law et al., 1996) explores different types of joins, including scenarios with three edges at a triple point and two edges at a junction. The method in (Guo et al., 2014) consists of six stages and groups edge segments based on cues from real images. Different from the ambiguity model developed in this work, ambiguities are considered as options for connecting mostly free-standing edge segments, are modeled using graphs, and the overall results are evaluated in terms of edge *detection* results. The method in (Pham et al., 2014) detects junctions in binary line drawings by searching for optimal meeting points and classifies them into different types. Unlike the method developed in this work, it does not model multi-pixel ambiguities, a larger number of edges connected to single ambiguities, or provide direct access to each segment.

The method in (Casadei and Mitter, 1999) is a multi-stage algorithm using a contour graph to group edge segments, where the concept of ambiguities is not as straightforward as in this work. The method in (Kimia et al., 2019) models ambiguities as possible edge connections based on propagated curve bundles, which is also more complex. The method in (Buades et al., 2018) detects line segments and other edge image elements, and the method in (K. Huang et al., 2018) detects junctions in real images to extract wireframe representations of man-made environments, but both methods do not provide an ambiguity model. The method in (Tamrakar and Kimia, 2007), a preliminary work to

(Kimia et al., 2019), represents edge segment combinations using curve bundles, and the ambiguity model is more complex and has different objectives than the method developed in this work. The method in (Zhu et al., 2007) groups edge segments to search for cyclic structures using a directed graph, and the method in (Maire et al., 2008) combines edge detection with junction detection in real images, but both methods do not provide an ambiguity model. Other methods aim to accelerate the contour tracing, as explored in (Gupta and Kar, 2022). In summary, most existing works focus on grouping edge segments to refine edge detection results, rather than directly describing these results in a flexible and intuitive manner. The method developed in this work could likely be integrated as an intermediate step in such works.

## 3.3 Ambiguity Model and Tracing

Ambiguities in binary edge images occur when edges are more than one pixel wide, when edges cross or meet at intersections or junctions, or when the edge detection process introduces artifacts. As a result, edges can share one or more pixels and they can be adjacent to or running through pixel clusters. When edges meet at a single pixel, such as at T-, Y-, and X-junctions, as shown in Figure 3.4b, they only share that pixel. However, more complex junctions can extend over several pixels (pixel clusters) with variable shapes and sizes, as shown in Figures 3.4–3.6, which cannot be effectively processed by implementing a separate case for each variation.

For some conceptual descriptions, the two types are distinguished here as single-pixel ambiguities (SPAs) and multi-pixel ambiguities (MPAs). For the ambiguity model and implementation, this explicit distinction is not required, and one can simply check the size (number of corresponding pixels) of an ambiguity if necessary. Depending on the context, pixels are referred to as points, especially when explicitly considering their spatial coordinates $(x, y)$.

### 3.3.1 Modeling Principles

As already outlined, edges can share single pixels (SPAs) and they can be connected to pixel clusters (MPAs). The model developed in this work integrates these cases in a general and intuitive way, without the need to distinguish the two types or various specific pixel configurations. It uses the following straightforward principles, which work in conjunction with each other (the reader is encouraged to refer to the principles alongside Figure 3.4):

1. Edges adjacent to ambiguities are connected to them via the shortest path.
2. The connection pixel is the start or end point of the respective edge.
3. Each pixel cluster forms a single coherent ambiguity.
4. Cluster pixels without edge connections are preserved but not traced.

These principles have been derived from extensive tests with real and artificial binary edge images to develop an effective model. Principle 1 is inherently satisfied for SPAs, as there is only one direct connection option present. For MPAs, it is consistent with edge tracing, where adjacent edge pixels are also connected via the shortest path. Principle 2 is inherently satisfied for SPAs, since all edges share the respective pixel, as shown in Figure 3.4b. It naturally extends to MPAs by integrating the connection pixel into each edge as well. Having only a single connection pixel is meaningful because edge paths within ambiguities are not clearly defined. From an implementation perspective, the principle provides a clear link between edges and ambiguities, which is helpful to determine potential edge connections. Principle 3 maintains clarity despite variable shapes and sizes of clusters. By treating each cluster as a single coherent ambiguity, all connected edges are linked by a single entity, just as with SPAs. Principle 4 can only take effect for MPAs (clusters), as SPAs always have edge connections by definition. It ensures that all pixels leading to an ambiguity are preserved and that the untraced pixels within an ambiguity are directly accessible. This is helpful to determine potential edge connections that may exclusively use detected edge pixels. In summary, Principles 1–3 can be seen as natural generalizations from SPAs to MPAs, and Principle 4 as a logical consequence of unclear edge paths.

### 3.3.2 Procedure and Pseudocode Implementation

The algorithm operates in two main passes: First, all ambiguities are identified so that the remaining pixels after this step are exclusively edge pixels to be traced. Second, these edge pixels are traced in sequential order, during which the resulting edges are connected to adjacent ambiguities, if present. This two-pass structure provides clear operational control and helps to verify that the algorithm works as intended.

In the following pseudocode implementation, Algorithm 1 controls the overall operation, which is divided into the two main passes mentioned (preprocessAmbiguities in Line 3 and the for-loop beginning at Line 4). The pseudocode conceptually aligns with the C++ implementation.

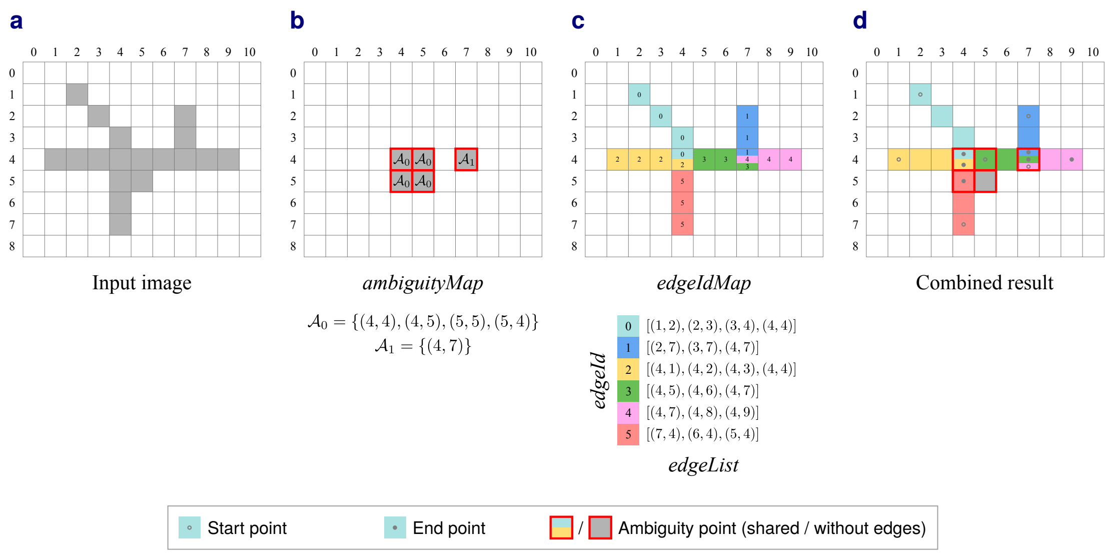


**Fig. 3.2:** Modeling concept and central data structures of the algorithm. **a** Binary edge image with two ambiguities. **b**, **c** Together, the *ambiguityMap* and *edgeIdMap* form an augmented edge map, where the traced edges are stored in the *edgeList*. **d** Combined result with shared pixels and connections to ambiguities.

### 3.3.2.1 Data Structures

As shown in Figure 3.2, the algorithm utilizes three central data structures: *ambiguityMap*, *edgeIdMap*, and *edgeList*. Both the *ambiguityMap* and *edgeIdMap* have the size of the original input image and jointly form an augmented edge map.

The *ambiguityMap* stores all ambiguity points, where each point stores the coordinates of *all* pixels belonging to the respective ambiguity $\mathcal{A}_i$. Thus, every point belonging to the respective ambiguity can be directly retrieved from every point in that ambiguity.

The *edgeList* holds all traced edges, where an *edge* is an ordered sequence of connected edge pixels. The position of each *edge* in this list is also its identifier, a simple integer referred to as *edgeId*.

The *edgeIdMap* stores, at each point, a list containing the *edgeIds* of every *edge* running through that point (the list is empty if no edges are running through). Thus, neighboring edges (and their data) can be directly accessed using quick local search operations in the *edgeIdMap*.

Together, the *ambiguityMap* and *edgeIdMap* provide direct access to every edge connected to an ambiguity. Without the *ambiguityMap*, it is not directly clear which edges are connected to specific ambiguities. Other edge-processing tasks, such as bridging gaps between nearby edges, can also be addressed in an efficient manner.

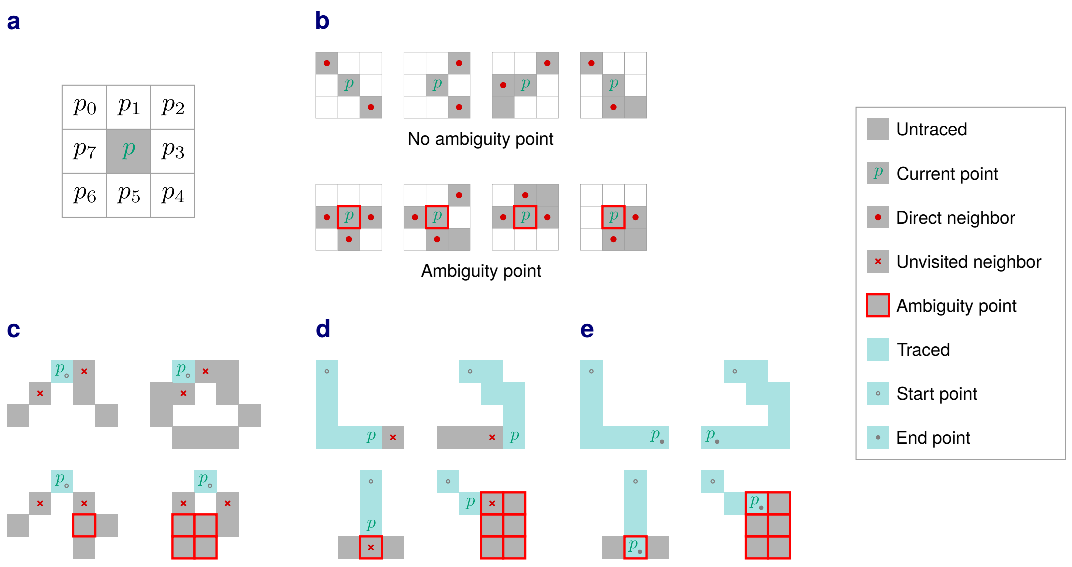


**Fig. 3.3:** Building blocks of the algorithm. **a** Naming of the 8-neighbors of the current point. **b** Mechanism for identifying ambiguity points. **c**–**e** Tracing with two, one, and no unvisited neighbors, respectively.

### 3.3.2.2 Core Subfunctions

The algorithm utilizes three straightforward core subfunctions: getDirectNeighbors, containsFourCluster, and mergeEdges. The first two functions are used in conjunction to check if the current point is part of an ambiguity, as in Lines 5 and 13 of Algorithm 2. Additionally, the function getDirectNeighbors is used in Line 5 of Algorithm 3 to trace (connect) adjacent edge pixels via the shortest path. The function mergeEdges is used in Algorithm 3 to combine two edges into a single edge during the tracing. The naming of the 8-neighbors of the current point $p$ for the following descriptions is shown in Figure 3.3a.

The function getDirectNeighbors analyzes the 8-neighbors of the current point $p$. The function operates in a specific way and does not simply return *all* neighbors that are set, as shown with examples in Figure 3.3b: It returns all orthogonal neighbors $(p_1, p_3, p_5, p_7)$ that are set, but from the diagonal neighbors $(p_0, p_2, p_4, p_6)$ *only those* which do not have any set orthogonal neighbor in $(p_1, p_3, p_5, p_7)$. The returned neighbors are denoted as *direct neighbors*. If the function returns more than two direct neighbors, the current point is definitely part of an ambiguity. Note that this check is not sufficient to identify all ambiguities; the result of the function containsFourCluster must also be considered (cf. examples in Figure 3.3b).

The function containsFourCluster checks if the current point $p$ is located within a four-cluster (a $2 \times 2$ block of set pixels). In this case, every point in at least one of the groups $(p_7, p_0, p_1)$, $(p_1, p_2, p_3)$, $(p_3, p_4, p_5)$, or $(p_5, p_6, p_7)$ is set. This check can be implemented in a straightforward manner by encoding the occupancy of $p_0$–$p_7$ in a binary number and checking if it contains the respective cases. If the current point is located within a four-cluster, it is definitely part of an ambiguity.

The function mergeEdges combines two edges into one and updates the *edgeIdMap* accordingly. This is required when the tracing starts at a point within an edge, where it initially runs in two directions and the resulting edges are finally merged. It can also be used during postprocessing, such as when connecting edges in ambiguities (see Section 3.3.6). The function operates on the *edgeList* based on two passed *edgeIds*. The merged edge retains the smaller of the two *edgeIds* (which is technically not necessary, but provides a clear system). The two edges must share an overlapping point at one of their sides, serving as the connection point. The function ensures the correct order of the traced points by considering the position of the overlapping point. There are four cases to consider: both edges start at the same point, both edges end at the same point, the first edge starts where the second ends, or the first edge ends where the second starts.

#### 3.3.2.3 Main Function (Algorithm 1)

The main function in Algorithm 1 controls the overall operation. In Line 1, the *ambiguityMap* and *edgeIdMap* are initialized as empty data structures with the size of the input image, and the *edgeList* as an empty list. These data structures are modified by the functions preprocessAmbiguities and traceEdge. In Line 3, the function preprocessAmbiguities identifies all ambiguities and stores them in the *ambiguityMap* (see Section 3.3.2.4 for details). After that, the construction of the *ambiguityMap* is finished (see Figure 3.2b for an example), but no edges have been traced yet. This is done in the subsequent for-loop beginning at Line 4. The if-statement in Line 5 checks if the current point is an edge pixel, is not part of an ambiguity, and has not been traced yet. If these conditions are met, a new *edge* is created as an empty PointList. This list is then passed to the function traceEdge, along with the input image and the current point, initiating the tracing of the edge to which that point belongs (see Section 3.3.2.5 for details).

#### 3.3.2.4 Preprocessing (Algorithm 2)

The function preprocessAmbiguities in Algorithm 2 implements the first pass of the algorithm. In summary, once a point is identified as part of an ambiguity, all direct neighbors with the same property are iteratively registered until all points belonging

**Algorithm 1** Main

```
Initialize: $ambiguityMap$, $edgeIdMap$, $edgeList$        ▷ Global data structures mod. by Algorithms 2 and 3
function main(Image $I_{\text{in}}$)
    preprocessAmbiguities($I_{\text{in}}$)                               ▷ Algorithm 2
    for each Point $p = (x, y)$ in $I_{\text{in}}$ do
        if $I_{\text{in}}[p] > 0$ and $ambiguityMap[p]$.size == 0 and $edgeIdMap[p]$.size == 0 then
            new PointList $edge$                                ▷ New empty $edge$
            traceEdge($I_{\text{in}}$, $p$, $edge$)                           ▷ Algorithm 3
```

**Algorithm 2** Preprocessing

```
function preprocessAmbiguities(Image $I_{\text{in}}$)
    for each Point $p = (x, y)$ in $I_{\text{in}}$ do
        if $I_{\text{in}}[p] > 0$ and $ambiguityMap[p]$.size == 0 then          ▷ Unprocessed pixel found
            PointList $neighbors$ = getDirectNeighbors($I_{\text{in}}$, $p$)
            if $neighbors$.size > 2 or containsFourCluster($I_{\text{in}}$, $p$) then   ▷ Point $p$ is part of an ambiguity
                new PointList $clusterPoints$.append($p$)               ▷ Append $p$ to new empty list
                $c = 0$
                while $c < clusterPoints$.size do     ▷ Iteratively append points belonging to this ambiguity
                    PointList $neighbors$ = getDirectNeighbors($I_{\text{in}}$, $clusterPoints[c]$)
                    for each Point $p_n$ in $neighbors$ do
                        if $p_n$ is not in $clusterPoints$ then                 ▷ Avoid double entries
                            PointList $neighbors$ = getDirectNeighbors($I_{\text{in}}$, $p_n$)
                            if $neighbors$.size > 2 or containsFourCluster($I_{\text{in}}$, $p_n$) then
                                $clusterPoints$.append($p_n$)
                    $c = c + 1$
                for each Point $p_c$ in $clusterPoints$ do
                    $ambiguityMap[p_c] = clusterPoints$        ▷ Save $clusterPoints$ at each cluster point $p_c$
```

**Algorithm 3** Recursive Edge Tracing

```
Initialize: $edgeId = 0$                          ▷ Global counter incremented with each traced edge
function traceEdge(Image $I_{\text{in}}$, Point $p$, PointList $edge$)
    $edge$.append($p$)                                       ▷ Append point $p$ to current edge
    $edgeIdMap[p]$.append($edgeId$)
    PointList $neighbors$ = getDirectNeighbors($I_{\text{in}}$, $p$)
    new PointList $unvisitedNeighbors$
    if $p$ is not in ambiguity then                         ▷ Initial exploration of direct neighbors
        for each Point $p_n$ in $neighbors$ do
            if $edgeIdMap[p_n]$.size == 0 or $p_n$ is in ambiguity then
                $unvisitedNeighbors$.append($p_n$)
    if $unvisitedNeighbors$.size == 2 then              ▷ Further processing based on unvisited neighbors
        new PointList $edgePartOne$.append($p$)              ▷ Create $edgePartOne$ and append point $p$
        traceEdge($I_{\text{in}}$, $unvisitedNeighbors[0]$, $edgePartOne$)
        new PointList $edgePartTwo$.append($p$)              ▷ Create $edgePartTwo$ and append point $p$
        traceEdge($I_{\text{in}}$, $unvisitedNeighbors[1]$, $edgePartTwo$)
        mergeEdges($edgeId - 2$, $edgeId - 1$)
    else if $unvisitedNeighbors$.size == 1 then
        traceEdge($I_{\text{in}}$, $unvisitedNeighbors[0]$, $edge$)    ▷ Continue with the only unvisited neighbor
    else if $unvisitedNeighbors$.size == 0 then                              ▷ Finish current edge
        $edgeList$.append($edge$)
        $edgeId = edgeId + 1$
```

to the current ambiguity are included (corresponding to modeling Principle 4). The while-loop beginning at Line 8 checks for each direct neighbor $p_n$ if it is also part of the current ambiguity. Such points are iteratively appended to the list. If the passed point $p$ is just a SPA, no further points are appended. Comparing the counter variable $c$ with the size of the list in Line 8 ensures that the direct neighbors of each appended point are also checked, similar to a region-growing process.

#### 3.3.2.5 Edge Tracing (Algorithm 3)

This function traces the edge to which the passed point $p$ belongs and connects the edge to any ambiguities on its sides, if present. The check in Line 7 ensures that the tracing does not continue within an ambiguity. The second check in line 9 ensures that a connection pixel from an adjacent ambiguity becomes part of the edge (corresponding to modeling Principle 2; cf. Figure 3.3d, e, bottom row). After these steps, the *unvisitedNeighbors* list contains either two, one, or no points.

Consider Figure 3.3c: In case of two points in the *unvisitedNeighbors* list, which is covered in Lines 11–16, the tracing of a new *edge* has just started and the passed point $p$ is located somewhere within the *edge*. In this case, two new edges, running in the directions defined by the two points in the list, are created and traced. The passed point $p$ serves as the starting point of both edges (as *mergeEdges* requires an overlapping point). After the tracing of both edges is finished, the edges are merged in Line 16. The corresponding *edgeIds* result from incrementing the *edgeId* after the two edges have been completely traced.

Consider Figure 3.3d: In case of one point in the *unvisitedNeighbors* list, which is covered in Lines 17–18, the tracing continues in the direction of this point.

Consider Figure 3.3e: In case the *unvisitedNeighbors* list is empty, which is covered in Lines 19–21, the tracing of the current *edge* is finished. In this case, the *edge* is appended to the *edgeList*, and the *edgeId* is incremented.

### 3.3.3 Proof of Correctness

The algorithm is considered to be correct when it aligns with the modeling principles introduced in Section 3.3.1. The proof can be broken down into verifying its three core steps: that all ambiguities are correctly identified during the preprocessing, that the edge pixels remaining after the preprocessing are traced in the correct order, and that edges are correctly connected to adjacent ambiguities during the tracing.

As reflected in Lines 5 and 13 of Algorithm 2, an ambiguity point is defined as the center of a $3 \times 3$ neighborhood that has more than two direct neighbors or is located within a four-cluster. The examples shown in Figure 3.3b can be easily extended and clearly demonstrate that this criterion is correct. For a systematic proof, one can verify all $2^8 = 256$ possible configurations of the 8-neighbors. As the same criterion is applied to all direct neighbors of an ambiguity point, all points belonging to a specific ambiguity are also correctly identified.

The remaining edge pixels are traced in the correct order because the tracing direction is determined by the points returned by the function getDirectNeighbors. This function does not return diagonal neighbors that have an orthogonal neighbor, thus giving priority to orthogonal connections. Since an orthogonal neighbor has a distance of 1 from the center, and a diagonal neighbor a distance of $\sqrt{2}$, the tracing follows the shortest and therefore correct path along the edge, as shown in Figure 3.3b (top row).

Since edges adjacent to ambiguities are connected to them using the principle from the edge tracing (following the shortest path), the connection step is also correct.

### 3.3.4 Simplified Implementation

Depending on the task, the implementation can also be simplified, particularly in terms of the data stored in the *ambiguityMap* and *edgeIdMap*. In its current form, the *ambiguityMap* stores a list of points at each ambiguity point, providing direct access to every connected edge via the *edgeIdMap*. If this access and checking the size of ambiguities are not required, the *ambiguityMap* can be replaced with a simple binary map that only encodes the presence of ambiguity points. In this case, Line 17 of Algorithm 2 can directly write binary values.

In a similar manner, the *edgeIdMap* can be replaced with a binary map, if direct access to the edges passing a specific point and local searches for neighboring edges are not required. In this case, Line 4 of Algorithm 3 can also directly write binary values.

If the objective is to exclusively trace edges that start or end at ambiguities according to the principles, the *ambiguityMap* and *edgeIdMap* can be merged into a single binary map that encodes whether a point has already been processed. However, this requires some deeper modifications in the code.

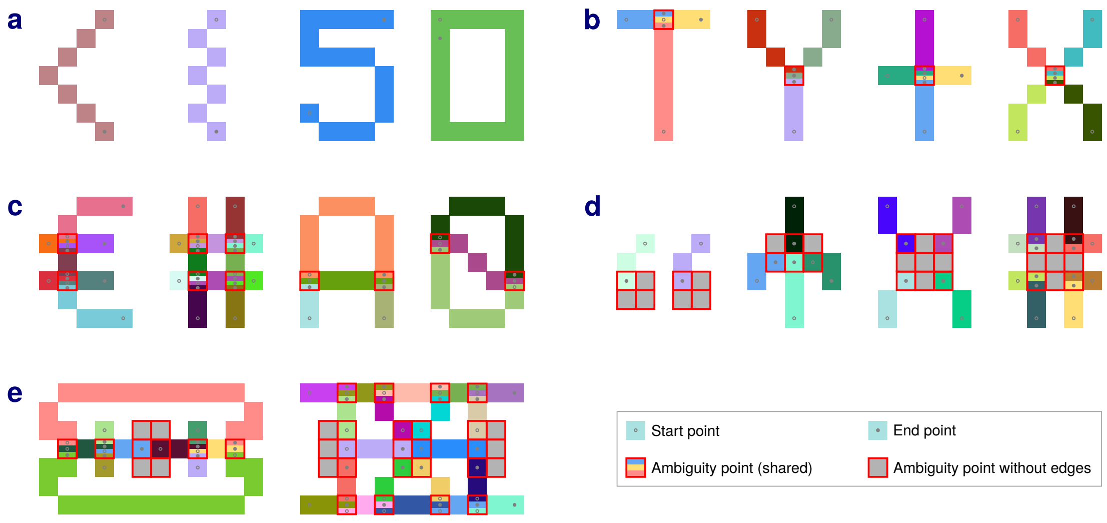


**Fig. 3.4:** Example results for some fundamental test cases, where different colors represent different edges. **a** Tracing without ambiguities. **b, c** Single-pixel ambiguities (SPAs). **d** Multi-pixel ambiguities (MPAs). **e** Complex nested cases.

## 3.3.5 Fundamental Test Cases

Figure 3.4 shows the results of the algorithm for some fundamental test cases. In all cases, all pixels from the input image have been registered and are displayed, either as traced edge pixels or as ambiguity points. Figure 3.4a shows three open edges and one closed edge without any ambiguities, so the edges are simply traced in sequential order. Figures 3.4b and 3.4c show examples for SPAs, including T-, Y-, and X-junctions. Figure 3.4d shows examples for MPAs, and Figure 3.4e complex nested cases. In summary, our method works as intended and the corresponding principles provide a clear, intuitive and natural description of all cases despite their sometimes significant complexity.

## 3.3.6 Application Examples

Figures 3.5 and 3.6 show the results of the algorithm for selected application examples and postprocessing steps. Such steps can be combined and applied in different orders in a modular fashion, depending on the specific input and objectives. Figures 3.5a–c and 3.6a are artificial examples.

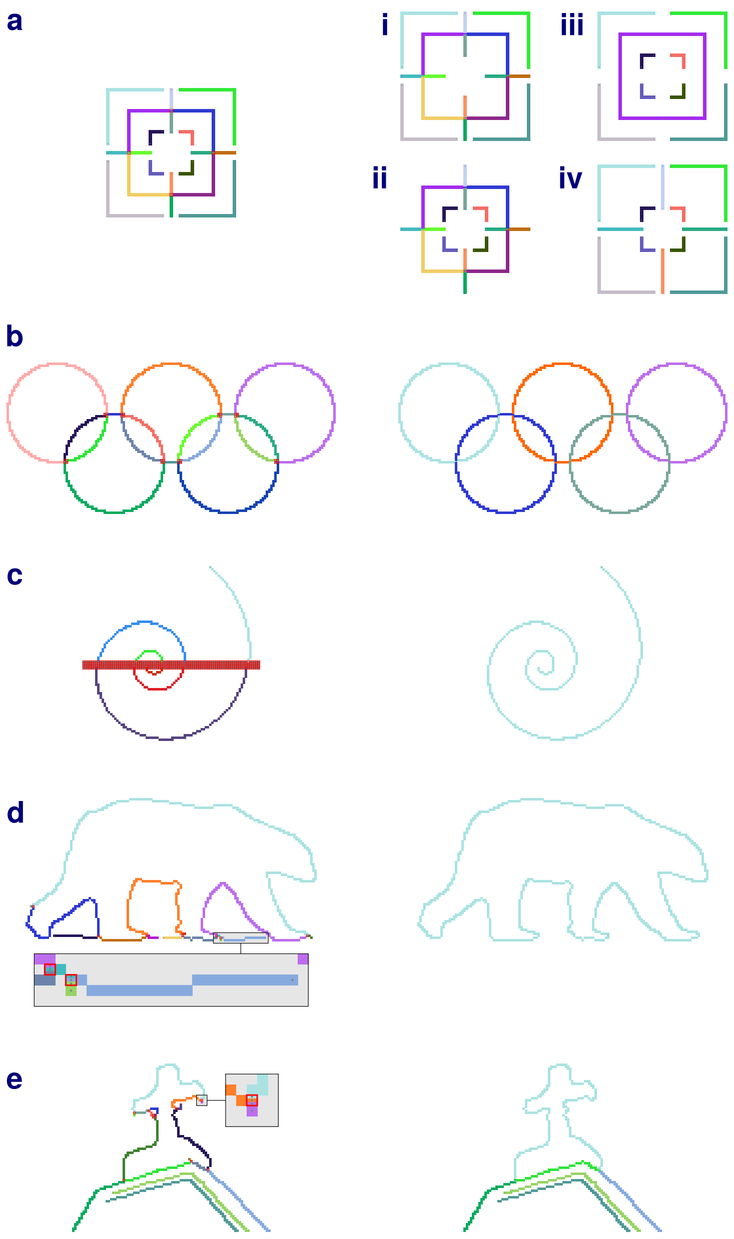


**Fig. 3.5:** Results for different examples and postprocessing steps (see text and zoom in for details). In each subfigure: Left: Initial model output. Right: Resulting edges. **a** Four different postprocessing strategies based on the edge length and connection to ambiguities (free, dangling, bridged). **b** The individual rings are separated. **c** The spiral segments are connected. **d** The bear is represented as a single, closed contour. **e** The cross is extracted as a single contour.

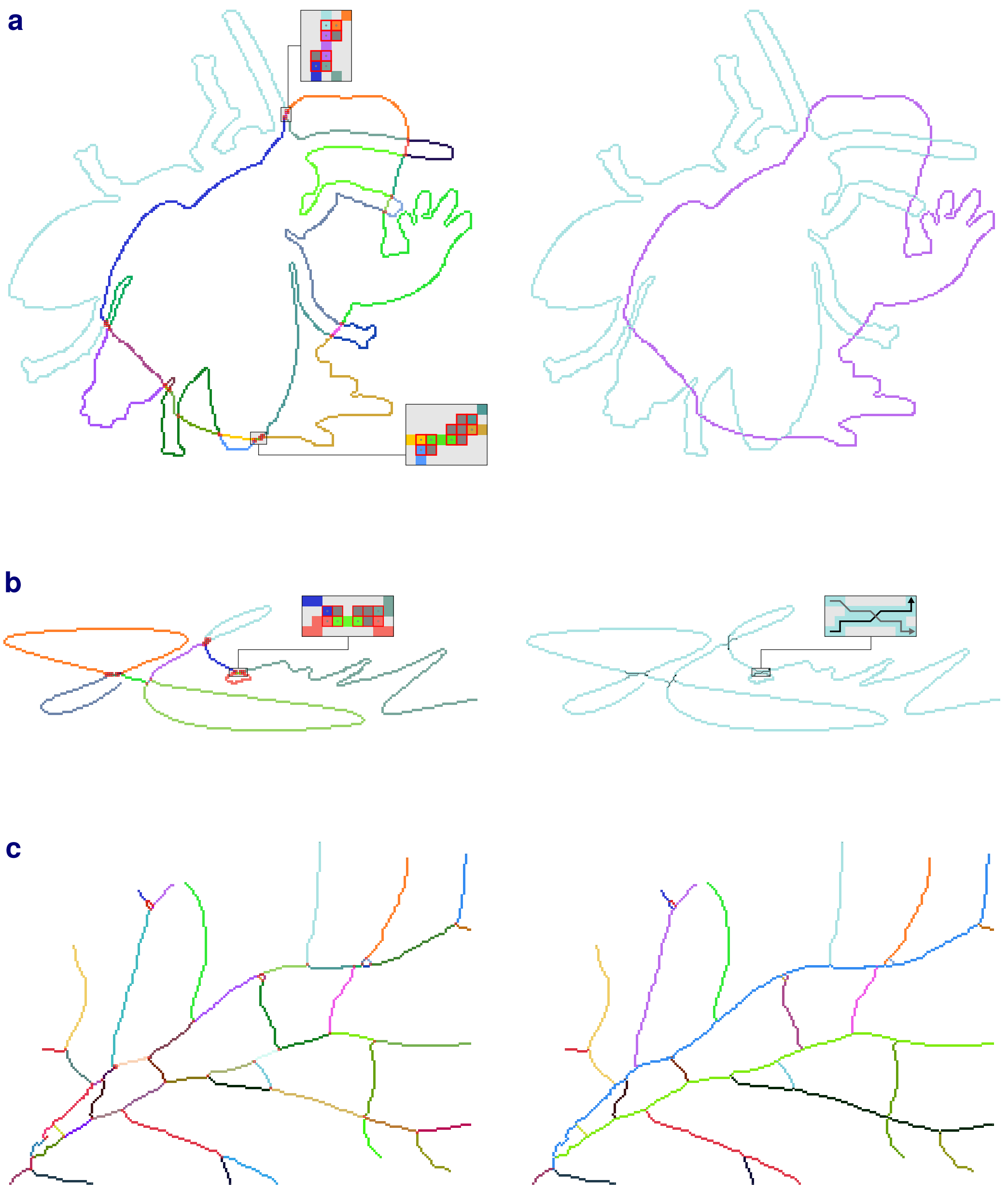


**Fig. 3.6:** Further results for different examples and postprocessing steps (see text and zoom in for details). In each subfigure: Left: Initial model output. Right: Resulting edges. **a** The two figures are separated despite complex ambiguities. **b** The path of the pen stroke followed during the signature is traced (resulting path directions indicated by arrows). **c** The main branches of the vessels are identified.

In Figure 3.5a, specific edges have been removed depending on their connection to ambiguities and length, with this information being directly accessible through the model: In Example i, all free-standing edges without connections to ambiguities and shorter than 20 pixels have been removed. In Example ii, all free-standing edges longer than 10 pixels have been removed. In Example iii, all edges that have a connection to only one ambiguity (either at the start or end, dangling) have been removed. Note that this resolves all ambiguities in the middle edge, resulting in a single, closed edge. In Example iv, all edges that have a connection to an ambiguity on both sides have been removed. Such postprocessing options could be helpful when short or long edges need to be disconnected from certain other edges, or to remove clutter from edge images.

In Figure 3.5b, the ambiguities have been resolved by connecting the respective edges using a simple cost function approach. Recall that each pixel cluster forms a single coherent ambiguity (Principle 3), so each ambiguity has been processed separately. The cost function takes into account the angle difference at which two edges approach an ambiguity and the Euclidean distance between the connection points, with the objective of connecting edges based on good continuity. The angle relative to an ambiguity is computed using a simple least squares line fit based on the last $N$ pixels from the connection point (in this example, 5 pixels have been used, but the exact number is not critical). An edge is connected to another edge or itself if the cost for the candidates is the lowest, provided that the cost is below a certain threshold. Formally, the cost function $C$ is defined as:

$$C = w_\theta \Delta\theta + w_d d\,, \tag{3.1}$$

where $\Delta\theta$ is the angle difference, $d$ is the Euclidean distance between the connection points, and $w_\theta$ and $w_d$ are weighting factors to control the contributions of continuity and proximity in the edge connection process. The process continues until all edges connected an ambiguity have been checked. Note that this approach is simple and effective, but not necessarily optimal in terms of all possible edge connections, as the next best match is directly taken. Edges are connected by straight lines between the connection points (which are start or end points), generated using the Bresenham line algorithm (Bresenham, 1965). To improve the fit, a method that takes into account the edge paths could be employed, such as interpolation using Euler spirals (Connor and Krivodonova, 2014).

Figure 3.5c has also been processed using the cost function approach. However, in this example, all edges are connected to the same large ambiguity. In the cost function, the Euclidean distance between the connection points has been weighted higher than the angle difference ($w_\theta \ll w_d$, so that larger distances lead to higher costs) to prioritize the connection of opposite edges.

Figure 3.5d shows a region cropped from image 100007 in the BSDS after applying the Canny edge detector. In this example, the bear cannot be directly analyzed as one coherent contour, as it is connected to some short edges caused by detection artifacts and some additional edges caused by shadows in the lower area. From the model output, all dangling edges shorter than 30 pixels (the exact number is not critical) have been removed twice in succession. Consider the magnified region in Figure 3.5d: Repeating this process twice is required because, after the first run, only the two dangling edges are removed, leading to one remaining dangling edge. In principle, such a process could be repeated until no more changes occur. Another strategy here and in general could be to analyze connections across multiple ambiguities to see if this results in closed contours, because it cannot be guaranteed that additional edges are always just dangling.

Figure 3.5e shows a region cropped from image 118035 in the BSDS after applying the Canny edge detector. In this example, all edges shorter than 6 pixels have been removed, independent of their connections to ambiguities. The contour of the cross was initially connected to some short edges, which have been removed, as shown in the magnified region, but also to an edge with connections to ambiguities on both sides.

Figure 3.6a shows an artificial example with many complex ambiguities. As shown in the magnified regions, ambiguities can be easily located close to each other, and due to the several edges converging at such points, such ambiguities should be considered together. Therefore, the first postprocessing step has been to merge ambiguities that are connected by edges with a length of 3 pixels into one combined ambiguity, so that all connected edges can be accessed and connected (the corresponding function is provided with our implementation). Next, connected edges at ambiguities have been processed using the previous cost function approach. In summary, only two postprocessing steps have been required to separate the two figures. As a result, the model could be helpful for figure-ground segmentation and similar tasks.

Figure 3.6b shows an example obtained by threshold-based binarization of image 39_24 from the CEDAR signature dataset (University at Buffalo, 2007). The example has also been postprocessed using the previous cost function approach. Additionally, the order of the resulting edge has been reversed so that the start point corresponds to the start of the pen stroke. A characteristic difference from Figure 3.6a is that it is a single self-intersecting contour rather than two mutually intersecting contours.

Figure 3.6c shows a region cropped from image 21 in the DRIVE retina image dataset (Staal et al., 2004) after applying binarization and morphological skeletonization. In the first step, all edges shorter than 3 pixels have been removed. Next, the cost function approach from the previous examples has been used. The next step could be to convert the connections of main and sub-vessels into a graph representation to obtain a model with data for medical purposes.

## 3.4 Evaluation

In Figure 3.1, CCL, MNT, and FCM have already been compared with the method developed in this work. For further insights into the corresponding component decomposition, redundancy, and missing pixels, a dataset with binary edge images based on the BSDS has been created and the results have been analyzed.

### 3.4.1 Dataset Construction

For each of the 500 images from the BSDS, four binary edge images have been created. An example is shown in Figure 3.7. For this purpose, one of the manual binary annotations (provided with the dataset) and the result of the gPb-owt-ucm method (Globalized Probability of Boundary with Oriented Watershed Transform and Ultrametric Contour Map; Xie and Tu, 2015; also provided with the dataset) have been used. Furthermore, the Canny edge detector (Canny, 1986) and the HED method (Holistically-Nested Edge Detection; Xie and Tu, 2015) have been applied to each image. Since multiple manual annotations are provided per image (five on average), one annotation has been randomly selected per image. The images from the gPb-owt-ucm method have twice the dimensions of the original input images and the other binary edge images ($481 \times 321$ px vs. $963 \times 643$ px). The Canny edge detector directly provides binary edge images through its included non-maximum suppression and hysteresis thresholding. Since the HED method produces broad, non-binary edges, the Zhang-Suen thinning algorithm (T. Y. Zhang and Suen, 1984) has been applied to obtain the corresponding binary edge images. As gPb-owt-ucm is a classical learning-based method, Canny a traditional gradient-based method, and HED a deep learning-based method, a representative range of edge detection methods is covered. The following evaluation compares the different methods from a general perspective without focusing on specific tasks. Which method is best generally depends on the specific task.

### 3.4.2 Method Characteristics Overview

CCL determines one component for each set of connected pixels so that each pixel from the input image belongs to only one connected component. The method does not lead

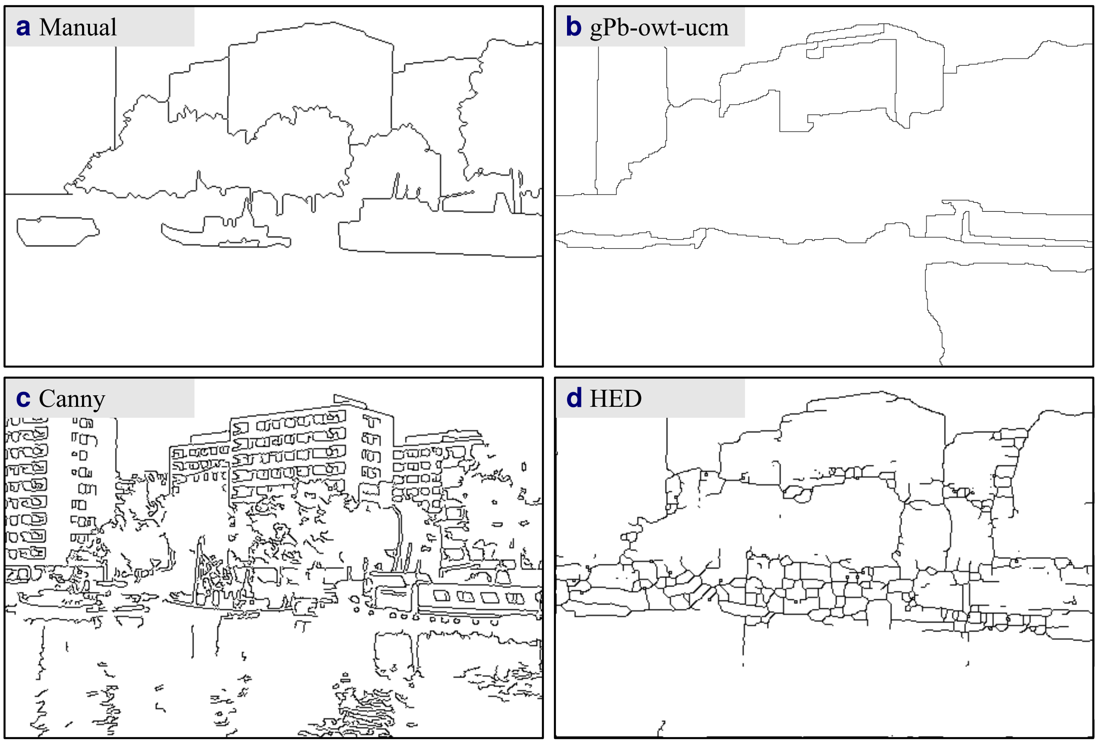


**Fig. 3.7:** Example test image from the dataset for the evaluation based on image 78098 from the BSDS (intensities inverted). **a** Manual binary annotation provided with the dataset. **b** Result of the gPb-owt-ucm provided with the dataset (scaled by a factor of 0.5). **c** Canny edge detector result. **d** HED edge detector (Xie and Tu, 2015) result with an additional thinning step for binarization. See text for further details.

to any missing pixels. In the following evaluation, CCL is used as the baseline method, as it provides information about the structure and complexity of the input image. In Figure 3.1b, for example, CCL identifies only one connected component.

MNT determines edges by following around the outer boundary of each connected component and therefore leads to one edge per component. In comparison, FCM and the method developed in this work can segment each component into several edges (and ambiguities). MNT can trace the same pixel twice in the same edge and lead to missing pixels, as shown in Figure 3.1c.

FCM can determine several edges for each connected component and also detects hierarchies between edges (parents and children), where the parent edges are essentially identical to the MNT results. FCM can trace the same pixel twice in the same edge, trace the same pixel in different edges, and also lead to missing pixels, as shown in Figure 3.1d.

The method from this work can determine several (meaningful) edges and ambiguities for each connected component. The same pixel can be traced in different edges (in that case, it is an ambiguity point), but unlike MNT and FCM, not in the same edge. The method from this work does not lead to any missing pixels, as each pixel is either part of an edge or ambiguity (or both), as shown in Figure 3.1e.

### 3.4.3 Method Analysis

In Tables 3.1–3.4, segments refer to the main entities extracted by each method. In CCL, one segment corresponds to one connected component, whereas in MNT, FCM, and the method from this work, to one edge. In the method from this work, ambiguity points without edges are not considered as segments, as they represent potential connection options rather than meaningful edges. Tables 3.1–3.3 include the results for Figure 3.1 to facilitate the interpretation of the data presented.

Table 3.1 shows the average number of pixels per segment for each method across the different edge detectors. To compute the values, the total number of segment pixels has been counted and divided by the number of segments. In Figure 3.1, for example, the method from this work has captured 52 pixels in 9 segments ($52/9 \approx 5.8$). MNT leads to the highest values as it traces certain pixels twice per segment (edge). In summary, the method from this work provides the smallest segments and the most detailed decomposition into edges.

Table 3.2 confirms these results from another perspective. Here, the total number of connected components has been counted and divided by the number of segments. In Figure 3.1, for example, the method from this work has traced 9 edges for the given component. CCL and MNT naturally have a value of $1.00$, since both methods provide exactly one segment per component.

Table 3.3 shows the total number of segment pixels in relation to the total number of edge pixels (pixels labeled 1 in the input image). In Figure 3.1, for example, the method from this work has captured 52 segment pixels and the input image contains 47 edge pixels ($52/47 \approx 1.11$). The values are interpreted here as a measure of redundancy and missing pixels, with smaller values indicating lower redundancy and a higher number of missing pixels. In general, good values are close to $1.00$. MNT has missed certain pixels so that the value is even smaller than $1.00$. In summary, the method from this work provides the lowest redundancy.

Table 3.4 shows the number of segments to which each segment pixel has been assigned (on average over the entire dataset). In Figure 3.1, for example, the method from this

| Method | Avrg. No. Pixels per Segment (px) | | | | |
|---|---|---|---|---|---|
| | Fig. 3.1 | Anno | gPb | Canny | HED |
| CCL | 47.0 | 597.2 | 2178.7 | 46.9 | 95.8 |
| MNT | 46.0 | 736.1 | 2441.0 | 69.1 | 114.9 |
| FCM | 22.0 | 208.1 | 422.0 | 58.0 | 86.5 |
| This work | **5.8** | **39.6** | **78.3** | **9.9** | **15.8** |

**Tab. 3.1:** Pixels per segment, smallest values bold.

| Method | Avrg. No. Segments per CC | | | | |
|---|---|---|---|---|---|
| | Fig. 3.1 | Anno | gPb | Canny | HED |
| CCL | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| MNT | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| FCM | 3.00 | 5.75 | 9.25 | 1.29 | 2.01 |
| This work | **9.00** | **15.47** | **28.27** | **5.12** | **6.46** |

**Tab. 3.2:** Segments per CC, highest values bold.

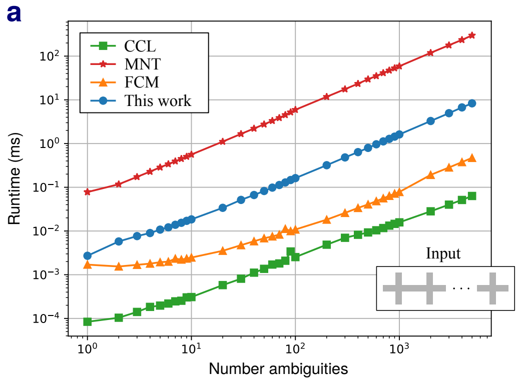
| Method | No. Segment Pixels vs. Edge Pixels | | | | |
|---|---|---|---|---|---|
| | Fig. 3.1 | Anno | gPb | Canny | HED |
| CCL | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| MNT | **0.98** | 1.23 | 1.12 | 1.47 | 1.20 |
| FCM | 1.41 | 2.00 | 1.79 | 1.59 | 1.81 |
| This work | 1.11 | **1.03** | **1.02** | **1.08** | **1.06** |

**Tab. 3.3:** Segment vs. edge pixels, closest to CCL bold.

| Method | Pixel to Segment Assignment (%) | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | >3 |
| CCL | 0.00 | 100.00 | – | – | – |
| MNT | 14.26 | 85.74 | – | – | – |
| FCM | 0.03 | 68.31 | 31.56 | 0.11 | 0.00 |
| This work | 0.33 | 96.95 | 0.31 | 2.40 | 0.02 |

**Tab. 3.4:** Assignment of pixels to how many segments.

work has traced 52 segment pixels, and 49 of them are in only one edge ($49/52 \approx 94.23\,\%$; not shown in the table). The "–" indicate that CCL and MNT cannot assign pixels to more than one element. The proportion assigned to zero segments in the method from this work corresponds to the number of ambiguity points without edges. Note that the method from this work does still capture these points; they simply do not



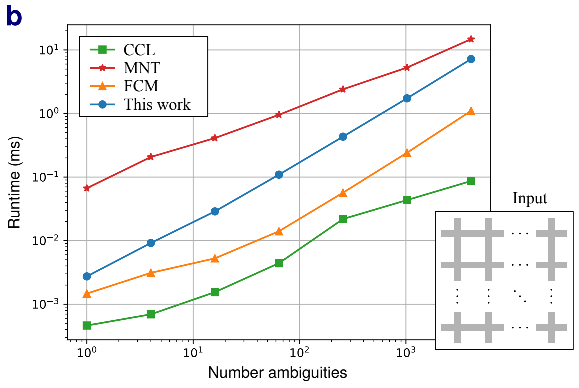


**Fig. 3.8:** Runtime analysis of the different methods for an increasing number of ambiguities (connected $5 \times 5$ px crosses, see bottom right of the subfigures). CCL, FCM, and the method from this work have been implemented in C++, and MNT in Python. **a** Increasing number of ambiguities in a row. **b** Increasing number of ambiguities in a square pattern.

contain any edges. In summary, the method from this work assigns most pixels to a single segment.

### 3.4.4 Runtime Analysis

To analyze the runtime of the different methods, regular patterns with an increasing number of ambiguities have been created, and the mean execution times have been calculated over 10,000 runs for each method (Processor: Intel i7-13700K). The results indicate an approximately linear relationship for all methods, both for an arrangement in a row (Figure 3.8a) and in a square pattern (Figure 3.8b). Concerning the method from this work, this is plausible because each additional ambiguity in the test pattern adds the same number of operations, both during the preprocessing and edge tracing. Note that the time is measured in milliseconds. Processing a typical dataset image with the method from this work takes $< 2$ ms, it is therefore real-time capable. Depending on the edge detection method, there are approximately 50 (Manual annotation) to 500 (Canny) ambiguities in a dataset image.

## 3.5 Summary and Final Remarks

Despite using only four straightforward principles (see Section 3.3.1), the ambiguity model can handle complex structures in binary edge images in an intuitive and effective manner. With the specialized design of the method, ambiguities can be resolved in a more direct manner compared to existing methods (without using graphs, etc.). It has been found that the model is a natural extension of the concept of single-pixel ambiguities. The implementation is divided into a preprocessing and an edge tracing part, which provides clear operational control and helped to formulate a proof of correctness. It has been shown that the method can effectively resolve complex ambiguities in different application examples. This demonstrates the potential of the method to extract coherent object contours from binary edge images, which can be determined using modern deep-learning-based methods (cf. Section 2.2.6). Compared to others, the method from this work provides the most detailed decomposition of binary edge images into meaningful segments while also reducing redundancy (double reading of edge pixels). While the method can also process regions, it is not intended for this purpose and simply identifies each region as one coherent ambiguity. In future works, the method could be applied to different tasks on larger datasets or used to generate training data for deep learning-based feature extraction methods (cf. Section 5.2).

A rather theoretical limitation of the method is the possible depth of recursion (which is not specific to the method, but a general characteristic of recursions). Since recursion is only used for the edge tracing, the maximum depth to be handled corresponds to the maximum edge length.

4

# Local Scale-Invariant Contour Features

As discussed in Section 2.2, assigning characteristic scales to local image features is a widely adopted approach to obtain scale-invariant feature descriptions. The methods discussed in that section are appearance-based, operating directly on the gray values in the image plane. Nonetheless, as discussed in Section 2.1.1, effective object recognition is facilitated by the integration of different feature types, particularly shape features derived from object contours. However, most shape description methods in the literature are global approaches, meaning that contours are described as coherent entities rather than based on explicit local features. In particular, there is no robust methodology for assigning characteristic scales to local contour features as there is for appearance-based features. In this chapter, a new method is presented to address this problem. The objective is to facilitate the integration of contour features with appearance-based features in a modular manner.

In Section 4.1, the importance of handling image structures at different scales is discussed. Furthermore, an overview of scale-space theory is provided, including 1D and 2D scale-space representations, fundamental axioms, and computational aspects. In Section 4.2, related work on shape description techniques is reviewed, with a focus on curvature scale-space (CSS) methods. In Section 4.3, the method developed in this work is presented in detail: starting from the limitations of simple approaches, the construction of the CSS representation is described, along with the detection and tracing of curvature extrema and the assignment of characteristic scales. Furthermore, the method is extended to open contours and a box filter approximation for the CSS computation is introduced. In Section 4.4, the method is evaluated under various transformations, and the section concludes with a brief summary.

## 4.1 Scale-Space Theory

Scale-space theory is the backbone of the method developed in this work to extract the local contour features and of many other feature extraction methods, such as SIFT. This section outlines the main ideas of this theory, its formalization, and its relevance in the field of vision systems and object recognition. Due to the mathematical nature of the theory, it is presented with a detailed formalization, such as by clearly defining function domains.

## 4.1.1 Problem Characterization

Handling different scales is a fundamental problem for any advanced vision system, whether biological or artificial, and an appropriate formalization helps to address this problem in a systematic manner. The problem is twofold: First, real-world objects are built based on structures of different scales (Ter Haar Romeny (1996) describes this as “the multiscale nature of things”), and the scale of observation determines which of these structures can be resolved as meaningful entities (for instance, observing an object at the nanometer level normally reveals entirely different structures than at the kilometer level). Second, the size of an object in the image plane changes with the distance to the vision system (zooming excluded), although its physical size does not change. Both aspects have to be considered to make a vision system flexible.

To illustrate the first aspect, the literature often refers to the entities of a tree (e.g., Koenderink, 1984; Ter Haar Romeny, 1996; Lindeberg, 2008): seen from a distance, a treetop may appear as one coherent round or cylindrical entity. On closer observation, individual leaves become visible, and in turn their texture when going even closer. Another example is given in (Lindeberg, 1994a): On closer observation of a cloud, individual droplets become visible, and in turn their water molecules. Hence, the structures are organized in a hierarchical manner. The popular short film *Powers of Ten* by C. Eames and R. Eames (1977) illustrates that such considerations can be extended to structures ranging from subatomic up to cosmic scales. The scale of observation depends on the optical instrument used (e.g., eye, camera) and can be extended with additional instruments (e.g., microscopes, telescopes). The problem is that vision systems have to deal with a vast diversity of different input images and often without prior knowledge about the content. Furthermore, the initial stages of processing should be flexible enough to enable subsequent stages to solve many different tasks. Therefore, it remains unclear which image structures should be considered as meaningful entities. All scales (i.e., sizes or spatial extents of image structures) could be interesting so that all image structures should be considered equally important in the initial stages of processing.

Scale-space theory is a formalized framework to analyze image structures at multiple scales based on smoothing operations to successively suppress fine-scale information (including noise). In other words, image structures are analyzed at different levels of detail (abstraction) representing different scales of observation. The stronger the smoothing, the higher the scale. For example, smoothing an image region that shows the leaves of a tree may lead to one coherent image structure (a treetop or a part of it). In principle, meaningful entities at different scales can be identified in a bottom-up manner. This process of primitive grouping also adapts to size changes due to distance changes (the second aspect described above). Similar processes can be found in the human visual system (cf. Section 2.1.3). In this perspective, Ter Haar Romeny (2003) uses the headline "Scale-space theory is biologically motivated computer vision", and Lindeberg (2008) notes that the theory "has been developed by the computer vision community with complementary motivations from physics and biologic vision."

Of course, the scale range in an image is bounded so that only structures within that range can be resolved. The lower bound is the *inner scale*, corresponding to the smallest visible element, which is the size of a pixel in the case of a digital image. The upper bound is the *outer scale*, corresponding to the largest visible element, which is the finite size of the entire digital image. As noted in the literature (e.g., Lindeberg and Ter Haar Romeny, 1994; Dabiri and Blaschke, 2019), inner and outer scales can also be assigned to photographic images (considering the grain size in the emulsion), optical instruments and their sensors, and also directly to real-world objects and their entities.

To extract information from an image, such as the location of features or objects, it is necessary to interact with the data using finite windows. Without specific prior knowledge, all possible window sizes between the inner and outer scale have to be considered (from a single pixel to the entire digital image) at all locations in an image. Such an approach is computationally expensive and, without additional steps, not suitable for practical applications. In fact, evaluating windows with many different sizes and aspect ratios at different locations is a common step in object detection methods and is only feasible by using specific fast filter operations or other optimization strategies (e.g., Kalal et al., 2012). As Zou et al. (2023) note, "multiscale detection of objects with different sizes and aspect ratios is one of the most technical challenges in object detection." Also CNNs for object detection use many image regions from an input image, which are then individually classified. However, these image regions must first be proposed in a way that objects are accurately captured (e.g., Zitnick and Dollár, 2014; Hosang et al., 2016).

An alternative approach is to use the smoothing operations to analyze image structures at multiple scales, as this leads to different levels of abstraction with locally stable states (identifying significant structures through a bottom-up approach). Based on the smoothing results, window sizes and operations can be directly adapted to the local image structure. Lindeberg (1994a) characterizes this as a principle "for guiding the focus-of-attention and tuning other early visual processes [...] to simplify their tasks."

This is an interesting property for tasks such as feature detection and image segmentation in a robust and scale-invariant manner (two main objectives for the contour features presented in this work). As discussed on Section 4.1.3, it can be shown that Gaussian and Gaussian derivative kernels are the only appropriate smoothing operators for such purposes.

Before going deeper into the theory for 2D images, some interesting side notes: while the structures of real-world objects in images can only be resolved over a specific range of scales, ideal mathematical entities, such as points or lines, are independent of the scale of observation. As Ter Haar Romeny (1996) notes, the characteristic difference is that physical measurements are associated with units (images are formed based on physical light measurements). Furthermore, scale considerations (with adapted scale concepts) also play an important role in many other disciplines. For example, Dabiri and Blaschke (2019) identified seven types of scale (including the scale of observation) in the context of geoscience disciplines based on a literature review and interviews with scientists. As described by Horton (2021), other domains are physics, economics, climate, philosophy, cartography, etc. For physics, the authors notes that scales refer to defined size domains, where "each scale is a conventionally derived slice of reality [. . . ] with characteristic entities and dynamics" (pp. 15–16), emphasizing that even physical laws are scale-dependent.

### 4.1.2 Scale-Space Representations

As discussed above, the significance of scale-space representations is to represent signals at multiple scales, to make their multi-scale characteristics explicit, and to simplify later processing steps. In the literature, the theoretical foundations for scale-space representations are usually derived for continuous signals. By this means, continuous mathematical operations, such as differentiation, integration, and Fourier transform, can be directly applied, which simplifies the theoretical analysis of specific properties. Furthermore, the theory can be directly linked to differential equations, particularly the diffusion equation. In practical applications and this work, however, discrete signals have to be processed. Section 4.1.4 summarizes corresponding considerations for a computational implementation of scale-space representations. This work focuses on 1D and 2D signals (contours and images) so that the following discussions are restricted to these cases. Extensions to higher (arbitrary) dimensions and spatio-temporal data are possible and described in (Lindeberg, 2013).

#### 4.1.2.1 1D Signals

Following the formalization in (Lindeberg, 1994a, p. 11), the current scale level (amount of smoothing) is defined by the scale parameter $t \in \mathbb{R}^0_+$. Given a 1D signal $f\colon \mathbb{R} \to \mathbb{R}$, its scale-space representation $L\colon \mathbb{R} \times \mathbb{R}^0_+ \to \mathbb{R}$ is defined such that the representation at zero scale is equal to the original signal:

$$L(x; t = 0) = f(x)\,. \tag{4.1}$$

Here and in other equations, the semicolon is used to separate the spatial variable(s) from the scale parameter. The representations at higher scales are obtained by convolving the original signal with 1D Gaussian functions $g\colon \mathbb{R} \times \mathbb{R}_+ \to \mathbb{R}$ with increasing scale $t$:

$$g(x; t) = \frac{1}{\sqrt{2\pi t}} \exp\left(\frac{-x^2}{2t}\right)\,. \tag{4.2}$$

In integral form, the convolution is computed as follows:

$$L(x; t) = \int_{\xi=-\infty}^{\infty} g(\xi; t) f(x - \xi)\, \mathrm{d}\xi\,. \tag{4.3}$$

This equation can be written in short as:

$$L(x; t) = g(x; t) \circledast f(x)\,, \tag{4.4}$$

where $\circledast$ denotes the continuous convolution operator. By systematically increasing $t$, the scale-space representation is obtained as a one-parameter family of signals derived from the original signal. In this context, the Gaussian $g$ is also referred to as the scale-space kernel. The standard deviation $\sigma$ of a Gaussian and the scale parameter are related by $t = \sigma^2$. The parameter $t$ is often used in the literature due to the connection of the scale-space representation to the diffusion equation, where $t$ represents time (details below).

The case of zero scale is excluded from Equation (4.2), as the Gaussian is undefined for $t = 0$. For $t \to 0$, however, the Gaussian converges to a Dirac delta function so that the convolution yields the original signal. This is mainly a theoretical consideration in terms of a well-defined scale-space representation. In practical applications, one can directly use the original signal (without the convolution) to represent the zero scale level.

An equivalent scale-space representation can be defined as the solution of the 1D heat or diffusion equation

$$\partial_t L = \frac{1}{2}\nabla^2 L = \frac{1}{2}\partial_{xx} L\,, \tag{4.5}$$

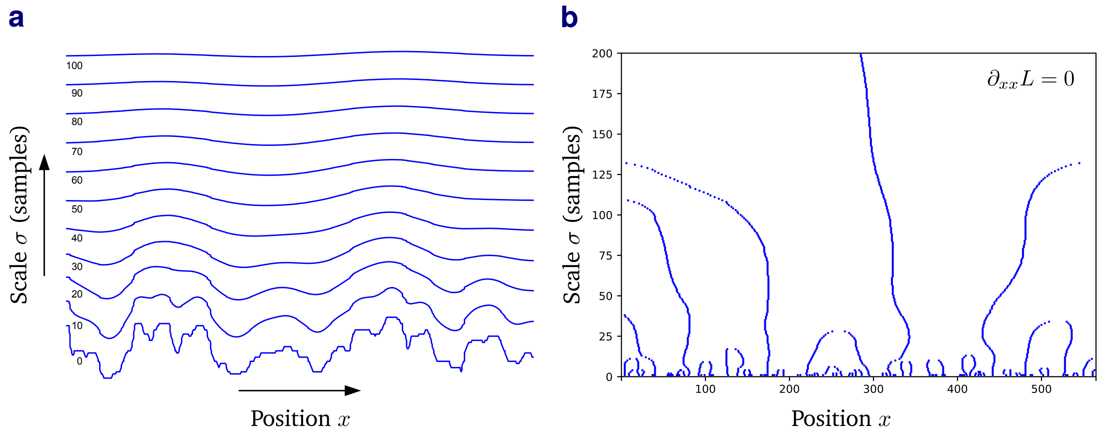


**Fig. 4.1:** Scale-space construction and analysis for a discrete 1D signal $f(x)$, implemented and extended based on (Witkin, 1983). **a** Convolution of the original signal with Gaussian kernels of increasing standard deviation $\sigma$, resulting in progressively smoothed signals. **b** Zero-crossings of $\partial_{xx}L$ (the second spatial derivative of the scale-space representation) across scales, confirming that the number of local extrema of $L$ gradually decreases, which is a fundamental property of scale-space representations (cf. Section 4.1.3.5).

where $\nabla^2$ denotes the Laplacian operator, $\partial_t$ the time derivative, $\partial_{xx}$ the second spatial derivative, and with Equation (4.1) as the initial condition. This relationship has first been described by Koenderink (1984) for 2D images with the note that the 1D case follows directly from the 2D case (refer to Section 4.1.2.2 for the 2D case, a physical interpretation, and a discussion of additional interesting characteristics of the diffusion equation for scale-space representations). The proof is straightforward: the first partial derivative of $g$ with respect to $t$ gives

$$\partial_t g(x;t) = \left(-\frac{1}{2t} + \frac{x^2}{2t^2}\right) \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) . \tag{4.6}$$

The second partial derivative of $g$ with respect to $x$ gives

$$\partial_{xx} g(x;t) = \left(-\frac{1}{t} + \frac{x^2}{t^2}\right) \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) . \tag{4.7}$$

Taking these results, substituting Equation (4.4) into Equation (4.5) (the diffusion equation), and considering the convolution derivative theorem leads to

$$f(x) \circledast \left(-\frac{1}{2t} + \frac{x^2}{2t^2}\right) \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) \\ = f(x) \circledast \frac{1}{2}\left(-\frac{1}{t} + \frac{x^2}{t^2}\right) \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) . \tag{4.8}$$

As a result, the diffusion equation is satisfied, and the independent convolution with $f$ shows that the process can be generalized to various input signals.

As Ter Haar Romeny (1996) points out, an important observation is that the spatial derivatives of the Gaussian function are also solutions of the diffusion equation. Consequently, derivative scale-space representations can be computed in the same way, and image structures at different (multiple) scales can also be analyzed in terms of their derivatives. This principle is fundamental for the development of scale-invariant feature detectors, such as the method for extracting the contour features in this work and SIFT in 2D images. As convolution commutes with differentiation, computing derivative scale-space representations can be achieved either by convolving a signal with Gaussian derivatives or by differentiating the scale-space representations directly, which can be employed for efficient algorithmic implementations, as discussed in Section 4.3.8. As outlined in (Lindeberg, 2008), the set of scale-space derivatives up to order $N$ at a specific position and scale in a 2D image is known as an $N$-jet, provides a compact characterization of the corresponding local image structure, and forms a basic type of scale-space feature. In practical applications, the case of $N = 2$ is of special interest because first-order derivatives (gradient information) are useful for identifying edges and textures, and second-order derivatives (intensity curvature) for identifying corners and blobs (as in SIFT).

#### 4.1.2.2 2D Signals

Following the formalization in (Lindeberg, 2008) (with slight modifications for better consistency with the 1D case), the scale-space representation of a given 2D signal $f\colon \mathbb{R}^2 \to \mathbb{R}$ can be defined as $L\colon \mathbb{R}^2 \times \mathbb{R}^0_+ \to \mathbb{R}$, and should be equal to the original signal in case of zero scale:

$$L(x, y; t = 0) = f(x, y)\,. \tag{4.9}$$

The representations at higher scales are obtained by convolving the original signal with 2D Gaussian functions $g\colon \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}$ with increasing scale $t$:

$$g(x, y; t) = \frac{1}{2\pi t} \exp\left(\frac{-(x^2 + y^2)}{2t}\right)\,. \tag{4.10}$$

In integral form, the convolution is computed as follows:

$$L(x, y; t) = \int_{\xi=-\infty}^{\infty} \int_{\eta=-\infty}^{\infty} g(\xi, \eta; t) f(x - \xi, y - \eta)\, \mathrm{d}\xi \mathrm{d}\eta\,. \tag{4.11}$$

This equation is often expressed in short form using the convolution operator:

$$L(x, y; t) = g(x, y; t) \circledast f(x, y) \,. \tag{4.12}$$

An equivalent scale-space representation can be defined as the solution of the 2D diffusion equation (cf. Koenderink, 1984; Weickert, 1998):

$$\partial_t L = \frac{1}{2}\nabla^2 L = \frac{1}{2}(\partial_{xx} L + \partial_{yy} L) \,, \tag{4.13}$$

with the notation analogous to the 1D case in Equation (4.5). The proof is analogous to the 1D case in Equation (4.8). This partial differential equation (PDE) is linear, space- or shift-invariant since the spatial derivatives are independent from the absolute position of $L$ (i.e., the derivatives are computed in the same manner at each position), homogeneous since all terms directly depend on $L$, and isotropic since the spatial derivatives are equally weighted in both directions. These are desirable characteristics (properties) for scale-space representations, as discussed in Section 4.1.3. The physical interpretation of this equation is that it describes the evolution of a given heat distribution (or other quantities, such as chemical concentration) without sources or sinks over time, which is particularly intuitive for functions on a plane. For example, it describes a hot spot on a metal plane gradually spreading out across the surface. In that case, $L$ represents temperature profiles instead of gray values in an image.

This work focuses on linear Gaussian scale-spaces as described above. Nonetheless, it should be noted that especially since Koenderink (1984) related scale-spaces to the diffusion equation and Perona and Malik (1990) extended the concept to an anisotropic and nonlinear form for edge detection, linear and nonlinear PDEs have been successfully adopted for many different image processing tasks, particularly smoothing and restoration (Weickert, 1998; Aubert and Kornprobst, 2006). As outlined in (Alt et al., 2023), where the authors investigate some structural connections between numerical methods for PDEs and CNN architectures, PDE-based models are compact and have a solid mathematical foundation. As a result, theoretical key properties like stability and well-posedness can be systematically analyzed. Furthermore, there are different approaches to analyze PDEs, such as numerical or variational methods, providing different perspectives to solve the specific problem at hand. The diffusion equation, as given in Equations (4.5) and (4.13), is probably the best-investigated PDE in image processing. In the Perona-Malik method mentioned above, the constant diffusion coefficient is replaced by a scalar function of the gradient magnitude with the objective to preserve edges (prefer intra over interregion smoothing). The equation is then solved in an iterative manner using a numerical method. Going further in the field of diffusion filters, using a tensor instead of a scalar diffusion coefficient provides even more control over the diffusion process with respect to the local image structure (cf. Weickert, 1998; Hund, 2009).

### 4.1.3 Scale-Space Axioms

Scale-space axioms are certain fundamental principles that have been formulated to guide the construction of scale-space representations. As motivated at the beginning of this chapter, a flexible (uncommitted) vision system not constrained to a specific domain or task should consider all image structures equally important in the first stages of processing. Furthermore, it should be possible to deal with different scales of observation. These and other properties are reflected by the axioms. Considering such axioms also leads to accurate predictions about the shape of "receptive fields observed in the retina, the LGN and the primary visual cortex (V1) of mammals" (Lindeberg, 2021).

As noted in (Lindeberg, 1994a), a list of desired principles may be long, but will also contain redundancies. While the outcome has already been described above—namely, that the process is defined by the diffusion equation, where the signal is convolved with Gaussians—the axioms provide a thorough formal basis for scale-space representations. A table listing 15 axioms and comparing their use across 14 different works can be found in (Weickert et al., 1999, p. 245), where each work employs a specific subset of these axioms. The most important and non-redundant axioms, especially in the context of scale-invariant feature detection, are linearity, spatial shift invariance, symmetry, semigroup property, causality, and scale-invariance, as discussed below. To formalize these axioms, a scale-space operator $\mathcal{T}_t$ is introduced and applied to the input signal $f$, as introduced in Sections 4.1.2.1 and 4.1.2.2, to obtain the 1D and 2D scale-space representations, respectively. Unless stated otherwise, the formalizations are based on (Lindeberg, 2013), with slight modifications for clarity and consistency across different references.

#### 4.1.3.1 Linearity

Ter Haar Romeny (2003) characterizes this principle as "no knowledge, no model, no memory". This refers to the idea that nonlinear processes often require some form of a priori knowledge or specific assumptions about the data, which contradicts the concept of uncommitted processing. An example of such a bias (nonlinearity) is the preservation of edges in the Perona-Malik method outlined in Section 4.1.2.2. Having no memory refers to the idea that the processing at each scale should be independent from previous steps. The axiom can be formalized as follows:

$$\mathcal{T}_t(a_1 f_1 + a_2 f_2) = a_1 \mathcal{T}_t(f_1) + a_2 \mathcal{T}_t(f_2)\,, \tag{4.14}$$

with constant scalars $a_1, a_2 \in \mathbb{R}$. As discussed, the diffusion equation is a linear PDE and therefore satisfies this axiom.

#### 4.1.3.2 Spatial Shift-Invariance

Also referred to as *translation invariance* or *spatial homogeneity*, this principle states that every location in an image should be processed in the same manner. In other words, the processing should only depend on the data itself. By introducing a shift operator $\mathcal{S}_{\Delta_{x,y}}$, which shifts $f$ in the $x$ and $y$-direction such that

$$\mathcal{S}_{\Delta_{x,y}}(f) = f(x - \Delta x, y - \Delta y)\,, \tag{4.15}$$

the axiom can be formalized as follows:

$$\mathcal{T}_t(\mathcal{S}_{\Delta_{x,y}}(f)) = \mathcal{S}_{\Delta_{x,y}}(\mathcal{T}_t(f))\,. \tag{4.16}$$

For tasks such as feature detection, spatial shift invariance ensures that features are detected solely based on their characteristics and not their position in the signal or image. As noted by Lindeberg (2013), who builds upon previous works in linear systems theory, combining the principles of linearity and shift-invariance leads to convolution transformations. This is also the reason why the table with the axioms in (Weickert et al., 1999) directly summarizes both principles as “convolution kernel”. The diffusion equation is consistent with this finding, as $L$ can be expressed as a convolution result, as shown in Section 4.1.2.1.

#### 4.1.3.3 Isotropy

Also referred to as *rotation invariance* or *rotational symmetry*, this principle states that all spatial directions should be processed in the same manner. The idea is that no particular orientation of an image structure should be preferred, as this would again contradict the concept of uncommitted processing. The axiom can be formalized as follows:

$$g(x, y; t) = h\,(r; t)\ , \tag{4.17}$$

for some 1D function $h\colon \mathbb{R}^0_+ \times \mathbb{R}_+ \to \mathbb{R}$, where $r = \sqrt{x^2 + y^2}$ is the radial distance to the origin of $h$. This means that the function value of $h$ respective $g$ at any point $(x, y)$ only depends on $r$, which is consistent with the use of Gaussians due to their radial symmetry.

#### 4.1.3.4 Semi-Group Property

This principle states that applying a scale-space transformation at one scale $t_1$ and then applying it again at a second scale $t_2$, should be equivalent to applying the transformation once with the sum of these scales. Given that convolution is associative when applying consecutive scale-space transformations, the axiom can be formalized as follows:

$$g(x, y; t_1) \circledast g(x, y; t_2) = g(x, y; t_1 + t_2)\,. \tag{4.18}$$

This equation is satisfied by Gaussians, as it is a well-known property that the convolution result of two Gaussians is itself a Gaussian, whose variance is the sum of the original variances:

$$\sigma^2_{g_1 \circledast g_2} = \sigma^2_{g_1} + \sigma^2_{g_2}\,. \tag{4.19}$$

A direct consequence of the semi-group property is the *cascade property*, which states that an image already transformed at a scale $t_1$ (e.g., the initial scale of observation) and to be transformed to a higher scale $t_2$ (where $t_2 \geq t_1$), only requires a transformation with the difference between the two scales:

$$L(x, y; t_2) = g(x, y; t_2 - t_1) \circledast L(x, y; t_1)\,. \tag{4.20}$$

Therefore, a consistent scale-space representation can be constructed starting from arbitrary scales, which is particularly important since the initial scale of observation is typically unknown. For a computational implementation, this property can be used to construct a scale-space representation incrementally without restarting the smoothing process from the original signal for each scale.

#### 4.1.3.5 Causality

This principle states that the representations at higher scales should be simplifications of those at lower scales. This implies that structures at higher scales should not be created or enhanced unless they are simplifications of existing structures in the original image. In short, the process should not introduce artificial features (“spurious structures”). This is a central aspect of scale-space theory, which describes how it accounts for the natural hierarchical organization of image structures, as illustrated by the tree example in Section 4.1.1. For the formalization of this principle, there is a significant difference between 1D and higher-dimensional signals: for 1D signals, it can be expressed as the requirement that the *number* of local extrema should not increase with higher scales (*non-creation* of local extrema); and for higher-dimensional signals that local extrema should not be enhanced with higher scales (*non-enhancement* of local extrema).

For a proof of the 1D case, the literature often refers to the original work by Witkin (1983): As local extrema of $L$ correspond to zero-crossings of $\partial_x L$, zero-crossings of derivatives of $L$ of any order form closed curves over scale and are never closed from below (which shows that the number of local extrema gradually decreases), as exemplarily shown for $\partial_{xx} L = 0$ in Figure 4.1b. For the 1D case, it can be shown that an appropriate scale-space kernel must be positive and unimodal in both the spatial and Fourier domains, and its integral over the entire domain should sum up to 1 to leave constant signals unaffected (see Lindeberg, 2013, for corresponding references). These characteristics are satisfied by Gaussians.

For higher-dimensional signals, the formalization has to be adapted, as Lifshitz and Pizer (1990) have presented a 2D example which shows that there are no non-trivial kernels (including Gaussians) that never increase the number of local extrema. In their example, two intensity peaks of different heights in the 2D plane are connected by a narrow ridge. During the smoothing process, the ridge vanishes faster than the peaks, which leads to two distinct local extrema where only one was apparent before (see Lindeberg, 1994a, p. 102, for an illustration). However, it is still a simplification of existing structures so that causality is preserved. In conclusion, the corresponding formalization has to be relaxed in such a way that local extrema are not enhanced at higher scales:

$$\begin{aligned} \partial_t L &\leq 0 \quad \text{at any non-degenerate local maximum,} \\ \partial_t L &\geq 0 \quad \text{at any non-degenerate local minimum.} \end{aligned} \tag{4.21}$$

The non-degenerate restriction excludes critical points such as ridges as described above, where the determinant of the Hessian matrix is zero. This formalization can be used for formal proofs and leads to the result that the scale-space kernel for higher-dimensional signals must also be a Gaussian (Lindeberg, 1997). From a mathematical perspective on the diffusion equation, the non-enhancement is reflected by the maximum principle, which is a characteristic property of parabolic PDEs (Evans, 2010). In a retroperspective of his work, Koenderink (2021) simply notes from a practical standpoint that “for a classical physicist it is entirely obvious that diffusion cannot generate, but only destroy spatial articulations.”

#### 4.1.3.6 Scale-Invariance

This principle states that the appearance and size of a feature in the scale-space representation should adjust relative to the scale parameter. This implies that when a feature is scaled (naturally, through varying scales of observation and distances, or artificially in a scale-space representation), a kernel scaled by the same factor should yield a similar response. This is a central requirement for scale-invariant feature detection, as the characteristic scale of a feature needs to be identified across different scales of observation and distances in a consistent manner. Scale-invariance requires that the convolution kernel is self-similar, which means that changing the scale parameter should yield a scaled version of the original kernel (a rescaled copy). This can be formalized as follows:

$$g(x, y; t) = \frac{1}{\varphi(t)} \bar{g} \left( \frac{x}{\varphi(t)}, \frac{y}{\varphi(t)}; 1 \right), \tag{4.22}$$

where $\varphi(t) \colon \mathbb{R}_+ \to \mathbb{R}_+$ is some function which transforms the scale parameter $t$, and $\bar{g}$ is a prototype (or reference) kernel. Therefore, scaled versions of the kernel should be obtained by scaling both its amplitude and spatial coordinates by the same factor. This relationship is inherently satisfied by Gaussians. Comparing Equations (4.10) and (4.22) shows that $\varphi(t)$ corresponds to $\sigma = \sqrt{t}$. As a result, scaling a Gaussian corresponds to changing its standard deviation by the respective factor, which is a well-known property. In feature detection, it is therefore $\sigma$ values that represent the characteristic scales at which a Gaussian or its derivatives best match an image structure (cf. Section 2.2.3).

#### 4.1.3.7 Summary

In summary, the Gaussian function and its derivatives represent a unique choice for scale-space representations in the continuous domain, which can be derived from different subsets of scale-space axioms. In particular, they provide a consistent and scale-invariant approach that is not biased towards any specific image structure, position, orientation, or scale, while preserving the natural hierarchical organization of image structures.

### 4.1.4 Computational Implementation

In practical applications and this work, it is necessary to process discrete rather than continuous signals. For this purpose, the usual approach in the literature consists of representing the Gaussian function and its derivatives by a sufficient number of samples. This approach is also followed in this work. In SIFT, for example, the Gaussian filter masks should be sufficiently large. In SURF, the second-order partial derivatives are even approximated using simple box filters (cf. Section 2.2.4.2). Concerning the number of samples, it is generally appropriate to ignore the values of a Gaussian function at a distance larger than $3\sigma$ from the mean (Gonzalez and Woods, 2018, p. 168).

Considering the constraints imposed by the formalized scale-space axioms, it is reasonable to ask whether this approach is theoretically justified without further considerations. Lindeberg (2024) provides a detailed analysis on this question. In summary, the discrete analogue of the Gaussian function and its derivatives are modified Bessel functions of integer order. However, this is only significant for very fine scales ($\sigma < 0.75$), which are below the scales relevant to this work or the feature extraction methods discussed.

## 4.2 Related Work

This section first provides an overview of established shape description methods and then focuses on curvature scale-space methods, to which the method presented in this work belongs.

### 4.2.1 Shape Description Methods

In general, 2D shape description methods can be divided into contour- and region-based methods, depending on the specific feature type used (Y. Li et al., 2015; M. Yang et al., 2008). The descriptions are required to measure the similarity of shapes, and many techniques have been developed for this purpose (D. Zhang and Lu, 2004). Traditional methods include chain codes, boundary signatures, and Fourier descriptors (Gonzalez and Woods, 2018), but these are not robust against geometric image transformations or noise, and they can only process fully segmented closed contours, which are often difficult to determine. Another well-known method is skeletonization, which reduces the contour of an object to its medial axis (skeleton) while still describing its topological and geometric shape properties (Saha et al., 2016). By analyzing the skeletons for specific singularities (shocks) and modeling their configuration, shock graphs are constructed (Sebastian et al., 2004). Bai et al. (2009) combine skeletons with contour segments to integrate local and global shape properties. Although skeletonization leads to compact shape representations, it also requires fully segmented closed contours.

Most shape description methods can also work when parts of a contour are missing. For example, regularly sampled points can be used to find point correspondences on different shapes. For this purpose, Belongie et al. (2002) introduced the *shape context* descriptor. For each sampled point, the descriptor captures the relative distribution of the surrounding points in log-polar histograms. Point correspondences are used to estimate an aligning transform, but the descriptors may not be robust when surrounding points are missing due to larger occlusions. Scale and rotation invariance can be achieved through specific normalizations. Ling and Jacobs (2007) also use shape context but replace the Euclidean distance, which is normally used to capture the relative distribution of the surrounding points, with the inner distance. This modification makes the descriptions of complex shapes with articulations more discriminative. Berg et al. (2005) also use regularly sampled points to estimate an aligning transform but employ geometric blur descriptors for matching. From a general perspective, the last two methods are examples of the use of specific problem-solving techniques, such as dynamic programming (Ling and Jacobs, 2007) and integer quadratic programming (Berg et al., 2005), to reduce the computational complexity of finding correspondences between large sets of edge points.

Several shape description methods use contour segments together with a voting scheme to verify the geometric consistency of matched segments. For example, Opelt et al. (2006) and Shotton et al. (2008) describe similar methods to learn class-specific codebooks of contour segments: they explicitly construct segments that are representative of class-specific training images. While the construction process differs between the two methods, both use chamfer matching to compare segments and boosting to consider a large number of matching combinations. The geometric consistency of matches is evaluated using a star-shaped voting model. Scale and rotation invariance are achieved through scaled and rotated codebook entries (Opelt et al., 2006) or through scale-normalized codebook entries and modified chamfer matching (Shotton et al., 2008), respectively. Due to the class-specific learning scheme, the features are not generic and are therefore difficult to reuse. Ferrari et al. (2008) exploit the connectedness of roughly straight adjacent contour segments. The authors use codebooks of connected segments together with sliding windows to classify different shapes. A specific normalization achieves scale invariance, while rotation invariance is not implemented by default.

Other shape description methods directly process edge images without explicitly following contours or contour segments. For example, Jurie and Schmid (2004) use scale-space analysis to determine circle- and arc-like structures of edge pixels as local image features. These features are scale and rotation invariant, but the specific structures may only represent certain shape classes. The descriptors used are similar to shape context (which may not be robust in cases of larger occlusions). Mikolajczyk et al. (2003) also use scale-space analysis but compute Laplacians for the edge points in edge images of different scales. The method follows a similar approach to SIFT, determining a high number of features but requiring a complex matching scheme.

The methods discussed integrate some of the most important shape description and matching techniques. The star-shaped voting models in (Opelt et al., 2006) and (Shotton et al., 2008) are examples of graph-based approaches to describe the geometric relationships of contour segments. Shape-trees are another graph-based approach used by Felzenszwalb and Schwartz (2007) to describe hierarchies of roughly straight contour segments. They also integrate elastic matching, a well-known method to measure the similarity of shapes by minimizing specific energy functions (D. Zhang and Lu, 2004). Elastic matching is closely related to aligning transforms and can also be used together with parametric contour representations, such as splines (Tuytelaars and Mikolajczyk, 2008). Codebook-based methods are also very common and represent a variant of the bag-of-features approach (Prasad, 2012). Since these methods usually only describe the frequency of features, their geometric configuration is not considered. X. Wang et al. (2014) use discrete contour evolution (Latecki and Lakämper, 1999) to determine contour segments, which is related to scale-space analysis. The segments are described using shape context descriptors and integrated with bag-of-features. Shen et al. (2016) present a similar method but use skeletal information instead of contour segments.

In this work, contour segments are extracted around curvature extrema. In general, curvature extrema are already common features in the context of shape description (Tuytelaars and Mikolajczyk, 2008). However, existing works identify curvature extrema only as corners using different variants of curvature scale-space analysis (X.-C. He and Yung, 2004; Mokhtarian and Suomela, 1998; X. Zhang et al., 2009). Compared to the method developed in this work, they do not determine a characteristic scale for each keypoint (curvature extremum), which serves as additional information to extract meaningful contour segments in a robust and scale-invariant manner. Additionally, it could be interpreted as a saliency measure in artificial visual attention systems (Tünnermann et al., 2013). As outlined, many shape description methods use roughly straight contour segments. In (Xu and Kuipers, 2011), such segments are even extracted by cutting contours at curvature extrema. Given that curvature extrema are the most informative points along object contours, this seems counterintuitive, as the corresponding descriptions do not make use of the encoded information. Additionally, it is difficult to assign robust characteristic orientations to straight segments. The contour segments from this work can be described by feature vectors for matching. Therefore, the computational complexity of aligning transforms or matching large sets of sampled points can be avoided. Furthermore, the features are generic and can be integrated with bag-of-features and other methods, and the characteristic scales can be used as additional geometric information for their description.

| Method | Scope | Feature Type | Open Contours |
|---|---|---|---|
| Standard CSS (Mokhtarian et al., 1992) | Global | Zero-crossings | No |
| Enhanced CSS (Abbasi et al., 2000) | Global | Zero-crossings | No |
| Extreme CSS (Silkan et al., 2009) | Global | Min/Max | No |
| Eigen CSS (Drew et al., 2009) | Global | CSS projections | No |
| Generalized CSS (Benkhlifa et al., 2017) | Global | Min/Max | No |
| Deep GCSS (Mziou-Sallami et al., 2023) | Global | Max. curvature | No |
| Diff. of Curvature (Kawamura et al., 2011) | Local | DoC Min/Max | Yes |
| **This work** | Local | Min/Max | Yes |

**Tab. 4.1:** Overview of representative CSS-based methods for contour-based shape description, adapted and extended based on (Kurnianggoro et al., 2018, Table 2).

### 4.2.2 Curvature Scale-Space Methods

While CSS-based methods are well-established and have a solid theoretical foundation in scale-space theory, they have not been used for assigning characteristic scales to individual curvature extrema—likely because the process is more complex than locating a distinct maximum in a signature function, as in appearance-based methods (cf. Section 2.2.3). The key idea in this work is to identify and distinguish local and global characteristics in the function profiles—an aspect that has not been explored in the literature. Details of the CSS computation steps for a given contour are presented in conjunction with the method developed in this work in Section 4.3. Note that there are also established CSS-based corner detectors (e.g., Mokhtarian and Suomela, 1998), which focus exclusively on identifying keypoints rather than providing a broader shape description or characteristic scales. These methods fall outside the scope of this comparison.

In the literature, CSS often refers to the specific method (referred to as Standard CSS here) in which curvature zero-crossings of closed contours are traced across scales to obtain a global, scale-invariant shape description (Mokhtarian and Mackworth, 1992; Abbasi et al., 1999; Mokhtarian and Bober, 2003; Berrada et al., 2011), which remains the most established and widely referenced approach in this domain. The contours are typically normalized or resampled, and the resulting representation consists of the positions of the maxima of the arcs formed by the zero-crossings. Two examples of Standard CSS representations are shown in Figure 4.2, illustrating the robustness of the approach with respect to noise. The Standard CSS method has been extended in many works, an overview of which is given in Table 4.1. In principle, one can distinguish between global and local approaches, with global approaches being much more common.

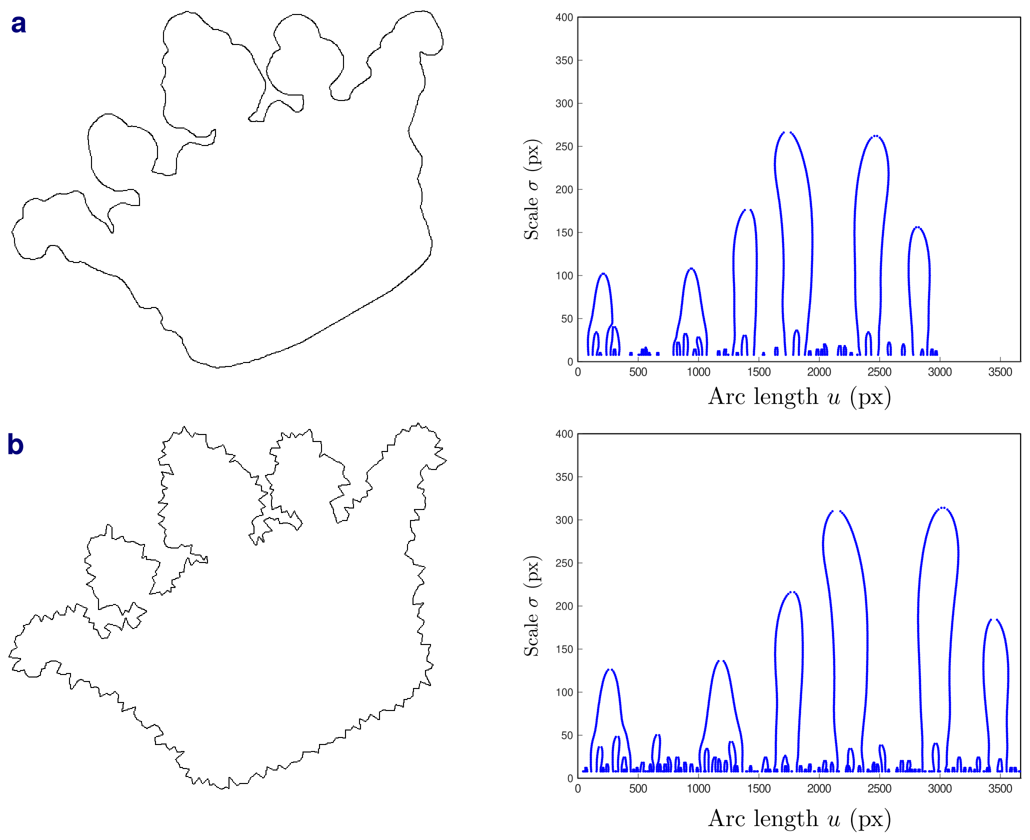


**Fig. 4.2:** Standard CSS representation without arc-length parametrization. Additional normalization steps lead to nearly identical representations. **a** Original contour. **b** Same contour with a significant amount of noise.

All global approaches in Table 4.1 require closed contours and do not assign local characteristic scales to individual keypoints. Abbasi et al. (2000) extend the Standard CSS representation by incorporating "additional information about the curvature value at different levels of scale." Silkan et al. (2009) build on the Standard CSS method by extracting curvature extrema instead of zero-crossings, but retain the global scope. Drew et al. (2009) propose a method that represents the CSS image using marginal-sum feature vectors, which are projected into an eigenspace for efficient shape comparison. Benkhlifa and Ghorbel (2017) extend the Standard CSS method by introducing a non-uniform iso-curvature parameterization. While this improves shape representation in high-curvature regions, it does not change the global nature of the method. Mziou-Sallami et al. (2023) present a deep learning model for contour classification that integrates handcrafted Generalized Curvature Scale-Space (GCSS) descriptors. Despite its robustness and explainability, the method remains global and does not assign local characteristic scales.

Besides the method presented in this work, the approach by Kawamura et al. (2011) is the only other local method in Table 4.1. It detects keypoints and their characteristic scales based on a Difference of Curvature (DoC) approach. Unlike the Standard CSS method, it avoids explicit scale tracking and also supports open contours. However, it assigns characteristic scales to DoC extrema rather than to curvature extrema, and thus not to the most informative points along object contours. In summary, the method presented in this work is the only one that assigns local characteristic scales to individual curvature extrema.

## 4.3 Keypoint Detection and Scale Assignment

While the general CSS computation employed in this work is similar to the Standard CSS method, the contours are not resampled, and a feature extraction scheme based on curvature extrema (minima and maxima) instead of zero-crossings is proposed. The general methodology is to analyze the signature functions of curvature extrema in the CSS representation. Retaining the original contour without resampling is required to determine the characteristic scales. The method developed in this work is first introduced for closed contours and then extended to open contours.

Some of the results presented in this section are part of a submission in preparation (Hennig, 2025).

### 4.3.1 Why Simple Methods are Insufficient

Since curvature extrema represent salient points along contours and can be detected in a robust manner under certain conditions, it is reasonable to ask whether characteristic scales could be assigned using a simple strategy—such as based on the nearest surrounding extrema. It is straightforward to demonstrate that this strategy has strong limitations. Consider Figure 4.3a–c: Here, the characteristic scale of each extremum has been computed as the mean Euclidean distance to the two nearest surrounding extrema. In comparison to Figure 4.3a, the contour in Figure 4.3b represents a higher-order Koch snowflake with additional fine-scale structures, while the contour in Figure 4.3c has been affected by noise.

Although the overall shape of the contours is similar, the characteristic scales of corresponding features are completely different, making them neither robust nor meaningful for practical purposes. One might first consider smoothing the contour to reduce the influence of fine-scale details. However, this raises the question of how much smoothing

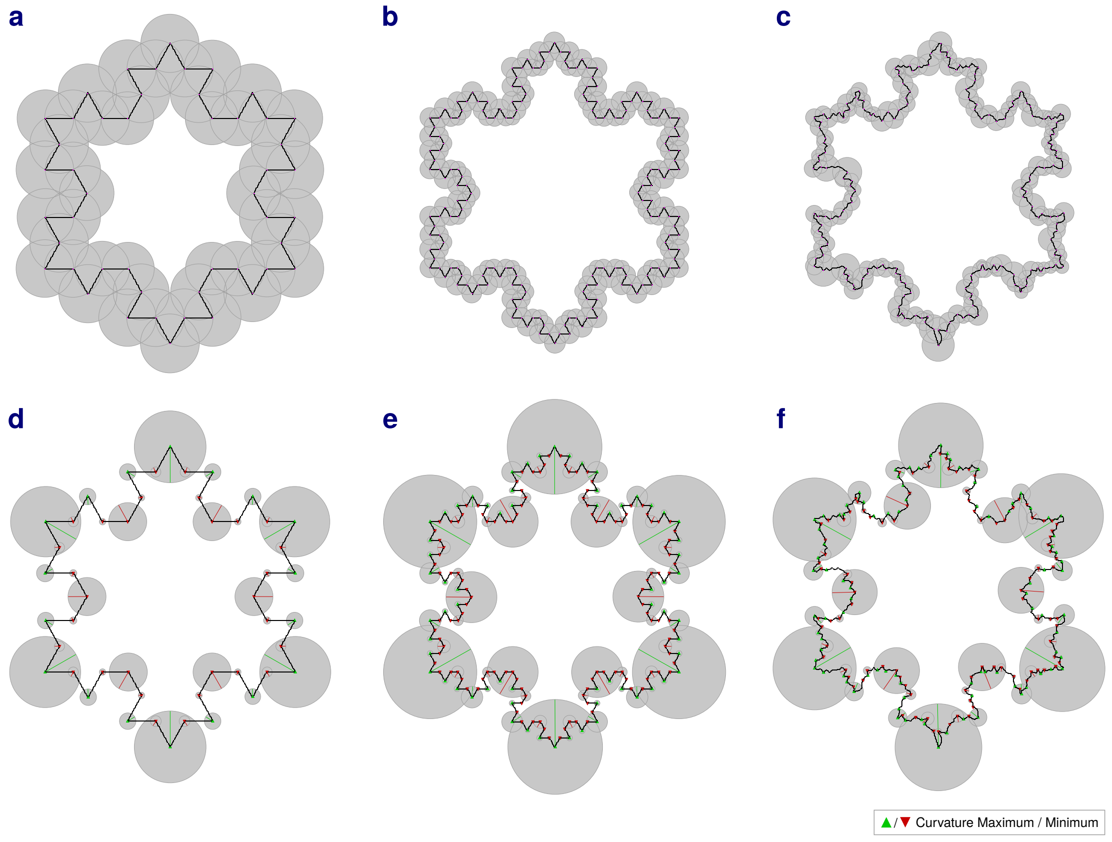


**Fig. 4.3:** Comparison of a simple method for assigning characteristic scales with the method developed in this work (applied to a Koch snowflake). **a–c** Scales determined by the mean Euclidean distance to the two nearest surrounding extrema. The results are not meaningful for practical purposes. **d–f** Method developed in this work.

is appropriate in such a single-scale approach. It may also seem natural to explore multiple levels of smoothing, and doing so in a systematic manner is already the central idea behind scale-space representations. Consider Figure 4.3d–f: Here, the scales have been determined using the CSS-based method developed in this work. Despite the additional fine-scale structures and noise, the characteristic scales correctly adapt to the overall (hierarchical) structure of the shapes. Further details about the method are described below.

Since CSS-based methods are theoretically well-grounded, this methodology is also employed in this work. However, handling contour structures at different scales can also be addressed using alternative approaches. For example, Teh and Chin (1989) compute individual regions of support by iteratively expanding initial regions based on perpendicular distance measurements, thereby adapting to local geometric properties. This approach was later refined by Pham et al. (2014). Another approach is to use alternative curvature measures—for example, based on the extreme points of height functions computed over multiple directions, angles between consecutive segments rather than individual points, or osculating circles (cf. Liu et al., 2008, and references therein).

### 4.3.2 Curvature Scale-Space Construction

Following the Standard CSS computation procedure (e.g., X.-C. He and Yung, 2004; Mokhtarian and Suomela, 1998; Berrada et al., 2011; Mziou-Sallami et al., 2023), the contour of interest is first parameterized by its discrete arc length $u \in \{0, \dots, N_\Gamma - 1\}$ to obtain a curve $\Gamma(u)$:

$$\Gamma(u) = (x(u), y(u))\,. \tag{4.23}$$

In other words, the contour consists of $N_\Gamma$ pixels. The parameterization with respect to $u$ requires that the contour is provided as an ordered sequence of connected pixels, which can be obtained using the tracing method presented in Chapter 3. To compute the CSS of this curve, it is convolved with 1D Gaussians $g$ with increasing standard deviation $\sigma \in \{\sigma_{\text{start}}, \dots, \sigma_{\text{end}}\}$, where $\sigma$ is incremented by $\sigma_{\text{step}}$:

$$g(u;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2}\right)\,. \tag{4.24}$$

This definition corresponds to Equation (4.2), adapted to the arc length $u$ and using the standard deviation $\sigma$ instead of $t$ as the scale parameter. To represent this function with high accuracy (cf. discussion in Section 4.1.4), $N_g = 2\lceil 4.5\sigma \rceil + 1$ samples are used throughout this work, where $\lceil \cdot \rceil$ denotes the ceiling function, and $N_g$ is odd to center the Gaussians on each contour pixel.

Let $x_\sigma(u;\sigma)$ and $y_\sigma(u;\sigma)$ denote the discrete convolution results with 1D Gaussians:

$$\begin{aligned} x_\sigma(u;\sigma) &= x(u) * g(u;\sigma)\,, \\ y_\sigma(u;\sigma) &= y(u) * g(u;\sigma)\,. \end{aligned} \tag{4.25}$$

Evolved (smoothed) versions of the contour are then given by

$$\Gamma_\sigma(u;\sigma) = (x_\sigma(u;\sigma), y_\sigma(u;\sigma))\,, \tag{4.26}$$

with examples shown in Figures 4.4 and 4.5. Let $\dot{x}_\sigma(u;\sigma)$ and $\ddot{x}_\sigma(u;\sigma)$ denote the first and second partial derivatives of $x_\sigma(u;\sigma)$ with respect to the arc length $u$, respectively. The same notation is used for the derivatives of $y_\sigma(u;\sigma)$. Since the contour is represented as a discrete signal, the derivatives have to be approximated using finite differences. The first derivatives are computed as:

$$\begin{aligned}\dot{x}_\sigma(u;\sigma) &= \frac{1}{2}(x_\sigma(u+1;\sigma) - x_\sigma(u-1;\sigma))\,,\\ \dot{y}_\sigma(u;\sigma) &= \frac{1}{2}(y_\sigma(u+1;\sigma) - y_\sigma(u-1;\sigma))\,.\end{aligned} \tag{4.27}$$

The second derivatives are computed as:

$$\begin{aligned}\ddot{x}_\sigma(u;\sigma) &= x_\sigma(u+1;\sigma) - 2x_\sigma(u;\sigma) + x_\sigma(u-1;\sigma)\,,\\ \ddot{y}_\sigma(u;\sigma) &= y_\sigma(u+1;\sigma) - 2y_\sigma(u;\sigma) + y_\sigma(u-1;\sigma)\,.\end{aligned} \tag{4.28}$$

Due to the convolution derivative theorem, an alternative for computing the derivatives would be to convolve the original signals with the discrete approximations of the first and second derivatives of Gaussians, respectively. However, as noted in (Lindeberg, 1993), such additional convolutions are computationally more expensive due to their larger support regions, while yielding equivalent results. Further discussions on this aspect are provided in conjunction with box filter approximations in Section 4.3.8.

Based on the derivatives, the discrete approximation of curvature $\kappa$ is computed as:

$$\kappa(u;\sigma) = \frac{\dot{x}_\sigma(u;\sigma)\ddot{y}_\sigma(u;\sigma) - \dot{y}_\sigma(u;\sigma)\ddot{x}_\sigma(u;\sigma)}{((\dot{x}_\sigma(u;\sigma))^2 + (\dot{y}_\sigma(u;\sigma))^2)^{3/2}}\,. \tag{4.29}$$

Computing $\kappa$ for increasing values of $\sigma$ and consolidating the results yields the CSS representation of the given contour $\Gamma$, where $u$, $\kappa$, and $\sigma$ are equally important, and which is further analyzed in subsequent steps to determine the keypoints and their characteristic scales.

### 4.3.3 Detection of Curvature Extrema

In general, curvature maxima are given by local maxima of $\kappa$ with positive values, and curvature minima by local minima of $\kappa$ with negative values. Local maxima with negative values and local minima with positive values do not represent curvature extrema. To detect curvature extrema, the peak detection method from (Billauer, 2012) has been adapted for this work. The method operates based on the principle that curvature extrema are the highest or lowest points relative to the most recent extrema. Simply searching for zero-crossings of the first derivative—which is a standard method for this task in the continuous domain—leads to many false detections, particularly at lower scales, due to the discrete nature of the data and its inherent noise. However, the principal approach used is not critical for the method developed in this work, provided the extrema are correctly identified.

Using the peak detection method, a curvature extremum is identified if $\kappa$ is the maximum or minimum value, is preceded by a value differing by at least $\Delta\kappa = 0.0001$, and has the correct sign. This approach ensures that an extremum is only identified if the change in curvature is significant enough. The value for $\Delta\kappa$ has been determined experimentally and accounts for the small differences in the curvature values, particularly at higher scales (smoothing a contour generally reduces its curvature). Extrema with very low $\kappa$ values are disregarded at low scales ($\sigma \leq 10\,\mathrm{px}$) to exclude noise-induced extrema that make it difficult to determine the positions $u$ of significant extrema along the original contour. For a parameterized *closed* contour, curvature extrema must be detected across both the first and last values of $\Gamma(u)$. This is required because curvature extrema can shift their position $u$ as the curve evolves. For each $\sigma$, the set of curvature maxima is denoted by $\{u^+_{\sigma,i}\}$, and the set of curvature minima is denoted by $\{u^-_{\sigma,i}\}$, where $i$ indexes the individual extrema at the given scale.

Many of the following discussions are based on a symmetric contour (Koch snowflake) and a natural contour (a fish) to demonstrate the underlying principles. However, a large variety of different shapes has been considered during the development. Examples of detected curvature extrema using the peak detection method are shown in Figures 4.4 and 4.5. As can be observed, the extrema form signature functions in the CSS representation, which are extracted for further analysis (see Section 4.3.4). The signature functions are conceptually similar to those discussed in connection with blob features in Section 2.2.3, but they exhibit a more complex behavior, as discussed in Section 4.3.5. Analyzing the evolution of the contours and the detected extrema reveals certain specific properties: first, while the number of contour points $N_\Gamma$ remains constant, the geometric lengths become shorter with increasing scales (*arc length evolution* or *curve-shortening flow*, eventually emerging into a singularity at extreme scales, cf. Mokhtarian and Mackworth, 1992). Second, the curves become increasingly convex, and only a few extrema remain at higher scales (here in each case two, marked with ① and ②, respectively), while the other extrema disappear at specific scales. Third, neighboring extrema can merge into a single extremum, as for example the two maxima marked with ① and ③ in Figure 4.5 (see Rattarangsi and Chin, 1992, for further discussions of this aspect). These and other observations are employed in the methodology developed in this work to determine the characteristic scales.

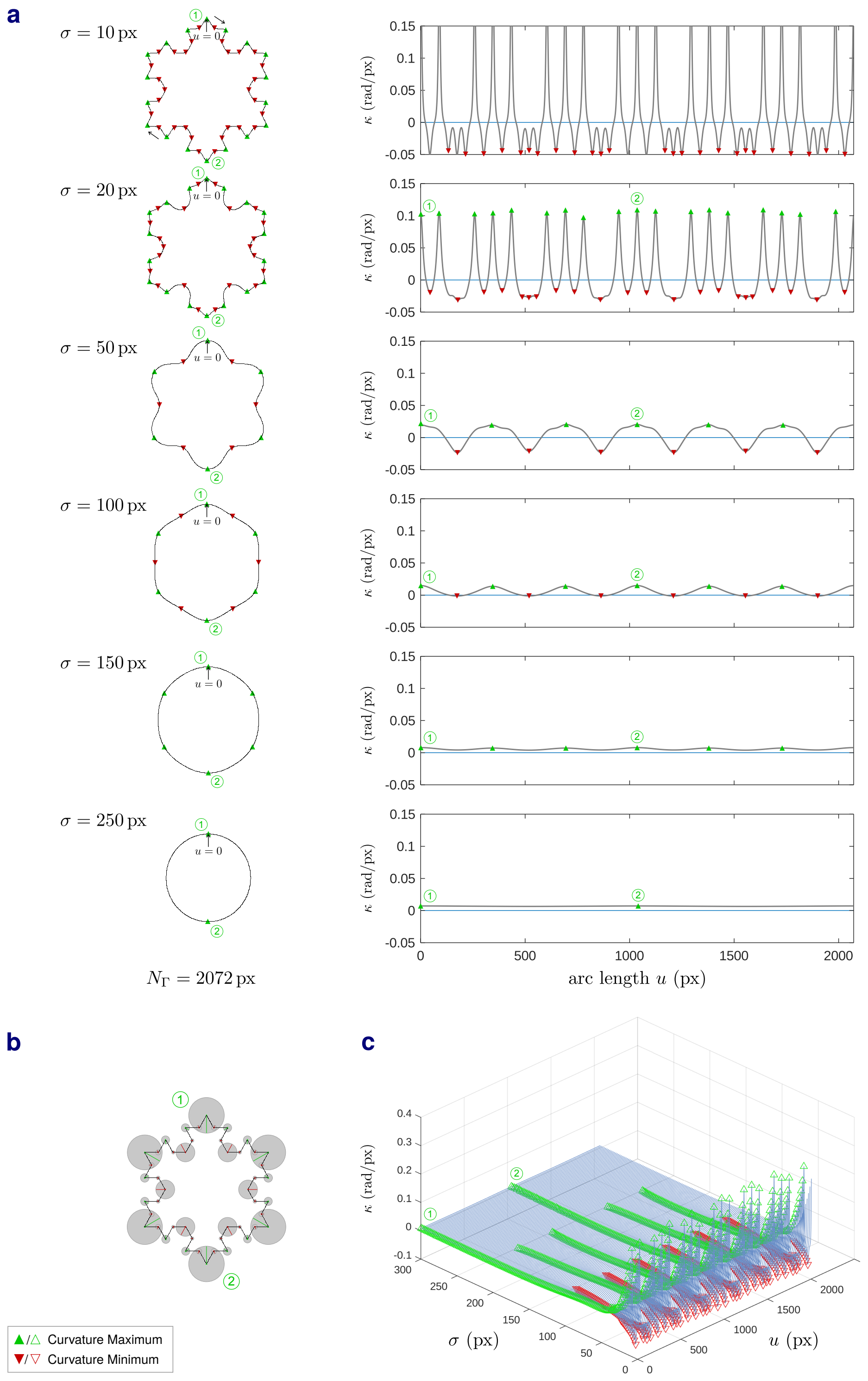


**Fig. 4.4:** CSS construction for an artificial contour (Koch snowflake). The arc length $u$ is counted in the clockwise direction. Selected extrema are labeled with numbers. See text for details. **a** Evolving contour, its curvature, and detected curvature extrema. **b** Original contour with detected curvature extrema and characteristic scales (based on further analysis steps). **c** CSS representation with detected extrema.

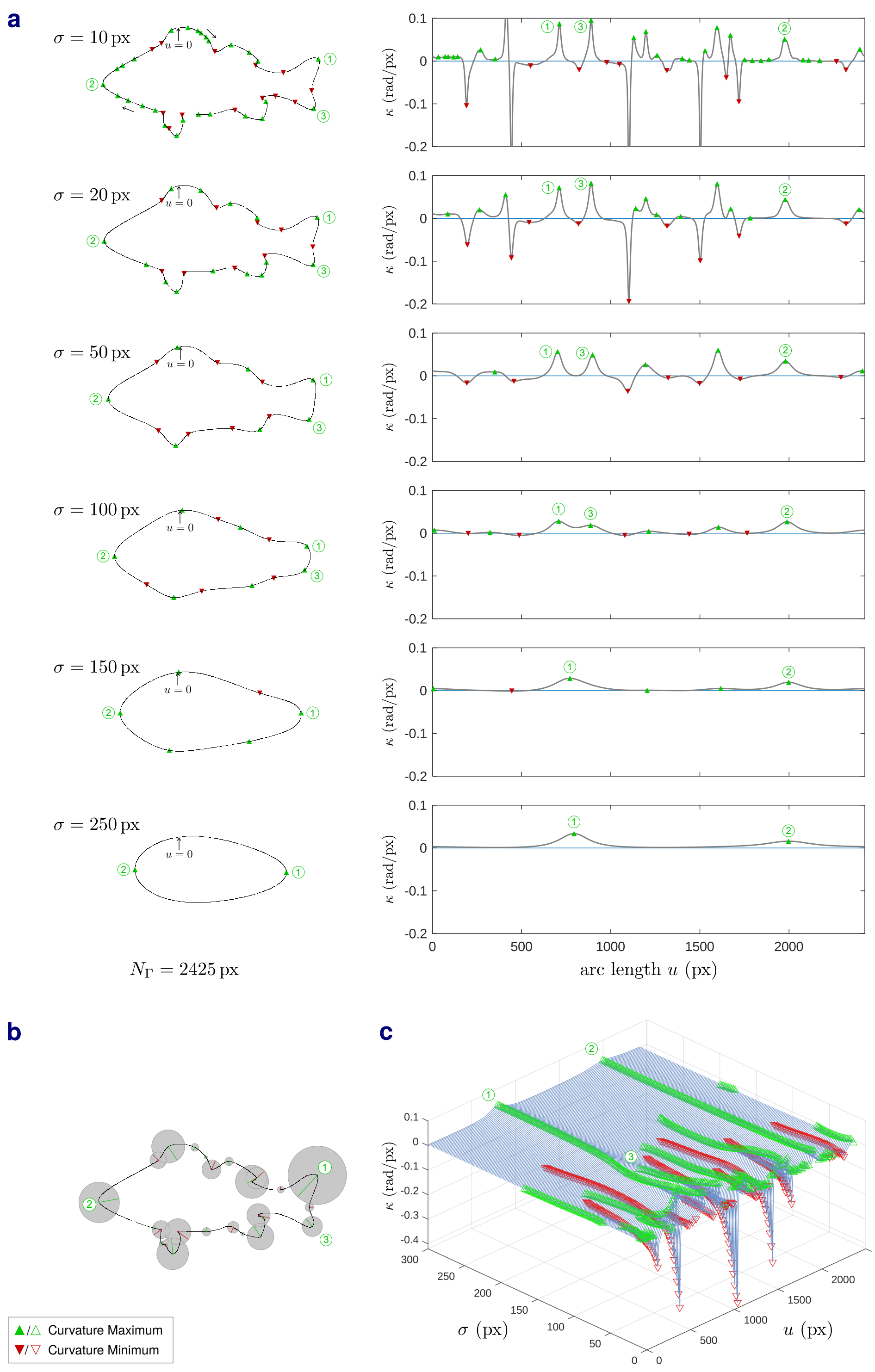


**Fig. 4.5:** The same process as shown in Figure 4.4 for a natural contour (a fish). Selected extrema are again labeled with numbers. See text for details. **a** Evolving contour, its curvature, and detected curvature extrema. **b** Original contour with detected curvature extrema and characteristic scales (based on further analysis steps). **c** CSS representation with detected extrema.

### 4.3.4 Tracing of Curvature Extrema

To analyze the signature functions of the curvature extrema, these functions have to be extracted from (traced within) the CSS representation. As can be observed in Figures 4.4c and 4.5c, the signature functions form connected curves in the $\sigma$-$u$-plane, so that distance measures in this plane can be used for the tracing. The following methodology has been developed in this work for this purpose.

The tracing algorithm is executed sequentially for each $\sigma$, starting with an initialization step at the lowest scale $\sigma_{\text{start}}$. For each detected extremum $u^{\pm}_{\sigma_{\text{start}},i}$, a corresponding signature function $S^{\pm}_i$ is created. This function records the evolution of the extremum over increasing scales $\sigma$ by storing a set of triplets $(\sigma_i, u_{\sigma,i}, \kappa_{\sigma,i})^{\pm}$, where $u_{\sigma,i}$ denotes the spatial position of the extremum (cf. Section 4.3.3), $\kappa_{\sigma,i}$ its curvature at scale $\sigma_i$, and the superscript its sign (curvature minimum or maximum). In summary, each signature function has the following form:

$$S^{\pm}_i = \{(\sigma_i, u_{\sigma,i}, \kappa_{\sigma,i})^{\pm}\} . \tag{4.30}$$

It is only necessary to create signature functions for the extrema detected at $\sigma_{\text{start}}$, as no new features can appear at higher scales due to the causality principle (cf. Section 4.1.3.5). For all subsequent scales after the initialization $\sigma > \sigma_{\text{start}}$, each detected curvature extremum $u^{\pm}_{\sigma,i}$ is assigned to the signature function of the closest extremum in terms of the $u$-distance at the previous scale $\sigma - \sigma_{\text{step}}$, and provided that the extremum has the same sign (i.e., the extremum is a curvature minimum or maximum). For a closed contour, the $u$-distance must be checked across both the first and last values of the smoothed contour $\Gamma_\sigma$.

After the tracing, the curvature component of each signature function $S^{\pm}_i$ is analyzed as a function of scale. Hence, the function employed for further analysis is $\kappa_{\sigma,i}(\sigma)$.

### 4.3.5 Assigning Characteristic Scales

The key idea in this work for assigning characteristic scales to curvature extrema is to identify and distinguish local and global characteristics in the function profiles of $\kappa_{\sigma,i}(\sigma)$. This idea has not been exploited in the literature. Distinguishing these characteristics is practical because the objective of this work is to extract local features. As shown below by computational experiments, the signature functions automatically adapt to scale changes, similar to the signature functions in Figure 2.5, so that the characteristic scales can be determined in a bottom-up manner.

Consider Figures 4.4 and 4.5: Most curvature extrema are gradually smoothed out until they disappear at specific scales, while only a few extrema remain at higher scales. If the $\sigma$ value at which an extremum disappears in the CSS representation were used as the characteristic scale, the values obtained for the remaining extrema would not be plausible and clearly too high. However, as can be seen for the dominant curvature maxima in Figure 4.4b—which should have the same characteristic scales due to the symmetric shape of the contour—this work has led to a methodology that identifies the same characteristic scales despite the ongoing signature functions. The corresponding characteristics of the signature functions identified and employed in this work are described below.

#### 4.3.5.1 The CSS Representations Under Scale Changes

Before going deeper into the analysis of the signature functions, the general behavior of the CSS representations under scale changes is examined. For this purpose, the shapes from Figures 4.4 and 4.5 have been resized, and the Canny edge detector (Canny, 1986) has then been applied to obtain the corresponding contours. After that, the scales at which specific extrema disappear are analyzed in the $u$-$\sigma$-plane. This plane corresponds to the top views in Figures 4.4c and 4.5c.

Consider Figures 4.6 and 4.7: Here, the scales at which specific extrema disappear are indicated at the end of the corresponding signature functions in the $u$-$\sigma$-planes. As can be observed, the scales adapt to the size of the shapes in an approximately proportional manner. This was one of the first observations when developing the method in this work and an early indication that local scale information can be extracted from the CSS representations. Slight variations from the ideal values result from the discretization of the contours and the discrete step width $\sigma_{\text{step}}$. However, as already noted in Section 4.3.3, some extrema do not disappear at all (marked with ① and ② in Figures 4.4–4.7, respectively). Furthermore, additional analyses show that using the scales at which curvature extrema disappear does not yield meaningful characteristic scales in all cases. This issue is particularly apparent for elongated structures and is discussed in further detail in Section 4.3.5.2.

#### 4.3.5.2 Analysis of the Signature Functions

Figure 4.8 shows the signature functions of selected curvature extrema under scaling transformations of the input shapes. As can be observed, the signature functions automatically adapt to the scale changes. In particular, it can be observed that, except

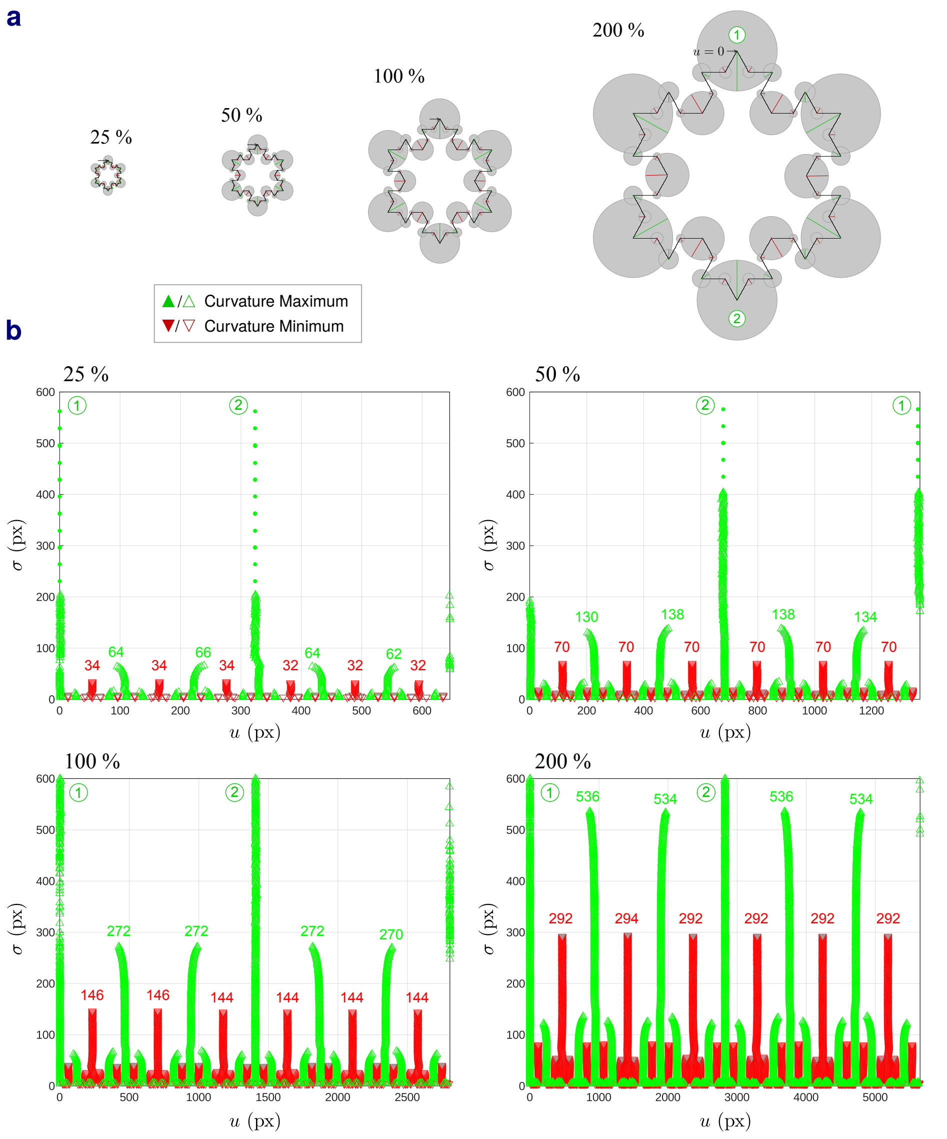


**Fig. 4.6:** CSS representations of the snowflake contour at different scales. **a** Contours with detected curvature extrema and characteristic scales (based on further analysis steps). **b** CSS representations, where the numerical values indicate the highest scale of the respective signature functions. Dots indicate continuing signature function.

for the extrema with ongoing signature functions, all other extrema disappear at scales that are approximately proportional to the scaling factor.

A reasonable starting point for further analyses are the signature functions of the dominant curvature maxima of the symmetric test contour (Koch snowflake) in Figures 4.4 and 4.6, where, except for the two remaining extrema, all maxima are gradually smoothed out until they disappear at specific scales. Consider the signature functions of the extrema in Figure 4.4a, marked with ① and ②: The signature functions of maximum

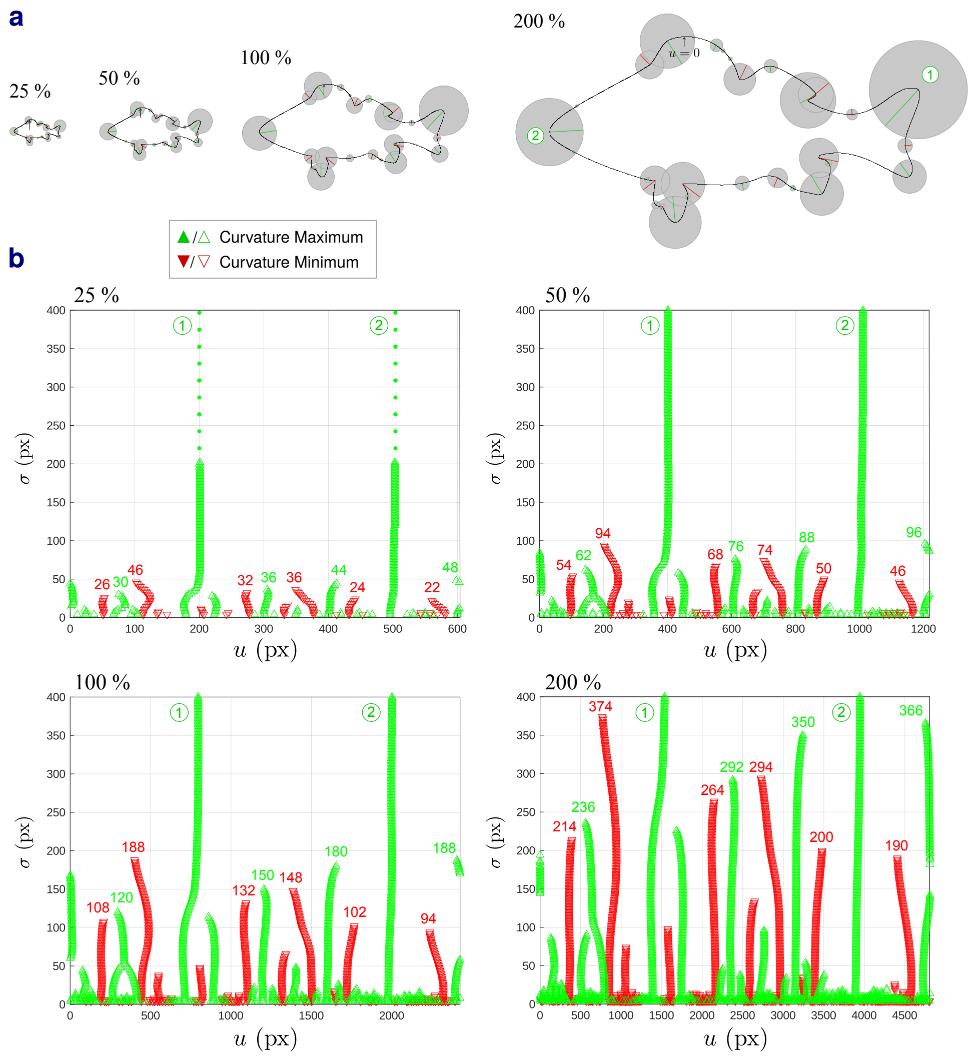


**Fig. 4.7:** The same process as shown in Figure 4.6 for the fish contour. **a** Contours with detected curvature extrema and characteristic scales. **b** CSS representations, where the numerical values indicate the highest scale of the respective signature function.

① exhibit a distinct minimum before the curvature starts increasing again. Now consider the signature functions of maximum ②: Interestingly, the scales at which these signature functions disappear are identical to the distinct minimum of the signature functions of maximum ①. This fundamental behavior has been consistently observed for arbitrary input contours within the analyses of this work.

Further experimental analyses reveal the reasons for this behavior: at a certain scale, all extrema are smoothed out, the contour adopts an elliptical shape, and its geometric length decreases with increasing scales (cf. Section 4.3.3). The increasing behavior of the signature functions is mathematically justified below. The characteristic scale at which an extremum evolves from a local to a global extremum can therefore be identified in the signature functions. This finding is employed in this work and has not been described in the literature.

However, it can be observed that, before an extremum disappears or evolves from a local to a global extremum, the signature functions do not necessarily decrease or increase in a uniform manner. Instead, they can exhibit complex patterns. For example, consider the signature functions of maximum ① in Figure 4.8b. The underlying reasons for this behavior are interferences with neighboring extrema (cf. Section 4.3.3) and, in the case of elongated structures, the fact that they remain distinct up to a specific degree of smoothing. For example, consider maximum ③ in Figure 4.9a: up to a specific scale, the signature function increases, then decreases, and increases again.

The increasing behavior (or decreasing in the case of curvature minima) of the signature functions of elongated structures and global extrema can be explained by considering that curvature can also be expressed as the angular change $\mathrm{d}\alpha$ per distance $\mathrm{d}u$ along the contour:

$$\kappa = \frac{\mathrm{d}\alpha}{\mathrm{d}u} \tag{4.31}$$

Note that $\kappa$ increases with decreasing $\mathrm{d}u$. During the smoothing, the geometric lengths of elongated structures become gradually shorter, while the structures remain distinctive up to a specific scale. In other words, $\mathrm{d}u$ becomes gradually smaller, while $\mathrm{d}\alpha$ remains relatively constant at the extremum. The same behavior can be observed for global extrema: the geometric length of the contour becomes gradually shorter, while $\mathrm{d}\alpha$ remains relatively constant due to the evolving elliptical shape (cf. Figures 4.4a and 4.5a). As a result, the curvature $\kappa$ increases.

### 4.3.5.3 Conclusions and Proposed Methodology

Taking all observations into account, it appears that after a convergence phase, if present, there is a segment in the signature functions where an extremum is systematically smoothed out until it either disappears or evolves from a local to a global extremum. Based on this, the signature functions can be divided into three segments, labeled I–III in Figure 4.9. This behavior can be observed for both curvature minima and maxima, with the effect being more prominent for maxima of closed contours due to their increasing convexity under smoothing. Based on these observations, the following principle for assigning a characteristic scale to each curvature extremum is derived:

The scale of the last local minimum of the absolute value of the signature function is assigned to the curvature extremum as its characteristic scale:

$$\hat{\sigma}_i := \underset{\sigma}{\arg\,\text{last local min}} \left|\kappa_{\sigma,i}(\sigma)\right| . \tag{4.32}$$

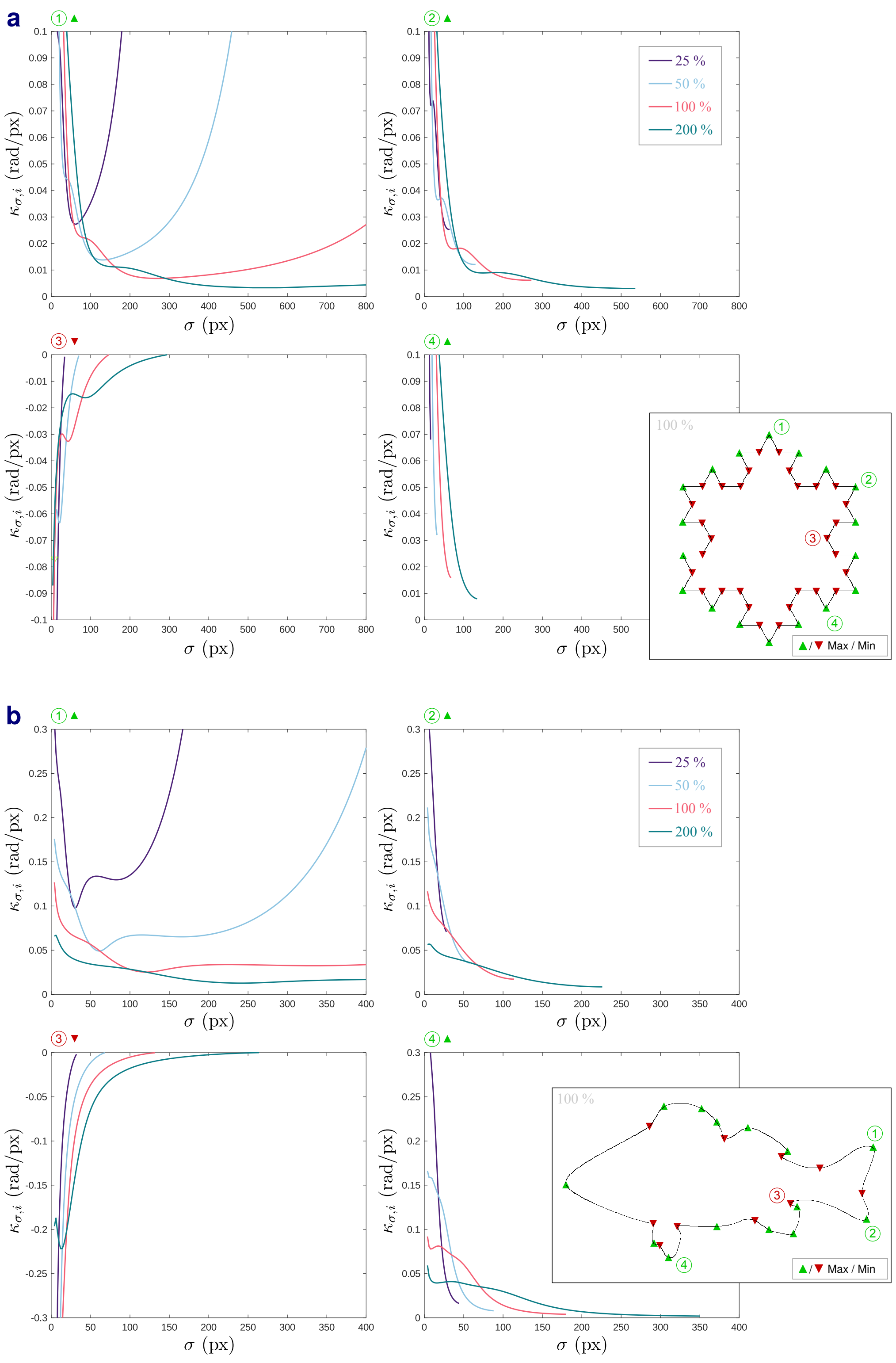


**Fig. 4.8:** Signature functions of selected curvature extrema (labeled with numbers) under scaling transformations. See text for details. **a** The symmetric snowflake contour simplifies the analysis of specific details. **b** The fish contour exhibits similar properties.

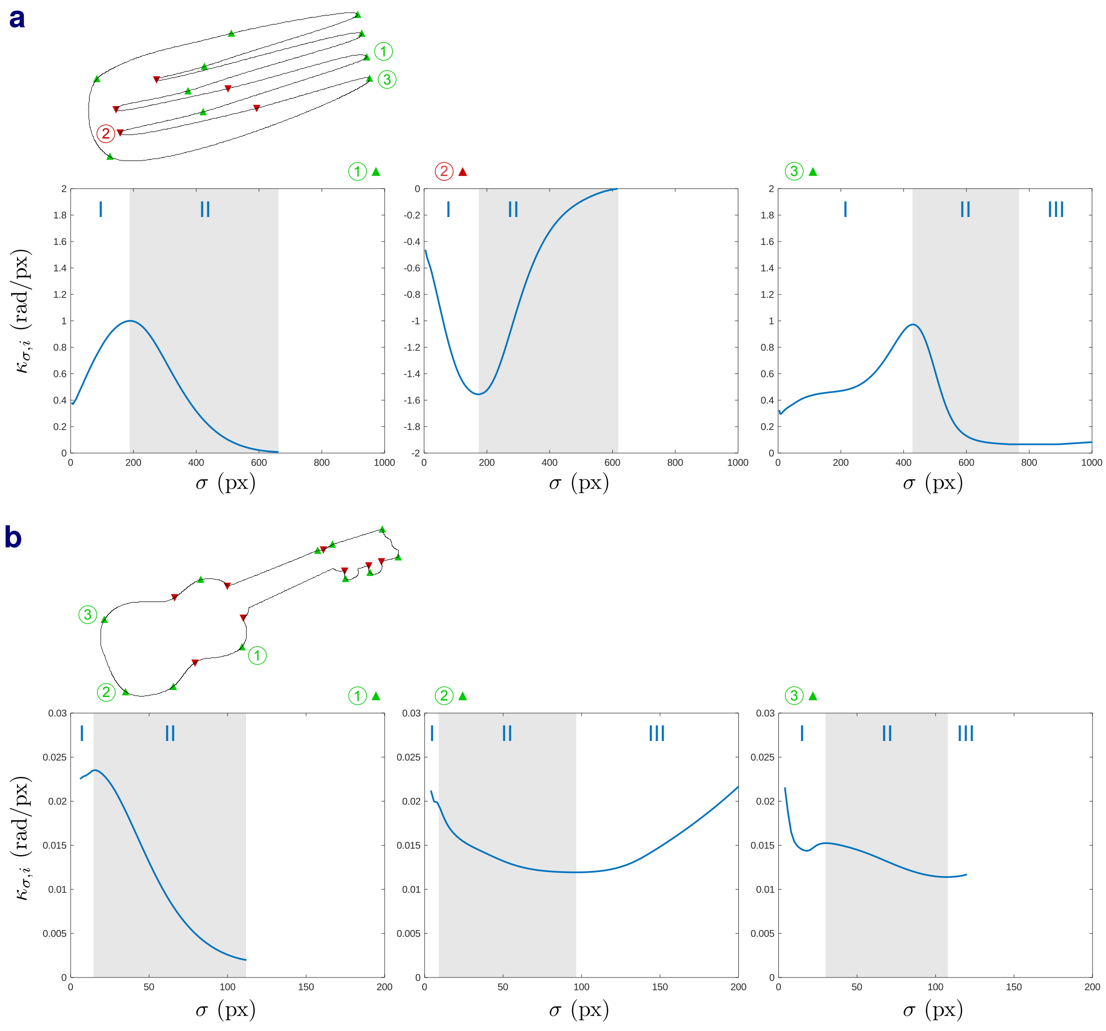


**Fig. 4.9:** Signature functions of selected curvature extrema (labeled with numbers) with divisions into segments, where local characteristics disappear at the end of Segment II. Segment III reflects global characteristics. **a** A shape with elongated structures. **b** Another shape for comparison.

This corresponds to the end of Segment II in the signature functions. Given the arbitrary shape and complexity of input contours, this is a straightforward criterion, effectively avoiding overfitting. The absolute value is used to treat curvature minima and maxima equivalently.

Finding the last local minimum requires that the maximum scale considered is chosen sufficiently large, with $\sigma_{\text{end}} = 0.2\,N_\Gamma$ used for closed contours throughout this work. For open contours, the corresponding value from the closed contour is used. The radius of each region is chosen as $r_i = 0.3\,\hat{\sigma}_i$, where the factor has been determined visually, and the optimal value can be determined experimentally based on the specific task (e.g., by evaluating matching performance).

### 4.3.6 Assigning Characteristic Orientations

In the results of the method developed in this work, orientation lines are used to indicate characteristic orientations, as is standard in the literature. The orientations could be used to compute a rotation-invariant descriptor vector for each contour segment, which corresponds to the normalization step in Figure 2.4. In SIFT, characteristic orientations are determined using histograms of oriented gradients and are thus derived from the gray-value distribution of the original image. However, when working directly with contour segments, orientations can only be computed from the shape of the segments.

In principle, the method developed in this work analyzes specific pixel subsets within the region defined by the characteristic scale of a curvature extremum. A smaller reference region is first selected around the extremum to minimize interference from neighboring extrema. This reference is then compared with the two pixel subsets within the region that are separated by the contour segment. The subset with fewer shared pixels is identified as the inner region, assuming that curvature extrema form an arc. Finally, the orientation is assigned based on the spatial relationship between the extremum and the center of the inner region. However, the results of the method are only reliable for clearly curved segments and not for relatively short, straight segments. Therefore, it is not systematically evaluated here.

If the original image from which the contour was extracted is taken into account, the corresponding gray values could be used to apply the orientation computation method from SIFT directly. However, in this case, changing backgrounds can introduce additional challenges, as the description should capture only the object itself. Another approach could involve using a deep learning-based descriptor that learns rotation-invariant representations (see Georgiou et al., 2020, for a survey on deep descriptors). In this case, the training data could include background variations so that the model learns to distinguish between the object and the background.

### 4.3.7 Extension to Open Contours

Up to this point, only closed contours have been considered. To process open contours as well, i.e., to convolve them with Gaussian functions, they must be padded at their boundaries while still supporting the scale assignment method developed in this work.

To achieve this, a new padding method is presented. Standard methods are insufficient here: with constant padding, the contour would be artificially fixed at its boundaries. As a result, it would not become systematically shorter, which prevents the detection of the scale at which an extremum evolves from a local to a global extremum. With reflection and circular padding, parts of the contour are reflected, which can significantly alter its evolution during smoothing.

Since there is usually no information about the correct extension of an open contour, it is reasonable to preserve its given shape while introducing a tendency towards closing. While one might consider using interpolation methods such as splines here, these require additional parameters, such as the polynomial degree and boundary conditions. The new padding method is a variant of constant padding, where the value from one side is repeatedly added to the opposite side of the contour.

The method works as follows: for a specific fraction of the outer padded samples on one side, the outer value from the opposite side of the contour is used. Formally, let $N_p$ denote the total number of samples to be padded on each side. A fraction $\gamma$ of these samples is then filled with the outer value from the opposite side, while the remaining $(1-\gamma)N_p$ samples are filled with the outer value from the same side. As an example, a padded signal $x$ could have the following form:

$$x = (x_{N_\Gamma-1}, x_0, x_0, x_0 \,|\, x_0, x_1, x_2, \dots, x_{N_\Gamma-1} \,|\, x_{N_\Gamma-1}, x_{N_\Gamma-1}, x_{N_\Gamma-1}, x_0)\ , \tag{4.33}$$

where the unpadded signal is given within the bars. In this example, $\gamma = 0.25$. Throughout this work, a value of $\gamma = 0.1$ (or $10\,\%$) is used, determined experimentally (the exact value is uncritical). As a result, the entire process is determined by a single parameter, effectively avoiding overfitting.

By integrating samples from the opposite side, the sides are gradually drawn towards each other during smoothing, effectively acting as a closing mechanism. Corresponding examples are shown in Figure 4.10. As can be observed, the open contours closely match the original closed contours, so that similar characteristic scales can be assigned to the remaining extrema. The closer an extremum is to the cut, the less accurate the results are. However, this is inevitable, as there is usually no information about the correct extension available. As the contours are extended in a smooth manner, the method presented follows the principle of good continuity (cf. Section 2.1.2).

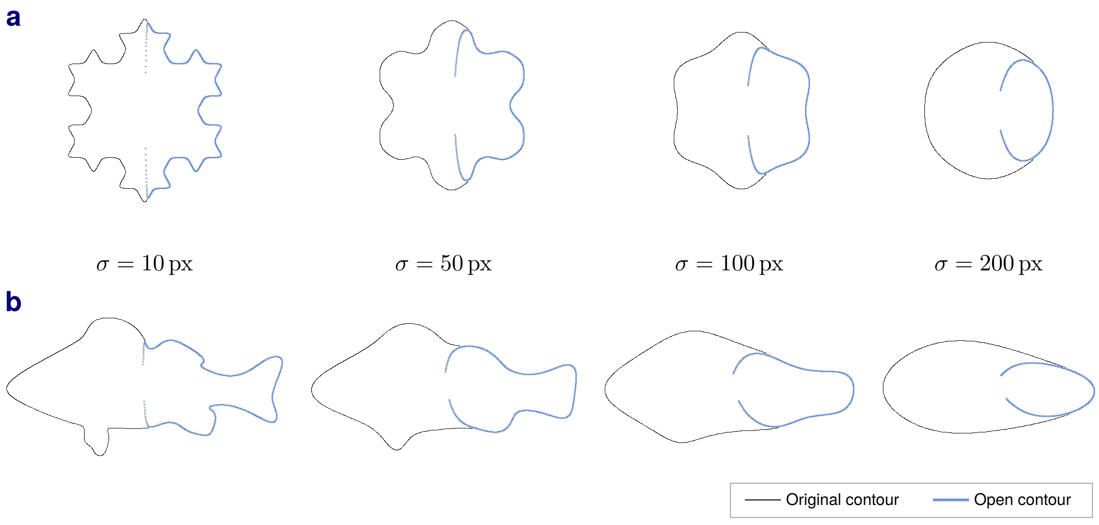

a
$\sigma = 10\,\mathrm{px}$
$\sigma = 50\,\mathrm{px}$
$\sigma = 100\,\mathrm{px}$
$\sigma = 200\,\mathrm{px}$
b
Original contour
Open contour

**Fig. 4.10:** Evolution of open contours under smoothing using the developed padding method. The original closed contours have been cut along their medial axes. The padded contours closely match the originals and their boundaries gradually converge. **a** Results for the snowflake contour. **b** Results for the fish contour.

## 4.3.8 Box Filter Approximation

Major parts of this section, including the two figures, are adapted from (Hennig and Mertsching, 2021), with all writing attributed to the author of this thesis.

Computing the CSS representation of a contour is computationally expensive due to the number of multiplications and additions required. For a contour with $N_\Gamma$ samples and a Gaussian with $N_g$ samples, the complexity at a single scale is $\mathcal{O}(2N_\Gamma N_g)$ (the factor 2 accounts for the $x$- and $y$-components of the contour), and this process is repeated across multiple scales. Note that the filter size increases with higher scales (cf. Section 4.1.4). Therefore, it is reasonable to ask how this process can be optimized. Two potential methods have already been discussed in Section 2.2.4: the pyramid approach in SIFT and the box filter approach in SURF. Since a pyramid approach makes it more difficult to trace the extrema across scales to obtain the signature functions, this work focuses on a box filter approach.

### 4.3.8.1 Variants for Computing the Curvature

Consider the curvature computation in Equation (4.29): This equation requires the first and second partial derivatives of $x_\sigma(u;\sigma)$ and $y_\sigma(u;\sigma)$ with respect to $u$, respectively. Since convolution commutes with differentiation, there are three principal variants to compute $\kappa(u;\sigma)$, as shown in Figure 4.11: (a) the signals are convolved with Gaussians

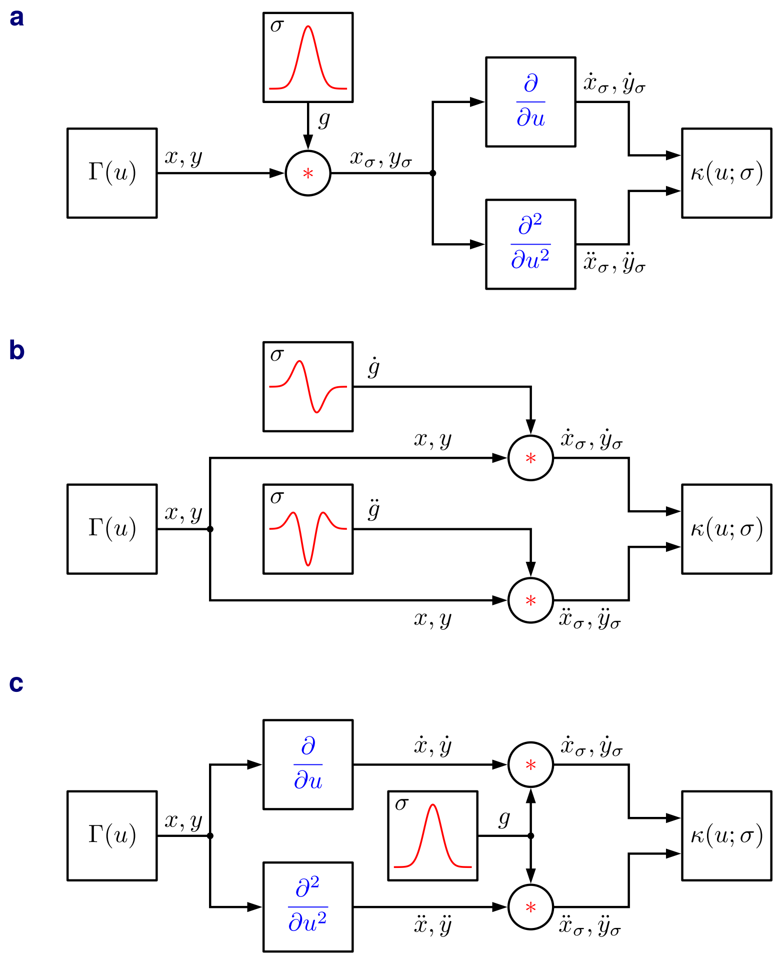


**Fig. 4.11:** Comparison of different variants to compute the curvature $\kappa$. **a** The signals are convolved with Gaussians and then differentiated. **b** The signals are convolved with Gaussian derivatives. **c** The signals are differentiated and then convolved with Gaussians.

and then differentiated, (b) the signals are convolved with Gaussian derivatives, or (c) the signals are first differentiated and then convolved with Gaussians. Note that the second derivatives can also be computed by differentiating the first derivatives, which is not explicitly considered here.

Which of the three variants should be used to compute $\kappa$ based on box filtering? This work argues for variant (a). While all variants should theoretically yield the same results, there are characteristic differences in the computation process. In particular, the number of convolutions is lowest in variant (a). Furthermore, it is easier to approximate Gaussians directly than to approximate their derivatives, as required in variant (b). In variant (c), the signals are first differentiated, which can lead to numerical inaccuracies for discrete input signals in Equation (4.29). In summary, variant (a) is the most efficient and robust approach and is therefore used in the remainder of this work.

#### 4.3.8.2 Box Filter Methods

Gaussian convolution can be accelerated and approximated in various ways, particularly using recursive infinite impulse response (IIR) filters, binomial filters, the discrete Fourier transform, and pyramid-based methods (Kovesi, 2010; Getreuer, 2013). Box filters are of particular interest here because they operate at a fixed cost per pixel (sample) when combined with running sums (cf. Section 2.2.4.2) and are straightforward to implement. Similar to Equation (2.9), the first step is to compute the running sum of the contour as follows:

$$x_\Sigma(\xi) = \sum_{u=0}^{u \le \xi} x(u)\,. \tag{4.34}$$

In the same manner, the running sum $y_\Sigma(\xi)$ has to be computed for $y(u)$. The sum of an arbitrary segment of the original signal—required for box filtering—can then be computed with a single subtraction, as shown in Figure 4.12a. By this means, for $k$ box filtering steps, the computational complexity at a single scale is reduced to $\mathcal{O}(2kN_\Gamma)$. Note that $k$ is typically a small integer between 3 and 5.

In (Hennig and Mertsching, 2021), the four common box filter methods shown in Figure 4.12 are considered. The simplest method is SBF, where the signal is filtered $k$ times with the same filter in an iterative manner. The filter size required to achieve the equivalent $\sigma$ of the Gaussian can be determined analytically (Kovesi, 2010). One issue with this method is that the ideal filter size, typically a real-valued number, must be rounded to an odd integer to center the filter on each contour pixel. As this rounding leads to inaccurate filter results, the method can be improved by using two filters of different sizes, which are also applied in an iterative manner ($k$ times in total). This method is known as FAG. The SB method uses $k$ box filters in a stacked manner to resemble the shape of the Gaussian. The box filters are constructed by minimizing a cost function that measures the deviation of the approximated version from the original Gaussian (Bhatia et al., 2010). Finally, EBF can be seen as a mixed method that integrates iterated filtering and stacked boxes. In this case, two boxes are stacked and applied iteratively $k$ times. Similar to SBF and FAG, the filter coefficients are determined analytically.

The results in (Hennig and Mertsching, 2021) show that the computation times of the four methods are roughly similar. The accuracies are also roughly similar, except for the SB method, which shows larger differences between the approximated and ideal filter

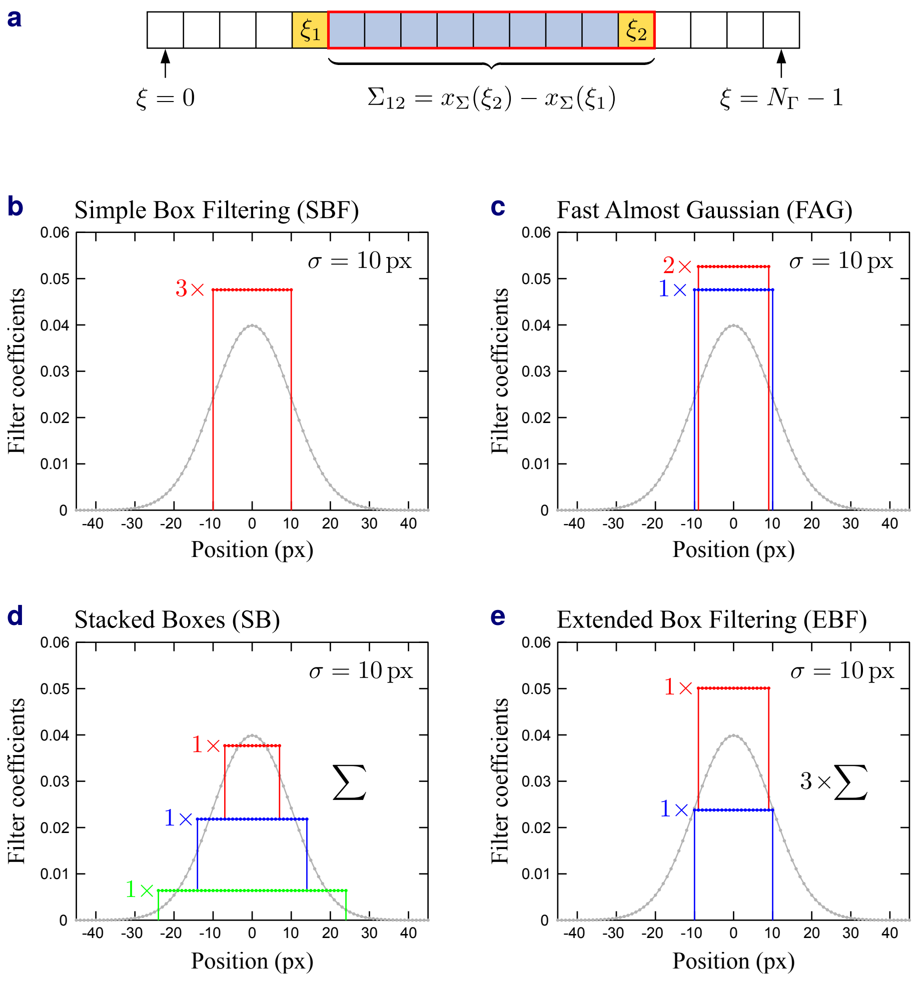


**Fig. 4.12:** Principle of running sums and four common box filter methods for $k = 3$ main filtering steps and $\sigma = 10\,\mathrm{px}$. The original Gaussian is shown in the background. All filter coefficients are up to scale. **a** The sum of an arbitrary segment of the signal can be computed with a single subtraction. **b** Simple Box Filtering (SBF; Wells, 1986; Kovesi, 2010). **c** Fast Almost Gaussian Smoothing (FAG; Kovesi, 2010). **d** Stacked Boxes (SB; Bhatia et al., 2010; Elboher and Werman, 2012). **e** Extended Box Filtering (EBF; Gwosdek et al., 2011).

results. In summary, FAG has been found to be the most efficient method and it should be used with $k = 5$ filtering steps in combination with variant (a) from Figure 4.11. Therefore, it is used in the remainder of this work.

#### 4.3.8.3 Runtime Analysis

The box filter approximation has a runtime of approximately $\mathcal{O}(N_\Gamma)$ with respect to the contour length $N_\Gamma$, compared to $\mathcal{O}(N_\Gamma^2)$ for standard convolution. This difference is both theoretically justified and confirmed by empirical measurements. The box filter variant is therefore real-time capable. For a contour of 10,000 pixels, it achieves an acceleration factor of over 600× compared to the standard convolution (0.005 s vs. 3.1 s, measured on a consumer PC).

## 4.4 Evaluation

The following evaluation is based on the MPEG-7 dataset (MPEG-7, 1999). The dataset contains 1,400 binary images from 70 different object classes, with 20 examples per class. The classes are visually distinctive but include significant intra-class variation, as well as differences in rotation, size, and image resolution.

### 4.4.1 Keypoint Detection Under Transformations

To evaluate the keypoint detection and the robustness of the characteristic scales, 10 shapes from the MPEG-7 dataset have been randomly selected and transformed using geometric (scaling, rotation) and non-geometric (noise, cutting) transformations. The selected shapes are shown in Figure 4.14e. The following transformations have been applied to each shape (contour):

- Scaling (of the $100\,\%$ version): $25\,\%$, $50\,\%$, $200\,\%$, $400\,\%$.
- Rotation (of the $100\,\%$ version): $15°$, $30°$, $45°$, $60°$, $75°$, $90°$.
- Gaussian noise ($\sigma_d$, applied to the $100\,\%$ version): $1\,\mathrm{px}$, $3\,\mathrm{px}$, $5\,\mathrm{px}$.
- Cutting (remaining length from the $100\,\%$ version): $75\,\%$, $50\,\%$, $25\,\%$.

The scaled and rotated versions have been created using vector representations generated from the $100\,\%$ versions of the shapes to reduce discretization artifacts. Next, the Canny edge detector (Canny, 1986) has been applied to obtain the outer closed contours. An exception is the set of $90°$ versions, which have been obtained by rotating the $100\,\%$ versions of the contours directly.

The versions with Gaussian noise have been created by positioning a point orthogonal to every 5th contour pixel of the $100\,\%$ versions. The distance of the point to the contour has been randomly drawn from a Gaussian distribution with the given standard deviation $\sigma_d$, and the points have then been connected. The cut versions have been created by cutting after the given percentage of the $N_\Gamma$ pixels of the $100\,\%$ version.

The method from this work uses generic parameters that are not tailored to a specific shape or transformation. From this perspective, the method can be considered parameter-free. The main parameters used for the evaluation are specified in Table 4.2.

| Parameter | Value | Description |
|---|---|---|
| $\sigma_{\text{start}}$ | $8.0$ | Minimum scale parameter (in $\mathrm{px}$) |
| $\sigma_{\text{step}}$ | $2.0$ | Step width of the scale parameter (in $\mathrm{px}$) |
| $\sigma_{\text{end}}$ | $0.2\,N_\Gamma$ | Maximum scale parameter (in $\mathrm{px}$) |
| $N_g$ | $2\lceil 4.5\sigma \rceil + 1$ | Number of samples to represent discrete Gaussians |
| $k$ | $5$ | Number of box filter steps (FAG method) |
| $r_i$ | $0.3\,\hat{\sigma}_i$ | Radius of the region around each keypoint (in $\mathrm{px}$) |

**Tab. 4.2:** Parameters used for the evaluation.

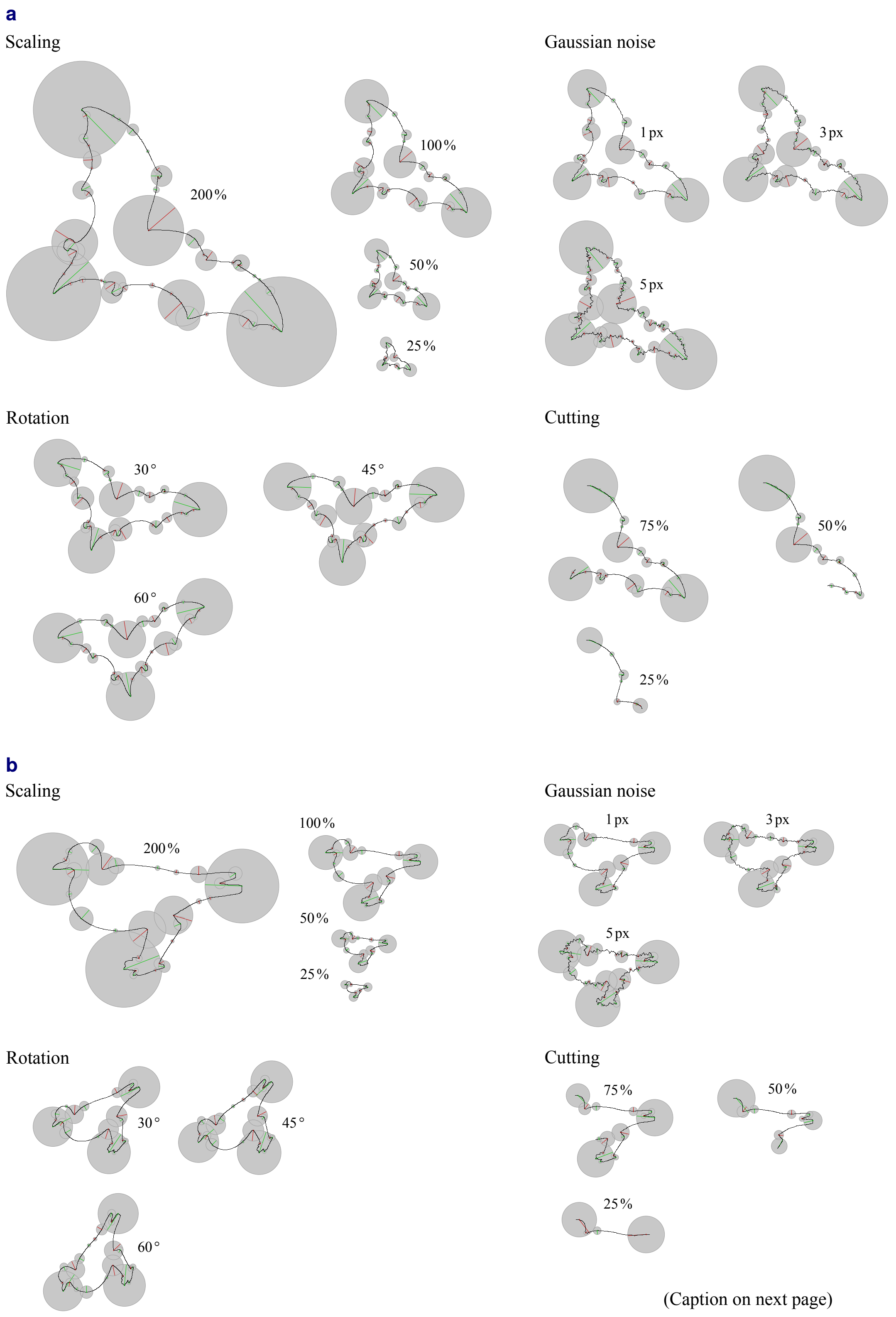
#### 4.4.1.1 Qualitative Analysis of Detected Keypoints

Figure 4.13 shows the detected keypoints and their characteristic scales under transformations for selected natural and artificial shapes of varying complexity from the MPEG-7 dataset (see figure caption for details). The results show that the dominant keypoints and their characteristic scales can be reliably detected even under extreme scale changes or strong noise. High scales tend to be more stable than low scales due to the limited resolution of fine-scale structures as a result of the discretization.

The scales adapt to the size (scaling) of the shapes in an approximately proportional manner and remain stable under rotations, which is one of the main objectives of this work. As the amount of noise increases, the scales become more unstable. This is expected, as the local contour structure is significantly altered, and a local feature should also adapt to such changes to a certain extent. Nevertheless, as expected, it is primarily the overall shape that determines the characteristic scales. For open contours, larger scale deviations can occur, depending on the remaining length. This is also plausible, as a significant loss of information about the overall structure can strongly affect the curve evolution. However, for many keypoints, the scales remain stable.

#### 4.4.1.2 Number of Detected Keypoints

Figure 4.14 shows the number of detected keypoints under the transformations for normal and box filtering. Both variants show similar trends and are consistent across all shapes, confirming that the method does not overfit to specific shapes. Small deviations of the box filtering from the normal results can be attributed to keypoints with small



(Caption on next page)

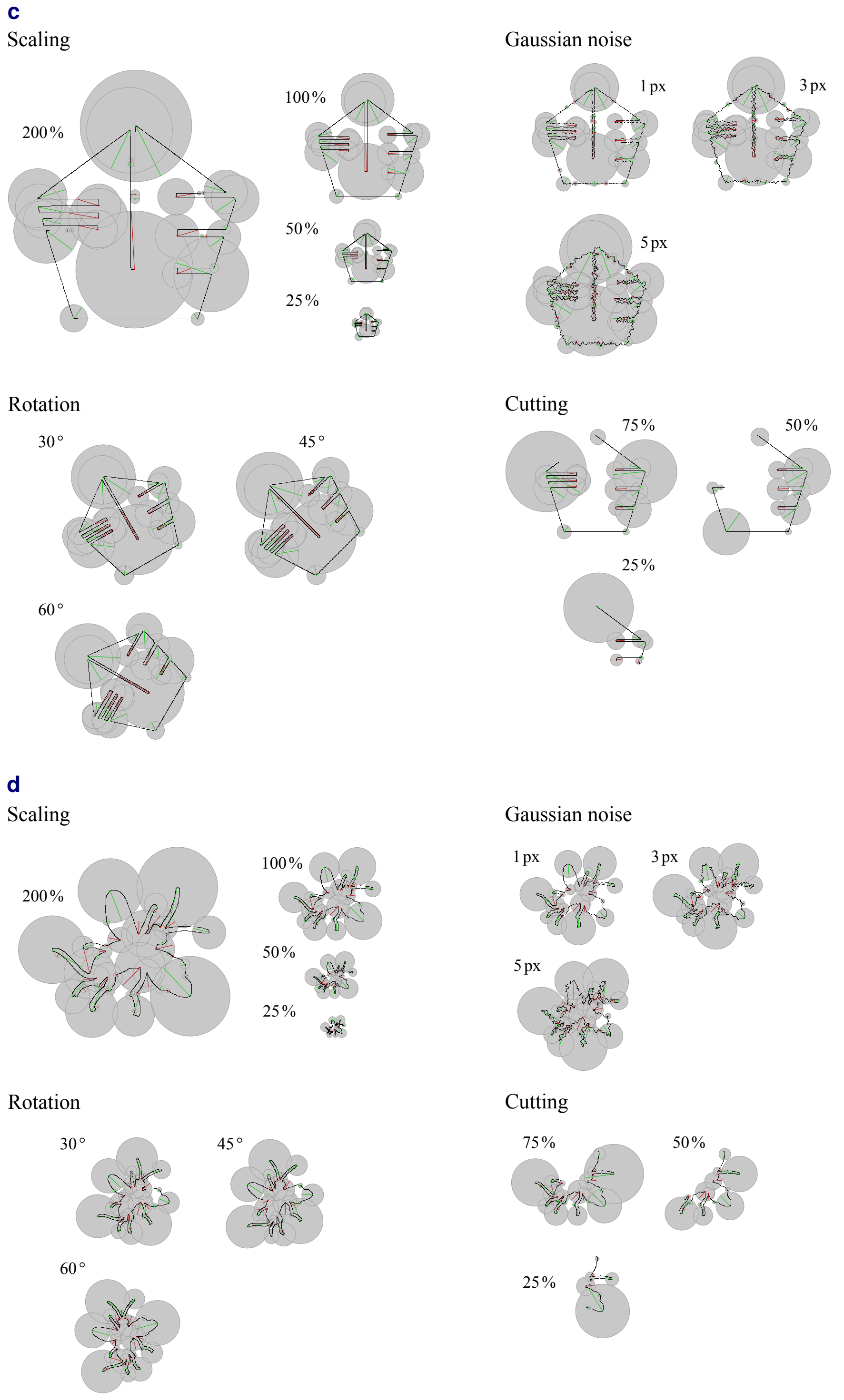


**Fig. 4.13:** Detected keypoints and their characteristic scales for four representative shapes from the MPEG-7 dataset under transformations. **a** *bat-1* (natural, low complexity). **b** *bird-1* (natural, medium complexity), **c** *device6-1* (artificial, geometric, medium complexity with elongated structures). **d** *fly-1* (natural, high complexity with thin structures).

characteristic scales. This is likely because box-filter approximations are represented by relatively few samples at low scales, leading to reduced approximation accuracy.

For the scaling transformations in Figure 4.14a, the number of keypoints decreases for smaller scaling factors and increases for larger scaling factors. This is the expected result, as downscaling removes keypoints at low scales, while upscaling introduces additional keypoints at low scales (cf. the scaling transformations in Figure 4.13). However, the dominant keypoints are reliably detected across the different scaling factors.

For the rotation transformations in Figure 4.14b, the number of keypoints remains approximately constant, indicating a high level of stability (rotation invariance). A minor deviation is observed for the test shape *bone-1* at a rotation of $60°$, with more keypoints detected than expected. This can be attributed to the extrema detection process (cf. Section 4.3.3), which is too sensitive along the longitudinal structure of the bone, leading to additional small-scale keypoints.

For the transformations with Gaussian noise in Figure 4.14c, the number of keypoints remains relatively stable up to $\sigma_d = 3\,\mathrm{px}$, underlining the robustness of the scale-space approach. The number of keypoints significantly increases at $\sigma_d = 5\,\mathrm{px}$, as the noise introduces many additional small-scale keypoints. Nonetheless, the dominant keypoints are reliably detected.

For the cutting transformations in Figure 4.14d, the number of keypoints decreases approximately in proportion to the remaining contour length. This implies that the detection process is robust to missing contour parts (since the remaining contour length corresponds to the number of remaining keypoints), which is particularly relevant for processing incomplete or occluded contours.

#### 4.4.1.3 Keypoint Matching Rates

Figure 4.15 shows the keypoint matching rates under the transformations for normal and box filtering, where both variants show similar trends. Matches have been determined based on the keypoint positions and their characteristic scales, providing insight into the stability of these attributes.

For each keypoint from the $100\,\%$ reference version of the respective test contour, the expected position and scale in the transformed contour are computed based on the known transformation. A match is only accepted if the keypoint types are the same (curvature minimum or maximum) and if the following conditions are met:

$$D < \left(\frac{\text{contour scale}}{100\,\%}\right) \cdot 20\,\mathrm{px} \quad \text{and} \quad \left|\frac{\hat{\sigma}_{\mathrm{src}} - \hat{\sigma}_{\mathrm{match}}}{\hat{\sigma}_{\mathrm{src}}}\right| \cdot 100\,\% < 20\,\%\,, \tag{4.35}$$

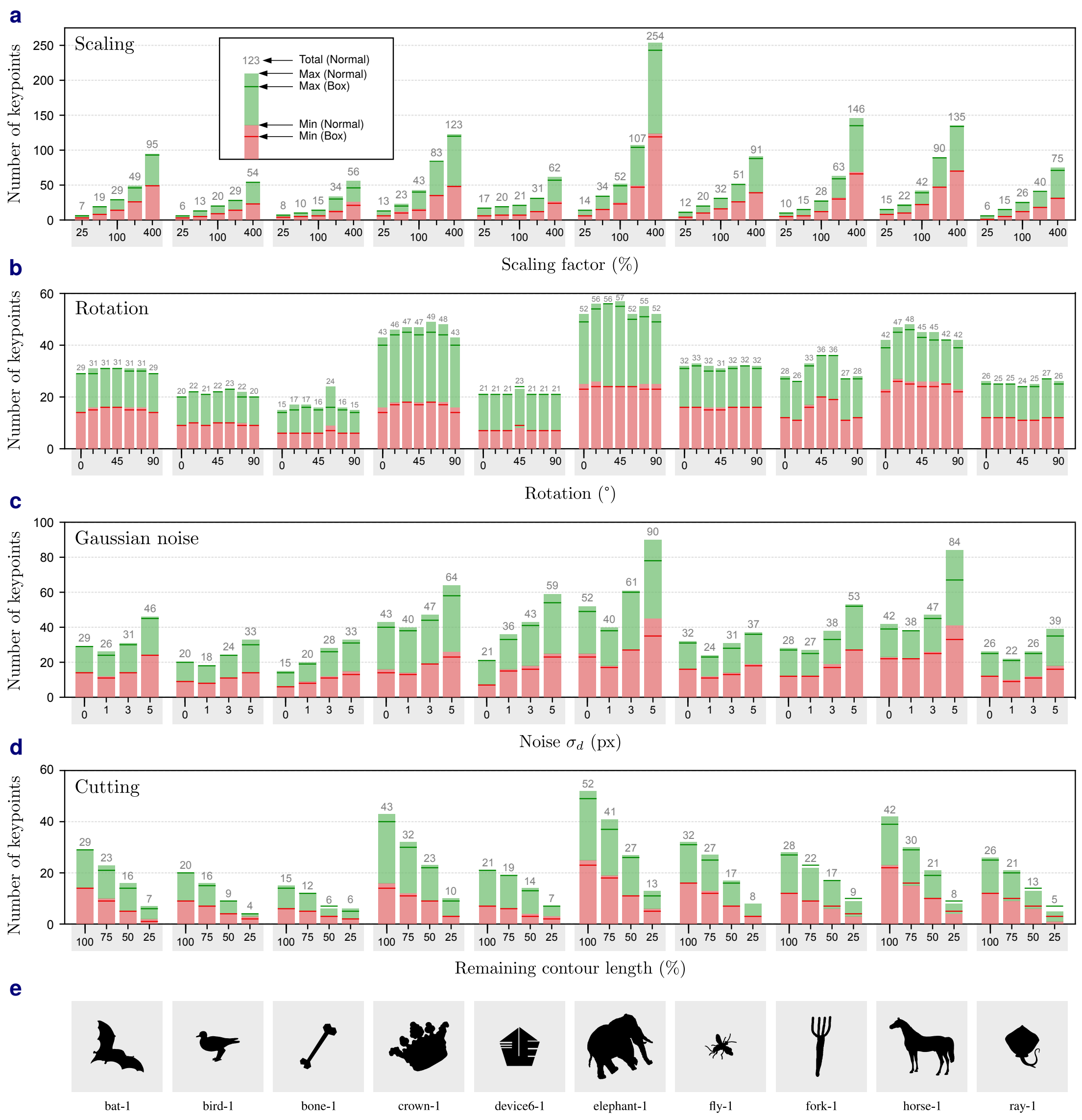


**Fig. 4.14:** Number of detected keypoints under different transformations for normal and box filtering. The legend is shown in subfigure a. Each column shows the results for one input shape. **a** Scaling. **b** Rotation. **c** Gaussian noise. **d** Cutting. **e** Input shapes.

where $D$ is the Euclidean distance between the expected and candidate keypoint positions in the transformed contour, and $\hat{\sigma}_{\text{src}}$ and $\hat{\sigma}_{\text{match}}$ denote their characteristic scales, respectively. This means that the matched keypoint must be sufficiently close to the expected position and scale, with the acceptable distance $D$ increasing for larger contour scaling factors. These conditions filter out obvious mismatches while preserving high recall. Examples of matched keypoints are shown in Figure 4.16.

For the scaling transformations in Figure 4.15a, the matching rate decreases for smaller scaling factors and only slightly decreases for larger factors. The decrease is the expected

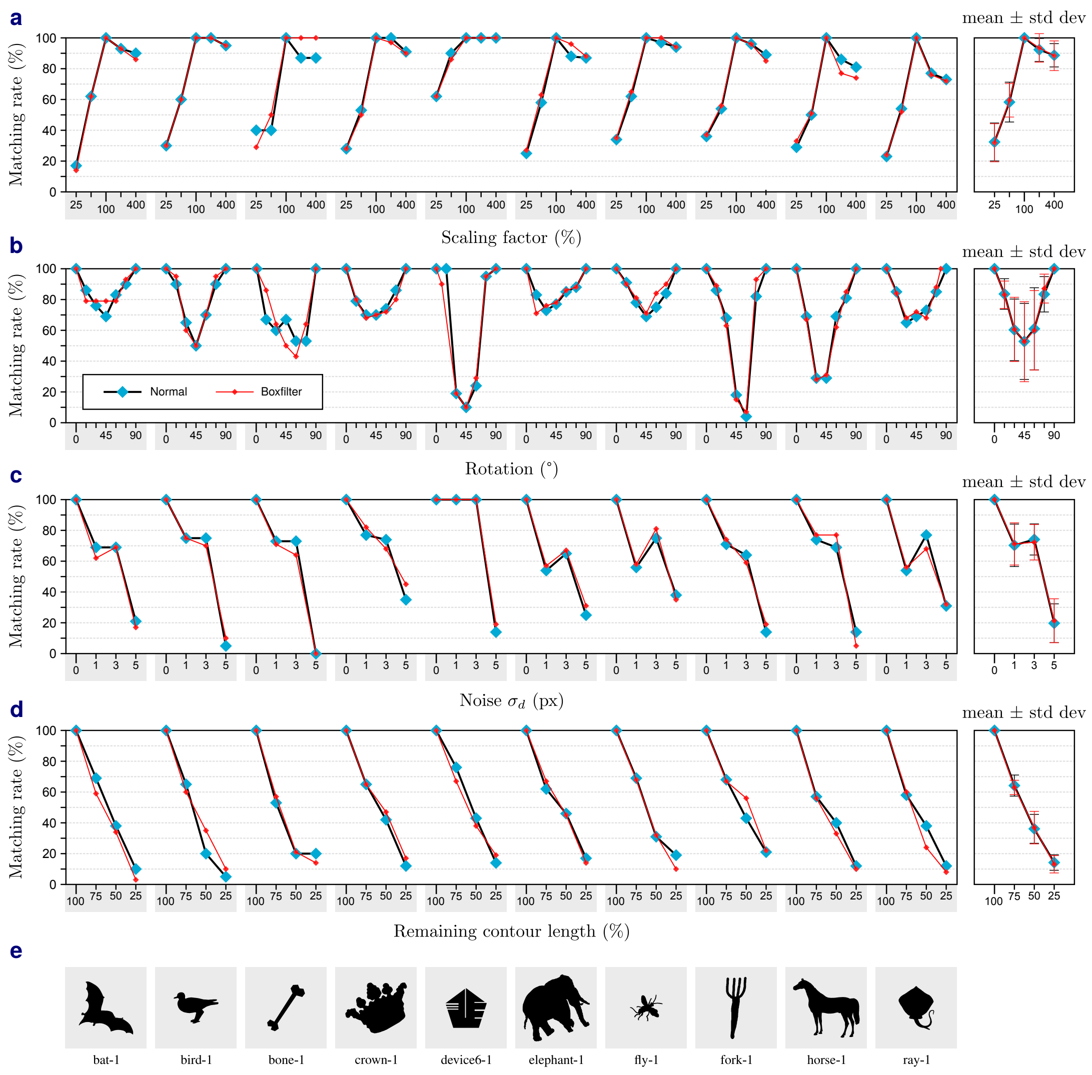


**Fig. 4.15:** Keypoint matching rates under different transformations for normal and box filtering. The legend is shown in subfigure b. Each column shows the results for one input shape, and the rightmost column the mean values with standard deviations across all input shapes. **a** Scaling. **b** Rotation. **c** Gaussian noise. **d** Cutting. **e** Input shapes.

result, as downscaling removes keypoints at low scales. However, the dominant keypoints are still reliably matched. Of particular importance is the fact that during upscaling, a significant proportion of the source keypoints can still be matched correctly. That is, the positions and characteristic scales are reliably detected even under extreme scale changes up to $400\,\%$. For the test shapes *horse-1* and *ray-1*, the proportion is slightly lower. Further analysis suggests that the characteristic scales of dominant keypoints with low curvature values—that is, extrema located in relatively smooth segments—are less stable and therefore can fall outside the tolerance range defined by Equation (4.35).

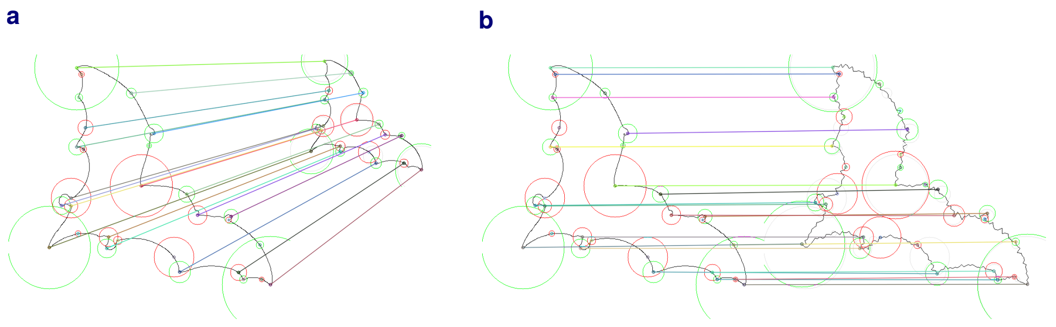


**Fig. 4.16:** Examples of matched keypoints used in the evaluation for the test contour *bat-1*. **a** Matching from $100\,\%$ to $50\,\%$. **b** Matching from $100\,\%$ to a version with Gaussian noise ($\sigma_d = 3\,\mathrm{px}$).

In some cases of the rotation transformations shown in Figure 4.15b, the matching rate significantly decreases with increasing rotation angles. Further analysis shows that this is primarily due to the characteristic scales, which fall outside the tolerance range defined by Equation (4.35), while the positions are reliably detected. The underlying reason is that the discretization of the contours does not ensure uniform sampling of the contour points (pixels) with respect to the Euclidean arc length (i.e., depending on the rotation angle the arc length can significantly change). The effect is particularly prominent for relatively straight segments, such as those present in *device6-1* and *fork-1*, and for small-scale keypoints. To improve the rotation invariance, future work should consider using arc-length parametrization to resample the contour in a uniform manner. However, this does not affect the general methodology.

For the transformations with Gaussian noise in Figure 4.15c, the matching rate only slightly decreases up to $\sigma_d = 3\,\mathrm{px}$, again underlining the robustness of the scale-space approach. At $\sigma_d = 5\,\mathrm{px}$, the local contour structure is significantly altered, and especially small-scale keypoints fall outside the tolerance range defined by Equation (4.35).

For the cutting transformations in Figure 4.15d, the matching rate decreases approximately in proportion to the remaining contour length. At $25\,\%$, the matching rate can be quite low, as the characteristic scales of high-scale keypoints are particularly difficult to estimate when only a small segment of the original contour remains.

## 4.4.2 Prototype Application: Shape Recognition

In preliminary shape recognition experiments on the MPEG-7 dataset (MPEG-7, 1999) using the contour features, a Bull's Eye Score of approximately $50\,\%$ was achieved. The score measures how accurately similar shapes are retrieved: each shape in the dataset is compared to all other shapes (including the query shape), and the number of correct class matches among the 40 most similar shapes is reported. Since each class contains 20 shapes, a maximum of 20 correct matches is possible per shape. The final score is the total number of correct matches across all shapes, divided by the maximum possible number of matches ($20 \cdot 1{,}400 = 28{,}000 \mathrel{\hat{=}} 100\,\%$).

For the experiments, the 10 highest characteristic scales of the curvature maxima and the 10 highest scales of the curvature minima were concatenated into a feature vector, sorted in descending order and grouped by type (curvature minimum or maximum). If a contour had fewer than 10 extrema in either group, the remaining positions were filled with zeros. Standard CSS achieves a score of approximately $75\,\%$ and already falls behind more advanced state-of-the-art methods. However, the contour features from this work are strictly local, their global geometric relations are not considered in these experiments, and using only one training example per shape makes the Bull's Eye Score a challenging benchmark. Therefore, the result is still interpreted as promising.

### 4.4.3 Summary and Final Remarks

In summary, the characteristic scales can be reliably estimated even under extreme scale changes, noise, and partial occlusion. This shows that the methodology of distinguishing local and global characteristics in the signature functions is a meaningful approach. Although the matching rate significantly decreases with increasing rotation angles in some cases, it should be noted that this is primarily due to unmatched small-scale keypoints. In other words, the characteristic scales of larger and therefore more relevant keypoints are often still estimated reliably. Furthermore, arc-length parametrization may considerably improve the robustness under rotation. Discretization artifacts and reduced detection performance are not unique to the method presented in this work and can also be observed in established approaches such as SIFT.

Another modification could be to assign multiple characteristic scales to a keypoint. In SIFT, for example, multiple orientations are assigned to a keypoint when the dominant orientation is ambiguous. As shown in the experiments in Section 4.3.5, the curvature extrema detected in this work can merge during curve evolution. This transition could be identified in the $\sigma$-$u$-plane to capture local and global characteristics simultaneously.

# 5 Conclusion and Future Work

In this work, two new methods for extracting local scale-invariant contour features were presented: the first method assigns characteristic scales to curvature extrema, while the second method introduces an ambiguity model with edge tracing to extract such features from binary edge images. Both methods are robust and parameter-free.

Based on computational experiments using geometric and non-geometric transformations of test contours from the MPEG-7 dataset, it was shown that the contour features can be reliably detected even under extreme scale changes, noise, and partial occlusion. Using the last local minimum of the signature functions as the characteristic scale represents a general principle that helps avoid overfitting. A box filter approximation has been introduced to enable real-time extraction of the contour features.

The contours analyzed in this work have been directly traced and then transformed into their curvature scale-space representation. While this approach has yielded meaningful results, it does not ensure uniform sampling of contour points with respect to the Euclidean arc length. This especially affects the accuracy of the characteristic scales under rotations. Therefore, future work could explore the use of arc-length parametrization to resample the contour in a uniform manner.

For the ambiguity model, it has been shown that the method can effectively resolve complex ambiguities in different application examples. This demonstrates its potential to extract coherent object contours from binary edge images, which can be obtained using modern deep learning-based methods. It has also been found that the model offers a natural and likely ideal way to describe ambiguities directly in the image plane. The ambiguity model is based on four straightforward principles and is designed to work in an intuitive and effective manner.

The next logical step is to use the proposed methods in specific tasks. However, given the progress in object recognition methods, it is reasonable to ask whether this should still be done directly on extracted contours. Instead, it may be more promising to integrate the methods into an end-to-end deep learning-based approach that operates directly on real input images and addresses specific higher-level tasks. A conceptual pathway for this integration is described in Section 5.2.

## 5.1 Traditional vs. Deep Learning-based Features

The contour features presented in this work are based on a traditional manual feature engineering approach (also referred to as hand-crafted features in the literature). Given that local features and even entire objects can be reliably detected using deep learning-based methods (e.g., Georgiou et al., 2020; Ma et al., 2021; Kaur and W. Singh, 2024), it is reasonable to discuss their respective advantages and disadvantages.

A clear disadvantage of deep learning-based methods is that they typically require significant computational resources for training and inference, especially for high-dimensional data. This makes them less practical for real-time applications compared to traditional features. Furthermore, deep learning-based methods require large amounts of labeled training data, which is often not available or expensive to obtain, especially when aiming for robustness under different operational conditions. Additionally, models trained on specific datasets may not generalize well to new domains (overfitting). In comparison, traditional features can be directly extracted for any input image. Another disadvantage of deep learning-based methods is their lack of direct interpretability: they often function as black boxes, making it difficult to understand why a particular feature has been chosen, which can be critical for specific application domains such as autonomous driving.

On the other hand, deep learning-based methods also have considerable advantages. For example, the extraction pipeline can be discussed: Extracting the contour features from real input images requires multiple steps. First, coherent object contours must be extracted from region-based image segmentations or edge images, which already leads to several challenges, such as resolving intersections or junctions of edges (cf. Chapter 3). The next step is analyzing the CSS representation, which—at least without box filter approximations or other optimizations—can be computationally expensive. After determining the characteristic scales, a descriptor can be computed, which can then be used for matching. Here, deep learning-based methods have the advantage that they can integrate all these steps into a single pipeline (Ma et al., 2021, Section 2.4).

While the contour features from this work provide a specific predefined feature set, deep learning-based methods automatically learn to extract a diverse set of robust features for a given task based on the respective objective function. This makes them more invariant to changes in viewpoint, illumination, and other transformations compared to traditional methods. Furthermore, deep learning-based methods can be fine-tuned for different datasets and tasks through transfer learning (Kaur and W. Singh, 2024).

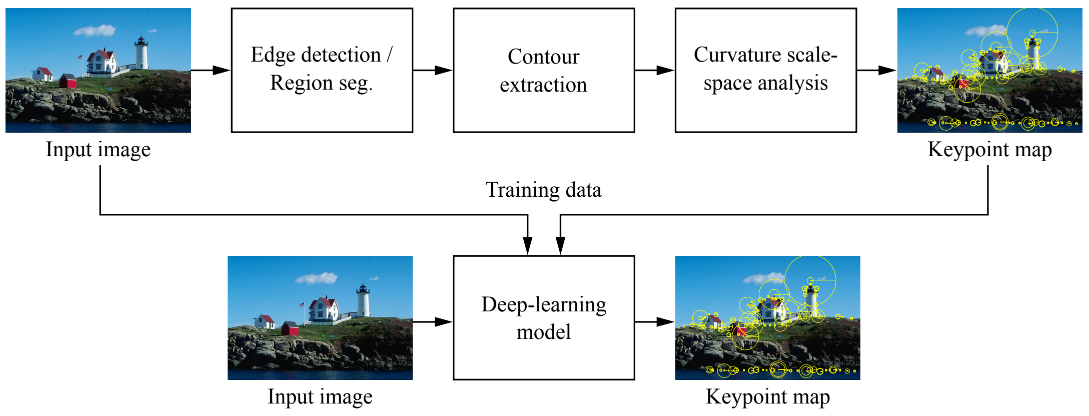


**Fig. 5.1:** Conceptual training setup for a deep learning-based end-to-end feature detection approach using the contour features from this work.

## 5.2 Integration into Deep Learning Frameworks

As deep learning-based methods can integrate feature extraction and matching into a single pipeline (e.g., Ma et al., 2021), it is proposed to use the methods developed in this work to generate training data for a deep learning-based end-to-end feature detection approach. Established approaches for learned feature detection and description include LIFT (Yi et al., 2016), SuperPoint (DeTone et al., 2018), and Key.Net (Barroso-Laguna et al., 2019). Using the methods developed in this work, local contour features can be extracted from both region segmentations and edge images (e.g., using the developed ambiguity model for binary edge images). To generate these intermediate representations, a broad range of deep learning-based methods can be employed (cf. Sections 2.2.5 and 2.2.6). Long runtimes—especially in favor of accuracy—are uncritical if the methods are used for generating training data. A conceptual training setup is shown in Figure 5.1.

The exact network architectures and training setup are left for future work. However, preliminary explorations have already revealed several challenges. One is the class imbalance problem: keypoints are sparse compared to the overall number of pixels. Additionally, since a keypoint is a single pixel, it is difficult to learn a sharp transition from non-keypoint to keypoint, as neighboring pixels are embedded in nearly identical local contexts. Furthermore, both local and global information are required to estimate meaningful characteristic scales. Sufficiently large receptive fields could be considered by using skip connections in U-Net-like architectures. Region segmentations and edge images should also be provided at multiple scales.

Regarding the final evaluation, it may not be ideal to rely solely on ground-truth positions of keypoints, as segmentation annotations—such as those in the BSDS—are manually labeled and inherently subjective. Instead, the performance when using the contour features for a specific task might be a more appropriate evaluation criterion.

# Bibliography

Abbasi, S., Mokhtarian, F., and Kittler, J. (2000). Enhancing CSS-Based Shape Retrieval for Objects With Shallow Concavities. In: *Image and Vision Computing* 18, pp. 199–211. DOI: 10.1016/S0262-8856(99)00019-0.

Abbasi, S., Mokhtarian, F., and Kittler, J. (1999). Curvature Scale Space Image in Shape Similarity Retrieval. In: *Multimedia Systems* 7, pp. 467–476. DOI: 10.1007/s005300050147.

Alt, T., Schrader, K., Augustin, M., Peter, P., and Weickert, J. (2023). Connections Between Numerical Algorithms for PDEs and Neural Networks. In: *Journal of Mathematical Imaging and Vision* 65, pp. 185–208. DOI: 10.1007/s10851-022-01106-x.

Amit, Y., Felzenszwalb, P., and Girshick, R. (2021). Object Detection. In: Computer Vision: A Reference Guide. Ed. by K. Ikeuchi. Springer International Publishing, pp. 875–883. DOI: 10.1007/978-3-030-63416-2_660.

Andreopoulos, A. and Tsotsos, J. K. (2013). 50 Years of Object Recognition: Directions Forward. In: *Computer Vision and Image Understanding* 117(8), pp. 827–891. DOI: 10.1016/j.cviu.2013.04.005.

Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour Detection and Hierarchical Image Segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(5), pp. 898–916. DOI: 10.1109/TPAMI.2010.161.

Attneave, F. (1954). Some Informational Aspects of Visual Perception. In: *Psychological Review* 61(3), pp. 183–193. DOI: 10.1037/h0054663.

Aubert, G. and Kornprobst, P. (2006). Partial Differential Equations and the Calculus of Variations. Mathematical Problems in Image Processing. Springer. DOI: 10.1007/978-0-387-44588-5.

Ayzenberg, V. and Behrmann, M. (2022a). Does the Brain's Ventral Visual Pathway Compute Object Shape? In: *Trends in Cognitive Sciences* 26(12), pp. 1119–1132. DOI: 10.1016/j.tics.2022.09.019.

Ayzenberg, V. and Behrmann, M. (2022b). The Dorsal Visual Pathway Represents Object-Centered Spatial Relations for Object Recognition. In: *The Journal of Neuroscience* 42(23), pp. 4693–4710. DOI: 10.1523/JNEUROSCI.2257-21.2022.

Bai, X., Liu, W., and Tu, Z. (2009). Integrating Contour and Skeleton for Shape Classification. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pp. 360–367. DOI: 10.1109/ICCVW.2009.5457679.

Baker, N., Lu, H., Erlikhman, G., and Kellman, P. J. (2018). Deep Convolutional Networks Do Not Classify Based on Global Object Shape. In: *Computational Biology* 14(12), pp. 1–43. DOI: 10.1371/journal.pcbi.1006613.

Barroso-Laguna, A., Riba, E., Ponsa, D., and Mikolajczyk, K. (2019). Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5836–5844. DOI: 10.48550/arXiv.1904.00889.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). In: *Computer Vision and Image Understanding* 110(3), pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape Matching and Object Recognition Using Shape Contexts. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(4), pp. 509–522. DOI: 10.1109/34.993558.

Benkhlifa, A. and Ghorbel, F. (2017). A Novel 2D Contour Description Generalized Curvature Scale Space. In: *Proc. of the International Workshop on Representations, Analysis and Recognition of Shape and Motion FroM Imaging Data*, pp. 129–140. DOI: 10.1007/978-3-319-60654-5_11.

Berg, A. C., Berg, T. L., and Malik, J. (2005). Shape Matching and Object Recognition Using Low Distortion Correspondences. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26–33. DOI: 10.1109/CVPR.2005.320.

Berrada, F., Aboutajdine, D., Ouatik, S. E., and Lachkar, A. (2011). Review of 2D Shape Descriptors Based on the Curvature Scale Space Approach. In: *Proc. of the International Conference on Multimedia Computing and Systems*, pp. 1–6. DOI: 10.1109/ICMCS.2011.5945600.

Bhatia, A., Snyder, W. E., and Bilbro, G. (2010). Stacked Integral Image. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1530–1535. DOI: 10.1109/ROBOT.2010.5509400.

Biederman, I. (1987). Recognition-by-Components: A Theory of Human Image Understanding. In: *Psychological Review* 94(2), pp. 115–147. DOI: 10.1037/0033-295X.94.2.115.

Biederman, I. and Ju, G. (1988). Surface versus Edge-based Determinants of Visual Recognition. In: *Cognitive Psychology* 20, pp. 38–64. DOI: 10.1016/0010-0285(88)90024-2.

Billauer, E. (2012). *Peakdet: Peak Detection Using MATLAB*. Last checked: 2025-01-15.

Bracci, S. and Op de Beeck, H. P. (2023). Understanding Human Object Vision: A Picture Is Worth a Thousand Representations. In: *Annual Review of Psychology* 74, pp. 113–135. DOI: 10.1146/annurev-psych-032720-041031.

Bramão, I., Reis, A., Petersson, K. M., and Faísca, L. (2011). The Role of Color Information on Object Recognition: A Review and Meta-Analysis. In: *Acta Psychologica* 138, pp. 244–253. DOI: 10.1016/j.actpsy.2011.06.010.

Bresenham, J. E. (1965). Algorithm for Computer Control of a Digital Plotter. In: *IBM Systems Journal* 4(1), pp. 25–30. DOI: 10.1147/sj.41.0025.

Brezovjakova, H., Tomlinson, C., Mohd Naim, N., et al. (2019). Junction Mapper is a Novel Computer Vision Tool to Decipher Cell–Cell Contact Phenotypes. In: *eLife* 8(e45413), pp. 1–48. DOI: 10.7554/eLife.45413.

Buades, A., Grompone von Gioi, R., and Navarro, J. (2018). Joint Contours, Corner and T-Junction Detection: An Approach Inspired by the Mammal Visual System. In: *Journal of Mathematical Imaging and Vision* 60, pp. 341–354. DOI: s10851-017-0763-z.

Bülthoff, H. H. and Edelman, S. (1992). Psychophysical Support for a Two-Dimensional View Interpolation Theory of Object Recognition. In: *Psychological Review* 89, pp. 60–64. DOI: 10.1037/0033-295X.94.2.115.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 778–792. DOI: 10.1007/978-3-642-15561-1_56.

Canny, J. (1986). A Computational Approach to Edge Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8(6), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

Casadei, S. and Mitter, S. (1999). Beyond the Uniqueness Assumption: Ambiguity Representation and Redundancy Elimination in the Computation of a Covering Sample of Salient Contour Cycles. In: *Computer Vision and Image Understanding* 76(1), pp. 19–35. DOI: 10.1006/cviu.1999.0790.

Cole, F., Sanik, K., DeCarlo, D., et al. (2009). How Well Do Line Drawings Depict Shape? In: *ACM Transactions on Graphics (Proc. SIGGRAPH)*. Vol. 28. 3, pp. 1–9. DOI: 10.1145/1576246.1531334.

Connor, D. and Krivodonova, L. (2014). Interpolation of Two-Dimensional Curves with Euler Spirals. In: *Journal of Computational and Applied Mathematics* 261, pp. 320–332. DOI: 10.1016/j.cam.2013.11.009.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual Categorization With Bags of Keypoints. In: *Proc. of the ECCV Workshop on Statistical Learning in Computer Vision*, pp. 1–16.

Dabiri, Z. and Blaschke, T. (2019). Scale Matters: A Survey of the Concepts of Scale Used in Spatial Disciplines. In: *European Journal of Remote Sensing* 52(1), pp. 419–434. DOI: 10.1080/22797254.2019.1626291.

De Winter, J. and Wagemans, J. (2006). Segmentation of Object Outlines Into Parts: A Large-Scale Integrative Study. In: *Cognition* 99(3), pp. 275–325. DOI: 10.1016/j.cognition.2005.03.004.

De Winter, J. and Wagemans, J. (2008). Perceptual Saliency of Points Along the Contour of Everyday Objects: A Large-Scale Study. In: *Perception & Psychophysics* 70(1), pp. 50–64. DOI: 10.3758/PP.70.1.50.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). SuperPoint: Self-Supervised Interest Point Detection and Description. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 337–349. DOI: 10.1109/CVPRW.2018.00060.

DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How Does the Brain Solve Visual Oject Recognition? In: *Neuron* 73(3), pp. 415–434. DOI: 10.1016/j.neuron.2012.01.010.

Dickinson, S. J. and Pizlo, Z., eds. (2013). Shape Perception in Human and Computer Vision – An Interdisciplinary Perspective. Advances in Computer Vision and Pattern Recognition. Springer. DOI: 10.1007/978-1-4471-5195-1.

Drew, M. S., Lee, T. K., and Rova, A. (2009). Shape Retrieval With Eigen-CSS Search. In: *Image and Vision Computing* 27, pp. 748–755. DOI: 10.1016/j.imavis.2008.07.011.

Eames, C. and Eames, R. (1977). *Powers of Ten*. Documentary, Short. Narrated by Phil Morrison, Produced by IBM.

Elboher, E. and Werman, M. (2012). Efficient and Accurate Gaussian Image Filtering Using Running Sums. In: *Proc. of the International Conference on Intelligent Systems Design and Applications*, pp. 897–902. DOI: 10.1109/ISDA.2012.6416657.

Evans, L. C. (2010). Partial Differential Equations. 2nd ed. Vol. 19. Graduate Studies in Mathematics. American Mathematical Society. DOI: 10.1007/978-0-8176-4552-6.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challengey. In: *International Journal of Computer Vision* 88(2), pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

Feldman, J. and Singh, M. (2005). Information Along Contours and Object Boundaries. In: *Psychological Review* 112(1), pp. 243–252. DOI: 10.1037/0033-295X.112.1.243.

Felzenszwalb, P. F. and Schwartz, J. D. (2007). Hierarchical Matching of Deformable Shapes. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. DOI: 10.1109/CVPR.2007.383018.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. In: *International Journal of Computer Vision* 59(2), pp. 167–181. DOI: 10.1023/B:VISI.0000022288.19776.77.

Ferrari, V., Fevrier, L., Jurie, F., and Schmid, C. (2008). Groups of Adjacent Contour Segments for Object Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(1), pp. 36–51. DOI: 10.1109/TPAMI.2007.1144.

Field, G. D. and Chichilnisky, E. J. (2007). Information Processing in the Primate Retina: Circuitry and Coding. In: *Annual Review of Neuroscience* 30, pp. 1–30. DOI: 10.1146/annurev.neuro.30.051606.094252.

Gauglitz, S., Höllerer, T., and Turk, M. (2011). Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking. In: *International Journal of Computer Vision* 94, pp. 335–360. DOI: 10.1007/s11263-011-0431-5.

Ge, Y., Xiao, Y., Xu, Z., Wang, X., and Itti, L. (2022). Contributions of Shape, Texture, and Color in Visual Recognition. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 369–386. DOI: 10.1007/978-3-031-19775-8_22.

Georgiou, T., Liu, Y., Chen, W., and Lew, M. (2020). A Survey of Traditional and Deep Learning-Based Feature Descriptors for High-Dimensional Data in Computer Vision. In: *International Journal of Multimedia Information Retrieval* 9, pp. 135–170. DOI: 10.1007/s13735-019-00183-w.

Getreuer, P. (2013). A Survey of Gaussian Convolution Algorithms. In: *Image Processing On Line* 2013, pp. 286–310. DOI: 10.5201/ipol.2013.87.

Ghuneim, A. G. (2000). *Moore-Neighbor Tracing*. https://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/index.html. Accessed: 2024-12-10.

Gonzalez, R. C. and Woods, R. E. (2018). Digital Image Processing. 4th Global Ed. Pearson.

Grauman, K. and Leibe, B. (2011). Visual Object Recognition. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 5(2). DOI: 10.1007/978-3-031-01553-3.

Guo, Y., Kumar, N., Narayanan, M., and Kimia, B. (2014). A Multi-stage Approach to Curve Extraction. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 663–678. DOI: 10.1007/978-3-319-10590-1_43.

Gupta, S. and Kar, S. (2022). Algorithms to Speed up Contour Tracing in Real Time Image Processing Systems. In: *IEEE Access* 10, pp. 127365–127376. DOI: 10.1109/ACCESS.2022.3226943.

Gwosdek, P., Grewenig, S., Bruhn, A., and Weickert, J. (2011). Theoretical Foundations of Gaussian Convolution by Extended Box Filtering. In: *Proc. of the International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 447–458. DOI: `10.1007/978-3-642-24785-9_38`.

Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. In: *Proc. of the Alvey Vision Conference*, pp. 147–152. DOI: `10.5244/C.2.23`.

He, L., Ren, X., Gao, Q., et al. (2017). The Connected-Component Labeling Problem: A Review of State-of-the-Art Algorithms. In: *Pattern Recognition* 70, pp. 25–43. DOI: `10.1016/j.patcog.2017.04.018`.

He, X.-C. and Yung, N. H. (2004). Curvature Scale Space Corner Detector With Adaptive Threshold and Dynamic Region of Support. In: *Proc. of the International Conference on Pattern Recognition (ICPR)*. Vol. 2, pp. 791–794. DOI: `10.1109/ICPR.2004.1334377`.

Heinly, J., Dunn, E., and Frahm, J.-M. (2012). Comparative Evaluation of Binary Features. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 579–773. DOI: `10.1007/978-3-642-33709-3_54`.

Heng, W. J. and Ngan, K. N. (2001). An Object-Based Shot Boundary Detection Using Edge Tracing and Tracking. In: *Journal of Visual Communication and Image Representation* 12(3), pp. 217–239. DOI: `10.1006/jvci.2001.0457`.

Hennig, M. (2025). Estimating Robust Characteristic Scales for Contour Curvature Extrema. In: Submission in Preparation.

Hennig, M., Leineke, M., and Mertsching, B. (2024). *A General Ambiguity Model for Binary Edge Images with Edge Tracing and its Implementation*. arXiv. DOI: `10.48550/arXiv.2408.01712`.

Hennig, M. and Mertsching, B. (2016). Contour Analysis in Biological and Artificial Cognitive Systems. In: *Proc. of the 5th Interdisciplinary Workshop on Cognitive Systems*, pp. 1–6.

Hennig, M. and Mertsching, B. (2021). Box Filtering for Real-Time Curvature Scale-Space Computation. In: *Journal of Physics: Conference Series* 1958, pp. 1–7. DOI: `10.1088/1742-6596/1958/1/012020`.

Horton, Z. (2021). The Cosmic Zoom: Scale, Knowledge, and Mediation. The University of Chicago Press. DOI: `10.7208/chicago/9780226742588.001.0001`.

Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2016). What Makes for Effective Detection Proposals? In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(4), pp. 814–830. DOI: `10.1109/TPAMI.2015.2465908`.

Huang, K., Wang, Y., Zhou, Z., et al. (June 2018). Learning to Parse Wireframes in Images of Man-Made Environments. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 626–635. DOI: 10.1109/CVPR.2018.00072.

Huang, Q., Guo, X., Wang, Y., Sun, H., and Yang, L. (2024). A Survey of Feature Matching Methods. In: *IET Image Processing* 18(6), pp. 1385–1410. DOI: 10.1049/ipr2.13032.

Hubel, D. H. and Wiesel, T. N. (2004). Brain and Visual Perception – The Story of a 25-year Collaboration. Oxford University Press. DOI: 10.1093/acprof:oso/9780195176186.001.0001.

Hund, M. (2009). Perzeptuelle Organisation von Objektgrenzen unter Verwendung anisotroper Regularisierungsmethoden. In German language. Ph.D. thesis. GET Lab, Paderborn University.

Jing, J., Liu, S., Wang, G., Zhang, W., and Sun, C. (2022). Recent Advances on Image Edge Detection: A Comprehensive Review. In: *Neurocomputing* 503, pp. 259–271. DOI: 10.1016/j.neucom.2022.06.083.

Jurie, F. and Schmid, C. (2004). Scale-Invariant Shape Features for Recognition of Object Categories. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2, pp. 90–96. DOI: 10.1109/CVPR.2004.1315149.

Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-Learning-Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(7), pp. 1409–1422. DOI: 10.1109/TPAMI.2011.239.

Kaur, J. and Singh, W. (2024). A Systematic Review of Object Detection From Images Using Deep Learning. In: *Multimedia Tools and Applications* 83(4), pp. 12253–12338. DOI: 10.1007/s11042-023-15981-y.

Kawamura, K., Ishii, D., and Watanabe, H. (2011). Automatic Scale Detection for Contour Fragment Based on Difference of Curvature. In: *IEICE Transactions on Information and Systems* E94-D(10), pp. 1998–2005. DOI: 10.1587/transinf.E94.D.1998.

Khaldi, B., Aiadi, O., and Kherfi, M. L. (2019). Combining Colour and Grey-Level Co-Occurrence Matrix Features: A Comparative Study. In: *IET Image Processing* 13(9), pp. 1401–1410. DOI: 10.1049/iet-ipr.2018.6440.

Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A Survey of the Recent Architectures of Deep Convolutional Neural Networks. In: *Artificial Intelligence Review* 53, pp. 5455–5516. DOI: 10.1007/s10462-020-09825-6.

Kimia, B. B., Li, X., Guo, Y., and Tamrakar, A. (2019). Differential Geometry in Edge Detection: Accurate Estimation of Position, Orientation and Curvature. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41(7), pp. 1573–1586. DOI: 10.1109/TPAMI.2018.2846268.

Kirillov, A., Mintun, E., Ravi, N., et al. (2023). Segment Anything. In: *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003. DOI: 10.1109/ICCV51070.2023.00371.

Koenderink, J. J. (1984). The Structure of Images. In: *Biological Cybernetics* 50, pp. 363–370. DOI: 10.1007/BF00336961.

Koenderink, J. J. (2021). The Structure of Images: 1984–2021. In: *Biological Cybernetics* 115, pp. 117–120. DOI: 10.1007/s00422-021-00870-0.

Kovesi, P. (2010). Fast Almost-Gaussian Filtering. In: *Proc. of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 121–125. DOI: 10.1109/DICTA.2010.30.

Kuijper, A. (2002). The Deep Structure of Gaussian Scale-Space Images. PhD Thesis. Utrecht University, Netherlands.

Kurnianggoro, L., Wahyono, and Jo, K.-H. (2018). A Survey of 2D Shape Representation: Methods, Evaluations, and Future Research Directions. In: *Neurocomputing* 300, pp. 1–16. DOI: 10.1016/j.neucom.2018.02.093.

Latecki, L. J. and Lakämper, R. (1999). Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution. In: *Computer Vision and Image Understanding* 73(3), pp. 441–454. DOI: 10.1006/cviu.1998.0738.

Law, T., Itoh, H., and Seki, H. (1996). Image Filtering, Edge Detection, and Edge Tracing Using Fuzzy Reasoning. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(5), pp. 481–491. DOI: 10.1109/34.494638.

Li, X., Hu, W., Shen, C., et al. (2013). A Survey of Appearance Models in Visual Object Tracking. In: *ACM Transactions on Intelligent Systems and Technology* 4(4), 58:1–58:48. DOI: 10.1145/2508037.2508039.

Li, Y., Wang, S., Tian, Q., and Ding, X. (2015). A Survey of Recent Advances in Visual Feature Detection. In: *Neurocomputing* 149, pp. 736–751. DOI: 10.1016/j.neucom.2014.08.003.

Lifshitz, L. and Pizer, S. (1990). A Multiresolution Hierarchical Approach to Image Segmentation Based on Intensity Extrema. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(6), pp. 529–540. DOI: 10.1109/34.56189.

Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common Objects in Context. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

Lindeberg, T. (1993). Discrete Derivative Approximations With Scale-Space Properties: A Basis for Low-Level Feature Extraction. In: *Journal of Mathematical Imaging and Vision* 3, pp. 349–376. DOI: 10.1007/BF01664794.

Lindeberg, T. (1994a). Scale-Space Theory in Computer Vision. In: vol. 256. The Springer International Series in Engineering and Computer Science. Springer. DOI: 10.1007/978-1-4757-6465-9.

Lindeberg, T. (1994b). Scale-Space Theory: A Basic Tool for Analyzing Structures at Different Scales. In: *Journal of Applied Statistics* 21(1-2), pp. 225–270. DOI: 10.1080/757582976.

Lindeberg, T. (1997). In: Gaussian Scale-Space Theory. Ed. by J. Sporring, M. Nielsen, L. Florack, and P. Johansen. Chap. On the Axiomatic Foundations of Linear Scale-Space, pp. 75–97. DOI: 10.1007/978-94-015-8802-7_6.

Lindeberg, T. (2008). In: Wiley Encyclopedia of Computer Science and Engineering. Ed. by B. Wah. Chap. Scale-Space, pp. 2495–2504. DOI: 10.1002/9780470050118.ecse609.

Lindeberg, T. (2013). Generalized Axiomatic Scale-Space Theory (Chapter 1). In: Advances in Imaging and Electron Physics. Ed. by P. W. Hawkes. Vol. 178. Elsevier, pp. 1–96. DOI: 10.1016/B978-0-12-407701-0.00001-7.

Lindeberg, T. (2014). In: Computer Vision: A Reference Guide. Ed. by K. Ikeuchi. Chap. Scale Selection, pp. 701–713.

Lindeberg, T. (2021). Normative Theory of Visual Receptive Fields. In: *Heliyon* 7(1), pp. 1–20. DOI: 10.1016/j.heliyon.2021.e05897.

Lindeberg, T. (2024). Discrete Approximations of Gaussian Smoothing and Gaussian Derivatives. In: *Journal of Mathematical Imaging and Vision* 66, pp. 759–800. DOI: 10.1007/s10851-024-01196-9.

Lindeberg, T. and Ter Haar Romeny, B. M. (1994). In: Geometry-Driven Diffusion in Computer Vision. Ed. by B. M. Ter Haar Romeny. Chap. Linear Scale-Space I: Basic Theory, pp. 1–72. DOI: 10.1007/978-94-017-1699-4.

Ling, H. and Jacobs, D. W. (2007). Shape Classification Using the Inner-Distance. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(2), pp. 286–299. DOI: 10.1109/TPAMI.2007.41.

Liu, H., Latecki, L. J., and Liu, W. (2008). A Unified Curvature Definition for Regular, Polygonal, and Digital Planar Curves. In: *International Journal of Computer Vision* 80, pp. 104–124. DOI: 10.1007/s11263-008-0131-y.

Loffler, G. (2008). Perception of Contours and Shapes: Low and Intermediate Stage Mechanisms. In: *Vision Research* 48(1), pp. 2106–2127. DOI: 10.1016/j.visres.2008.03.006.

Lowe, D. G. (1999). Object Recognition from Local Scale-Invariant Features. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*. Vol. 2, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60(2), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94.

Ma, J., Jiang, X., Fan, A., Jiang, J., and Yan, J. (2021). Image Matching from Handcrafted to Deep Features: A Survey. In: *International Journal of Computer Vision* 129, pp. 23–79. DOI: 10.1007/s11263-020-01359-2.

Maire, M., Arbelaez, P., Fowlkes, C., and Malik, J. (2008). Using Contours to Detect and Localize Junctions in Natural Images. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. DOI: 10.1109/CVPR.2008.4587420.

Marr, D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. MIT Press Edition, 2010. The MIT Press. DOI: 10.7551/mitpress/9780262514620.001.0001.

Mikolajczyk, K., Zisserman, A., and Schmid, C. (2003). Shape Recognition With Edge-Based Features. In: *Proc. of the British Machine Vision Conference*. Vol. 2, pp. 779–788.

Minaee, S., Boykov, Y., Porikli, F., et al. (2022). Image Segmentation Using Deep Learning: A Survey. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(7), pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

Mokhtarian, F. and Mackworth, A. (1992). A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(8), pp. 789–805. DOI: 10.1109/34.149591.

Mokhtarian, F. and Bober, M. (2003). Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization. Springer Science & Business Media. DOI: 10.1007/978-94-017-0343-7.

Mokhtarian, F. and Suomela, R. (1998). Robust Image Corner Detection Through Curvature Scale Space. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(12), pp. 1376–1381. DOI: 10.1109/34.735812.

MPEG-7 (1999). *MPEG-7 Core Experiment CE-Shape-1 Test Set (Part B)*. Accessed: 2025-02-13; Prepared by Ralph, R.

Mziou-Sallami, M., Khalsi, R., Smati, I., Mhiri, S., and Ghorbel, F. (2023). DeepGCSS: A Robust and Explainable Contour Classifier Providing Generalized Curvature Scale Space Features. In: *Neural Computing and Applications* 35, pp. 17689–17700. DOI: 10.1007/s00521-023-08639-1.

Nilsback, M.-E. and Zisserman, A. (2008). Automated Flower Classification over a Large Number of Classes. In: *Proc. of the Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 722–729. DOI: 10.1109/ICVGIP.2008.47.

Norman, J. F., Phillips, F., and Ross, H. E. (2001). Information Concentration Along the Boundary Contours of Naturally Shaped Solid Objects. In: *Perception* 30(11), pp. 1285–1294. DOI: 10.1068/p3272.

Opelt, A., Pinz, A., and Zisserman, A. (2006). A Boundary-Fragment-Model for Object Detection. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 575–588. DOI: 10.1007/11744047_44.

Pasupathy, A., Kim, T., and Popovkina, D. V. (2019). Object Shape and Surface Properties Are Jointly Encoded in Mid-Level Ventral Visual Cortex. In: *Current Opinion in Neurobiology* 58, pp. 199–208. DOI: 10.1016/j.conb.2019.09.009.

Perona, P. and Malik, J. (1990). Scale-Space and Edge Detection Using Anisotropic Diffusion. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(7), pp. 629–639. DOI: 10.1109/34.56205.

Pham, T.-A., Delalandre, M., Barrat, S., and Ramel, J.-Y. (2014). Accurate Junction Detection and Characterization in Line-Drawing Images. In: *Pattern Recognition* 47(1), pp. 282–295. DOI: 10.1016/j.patcog.2013.06.027.

Poggio, T. and Ullman, S. (2013). Vision: Are Models of Object Recognition Catching Up With the Brain? In: *Annals of the New York Academy of Sciences* 1305, pp. 72–82. DOI: 10.1111/nyas.12148.

Prasad, D. K. (2012). Survey of the Problem of Object Detection in Real Images. In: *International Journal of Image Processing* 6(6), pp. 441–466.

Pu, M., Huang, Y., Liu, Y., Guan, Q., and Ling, H. (2022). EDTER: Edge Detection With Transformer. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1402–1412. DOI: 10.1109/CVPR52688.2022.00146.

Rattarangsi, A. and Chin, R. (1992). Scale-Based Detection of Corners of Planar Curves. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(4), pp. 430–449. DOI: 10.1109/34.126805.

Rock, I. and Palmer, S. (1990). The Legacy of Gestalt Psychology. In: *Scientific American* 263(6), pp. 84–91. DOI: 10.1038/scientificamerican1290-84.

Rosten, E. and Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 430–443. DOI: 10.1007/11744023_34.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

Saha, P. K., Borgefors, G., and Baja, G. S. di (2016). A Survey on Skeletonization Algorithms and Their Applications. In: *Pattern Recognition Letters* 76, pp. 3–12. DOI: 10.1016/j.patrec.2015.04.006.

Sebastian, T. B., Klein, P. N., and Kimia, B. B. (2004). Recognition of Shapes by Editing Their Shock Graphs. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(5), pp. 550–571. DOI: 10.1109/TPAMI.2004.1273924.

Seo, J., Chae, S., Shim, J., et al. (2016). Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors. In: *Sensors* 16(3), pp. 1–27. DOI: 10.3390/s16030353.

Shen, W., Jiang, Y., Gao, W., Zeng, D., and Wang, X. (2016). Shape Recognition by Bag of Skeleton-Associated Contour Parts. In: *Pattern Recognition Letters* 83, pp. 321–329. DOI: 10.1016/j.patrec.2016.02.002.

Shotton, J., Blake, A., and Cipolla, R. (2008). Multiscale Categorical Object Recognition Using Contour Fragments. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(7), pp. 1270–1281. DOI: 10.1109/TPAMI.2007.70772.

Silkan, H., Ouatik, S. E., Lachkar, A., and Meknassi, M. (2009). A Novel Shape Descriptor Based on Extreme Curvature Scale Space Map Approach for Efficient Shape Similarity Retrieval. In: *Proc. of the International Conference on Signal Image Technology and Internet Based Systems*, pp. 160–163. DOI: 10.1109/SITIS.2009.35.

Singh, M. (2015). Visual Representation of Contour and Shape. In: The Oxford Handbook of Perceptual Organization. Oxford University Press. DOI: 10.1093/oxfordhb/9780199686858.013.061.

Soori, M., Arezoo, B., and Dastres, R. (2023). Artificial Intelligence, Machine Learning, and Deep Learning in Advanced Robotics: A Review. In: *Cognitive Robotics* 3, pp. 54–70. DOI: 10.1016/j.cogr.2023.04.001.

Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., and Ginneken, B. van (2004). Ridge-based Vessel Segmentation in Color Images of the Retina. In: *IEEE Transactions on Medical Imaging* 23(4), pp. 501–509. DOI: 10.1109/TMI.2004.825627.

Su, Z., Zhang, J., Wang, L., et al. (2023). Lightweight Pixel Difference Networks for Efficient Visual Representation Learning. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(12), pp. 14956–14974. DOI: 10.1109/TPAMI.2023.3300513.

Suzuki, S. et al. (1985). Topological Structural Analysis of Digitized Binary Images by Border Following. In: *Computer Vision, Graphics, and Image Processing* 30(1), pp. 32–46. DOI: 10.1016/0734-189X(85)90016-7.

Tamrakar, A. and Kimia, B. B. (2007). No Grouping Left Behind: From Edges to Curve Fragments. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8. DOI: 10.1109/ICCV.2007.4408919.

Tarr, M. J. and Bülthoff, H. H. (1998). Image-Based Object Recognition in Man, Monkey, and Machine. In: *Cognition* 67(1–2), pp. 1–20. DOI: 10.1016/s0010-0277(98)00026-2.

Teh, C.-H. and Chin, R. (1989). On the Detection of Dominant Points on Digital Curves. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11(8), pp. 859–872. DOI: 10.1109/34.31447.

Ter Haar Romeny, B. M. (1996). *Introduction to Scale-Space Theory – Multiscale Geometric Image Analysis*. Tech. rep. ICU-96-21. Tutorial for the 4th International Conference on Visualization in Biomedical Computing. Utrecht University.

Ter Haar Romeny, B. M. (2003). Front-End Vision and Multi-Scale Image Analysis – Multi-Scale Computer Vision Theory and Applications, written in Mathematica. Computational Imaging and Vision. Springer. DOI: 10.1007/978-1-4020-8840-7.

Tsourounis, D., Kastaniotis, D., Theoharatos, C., Kazantzidis, A., and Economou, G. (2022). SIFT-CNN: When Convolutional Neural Networks Meet Dense SIFT Descriptors for Image and Sequence Classification. In: *Journal of Imaging* 8(10), pp. 1–18. DOI: 10.3390/jimaging8100256.

Tünnermann, J., Hennig, M., Silbernagel, M., and Mertsching, B. (2013). *A Prototyping Environment for Integrated Artificial Attention Systems*. arXiv. DOI: 10.48550/arXiv.1307.8233.

Tuytelaars, T. and Mikolajczyk, K. (2008). Local Invariant Feature Detectors: A Survey. In: *Foundations and Trends in Computer Graphics and Vision* 3(3), pp. 177–280. DOI: 10.1561/0600000017.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective Search for Object Recognition. In: *International Journal of Computer Vision* 104, pp. 154–171. DOI: 10.1007/s11263-013-0620-5.

University at Buffalo (2007). *CEDAR Signature Dataset*. Accessed: 2024-12-18.

Walther, D. B., Chai, B., Caddigan, E., Beck, D. M., and Fei-Fei, L. (2011). Simple Line Drawings Suffice for Functional MRI Decoding of Natural Scene Categories. In: *Proc. of the National Academy of Sciences* 108(23), pp. 9661–9666. DOI: 10.1073/pnas.1015666108.

Wang, G., Huang, Y., Ma, K., et al. (2023). Automatic Vessel Crossing and Bifurcation Detection Based on Multi-attention Network Vessel Segmentation and Directed Graph Search. In: *Computers in Biology and Medicine* 155, pp. 1–11. DOI: 10.1016/j.compbiomed.2023.106647.

Wang, L., Li, S., Sun, Z., et al. (2018). Segmentation of Yeast Cell's Bright-field Image with an Edge-tracing Algorithm. In: *Journal of Biomedical Optics* 23(11). DOI: 10.1117/1.JBO.23.11.116503.

Wang, X., Feng, B., Bai, X., Liu, W., and Latecki, L. J. (2014). Bag of Contour Fragments for Robust Shape Classification. In: *Pattern Recognition Letters* 47(6), pp. 2116–2125. DOI: 10.1016/j.patcog.2013.12.008.

Weickert, J. (1998). Anisotropic Diffusion in Image Processing. B.G. Teubner.

Weickert, J., Ishikawa, S., and Imiya, A. (1999). Linear Scale-Space Has First Been Proposed in Japan. In: *Journal of Mathematical Imaging and Vision* 10(3), pp. 237–252. DOI: 10.1023/A:1008344623873.

Wells, W. M. (1986). Efficient Synthesis of Gaussian Filters by Cascaded Uniform Filters. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8(2), pp. 234–239. DOI: 10.1109/TPAMI.1986.4767776.

Wichmann, F. A. and Geirhos, R. (2023). Are Deep Neural Networks Adequate Behavioral Models of Human Visual Perception? In: *Annual Review of Vision Science* 9(1), pp. 501–524. DOI: 10.1146/annurev-vision-120522-031739.

Witkin, A. P. (1983). Scale-Space Filtering. In: *Proc. of the 8th International Joint Conference on Artificial Intelligence*. Vol. 2, pp. 1019–1022.

Wolfe, J. M., Levi, D. M., Holt, L. L., et al. (2024). Sensation & Perception. 7th Edition. Oxford University Press.

Xie, S. and Tu, Z. (2015). Holistically-Nested Edge Detection. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1395–1403. DOI: 10.1109/ICCV.2015.164.

Xu, C. and Kuipers, B. (2011). Object detection using principal contour fragments. In: *Proc. of the Canadian Conference on Computer and Robot Vision*, pp. 363–370. DOI: 10.1109/CRV.2011.55.

Yang, D., Peng, B., Al-Huda, Z., Malik, A., and Zhai, D. (2022). An Overview of Edge and Object Contour Detection. In: *Neurocomputing* 488, pp. 470–493. DOI: 10.1016/j.neucom.2022.02.079.

Yang, M., Kpalma, K., and Ronsin, J. (2008). A Survey of Shape Feature Extraction Techniques. In: *Pattern Recognition Techniques, Technology and Applications*, pp. 43–90. DOI: 10.5772/6237.

Yang, X., Yan, J., Wang, W., et al. (2022). gopired Models for Visual Object Recognition: An Overview. In: *Artificial Intelligence Review* 55, pp. 5263–5311. DOI: 10.1007/s10462-021-10130-z.

Yi, K. M., Trulls, E., Lepetit, V., and Fua, P. (2016). LIFT: Learned Invariant Feature Transform. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 467–483. DOI: 10.1007/978-3-319-46466-4_28.

Young, R. A. (1987). The Gaussian Derivative Model for Spatial Vision: I. Retinal Mechanisms. In: *Spatial Vision* 2(4), pp. 273–293. DOI: 10.1163/156856887x00222.

Yu, Y., Wang, C., Fu, Q., et al. (2023). Techniques and Challenges of Image Segmentation: A Review. In: *Electronics* 12(5), pp. 1–24. DOI: 10.3390/electronics12051199.

Zhang, D. and Lu, G. (2004). Review of Shape Representation and Description Techniques. In: *Pattern Recognition* 37(1), pp. 1–19. DOI: 10.1016/j.patcog.2003.07.008.

Zhang, T. Y. and Suen, C. Y. (1984). A Fast Parallel Algorithm for Thinning Digital Patterns. In: *Communications of the ACM* 27(3), pp. 236–239. DOI: 10.1145/357994.358023.

Zhang, X., Wang, H., Hong, M., et al. (2009). Robust Image Corner Detection Based on Scale Evolution Difference of Planar Curves. In: *Pattern Recognition Letters* 30(4), pp. 449–455. DOI: 10.1016/j.patrec.2008.11.002.

Zhao, J., Zhao, W., Deng, B., et al. (2024). Autonomous Driving System: A Comprehensive Survey. In: *Expert Systems with Applications* 242, pp. 1–27. DOI: 10.1016/j.eswa.2023.122836.

Zhu, Q., Song, G., and Shi, J. (2007). Untangling Cycles for Contour Grouping. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8. DOI: 10.1109/ICCV.2007.4408929.

Zitnick, C. L. and Dollár, P. (2014). Edge Boxes: Locating Object Proposals from Edges. In: *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 391–405. DOI: 10.1007/978-3-319-10602-1_26.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object Detection in 20 Years: A Survey. In: *Proc. of the IEEE* 111(3), pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.

# List of Acronyms

| | |
|---|---|
| BRIEF | Binary Robust Independent Elementary Features (Calonder et al., 2010) |
| BSDS | Berkeley Segmentation Dataset (BSDS500; Arbeláez et al., 2011) |
| CCL | Connected Component Labeling (cf. L. He et al., 2017) |
| CNN | Convolutional neural network |
| CSS | Curvature Scale Space |
| DNN | Deep neural network |
| DoC | Difference of Curvature |
| DoG | Difference of Gaussians |
| EBF | Extended Box Filtering (Gwosdek et al., 2011) |
| FAG | Fast Almost Gaussian Smoothing (Kovesi, 2010) |
| FAST | Features from Accelerated Segment Test (Rosten and Drummond, 2006) |
| FCM | Find Contours Method (Suzuki et al., 1985) |
| fMRI | Functional Magnetic Resonance Imaging |
| GCSS | Generalized Curvature Scale Space |
| gPb | Globalized Probability of Boundary |
| HED | Holistically-Nested Edge Detection (Xie and Tu, 2015) |
| IT | Inferior temporal cortex |
| LoG | Laplacian of Gaussian |
| LGN | Lateral geniculate nucleus |

| | |
|---|---|
| MNT | Moore-Neighbor Tracing (cf. Ghuneim, 2000) |
| MPAs | Multi-pixel ambiguities |
| OpenCV | Open Source Computer Vision Library |
| ORB | Oriented FAST and Rotated BRIEF (Rublee et al., 2011) |
| owt | Oriented Watershed Transform |
| PDE | Partial differential equation |
| SB | Stacked Boxes (Bhatia et al., 2010; Elboher and Werman, 2012) |
| SBF | Simple Box Filtering (Wells, 1986; Kovesi, 2010) |
| SIFT | Scale-Invariant Feature Transform (Lowe, 1999; 2004) |
| SPAs | Single-pixel ambiguities |
| SURF | Speeded-Up Robust Features (Bay et al., 2008) |
| ucm | Ultrametric Contour Map |
| V1, V2, V4 | Functional areas of the visual cortex in the brain |

# List of Figures

# List of Tables

# List of Symbols

Note: Some symbols with strictly local scope and variants of the same variable (e.g., with additional indices) are not listed for brevity.

## General Symbols and Operators

| | |
|---|---|
| $\mathbb{B}$ | Set of binary values $\{0, 1\}$ |
| $\partial_t$ | Partial derivative w.r.t. time $t$ (scale), short for $\partial/\partial t$ |
| $\partial_{xx}$ | Second partial derivative w.r.t. $x$, short for $\partial^2/\partial x^2$ |
| $\partial_{yy}$ | Second partial derivative w.r.t. $y$, short for $\partial^2/\partial y^2$ |
| $\nabla^2$ | Laplace operator (second-order differential operator) |
| $\exp$ | Exponential function (base $e$) |
| $\ln$ | Natural logarithm |
| $\mathcal{O}$ | Big-O notation (upper bound of algorithmic complexity) |
| $\mathbb{R}$ | Set of real numbers |
| $\mathbb{R}_+$ | Set of positive real numbers |
| $\mathbb{R}_+^0$ | Set of non-negative real numbers (includes zero) |
| $\mathcal{S}_{\Delta_{x,y}}$ | Shift operator in $x$ and $y$ direction |
| $\mathcal{T}_t$ | Scale-space operator at scale $t$ |
| $\circledast$ | Continuous spatial convolution operator |
| $*$ | Discrete spatial convolution operator |
| $\pm$ | (Superscript) Indicates curvature maximum or minimum |
| $\lceil \cdot \rceil$ | Ceiling function (rounding up to next integer) |

## Mathematical Functions

| | |
|---|---|
| $D(x, y; \sigma)$ | Difference of Gaussians image (DoG) at scale $\sigma$ |
| $f(x)$ | 1D input signal |
| $f(x, y)$ | Image intensity at position $(x, y)$ |
| $f_\Sigma(x, y)$ | Integral image (sum of all pixel intensities from origin) |
| $g(x; t)$ | 1D Gaussian function at scale $t$ |
| $g(u; \sigma)$ | 1D Gaussian function at scale $\sigma$, defined over arc length $u$ |
| $g(x, y; t)$ | 2D Gaussian function at scale $t$ |
| $g(x, y; \sigma)$ | 2D Gaussian function at scale $\sigma$ |
| $\Gamma(u)$ | Parameterized contour, represented as $(x(u), y(u))$ |
| $\Gamma_\sigma(u; \sigma)$ | Smoothed parameterized contour at scale $\sigma$ |
| $\bar{g}(x, y; 1)$ | Prototype (reference) kernel at unit scale |
| $h(r; t)$ | Radially symmetric 1D function representing $g(x, y; t)$ |
| $H(\alpha)$ | Information content of turning angle $\alpha$ |
| $\mathbf{H}(x, y; \sigma)$ | Hessian matrix at scale $\sigma$ |
| $\kappa(u; \sigma)$ | Discrete curvature at arc length $u$ and scale $\sigma$ |
| $\kappa_{\sigma,i}(\sigma)$ | Discrete curvature of an extremum as a function of scale $\sigma$ |
| $L(x; t)$ | Scale-space representation of signal $f(x)$ at scale $t$ |
| $L(x, y; t)$ | Scale-space representation of signal $f(x, y)$ at scale $t$ |
| $L(x, y; \sigma)$ | Gaussian-filtered image at scale $\sigma$ |
| $L_{xx}$, $L_{yy}$, $L_{xy}$ | Second partial derivatives w.r.t. $x$, $y$, and $x$-$y$ |
| $\text{LoG}(x, y; \sigma)$ | Laplacian of Gaussian (LoG) image at scale $\sigma$ |
| $\varphi(t)$ | Scale transformation function; maps formal scale $t$ to scale $\sigma$ |
| $p(\alpha)$ | Distribution modeling expected turning angle $\alpha$ |
| $P(\alpha)$ | Probability of observing turning angle $\alpha$ |
| $R(x_c, y_c; \sigma)$ | Response of the LoG filter at position $(x_c, y_c)$ and scale $\sigma$ |
| $S_i^\pm$ | Signature function of a curvature extremum |
| $x_\sigma(u; \sigma), y_\sigma(u; \sigma)$ | Smoothed contour components at scale $\sigma$ |
| $\dot{x}_\sigma(u; \sigma), \dot{y}_\sigma(u; \sigma)$ | First derivatives w.r.t. arc length $u$ |
| $\ddot{x}_\sigma(u; \sigma), \ddot{y}_\sigma(u; \sigma)$ | Second derivatives w.r.t. arc length $u$ |
| $x_\Sigma(\xi)$ | Running sum of $x(u)$ up to position $\xi$ |

## Variables and Parameters

| | |
|---|---|
| $\alpha$ | Turning angle at a contour point |
| $\mathcal{A}_i$ | Set of pixels belonging to ambiguity $i$ |
| $C$ | Cost for connecting two edge segments |
| $d$ | Distance between two connection points of edge segments |
| $D$ | Distance between expected and candidate keypoint positions |
| $\Delta\theta$ | Angle difference between two edge segments |
| $\Delta x, \Delta y$ | Spatial shifts in $x$ and $y$ direction |
| $k$ | Scale multiplier in SIFT; number of steps in box filtering |
| $\kappa_{\sigma,i}$ | Curvature of a detected extremum at scale $\sigma$ |
| $M \times N$ | Input image dimensions (rows $\times$ columns) |
| $m \times n$ | Filter kernel dimensions (rows $\times$ columns) |
| $N_g$ | Number of samples to represent discrete Gaussians |
| $N_\Gamma$ | Number of contour points (length of the contour) |
| $p$ | Image point with coordinates $(x, y)$ |
| $r$ | Radial distance from origin, defined as $r = \sqrt{x^2 + y^2}$ |
| $r_i$ | Radius of the region around keypoint $i$ |
| $s$ | Number of DoG images per octave in SIFT |
| $\sigma$ | Scale parameter and standard deviation of the Gaussian |
| $\sigma_d$ | Standard deviation of Gaussian noise on contour |
| $\sigma_{\text{start}}, \sigma_{\text{step}}, \sigma_{\text{end}}$ | Start, step size, and end value of the scale parameter |
| $\sigma_i$ | Scale at which a curvature extremum has been detected |
| $\hat{\sigma}_i$ | Characteristic scale assigned to a curvature extremum |
| $\hat{\sigma}_{\text{match}}, \hat{\sigma}_{\text{src}}$ | Characteristic scales of the matched and source keypoints |
| $t$ | Formal scale parameter; time in diffusion equation ($t = \sigma^2$) |
| $u$ | Discrete arc-length parameter (contour coordinate) |
| $u_{\sigma,i}$ | Spatial position of a detected curvature extremum at scale $\sigma$ |
| $w_\theta$, $w_d$ | Weights for angle and distance in cost function |
| $x, y$ | Spatial coordinates |
| $x_c$, $y_c$ | Spatial coordinates of the image center |
| $\xi$, $\eta$ | Integration variables (shift variables in 2D convolution) |