TECHNISCHE UNIVERSITÄT WIEN
FAKULTÄT FÜR INFORMATIK

DISSERTATION

The Eccentricity Transform of n-Dimensional Shapes with and without Boundary

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

o. Univ. Prof. Dipl.-Ing. Dr. techn. **Walter G. Kropatsch**
183/2
Institut für rechnergestützte Automation
Arbeitsgruppe Mustererkennung und Bildverarbeitung

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Dipl.-Ing. **Adrian Ion**

Matrikelnummer 0527952
1060 Wien, Damböckgasse 10/6

Wien, am 23.03.2009

Adrian Ion

Acknowledgments

I can no other answer make but thanks, and thanks.
∼ William Shakespeare

I would like to thank my adviser, Prof. Walter G. Kropatsch, for welcoming me into his research group and for his continuous guidance. He has taught me how to work scientifically, always pushing me to improve myself and everything I do. Thank you for believing in me and giving me the opportunity to push my limits!

I would also like to thank Prof. Eric Andres for the nice collaboration on one of the papers that became part of this thesis, and for serving on my thesis committee. Thank you!

My colleagues, Yll Haxhimusa, Samuel Peltier, and Nicole M. Artner, I would like to thank for the nice collaboration, the hard work, and all the funny moments – at the beginning, the middle, and the end of my study. Yll, thank you for being like a big brother! Nicole, thank you for the German version of the abstract!

My parents, Michael and Iudit, have continuously emphasised the importance of education. If it wasn't for them I would have probably never considered this step. Together with my sister, Ileana, they have given me their unconditional love and support, and the freedom to pursue my dreams. Thank You!

My deepest gratitude goes to my wife Adriana for her endless support and patience, standing by me troughout the whole "journey", always finding a way to boost my self-confidence whenever I needed it. Thank you for everything!

Kurzfassung

Form (engl. *shape*) ist eine Eigenschaft von Objekten. Sie charakterisiert die räumliche Ausdehnung des Objektes und identifiziert die Punkte die Teil des Objektes sind. Form ist sowohl komplex als auch strukturiert und ermöglicht es Objekte zu identifizieren (erkennen). Die Form eines Objektes kann als binäres Bild repräsentiert werden, z.B. ein Bild in dem jedes Element (Pixel, Voxel) entweder Teil des Objektes ist oder nicht. Bildtransformationen extrahieren aus einem Eingabebild Informationen einer höheren Abstraktionsebene (z.B. die Anzahl der verbundenen Komponenten, das Skeleton einer Form, etc.).

Die Exzentrizitätstransformation (engl. *eccentricity transform*) assoziiert zu jedem Punkt einer Form die geodätische Distanz zu dem am weitesten entfernten Punkt. Das heißt sie assoziiert zu jedem Punkt die Länge des längsten, kürzesten Pfades der innerhalb der Form liegt und den berücksichtigten Punkt mit einem der anderen Punkte der Form verbindet. Die Exzentrizitätstransformation gehört zu einer Klasse von Bildtransformationen die jedem Punkt einer Form eine Funktion der Distanz zu den anderen Punkten der Form zuweisen.

Diese Arbeit präsentiert die Exzentrizitätstransformation, ihre Eigenschaften und ihre Berechnung, und zeigt zwei Anwendungsbeispiele aus dem Bereich Computer Vision. Die Definition der Transformation wird in einem metrischen Raum formuliert und vereint die Konzepte von Exzentrizität (engl. *eccentricity*) aus der Graphentheorie, vom weitest entfernten Punkt (engl. *furthest point*) in "Computational Geometry" und von der Ausbreitungsfunktion (engl. *propagation function*) in der mathematischen Morphologie. Die Exzentrizitätstransformation ist robust gegenüber Rauschen (z.B.: Salz- und Pfefferrauschen – zufällig fehlende Punkte in der Form, und kleine Segmentierungsfehler). Sie ist außerdem quasi-invariant gegenüber den Artikulationen einer Form. Die exzentrischen Punkte einer Form (Punkte die am weitesten entfernt zu einem anderen Punkt der Form sind) werden immer an ihrem Rand gefunden, wenn die Form planar ist und nicht mehr als zwei Löcher hat. Bei planaren, einfach verbundenen Formen liegen diese Punkte auf den konvexen Teilen des Randes. Das geodätische Zentrum (der Punkt mit der kleinsten Exzentrizität) einer planaren, einfach verbundenen Form ist ein einzelner Punkt. Bei Formen mit Löchern oder einer nicht planaren 2D Mannigfaltigkeit, kann das Zentrum eine nicht verbundene Menge sein.

Die Exzentrizitätstransformation wird an einem Satz von Grundformen mit steigender Komplexität studiert (z.B.: Linien, konvexe, planare Formen, nicht konvexe, einfach verbundene Formen, planare Formen mit einem Loch, etc.). Eigenschaften, wie die Position des geodätischen Zentrums, die exzentrischen Punkte und die Möglichkeiten eine Form zu zerlegen, um eine effizientere Berechnung zu ermöglichen, werden analysiert. Diese Eigenschaften motivierten die präsentierten Berechnungsalgorithmen und -aspekte. Methoden zur genauen Berechnung der Exzentrizitätstransformation und effiziente Approximationen werden vorgestellt und evaluiert.

Zusätzlich werden weitere Verbesserungen dieser Methoden und mögliche Parallelisierungen der Berechnungen diskutiert.

Zwei Anwendungsbeispiele für die Exzentrizitätstransformation werden vorgestellt. Ein Beispiel ist der Einsatz zum Beschreiben und Vergleichen von Formen. Dazu werden Histogramme der Exzentrizitätstransformation von 2D und 3D Formen erzeugt und als Deskriptoren verwendet. Die Anwendbarkeit der Exzentrizitätstransformation auf diesem Gebiet wird durch experimentelle Resultate und eine detaillierte Studie analysiert. Das zweite Anwendungsbeispiel ist die Verwendung von Exzentrizitätstransformation als Basis für ein formabhängiges Koordinatensystem für einfach verbundene, planare Formen. Der Zweck ist die Adressierung korrespondierender oder nahe beieinander liegender Punkte in unterschiedlich artikulierten Posen derselben Form.

Abstract

Shape is a property of an object. It characterizes an objects' spatial form and identifies the points that are part of the object. It is both complex and structured, and allows an object to be identified. A shape can be represented as a binary image i.e. an image where each picture element (pixel, voxel) is either part of the shape or not. Image transforms take as input an image and extract higher abstraction level information from it (e.g. count the number of connected components, compute the skeleton of a shape in the image, etc.).

The *eccentricity transform* associates to each point of a shape the geodesic distance to the point which is furthest away from it i.e. it associates to each point the length of the longest of the shortest paths that are contained in the shape and that connect the considered point with any other point of the shape. The eccentricity transform is part of a class of image transforms that associate to each point a function of the distance to other points of the shape.

This thesis presents the *eccentricity transform*, important properties, considers its computation, and gives two example applications in the context of computer vision. The definition unifies the concepts of *eccentricity* from graph theory, *furthest point* from computational geometry, and *propagation function* from mathematical morphology. The transform is robust with respect to noise (Salt and Pepper i.e. random missing points in the shape, and minor segmentation errors). It is quasi invariant with respect to articulation of the shape. For planar shapes with less than two holes, eccentric points (points which are furthest away for at least one other point of the shape) are always located on the boundary of the shape. For planar simply-connected shapes they lie on convex parts of the boundary. The geodesic center (points with minimum eccentricity) of a planar simply connected shape is a single point. For shapes with holes, or non planar 2D manifolds, it can be a disconnected set.

A study of the eccentricity transform of a set of basic shapes of increasing complexity (e.g. line, convex planar shapes, non-convex simply-connected shapes, planar shapes with a hole) is presented. Properties like the position of the geodesic center and eccentric points, as well as the possibility to decompose the shapes for a more efficient computing are studied. These properties have motivated the presented algorithms and aspects. Precise computation and efficient approximation methods are given and evaluated. Further improvement and parallelisation possibilities are discussed.

Two example applications of the eccentricity transform are presented. First, histograms of the eccentricity transform of 2D and 3D shapes are used as shape descriptors in a shape matching scenario. Experimental results and a detailed study are given. Following is the application of the eccentricity transform as a basis for a shape centered coordinate system for simply-connected planar shapes. The purpose is to address corresponding or nearby points in different articulated poses of the same shape.

Contents

Notation

Bellow is an alphabetically sorted list of the symbols with consistent meaning, used in this document. Formatting differs depending on the type: set (e.g. $\mathcal{S}, \mathcal{Q}$), function (e.g. f, d), multidimensional value (point, vector) (e.g. $\mathbf{p}, \mathbf{v}$), and scalar (e.g. r, s, h). The following labels are placed as helpers: s - set(s), f - function, p - point/vector. Some symbols have different meaning depending on the context, case in which all are mentioned.

label	symbol	description
s	$\mathcal{A}$	(discrete) arc
s	$\mathcal{B}$	part of a shape boundary, (discrete) arc
p	$\mathbf{c}$	center point, cut point
s	$\mathcal{C}$	(discrete) circle, cut
s	C	geodesic center of a shape
f	d	distance, superscript $d^{\mathcal{S}}$ means geodesic distance bounded to $\mathcal{S}$, subscript d_α means using α connectivity (α can be 4,6,8,26), and d_ε means Euclidean distance for pairwise visible points
f	D	distances computed by discrete circle propagation
f	$D^{\mathcal{S}}(\mathcal{Y})$	geodesic distance function, bounded to $\mathcal{S}$, with marker/source points $\mathcal{Y}$
f	DT	distance transform
s	$\mathcal{D}$	definition domain for an image
p	$\mathbf{e}$	edge of a graph, eccentric point
s	$\mathcal{E}$	edge set of a graph, ellipse halves
s	E	set of eccentric points of a shape
f	ECC	eccentricity transform/function
f	F	speed function of the contour for Fast Marching
f	f	function in general
s	$\mathcal{F}$	value domain of an image element
s	$\mathcal{G}$	graph
f	g	function in general
s	$\mathbf{h}$	histogram
s	$\mathcal{H}$	hyperplane
f	I	image
s	$\mathcal{K}$	set of "known" points/time values in Fast Marching
p	$\mathbf{l}$	a point
s	$\mathcal{L}$	separation line, a line

label	symbol	description
f	L	label of object/shape (shape matching)
f	λ	length of a path or line segment
p	$\mathbf{m}$	any point
s	$\mathcal{M}$	domain for geodesic distance, local maximas in $D^{\mathcal{S}}$
s	ν	geodesic
s	$\mathcal{N}_\alpha(\mathbf{p})$	the smallest neighborhood of $\mathbf{p}$ in $\mathbb{Z}^n$, ($\alpha \in \{4, 8, 6, 26\}$)
p	$\mathbf{o}$	circle center
p	$\mathbf{p}$	a point
s	π	path
s	$\mathcal{P}$	shape part, poly line, vertices to be processed (Dijkstra)
s	$\Pi(\mathbf{p}, \mathbf{q})$	set of paths between two points $\mathbf{p}, \mathbf{q}$ (see π and ν also)
p	$\mathbf{q}$	a point
s	$\mathcal{Q}$	set of shapes with the same label $\mathcal{Q}(l)$ (used for shape matching)
p	$\mathbf{r}$	arc center, reference point
f	$\boldsymbol{r}$	radial coordinate (coordinate mapping)
s	$\mathbb{R}$	set of real numbers
s	$\mathcal{R}$	rectangle
p	$\mathbf{s}$	end points of a discrete arc
s	$\mathcal{S}$	shape
s	$\partial\mathcal{S}$	boundary of shape $\mathcal{S}$
f	T	time function of the contour for Fast Marching
p	$\mathbf{u}$	a point, a vertex
s	$\mathcal{U}_\varepsilon(\mathbf{p})$	(open) ε-neighborhood of $\mathbf{p}$
p	$\mathbf{v}$	vertex of a graph, a point
s	$\mathcal{V}$	vertex set of a graph
f	w	weight of edges of a graph
s	$\mathcal{W}$	wave front, isoline
p	$\mathbf{x}$	maximal point for a given $\mathcal{S}$ and d i.e. point that is a local maximum
s	$\mathcal{X}$	set of "maximal" points i.e. local maximum in a geodesic distance function
p	$\mathbf{y}$	marker/source points for geodesic distance function
s	$\mathcal{Y}$	set of markers/source points for the geodesic distance function
s	$\mathbb{Z}$	set of integer numbers
	(x, y)	notation for point of coordinates x and y
	$\mathbf{pq}$	notation for line segment joining $\mathbf{p}$ and $\mathbf{q}$

1 Introduction

Like color, size, and weight, *shape* is a property of an object. It characterizes its spatial form, and identifies the points that are part of the object. "An object's shape is a *unique* perceptual property of the object in the sense that it is the *only* perceptual property that has sufficient complexity to allow an object to be identified" [Pizlo 08].

A major task in image analysis is to extract information at high levels of abstraction (e.g. skeleton, connected components) from the information at low levels of abstraction (pixels and color measurements), contained in an image. Image transforms have been widely used to move from the low abstraction levels of the input data to higher abstraction levels that form the output data. The purpose is to have a reduced amount of (significant) information at the higher abstraction levels.

One class of image transforms that is applied to shapes, associates to each point of the shape a value that characterizes in some way its relation to the rest of the shape. This value can be the distance to other points of the shape (e.g. landmarks). Examples of such transforms include:

- the **distance transform** [Rosenfeld 83] which associates to each point the length of the **shortest** path to the closest *boundary* point,
- the **Poisson equation** [Gorelick 04] which associates to each point the **average** time to reach the *boundary* by a random path (average length of the random paths from the point to the boundary),
- the **global geodesic function** [Aouada 08] which associates to each point the **mean** of the length of the shortest paths to *all points* of the shape, and
- the **eccentricity transform** [Kropatsch 06] which associates to each point the length of the **longest** of the shortest paths to *any other point* of the shape.

Using the transformed images one tries to come up with an abstracted representation, like the skeleton [Ogniewicz 95] or the shock graph [Siddiqi 99], build on the distance transform, which

can be used in e.g. shape classification or retrieval. Minimal path computation [Paragios 06] as well as the distance transform [Soille 02] are commonly used in 2D and 3D image segmentation.

The eccentricity transform (ECC) has its origins in the graph based eccentricity [Buckley 90, Diestel 97]. It has been defined in the context of digital images in [Kropatsch 06, Klette 04, Soille 02][1]. It was applied in the context of shape matching in [Ion 07b, Ion 08a], and used as a basis for a shape centered coordinate system for 2D planar shapes of articulated objects [Ion 08b]. The transform is robust with respect to noise (Salt and Pepper i.e. random missing points in the shape, and minor segmentation errors). Due to the fact that it is defined using geodesic distances, it is also quasi invariant with respect to articulation of the shape [Ling 07].

The eccentricity transform is defined for continuous objects of any dimension (e.g. 3D ellipsoid or the 2D surface of the 3D ellipsoid) and for discrete objects of any dimension (e.g. 2D binary shape, or the 3D mesh of the surface of an ellipsoid), represented using regular grids or graphs. It does not require a priori knowledge of the boundary of the object and is defined also for objects without a boundary (e.g. the 2D surface of an ellipsoid).

This thesis focuses on the eccentricity transform. It gives a common definition for both continuous and discrete shapes, defines derived notions and important properties, considers computation and efficient approximation algorithms, and presents its application in the context of computer vision using two examples: shape matching and mapping a coordinate system to an articulated shape.

The following sections give a summary of the original contributions, and present the organisation of the thesis – a brief overview of each chapter is given.

1.1 Summary of Original Contributions

The work presented in this thesis constitutes novel results, to which the author has significantly contributed. The most important original contributions by the author are:

(1.) the formulation of the eccentricity transform over a metric space, unifying the concepts of *eccentricity* from graph theory, *furthest point* from computational geometry, and *propagation function* from mathematical morphology (Chapter 3);

(2.) the properties regarding the existence of eccentric points only on the shape boundary or also inside the shape (Section 3.4.1);

(3.) the existence of a unique geodesic center for simply connected planar 2D shapes (Section 3.4.3);

[1] In [Soille 02] it appears under the name *propagation function*.

(4.) properties of the eccentricity transform in the presence of holes in the shape (non-simply connected shapes) (Chapter 3);

(5.) the eccentricity transform of the majority of the simple shapes in Section 3.5;

(6.) the robustness and articulation experiments (Section 3.6);

(7.) the algorithms computing the eccentricity transform and the improvements (sequential and divide et impera) in Chapter 4, except algorithm ECC06 (Section 4.2.1), and the decomposition of a tree along a junction (Section 4.5.2);

(8.) the presented applications: shape matching, and mapping a coordinate system to an articulated 2D shape (Chapter 5).

Parts of this thesis have been previously published in [Kropatsch 06, Ion 07a, Ion 07b, Kropatsch 07, Ion 08a, Ion 08b, Ion 08c, Ion 08d].

1.2 Thesis Organization

The thesis is organized as follows:

Chapter 1 introduces the main problem area that the thesis addresses, summarizes the main contributions and gives a detailed chapter plan.

Chapter 2 recalls the main notions and properties required in the following chapters (e.g. object, shape, metric, neighborhood, path, geodesic distance, graph, image, and geodesic convexity). It also defines the geodesic distance function and considers its properties and computation. The distance transform, a widely used "special case" of the geodesic distance function, is also recalled.

Chapter 3 defines the eccentricity transform, introduces terms related to it (e.g. eccentric point, eccentric path), and presents properties related to the eccentricity transform and the related terms. It gives a detailed study for variations of ten basic shapes, progressively emphasizing different properties. The chapter concludes with comparative experiments regarding the robustness of the eccentricity transform, with respect to Salt and Pepper noise, minor segmentation errors, and articulation.

Chapter 4 presents algorithms for computation and efficient approximation of the eccentricity transform of discrete shapes. The "in action" behavior of these algorithms is shown using experimental results. Possible improvements and ways to approach them are discussed in detail.

Chapter 5 presents two example applications: 2D and 3D shape matching, and mapping a shape centered coordinate system to the 2D shape of an articulated object. Both applications are presented to further motivate the relevance of the eccentricity transform in the context of computer vision. The research related to the former one (shape matching), is more mature and thus the application and results are presented more thoroughly. In the case of the latter one (coordinate system) mainly proof of concept steps have been made and much more open questions exist.

Chapter 6 concludes the thesis and lists important open questions.

2

Shapes and Distances

This chapter introduces the notions of *object*, *shape* and *distance*, which are required to define the eccentricity transform. Used shape representations, properties, and distance functions are given. The concepts of metric and metric space are recalled, with special attention given to the geodesic distance and its computation. Other relevant notions like the distance transform and geodesic convexity are recalled at the end of the chapter.

2.1 Recall of Basic Notions and Properties

The used notation and definitions follow [Klette 04] and [Soille 02]. This section defines the concepts of object, shape, metric, path, geodesic distance, graph, graph pyramid, and image.

2.1.1 Object and Shape

Definition 1. (Object) *An object is a collection of atomic parts taken to be one. It refers usually to something material that may be perceived by the senses, something toward which thought, feeling, or action is directed [Mer 03].*

Objects can have properties like color, size, weight, *shape*, etc.

"An object's shape is a *unique* perceptual property of the object in the sense that it is the *only* perceptual property that has sufficient complexity to allow an object to be identified" [Pizlo 08].

As noted by [Newman 00] there are not many mathematical definitions of the term *shape*. Socrates defined shape as "the limit of a solid" [Day 94]. The English dictionary [Mer 03] defines shape as the "spatial form" of an object. We consider the following definition:

Definition 2. (Shape) *The Shape of an object is the (connected) part of space occupied by the object, as determined by its boundary.*

Defining an object/shape usually includes the definition domain of the atomic parts, e.g. points in $\mathbb{R}^2$, vertices of a graph, etc. and a *neighborhood* or *adjacency relation* specifying how the parts are put together.

Objects and shapes are *continuous* if the definition domain of the parts is a nonempty subset of a continuous domain (e.g. a disk in $\mathbb{R}^2$, a sphere in $\mathbb{R}^3$, the points on the unit circle in $\mathbb{R}^2$) and *discrete* if the definition domain is a discrete one (e.g. objects made out of points in $\mathbb{Z}^2$, or the vertices of a graph).

This thesis deals with continuous objects and shapes in $\mathbb{R}^n$, and discrete objects and shapes represented either as points in $\mathbb{Z}^n$ or as graphs (Section 2.1.7).

In [Ling 07] a model of *articulated objects* is presented. It is defined as a union of (rigid) parts $\mathcal{O}_i$ and joints (named "junctions" by the authors). An articulation is defined as a transformation that is rigid when limited to any part $\mathcal{O}_i$, but can be non-rigid on the junctions. An articulated instance of an object is an articulated object itself (actually the same object) that can be articulated back to the original one. The term *articulated shape* refers to the shape of an articulated object in a certain pose.

Definition 3. (Shape Boundary) *For a shape $\mathcal{S}$, continuous or discrete, $\partial\mathcal{S}$ denotes its boundary (border). The boundary is made out of the elements of $\mathcal{S}$ separating the inside of $\mathcal{S}$ (elements of $\mathcal{S}$) from the background (elements of the space, not part of $\mathcal{S}$).*

The boundary $\partial\mathcal{S}$ can be explicitly given (e.g. for graphs), or implicit (e.g. for objects in $\mathbb{Z}^n$ the boundary pixels/voxels can be determined using the associated neighborhood (Section 2.1.3)).

Shapes are usually associated with "real life objects". Thus, when doing matching or classification, differences resulting from changes in pose (e.g. rotation, translation, scaling, part articulation) are ignored. Nevertheless, when computing features (e.g. image transforms) the given pose is usually considered for computation, e.g. no normalization is required for computing the distance transform (Section 2.2.3).

2.1.2 Metric and Metric Space

Definition 4. (Metric) *Let $\mathcal{S}$ be an arbitrary nonempty set. A function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{G}$, $\mathbb{G} \subseteq \mathbb{R}$, is a metric on $\mathcal{S}$* **iff** *it has the following properties:*

(1.) for all $\mathbf{p}, \mathbf{q} \in \mathcal{S}$, we have $d(\mathbf{p}, \mathbf{q}) \geqslant 0$, and $d(\mathbf{p}, \mathbf{q}) = 0$ **iff** *$\mathbf{p} = \mathbf{q}$ (positive definiteness),*

(2.) for all $\mathbf{p}, \mathbf{q} \in \mathcal{S}$, we have $d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p})$ (symmetry),

(3.) for all $\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathcal{S}$, we have $d(\mathbf{p}, \mathbf{r}) \leqslant d(\mathbf{p}, \mathbf{q}) + d(\mathbf{q}, \mathbf{r})$ (the triangle inequality).

If d is a metric on $\mathcal{S}$, the pair $[\mathcal{S}, d]$ defines a *metric space*. If $\mathbb{G} \subseteq \mathbb{Z}$ the metric is said to be *integer-valued*, otherwise it is called *real-valued*.

The most well known real-valued metric is the *Euclidean metric*. Let $\mathbf{p} = (x_1, x_2, ..., x_n)$ and $\mathbf{q} = (y_1, y_2, ..., y_n)$, $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$, the Euclidean metric is defined as:

$$d_\varepsilon(\mathbf{p}, \mathbf{q}) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \tag{2.1}$$

The *n-dimensional Euclidean space* $\mathcal{E}^n = [\mathbb{R}^n, d_\varepsilon]$ is defined on a Cartesian coordinate system on $\mathbb{R}^n$ and using the *Euclidean metric* d_ε.

Integer-valued metrics for $\mathbb{Z}^2$ include the *city-block metric* (also called Minkowski metric, L_1), defined for $\mathbf{p}, \mathbf{q} \in \mathbb{Z}^2$, $\mathbf{p} = (x_1, y_1), \mathbf{q} = (x_2, y_2)$ as:

$$d_4(\mathbf{p}, \mathbf{q}) = |x_1 - x_2| + |y_1 - y_2|$$

and the *chessboard metric*:

$$d_8(\mathbf{p}, \mathbf{q}) = \max\{|x_1 - x_2|, |y_1 - y_2|\}.$$

Let $\mathbf{p}, \mathbf{q} \in \mathbb{Z}^3$, $\mathbf{p} = (x_1, y_1, z_1)$ and $\mathbf{q} = (x_2, y_2, z_2)$. Well known integer-valued metrics in $\mathbb{Z}^3$ are:

$$d_6(\mathbf{p}, \mathbf{q}) = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

and

$$d_{26}(\mathbf{p}, \mathbf{q}) = \max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|\}.$$

Example metric spaces defined using the integer-valued metrics above include: $[\mathbb{Z}^2, d_4]$, $[\mathbb{Z}^2, d_8]$, $[\mathbb{Z}^3, d_6]$, and $[\mathbb{Z}^2, d_{26}]$. Note that the subscript of the (distance) function d identifies the type of neighborhood corresponding to the metric (Section 2.1.3).

2.1.3 Neighborhood

Definition 5. *For any metric space $[\mathcal{S}, d]$, any $\mathbf{p} \in \mathcal{S}$, and any $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, the (open) ε-neighborhood of $\mathbf{p}$ in $\mathcal{S}$ is defined as:*

$$\mathcal{U}_\varepsilon(\mathbf{p}) = \{\mathbf{q} \mid d(\mathbf{p}, \mathbf{q}) < \varepsilon\}$$

For any discrete point $\mathbf{p}$, the smallest neighborhood of $\mathbf{p}$ in $[\mathbb{Z}^2, d_\alpha]$ ($\alpha \in \{4, 8\}$) or in $[\mathbb{Z}^3, d_\alpha]$ ($\alpha \in \{6, 26\}$) is:

$$\mathcal{N}_\alpha(\mathbf{p}) = \{\mathbf{q} \mid d(\mathbf{p}, \mathbf{q}) \leqslant 1\}$$

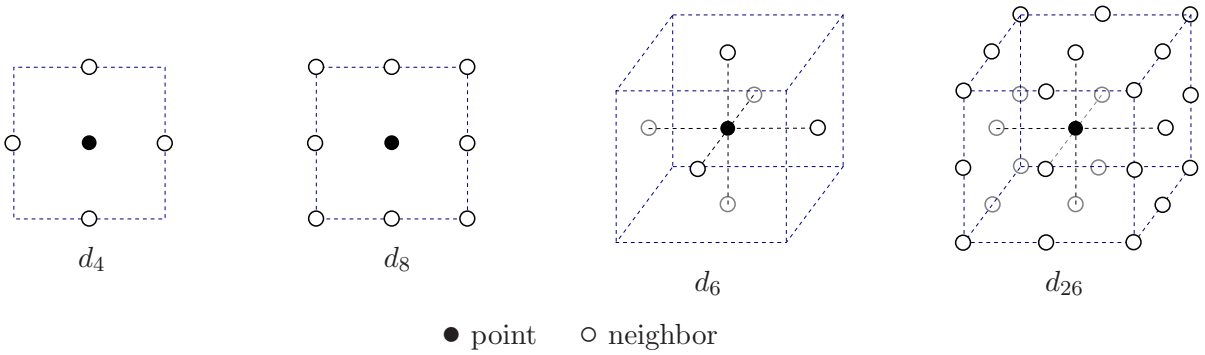


Figure 2.1: The smallest neighborhood for integer-valued metrics in $\mathbb{Z}^2$: d_4, d_8, and $\mathbb{Z}^3$: d_6, d_{26}.

Note that $\mathcal{N}_\alpha(\mathbf{p}) - \{p\}$ has cardinality α. If $\mathbf{q} \in \mathcal{N}_\alpha(\mathbf{p})$, $\mathbf{q}$ is said to be α-*adjacent* to $\mathbf{p}$.

2.1.4 Path

Continuous domains $\mathcal{S}$: a *continuous path*[1] π between two points $\mathbf{p}, \mathbf{q} \in \mathcal{S}$ is a continuous mapping from the interval $[0, 1]$ to $\mathcal{S}$ and will be denoted by $\pi(\mathbf{p}, \mathbf{q})$.

The *length* $\lambda(\pi)$ of a continuous path π is:

$$\lambda(\pi) = \int_0^1 |\dot{\pi}(t)| dt$$

where $\pi(t)$ is a parametrization of the continuous path from $\mathbf{p} = \pi(0)$ to $\mathbf{q} = \pi(1)$, and $\dot{\pi}(t)$ is the differential of the arc length.

Discrete domains $\mathbb{Z}^n$, $n \in \{2, 3\}$: the concept of (discrete) path is defined using the concept of smallest neighborhood $\mathcal{N}_\alpha$ as follows: an α-*connected discrete path* (α-path) between points $\mathbf{p}, \mathbf{q} \in \mathbb{Z}^n$ is denoted by $\pi(\mathbf{p}, \mathbf{q})$ and is defined as the sequence $\pi(\mathbf{p}, \mathbf{q}) = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_l)$ with $\mathbf{u}_k \in \mathbb{Z}^n$, $0 \leqslant k \leqslant l$ s.t. $\mathbf{u}_0 = \mathbf{p}$, $\mathbf{u}_l = \mathbf{q}$, and $\mathbf{u}_{i+1}$ is α-adjacent to $\mathbf{u}_i$, $0 \leqslant i < l$. $\alpha \in \{2, 4\}$ for $n = 2$ and $\alpha \in \{6, 26\}$ for $n = 3$.

The length $\lambda(\pi)$ of a discrete path $\pi = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_l)$ is:

$$\lambda(\pi) = \sum_{i=0}^{i<l} d_\alpha(\mathbf{u}_i, \mathbf{u}_{i+1})$$

The labels *continuous-* and *discrete-* are used to emphasize the nature of the definition domain, and will be replaced with the shorter *path* when the nature of the domain is implicit

[1] When considering continuous domains, the term *curve* is probably more spread. We use *path* to obtain a uniform naming in both continuous and discrete domains.

or does not play an important role. In both discrete and continuous domains, we will denote by $\Pi(\mathbf{p}, \mathbf{q})$ the set of all paths connecting $\mathbf{p}$ and $\mathbf{q}$.

2.1.5 Connectivity

The following definition is used for continuous domains.

Definition 6. (Connected) *A set $\mathcal{S}$ is said to be (path) connected if $\forall \mathbf{p}, \mathbf{q} \in \mathcal{S}$ there exists a path $\pi(\mathbf{p}, \mathbf{q})$.*

The following definition is used for discrete domains $\mathbb{Z}^n$.

Definition 7. (α-Connected) *A discrete set $\mathcal{S} \in \mathbb{Z}^n$ (in our case a shape) is said to be alpha−connected if $\forall \mathbf{p}, \mathbf{q} \in \mathcal{S}$ there exists an α-connected path $\pi(\mathbf{p}, \mathbf{q})$ in $\mathcal{S}$.*

2.1.6 Geodesic distance

The term "geodesic" comes from *geodesy*, the science of measuring the size and shape of the Earth; in the original sense, a geodesic was the shortest route between two points on the Earth's surface, namely, a segment of a great circle.

In mathematics the term geodesic (path/curve/line) has many meanings and can depend on the particular structure of the metric space [Hazewinkel 89, Bronshtein 97]. In the geometry of spaces where the metric is specified in advance, it is defined as the (locally) shortest path between points in a space. In spaces with an affine connection[2], it is defined as the curve for which the tangent vector field is parallel along this curve. In Riemannian geometries, geodesics are defined as extremals of the (path-) length function. In this document we use the latter one, which is the more common in the fields of discrete and computational geometry [Goodman 04].

Let $\mathcal{M} \supseteq \mathcal{S}$ be a set with a given metric, and let $\Pi^{\mathcal{M}}(\mathbf{p}, \mathbf{q})$ denote the set of all paths connecting two points $\mathbf{p}, \mathbf{q} \in \mathcal{M}$, that are contained in $\mathcal{M}$ i.e. $\mathbf{u} \in \pi \in \Pi^{\mathcal{M}}(\mathbf{p}, \mathbf{q}) \implies \mathbf{u} \in \mathcal{M}$. The set $\mathcal{M}$ is referred to as the *geodesic mask* and will be in most cases equal to $\mathcal{S}$, the shape itself.

Definition 8. (Geodesic path) *A geodesic path, or simply geodesic, between two points $\mathbf{p}, \mathbf{q} \in \mathcal{S} \subseteq \mathcal{M}$ is the shortest path (with minimal length) connecting $\mathbf{p}$ and $\mathbf{q}$ in $\mathcal{M}$. It will be denoted by $\nu(\mathbf{p}, \mathbf{q})$ or simply ν. More formally, $\nu \in \Pi^{\mathcal{M}}(\mathbf{p}, \mathbf{q})$ is a geodesic in $\mathcal{M}$* **iff** *its length $\lambda(\nu)$ is minimal.*

The notion above is identical to the concept of "inner distance" in [Ling 07].

[2] In differential geometry, an affine connection is a geometrical object on a smooth manifold which connects nearby tangent spaces.

Property 1. (Many geodesics) *More than one path linking* $\mathbf{p}$ *and* $\mathbf{q}$ *can have the minimal length i.e. there can be more than one geodesic between two points*[3].

Definition 9. (Geodesic distance) *The geodesic distance* $d^{\mathcal{M}}(\mathbf{p}, \mathbf{q})$ *between two points* $\mathbf{p}, \mathbf{q} \in \mathcal{S} \subseteq \mathcal{M}$ *is defined as the length of the geodesic(s) from* $\mathbf{p}$ *to* $\mathbf{q}$*, in* $\mathcal{M}$*.*

Property 2. (Geodesic distance = metric) *The function* $d^{\mathcal{M}} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ *is a metric [Soille 02].*

Property 3. (Geodesic distances – articulation) *Geodesic distances are bounded under articulation of the shape. Proof due to [Ling 07].*

The proof of the property above shows that the variation of the length of a geodesic path going over a joint is smaller than the width of the joint. Assuming joints have small width compared to the rigid parts, makes the length of most geodesic paths stable (constant if the joints have close to zero width).

Euclidean distance vs. Euclidean based geodesic distance: to differentiate between the two options we will use:

- *Euclidean distance* or ℓ^2*-norm*: to denote distances computed in the space in which $\mathcal{S}$ is embedded (ℓ^2*-norm*, Equation 2.1);
- and the term *Euclidean based geodesic distance*: to denote geodesic distances computed in $\mathcal{S}$.

2.1.7 Graph

A *graph* is the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the elements of $\mathcal{V} \neq \emptyset$ are called *vertices* and the elements of $\mathcal{E}$ are called *edges* [Diestel 97, Thulasiraman 92]. Each edge $\mathbf{e} = (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}$ is said to be incident to two vertices $\mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}$. If $\mathbf{v}_i = \mathbf{v}_j$ than $\mathbf{e}$ is said to be a *self-loop*. Vertices and edges can be attributed (or weighted). The edges can be weighted using a function $w : \mathcal{E} \rightarrow \mathcal{D}$ with values in some domain $\mathcal{D}$. The weights can be used to encode a certain kind of distance or difference between the elements represented by the vertices ($\mathcal{D} = \mathbb{R}$). Graphs can be visualized as points (one for each vertex) connected by lines (one for each edge).

Neighborhood: in the case of graphs, the neighborhood of a vertex is explicitly given by the existing edges incident to it. More formal, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the neighborhood $\mathcal{N}_{\mathcal{G}}(\mathbf{v}_i)$ of a vertex $\mathbf{v}_i \in \mathcal{V}$ is:

$$\mathcal{N}_{\mathcal{G}}(\mathbf{v}_i) = \{\mathbf{v}_j \in \mathcal{V} \mid \exists \mathbf{e} = (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}\}$$

[3]E.g. two opposing points on a sphere can be connected by many shortest paths, all having the length of half the circumference.

Path: a *path* in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of vertices $\pi = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n)$ such that for all $0 \leqslant i < n$, $\mathbf{v}_{i+1} \in \mathcal{N}_{\mathcal{G}}(\mathbf{v}_i)$. The vertices $\mathbf{v}_0$ and $\mathbf{v}_n$ are called the *start vertex*, respectively the *end vertex* of the path. If $\mathbf{v}_0 = \mathbf{v}_n$ the path is called a *cycle*.

For the case of the edges weighted with their length, the *length* $\lambda(\pi)$ of a path of a weighted graph $\mathcal{G}$, is the sum of the weights of the edges traversed i.e.

$$\lambda(\pi) = \sum_{i=0}^{n-1} w(\mathbf{e}_i)$$

where $\mathbf{e}_i = (\mathbf{v}_i, \mathbf{v}_{i+1})$ are the edges of the path $\pi = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n)$. One could also consider that all edges have a fixed length l, case in which $\lambda(\pi) = nl$.

Connectivity: the concept of connectivity in graph theory is defined as follows: a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be *connected* if $\forall \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}$ there exists a path $\pi = (\mathbf{v}_i, \ldots, \mathbf{v}_j)$ in $\mathcal{G}$.

Distance: If the lengths of all edges are greater than zero i.e. $\forall \mathbf{e} \in \mathcal{E} \implies w(\mathbf{e}) > 0$, the *distance* $d_{\mathcal{G}}(\mathbf{v}_i, \mathbf{v}_j)$ between two vertices is defined as the length of the shortest path connecting them:

$$d_{\mathcal{G}}(\mathbf{v}_i, \mathbf{v}_j) = \min\{\lambda(\pi) \mid \pi = (\mathbf{v}_i, \ldots, \mathbf{v}_j)\}$$

The distance $d_{\mathcal{G}}(\mathbf{v}_i, \mathbf{v}_j)$ is also known as the *graph geodesic distance* [Bouttier 03].

Property 4. *Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with edges weighted with their lengths, $w(e) > 0, \forall \mathbf{e} \in \mathcal{E}$, the vertex set and the distance function form a metric space $[\mathcal{V}, d_{\mathcal{G}}]$.*

2.1.8 Image

The dictionary definition of an *image*, is as follows [Mer 03]: "an image is the optical counterpart of an object, produced by an optical device (as a lens or mirror) or an electronic device; a visual representation of something as: a likeness of an object produced on a photographic material, or a picture produced on an electronic display (as a television or computer screen)".

A *digital image*, or digital representation of an image, is a function $I : \mathcal{D} \to \mathcal{F}$ mapping the discrete domain $\mathcal{D} \subset \mathbb{Z}^n$ to the set of values $\mathcal{F}$. For $n = 2$, I is a two dimensional image and the elements of $\mathcal{D}$ are called *pixels*, for $n = 3$, I is a three dimensional image and the elements of $\mathcal{D}$ are called *voxels*. The values in $\mathcal{F}$ can be for example gray values or colors, and depending on $\mathcal{F}$ images can be *binary*, $|\mathcal{F}| = 2$ (usually $\mathcal{F} = \{0, 1\}$), *grayscale* if $\mathcal{F}$ is a one dimensional domain representing the gray value of the pixel/voxel, *color* if $\mathcal{F}$ is a representation of the color of a pixel/voxel (e.g. RGB, HSV value, or index in a color palette, etc.). Digital images are usually

the result of an imaging process, which maps the continuous real world domain to a discrete one.

This thesis considers 2D and 3D digital images represented by attributed points on the regular grid $\mathbb{Z}^2$ respectively $\mathbb{Z}^3$. 2D images have the shape of a rectangle and 3D images the shape of a cuboid[4]. They can be represented as matrices or as graphs.

Identifying the image elements that belong to a certain object can be implicit (e.g. in a binary image, 1 usually means object and 0 background) or explicit (e.g. in addition to the gray value/color information, to each element an additional label is associated and all elements with the same label belong to the same object). An (automatic) mean to try to estimate image elements that belong to the same object, given a digital image, is segmentation (e.g. [Haxhimusa 03, Shi 00]).

Representing images as graphs: *Neighborhood graphs* represent images by associating a vertex to each pixel/voxel, and connecting them through an edge if the respective image elements are neighbors (e.g. they share a common boundary). If the image elements are grouped into regions, one can associate a vertex to each region and connect two vertices if the respective regions are neighbors. Such a graph is called a *region adjacency graph* (RAG).

If the image is *well composed* [Latecki 97] (see Jordan curve Theorem[5] in [Klette 04]) and the embedding plays an important role then inclusion relations can be additionally encoded by using a pair of *dual graphs* [Kropatsch 95]. One of the two graphs is an *extended region adjacency graph* that encodes 2D adjacency and inclusion relations and the other one, sometimes called *boundary graph*, encodes the region boundaries (there is a one-to-one association between the edges of the two graphs). To handle dimensions higher than 2, other graph based representations like *combinatorial maps* [Brun 06, Damiand 05] or *generalized maps* [Lienhardt 94] can also be used.

Embeddable graph based representations are usually used in conjunction with the 4-adjacency in 2D and the 6-adjacency in 3D. One can also use a higher adjacency, which would be locally adapted to ensure the validity of the Jordan curve theorem [Klette 04].

Using a planar graph is not always possible. For example, consider a set of more than five 2D points with no three points located on a straight line. To encode the Euclidean distance between the points (represented as vertices), by weighting edges with their length and using the distance function in Section 2.1.7, one would have to use a non planar, fully connected graph.

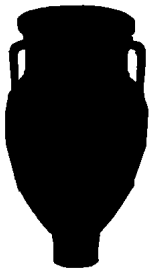
[4]The used models for 2D and 3D digital images are the most spread ones, but still only a special case of the high variety of possible representations for digital images.

[5]The Jordan curve theorem states that every non-self-intersecting closed curve in the plane divides the plane into two disconnected regions: an "inside" and an "outside". In other words, every path that connects one point from the inside with one from the outside has to intersect the close curve.

image showing a leopard

2D binary shape of the leopard in the image to the left

image showing a recipient



2D binary shape of the recipient in the image to the left

Figure 2.2: Example images, objects and 2D shapes. (Images from [Martin 01])

2.1.9 Shapes in this thesis

The methods presented in this thesis use as one of their input arguments a binary shape (defined in Section 2.1.1). To represent binary shapes we use binary images and associate one of the two values of each element with the shape and the other one with the background.

Table 2.1 gives an overview of the shape representations and associated metrics, considered in this thesis. If not otherwise stated, the theoretical part in Chapter 3 refers to continuous shapes in $\mathbb{R}^n$ using d_ε. Image 2.2 shows example images (containing objects), and the binary shape of the object in each image. The columns of Table 2.1 give:

- a short description;
- the notation used to identify the type of representation;
- the domain of the image elements that are used to represent the shape;
- the neighborhood;
- the metric;
- how the shape is actually represented.

2.1.10 Graph pyramids

This section gives a brief recall of (irregular) graph pyramids [Meer 89, Jolion 94]. A graph pyramid is a hierarchical representation for image partitions at multiple levels of detail. As opposed to their regular counterparts, graph pyramids are shift invariant and their structure

Table 2.1: Shape representations that appear in this document.

description	id.	domain	nb.	metric	represented as
a continuous 2D shape with holes, with polygonal lines as boundaries	$\mathcal{P}(\mathbb{R}^2), d_\varepsilon$	$\mathcal{S} \subset \mathbb{R}^2$	$\mathcal{U}_\varepsilon$	d_ε	formula, or corner points for boundary polygons (including holes)
4-connected 2D shape using Euclidean distance	$\mathbb{Z}^2, d_\varepsilon$	$\mathcal{S} \subset \mathbb{Z}^2$	4	d_ε in support squares for $\mathcal{S}$	binary 2D image
4-connected 2D shape using city-block distance	$\mathbb{Z}^2, d_4$	$\mathcal{S} \subset \mathbb{Z}^2$	4	d_4	binary 2D image
8-connected 2D shape using chess-board distance	$\mathbb{Z}^2, d_8$	$\mathcal{S} \subset \mathbb{Z}^2$	8	d_8	binary 2D image
6-connected 3D shape using Euclidean distance	$\mathbb{Z}^3, d_\varepsilon$	$\mathcal{S} \subset \mathbb{Z}^3$	6	d_ε in support cubes for $\mathcal{S}$	binary 3D image
triangle mesh of surface of 3D shape, using Euclidean distance on the surface	$\mathbb{Z}^2/\mathbb{Z}^3, d_\varepsilon$	$\mathcal{S} \subset \mathbb{Z}^2$	x	d_ε in triangles of $\mathcal{S}$	triangle mesh in 3D
image represented by a graph with explicit distance encoding	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	$\mathcal{V}$	$\mathcal{E}$	$\mathcal{E}$	graph

All distances are **geodesic distances** i.e. d_ε, d_e, d_8 is actually $d_\varepsilon^{\mathcal{S}}, d_4^{\mathcal{S}}, d_8^{\mathcal{S}}$. The given (non geodesic) metric is used only if two points are directly visible.

adapts to the input data. A graph pyramid will be employed in Section 5.2 to represent the decomposition of a shape based on its eccentricity transform.

A *graph pyramid* $P = \{G_0, \ldots, G_t\}$ is a stack of successively reduced graphs. Each level $G_k = (V_k, E_k)$, $1 \leqslant k \leqslant t$, is obtained by *contracting* and *removing* edges in the level G_{k-1} below. Successive levels reduce the size of the data by $\gamma > 1$. The edges and vertices of G_k can be attributed. The *reduction window* relates a vertex at a level G_k with a set of vertices in the level directly below (G_{k-1}). Higher level descriptions are related to the original input data. The *receptive field* (RF) of a given vertex $v \in G_k$ aggregates all vertices in G_0 of which v is an ancestor.

Each level represents a partition of the base level into connected subgraphs i.e. *connected subsets of pixels*, if the pyramid is build in the context of an image. The construction of an irregular pyramid is iteratively local [Meer 89]. In the base level G_0 of an irregular pyramid the vertices represent single pixels and the neighborhood of the cells is defined by the 4-connectivity of the pixels (higher connectivity can be used locally, but planarity should be kept). The union of neighboring vertices on level $k-1$ (children) to a vertex on level k (parent), is controlled by trees called contraction kernels (CK) [Kropatsch 95] chosen by the algorithm (e.g. segmentation,

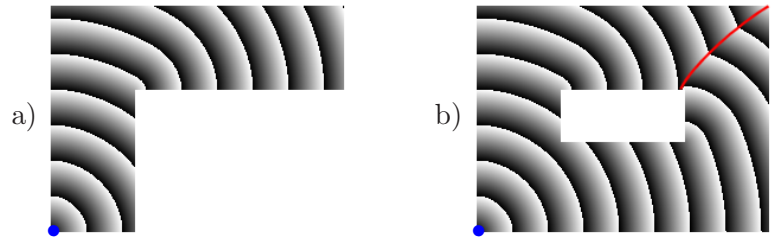


Figure 2.3: Geodesic distance functions. Gray values are distance values modulo a constant. The marker set $\mathcal{Y}$ (source point) is the bottom-left corner. b) the separation line is shown (red).

connected component labeling, etc.). Every vertex computes its values independently of other vertices on the same level. Thus local independent (and parallel) processes propagate information up and down and laterally in the pyramid [Kropatsch 05].

In [Haxhimusa 02, Kropatsch 05], methods for optimally building irregular pyramids are presented. Methods like MIS and MIES ensure logarithmic height of the pyramid by choosing efficient contraction kernels i.e. contraction kernels achieving high reduction factors.

2.2 The Geodesic Distance Function

Definition 10. (Geodesic distance function) *The geodesic distance function* $D^{\mathcal{S}} : \{\Re(\mathcal{S}) \setminus \{\emptyset\}\} \times \mathcal{S} \rightarrow \mathbb{R}$, *where* $\Re(\mathcal{S})$ *is the powerset of* $\mathcal{S}$, *is calculated from a set of points* $\mathcal{Y} \subseteq \mathcal{S}$ *and assigns* $\min\{d^{\mathcal{S}}(\mathbf{p}, \mathbf{y}) \mid \mathbf{y} \in \mathcal{Y}\}$ *to all points* $\mathbf{p} \in \mathcal{S}$ *i.e.:*

$$D^{\mathcal{S}}(\mathcal{Y}, \mathbf{p}) = \min\{d^{\mathcal{S}}(\mathbf{p}, \mathbf{y}) \mid \mathbf{y} \in \mathcal{Y}\}.$$

The set $\mathcal{Y}$ is also called *marker set* [Soille 02], *source points*, or *starting points*. If $\mathcal{Y}$ is considered fixed, the function can be formulated as $D^{\mathcal{S}}(\mathcal{Y}) : \mathcal{S} \rightarrow \mathbb{R}$. This notation will be used to refer to the geodesic distance function in general, or to the function $[D^{\mathcal{S}}(\mathcal{Y})](\mathbf{p})$ which has the point $\mathbf{p}$ as the single argument and is obtained after fixing $\mathcal{Y}$. As in the case of the geodesic distance $d^{\mathcal{S}}$ (Definition 9), the subscript specifying the neighborhood and the superscript specifying the domain, can be left out if it is clear from the context.

If the set $\mathcal{Y}$ is a single point, like in our case, the function is also known as the *shape-bounded single source distance transform* (***SBDT***) [Ion 08d]. The term *Euclidean based geodesic distance function* and symbol D_ε will be used to denote the geodesic distance function defined using d_ε (Euclidean based geodesic distance, Section 2.1.6). Figure 2.3 shows examples of geodesic distance functions.

Definition 11. (Separation set) *A separation set of the geodesic distance function* $D^{\mathcal{S}}(\mathbf{y})$ *is the*

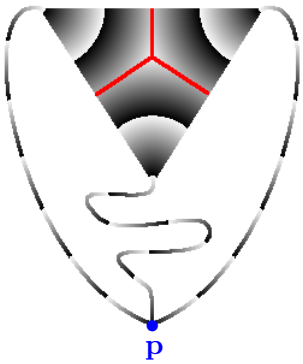


Figure 2.4: Shape and point **p**, with complex separation lines in $D_\varepsilon^{\mathcal{S}}(\mathbf{p})$. (Gray values are distances to **p** modulo a constant.)

set formed by points $\mathbf{p} \in \mathcal{S}$ *s.t.* $\exists \pi_1, \pi_2 \in \Pi^{\mathcal{S}}(\mathbf{p}, \mathbf{y})$, $\pi_1 \neq \pi_2$, *with* $\lambda(\pi_1) = \lambda(\pi_2) = d^{\mathcal{S}}(\mathbf{p}, \mathbf{y})$.

Separation sets are piecewise connected. For 2D shapes, the connected components of the separation set are 1D manifolds, and are called *separation lines*. In the case of 3D shapes, they are called *separation surfaces*. Separation sets appear when the shape has holes of any dimension i.e. cavities, tunnels, etc. Figure 2.3.b shows a separation line produced by the geodesic distance function for a shape with one hole.

In higher dimensions ($n \geqslant 3$) separation sets can also appear due to concavities in the shape. An example could be a room with a tall box in the middle, and distance computed by "air distance" (length of shortest path not going through the box). Even though the space defined by the air in the room does not have a hole in the topological sense, some points behind the box can be much easier reached by going around the box, on the left or right side, than climbing over it. When the distances computed going on the left and right sides are equal, the point belongs to a separation set.

Less formal, a separation set is the set of points that can be reached in a shortest distance over more than one path. For 2D shapes, it is the border between regions for which the geodesics to the source point **y** go on one or the other side of a hole. It characterizes the fact that a certain part of the shape can be reached in a shortest distance from more than one direction (on both sides of the hole). If the shape has many holes, these lines can intersect producing more complex structures formed of a union of lines/surfaces (e.g. a tree). An example is given in Figure 2.4.

Adding a hole (obstacle) to a shape results in some of the remaining points having a higher distance to the source point **y** than without the hole.

Definition 12. (Shadow of a hole) *The shadow of a hole, when considering the source point* **y** *is the (connected) set of pixels* $\mathbf{p} \in \mathcal{S}$*, for which* $d(\mathbf{y}, \mathbf{p})$ *changes when the shape is considered with and without the hole.*

Figure 2.5: Shadow of the hole in Figure 2.3.b, considering the same source point (bottom-left).

These are the points that are occluded by the hole and can now be reached only by going around it. Figure 2.5 shows the shadow of the hole in Figure 2.3.b when considering the same source point.

Property 5. *Separation sets lie in the shadow of holes i.e. shadows are supersets of the separation sets.*

Proof. The property above rises from the definition of the shadow and the separation set. Consider the point $\mathbf{y} \in \mathcal{S}$ as the point for which we compute $D^{\mathcal{S}}(\mathbf{y})$. Now assume $\exists \mathbf{p} \in \mathcal{S}$, $\mathbf{p}$ not in any shadow, such that $\mathbf{p}$ is in the separation set of $D^{\mathcal{S}}(\mathbf{y})$. But, $\mathbf{p}$ not in any shadow implies that the geodesic(s) $\nu(\mathbf{p}, \mathbf{y})$ is/are the same when computing on $\mathcal{S}$ with and without hole. As geodesics on a simply connected shape are unique, $\nu(\mathbf{p}, \mathbf{y})$ is unique, and $\mathbf{p}$ could not be in the separation set of $D^{\mathcal{S}}(\mathbf{y})$. □

2.2.1 Computing the geodesic distance function

Depending on the representation of the shape and the type of metric, different algorithms can be used. Table 2.2 gives an overview of the presented methods.

Other methods, like the Chamfer distance computation [Rosenfeld 66, Borgefors 86] use properties of the metric to approximate the distance corresponding to a pixel, based on the distances corresponding to the neighbors. Even though they are very efficient in the case of the Distance Transform (DT) (Section 2.2.3), which is a special case of the geodesic distance function, in general much more iterations may be required.

Graphs and discrete shapes using d_α:

For $\mathcal{S}$ represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathbf{y} \in \mathcal{V}$ a vertex, and for using the 4, 8, 6, or 26 connectivity, which can be easily transformed into graphs, Dijkstra's algorithm [Atallah 98] solves the shortest path problem for a single-source (geodesic distance function with a single marker) in $O(|\mathcal{V}|^2)$ in a simpler implementation, and $O(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$ if using more efficient

Table 2.2: Overview of the presented methods for computing the geodesic distance function.

Name	compute on	complexity	exact	notes
Dijkstra's alg.	graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$	$O(\lvert\mathcal{E}\rvert + \lvert\mathcal{V}\rvert \log \lvert\mathcal{V}\rvert)$	exact	discrete (given) neighborhood
FM	discrete shape $\mathcal{S}$	$O(\lvert\mathcal{S}\rvert \log \lvert\mathcal{S}\rvert)$	approx.	D_ε, easy to extend to $\mathbb{R}^n$
DCP	discrete shape $\mathcal{S}$	$O((k+1)\lvert\mathcal{S}\rvert \log \lvert\mathcal{S}\rvert)$	exact	D_ε, harder to extend to $\mathbb{R}^n$

(FM = Fast Marching, DCP = Discrete Circle Propagation, k = number of holes of $\mathcal{S} \subset \mathbb{R}^2$)

Algorithm 1 *Dijkstra*$(\mathcal{G}, \mathbf{y})$ - Compute SBDT using Dijkstra's algorithm.

Input: Discrete shape $\mathcal{S}$ represented as weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and starting vertex $\mathbf{y} \in \mathcal{V}$.

```
1: for all v ∈ V do D(v) ← ∞ /*initialize distance vector*/
2: D(y) ← 0
3: P = V /*all vertices are to be processed*/
4:
5: while P ≠ ∅ do
6:    v ↚ arg min{D(v) | v ∈ P} /*select and remove vertex with smallest estimated distance*/
7:
8:    for all u ∈ V | e = (v, u) ∈ E do
9:       d ← D(v) + w(e) /*dist. to u over v = dist. to v plus the weight of e = (v, u).*/
10:      if d < D(u) then
11:         D(u) ← d
12:      end if
13:   end for
14:
15: end while
```

Output: Distances D.

structures (e.g. a Fibonacci heap [Atallah 98]). Algorithm 1 gives the pseudo-code for Dijkstra's algorithm.

Discrete shape with Euclidean distance

For discrete shapes using d_ε, creating a graph as mentioned above, and applying Dijkstra's algorithm is in most cases not the most efficient solution. To obtain a correct result, the corresponding vertices of all pairwise visible points in $\mathcal{S}$ have to be connected by an edge, weighted with the Euclidean distance between the two points. Depending on the shape, the obtained graph can have a very high degree of connectivity (fully connected for convex shapes).

Two algorithms are proposed for this case: the first one, called *Fast Marching* (FM) [Sethian 99], relies on the local approximation of $D^{\mathcal{S}}$ using a formulation of a wave propagation as a partial

differential equation, the second one, *Discrete Circle Propagation* (DCP) [Ion 08d], relies on the simulation of the distance propagation using discrete analytical circles [Andres 94]. The approaches are different. FM computes an approximation[6], is easy to implement and extends to any dimension. DCP tries to give the exact result, with an algorithm for 2D given below, but the extension to 3D not straightforward (instead of working with angle intervals, like in 2D, the algorithm has to handle connected regions on a sphere). The complexities of the algorithms are comparable. A description of FM and DCP follows.

Fast marching (FM)

Related to Dijkstra's algorithm, Fast Marching [Sethian 99] is a method to solve boundary value problems of the form:

$$|\nabla T|F = 1 \tag{2.2}$$

The formula above describes the evolution of a closed curve as a function of time T and speed F. $T(\mathbf{p})$ is the time at which the curve reaches the point $\mathbf{p}$, $F(\mathbf{p})$ is the speed in the normal direction at a point $\mathbf{p}$ on the curve, and $\mathcal{Y} = \{\mathbf{p} \mid T(\mathbf{p}) = 0\}$ is the starting/source position of the curve. FM was designed for problems in which the speed never changes sign (direction), so that the front is always moving forward or backward and passing through each point only once. This allows to consider the problem as a stationary problem i.e. for each grid point $\mathbf{p}$ the value of $T(\mathbf{p})$ is "recorded". In the following paragraph, the case of $\mathcal{S} \subset \mathbb{Z}^2$ is discussed in more detail. Similar formulations exist for $\mathcal{S} \subset \mathbb{Z}^3$ and $\mathcal{S}$ a 2D manifold in 3D (e.g. a triangle mesh in $\mathbb{R}^3$).

For $\mathbf{p} \in \mathcal{S} \subset \mathbb{Z}^2$, Equation 2.2 can be approximated by [Rouy 92, Sethian 99]:

$$\sqrt{\max(\Delta^{-x}(\mathbf{p})T, \Delta^{+x}(\mathbf{p})T, 0)^2 + \max(\Delta^{-y}(\mathbf{p})T, \Delta^{+y}(\mathbf{p})T, 0)^2} = \frac{1}{F(\mathbf{p})} \tag{2.3}$$

where $\Delta^{-x}(\mathbf{p})$, $\Delta^{+x}(\mathbf{p})$, $\Delta^{-y}(\mathbf{p})$, $\Delta^{+y}(\mathbf{p})$ are the forward ($+x$ and $+y$) and backward ($-x$ and $-y$) difference operators of T at $\mathbf{p}$. The solution of Equation 2.3 is the value $t = T(\mathbf{p})$. Note that when implementing, for a point $\mathbf{p} = (x, y)$ with discrete coordinates x and y, $\Delta^{-x}(\mathbf{p})T$ is approximated by $T((x, y)) - T((x-1, y))$, $\Delta^{+x}(\mathbf{p})T$ by $T((x+1, y)) - T((x, y))$, and so on. $T(\mathbf{p})$ is the unknown value t, and $T((x-1, y))$, $T((x+1, y))$,..., are already known values.

The idea behind FM is to construct the solution T by using only upwind values, as it is permitted by the formulation above. A list $\mathcal{K}$ of known values for T is initialized with $\mathcal{Y}$ and maintained. In each step, one grid point $\mathbf{q}$ not in $\mathcal{K}$, with the smallest estimated $T(\mathbf{q})$ is considered computed and added to $\mathcal{K}$ [7]. The point $\mathbf{q}$ has to be 4-connected to a point in $\mathcal{K}$. If

[6] [Sethian 99] contains an in depth study of the approximation errors of first and second order schemes.
[7] This way of approaching the problem is similar to Dijkstra's algorithm.

efficient structures (e.g. a Fibonacci heap) are used to represent the set of points in $\mathcal{S}-\mathcal{K}$, that are neighbors with points from $\mathcal{K}$ and have estimated values for T (candidates to be moved to $\mathcal{K}$), the complexity of FM is $O(n \log n)$ in the number of grid points $n = |\mathcal{S}|$.

Considering that when moving with constant speed $F = 1$, the time a moving object needs to reach a certain point is equal to the distance covered, the following applies:

Property 6. *If the set of computed points is the shape $\mathcal{S}$, if $F = 1$ for all points, and if $\mathcal{Y} = \{\mathbf{y}\}$ with $\mathbf{y}$ a point in $\mathcal{S}$, the solution T is an approximation of the SBDT, $D^{\mathcal{S}}(\mathbf{y})$.*

2.2.2 Discrete circle propagation (DCP)

Another possibility to compute the SBDT (geodesic distance function with one source point) is to simulate the propagation of distances inside the shape, similar to the propagation of a wave starting at the source point $\mathbf{y}$ and traveling with constant speed [Ion 08d]. At any time $t > 0$ the wave front is a set of discrete arcs. Each arc $\mathcal{A}$ is centered at the point $\mathbf{r}$ where the geodesic from $\mathbf{p} \in \mathcal{A}$ to $\mathbf{y}$ first touches the boundary of $\mathcal{S}$. For each point of an arc, the geodesic distance to $\mathbf{y}$ satisfies $t \leqslant d^{\mathcal{S}}(\mathbf{p}, \mathbf{y}) < t + 1$. The arc centers $\mathbf{r}$ are called *reference points*.

Discrete analytical circles

The used discrete circle definition has to satisfy the property that circles centered on a point must fill, preferably pave, the discrete space. Points must not be missed during the propagation phase.

There exists several different discrete circle definitions. The best known circle is an algorithmic definition proposed by Bresenham [Bresenham 77] but this does not correspond to the requirements of our problem. The center coordinates and radius have to be integer, and circles with increasing integer radii do not pave the space. The discrete analytical circle proposed in [Andres 94] fits the requirements:

Definition 13. *A discrete analytical circle $\mathcal{C}(r, \mathbf{o})$ is defined by the double inequality:*

$$\mathbf{p} \in \mathcal{C}(r, \mathbf{o}) \Longleftrightarrow r \leqslant d(\mathbf{p}, \mathbf{o}) < r + 1$$

with $\mathbf{p} \in \mathbb{Z}^2, \mathbf{o} \in \mathbb{R}^2$, and $r \in \mathbb{R}^+$ the radius using Euclidean distance.

This circle has arithmetical thickness 1. The circle definition is similar to the discrete analytical line definition proposed by Reveillès [Reveillès 91].

The fact that this circle definition is analytical, and not just algorithmic like Bresenham's circle, has many advantages. Point localization i.e. knowing if a point is inside, outside or on

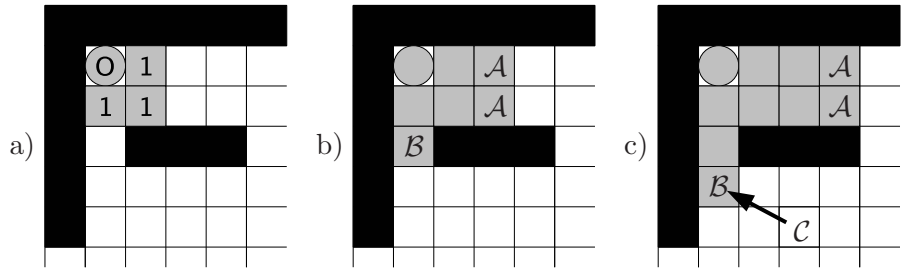


Figure 2.6: The three steps used during wave front propagation (white: shape, black: background). a) radius 1: circle with radius 1 and center $\mathbf{o}$ is bounded to an arc. b) radius 2: the front is split in two arcs $\mathcal{A}$, $\mathcal{B}$. c) radius 3: arc $\mathcal{B}$, touching the hole at radius 2 (the next arc pixel would fall in the hole) but not at radius 3, creates arc $\mathcal{C}$ with the center at the current point of $\mathcal{B}$.

the circle, is trivial. The center coordinates and the radius do not have to be integers. The circle definition can easily be extended to any dimension (see [Andres 97]). Circles with the same center pave the space:

$$\biguplus_{r=0}^{\infty} \mathcal{C}\,(r, \mathbf{o}) = \mathbb{Z}^2$$

Each integer coordinate point in space belongs to one and only one circle[8]. As shown in [Andres 97] a discrete circle of arithmetical thickness equal to 1 (this is the case with the given definition) is at least 8-connected. The discrete points of the circle can be ordered and there exists a generation algorithm with linear complexity [Andres 94, Andres 97]. Discrete circles have been used before to simulate discrete wave propagation [Mora 03].

Simulating the wave propagation to compute the distances

The wave-front propagating from a point $\mathbf{o}$ will have the form of a circle centered at $\mathbf{o}$. If the wave front is blocked by obstacles, the circle is "interrupted" and disjoint arcs of the same circle continue propagating in the unblocked directions. For a start point $\mathbf{o}$, the wave front at time t is the set of points $\mathbf{p} \in \mathcal{S}$ at distance $t \leqslant d(\mathbf{o}, \mathbf{p}) < t + 1$. The wave front at any time t consists of a set of arcs $\mathcal{W}(t) \subset \mathcal{S}$. Each arc $\mathcal{A} \in \mathcal{W}(t)$ lies on a circle centered at the point where the path from $\mathbf{p} \in \mathcal{A}$ to $\mathbf{o}$ first touches $\partial\mathcal{S}$ (the boundary pixels of $\mathcal{S}$).

The computation starts with a circle of radius 1 centered at the source point $\mathbf{o}$, $\mathcal{W}(1) = \{\mathcal{C}(1, \mathbf{o})\}$. It propagates and clusters as presented above (Figure 2.6), with the addition that pixels with already computed distance smaller than the current wave front also block the prop-

[8]When drawing Bresenham circles with the same center and increasing radii, there are discrete points that do not belong to any of the concentric circles.

agation. An arc $\mathcal{A} \in \mathcal{W}(t)$ of the wave front $\mathcal{W}(t)$, not touching $\partial\mathcal{S}$ at time t, but touching at $t-1$, diffracts new circles centered at the endpoints $\mathbf{s} \in \mathcal{A}$. The added arcs start with radius one and handicap $d(\mathbf{p}, \mathbf{s})$. The handicap of an arc accumulates the length of the shortest path to the center of the arc such that the distance of the wave front from the initial point $\mathbf{o}$ is always the sum of the handicap and the radius of the arc. No special computation is required for the initial angles of arcs with centers on the boundary pixels of $\mathcal{S}$. They will be corrected by the clustering and bounding steps when drawing with radius 1. See Figure 2.6 and Algorithm 2.

The complexity of Algorithm 2 is determined by the number of pixels in $\mathcal{S}$, denoted $|\mathcal{S}|$, and the number of arcs of the wave front. Arcs are drawn in $O(n)$ where n is the number of pixels of the arc. Pixels in the shadow of a hole, where different parts of the wave front meet (i.e. pixels on the separation lines), are drawn by each part. All other pixels are drawn only once. Adding and extracting arcs to/from the wave front ($\mathcal{W}$ in Algorithm 2) can be done in $\log(|\mathcal{W}|)$. For convex shapes, the size of $\mathcal{W}$ is 1 all the time, so the algorithm executes in $O(|\mathcal{S}|)$. For simply connected shapes, each pixel is drawn only once. Considering the upper bound $|\mathcal{W}| = |\mathcal{S}|$ and each arc only draws one pixel, the complexity for simply connected shapes is $O(|\mathcal{S}| \log |\mathcal{S}|)$. Each hole creates an additional direction to reach a point, e.g. no hole: 1 direction; 1 hole: 2 directions - one on each side of the hole; 2 holes: maximum 3 directions - one side, between holes, other side, etc.[9] For a non-simply connected 2D shape with k holes, a pixel is set a maximum of $k+1$ times (worst case). The worst case running time complexity for non-simply connected 2D shapes with k holes is $O((k+1)|\mathcal{S}| \log |\mathcal{S}|)$.

2.2.3 The distance transform (DT)

The *distance transform* (DT) [Rosenfeld 66, Rosenfeld 68, Rosenfeld 83] is probably the most known distance related transform in the image processing community. The DT is studied in depth in the discrete geometry and in the morphological image analysis communities (known also under the name *distance function* [Soille 02]). The DT can be formulated as a special case of the geodesic distance function presented in Section 2.1.6.

A feature that can be derived from the DT is the skeleton [Ogniewicz 95, Borgefors 99]. The skeleton is made out of the centers of maximally inscribed disks, and can then be used to describe shapes and their topology in a compact way (skeletons lead to even more robust shape descriptors, like the shock-graphs [Siddiqi 99, Sebastian 04]). Another feature is the "local width" of the shape, which is given by the DT values of points that have at least two boundary points which are closest (skeleton points).

Given a binary shape, the DT associates to each point of the shape (e.g. foreground pixel

[9] Note that we do not count the number of possible paths, but the number of directions from which connected wave fronts can reach a point at the shortest geodesic distance.

Algorithm 2 $DCP(\mathcal{S}, \mathbf{y})$ - Compute SBDT using discrete circles.

Input: Discrete shape $\mathcal{S}$ and pixel $\mathbf{y} \in \mathcal{S}$.

for all $\mathbf{q} \in \mathcal{S}$ **do** $D(\mathbf{q}) \leftarrow \infty$ /**initialize distance matrix**/
$D(\mathbf{y}) \leftarrow 0$
$\mathcal{W} \leftarrow \mathrm{Arc}(\mathbf{y}, 1, [0; 2\pi], 0, \emptyset)$ /**Arc(**c**enter, **r**adius, **a**ngles, **h**andicap, **p**arent)**/

while $\mathcal{W} \neq \emptyset$ **do**
  $\mathcal{A} \nleftarrow \arg\min\{\mathcal{A}.r + \mathcal{A}.h \mid \mathcal{A} \in \mathcal{W}\}$ /**select and remove arc with smallest radius+handicap**/

  /**draw arc points with lower distance than known before, use Euclidean distance**/
  $D(\mathbf{m}) \leftarrow \min\{D(\mathbf{m}), \mathcal{A}.h + d(\mathcal{A}.\mathbf{c}, \mathbf{m}) \mid \mathbf{m} \in \mathcal{A} \cap \mathcal{S}\}$

  $\mathcal{P}_1, \ldots, \mathcal{P}_k \leftarrow$actually drawn (sub)arcs/parts of $\mathcal{A}$ /**split and bound**/
  $\mathcal{W} \leftarrow \mathcal{W}+\mathrm{Arc}(\mathcal{A}.\mathbf{c}, \mathcal{A}.r + 1, \mathcal{P}_i.\mathbf{a}, \mathcal{A}.h, \mathcal{A}), \forall i = 1 \ldots k$ /**propagate**/

  /**diffract if necessary**/
  if $\mathcal{A}.p$ touches $\partial\mathcal{S}$ on either side **then**
    $\mathbf{s} \leftarrow$ last point of $\mathcal{A}$, on side where $\mathcal{A}.p$ was touching $\partial\mathcal{S}$
    $\mathcal{W} \leftarrow \mathcal{W}+\mathrm{Arc}(\mathbf{s}, 1, [0; 2\pi], D(\mathbf{s}), \mathcal{A})$
  end if
end while

Output: Distances D.

of a binary digital image) the distance to the closest boundary point. Figure 2.7 shows example binary shapes and their associated DT. More formally,

Definition 14. (Distance Transform) *Let $\mathcal{S}$ be a shape and d the used metric. The distance transform of $\mathcal{S}$ and point $\mathbf{p} \in \mathcal{S}$ is:*

$$DT(\mathcal{S}, \mathbf{p}) = \{\min(d(\mathbf{p}, \mathbf{q})) \mid \mathbf{q} \in \partial\mathcal{S}\}. \tag{2.4}$$

For nD shapes embedded in nD, the geodesic path to the closest boundary point is a straight line. In this case, it does not make any difference whether d is a geodesic distance defined with the geodesic mask $\mathcal{S}$, like in the case of the geodesic distance function (Section 2.1.6), or it is computed in the nD space in which $\mathcal{S}$ is embedded (e.g. ℓ^2*-norm*). This fact is important for computation, as it simplifies the problem (for example, instead of using Euclidean based geodesic distance, which depends on the two end points and the shape, one can consider the Euclidean distance itself, which depends only on the two end points). The used metrics span from the classical d_4, d_8, to so called *chamfer metrics* [Borgefors 86, Verwer 91, Fouard 05], which approximate the Euclidean distance using only integer numbers, to the "real" d_ε.

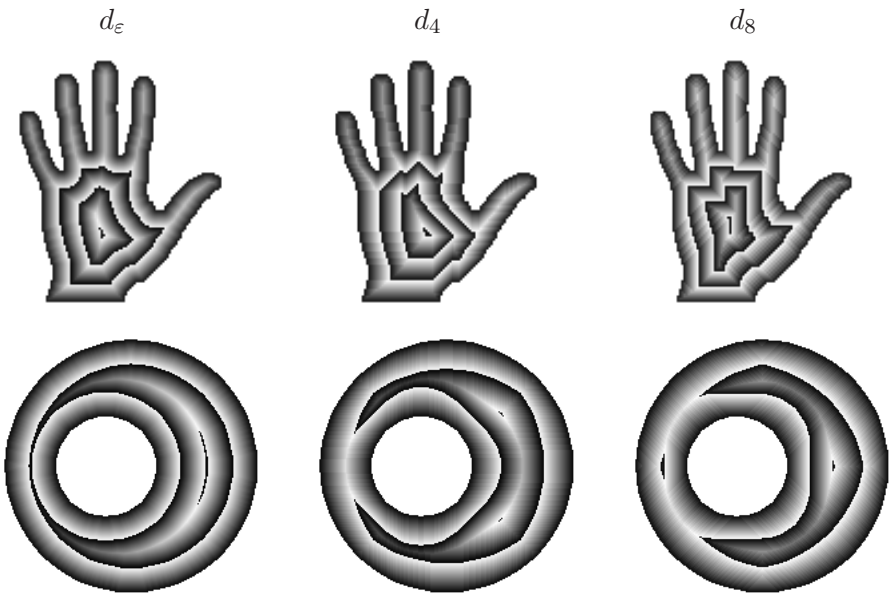


Figure 2.7: Distance transform for two shapes, using d_ε, d_4, and d_8. (Gray values are distances modulo a constant.)

The existing computation algorithms for the distance transform of a discrete shape can be divided into three categories:

(1.) chamfer based: [Rosenfeld 66, Rosenfeld 68, Borgefors 86];

(2.) Euclidean based:

- considering the steepest descent vector [Danielsson 80, Mullikin 92, Cuisenaire 99];
- computing the square of Euclidean distance [Saito 94, Hirata 96, Meijster 00];

See [Fabbri 08] for a survey of 2D Euclidean distance transform algorithms.

(3.) Voronoï [Voronoï 1907, Aurenhammer 96] based: [Breu 95, Guan 98, Coeurjolly 02], [Maurer Jr. 03].

To use the additional information available in grayscale images (shapes), paths/edges are weighted by a function of the gray value of their pixels/voxels [Rutovitz 68, Rutovitz 78, Piper 87, Shih 92, Huang 94, Toivanen 96, Qian 97].

2.3 Geodesic convexity

This section recalls the concept of geodesically convex sets (in our case shapes) and gives properties related to their intersection (see [Papadopoulos 03, Aleksandrov 04] for more details). They

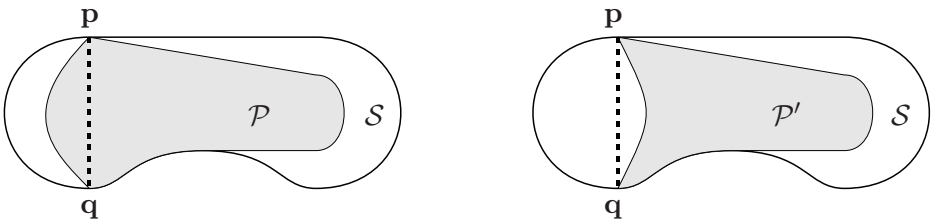


Figure 2.8: Example geodesically convex set $\mathcal{P} \subset \mathcal{S}$ (left) and not geodesically convex set $\mathcal{P}' \subset \mathcal{S}$ (right). The geodesic in $\mathcal{S}$ between points $\mathbf{p}, \mathbf{q}$ (dotted line) is contained in $\mathcal{P}$ but not in $\mathcal{P}'$.

are required for the proof of Property 23 in Section 3.4.3 (simply-connected, planar 2D shapes have a unique geodesic center).

The two properties below extend the concepts of convex and strictly convex sets to geodesics and geodesic distances. Instead of considering shortest paths in the space in which the shape $\mathcal{P}$ is embedded (e.g. the 2D Euclidean plane), geodesics computed using the geodesic mask $\mathcal{S} \supseteq \mathcal{P}$ are considered. See Figure 2.8 examples.

Definition 15. (Geodesically Convex) *Let $\mathcal{S}$ be a connected set with a geodesic distance $d^{\mathcal{S}}$, the connected set $\mathcal{P} \subseteq \mathcal{S}$ is said to be (geodesically) convex* **iff** *for any two points $\mathbf{p}, \mathbf{q} \in \mathcal{P}$ a geodesic $\nu \in \Pi^{\mathcal{S}}(\mathbf{p}, \mathbf{q})$ is contained in $\mathcal{P}$ i.e. $\nu \subseteq \mathcal{P}$.*

Definition 16. (Strictly geodesically convex) *A set $\mathcal{P} \subseteq \mathcal{S}$ is said to be strictly geodesically convex, if it is geodesically convex and for any two points $\mathbf{p}, \mathbf{q} \in \partial\mathcal{P}$, the geodesic $\nu \in \Pi^{\mathcal{S}}(\mathbf{p}, \mathbf{q})$, $\nu \subseteq \mathcal{P}$ touches $\partial\mathcal{P}$ only at $\mathbf{p}$ and $\mathbf{q}$.*

In other words, no interior point of any geodesic touches the boundary of $\mathcal{P}$.

The following properties consider the intersection of geodesically convex sets and the division of the space in which they are defined.

Theorem 1. (Intersection of geodesically convex sets) *The intersection of any collection of (strictly) geodesically convex sets defined over the same domain is itself (strictly) geodesically convex [Aleksandrov 04].*

Definition 17. (Supporting Hyperplane) *Suppose $\mathcal{P}$ is a geodesically convex set and $\mathbf{q} \in \partial\mathcal{P}$ is a point of the boundary. A supporting hyperplane is a hyperplane $\mathcal{H}$ with $\mathbf{q} \in \mathcal{H}$ such that $\mathcal{P}$ is entirely contained in one of the two regions that $\mathcal{H}$ divides the space into. For $\mathcal{P} \subseteq \mathcal{S}$ a two dimensional and geodesically convex shape, $\mathcal{H}$ is a geodesic dividing $\mathcal{S}$ in two parts ($\mathcal{H}$ is a cut of $\mathcal{S}$).*

For $\mathcal{S} \in \mathbb{R}^2$ the supporting hyperplane is a line, which at points where $\partial\mathcal{S}$ is smooth is equal to the tangent to $\mathcal{S}$.

Theorem 2. (Supporting Hyperplane Theorem) *A geodesically convex set has at least one supporting hyperplane at each point of its boundary [Aleksandrov 04]. (Can be related to the Hahn-Banach Separation Theorem [Narici 97]).*

A supporting hyperplane exists also when $\partial\mathcal{S}$ is not differentiable, and thus can be used also when $\partial\mathcal{S}$ is not smooth.

2.4 Chapter Summary

This chapter introduced the notions of *object*, *shape* and *distance*, which are required to define the eccentricity transform. Shapes characterize the form (space occupied) by objects, and can be continuous or discrete. Discrete shapes are represented as discrete points in $\mathbb{Z}^n$ or as graphs. Geodesic distances are lengths of shortest paths included in the shape. Given a set of source points, the geodesic distance function gives the geodesic distance of all points of a shape to the closest source point. Three methods to compute the geodesic distance function have been given. A known distance related image transform, is the distance transform. The distance transform takes a shape as an input and associates to each point of the shape the distance to the closest boundary point. The concept of geodesic convexity and important properties are given, as prerequisites for the properties in the following chapter.

3

The Eccentricity Transform

This chapter gives the formal definition of the eccentricity transform and related terms. For a list of basic shapes, their eccentricity transform and related properties are given. These shapes were chosen to illustrate the behavior of the eccentricity transform with increasing complexity of the shapes (dimension, convexity, holes, etc.) and to demonstrate properties that have lead to the eccentricity transform algorithms in Chapter 4. Experiments regarding robustness conclude the chapter.

3.1 Related Work

The concept of distance to the point furthest away has been approached independently in different communities and is known under different names.

In graph theory [Buckley 90, Berge 91, Diestel 97], the length of the shortest path connecting a vertex to a vertex furthest away is called *eccentricity* [Buckley 90]. It is used to define the *radius* (smallest eccentricity) and diameter (largest eccentricity) of a graph, but computing the value for all vertices at once (transform) is not approached.

In the computational geometry community, the concept is known under the term *furthest neighbor/furthest point*. Computation is approached for the corners of simple polygons in [Suri 87], where an algorithm with a running time of $O(n \log n)$ in the number of polygon corners is given. Asano and Toussaint [Asano 87] have studied the problem of computing the *geodesic center* of a simple polygon, which is a point that minimizes the maximum geodesic distance to any point in the polygon (distance to point furthest away). A faster algorithm than the one by Asano et al. ($O(n^4 \log n)$), with the worst-case running time $O(n \log^2 n)$ has been proposed by Pollack et al. [Pollack 89].

In the morphological image analysis community, the function associating to each point of a shape the geodesic distance to the point furthest away is called the *propagation function* [Soille 02]. Maisonneuve and Schmitt [Maisonneuve 89, Schmitt 93] have looked at the

computation for simply connected compact polygons using digital metrics[1]. Soille [Soille 94] has considered the propagation function using generalized geodesic distances that are less sensitive to bending. The discussion considered measuring the length (diameter) of a simply connected shape.

This thesis brings together the similar concepts from the three communities into a general formulation. It studies further properties and explicitly considers shapes that are not simply connected, which was not considered before.

3.2 Definition of Eccentricity and Eccentricity Transform

This section defines the eccentricity of a point and the eccentricity transform of a shape. Features and properties are presented in the following sections.

The definitions and properties follow [Kropatsch 06] (the eccentricity transform of a graph and of 4 and 8 connected planar shapes), [Ion 07a] (ECC of an ellipse, using d_ε), [Ion 07b] (4 connected planar shapes using d_ε), [Kropatsch 07] (polygonal shapes using d_ε), [Ion 08a] (6 connected 3D shapes using d_ε), and [Ion 08d] (efficient approximation and related properties).

Definition 18. (Eccentricity) *Let $\mathcal{S}$ be a shape and $d^{\mathcal{S}}$ the used geodesic distance. The eccentricity of a point $\mathbf{p} \in \mathcal{S}$ is:*

$$ECC(\mathcal{S}, \mathbf{p}) = \{\max(d^{\mathcal{S}}(\mathbf{p}, \mathbf{q})) \mid \mathbf{q} \in \mathcal{S}\}. \tag{3.1}$$

The eccentricity of a point $\mathbf{p}$ is the length of the longest geodesic that has $\mathbf{p}$ as one of its end points. This is the same as the length of the longest of the shortest paths connecting $\mathbf{p}$ to any other point $\mathbf{q} \in \mathcal{S}$.

Note that the definition does not depend on the existence or a priori knowledge of $\partial\mathcal{S}$.

Definition 19. (Eccentricity Transform) *The eccentricity transform (ECC) of a shape $\mathcal{S}$, using the geodesic distance $d^{\mathcal{S}}$, assigns the eccentricity $ECC(\mathcal{S}, \mathbf{p})$ to each point $\mathbf{p}$ of $\mathcal{S}$:*

$$[ECC(\mathcal{S})](\mathbf{p}) = ECC(\mathcal{S}, \mathbf{p}) = \{\max(d^{\mathcal{S}}(\mathbf{p}, \mathbf{q})) \mid \mathbf{q} \in \mathcal{S}\}.$$

Figure 3.1 shows an example 2D shape and its eccentricity transform using different metrics. The eccentricity transform is also known by the name *propagation function* [Soille 02] in the morphological image processing community.

[1] Digital metrics are continuous non-Euclidean distances that depend on a regular polygon inscribed in the circle of radius 1.

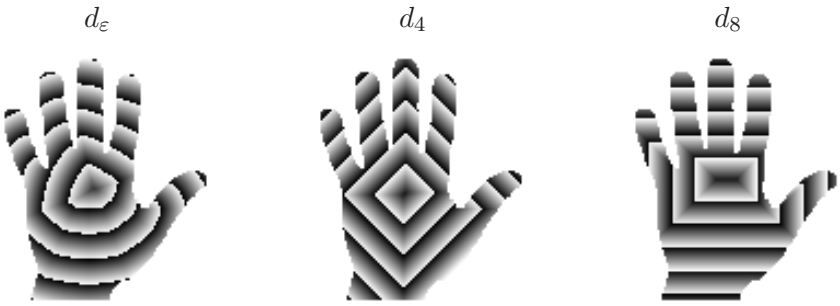


Figure 3.1: Eccentricity transform of a shape, using d_ε, d_4, and d_8. (Gray values are distances modulo a constant.)

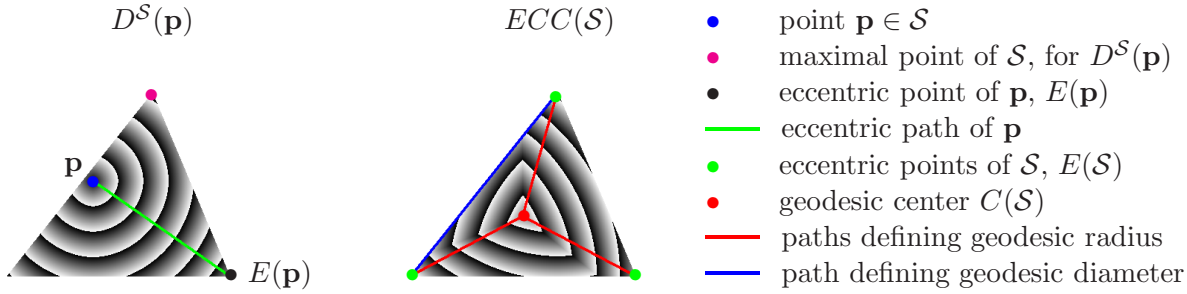


Figure 3.2: Examples for eccentricity related terms. (Gray values are distances modulo a constant.)

Note that, when computing the eccentricity transform, **the distance d and the distance function D are always the geodesic distance $d^{\mathcal{S}}$, respectively the geodesic distance function $D^{\mathcal{S}}$**, even though to help reading, the superscript $^{\mathcal{S}}$ might be left out.

3.3 Points with Special ECC Meaning, ECC Derived Features

The terms and properties in this section are defined from and for the ECC of a shape. In addition to the references mentioned before, for Definitions 24, 25 and 26, and Property 8, [Soille 02] was also considered. Figure 3.2 shows examples of the terms defined below.

Definition 20. (Maximal Point) *Given a shape $\mathcal{S}$ and a geodesic distance $d^{\mathcal{S}}$, a point $\mathbf{x} \in \mathcal{S}$ is called maximal* **iff** *$d^{\mathcal{S}}(\mathbf{p}, \mathbf{x})$ is a local maximum for some $\mathbf{p} \in \mathcal{S}$ i.e. $\mathbf{x}$ is a local maximum in $D^{\mathcal{S}}(\mathbf{p})$, the geodesic distance function associated to $d^{\mathcal{S}}$. $\mathcal{X}(\mathcal{S})$ denotes the set of all maximal points of shape $\mathcal{S}$.*

Definition 21. (Eccentric Point) *An eccentric point of a shape $\mathcal{S}$ is a point $\mathbf{q} \in \mathcal{S}$ that is farthest away in $\mathcal{S}$ from at least one other point $\mathbf{p} \in \mathcal{S}$ i.e. $\exists \mathbf{p} \in \mathcal{S}$ s.t. $ECC(\mathcal{S}, \mathbf{p}) = d^{\mathcal{S}}(\mathbf{p}, \mathbf{q})$.*

$E(\mathcal{S})$ denotes the set of eccentric points of the shape $\mathcal{S}$, and $E(\mathcal{S}, \mathbf{p})$ the set of eccentric points of a point $\mathbf{p}$ in $\mathcal{S}$.

Definition 22. (Eccentric Path) *An eccentric path of a point $\mathbf{p} \in \mathcal{S}$ is a geodesic connecting $\mathbf{p}$ with one of its eccentric points.*

Property 7. *The set of eccentric points $E(\mathcal{S})$ is a subset of the set of maximal points $\mathcal{X}(\mathcal{S})$.*

Eccentric points are global maxima in $D^{\mathcal{S}}(\mathbf{y})$, for some $\mathbf{y}$, while maximal points are local maxima.

Definition 23. (Eccentric point cluster) *An eccentric point cluster is a maximal connected set of eccentric points i.e. a connected set that is not a strict subset of any other connected set of eccentric points.*

Definition 24. (Geodesic ends of a shape) *Given a shape $\mathcal{S}$, a point $\mathbf{q} \in \mathcal{S}$ is called a geodesic end of $\mathcal{S}$ if $\mathbf{q}$ is a local maximum of the eccentricity transform $ECC(\mathcal{S})$ of the shape.*

Property 8. *The geodesic ends of a shape $\mathcal{S}$ are located on its boundary $\partial\mathcal{S}$ [Soille 02].*

Definition 25. (Geodesic radius and geodesic center) *The geodesic radius of a shape is the smallest eccentricity of a shape. The geodesic center or simply center of a shape $\mathcal{S}$ is the set of points of $\mathcal{S}$ having the eccentricity value equal to the radius:*

$$C(\mathcal{S}) = \{\mathbf{p} \in \mathcal{S} \mid ECC(\mathcal{S}, \mathbf{p}) = \min\{ECC(\mathcal{S})\}\}$$

By definition $C(\mathcal{S})$ belongs to $\mathcal{S}$, also if $\mathcal{S}$ is not convex or contains holes (as opposed to the centroid of a shape which can lie outside of it). The elements of $C(\mathcal{S})$ are called *geodesic center points* or simply *center points*.

Definition 26. (Geodesic length/diameter) *The geodesic length, or diameter, of a shape $\mathcal{S}$ is the length of the longest geodesic of $\mathcal{S}$, which coincides with the highest value of $ECC(\mathcal{S})$.*

3.3.1 Shape decomposition based on eccentric or reference points

Each point of a shape has at least one eccentric point. When the used geodesic distance function is continuous or a discrete approximation of a continuous function (e.g. d_α) the set of eccentric points of neighboring shape points has the "tendency" to be piece-wise constant i.e. $\exists \mathcal{P}_i \subset \mathcal{S}$ s.t. all $\mathbf{p} \in \mathcal{P}_i$ have the same set of eccentric points.

The *decomposition of a shape based on its eccentric points* (*ECC decomposition*) is done by grouping together neighboring points that have the same eccentric points. Figure 3.3 shows an

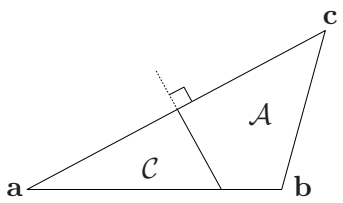
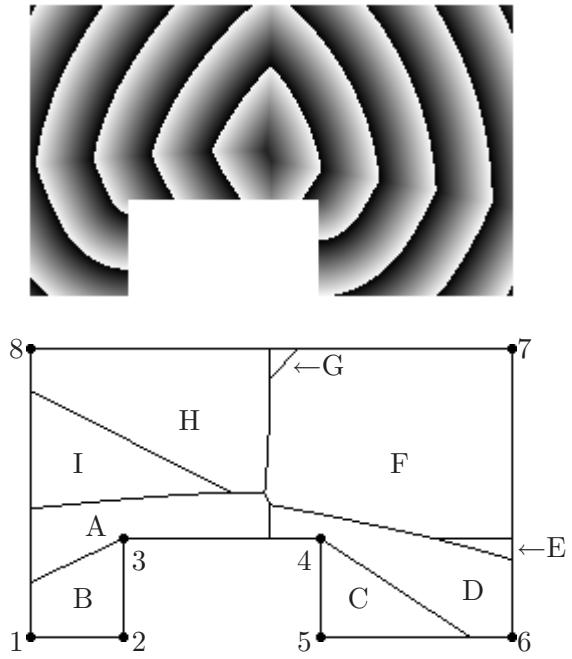


Figure 3.3: Decomposition based on eccentric points: $\mathcal{A}$, $\mathcal{C}$ regions corresponding to point **a**, respectively **c**.



patch	reference point	eccentric point
A	7	7
B	3	7
C	4	8
D	8	8
E	4	1
F	3	1
G	1	1
H	6	6
I	4	6

Figure 3.4: Eccentricity transform of a polygonal shape and its decomposition based on the *ECC* reference points. Some region borders (e.g. between regions H and F) are noticeable also on the eccentricity transform image.

example shape decomposition based on the eccentric points (Section 3.5.3 treats the eccentricity transform of a triangle in detail).

If the used distance and the shape $\mathcal{S}$ are continuous then the boundaries of the obtained regions are made out of points that have the union of the eccentric points of all the regions they are incident to (e.g. the points on the line separating regions $\mathcal{A}$ and $\mathcal{C}$ in Figure 3.3 have both **a** and **c** as eccentric points). If the set of eccentric points is finite then the decomposition has a finite number of regions, otherwise it can have any number of regions (e.g. the continuous 2D disk with d_ε has an infinite number of eccentric points and regions - Section 3.5.4).

For convex shapes, the geodesics are straight lines and only touch the boundary if one of the end points is a boundary point. For non convex shapes, geodesics $\nu(\mathbf{y}, \mathbf{p})$ are polygonal lines of

the form $(\mathbf{y}, \mathbf{l}_1, \ldots, \mathbf{l}_n, \mathbf{p})$. Where $\mathbf{l}_i$, $1 \leqslant i \leqslant n$, are $n \geqslant 0$ points of $\partial\mathcal{S}$. For smooth shapes, n can be infinitely large. If $n = 0$ in the previous relation, the *reference point* for a point $\mathbf{p}$ is the point $\mathbf{y}$, otherwise it is $\mathbf{l}_n$. In other words, reference points are the points $\mathbf{r}$ where the geodesic $\nu(\mathbf{y}, \mathbf{p})$ last touches the shape boundary i.e. $\{\mathbf{r} \in \nu(\mathbf{y}, \mathbf{r}) \mid \mathbf{r} \in \partial\mathcal{S} \text{ and } d(\mathbf{r}, \mathbf{p}) = \min\}$. If $\nu(\mathbf{y}, \mathbf{p})$ does not touch $\partial\mathcal{S}$, the reference point of $\mathbf{p}$ is the starting point $\mathbf{y}$ itself. (Section 2.2.2 shows the usage of reference points as discrete circle centers.)

The *decomposition of a shape based on its ECC reference points* is done by grouping together neighboring points that have the same reference point(s) for their corresponding eccentric paths and have the same eccentric point(s). Figure 3.4 shows an example shape decomposition based on the reference points. In [Kropatsch 07] a method to compute the decomposition is given and discussed. The method decomposes a polygonal shape, in parallel, for each candidate eccentric point. Then it combines the patches to produce the final decomposition. This decomposition also gives a way to represent the eccentricity transform of a continuous polygonal shape with or without holes (without the need for discretization or approximation).

3.4 Properties Related to the Eccentricity

The properties given in this section refer to the case of using Euclidean distance i.e. $d = d_\varepsilon^{\mathcal{S}}$. Nevertheless, Properties 9-12, 15, 19

3.4.1 Eccentric and Maximal Points

Property 9. *All eccentric points of a simply connected planar shape $\mathcal{S}$ are located on its boundary $\partial\mathcal{S}$.*

Proof. Any geodesic $\nu \in \Pi(\mathbf{p}, \mathbf{q})$ with $\mathbf{q} \notin \partial\mathcal{S}$ can be prolongated by a line segment starting at $\mathbf{q}$ in the direction pointed by the tangent to ν at $\mathbf{q}$. If $\mathbf{q} \in \partial\mathcal{S}$ the direction of the tangent to ν at $\mathbf{q}$ can point outside $\mathcal{S}$ (e.g. when ν is normal to $\mathcal{S}$ at $\mathbf{q}$) - in this case ν can no longer be prolongated and from all points of ν, $\mathbf{q}$ is the point farthest away from $\mathbf{p}$. □

For $\mathcal{S}$ strictly convex at $\mathbf{q}$ and $d = d_\varepsilon$, any line segment partially contained in $\mathcal{S}$, passing through $\mathbf{q}$, will "exit" $\mathcal{S}$ at $\mathbf{q}$.

Property 10. *All eccentric points of a planar shape with one hole are located on its boundary.*

Proof. A single hole in $\mathcal{S}$ produces a separation line $\mathcal{L}$ in $D^{\mathcal{S}}(\mathbf{p})$. One can consider cutting $\mathcal{S}$ along $\mathcal{L}$, producing a simply connected shape $\mathcal{S}'$. From the definition of the separation set, a path passing over $\mathcal{L}$ cannot be a geodesic, which implies that no geodesics in $D^{\mathcal{S}}(\mathbf{p})$ go over $\mathcal{L}$ and $D^{\mathcal{S}}(\mathbf{p}) = D^{\mathcal{S}'}(\mathbf{p})$. Property 10 is then equivalent to using Property 9 for $\mathcal{S}'$. □

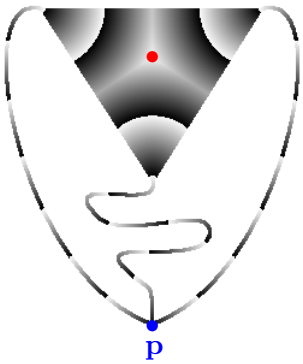


Figure 3.5: Shape and point **p**, with an eccentric point $E(\mathbf{p})$ inside the shape. (Gray values are distances to **p** modulo a constant.)

Property 11. *If the shape has more than one hole, eccentric points could exist inside the shape i.e.* $E(\mathcal{S}) \nsubseteq \partial\mathcal{S}$

Figure 3.5 shows an example shape. The distance from **p** to the three triangle corners is equal, and the triangle is equilateral. For the point **p**, $D^{\mathcal{S}}(\mathbf{p})$ contains a more complex separation set $\mathcal{L}$ which is the result of the two or three ways that some points can be reached in a shortest distance. Each part of $\mathcal{L}$ is the result of a combination of two ways that distances can be propagated from. The intersection point of the three lines has the highest distance. Not for all shapes with two holes, the separation lines intersect. Having an eccentric point in the inside of the shape happens only if such very special configurations exist, in addition to no other part of the shape being further away.

Property 12. (Eccentric points on $\partial\mathcal{S}$) *For planar shapes with less than two holes, all eccentric points are located on* $\partial\mathcal{S}$ *(immediate result of Properties 9, 10 and 11).*

Property 13. *No eccentric points* $E(\mathcal{S})$ *of a simply connected shape lie on a concave or straight part of the boundary of* $\mathcal{S}$ *i.e.* $\nexists \mathbf{e} \in E(\mathcal{S})$ *s.t.* $\partial\mathcal{S}$ *is concave or straight at* **e**.

Proof. In a 2D plane, all points at the same distance to a point **p** lie on a circle $\mathcal{C}(\mathbf{p}, r)$. If the circle is contained in $\mathcal{S}$ then there are points further away from **p**. If $\mathbf{l} \in \partial\mathcal{S}$ is on a straight or concave part of $\partial\mathcal{S}$ then any circle $\mathcal{C}$ tangent to $\partial\mathcal{S}$ at **l** is partly inside the shape around **l**. Thus there exists a point $\mathbf{q} \in \mathcal{S}$ with $\mathbf{q} \notin \mathcal{C}$ s.t. $d(\mathbf{p}, \mathbf{q}) > r$. The previous is valid for higher dimensional spaces ($\mathbb{R}^n$) if considering hyperspheres instead of circles. □

If $\mathcal{S}$ has a hole and **l** is located on the separation set of $D^{\mathcal{S}}(\mathbf{p})$, the additional constraint of paths not crossing the separation set $\mathcal{L}$ applies. In this case, eccentric points can be found also on straight and concave parts of the boundary of $\mathcal{S}$ (Figure 3.6). Property 13 is only valid for $\mathcal{S}$ with no holes.

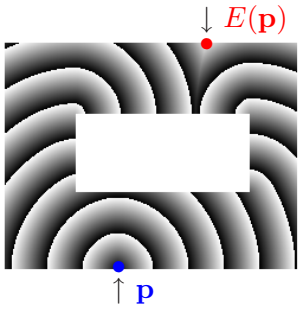


Figure 3.6: Case where an eccentric point lies on a straight part of the boundary. (Gray values are distances modulo a constant.)

.

Property 14. *(reformulation of Property 13) All eccentric points $E(\mathcal{S})$ of a simply connected shape lie on convex parts of the boundary of $\mathcal{S}$ i.e $\forall \mathbf{e} \in E(\mathcal{S}) \Rightarrow \partial\mathcal{S}$ is convex at $\mathbf{e}$.*

Properties 12, 13 and 14 also apply to maximal points.

Property 15. (Being an eccentric point is not a local property) *For any shape $\mathcal{S}$ and for any point $\mathbf{p} \in \mathcal{S}$, $\exists$ a shape $\hat{\mathcal{S}} = \mathcal{S} \cup \mathcal{S}'$, $\mathcal{S}'$ simply connected and $\mathcal{S} \cap \mathcal{S}' = \emptyset$, such that $\mathbf{p}$ is not an eccentric point of $\hat{\mathcal{S}}$ and $E(\hat{\mathcal{S}}) \subset \mathcal{S}'$.*

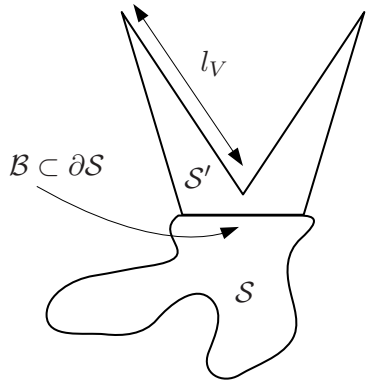


Figure 3.7: Adding part $\mathcal{S}'$ to $\mathcal{S}$ s.t. no eccentric point lies in $\mathcal{S}$ (see Property 15).

Proof. Let l be the length of $\partial\mathcal{S}$, and let $\mathcal{B} \subset \partial\mathcal{S}$, $\mathcal{B} \neq \emptyset$, $\lambda(\mathcal{B}) \ll l$, a boundary part of $\mathcal{S}$. We construct $\mathcal{S}'$ with the shape of a capital "V" glued at its endpoints with the endpoints of $\mathcal{B}$, and the length of the two branches $l_V > 2l \geqslant 2\max(ECC(\mathcal{S}))$ (Figure 3.7). The obtained shape $\hat{\mathcal{S}} = \mathcal{S} \cup \mathcal{S}'$ will have two eccentric points at the top of the two branches of the "V", a diameter $\max(ECC(\hat{\mathcal{S}})) = 2l_V$, and no eccentric point will lie on $\mathcal{S}$. If $\mathcal{S}$ has holes, $\mathcal{B}$ is taken from the

part of $\partial\mathcal{S}$ separating $\mathcal{S}$ from the infinite space (not from the enclosed holes), and l is the sum of the length of all parts of $\partial\mathcal{S}$. □

Property 16. *Every shape $\mathcal{S}$, $|\mathcal{S}| > 1$ ($\mathcal{S}$ is made out of more than one point), has at least two points with the highest eccentricity (global maxima).*

Proof. Let $\mathbf{p} \in \mathcal{S}$ with $ECC(\mathcal{S}, \mathbf{p}) = \max\{ECC(\mathcal{S})\}$ be a point with the highest eccentricity i.e. $\forall \mathbf{u} \in \mathcal{S}$, $ECC(\mathcal{S}, \mathbf{u}) \leqslant ECC(\mathcal{S}, \mathbf{p})$. From the definition of eccentricity, $\exists \mathbf{q} \in \mathcal{S}$, $\mathbf{q} \neq \mathbf{p}$ s.t. $d(\mathbf{p}, \mathbf{q}) = ECC(\mathcal{S}, \mathbf{p}) \implies ECC(\mathcal{S}, \mathbf{q}) \geqslant d(\mathbf{p}, \mathbf{q}) = ECC(\mathcal{S}, \mathbf{p})$. As $\mathbf{p}$ has maximum eccentricity, $ECC(\mathcal{S}, \mathbf{q}) \leqslant ECC(\mathcal{S}, \mathbf{p}) \implies ECC(\mathcal{S}, \mathbf{q}) = ECC(\mathcal{S}, \mathbf{p})$. □

The following property is a general property for symmetric (having one axis of symmetry), simply connected, convex shapes.

Property 17. (decomposition of symmetric shapes) *Let $\mathcal{S}$ be a symmetric, simply connected and convex shape, with A its axis of symmetry. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ denote the two symmetric parts of $\mathcal{S}$ separated by A. Any point $\mathbf{p} \in \mathcal{S}_1$ has its eccentric point $\mathbf{e}$ in $\mathcal{S}_2$, and vice versa.*

Proof. Let $\mathbf{p} \in S_1$ and $\mathbf{e} \in \mathcal{S}_1$ its eccentric point. Point $\mathbf{e} \in \mathcal{S}_1$ has the symmetric point $\mathbf{e}' \in \mathcal{S}_2$. $\mathcal{S}$ is convex, thus geodesics are straight lines. The straight line connecting $\mathbf{p}$ and $\mathbf{e}'$ intersects A in point $\mathbf{q} \in A$ having the same distance to $\mathbf{e}$ and $\mathbf{e}'$ ($\mathbf{q}$ is on the axis of symmetry and $\mathbf{e}, \mathbf{e}'$ are symmetric). Then $d(\mathbf{p}, \mathbf{e}') = d(\mathbf{p}, \mathbf{q}) + d(\mathbf{q}, \mathbf{e}') = d(\mathbf{p}, \mathbf{q}) + d(\mathbf{q}, \mathbf{e})$. Due to the triangular inequality $d(\mathbf{p}, \mathbf{q}) + d(\mathbf{q}, \mathbf{e}) > d(\mathbf{p}, \mathbf{e})$ and thus $\mathbf{e}$ cannot be the eccentric point of $\mathbf{p}$. □

3.4.2 Center - eccentric points and paths

Property 18. *A center point $\mathbf{c} \in C(\mathcal{S})$ of a simply connected planar shape $\mathcal{S}$ has at least two eccentric points.*

Proof. $D^{\mathcal{S}}(\mathbf{p})$ defines for each point $\mathbf{q} \in \mathcal{S}$ a unique direction of steepest descent and a neighbor closer to $\mathbf{p}$ in that direction. If a center point $\mathbf{c}$ would have a single eccentric point $\mathbf{e} = E(\mathcal{S}, \mathbf{c})$ then the neighbor $\mathbf{c}'$ of $\mathbf{c}$ in the direction of steepest descent of $D^{\mathcal{S}}(\mathbf{e})$ would have $d(\mathbf{c}', \mathbf{e}) < d(\mathbf{c}, \mathbf{e})$ making $ECC(\mathcal{S}, \mathbf{c}') < ECC(\mathcal{S}, \mathbf{c})$ which contradicts the assumption that $\mathbf{c}$ is a center point. □

If the shape $\mathcal{S}$ has a hole or the shape is not planar, there exist points (e.g. the points on the separation set) that do not have a unique direction of steepest descent, and have two geodesics connecting them to the same source point $\mathbf{p}$. In this case, a center point can have a single eccentric point (Figure 3.33 shows an example shape).

Property 19. *Any center point $\mathbf{c}$ of any shape $\mathcal{S}$ has at least two eccentric paths.*

Proof. Any geodesic $\nu(\mathbf{q},\mathbf{p})$ in $\mathcal{S}$ defines for each point $\mathbf{q}\in\mathcal{S}$, a neighbor $\mathbf{q}'$ closer to $\mathbf{p}$, and a direction $\overrightarrow{\mathbf{qq}'}$. If a center point $\mathbf{c}$ would have a single eccentric path $\nu(\mathbf{c},\mathbf{e})$ then the neighbor $\mathbf{c}'\in\nu$ of $\mathbf{c}$, in the direction $\overrightarrow{\mathbf{cc}'}$ (direction of tangent to ν at $\mathbf{c}$) would have $d(\mathbf{c}',\mathbf{e})<d(\mathbf{c},\mathbf{e})$ making $ECC(\mathcal{S},\mathbf{c}')<ECC(\mathcal{S},\mathbf{c})$ which contradicts the assumption that $\mathbf{c}$ is a center point. □

3.4.3 Single center (minimum) point

This section focuses on showing that planar simply connected continuous shapes $\mathcal{S}$ using the Euclidean based geodesic distance $d_\varepsilon^{\mathcal{S}}$ have a geodesic center made out of a single point (Property 23). The proof of the property has been divided in smaller properties and their proofs, which are first given.

Overall concept

Equation 3.1 (definition of the eccentricity transform) can be rewritten using functions instead of points as:

$$ECC(\mathcal{S})=\max\{D^{\mathcal{S}}(\mathbf{y})\ \mid \mathbf{y}\in\mathcal{S}\} \tag{3.2}$$

Note that $D^{\mathcal{S}}(\mathbf{y})$ is a function defined for all points $\mathbf{p}\in\mathcal{S}$ i.e. we can compute $[D^{\mathcal{S}}(\mathbf{y})](\mathbf{p})$ to get the distance value associated to a point $\mathbf{p}$. The 'max' above operates on functions defined over $\mathcal{S}$ and not on single scalar values. As the 'max' operator is distributive and commutative, the previous is equal to:

$$ECC(\mathcal{S})=\max\{D^{\mathcal{S}}(\mathbf{y}_1),\max\{D^{\mathcal{S}}(\mathbf{y}_2),\max\{D^{\mathcal{S}}(\mathbf{y}_3),\dots\}\}\},\ \cup\{\mathbf{y}_i\}=\mathcal{S}$$

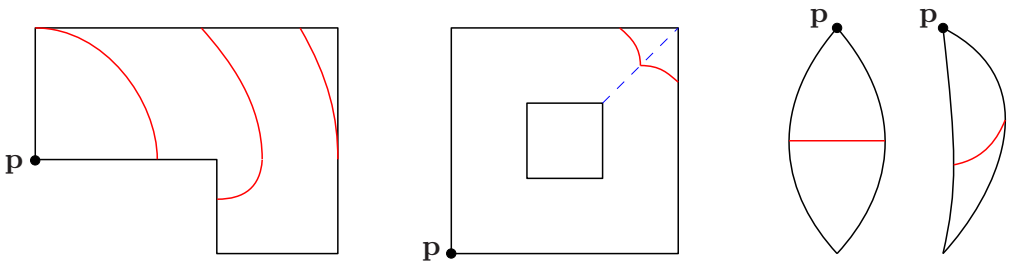
In the following we will show that a function $f_m:\mathcal{S}\to\mathbb{R}$ of the form $f_m=\max(f_1,f_2)$ has common properties with the two functions f_1 and f_2, if f_1,f_2 share the same (five) properties discussed in the following (Page 38).

The remaining part of this subsection considers functions $f:\mathcal{S}\to\mathbb{R}$, continuous, with a single minimum $\mathbf{m}$. Any cut divides $\mathcal{S}$ in two disjoint parts $\mathcal{S}'$ and $\mathcal{S}''$, $\mathcal{S}'$ is considered the part containing the minimum $\mathbf{m}$.

Prerequisites

Definition 27. (Level set) *The level set [Weisstein 02] of a function $f:\mathbb{R}^n\to\mathbb{R}$, corresponding to a value h, is the set of points $\mathbf{p}\in\mathbb{R}^n$ s.t. $f(\mathbf{p})=h$ i.e.*

$$\{\mathbf{p}\in\mathcal{S}\mid f(\mathbf{p})=h\}=f^{-1}(h)$$



a) rectangle and $D_\varepsilon^{\mathcal{S}}(\mathbf{p})$ b) one hole and $D_\varepsilon^{\mathcal{S}}(\mathbf{p})$ c) section of sphere and $D_\varepsilon^{\mathcal{S}}(\mathbf{p})$ on surface

Figure 3.8: Example (a) iso-convex and (b,c) non iso-convex functions. Isolines in red.

If $n = 2$ and $f : \mathbb{R}^2 \to \mathbb{R}$, the connected components of the level sets of f form one dimensional manifolds called *isolines*. If $n = 3$ and $f : \mathbb{R}^3 \to \mathbb{R}$, they are called *isosurfaces*.

Definition 28. (Isoline-cut) *A function $f : \mathcal{S} \to \mathbb{R}$, with $\mathcal{S} \subset \mathbb{R}^2$, is called isoline-cut* **iff** $\forall \mathcal{W} \subset \mathcal{S}$ *an isoline of f i.e. $\mathcal{W}$ is a connected component of the level set $f^{-1}(h), h \in (\min\{f\}, \max\{f\}]$, $\mathcal{W}$ cuts $\mathcal{S}$ in two disjoint parts $\mathcal{S}'$ and $\mathcal{S}''$, $\mathcal{S}' \cup \mathcal{S}'' \cup f^{-1}(h) = \mathcal{S}$. $\mathcal{S}'$ is the part containing* $\mathbf{m}$.

Definition 29. (Iso-convex) *A function $f : \mathcal{S} \to \mathbb{R}$, with $\mathcal{S} \subset \mathbb{R}^2$, is called iso-convex* **iff** *f is isoline-cut, and for any $\mathcal{W}$ corresponding to a level $h \in (\min\{f\}, \max\{f\}]$ we have: $\forall \mathbf{p}, \mathbf{q} \in \mathcal{W}$, $\mathbf{p} \neq \mathbf{q}$, $\mathcal{W} \cap \nu(\mathbf{p}, \mathbf{q}) = \{\mathbf{p}, \mathbf{q}\}$ and $\{\mathbf{u} \in \nu(\mathbf{p}, \mathbf{q}) \mid \mathbf{u} \neq \mathbf{p} \wedge \mathbf{u} \neq \mathbf{q}\} \subseteq \mathcal{S}'$ i.e. the isolines $\mathcal{W}$ are strictly convex when considering the part $\mathcal{S}'$ with the minimum $\mathbf{m}$ (see Definition 16).*

An example iso-convex function is the Euclidean based geodesic distance function $D_\varepsilon^{\mathcal{S}}(\mathbf{y}) : \mathcal{S} \to \mathbb{R}$ (Section 2.2) for $\mathcal{S} \subset \mathbb{R}^2$, planar and simply connected (Figure 3.8.a). A counter example is $\mathcal{S} \subset \mathbb{R}^2$ not simply connected (Figure 3.8.b) or $\mathcal{S}$ a slice of the surface of a sphere (e.g. between meridian 0° and 5°, Figure 3.8.c) - in both cases there are points where the isoline is straight or not convex. The D_4 and D_8 geodesic distance functions on a rectangle are also not iso-convex, as their isolines can contain straight lines.

The following two properties consider the values of f in $\mathcal{S}''$ when cutting $\mathcal{S}$ along an isoline $\mathcal{W}$.

Property 20. (Monotonic path always exists) *For all points $\mathbf{p} \in \mathcal{S}$, $\mathbf{p} \neq \mathbf{m}$, $\exists \pi \in \Pi(\mathbf{p}, \mathbf{m})$ s.t. $g : \pi \to \mathbb{R}$, $g(\mathbf{q}) = f(\mathbf{q})$, is strictly monotonic with $g(\mathbf{p})$ its maximum and $g(\mathbf{m})$ its minimum.*

The property above is a direct result of the fact that every point, except $\mathbf{m}$ has at least one neighbor with a smaller value.

Property 21. (All points in $\mathcal{S}''$ have values larger than the points of the cut $\mathcal{W}$) *For f continuous, with a single minimum $\mathbf{m}$, and isoline-cut: for all isolines $\mathcal{W}$ corresponding to a level $h \in (\min\{f\}, \max\{f\}]$, all $\mathbf{p} \in \mathcal{S}''$ have $f(\mathbf{p}) > h$.*

Proof. Assume $\exists \mathbf{q} \in S''$ s.t. $f(\mathbf{q}) \leqslant h$. Property 20 implies $\exists \pi(\mathbf{q}, \mathbf{m})$ strictly monotonic. From Definition 28, any path from $\mathbf{q} \in \mathcal{S}''$ to $\mathbf{m} \in \mathcal{S}'$ has to contain at least a point $\mathbf{l} \in \mathcal{W}$ of the cut. But $f(\mathbf{l}) = h \geqslant f(\mathbf{q})$, which contradicts the assumption. □

The five properties

The following five properties, that are satisfied by $D_{\varepsilon}^{\mathcal{S}}(\mathbf{y})$ for $\mathcal{S}$ continuous, planar and simply connected and $\mathbf{y} \in \mathcal{S}$, are required for Property 23. They concern functions of the form $f : \mathcal{S} \to \mathbb{R}$.

i. (*simply connected domain*) $\mathcal{S}$ is simply connected;

ii. (*continuous*) f is continuous over the whole domain $\mathcal{S}$;

iii. (*single min*) f has a single local minimum $\mathbf{m}$, which is also the global minimum of f;

iv. (*isoline-cut*) f is isoline-cut (Definition 28);

v. (*iso-convex*) f is iso-convex (Definition 29).

These properties are referred to using the roman numbers on the left (**i.**–**v.**).

The following lemmas and properties consider functions $f_m, f_1, f_2 : \mathcal{S} \to \mathbb{R}$, where f_m is defined as $f_m(\mathbf{p}) = \max(f_1(\mathbf{p}), f_2(\mathbf{p}))$, $\forall \mathbf{p} \in \mathcal{S}$, and $\mathcal{S} \subset \mathbb{R}^2$. The domain $\mathcal{S}$ is simply connected, and the functions f_1 and f_2 are (**Properties i.–v.** on Page 38): continuous, have a single minimum $\mathbf{m}_1$ respectively $\mathbf{m}_2$, are isoline-cut (Definition 28) and iso-convex (Definition 29). The purpose is to show that **the function $f_m = \max\{f_1, f_2\}$ also possesses the same 5 properties**. Figure 3.9 illustrates the used notation.

Properties i. and ii.:

From f_m defined on the same simply connected domain as f_1 and f_2, the definition domain of f_m is simply connected (Property i.). Also the maximum of two continuous functions is continuous (Property ii.).

Property iii.:

For Property iii., the following two options exist for a point $\mathbf{p} \in \mathcal{S}$: $f_1(\mathbf{p}) \neq f_2(\mathbf{p})$ (Lemma 1) and $f_1(\mathbf{p}) = f_2(\mathbf{p})$ (Lemma 2).

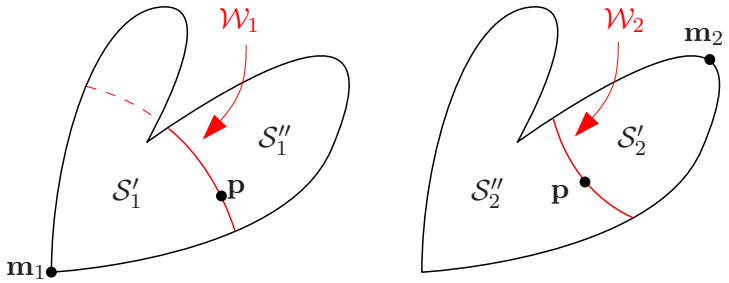
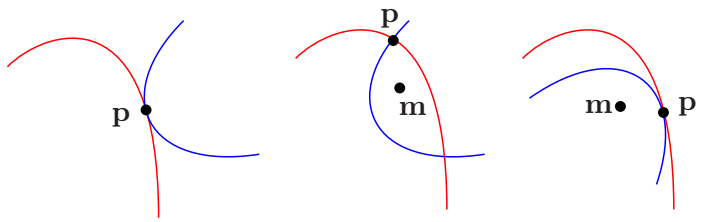


Figure 3.9: Notation used for proof of Properties **i.**–**v.** (Page 38) and Property 22. Level set in red.



a) $\mathcal{P} = (\mathcal{S}'_1 \cap \mathcal{S}'_2) = \emptyset$ b) $\mathcal{P} \neq \emptyset$ c) $\mathcal{P} \neq \emptyset$

Figure 3.10: Example configurations of two convex isolines (see Lemma 2).

Lemma 1. *For $f_1(\mathbf{p}) \neq f_2(\mathbf{p})$, f_1, f_2 as defined above, $\mathbf{p}$ is the single minimum of f_m* **iff** *$\mathbf{p}$ is the single minimum of the function f_1 or f_2 that has the highest value at $\mathbf{p}$.*

Proof. Assume $f_1(\mathbf{p}) > f_2(\mathbf{p})$, without constraining the result. The following options exist:

(1.) $\mathbf{p} = \mathbf{m}_1$ ($\mathbf{p}$ is the point that reaches the minimum of f_1)
$\implies \forall \mathbf{q} \in \mathcal{S}, \mathbf{q} \neq \mathbf{p}$, $f_m(\mathbf{q}) \geqslant f_1(\mathbf{q}) > f_1(\mathbf{m}) = f_m(\mathbf{p}) \implies f_m(\mathbf{q}) > f_m(\mathbf{p})$ and $\mathbf{p}$ is the single minimum of f_m;

(2.) $\mathbf{p} \neq \mathbf{m}_1$ ($\mathbf{p}$ is **not** the point that reaches the minimum of f_1)
$\implies \exists \mathbf{u} \in \mathcal{U}_\varepsilon(\mathbf{p})$ ($\mathbf{u}$ neighbor of $\mathbf{p}$) s.t. $f_2(\mathbf{u}) < f_m(\mathbf{u}) = f_1(\mathbf{u}) < f_1(\mathbf{p}) = f_m(\mathbf{p})$
$\implies f_m(\mathbf{u}) < f_m(\mathbf{p})$ and $\mathbf{p}$ is not a local thus also not a global minimum of f_m.

□

Lemma 2. *For $f_1(\mathbf{p}) = f_2(\mathbf{p})$, f_1, f_2 as defined above; either $\mathbf{p}$ is not a local minimum or it is the single minimum of f_m.*

Proof. Let $h = f_1(\mathbf{p}) = f_2(\mathbf{p})$. Considering whether $\mathbf{p}$ is the point where one of the two functions f_1, f_2 has its minimum, we have:

(1.) $\mathbf{p} = \mathbf{m}_1$ or $\mathbf{p} = \mathbf{m}_2$ (f_1 or f_2 have their minimum at $\mathbf{p}$)
assume $\mathbf{p} = \mathbf{m}_1 \implies \forall \mathbf{q} \in \mathcal{S},\ \mathbf{q} \neq \mathbf{p},\ f_m(\mathbf{q}) \geqslant f_1(\mathbf{q}) > f_1(\mathbf{m}_1) = f_m(\mathbf{p}) > f_2(\mathbf{m}_2)$ $\implies f_m(\mathbf{q}) > f_m(\mathbf{p})$ and $\mathbf{p}$ is the single minimum of f_m. The same reasoning is valid for $\mathbf{p} = \mathbf{m}_2$;

(2.) $\mathbf{p} \neq \mathbf{m}_1$ and $\mathbf{p} \neq \mathbf{m}_2$ (none of f_1, f_2 has the minimum at $\mathbf{p}$)
in other words $\{\mathbf{q} \in \mathcal{S} \mid f_1(\mathbf{q}) < h\} \neq \emptyset$ and $\{\mathbf{q} \in \mathcal{S} \mid f_2(\mathbf{q}) < h\} \neq \emptyset$. Let $\mathcal{W}_1, \mathcal{W}_2 \subseteq \mathcal{S}$ be the isolines (connected components of the level sets) of f_1 and f_2, containing $\mathbf{p}$ (Figure 3.9 illustrates the used notation). As f_1, f_2 are isoline-cut (Definition 28), each of $\mathcal{W}_1$ and $\mathcal{W}_2$, cuts $\mathcal{S}$ in two disjoint parts $\mathcal{S}_1', \mathcal{S}_1''$ and $\mathcal{S}_2', \mathcal{S}_2''$, with $\mathcal{S}_1'$ and $\mathcal{S}_2'$ containing $\mathbf{m}_1$, respectively $\mathbf{m}_2$. From Property 21 we have $\forall \mathbf{u}_1 \in \mathcal{S}_1''$, $f_1(\mathbf{u}_1) > h$ and $\forall \mathbf{u}_2 \in \mathcal{S}_2''$, $f_2(\mathbf{u}_2) > h \implies$ for $\mathcal{S}_1'' \cup \mathcal{S}_2''$, f_m contains only values larger than h, and all values of f_m smaller than h are in $\mathcal{S}_1' \cap \mathcal{S}_2'$. For $\mathcal{P} = (\mathcal{S}_1' \cap \mathcal{S}_2')$ we have:

(2.a) $\mathcal{P} = \emptyset$ (Figure 3.10.a)
As f_1, f_2 are iso-convex (Definition 29), the above implies that $\mathcal{W}_1 \cap \mathcal{W}_2 = \{\mathbf{p}\}$ ($\mathcal{W}_1$ and $\mathcal{W}_2$ lie on different sides of a supporting line at $\mathbf{p}$) $\implies \mathcal{S}_1'' \cup \mathcal{S}_2'' \cup \{\mathbf{p}\} = \mathcal{S}$ and $\mathbf{p}$ is the single local/global minimum of f_m.

(2.b) $\mathcal{P} \neq \emptyset$ (Figures 3.10.b and 3.10.c)
As f_1, f_2 are iso-convex, $\mathcal{P} \cap \mathcal{U}_\varepsilon(\mathbf{p}) \neq \emptyset$ (see Theorem 1, strictly convex implies connected). As f_1, f_2 are continuous and $\{\mathbf{m}_1, \mathbf{m}_2\} \subseteq \mathcal{P}$, $\exists \mathbf{q} \in \mathcal{P} \cap \mathcal{U}_\varepsilon(\mathbf{p})$ s.t. $f_1(\mathbf{q}) < f_1(\mathbf{p})$ and $f_2(\mathbf{q}) < f_2(\mathbf{p})$, thus $\mathbf{p}$ cannot be a local (global) minimum of f.

□

Properties iv. and v.:

For **Properties iv.** (isoline-cut) and **v.** (iso-convex), consider an isoline $\mathcal{W}$ of f_m corresponding to a value h. We can distinguish the following two cases:

(1.) $\mathcal{W}$ consists entirely of an isoline $\mathcal{W}_1$ of f_1 or $\mathcal{W}_2$ of f_2
Assume $\mathcal{W} = \mathcal{W}_1 \implies \mathcal{W}$ has the same properties as $\mathcal{W}_1$ i.e. isoline-cut (Definition 28) and iso-convex (Definition 29).

(2.) $\mathcal{W}$ consists of parts of two isolines $\mathcal{W}_1$ of f_1 or $\mathcal{W}_2$ of f_2
As both $\mathcal{W}_1$ and $\mathcal{W}_2$ are convex (f_1, f_2 are iso-convex), $\mathcal{W}$ is also convex (Theorem 1), thus f_m is iso-convex too.
Consider a path $\pi(\mathbf{q}, \mathbf{q}')$ with $\mathbf{q} \in \mathcal{S}_1'' \cup \mathcal{S}_2''$ and $\mathbf{q}' \in \mathcal{S}_1' \cap \mathcal{S}_2'$ thus $f_m(\mathbf{q}) > h$ and $f_m(\mathbf{q}') < h$. Assuming that $\mathcal{W}$ is not a cut $\exists \pi(\mathbf{q}, \mathbf{q}') \subset \mathcal{S}$ s.t. π does not contain any point of $\mathcal{W}$ i.e.

$\pi \cap \mathcal{W} = \emptyset$. As $\mathcal{W}$ is connected and f_m is continuous, $\exists \mathbf{p} \in \pi$ s.t. $f_m(\mathbf{p}) = h$. As both f_1, f_2 are isoline-cut, we have $\mathbf{p} \in \mathcal{W}_1$ if $f_m(\mathbf{p}) = f_1(\mathbf{p})$ or $\mathbf{p} \in \mathcal{W}_2$ if $f_m(\mathbf{p}) = f_2(\mathbf{p})$, but this implies that $\mathbf{p} \in \mathcal{W}$ which contradicts the assumption that $\pi \cap \mathcal{W} = \emptyset \implies f_m$ is isoline-cut.

Putting it all together:

In the previous we have shown that:

Property 22. *If $f_1, f_2 : \mathcal{S} \to \mathbb{R}$ possess Properties i.–v. (Page 38) i.e. continuous on the simply connected domain $\mathcal{S}$, they have a single minimum, they are isoline-cut and iso-convex, then the function $f_m : \mathcal{S} \to \mathbb{R}$, $f_m = \max\{f_1, f_2\}$ has the same properties.*

Now we can formulate the property we were aiming at:

Property 23. *The geodesic center of a planar, simply connected, continuous shape $\mathcal{S}$, using $d_\varepsilon^{\mathcal{S}}$, is a single point i.e. $C(\mathcal{S}) = \{\mathbf{c}\}$, $\mathbf{c} \in \mathcal{S}$ i.e. $ECC(\mathcal{S})$ has a unique global minimum.*

Proof. At the beginning of the subsection we have rewritten Equation 3.1 (definition of the eccentricity transform) using functions instead of points as:

$$ECC(\mathcal{S}) = \max\{D^{\mathcal{S}}(\mathbf{y}_1), \max\{D^{\mathcal{S}}(\mathbf{y}_2), \max\{D^{\mathcal{S}}(\mathbf{y}_3), \dots\}\}\},\ \cup\{\mathbf{y}_i\} = \mathcal{S}$$

The geodesic distance function $D_\varepsilon^{\mathcal{S}}(\mathbf{y})$ for a simply connected planar shape $\mathcal{S}$ is continuous, and has a single minimum $[D_\varepsilon^{\mathcal{S}}(\mathbf{y})](\mathbf{p}) = 0$ at $\mathbf{p} = \mathbf{y}$. The isolines are either closed or a set of disconnected lines, a case in which both endpoints of the isolines are located on the boundary of $\mathcal{S}$. As $\mathcal{S}$ is simply connected two points on different sides of an isoline $\mathcal{W}$ cannot be connected by a path not intersecting $\mathcal{W}$ and thus $D_\varepsilon^{\mathcal{S}}$ is also isoline cut (Definition 28). The isolines $\mathcal{W}$ are connected arcs of decreasing radii, with the centers of the arcs on the geodesic from the isoline points to the source point (marker) $\mathbf{y} \implies D_\varepsilon^{\mathcal{S}}(\mathbf{y})$ is iso-convex (Definition 29).

As $D_\varepsilon^{\mathcal{S}}(\mathbf{y})$ possesses Properties **i.–v.** (Page 38), applying Property 22 in the ECC definition above, shows that $ECC(\mathcal{S})$ possesses the same properties, specifically Property **iii.** i.e. having a single minimum. □

3.4.4 Properties Contradicting Early Intuition

Property 24. *The longest geodesics of a shape (path defining the diameter) do not necessarily pass through any center point.*

Figure 3.16 presents the eccentricity transform of a triangle, an example that illustrates the above.

Property 25. *Eccentric paths with different orientation and different length can pass through a single point.*

The points on the small diameter of an ellipse are a good example for this (Section 3.5.5). Eccentric paths going from one side to the other pass through the small diameter. Another example is the center of a 2D disk. Eccentric paths in all directions pass through it. This property helps understand why we cannot, like in the case of the distance transform, compute the eccentricity transform in a single pass by associating a single value to each processed point.

Property 26. *The set of eccentric points $E(\mathcal{S})$ of a shape is not necessarily equal to its set of geodesic ends i.e. not all eccentric points are local maxima of the eccentricity transform.*

Property 8 states that all geodesic ends are located on $\partial\mathcal{S}$, which is not always the case for eccentric points (Property 11). An example of a simply connected shape where not all eccentric points are geodesic ends is the ellipse (Figure 3.18).

Property 27. *Not all eccentric point clusters will have at least one local maxima in any $D^{\mathcal{S}}(\mathbf{y})$, $\mathbf{y} \in \mathcal{S}$. In other words, there exists a point $\mathbf{y} \in \mathcal{S}$ and an eccentric point cluster $\mathcal{P} \subset E(\mathcal{S})$ of $\mathcal{S}$ s.t. no $\mathbf{p} \in \mathcal{P}$ is a local maxima of $D^{\mathcal{S}}(\mathbf{y})$.*

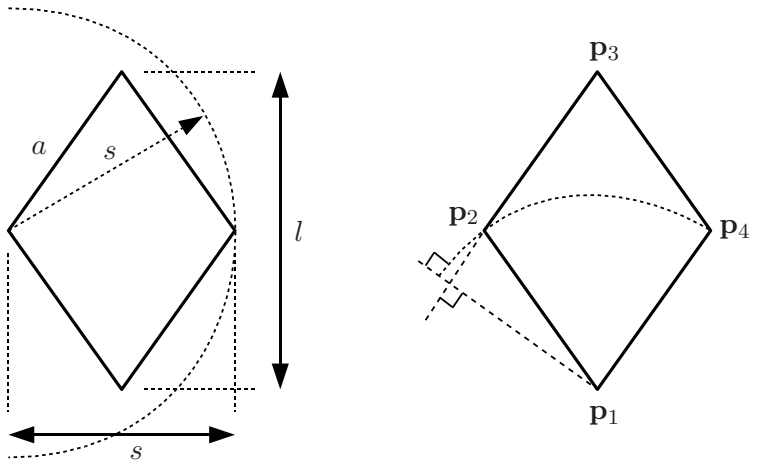


Figure 3.11: Eccentric point $\mathbf{p}_2$ is not a local maximum in $D^{\mathcal{S}}(\mathbf{p}_1)$.

Proof. An example for this is a rhombus (diamond) with the short diagonal longer than a side ($s > a$, in Figure 3.11). Points $\mathbf{p}_2$ and $\mathbf{p}_4$ are eccentric points for each other, as $s > a$. The isolines of the geodesic distance function $D^{\mathcal{S}}(\mathbf{y})$ of a convex shape are circles centered at the point $\mathbf{y}$, limited to the points of the shape. As $s < l$, the angle at $\mathbf{p}_2$ is greater than $\pi/2$, and

the normal to the supporting line of the side $\mathbf{p}_2\mathbf{p}_3$ passing through $\mathbf{p}_1$ is outside the rhombus. All points of $\mathbf{p}_2\mathbf{p}_3$ are outside the circle centered at $\mathbf{p}_1$ with radius a. Also the distances of the points of $\mathbf{p}_2\mathbf{p}_3$ to $\mathbf{p}_1$ are increasing in the direction from $\mathbf{p}_2$ to $\mathbf{p}_3$. Similarly $\mathbf{p}_4$ is not a local maximum in $D^{\mathcal{S}}(\mathbf{p}_1)$. □

Property 27 is relevant for the eccentricity transform algorithms in Section 4.2. Some eccentric points $\mathbf{e} \in E(\mathcal{S})$ (e.g. $\mathbf{p}_1$ in Figure 3.11) are local maxima in any $D^{\mathcal{S}}(\mathbf{y})$, $\mathbf{y} \neq \mathbf{e}$. This makes them very easy to detect. This is partially a local property, as one could pinch the rhombus at point $\mathbf{p}_2$ and make the corner much sharper (non linear deformation) s.t. $\mathbf{p}_2$ is a local maxima in every $D^{\mathcal{S}}(\mathbf{y})$.

3.4.5 Relation to the Hausdorff Distance

The *Hausdorff distance* [Hausdorff 05], named after Felix Hausdorff, measures how far two compact non-empty subsets of a metric space are from each other. (A subset of $\mathbb{R}^n$ is called *compact* if it is closed and bounded.) Let $\mathcal{X}, \mathcal{Y}$ be two compact subsets of a metric space $[\mathcal{M}, d]$, the Hausdorff distance d_H between $\mathcal{X}$ and $\mathcal{Y}$ is:

$$d_H(\mathcal{X}, \mathcal{Y}) = \max\{\sup_{\mathbf{x}\in\mathcal{X}} \inf_{\mathbf{y}\in\mathcal{Y}} d(\mathbf{x}, \mathbf{y}), \sup_{\mathbf{y}\in\mathcal{Y}} \inf_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x}, \mathbf{y})\}.$$

Considering $\mathcal{M} = \mathcal{S}$, we can write the eccentricity of $\mathbf{p} \in \mathcal{S}$ (Equation 3.1):

$$ECC(\mathcal{S}, \mathbf{p}) = \sup_{\mathbf{q}\in\mathcal{S}} d(\mathbf{p}, \mathbf{q}) = \max\{\inf_{\mathbf{q}\in\mathcal{S}} d(\mathbf{p}, \mathbf{q}), \sup_{\mathbf{q}\in\mathcal{S}} d(\mathbf{p}, \mathbf{q})\} = d_H(\{\mathbf{p}\}, \mathcal{S}).$$

3.5 Eccentricity of Simple Shapes

This section gives the eccentricity transform of basic one and two dimensional shapes, using d_ε (Table 3.1 shows an overview). These examples visualize and help to understand the properties enumerated before. Aspects regarding decomposing the shape are considered because they could lead to even more efficient, divide-et-impera based algorithms for the eccentricity transform (Section 4.5). For each shape, the position of the center and eccentric points are presented and decomposition of the shape is considered.

Table 3.1: Simple shapes presented in Section 3.5

shape	dimension	comments
open curve	1D	simply connected
closed curve	1D	not simply connected
triangular region	2D	convex, finite number of eccentric points
circular region (disk)	2D	convex
elliptic region	2D	convex, 2 eccentric point clusters
rectangle	2D	convex, no 1-to-1 association to eccentric point
elongated	2D	convex, more complex
elongated - bent	2D	not convex, symmetric and not symmetric
elongated general	2D	not convex, not symmetric
circular with hole	2D	not simply connected, symmetric and not symmetric

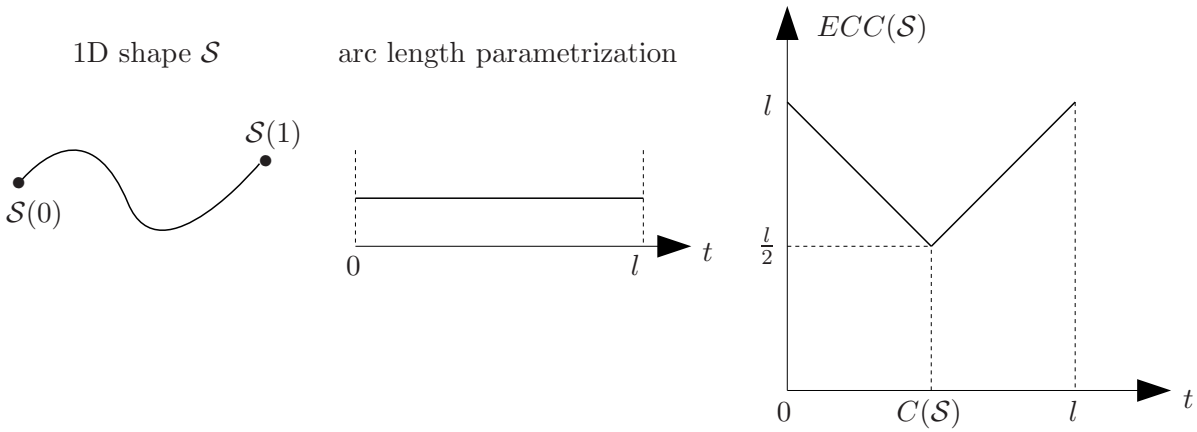


Figure 3.12: 1D eccentricity: open curve

3.5.1 1D eccentricity - open curve

Let $\mathcal{S}$ be a non self-intersecting one dimensional manifold in $\mathbb{R}^n$, and let $\mathcal{S}(t)$ be the arc length parametrization[2] of $\mathcal{S}$ with $t \in [0, l]$, $l = \lambda(\mathcal{S})$, s.t. $\mathcal{S}(0)$ and $\mathcal{S}(l)$ are the two end points i.e. the only two points having a single neighbor in $\mathcal{S}$. Figure 3.12 shows an example.

The shape has two eccentric points $E(\mathcal{S}) = \{\mathcal{S}(0), \mathcal{S}(l)\}$ and its center is the middle of the curve $C(\mathcal{S}) = \{\mathcal{S}(l/2)\}$. The points $0 \leqslant t \leqslant l/2$ have $\mathcal{S}(l)$ as their eccentric point. The points

[2]For the arc length parametrization, the value of the parameter t coincides with the length of the path from $\mathcal{S}(0)$ to $\mathcal{S}(t)$, in $\mathcal{S}$.

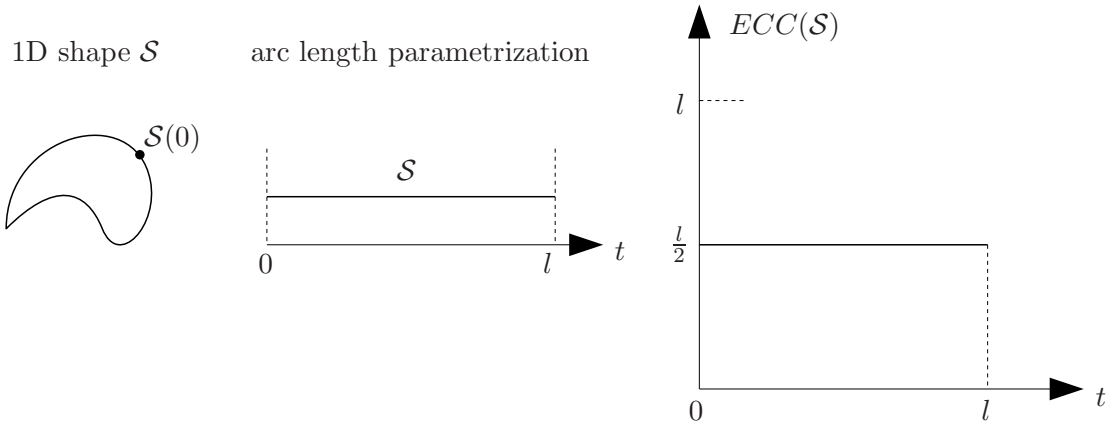


Figure 3.13: 1D eccentricity: closed curve

$l/2 \leqslant t \leqslant l$ have $\mathcal{S}(0)$ as their eccentric point. The eccentricity transform of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} d(\mathbf{p}, \mathcal{S}(l)) = l - t & \text{if } t \leqslant l/2 \\ d(\mathbf{p}, \mathcal{S}(0)) = t & \text{otherwise} \end{cases}$$

where $t \in [0, l]$ s.t. $\mathbf{p} = \mathcal{S}(t)$ and l is the length of $\mathcal{S}$.

3.5.2 1D eccentricity - closed curve

Let $\mathcal{S}$ be a closed one dimensional manifold in $\mathbb{R}^n$, and let $\mathcal{S}(t)$ be the arc length parametrization of $\mathcal{S}$ with $t \in [0, l]$, l is the length of $\mathcal{S}$, s.t. $\mathcal{S}(0)$ and $\mathcal{S}(l)$ is the same point of $\mathcal{S}$ and $\mathcal{S}(t)$ is a continuous and linear mapping from the interval $[0, l]$ to $\mathcal{S}$. A simple example of such a shape is the circle with $t = \phi r$ where ϕ is the angle in radians for the classical parametric definition using sinus and cosine, and r is the radius. Figure 3.13 shows a more general example.

All points of $\mathcal{S}$ are eccentric points, $E(\mathcal{S}) = \mathcal{S}$, and the center coincides with $\mathcal{S}$, $C(\mathcal{S}) = \mathcal{S}$. There are two different paths connecting every point with its eccentric point, which is located on the "opposite" side of the curve. The eccentricity transform of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \frac{l}{2}$$

where l is the length of $\mathcal{S}$ and $\mathbf{p} \in \mathcal{S}$.

3.5.3 Triangular region

Let $\mathcal{S}$ be a triangular region with corners $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. The 3 perpendicular bisectors divide the triangle into 6 or 4 regions (see Figure 3.14) depending on whether the triangle is obtuse or

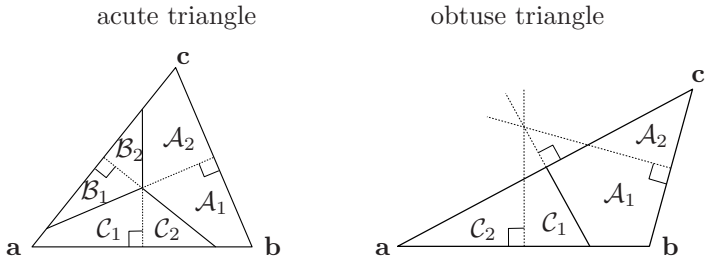


Figure 3.14: Regions delineated by the perpendicular bisectors: regions labeled $\mathcal{A}, \mathcal{B}, \mathcal{C}$ have $\mathbf{a}, \mathbf{b}$ respectively $\mathbf{c}$ as their eccentric point.

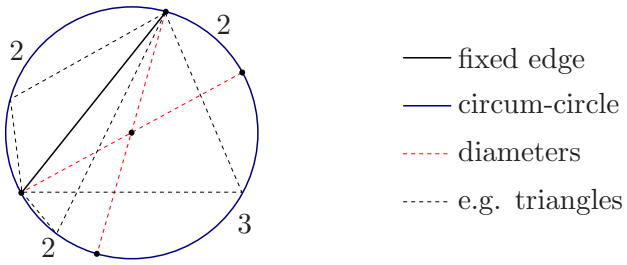


Figure 3.15: Fixing two corners and moving one: how many eccentric points are there?

not i.e the circum-center lies outside or within the triangle (Figure 3.15). All points inside the same region have the same single eccentric point, and the eccentricity value is the distance to that point. The points on the perpendicular bisectors incident to regions with different eccentric points, have both points as eccentric (e.g. points on the common boundary between a region marked $\mathcal{C}$ and one marked $\mathcal{A}$ have both $\mathbf{a}$ and $\mathbf{c}$ as their eccentric points). If inside the triangle, the circum-center has all three corners as eccentric points. The ECC decomposition contains 3 or 2 regions, depending on whether the triangle is acute or obtuse. The regions are bounded by the perpendicular bisectors incident to regions having different eccentric points. The eccentricity transform of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} d_\varepsilon(\mathbf{p}, \mathbf{a}) & \text{if } \mathbf{p} \in \mathcal{A}_1 \cup \mathcal{A}_2 \\ d_\varepsilon(\mathbf{p}, \mathbf{b}) & \text{if } \mathbf{p} \in \mathcal{B}_1 \cup \mathcal{B}_2 \\ d_\varepsilon(\mathbf{p}, \mathbf{c}) & \text{if } \mathbf{p} \in \mathcal{C}_1 \cup \mathcal{C}_2 \end{cases}$$

or more general

$$ECC(\mathcal{S}, \mathbf{p}) = \max\{d_\varepsilon(\mathbf{p}, \mathbf{a}), d_\varepsilon(\mathbf{p}, \mathbf{b}), d_\varepsilon(\mathbf{p}, \mathbf{c})\}$$

The isolines of the eccentricity transform of a triangle are: a single closed curve made of 3 arcs, or a maximum of 3 disconnected circle arcs. Figure 3.16 shows the isolines of the two triangles in Figure 3.14.

Figure 3.16: Eccentricity transform for the triangles in Figure 3.14. (Gray values are distances modulo a constant.)

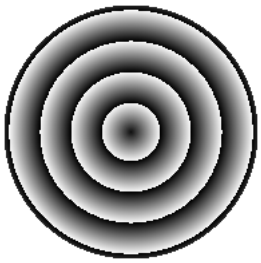


Figure 3.17: Eccentricity transform of disk. (Gray values are distances modulo a constant.)

3.5.4 Circular region (disk)

Let $\mathcal{S}$ be a 2D disk with radius r and center $\mathbf{c}$ i.e $\mathbf{p} \in \mathcal{S} \Longleftrightarrow d_\varepsilon(\mathbf{p}, \mathbf{c}) \leqslant r$. The eccentric points of $\mathcal{S}$ are the points on the circle i.e. $E(\mathcal{S}) = \mathcal{C}(r, \mathbf{c}) \subset \mathcal{S}$. Each point $\mathbf{e} \in \mathcal{C}(r, \mathbf{c})$ of the circle is an eccentric point for the points on the line segment $\mathbf{cp}$ where $\mathbf{p}$ is the point where the diameter containing $\mathbf{e}$ intersects the circle the second time. The eccentric points of the center $\mathbf{c}$ are all points of the circle $\mathcal{C}(r, \mathbf{c})$, $E(\mathbf{c}) = E(\mathcal{S})$, and all eccentric paths are normal to the circle at the respective eccentric point. The isolines of the eccentricity of $\mathcal{S}$ are circles centered at $\mathbf{c}$ and the eccentricity transform of $\mathcal{S}$ is (Figure 3.17):

$$ECC(\mathcal{S}, \mathbf{p}) = r + d_\varepsilon(\mathbf{p}, \mathbf{c})$$

For the disk, the eccentricity transform and the distance transform (Section 2.2.3) are linearly dependent:

$$ECC(\mathcal{S}, \mathbf{p}) = 2r - DT(\mathcal{S}, \mathbf{p})$$

3.5.5 Elliptic region

In this subsection we will use the following notation: the set of points $(x, y) \in \mathcal{S}$ such that $x > 0$, is denoted $\mathcal{S}_r$ and called the *right part* of $\mathcal{S}$, whereas the set of points $(x, y) \in \mathcal{S}$ such that $x < 0$,

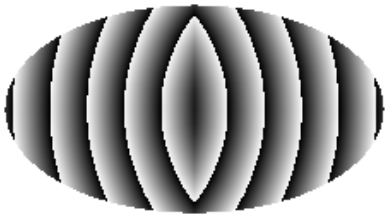


Figure 3.18: Eccentricity transform of an ellipse. (Gray values are distances modulo a constant.)

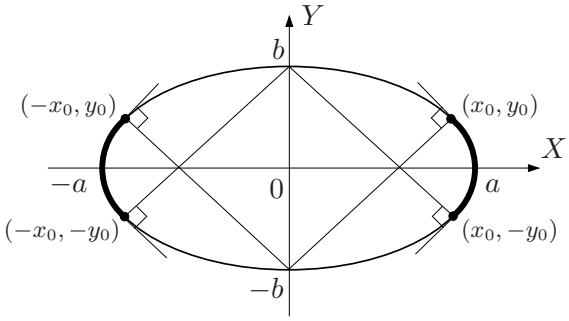


Figure 3.19: Eccentric point clusters of the ellipse (shown with thick line).

is denoted $\mathcal{S}_l$ and called the *left part* of $\mathcal{S}$.

Ellipse recalls

The elliptical curve of points (x, y) around the origin, axes parallel to the coordinate system axes, and with parameters $a > 0$ and $b > 0$ is defined by:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \tag{3.3}$$

In point (x, y) it has a tangent direction $(\dot{x}, \dot{y})$ satisfying

$$\frac{x\dot{x}}{a^2} + \frac{y\dot{y}}{b^2} = 0. \tag{3.4}$$

Figure 3.18 shows the eccentricity transform of an ellipse.

Bounding extremal points

We consider an elliptical region $\mathcal{S}$ centered the origin, defined by $\frac{x^2}{a^2} + \frac{y^2}{b^2} \leqslant 1$. In the following, we assume that $a > b$, thus the *small axis* of $\mathcal{S}$ is the segment $[-b, b]$, and the *long axis* of $\mathcal{S}$ is the segment $[-a, a]$.

The eccentric point(s) of any point $(x, y) \in \mathcal{S}$, are on the boundary of $\mathcal{S}$ (Property 12). We now provide two additional properties regarding eccentric points on an elliptical region.

Property 28. (Eccentric paths orthogonal to the tangent) *Let (x_e, y_e) be an eccentric point for $(x, y) \in \mathcal{S}$, $\mathcal{S}$ simply connected with smooth $\partial\mathcal{S}$. Then, the tangent to $\mathcal{S}$ at the point (x_e, y_e) is orthogonal to the geodesic ν, from (x, y) to (x_e, y_e)*

Proof. Suppose that the tangent is not orthogonal to the geodesic ν. Let $\mathcal{L}^t$ be the tangent at (x_e, y_e) to ν, and let $\mathcal{L}$ be the line passing through (x_e, y_e) that is orthogonal $\mathcal{L}^t$. As $\partial\mathcal{S}$ is smooth, and $\mathcal{L}$ is not the tangent to $\partial\mathcal{S}$ at (x_e, y_e), $\mathcal{L}$ will be partially contained in $\mathcal{S}$. Then there exists a point $(x'_e, y'_e) \in \mathcal{L} \cap \mathcal{S}$, $(x'_e, y'_e) \neq (x, y)$ in the neighborhood of (x_e, y_e), which is farther away from (x, y) than (x_e, y_e). This would contradict the fact that (x_e, y_e) is an eccentric point for (x, y). □

The ellipse is a simply connected convex shape and the smaller axis is an axis of symmetry. For the ellipse $\mathcal{S}$ in this section, we can choose $\mathcal{S}_1 = \mathcal{S}_l$ and $\mathcal{S}_2 = \mathcal{S}_r$ and thus, due to Property 17 all points of $\mathcal{S}_l$ have their eccentric points int $\mathcal{S}_r$ and vice versa.

We compute the eccentric points of $(0, b)$ and $(0, -b)$. This allows to partition the ellipse into 4 subsegments alternating the property of being eccentric or not (see Figure 3.19). Let us consider the line $\mathcal{L}$ that goes through the point $(0, -b)$ and crosses the ellipse with orthogonal tangent at point (x_0, y_0), such that $x_0 \geqslant 0$ and $y_0 \geqslant 0$. This line $\mathcal{L}(t)$ is defined by:

$$\begin{cases} x &= t\dot{y_0} \\ y &= -b - t\dot{x_0} \end{cases} \tag{3.5}$$

As $(x_0, y_0) \in \mathcal{L}$, we can deduce from Equation 3.5 that:

$$\begin{cases} t_0 &= \frac{x_0}{\dot{y_0}} \\ y_0 &= -b - \frac{\dot{x_0}}{\dot{y_0}} x_0 \end{cases}$$

From Equation 3.4, we obtain $-\frac{\dot{x_0}}{\dot{y_0}} = \frac{y_0 a^2}{x_0 b^2}$. Using Equation 3.5, we obtain $y_0 = -b + x_0 \frac{y_0 a^2}{x_0 b^2} = -b + \frac{y_0 a^2}{b^2}$, so

$$y_0 = \frac{b^3}{a^2 - b^2}$$

The x-coordinate is then determined using the ellipse formula (Equation 3.3):

$$x_0^2 = a^2 \left(1 - \frac{b^4}{(a^2 - b^2)^2}\right)$$

Symmetry considerations deliver the eccentric point for $(0, -b)$ with $x_0 < 0$ and $y_0 \geqslant 0$, and the two eccentric points for the point $(0, b)$.

One can directly deduce that any point (x_c, y_c) of the ellipse, s.t. $x_c^2 < x_0^2$ has normals that do not intersect the segment $[-b, b]$. Thus, according to Property 17, all these points cannot be eccentric points for any point inside the ellipse. Hence the points (x_0, y_0), $(x_0, -y_0)$, $(-x_0, -y_0)$, $(-x_0, y_0)$ partition the ellipse into 4 subsegments alternating the property of being extremal or not.

Eccentric lines through the smaller axis

We show how to efficiently compute the eccentricity transform of an elliptical region $\mathcal{S}$, by considering separately $\mathcal{S}_l$ and $\mathcal{S}_r$. Using Property 28, we first show how to compute the eccentricity of all the points of the small axis.

Let $\mathbf{p} = (0, \mu b)$, $-1 \leqslant \mu \leqslant 1$ be a point on the small axis, and let $\mathbf{e} = (x_e, y_e)$ be its eccentric point in $\mathcal{S}_r$. Using Property 28, the points (x, y) of the line $\mathcal{L}(t)$ defined by $\mathbf{pe}$ satisfy:

$$\begin{cases} x &= t\dot{y_e} \\ y &= \mu b + t\dot{x_e}. \end{cases}$$

In particular, $\mathbf{e} \in \mathcal{L}$, so we have $x_e = t_e\dot{y_e}$ and $y_e = \mu b + t_e\dot{x_e}$. Thus we deduce that $t_e = \frac{x_e}{\dot{y_e}}$ and $y_e = \mu b + \frac{\dot{x_e}}{\dot{y_e}}x_e = \frac{\mu b^3}{b^2 - a^2}$.

The x-coordinate is then determined using the ellipse Equation 3.3

$$x_e^2 = a^2(1 - \frac{b^4\mu^2}{(b^2 - a^2)^2}).$$

The eccentricity of any point $(0, \mu b), -1 \leqslant \mu \leqslant 1$ of the small axis, can directly be computed by the above formula using a, b and μ. The direction to their eccentric point is also known and can be stored in each point.

Note that, for any $\mathbf{p}$ on the smaller axis, the segments connecting it to $(0, b)$ and to $(0, -b)$ are also orthogonal to the tangent of the ellipse at the respective points, but are shorter than the ones considered above and thus not relevant for the eccentricity. When the ellipse is a circle, the points (x_0, y_0) and $(-x_0, y_0)$, respectively $(x_0, -y_0)$ and $(-x_0, -y_0)$ coincide.

As the ellipse is a convex shape, the eccentric path from any point of an elliptic region to its eccentric point is a straight line. Moreover, from Property 17, we know that the eccentric point $\mathbf{e}$ of any point $\mathbf{p}_l \in \mathcal{S}_l$ is in $\mathcal{S}_r$. Thus, for computing the eccentricity of the point $\mathbf{p}_l$, we have to find the point $\mathbf{p}$ of the small axis, such that the direction $\overrightarrow{\mathbf{p}_l\mathbf{p}}$ is the same as the direction

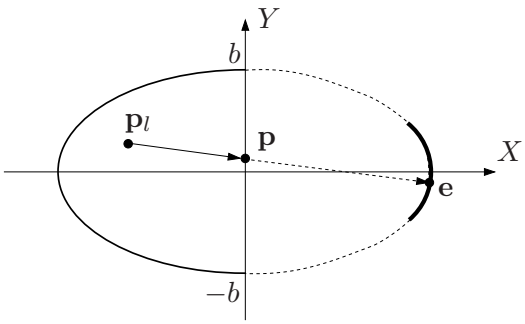


Figure 3.20: Efficient eccentricity transform based on decomposition.

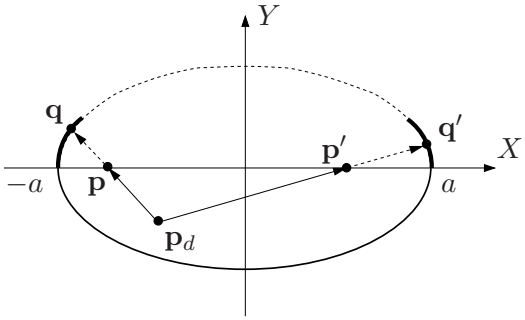


Figure 3.21: Ellipse decomposition along the bigger axis: more than one line orthogonal to the ellipse tangent at the point of intersection can go through one point.

stored in $\mathbf{p}$ (Figure 3.20).

Eccentric lines through the bigger axis

We have shown that it is possible to decompose an ellipse $\mathcal{S}$ along its smaller axis, to efficiently compute the eccentricity $ECC(\mathcal{S})$. This is not the case when decomposing $\mathcal{S}$ into $\mathcal{S}_u$ (up) and $\mathcal{S}_d$ (down) along the bigger axis $[-a, a]$ because (Figure 3.21 shows the used notation):

- the eccentric points $\mathbf{e}$ of any point $\mathbf{p} = (\mu a, 0)$, $-1 \leqslant \mu \leqslant 1$ are either $(-a, 0)$ or $(a, 0)$, which is not helpful for deducing $ECC(\mathcal{S})$ based on the ECC of the parts;
- even if we associate to each point $\mathbf{p} = (\mu a, 0)$, $-1 \leqslant \mu \leqslant 1$, the point $\mathbf{q} \in \partial\mathcal{S}_u \setminus [-a, a]$ s.t. the segment $\mathbf{pq}$ is orthogonal to the tangent at $\mathbf{q}$, $\exists \mathbf{p}_d \in \mathcal{S}_d \setminus [-a, a]$, with at least two points $\mathbf{q}$ and $\mathbf{q}'$ in $\partial\mathcal{S}_u$ s.t. $\mathbf{p}_d\mathbf{q}$ and $\mathbf{p}_d\mathbf{q}'$ are orthogonal to the tangent at $\mathbf{q}$ respectively $\mathbf{q}'$. A one to one mapping between points $\mathbf{p}_d \in \mathcal{S}_d$ and points $\mathbf{p}$ on the bigger axis cannot be made just based on the angle of the segments $\mathbf{p}_d\mathbf{p}$, $\mathbf{p}_d\mathbf{p}'$. The distances $d(\mathbf{p}, \mathbf{q})$ and $d(\mathbf{p}', \mathbf{q}')$ have also to be considered.

Figure 3.22: Eccentricity transform of a rectangle. (Gray values are distances modulo a constant.)

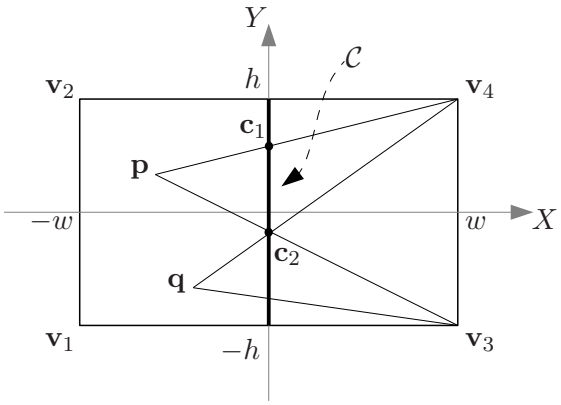


Figure 3.23: Eccentric paths inside a rectangle - cutting along a medial line

3.5.6 Rectangle

Let $\mathcal{S}$ be a rectangle with sides parallel to the coordinate system axes and lengths $2w$ and $2h$ (see Figure 3.23). It is centered at the origin. The four corners of the rectangle $\mathbf{v}_1 = (-w, -h)$, $\mathbf{v}_2 = (-w, h)$, $\mathbf{v}_3 = (w, -h)$, $\mathbf{v}_4 = (w, h)$ make up the set of eccentric points of $\mathcal{S}$. The eccentricity transform of $\mathcal{S}$ is (Figure 3.22):

$$ECC(\mathcal{S}, \mathbf{p}) = \max\{d_\varepsilon(\mathbf{p}, \mathbf{v}_i)\}, \;\; i = 1 \ldots 4.$$

In each quadrant V_i the isolines are made out of arcs of circles centered at the eccentric point $\mathbf{v}_i$. Let $r = \min\{d((-w, h), (0, -h)),\; d((-w, h), (w, 0))\}$ and $R = \max\{d((-w, h), (0, -h)),\; d((-w, h), (w, 0))\}$. The isolines can be:

- if $d(\mathbf{p}, E(\mathbf{p})) \leqslant r$, a closed curve made out of four arcs;
- if $r < d(\mathbf{p}, E(\mathbf{p})) \leqslant R$, two disconnected lines each made out of two arcs;
- if $d(\mathbf{p}, E(\mathbf{p})) > R$, four disconnected arcs.

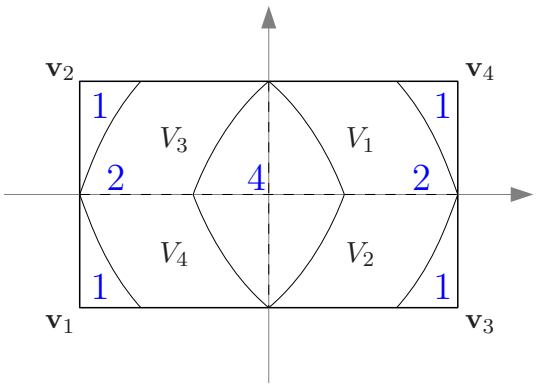


Figure 3.24: ECC of rectangle. V_1, V_2, V_3, V_4 regions associated to $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ respectively $\mathbf{v}_4$. Number of circle arcs in each isoline (blue).

The rectangle $\mathcal{S}$ can be decomposed in two subparts $\mathcal{S}_l$ and $\mathcal{S}_r$, along the cut (segment) $\mathcal{C}$ from $(0, -h)$ to $(0, h)$. The corners $\mathbf{v}_1$, $\mathbf{v}_2$ respectively $\mathbf{v}_3, \mathbf{v}_4$ are the eccentric points of all the points $\mathbf{p} \in \mathcal{C}$. Compared to decomposing an ellipse along the smaller axis, in the case of the rectangle we cannot associate to each point of $\mathcal{C}$ a single pair made of a direction and distance, because eccentric paths with more than one orientation can pass through the same point of the cut[3] (e.g. $\mathbf{c}_2$ in Figure 3.23). In this case, to each point $\mathbf{c} \in \mathcal{C}$ we associate two pairs of distances and directions, connecting $\mathbf{c}$ to the corners $\mathbf{v}_1$, $\mathbf{v}_2$ respectively $\mathbf{v}_3, \mathbf{v}_4$. Thus, for any $\mathbf{p} \in \mathcal{S}_l$, $\exists \mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ s.t. $\mathbf{c}_1 \in \mathbf{pv}_4$ and $\mathbf{c}_2 \in \mathbf{pv}_3$, similarly for $\mathbf{p} \in \mathcal{S}_r$ $\exists \mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ s.t. $\mathbf{c}_1 \in \mathbf{pv}_2$ and $\mathbf{c}_2 \in \mathbf{pv}_1$. Then we can rewrite the eccentricity transform of $\mathcal{S}$ as:

$$ECC(\mathcal{S}, \mathbf{p}) = \max\{d(\mathbf{p}, \mathbf{c}_1) + ECC(\mathcal{S}, \mathbf{c}_1), d(\mathbf{p}, \mathbf{c}_2) + ECC(\mathcal{S}, \mathbf{c}_2)\}.$$

A one to one association between points on the cut and eccentric paths cannot be made. For each point on the line segment cut, two candidates for eccentric points exist.

Because the number of eccentric points is finite, like in the case of the triangle (Section 3.5.3), the rectangle can be divided by the perpendicular bisectors of the sides in a finite number of regions (ECC decomposition, Section 3.3.1) s.t. all points in a region have the same eccentric point (Figure 3.24).

3.5.7 Elongated - straight (rounded rectangle)

Let $\mathcal{S}$ be an elongated shape obtained by gluing two opposite sides of a rectangle $\mathcal{R}$ with two halves of ellipses $\mathcal{E}_l$ and $\mathcal{E}_r$. Let us assume that: the width of the rectangle is $2w > 0$, and its

[3]This is similar to decomposing the ellipse along the bigger axis (Section 3.5.5).

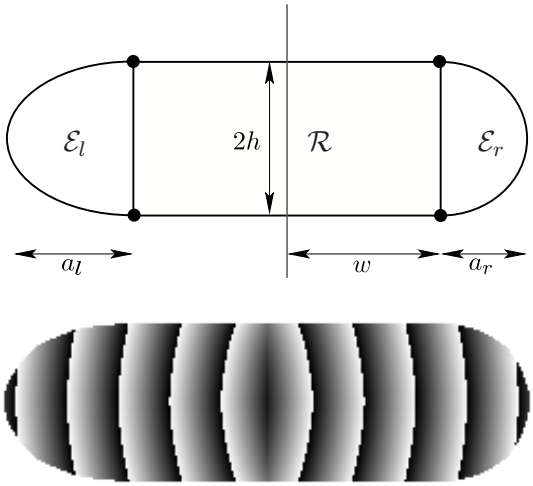


Figure 3.25: Elongated shape formed of two half ellipses $\mathcal{E}_l, \mathcal{E}_r$ and a rectangle $\mathcal{R}$, and its eccentricity transform. (Gray values are distances modulo a constant.)

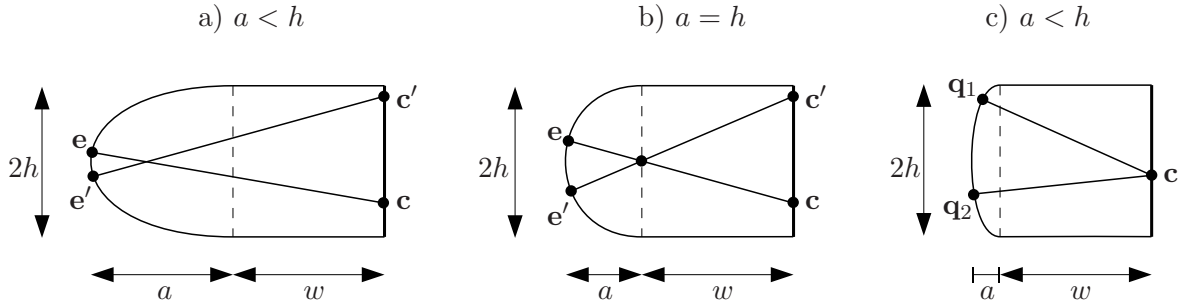


Figure 3.26: Elongated, ellipse cut along: a) smaller axis, b) is circle, c) bigger axis.

height is $2h > 0$, with $w > h$; $\mathcal{E}_l$ is the left half ellipse, defined by the parameters a_l and h; $\mathcal{E}_r$ is the right half ellipse, defined by the parameters a_r and h (Figure 3.25).

Shape $\mathcal{S}$ symmetric along the vertical axis:

Let us assume that $a_l = a_r = a$ and decompose $\mathcal{S}$ along the cut $\mathcal{C} = (0, -\mu h)$, $1 \leqslant \mu \leqslant 1$. For any point $\mathbf{p} \in \mathcal{S}_l$, its eccentric point $\mathbf{e}$ is in $\mathcal{S}_r$ (Property 17), and the segment $\mathbf{pe}$ is orthogonal to the ellipse tangent at $\mathbf{e}$ (Property 28).

A first formulation of the eccentricity of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \max\{d(\mathbf{p}, \mathbf{q})\},\ \mathbf{q} \in \partial\mathcal{S} \text{ s.t. } \mathbf{pq} \perp \text{ tangent to } \partial\mathcal{S} \text{ at } \mathbf{q}$$

Depending on the relation of a and h (Figure 3.26) we can have the following cases:

$a > h$ **:** The ellipses where cut along the shorter diameter. As $w > 0$, for every point $\mathbf{c} \in \mathcal{C}$ there are two points $\mathbf{e}_l \in \partial\mathcal{S} \cap \mathcal{E}_l$ and $\mathbf{e}_r \in \partial\mathcal{S} \cap \mathcal{E}_r$ s.t. $\mathbf{ce}_l$ respectively $\mathbf{ce}_r$ are normal to the tangent to $\partial\mathcal{S}$ at $\mathbf{e}_l$ respectively $\mathbf{e}_r$ (Figure 3.26.a). Both $\mathbf{e}_l$ and $\mathbf{e}_r$ are eccentric points for $\mathbf{c}$ and can be found through a reasoning similar to the one in Section 3.5.5 (ECC of ellipse). Like in in the case of the ellipse, we can compute the eccentricity and the eccentric path orientation for all points of $\mathcal{C}$ separately in $\mathcal{S}_l$ and $\mathcal{S}_r$ i.e. $d(\mathbf{c}, \mathbf{e}_l)$ and $d(\mathbf{c}, \mathbf{e}_r)$ (which are actually equal, as $a_l = a_r$) and the orientations of $\mathbf{ce}_l$ and $\mathbf{ce}_r$. The eccentricity transform can then be computed as:

$$ECC(\mathcal{S}, \mathbf{p}) = d(\mathbf{p}, \mathbf{c}) + \begin{cases} ECC(\mathcal{S}_r, c) & \text{if } \mathbf{p} \in \mathcal{S}_l \\ ECC(\mathcal{S}_l, c) & \text{otherwise} \end{cases}$$

$a = h$ **:** $\mathcal{E}_l$ and $\mathcal{E}_r$ are two circle halves and all eccentric paths go through one of the two circle centers (Figure 3.26.b).

$a < h$ **:** The ellipses $\mathcal{E}_l$ and $\mathcal{E}_r$ correspond to ellipses that have been cut along their bigger axis (see Section 3.5.5). In this case, there exist points $\mathbf{p} \in \mathcal{S}_l$ which have two points $\mathbf{q}_1, \mathbf{q}_2 \in \partial\mathcal{S} \cap \mathcal{E}_r$, such that $\mathbf{pq}_i$, $i \in \{1, 2\}$, is normal to the ellipse tangent at $\mathbf{q}_i$. As such points can exist also on the cut $\mathcal{C}$, we cannot associate each point with a single direction and distance based only on the orthogonality with the ellipse tangent. Like in the case of the ellipse decomposed along the bigger axis, and the case of the rectangle, we need to take the maximum of the two distances (Figure 3.26.c).

Shape $\mathcal{S}$ not symmetric along the vertical axis:

If $a_l \neq a_r$, $\mathcal{S}$ is no longer symmetric along the shorter, vertical axis. The shape cannot be decomposed by a line segment s.t. for any point $\mathbf{p} \in \mathcal{S}$ all its eccentric points are contained in the part not containing $\mathbf{p}$. To overcome this problem we can *decompose as above, along a line segment, and take for each point the maximum between the eccentricity computed separately on the part containing the point and the one obtained by using the decomposition.*

3.5.8 Elongated - bent

Let $\mathcal{S}$ be an elongated shape obtained by gluing two rectangles, each with a half of a circle $\mathcal{E}_l$ and $\mathcal{E}_r$ glued on one side (Figure 3.27). The rectangle edges opposite to the circle halves are joint by a circular arc. The width of the two rectangles is $w_l > 0$, $w_r > 0$ and their height is $2h > 0$; $\mathcal{E}_l$ is the left circle half, defined by the parameter h (radius of circle); $\mathcal{E}_r$ is the right

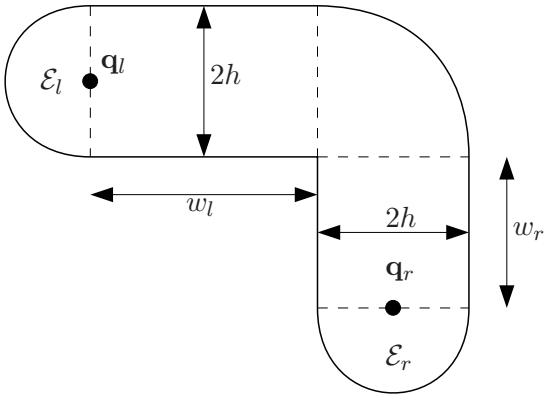


Figure 3.27: Elongated-bent shape: two half circles $\mathcal{E}_l$, $\mathcal{E}_r$, two rectangles, and a circle arc.

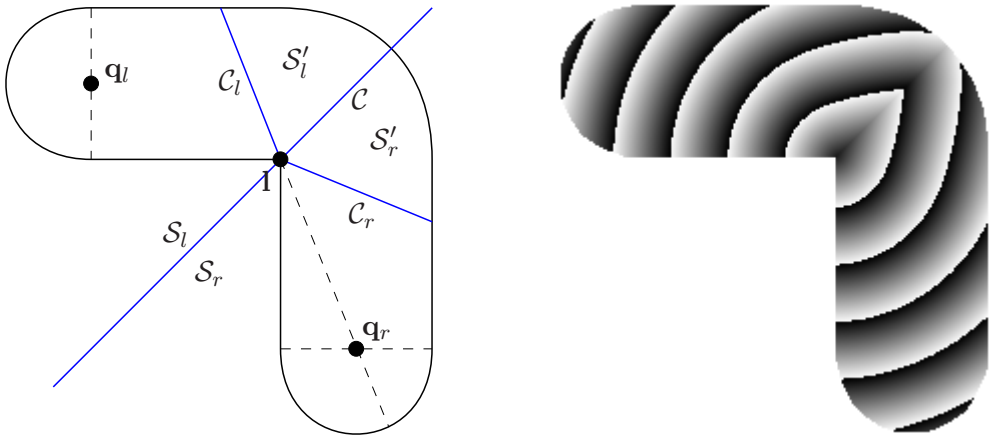


Figure 3.28: Symmetric bent shape (left) and its eccentricity transform (right). (Gray values are distances modulo a constant.)

(bottom) circle half, defined by the same parameter h. For the shape to be an elongated one, we need $w_l \geqslant 2h$ and $w_r \geqslant 2h$ (Figure 3.27).

All eccentric points lie on the two circle halves $\mathcal{E}_l$ and $\mathcal{E}_r$, and all eccentric paths are orthogonal to $\partial\mathcal{S}$ at the respective eccentric points. As $\mathcal{E}_l$ and $\mathcal{E}_r$ are circle parts, all eccentric paths go through the circle centers $\mathbf{q}_l$ or $\mathbf{q}_r$ (Section 3.5.4 gives the eccentricity transform of a circle, which illustrates the previous). We consider the following two cases:

Symmetric shape S ($w_l = w_r$):

Assume that $w_l = w_r$ and decompose $\mathcal{S}$ along the cut $\mathcal{C}$ (blue in Figure 3.28). $\mathcal{C}$ is an axis of symmetry for $\mathcal{S}$, going 45 degrees up-right from l, across the "articulation". $\mathcal{C}$ divides $\mathcal{S}$ in $\mathcal{S}_l$ (the left side) and $\mathcal{S}_r$ (the right/bottom side). All points in $\mathcal{S}_l$ will have their eccentric point

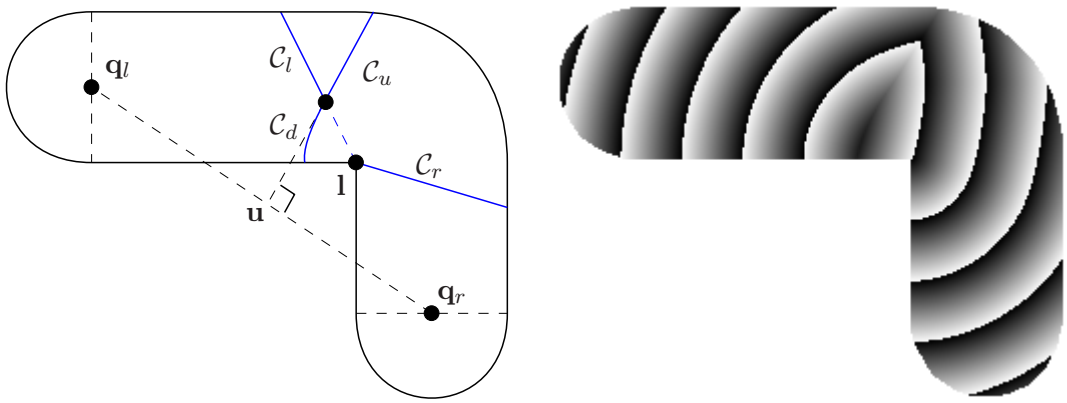


Figure 3.29: Non-symmetrically bent elongated shape (left): two half circles, two rectangles and a circle arc. Its eccentricity transform (right). (Gray values are distances modulo a constant.)

in $\mathcal{E}_r$ and vice versa. $\mathcal{C}_l$ is the segment connecting $\mathbf{l}$ with the upper horizontal line of the left rectangle and is obtained by prolongating the segment $\mathbf{q}_r\mathbf{l}$ until it leaves the shape. In the same way, $\mathcal{C}_r$ is obtained for prolongating the line segment $\mathbf{q}_l\mathbf{l}$. Lets denote with $\mathcal{S}'_l$ and $\mathcal{S}'_r$ the part of $\mathcal{S}$ between $\mathcal{C}_l$ and $\mathcal{C}$ respectively $\mathcal{C}_r$ and $\mathcal{C}$. The eccentricity of $\mathcal{S}$ is then:

$$ECC(\mathcal{S},\mathbf{p}) = h + \begin{cases} d(\mathbf{p},\mathbf{l}) + d(\mathbf{l},\mathbf{q}_r) & \text{if } \mathbf{p} \in \mathcal{S}_l - \mathcal{S}'_l \\ d(\mathbf{p},\mathbf{q}_r) & \text{if } \mathbf{p} \in \mathcal{S}'_l \\ d(\mathbf{p},\mathbf{q}_l) & \text{if } \mathbf{p} \in \mathcal{S}'_r \\ d(\mathbf{p},\mathbf{l}) + d(\mathbf{l},\mathbf{q}_l) & \text{if } \mathbf{p} \in \mathcal{S}_r - \mathcal{S}'_r \end{cases}$$

The geodesic center of $\mathcal{S}$ is $C = \{\mathbf{l}\}$ and all points of the cut $\mathcal{C}$ have two eccentric points. If the angle between the two rectangles is decreased, the angle between $\mathcal{C}_l$ and $\mathcal{C}_r$ increases until $\mathcal{S}_l = \mathcal{S}'_l$ and $\mathcal{S}_r = \mathcal{S}'_r$ i.e. all points of $\mathcal{S}$ are directly visible from their corresponding eccentric points. When $\mathcal{C}_l$ and $\mathcal{C}_r$ fall over $\partial\mathcal{S}$, $\mathcal{S}$ is a non-convex shape with all eccentric paths being straight lines.

Non-symmetric shape $\mathcal{S}$ ($w_l \neq w_r$):

For the case of the non symmetrical bent shape $\mathcal{S}$ ($w_l > w_r$ in Figure 3.27) we define the following (Figure 3.29). Like in the previous case, $\mathcal{C}_l$ and $\mathcal{C}_r$ (blue in Figure 3.29) are the line segments from $\mathbf{l}$ to $\partial\mathcal{S}$ obtained by prolongating the line segments $\mathbf{q}_r\mathbf{l}$ and $\mathbf{q}_l\mathbf{l}$. The point $\mathbf{u}$ is located in the middle of the line segment $\mathbf{q}_l\mathbf{q}_r$ i.e. $(\mathbf{u} = (\mathbf{q}_l + \mathbf{q}_r)/2)$. The segment $\mathcal{C}_u$ (blue in

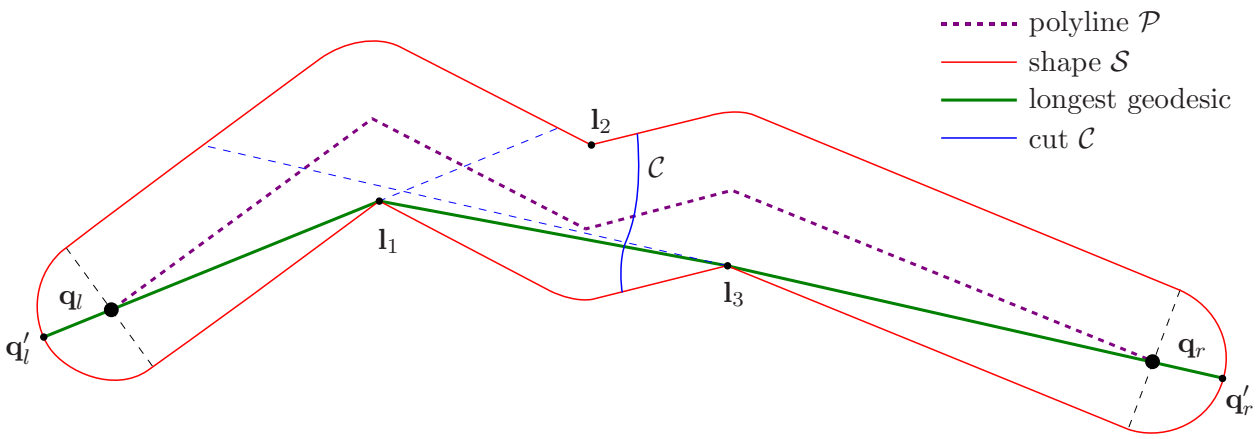


Figure 3.30: Elongated shape obtained by dilating a polyline.

Figure 3.29) is defined by the intersection of $\mathcal{C}_l$ with the perpendicular bisector[4] of the segment $\mathbf{q}_l\mathbf{q}_r$. $\mathcal{C}_d$ (blue in Figure 3.29) is part of the hyperbola[5] defined by $\mathbf{q}_l$ and $\mathbf{l}$, and the difference $d(\mathbf{l}, \mathbf{q}_r)$. We can decompose $\mathcal{S}$ along $\mathcal{C} = \mathcal{C}_d \cup \mathcal{C}_u$ in $\mathcal{S}_l$ (left of $\mathcal{C}$) and $\mathcal{S}_r$ (right/bottom of $\mathcal{C}$). All points in $\mathcal{S}_l$ will have their corresponding eccentric point in $\mathcal{E}_r$ and vice versa. The eccentricity of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = h + \begin{cases} d(\mathbf{p}, \mathbf{l}) + d(\mathbf{l}, \mathbf{q}_r) & \text{if } \mathbf{p} \text{ on the left side of } \mathcal{C}_d \text{ and } \mathcal{C}_l \\ d(\mathbf{p}, \mathbf{q}_r) & \text{if } \mathbf{p} \text{ between } \mathcal{C}_l \text{ and } \mathcal{C}_u \\ d(\mathbf{p}, \mathbf{q}_l) & \text{if } \mathbf{p} \text{ between } \mathcal{C} = \mathcal{C}_d \cup \mathcal{C}_u \text{ and } \mathcal{C}_r \\ d(\mathbf{p}, \mathbf{l}) + d(\mathbf{l}, \mathbf{q}_l) & \text{if } \mathbf{p} \text{ below } \mathcal{C}_r \end{cases}$$

The geodesic center of $\mathcal{S}$ is the point where $\mathcal{C}$ and $\mathbf{q}_l\mathbf{l}$ intersect[6], and all points of the cut $\mathcal{C}$ have two eccentric points. If the angle between the two rectangles is decreased, the angle between $\mathcal{C}_l$ and $\mathcal{C}_r$ increases until all points of $\mathcal{S}$ are directly visible from their eccentric points.

3.5.9 Elongated - general (n rounded rectangles)

Let $\mathcal{S}$ be an elongated shape obtained by dilating a polyline $\mathcal{P}$ with a circle of radius $h > 0$ (Figure 3.30). We consider a simply connected shape $\mathcal{S}$ (no holes), with the longest geodesic of $\mathcal{S}$ passing through the endpoints of $\mathcal{P}$, $\mathbf{q}_l$ and $\mathbf{q}_r$. Let $\mathbf{l}_i$ denote the concave points of

[4]The perpendicular bisector of a line segment is the straight line perpendicular to the segment and passing through its midpoint.

[5]A hyperbola is the locus of points where the difference in the distance to two fixed points (called the foci) is constant.

[6]Intersection of $\mathcal{C}_d$ and $\mathbf{q}_l\mathbf{l}$ for the shape in Figure 3.29.

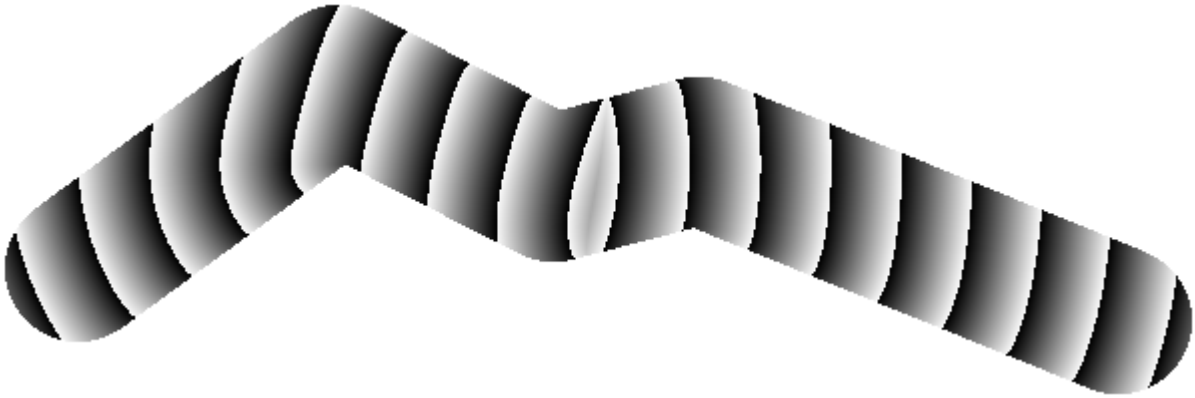


Figure 3.31: Eccentricity transform of elongated shape in Figure 3.30. (Gray values are distances modulo a constant.)

the boundary of $\mathcal{S}$ ($\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ in Figure 3.30). They are induced by corner points of $\mathcal{P}$, on the side where the angle of $\mathcal{P}$ is less than 180 degrees. Geodesics in $\mathcal{S}$ are polylines of the form $\nu(\mathbf{p}_1, \mathbf{p}_2) = (\mathbf{p}_1, \mathbf{l}_{k_1}, \ldots, \mathbf{l}_{k_n}, \mathbf{p}_2)$, where $(\mathbf{l}_{k_1}, \ldots, \mathbf{l}_{k_n})$ is an ordered subset of the set of concave points of the boundary.

The longest geodesic of $\mathcal{S}$ is the polyline $\mathcal{P}' = (\mathbf{q}'_l, \mathbf{l}_{j_1}, \ldots, \mathbf{l}_{j_n}, \mathbf{q}'_r)$. The points $\mathbf{q}'_l, \mathbf{q}'_r$ are on the boundary of $\mathcal{S}$ (Property 12) and the polyline $\mathcal{P}'$ is orthogonal to the tangent to $\partial\mathcal{S}$ at $\mathbf{q}'_l$ and $\mathbf{q}'_r$[7]. The first and last segments of $\mathcal{P}'$ contain the endpoints of $\mathcal{P}$ i.e. $\mathbf{q}_l \in (\mathbf{q}'_l, \mathbf{l}_{j_1})$ and $\mathbf{q}_r \in (\mathbf{l}_{j_n}, \mathbf{q}'_r)$. The longest geodesic of the shape in Figure 3.30 is $(\mathbf{q}'_l, \mathbf{l}_1, \mathbf{l}_2, \mathbf{q}'_r)$.

The eccentric points of $\mathcal{S}$ are located on $\partial\mathcal{S}$ (Property 12), on the two half-circles that have $\mathbf{q}_l$ or $\mathbf{q}_r$ as the closest points of $\mathcal{P}$. All eccentric paths pass through either $\mathbf{q}_l$ or $\mathbf{q}_r$. We can decompose $\mathcal{S}$ along a cut $\mathcal{C}$ s.t. $\forall \mathbf{p} \in \mathcal{C}$, $d(\mathbf{p}, \mathbf{q}_l) = d(\mathbf{p}, \mathbf{q}_r)$. The cut $\mathcal{C}$ is formed of one or more hyperbolas. All points on one side of the cut $\mathcal{C}$ will have their eccentric points on the other side. The eccentricity transform of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = h + \begin{cases} d(\mathbf{p}, \mathbf{q}_r) & \text{if } \mathbf{p} \text{ is on the side of } \mathcal{C} \text{ that contains } \mathbf{q}_l \\ d(\mathbf{p}, \mathbf{q}_l) & \text{if } \mathbf{p} \text{ is on the side of } \mathcal{C} \text{ that contains } \mathbf{q}_r \end{cases}$$

The geodesic center of $\mathcal{S}$ is the point in the middle of $\mathcal{P}'$ and is the intersection of the cut $\mathcal{C}$ with $\mathcal{P}'$, i.e. $\{\mathbf{c}\} = \mathcal{P}' \cap \mathcal{C}$.

[7] A straight line passing through the center of a circle is orthogonal to the tangents to the circle at the intersection points.

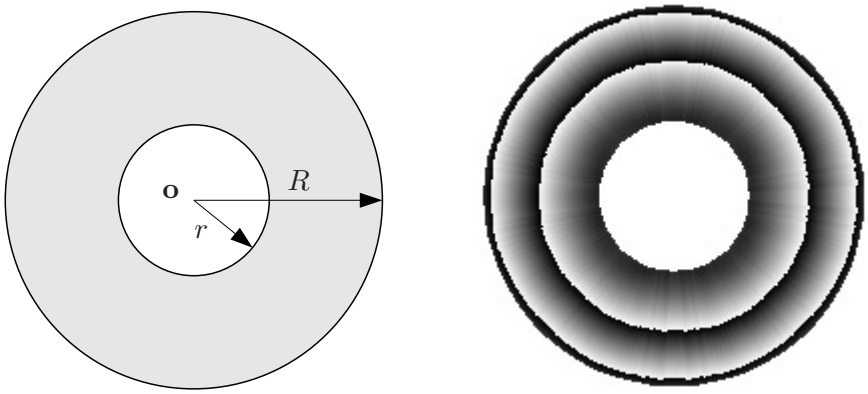


Figure 3.32: Disk with one circular hole in the middle (left) and its eccentricity transform (right). (Gray values are distances modulo a constant.)

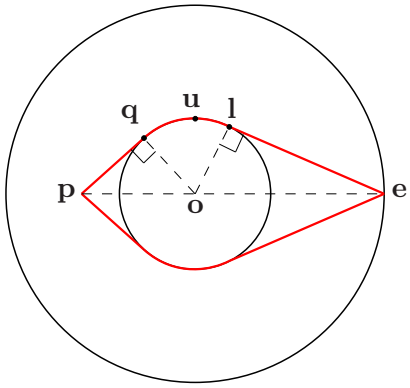


Figure 3.33: Eccentric paths in a disk with one circular hole in the middle.

3.5.10 1 hole - disk with circular hole in the middle

Let $\mathcal{S}$ be a disk of radius R with a circular hole of radius $r \leqslant R$ in the middle (Figure 3.32). The set of eccentric points of $\mathcal{S}$ is the outer circle $\mathcal{C}(R, \mathbf{o})$ and the geodesic center is the inner circle $\mathcal{C}(r, \mathbf{o})$.

Any point $\mathbf{p} \in \mathcal{S}$ has a single eccentric point $\mathbf{e} \in \mathcal{C}(R, \mathbf{o})$ and two eccentric paths $\nu_1, \nu_2 \in \Pi(\mathbf{p}, \mathbf{e})$ going on each side of the hole (Figure 3.33). Each eccentric path is made out of three parts: the line segment $\mathbf{pq}$, with $\lambda(\mathbf{pq}) \geqslant 0$ and $\mathbf{pq} \perp \mathbf{qo}$, the arc $\mathbf{ql}$, and the line segment $\mathbf{le}$ with $\lambda(\mathbf{le}) \geqslant 0$ and $\mathbf{le} \perp \mathbf{lo}$. The points $\mathbf{p}$, $\mathbf{o}$ and $\mathbf{e}$ are collinear. The following relations exist for the straight parts:

$$\lambda(\mathbf{pq}) = \sqrt{d(\mathbf{p}, \mathbf{o})^2 - r^2}$$

$$\lambda(\mathbf{le}) = \sqrt{R^2 - r^2}$$

For the circle arc we have:

$$\widehat{\mathbf{poq}} = \arctan(\frac{\lambda(\mathbf{pq})}{r})$$

$$\widehat{\mathbf{eol}} = \arctan(\frac{\lambda(\mathbf{le})}{r})$$

$$\lambda(\mathbf{ql}) = r(\pi - (\widehat{\mathbf{poq}} + \widehat{\mathbf{eol}}))$$

The eccentricity transform of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \lambda(\mathbf{pq}) + \lambda(\mathbf{ql}) + \lambda(\mathbf{le}) \tag{3.6}$$

The geodesic center

Consider the point $\mathbf{u} \in \mathcal{C}(r, \mathbf{o})$ s.t. $\mathbf{uo} \perp \mathbf{pe}$ (remember that $\mathbf{o} \in \mathbf{pe}$). The length of $(\mathbf{ql})$ can be rewritten as $\lambda(\mathbf{ql}) = \lambda(\mathbf{qu}) + \lambda(\mathbf{ul})$, with $\lambda(\mathbf{qu})$ depending only on $\mathbf{p}$ and r, and $\lambda(\mathbf{ul})$ on r and R (fixed for $\mathcal{S}$):

$$ECC(\mathcal{S}, \mathbf{p}) = \lambda(\mathbf{pq}) + \lambda(\mathbf{qu}) + \lambda(\mathbf{ul}) + \lambda(\mathbf{le}) \tag{3.7}$$

Property 29. *The geodesic center of $\mathcal{S}$ is $\mathcal{C}(r, \mathbf{o}) \subset \partial\mathcal{S}$ i.e. the eccentricity $ECC(\mathcal{S}, \mathbf{p})$ of a point $\mathbf{p}$ is minimum iff $\mathbf{p} \in \mathcal{C}(r, \mathbf{o})$.*

Proof. We show that the function in Equation 3.7 has its minimum when $\mathbf{p} = \mathbf{q}$.
The length $\lambda(\mathbf{qu}) = r(\pi/2 - \widehat{\mathbf{poq}})$ and Equation 3.7 can be rewritten as:

$$ECC(\mathcal{S}, \mathbf{p}) = g(l) = l + r(\frac{\pi}{2} - \arctan\frac{l}{r}) + k,$$

where $l = \lambda(\mathbf{pq}) = d(\mathbf{p}, \mathbf{q})$ and $k = \lambda(\mathbf{ul}) + \lambda(\mathbf{le})$ does not depend on $\mathbf{p}$ respectively on l. For $g : \mathbb{R}^+ \to \mathbb{R}$ we have:

$$g'(l) = 1 - r(\frac{1}{1 + (\frac{l}{r})^2})\frac{1}{r} = 1 - \frac{r^2}{r^2 + l^2} = \frac{l^2}{r^2 + l^2},$$

where $\arctan'(l) = 1/(1 + l^2)$ and we have applied the *chain rule*[8]. $g'(l) > 0, \forall l > 0 \implies g$ has a single minimum at $l = 0 \iff \lambda(\mathbf{pq}) = d(\mathbf{p}, \mathbf{q}) = 0$. As d is a metric, $d(\mathbf{p}, \mathbf{q}) = 0 \iff \mathbf{p} = \mathbf{q}$ $\implies \mathbf{p} \in \mathcal{C}(r, \mathbf{o})$. □

This example shows that the geodesic center can be more complex than a single point. Also, because each point $\mathbf{p}$ defines a separation line behind the obstacle $\mathcal{C}(r, \mathbf{o})$ a decomposition of $\mathcal{S}$

[8] $(f \circ g)'(x) = f'(g(x))g'(x)$ [Marsden 86].

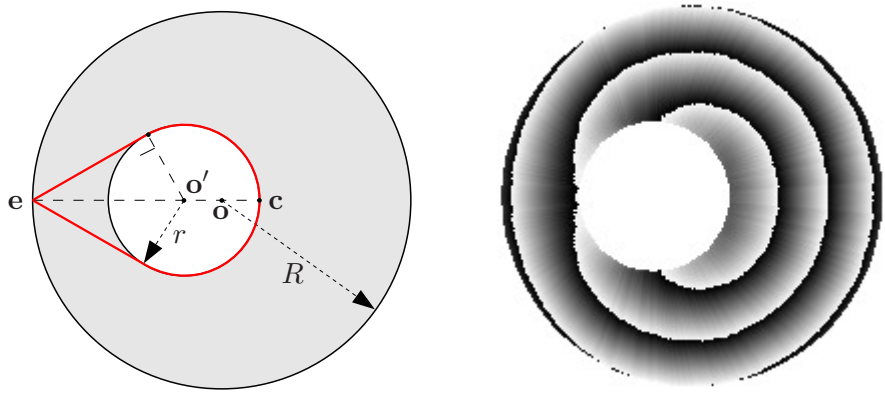


Figure 3.34: Geodesic center $\mathbf{c}$ for disk with one circular hole ($0 < d(\mathbf{o}, \mathbf{o}') < \min(r, R - r)$) (left). Its eccentricity transform (right). (Gray values are distances modulo a constant.)

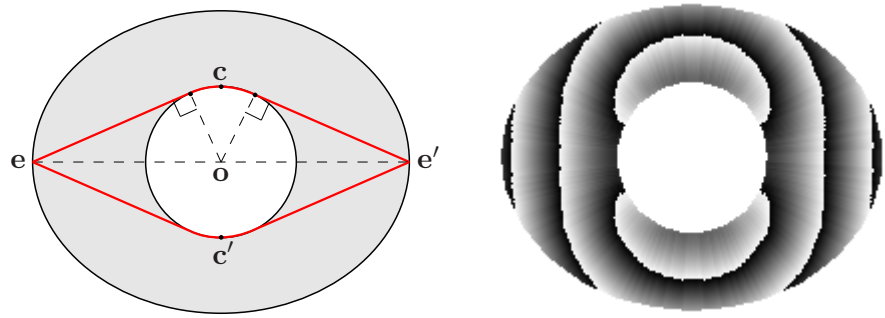


Figure 3.35: Geodesic center $\{\mathbf{c}, \mathbf{c}'\}$ for ellipse with one circular hole in the middle (left) and its eccentricity transform (right). (Gray values are distances modulo a constant.)

in two parts by associating an angle and a distance to each point on the cut is no longer enough to compute the eccentricity of the parts (like it was in the case of the simply connected shapes). Eccentric paths with the same direction (angle) go through the same point on the cut and lead to different eccentric points. For example, there are many points $\mathbf{p}$ for which the eccentric paths go over $\mathbf{u}$ and are tangent to $\mathcal{C}(r, \mathbf{o})$, but end in different eccentric points $\mathbf{e}$.

Changing the form of the shape $\mathcal{S}$

To see how stable the geodesic center configuration (circle) above is, we look at the following deformations for the shape $\mathcal{S}$:

$\mathcal{S}'$: moving the hole s.t. the two circles $\mathcal{C}(R, \mathbf{o})$ and $\mathcal{C}(r, \mathbf{o}')$ no longer have the same center ($0 < d(\mathbf{o}, \mathbf{o}') < \min(r, R - r)$) (Figure 3.34). $\mathcal{S}'$ has one single center point $C(\mathcal{S}') = \{\mathbf{c}\}$ which is the point of $\mathcal{C}(r, \mathbf{o}')$ closest[9] to $\mathbf{o}$. The single eccentric point corresponding to $\mathbf{c}$

[9]Euclidean distance in the plane in which $\mathcal{S}'$ is embedded.

is located on the circle $\mathcal{C}(R, \mathbf{o})$ s.t. $\mathbf{o}' \in \mathbf{ec}$, and $\mathbf{c}$ has two eccentric paths.

$\mathcal{S}''$: stretching $\mathcal{C}(R, \mathbf{o})$ s.t. it becomes an ellipse $\mathcal{E}(R_a, R_b, \mathbf{o})$, $R_a > R_b$ (Figure 3.35). $\mathcal{S}''$ has two center points $\mathbf{c}, \mathbf{c}'$ located on $\mathcal{C}(r, \mathbf{o})$ at the intersection with the shorter axis (length $2R_b$; vertical axis in Figure 3.35). Both center points have the same two eccentric points $E(\mathbf{c}) = E(\mathbf{c}') = \{\mathbf{e}, \mathbf{e}'\}$.

When considering the two deformations above, the configuration for the geodesic center as a circle is just a transition between the configurations with one and two center points - one center point (dis)appears.

Conclusion

This section gave the eccentricity transform of basic one and two dimensional shapes, using d_ε (Table 3.1 shows an overview). For each shape, the position of the center and eccentric points haven been given and decomposition of the shape was considered. Aspects regarding decomposing could lead to even more efficient, divide-et-impera based algorithms for the eccentricity transform (Section 4.5). The topology of the shapes plays an important role.

3.6 Robustness Experiments

Robustness is the capability to "perform without failure under a wide range of conditions" [Mer 03]. In the case of an image transform it is said to be robust if the changes in the output are minimal for typical changes in the input. This section presents experiments considering the robustness of the eccentricity transform with respect to acquisition (Salt and Pepper noise and minor segmentation errors), and deformation of the shape through articulation.

3.6.1 Robustness against Salt and Pepper noise

Some metrics (e.g. d_4, d_8) have the property that even for convex shapes without holes, more than one geodesic exists between two points (cases exist where they only share the end points). When using these metrics, one noisy pixel is not enough to change the geodesic distance between two points. One pixel could "block" only one of the geodesics, making it longer, but other shortest paths will keep their original length.

Valid for all metrics is the fact that geodesic distance between points far away is perturbed less than for points nearby (the additional length introduced by the need to "go around" the noisy pixel is a smaller fraction of the longer path than of the shorter). For comparison, this section considers the distance transform (DT) (Section 2.2.3) as it is the most spread image

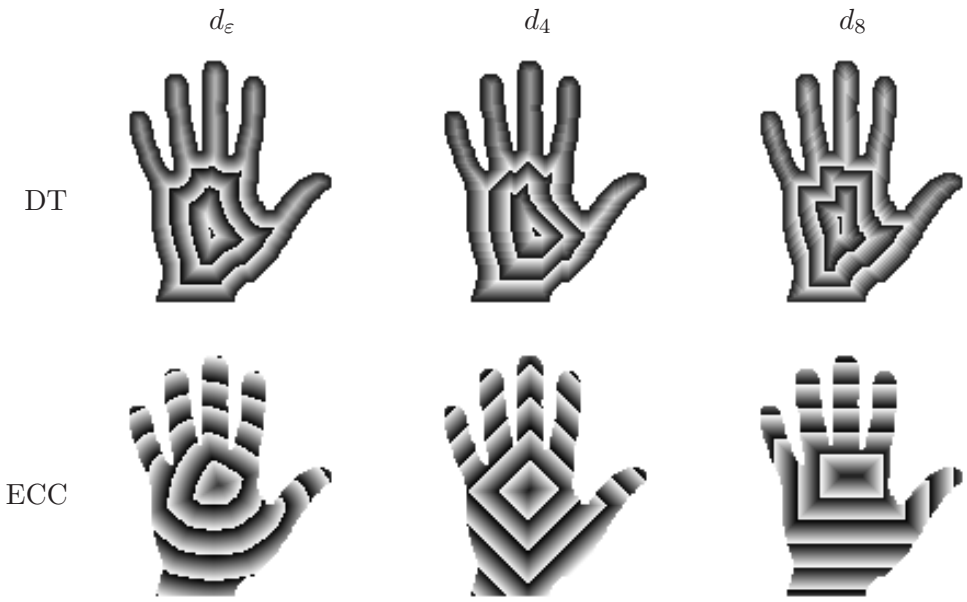


Figure 3.36: Example distance (DT) and eccentricity (ECC) transforms for a shape, using d_ε, d_4, and d_8. (Gray values are distances modulo a constant.)

transform for binary shapes. Figure 3.36 shows the isolines for the eccentricity and distance transforms of the shape of a hand using d_ε, d_4, d_8.

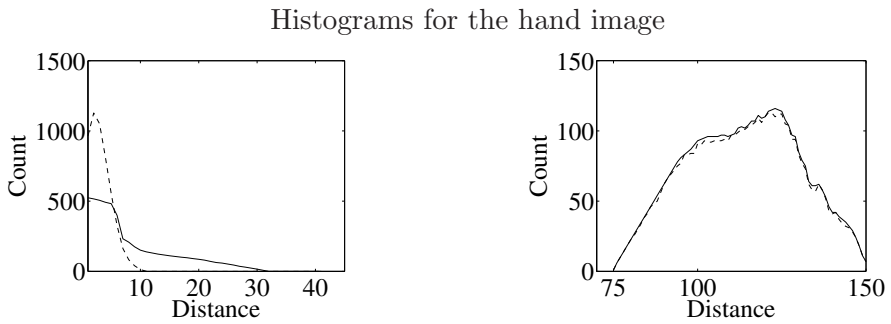
To test the robustness against Salt and Pepper noise (random missing pixels in the shape), we have calculated the eccentricity and distance transforms of the 99 shapes from [Sebastian 04] using d_ε, d_4, and d_8 (for some example shapes see the top row of Table 3.2). We have applied 5% Salt and Pepper like noise to the binary shapes i.e. we have randomly removed pixels from the shape boundary as well as their interior, and calculated the two transforms again.

For each image, neighborhood, and transform, the root mean square error (RMSE) between the values obtained for the original and noisy images are calculated (calculation was done using the values of the pixels part of the shape in both images i.e. noisy pixels are excluded). The produced error measure is the inverse of the mean ratio between the RMSE of the ECC and the DT:

$$Error = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \frac{RMSE(ECC(\mathcal{S}_i))}{RMSE(DT(\mathcal{S}_i))}}$$

where $n = 99$, the number of images. The error is **4.16** for d_ε, **10.22** for d_4, and **16.85** for d_8, meaning that the average change in value of the DT was that many times higher than for ECC.

Figure 3.37 shows the histogram of the eccentricity and distance transforms for one of the images, the hand (original and noisy) using d_4. Also shown is the RMSE between the values



a) Distance transform (DT) with d_4 b) Eccentricity transform (ECC) with d_4

RMSE	8.50	1.16
Max. diff.	26.00	4.00

Figure 3.37: Distance and eccentricity transform histograms, RMSE and Max. Diff. Solid line - original image, dotted line - noisy image.

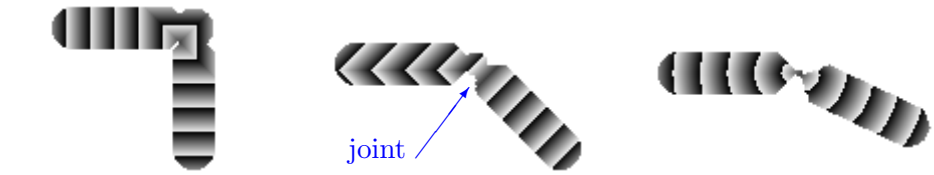
of the transforms for the original and noisy images, and the maximum difference value for each transform. One can see that the error and maximum deviation of the eccentricity transform is much smaller than that of the distance transform. Note that in the case of the noisy image, a valid transform value has been calculated for less pixels. This makes the histogram of the eccentricity transform of the noisy image lie below the histogram of the original one.

3.6.2 Minor segmentation errors

For a few shapes (10) we have simulated segmentation errors and partial occlusion by removing some parts (simulated noise on the boundary of the shape). We have calculated the correlation between the local maxima of the eccentricity transforms of the original and the images with partial occlusion i.e. for each image, original and partially occluded, we have created a matrix where the positions of the eccentricity transform regional maxima were marked with 1, and the rest with 0, and calculated the correlation between the 2 matrices - only maxima that where located inside the partially occluded shape were taken into consideration. Table 3.2 shows these shapes and the obtained correlation values. The lower values obtained for d_ε on the more compact and rounded shapes (e.g. car) are due to the more uniform distribution of eccentric points on the boundary of compact shapes (e.g. the circular region in Section 3.5.4) and eccentric paths being normal to the tangent to the boundary (planar shapes with no holes). The latter property makes the position of the eccentric points depend also on the direction of the source point (e.g. the ellipse in Section 3.5.4). These changes in position happen usually only in a small neighborhood, but still do, and thus the same pixel might not be a local maxima, but another

Table 3.2: Correlation results for local maxima in the eccentricity transform of original (top row) and partially occluded shapes (middle and bottom rows).

	Original Shapes									
	Partially occluded set 1									
d_ε	0.55	0.87	1.00	0.73	0.96	0.82	0.61	1.00	0.87	0.95
d_4	0.73	1.00	1.00	0.96	0.96	1.00	0.77	1.00	1.00	0.95
d_8	0.93	1.00	0.72	0.97	1.00	0.82	1.00	1.00	1.00	0.98
	Partially occluded set 2									
d_ε	0.32	0.66	0.89	0.88	0.65	0.64	0.41	0.89	0.71	0.95
d_4	0.71	0.79	0.97	0.96	0.89	0.97	0.71	0.98	0.87	0.92
d_8	0.48	0.45	0.90	0.96	0.72	0.65	0.98	0.97	0.73	0.98



$\varphi = 90°$, $j = 90\%$, d_8 $\varphi = 135°$, $j = 52\%$, d_4 $\varphi = 155°$, $j = 14\%$, d_ε

Figure 3.38: Example images used for testing the variation under articulated motion.

one located nearby.

3.6.3 Articulation

The length variation of a geodesic path going through an articulation is bounded by the thickness of the shape around the articulation points i.e. the width of the joints [Ling 07] (Figure 3.38 shows example articulated shapes with different joint widths). To test how this works in practice we have done the following experiment.

For a given angle φ and a given joint width j the shape $\mathcal{S}(\varphi, j)$ is created by gluing two identical elongated parts at angle φ and making the joint width j (Figure 3.38).

Table 3.3: Mean and standard deviation of the eccentricity values for the simulated joint.

		Min		**Max**		**Average**	
j	metric	mean	std	mean	std	mean	std
14%	d_ε	70.75	0.42	141.04	0.82	108.66	0.69
	d_4	82.21	4.39	164.00	8.64	126.89	6.36
	d_8	66.84	2.79	133.47	5.67	101.82	4.19
52%	d_ε	69.52	0.99	138.54	2.10	105.34	1.89
	d_4	82.21	4.39	164.00	8.64	124.79	6.45
	d_8	65.58	3.04	130.95	6.18	98.83	4.73
90%	d_ε	68.50	1.73	136.50	3.61	102.39	3.04
	d_4	82.21	4.39	164.00	8.64	122.79	6.45
	d_8	64.47	3.72	128.63	7.52	95.94	5.80

For angles $\varphi = 90° + 5°k$, $0 \leqslant k \leqslant 18$ and joint width $j \in \{14\%, 52\%, 90\%\}$ of the width of the part, we have applied the eccentricity transform and calculated the minimum, maximum, and average eccentricity over all angles. Table 3.3 shows the mean and standard deviation of the 3 values over all widths tested. Note that the values are stable under all joint widths (std $< 5\%$ of the corresponding mean) and get close to constant as the width of the joint decreases (d_ε, d_8). In the case of d_4, the high error even for the smallest joint width is due to the large variation of distances under Euclidean rotation (e.g. for $\varphi = 135°$ the geodesic center is no longer located on the joint, but lower down the part that is oriented 45° from the horizontal axis). Consider a rectangle with sides aligned with the coordinate axis. Two opposite points will have the same d_4 whether considering the diagonal or just a path on the rectangle (boundary).

3.7 Chapter Summary

This chapter gave the formal definition of eccentricity and of the eccentricity transform. Concepts like the geodesic center, diameter, and eccentric points have been introduced. Properties related to these concepts have been given and exemplified on a set of basic shapes. These basic shapes emphasize aspects that have motivated the algorithms and considerations in the following chapter (Computation of the eccentricity transform). Experiments verifying the robustness with respect to noise and articulation of the shape have been considered.

4

Computation of The Eccentricity Transform

This chapter considers computation aspects for the eccentricity transform of discrete shapes. First, error free computation is presented (Section 4.1), followed by efficient approximation algorithms (Sections 4.2, 4.3, and 4.4). A computation using the *Divide and Conquer* principle (see for example [Atallah 98]) is discussed in Section 4.5.

Algorithms 4-8, and the experiments in Section 4.3 have been previously published in [Kropatsch 06, Ion 08d].

The running time complexity of the algorithms in Sections 4.1 and 4.2 has been computed assuming an $O(n \log n)$ complexity for the geodesic distance function (SBDT), independent of the used algorithm (Dijkstra, Fast Marching, and Discrete Circle Propagation in Section 2.2). The term *error free* considers the possible approximation introduced by the ECC algorithms and not by the algorithm employed to compute the geodesic distance function.

Section 3.1 has given a recall of the existing work related to the eccentricity transform, and has mentioned the contexts in which computation has been previously approached.

4.1 The Basic Algorithm - Implementing The Formula

The most intuitive algorithm for computing the eccentricity transform is to strictly follow the formal definition as described in Equation 3.1

$$ECC(\mathcal{S}, \mathbf{p}) = \{\max(d^{\mathcal{S}}(\mathbf{p}, \mathbf{q})) \mid \mathbf{q} \in \mathcal{S}\}$$

i.e. to take each point $\mathbf{p}$ of $\mathcal{S}$ at a time, using the geodesic distance function $D^{\mathcal{S}}$, compute the geodesic distance to all other points ($\mathbf{q}$) and keep the maximum (Algorithm 3).

Instead of first iterating over $\mathbf{p}$, one could also consider first iterating over $\mathbf{q}$ (see Algorithm 4). Line 3 in Algorithm 3 has to also go over all points of $\mathcal{S}$, and the two algorithms (Algorithm 3 and 4) have the same complexity $O(n^2 \log n)$, where n is the number of points (pixels, voxels,

Algorithm 3 $ECC(\mathcal{S})$: Compute ECC of $\mathcal{S}$ - basic.

Input: Discrete shape $\mathcal{S}$.

1: **for all** $\mathbf{p} \in \mathcal{S}$ **do**
2: $D \leftarrow D^{\mathcal{S}}(\mathbf{p})$ /*compute SBDT for $\mathbf{p}$*/
3: $ECC(\mathbf{p}) \leftarrow \max\{D\}$ /*the maximum of D is the value for $ECC(\mathcal{S}, \mathbf{p})$*/
4: **end for**

Output: Distances ECC.

Algorithm 4 $ECC(\mathcal{S})$: Compute ECC of $\mathcal{S}$ - basic, inverse.

Input: Discrete shape $\mathcal{S}$.

1: **for all** $\mathbf{p} \in \mathcal{S}$ **do** $ECC(\mathbf{p}) \leftarrow 0$ /*initialize*/
2: **for all** $\mathbf{q} \in \mathcal{S}$ **do**
3: $D \leftarrow D^{\mathcal{S}}(\mathbf{q})$ /*compute SBDT for $\mathbf{q}$*/
4: **for all** $\mathbf{p} \in \mathcal{S}$ **do** $ECC(\mathbf{p}) \leftarrow \max(ECC(\mathbf{p}), D(\mathbf{q}))$ /*further away than we had before?*/
5: **end for**

Output: Distances ECC.

vertices, etc.) in $\mathcal{S}$ and $O(n \log n)$ is the complexity for computing the geodesic distance function $D^{\mathcal{S}}$.

The nice part about the formulation in Algorithm 4 is that it poses the problem in the opposite way: not "who is farthest away from me?" but "for whom am i farther(st) away?". As for each point we are interested only in the eccentricity i.e. distance to the point farthest away, points which are not farthest away for any point in $\mathcal{S}$ (points which are not eccentric) do not even need to be considered. Algorithm 5 incorporates the above - Line 2 has changed compared to Algorithm 4. It can be used when a priory information about the shape exists: e.g. if the shape has less than two holes we can consider the eccentric point candidates as $\mathcal{P} = \partial\mathcal{S}$ (Properties 9, 10). Less points in $\mathcal{P}$ gives a faster computation.

The smallest $\mathcal{P}$ for which Algorithm 5 produces an error free result is the set of eccentric points of $\mathcal{S}$ i.e. $\mathcal{P} = E(\mathcal{S})$. Unfortunately this is a "chicken-egg" problem as in most of the

Algorithm 5 $ECC(\mathcal{S})$: Compute ECC of $\mathcal{S}$ - inverse, bounded.

Input: Discrete shape $\mathcal{S}$, eccentric point candidates $\mathcal{P} \subseteq \mathcal{S}$.

1: **for all** $\mathbf{p} \in \mathcal{S}$ **do** $ECC(\mathbf{p}) \leftarrow 0$ /*initialize*/
2: **for all** $\mathbf{q} \in \mathcal{P}$ **do**
3: $D \leftarrow D^{\mathcal{S}}(\mathbf{q})$ /*compute SBDT for $\mathbf{q}$*/
4: **for all** $\mathbf{p} \in \mathcal{S}$ **do** $ECC(\mathbf{p}) \leftarrow \max(ECC(\mathbf{p}), D(\mathbf{q}))$ /*further away than we had before?*/
5: **end for**

Output: Distances ECC.

cases the set of eccentric points $E(\mathcal{S})$ is only known after the eccentricity transform has been computed.

4.2 Progressive Refinement Eccentricity Transform

This section presents three approximation algorithms following the philosophy of Algorithm 5. They approach the problem of knowing the eccentric point candidates before computing the eccentricity transform ("chicken-egg" problem at the end of Section 4.1) by sequentially refining an estimate for the eccentricity transform and the candidate eccentric points - one at a time. All three algorithms try to find $E(\mathcal{S})$ and compute $D^{\mathcal{S}}(\mathbf{e})$ for all $\mathbf{e} \in E(\mathcal{S})$. As $E(\mathcal{S})$ is actually known only after $ECC(\mathcal{S})$ is computed, the algorithms incrementally refine an initial approximation of $ECC(\mathcal{S})$ by computing $D^{\mathcal{S}}(\mathbf{q})$ for candidate eccentric points $\mathbf{q}$ that are identified during the progress of the approximation. Different heuristics based on different properties and observations have lead to the different algorithms. They all share the fact that $D^{\mathcal{S}}$ is not computed twice for any point, so the complexity is below the one of the algorithms in the previous section. Every computation of $D^{\mathcal{S}}$ is accumulated like in Line 4 of Algorithm 5. For ECC06' (Section 4.2.2) and ECC08 (Section 4.2.3), discovering one eccentric point per eccentric point cluster is sufficient to produce the correct result. Experiments comparing the three methods follow in Section 4.3.

4.2.1 ECC06 - center to periphery

Algorithm 6 (*ECC06*) [Kropatsch 06] tries to identify points of the geodesic center (minimum ECC) and use those to find eccentric point candidates. Computing $D^{\mathcal{S}}(\mathbf{c})$ for a center point $\mathbf{c} \in C(\mathcal{S})$ is expected to create local maxima where eccentric points lie. In a first phase, the algorithm identifies at least two diameter ends by repeatedly "jumping" (computing $D^{\mathcal{S}}(\mathbf{p})$) to the point that had the highest distance value in the previous estimation (Phase 1 in Algorithm 6). In the second phase, the center points $\mathbf{c}_i$ are estimated as the points with the minimum eccentricity and all local maxima in $D^{\mathcal{S}}(\mathbf{c})$ are marked as eccentric points candidates, for which $D^{\mathcal{S}}$ is computed and accumulated. When no new (uncomputed) local maxima exist, the algorithm is stopped.

Algorithm ECC06 is faster than the naive ones (Section 4.1) and the fastest of the approximation algorithms presented in this section. It produces the 100% correct results only for a class of simply connected shapes. In general it also gives the highest error (see Section 4.3 for a feeling of how big this highest error actually is). Properties 26 and 27 explain why: ECC06 just considers local maxima as eccentric points candidates.

Algorithm 6 $ECC06(\mathcal{S})$: Eccentricity transform by progressive refinement - use center.

Input: Discrete shape $\mathcal{S}$.

1: **for all** $\mathbf{q} \in \mathcal{S}, ECC(\mathbf{q}) \leftarrow 0$ /**initialize distance matrix*/
2: $\mathbf{p} \leftarrow$ random point of $\mathcal{S}$ /**find a starting point*/
3:
4: /**Phase 1: find a diameter*/
5: **while** $\mathbf{p}$ not computed **do**
6: $ECC \leftarrow \max\{ECC, D^{\mathcal{S}}(\mathbf{p})\}$ /**accumulate and mark* $\mathbf{p}$ *as computed*/
7: $\mathbf{p} \leftarrow \arg\max\{ECC(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}\}$ /**highest current ECC (farthest away)*/
8: **end while**
9:
10: /**Phase 2: find center points and local maxima*/
11: $pECC \leftarrow 0$ /**make sure we enter the loop*/
12: **while** $pECC \neq ECC$ **do**
13: $pECC \leftarrow ECC$
14: $C \leftarrow \arg\min\{ECC(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}\}$ /**find all points with minimum ECC*/
15: **for all** $\mathbf{c} \in C$, $\mathbf{c}$ not computed **do**
16: $D \leftarrow D^{\mathcal{S}}(\mathbf{c})$ /**compute distances from the center*/
17: $ECC \leftarrow \max\{ECC, D\}$ /**accumulate and mark* $\mathbf{c}$ *as computed*/
18:
19: $\mathcal{M} \leftarrow \{\mathbf{q} \in \mathcal{S} \mid D(\mathbf{q})$ local maximum in $\mathcal{S}$ and $\mathbf{q}$ not computed$\}$
20: **for all** $\mathbf{m} \in \mathcal{M}$, $\mathbf{m}$ not computed **do**
21: $ECC \leftarrow \max\{ECC, D^{\mathcal{S}}(\mathbf{m})\}$ /**accumulate and mark* $\mathbf{m}$ *as computed*/
22: **end for**
23: **end for**
24: **end while**

Output: Distances ECC.

4.2.2 ECC06' - center to periphery and grow clusters

Algorithm 7 (*ECC06'*) [Ion 08d] extends ECC06 with a third phase similar to region growing, initiated at each eccentric point estimated by phases 1-2 in ECC06. The third phase has the purpose to explore the limits of the identified eccentric point clusters. Growing is continued while new points (neighbors of previously known eccentric points) are also eccentric. The type of region that is "grown" is decided based on the properties of the shape (e.g. if $\mathcal{S}$ is a 2D shape with less than 2 holes, growing on the boundary of $\mathcal{S}$ is enough. For $\mathcal{S}$ in 3D, the option between growing on the surface of $\mathcal{S}$ or in the volume exists, etc.). In most cases $\mathcal{P} = \partial\mathcal{S}$ will be enough in Algorithm 7 – bigger $\mathcal{P}$ = slower but could produce smaller errors.

For non-simply connected shapes the center can become very complex, it can contain many points and it can be disconnected (e.g. the shape in Section 3.5.10). This makes identifying all center points harder, as not all eccentric points are farthest away from all center points. Missing

Algorithm 7 $ECC06'(\mathcal{S})$: ECC by progressive refinement - use center & grow clusters.

Input: Discrete shape $\mathcal{S}$, eccentric points domain $\mathcal{P} \subseteq \mathcal{S}$.

```
1: ECC ← ECC06(S) /*call ECC06 for first two phases*/
2:
3: /*Phase 3: find limits of clusters of eccentric points */
4: ToDo ← all neighbours in P of all eccentric points in ECC
5: while ToDo ≠ ∅ do
6:    p ↚ arg max{ECC(p) | p ∈ ToDo} /*remove point with highest current ECC*/
7:    ECC ← max{ECC, D^S(p)} /*accumulate and mark p as computed*/
8:
9:    /*do we need to continue in this direction?*/
10:   if ECC changed previously i.e. p is an eccentric point then
11:      ToDo ← ToDo ∪ {q ∈ P | q is a neighbour of p in S and q not computed}
12:   end if
13: end while
```

Output: Distances ECC.

center points can lead to missing eccentric point clusters, which leads to approximation errors.

4.2.3 ECC08 - sample candidates and grow clusters

Algorithm 8 (*ECC08*) [Ion 08d] first attempts to identify at least one point from each cluster of eccentric points. Like in ECC06', growing is then used to find the limits of each cluster.

Similar to phase 1 in Algorithms ECC06 and ECC06', inspecting the shape $\mathcal{S}$ is done by repeatedly computing and accumulating $D^{\mathcal{S}}(\mathbf{p})$ for the highest uncomputed local maximum in the current and past approximations of $ECC(\mathcal{S})$ (Phase 1 in Algorithm 8). While in ECC06 and ECC06' this "jumping" around has the purpose to quickly find a diameter, in ECC08 it is the search for eccentric point clusters. Each point $\mathbf{p}$ that was in some iteration a local maximum in the approximation of ECC, is inserted into a $ToDo$ list and $D^{\mathcal{S}}(\mathbf{p})$ will be computed for it.

Without the "only compute once" condition this process could enter an infinite loop when all reachable points have been already visited. Such a configuration is called an oscillating configuration and the visited points are called oscillating points [Schmitt 93]. If $D^{\mathcal{S}}(\mathbf{p})$ with $\mathbf{p} \in \mathcal{S}$ is considered an approximation for $ECC(\mathcal{S})$, the error is expected to be higher around $\mathbf{p}$ and smaller around the points farther away from $\mathbf{p}$ i.e. the points with highest values in $D^{\mathcal{S}}(\mathbf{p})$.

Whenever an oscillating configuration is reached all points of $\mathcal{P} \subseteq \mathcal{S}$ which are local minima in the current ECC approximation are selected for distance computation. If the last operation does not produce any unvisited points as ECC local maxima, the search is terminated. Phase 2 in ECC08 is equivalent to phase 3 in ECC06' and "grows" eccentric point clusters based on at least one representative point found before.

Algorithm 8 $ECC08(\mathcal{S})$: ECC by progressive refinement - use $\mathcal{P}$ minima & grow clusters.

Input: Discrete shape $\mathcal{S}$, eccentric points domain $\mathcal{P} \subseteq \mathcal{S}$.

for all $\mathbf{q} \in \mathcal{S}, ECC(\mathbf{q}) \leftarrow 0$ /**initialize distance matrix**/
$ToDo \leftarrow$ random point of $\mathcal{S}$ /**find a starting point**/

/**Phase 1: inspect shape**/
while $ToDo \neq \emptyset$ **do**
  $\mathbf{p} \nleftarrow \arg\max\{ECC(\mathbf{p}) \mid \mathbf{p} \in ToDo\}$ /**remove point with highest current ECC**/
  $ECC \leftarrow \max\{ECC, D^{\mathcal{S}}(\mathbf{p})\}$ /**accumulate and mark* $\mathbf{p}$ *as computed**/

  /**add not computed local maxima to ToDo**/
  $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \mathcal{S} \mid ECC(\mathbf{q})$ local maximum in $\mathcal{S}$ and $\mathbf{q}$ not computed$\}$

  /**test if an oscillating configuration was found**/
  if $ToDo = \emptyset$ **then**
    $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \mathcal{P} \mid ECC(\mathbf{q})$ local minimum in $\mathcal{P}$ and $\mathbf{q}$ not computed$\}$
  end if
end while

/**Phase 2: find limits of clusters of eccentric points (identical to Phase 3 in ECC06') */
$ToDo \leftarrow$ all neighbors in $\mathcal{P}$ of all eccentric points in ECC
while $ToDo \neq \emptyset$ **do**
  $\mathbf{p} \nleftarrow \arg\max\{ECC(\mathbf{p}) \mid \mathbf{p} \in ToDo\}$ /**remove point with highest current ECC**/
  $ECC \leftarrow \max\{ECC, D^{\mathcal{S}}(\mathbf{p})\}$ /**accumulate and mark* $\mathbf{p}$ *as computed**/

  /**do we need to continue in this direction?**/
  if ECC changed previously i.e. $\mathbf{p}$ is an eccentric point **then**
    $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \mathcal{P} \mid \mathbf{q}$ is a neighbor of $\mathbf{p}$ in $\mathcal{S}$ and $\mathbf{q}$ not computed$\}$
  end if
end while

Output: Distances ECC.

4.3 Experiments

We have compared the three algorithms, ECC06, ECC06', and ECC08, on 70 shapes from the MPEG7 CE-Shape1 database [Latecki 00], 6 from [Flanitzer 06], and one additional new shape (see Table 4.5).

The MPEG7 database contains 1400 shapes from 70 object classes. One shape from each class was taken (the first one) and reduced to about 36,000 pixels (aspect ratio and connectivity preserved, e.g. 192x192 for square images). Table 4.1 summarizes the main characteristics of the 70 shapes, their sizes and the range of smallest and largest eccentricity values. The smallest

Table 4.1: Characteristics of shapes from the MPEG7 database.

measure	ranges from	to
sizes in pixel	683	28821
smallest eccentricity in pixel (ECC_{min})	28	235
maximum eccentricity in pixel (ECC_{max})	55	469

Table 4.2: Results of 70 images from the MPEG7 database.

measure	**ECC06**	**ECC06'**	**ECC08**
max.pixel error	4.45 / 221.4	4.27 / 221.4	6.57 / 266.6
max.pixel error (%)	2%	2%	2.46%
max.error size	4923 / 19701	2790 / 19701	2359 / 19701
#DT(ECC) / #DT(RECC)	8%	10%	15%
100% correct	44 / 70	60 / 70	56 / 70

Table 4.3: "worst" results from the MPEG7 database.

shape characteristics					max.ECC.diff.			size of ECC.diff.		
no	name	ECC_{min}	ECC_{max}	size	ECC06	ECC06'	ECC08	ECC06	ECC06'	ECC08
58	pocket	170.7	266.6	13815	3.750	0.000	6.568	2241	0	1318
48	hat	126.5	221.4	19701	4.454	4.274	4.274	4923	2790	2359
5	Heart	108.1	213.4	24123	2.784	0.731	0.731	2378	482	482
4	HCircle	127.0	250.2	28821	0.000	0.000	1.454	0	0	404
18	cattle	99.6	198.2	9764	1.223	1.223	1.223	2154	258	258
11	bird	116.0	230.1	14396	1.209	0.000	0.000	3963	0	0

eccentricity appears at the geodesic center of the shape, and the largest eccentricity corresponds to its diameter.

Reference ECC values are computed by the naive algorithm (Algorithm 5) denoted by RECC, as the maximum of the distance transforms of all boundary points $\partial\mathcal{S}$.

Table 4.2 compares the performance of the 3 algorithms:

max.pixel error: maximum difference between RECC and ECC per pixel / max.eccentricity for this shape;

max.error size: maximum number of pixels that differ between RECC and ECC / size of this shape;

#DT(ECC) / #DT(RECC): average number of times the geodesic distance function $D^{\mathcal{S}}$ is computed w.r.t. RECC (in percent);

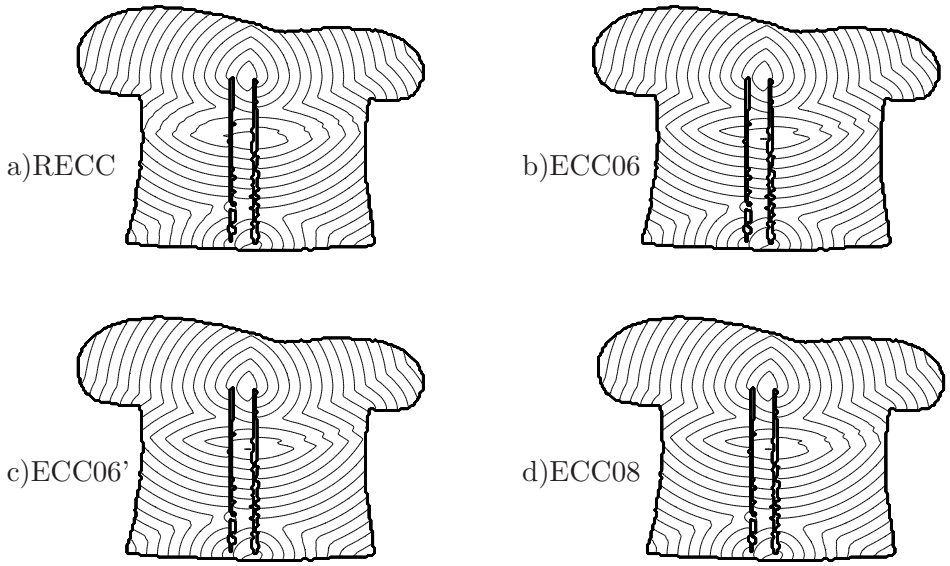


Figure 4.1: Results of example shape *hat*

100% correct: the number of shapes for which the error was 0 for all pixels / the total number of shapes.

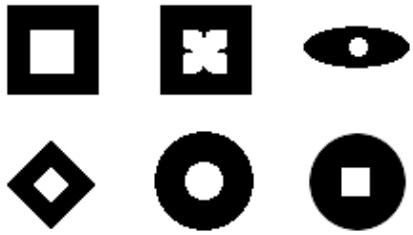
All three algorithms produce a good ECC approximation in about 8% to 15% of the time of RECC. There are only a few shapes for which the estimation (computation) is not 100% correct and the highest difference in eccentricity was about 7 (pixel units) in an image where the eccentricities varied from 170 to 266.6.

Table 4.3 lists the 6 worst results with the three algorithms. Each shape is characterized by its number, its name, the range of eccentricities of RECC, and the number of pixels (size). The next columns list the largest difference in eccentricity value and the number of pixels that were different. To judge the quality of the results we selected the example *hat* which had errors in all three algorithms (Figure 4.1 shows the results by a contour line plot with the same levels). Algorithms ECC06' and ECC08 compute the correct eccentricity transform for all of the "problem" shapes showing the improvement with respect to ECC06 with 4- and 8-connectivity used in [Flanitzer 06] (see Table 4.4).

Table 4.5 shows the results of the three algorithms on a more complex 2D shape (3 holes). On this example ECC06 and ECC06' produce better results than ECC08.

Overall ECC06' produces the best results with a computation speed between ECC06 and ECC08. ECC06 is the fastest in this experiment.

Table 4.4: Results on the 6 "problem" shapes from [Flanitzer 06].



measure in %	ECC06	ECC06'	ECC08
max.pixel error	1.521 / 74.7	0.00 / 74.7	0.00 / 74.7
max.error size	675 / 1784	0 / 1784	0 / 1784
$\frac{\#\text{DT(ECC)}}{\#\text{DT(RECC)}}$	27%	50%	48%
100% correct	1 /6	6/6	6/6

Table 4.5: Results for image "3holes" ($|\mathcal{S}| = 19919$ pixels).

measure	ECC06	ECC06'	ECC08
max.pixel error	1.913 /409.2	1.100 / 409.2	12.773 / 409.2
max.error size	360 / 19919	119 / 19919	698 / 19919
$\frac{\#\text{DT(ECC)}}{\#\text{DT(RECC)}}$	7%	8%	9%

4.4 Discussion

All presented algorithms rely on the computation of the geodesic distance function $D^{\mathcal{S}}$. If considering the eccentricity transform in any dimension, one has to look at $D^{\mathcal{S}}$ first. Fast Marching (FM) (Section 2.2.1) works in a similar manner in higher dimensions (3D, 4D), and the Discrete Circle Propagation (DCP) (Section 2.2.2) was not studied yet in dimensions higher than 2. On the other side, DCP computes exact distances while FM produces only an approximation. For regular grids, or structures with fixed neighborhoods (e.g. 4, 8, 6, 26, graphs) efficient classical algorithms exist (e.g. Dijkstra, Section 2.2).

One advantage of the algorithms presented in this chapter is that they can be easily extended/applied for discrete shapes of any dimension. For Algorithms 3, 4, and 6 (ECC06) once the geodesic distance function $D^{\mathcal{S}}$ is given, nothing has to be changed (Section 5.1 uses ECC06 for the computation of the eccentricity transform of 3D shapes - in the volume and on the 2D manifold defined by the boundary of the 3D shapes). As the parameter $\mathcal{P}$ for Algorithms 5, 7,

a) $D^{\mathcal{S}}(\mathbf{p}_1)$ b) $D^{\mathcal{S}}(\mathbf{p}_2)$ c) cut

Figure 4.2: Example where $D^{\mathcal{S}}$ does not have to be fully computed.

and 8 will probably depend on the shape ($\partial\mathcal{S}$ is one dimensional if $\mathcal{S}$ is two dimensional, and two dimensional if $\mathcal{S}$ is three dimensional) some considerations have to be made e.g. one could use the 6 connected boundary of a 3D shape, or the whole shape, etc. Other than that, these algorithms can also be directly applied.

In algorithms 3, 4, and 5, computing the geodesic distance can be done in parallel (each computation of $D^{\mathcal{S}}$ can be done separately) with a final step to compute the maximum (ECC). In Algorithm 6, Line 21 can be executed in parallel for all local maxima. The eccentric point cluster growing (last phase in Algorithms 7 and 8) can also be done in parallel for each cluster, and inside each cluster (all points of the grown region boundary at a certain time step).

4.4.1 Full $D^{\mathcal{S}}$ not always needed.

If running the algorithms sequentially, one can notice that in many cases the computation of $D^{\mathcal{S}}(\mathbf{p})$ for some $\mathbf{p} \in \mathcal{S}$ does not have to be completed, as it is clear that no higher value will be obtained through this. Figure 4.2 shows an example: a) first $D^{\mathcal{S}}$, b) second $D^{\mathcal{S}}$, and c) cut at which we could abandon the computation of b). This is mainly possible because the distance function creates an ordering and a dependency of points s.t. if a state is weaker by some criteria than a previous one, all dependent points will also be weaker (e.g. same as saying: everybody needs the same time from A to B, so if someone was in A before me, I cannot get to B before them).

For a shape represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such a cut $\mathcal{C} \subseteq \mathcal{E}$, where we could abandon the computation, has to have the following properties:

- $\mathcal{C}$ has to be a cut by the definition from graph theory [Diestel 97] i.e. for $\mathcal{V}_1, \mathcal{V}_2$ s.t. $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$, $\mathcal{C}$ is the set of edges $\{(\mathbf{v}_1, \mathbf{v}_2) \mid \mathbf{v}_1 \in \mathcal{V}_1, \mathbf{v}_2 \in \mathcal{V}_2\}$.

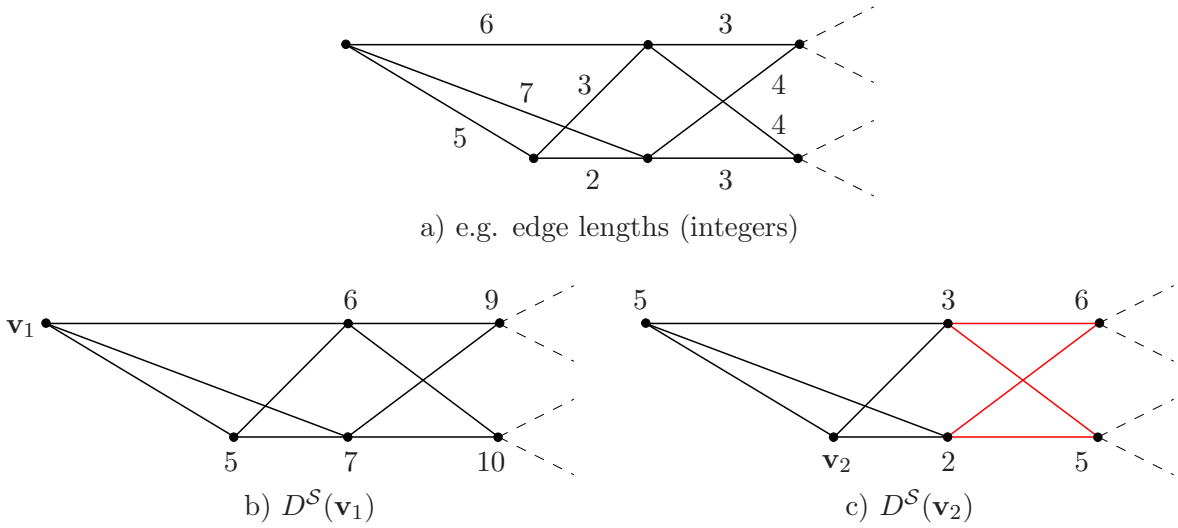


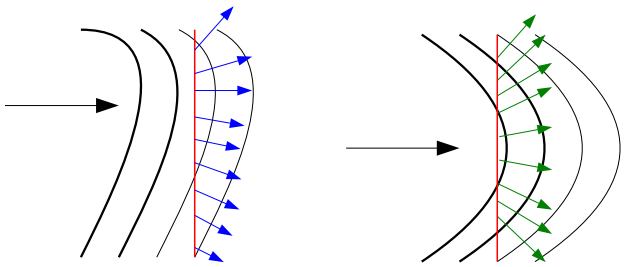
Figure 4.3: Detecting when to abort $D^{\mathcal{S}}$ in a graph.

- for both distance functions $D_1 = D^{\mathcal{S}}(\mathbf{v}_1)$ and $D_2 = D^{\mathcal{S}}(\mathbf{v}_2)$, $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}$, all edges of $\mathcal{C}$ should have been passed in the same direction i.e. $\forall(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{C}$, the ordering of $d(\mathbf{v}_1, \mathbf{v}_i)$ and $d(\mathbf{v}_2, \mathbf{v}_i)$ has to be a strict one, and the same as the ordering of $d(\mathbf{v}_1, \mathbf{v}_j)$ and $d(\mathbf{v}_2, \mathbf{v}_j)$.

In words, it means that for comparing two distance functions (distance propagations) at a cut, we need a cut that divides $\mathcal{S}$ in at least two parts and that is being crossed by both functions in the same direction. None of the functions crosses the cut in both directions and both functions go from the same side to the other. Figure 4.3 shows an example for a graph. The edges in red show a cut where $D^{\mathcal{S}}(\mathbf{v}_2)$ could be aborted if $D^{\mathcal{S}}(\mathbf{v}_1)$ has been computed before. All edges of the cut share the same ordering (vertices on the left are closer - propagation from left to right) and all have smaller distances in D_2 than in D_1, so no higher distance will be produced on the right side of the cut.

For the more general, continuous case with $\mathcal{S} \in \mathbb{R}^n$, $\mathcal{C} \subset \mathcal{S}$ is an $n-1$ dimensional manifold i.e. for $n = 2$, $\mathcal{C}$ is one dimensional (line), for $n = 3$, $\mathcal{C}$ is two dimensional (surface). $\mathcal{C}$ divides $\mathcal{S}$ in $\mathcal{S}'$ and $\mathcal{S}''$. Consider two geodesic distance functions $D_1 = D^{\mathcal{S}}(\mathbf{x}_1)$ and $D_2 = D^{\mathcal{S}}(\mathbf{x}_2)$, $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}$. For $\mathcal{C}$ to be a proper cut for comparing D_1, D_2, we need that for all $\mathbf{c} \in \mathcal{C}$ the direction of propagation of both D_1 and D_2 at $\mathbf{c}$ has to be from $\mathcal{S}' - \mathcal{C}$ to $\mathcal{S}'' - \mathcal{C}$ (this assumes that $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}'$). Figure 4.4 shows an example: $\mathcal{C}$ (in red) is a proper cut for a) and b) as all the normals to the wavefront at points on the cut are oriented toward the same side. If $\forall \mathbf{c} \in \mathcal{C}$, $d(\mathbf{x}_1, \mathbf{c}) > d(\mathbf{x}_2, \mathbf{c})$ then $D^{\mathcal{S}}(\mathbf{x}_2)$ can be stopped at $\mathcal{C}$. Figure 4.5 shows an example improper cut: the normals change the side to which they point to.

Checking a cut for stopping can be done with low additional cost, if the cut is the wavefront.



a) $D^{\mathcal{S}}(\mathbf{x}_1)$ b) $D^{\mathcal{S}}(\mathbf{x}_2)$

Figure 4.4: Example proper cut (red) and normals (blue, green) for a continuous shape.

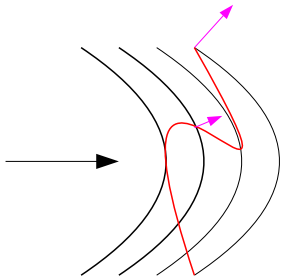


Figure 4.5: Improper cut (red): normals (magenta) do not point to the same side of the cut.

For example, in Dijkstra's algorithm (Section 2.2) just keep the number of "good" edges in the cut, and update every time a new vertex is computed. Unfortunately the wavefront is not a very good option for a cut, as in many cases, after the wavefront is split, one should continue propagating on one side, but stop on another (e.g. continue with only one finger of the hand). This is a more complex and still open problem, as it is not just connected to the properties of the front, but to the structure of the shape itself - one can stop propagating in tree structures, but has to probably continue around holes.

4.5 A Different Approach: Divide and Conquer

The ellipse (Section 3.5.5) is a shape for which it is possible to compute the eccentricity transform by decomposing it along the short diameter, computing the eccentricity transform of the parts and then obtaining the eccentricity transform of the whole from the eccentricity transform of the parts. Consider the more general case of a shape $\mathcal{S}$ with $|\mathcal{S}| = n$ points (pixels, voxels). The worst time complexity for computing $ECC(\mathcal{S})$ is $O(n^2 \log n)$ (Algorithm 3). Decompose $\mathcal{S}$ in two parts $\mathcal{S}_l$ and $\mathcal{S}_r$ with $|\mathcal{S}_l| = n_l$ and $|\mathcal{S}_r| = n_r$. If putting together the eccentricity of the whole ($\mathcal{S}$), from the parts $(\mathcal{S}_l, \mathcal{S}_r)$ can be done, an estimated complexity would be $O(n_l^2 \log n_l) +$

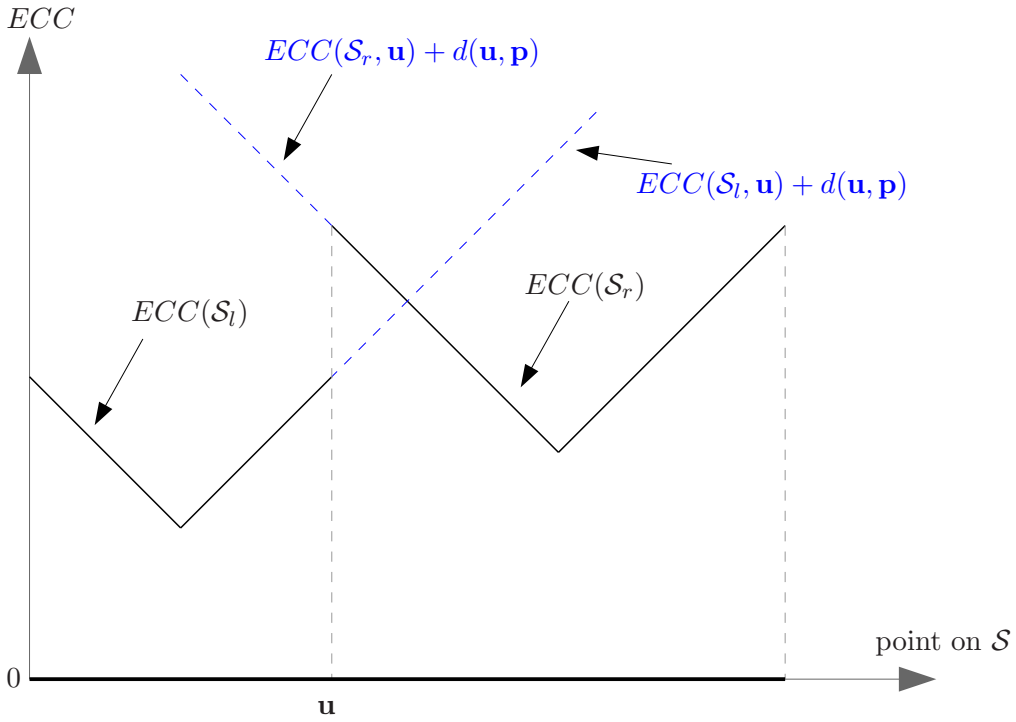


Figure 4.6: Divide and Conquer for the 1D curve.

$O(n_r^2 \log n_r) + O(k)$ where n_l, n_r are the number of pixels of the parts, and k is the complexity for "putting the parts together". The problem has two important aspects:

- What is a good (preferably best) decomposition?
- How to do the reconstruction from the decomposition?

4.5.1 open curve (1D eccentricity)

Let $\mathcal{S}$ be a non self intersecting one dimensional manifold in $\mathbb{R}^n$ (Section 3.5.1 gives the eccentricity transform of such a shape). We subdivide $\mathcal{S}$ at point $\mathbf{u} \in \mathcal{S}$ and compute the eccentricity of $\mathcal{S}_l$ and $\mathcal{S}_r$ (Figure 4.6). The eccentricity of $\mathcal{S}$ is:

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} \max\{ECC(\mathcal{S}_l, \mathbf{p}), ECC(\mathcal{S}_r, \mathbf{u}) + d(\mathbf{u}, \mathbf{p})\} & \text{if } \mathbf{p} \in \mathcal{S}_l \\ \max\{ECC(\mathcal{S}_r, \mathbf{p}), ECC(\mathcal{S}_l, \mathbf{u}) + d(\mathbf{u}, \mathbf{p})\} & \text{if } \mathbf{p} \in \mathcal{S}_r \end{cases}$$

Compared to the ellipse, for the points on the cut, there is no need for additional information besides the eccentricity of the points (like the angle in the case of the ellipse). The association between $\mathbf{p}$ and the point on the cut is clear, as the cut has only one point. If $\mathbf{u}$ is the middle of $\mathcal{S}$ (line), $ECC(\mathcal{S}_l)$ and $ECC(\mathcal{S}_r)$ do not have to be fully computed, only $ECC(\mathcal{S}_l, \mathbf{u})$ and

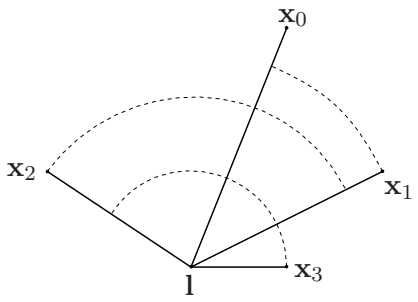


Figure 4.7: Example tree $\mathcal{S}_t$ with a single junction and $n = 4$ end vertices.

$ECC(\mathcal{S}_r, \mathbf{u})$. Because, like in the case of the ellipse, all points in one part $(\mathcal{S}_l, \mathcal{S}_r)$ will have their eccentric points in the other part $(\mathcal{S}_r, \mathcal{S}_l)$.

4.5.2 A tree

Let $\mathcal{S}$ be a tree like structure made out of 1D curves (edges) and junction points (vertices) in $\mathbb{R}^n$, no cycles. We consider two cases:

- decomposing on the curves (edges);
- decomposing at a junction point.

The basic building blocks will be the eccentricity transform of the 1D curve (section above) and the eccentricity transform of a tree $\mathcal{S}_t$ with a single junction vertex $\mathbf{l}$, and end vertices (leaves) $\mathbf{x}_i$, $0 \leqslant i < n$ (in decreasing order of their length) (Figure 4.7 shows an example):

$$ECC(\mathcal{S}_t, \mathbf{p}) = \begin{cases} d(\mathbf{l}, \mathbf{x}_0) & \text{if } \mathbf{p} = \mathbf{l} \\ d(\mathbf{p}, \mathbf{l}) + d(\mathbf{l}, \mathbf{x}_0) & \text{if } \mathbf{p} \notin \text{ branch with } \mathbf{x}_0 \\ d(\mathbf{p}, \mathbf{l}) + d(\mathbf{l}, \mathbf{x}_1) & \text{if } \mathbf{p} \in \text{ branch with } \mathbf{x}_0 \end{cases}$$

Decomposing $\mathcal{S}$ along an edge

This case is very similar to the open curve (1D), as eccentric paths of the whole, either stay inside the respective part, or cross the single point of decomposition (cut). Using the same notation as in Section 4.5.1 ($\mathbf{u}$ decomposition point and $\mathcal{S}_l$, $\mathcal{S}_r$, the two parts):

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} \max\{ECC(\mathcal{S}_l, \mathbf{p}), ECC(\mathcal{S}_r, \mathbf{u}) + d(\mathbf{u}, \mathbf{p})\} & \text{if } \mathbf{p} \in \mathcal{S}_l \\ \max\{ECC(\mathcal{S}_r, \mathbf{p}), ECC(\mathcal{S}_l, \mathbf{u}) + d(\mathbf{u}, \mathbf{p})\} & \text{if } \mathbf{p} \in \mathcal{S}_r \end{cases}$$

Decomposing $\mathcal{S}$ at a junction

The shape $\mathcal{S}$ is decomposed at $\mathbf{u}$ into n parts $\mathcal{S}_i$, $0 \leqslant i < n$. We get, for all $\mathbf{p} \in \mathcal{S}$:

$$ECC(\mathcal{S}, \mathbf{p}) = \max\{ECC(\mathcal{S}_i, \mathbf{p}), d(\mathbf{p}, \mathbf{u}) + \max_{0 \leqslant k < n, k \neq i}\{ECC(\mathcal{S}_k, \mathbf{u})\}\},$$

where $\mathcal{S}_i \subset \mathcal{S}$ is the part containing the point $\mathbf{p}$ i.e. $\mathbf{p} \in \mathcal{S}_i$. Except for one case, the "inner" maximum ($\max\{ECC(\mathcal{S}_k, \mathbf{u})\}$) always returns the same value $ECC(\mathcal{S}_m, \mathbf{u}), 0 \leqslant m < n$. If $i = m$, the second highest value $ECC(\mathcal{S}_{m'}, \mathbf{p})$ will be taken:

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} \max\{ECC(\mathcal{S}_i, \mathbf{p}), d(\mathbf{p}, \mathbf{u}) + ECC(\mathcal{S}_m, \mathbf{u})\} & \text{if } i \neq m \\ \max\{ECC(\mathcal{S}_i, \mathbf{p}), d(\mathbf{p}, \mathbf{u}) + ECC(\mathcal{S}_{m'}, \mathbf{u})\} & \text{if } i = m \end{cases}$$

for $\mathbf{p} \in \mathcal{S}_i$, $0 \leqslant i < n$, and $\mathcal{S}_m, \mathcal{S}_{m'}$ the parts with the highest respectively second highest eccentricity for $\mathbf{u}$. In the case of this decomposition, eccentric paths in more than one direction can pass over the decomposition point.

4.5.3 Convex 2D shape

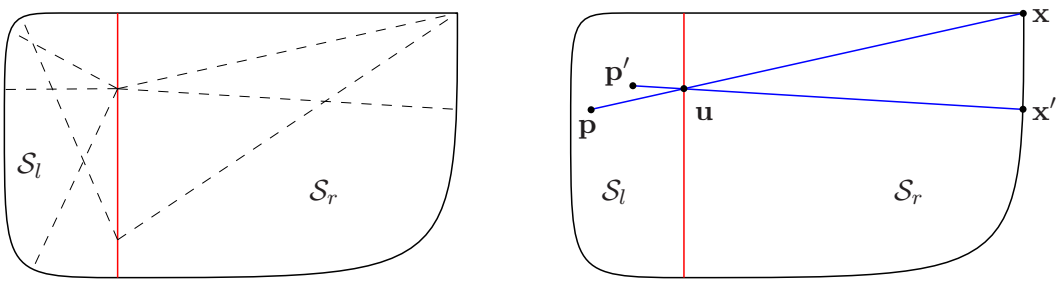
This case is similar to the ellipse (Section 3.5.5). Eccentric paths have to be orthogonal to $\partial\mathcal{S}$ or end in corner points. The shape can be decomposed along a straight line segment $\mathcal{C}$ and pairs of distances and angles can be associated to the points of $\mathcal{C}$. To each point, zero or more distance-direction pairs (i.e. a pair of two values, a distance and a direction/angle) are associated, one for each direction in which the point is on a normal to $\partial\mathcal{S}$, or $\partial\mathcal{S}$ is a corner point (Figure 4.8.a shows example distance-direction pairs). The eccentricity of the whole is found by choosing the point on $\mathcal{C}$ which maximizes the sum of the two distances (to the point and from the point to the boundary), and the angles match (the eccentric path is straight over $\mathcal{C}$).

Given a point $\mathbf{p} \in \mathcal{S}$ (Figure 4.8.b), consider $\mathbf{u} \in \mathcal{C}$ and $\mathbf{x} \in \partial\mathcal{S}$ s.t. $\mathbf{u}$ is a point of the line segment $\mathbf{px}$, and $\mathbf{x}$ is a corner point or the segment $\mathbf{ux} \perp \partial\mathcal{S}$ at $\mathbf{x}$. The eccentricity transform of $\mathcal{S}$ using the decomposition into parts $\mathcal{S}_l$ and $\mathcal{S}_r$ is (Figure 4.8 illustrates the used notation):

$$ECC(\mathcal{S}, \mathbf{p}) = \begin{cases} \max\{ECC(\mathcal{S}_l, \mathbf{p}), \max_{\mathbf{x} \in \mathcal{S}_r}\{d(\mathbf{p}, \mathbf{u}) + d(\mathbf{u}, \mathbf{x})\}\} & \text{if } \mathbf{p} \in \mathcal{S}_l \\ \max\{ECC(\mathcal{S}_r, \mathbf{p}), \max_{\mathbf{x} \in \mathcal{S}_l}\{d(\mathbf{p}, \mathbf{u}) + d(\mathbf{u}, \mathbf{x})\}\} & \text{if } \mathbf{p} \in \mathcal{S}_r \end{cases}$$

with $\mathbf{u} \in \mathcal{C}$ and $\mathbf{x} \in \partial\mathcal{S}$ as defined above.

The eccentricity transform of the parts is no longer enough to compute the eccentricity of the whole. A curved cut $\mathcal{C}$ makes it impossible to compute the distance-direction pairs independently on each part, as geodesic distances between two points of the same part could be different if



a) distance-direction pair examples b) paths through the same point

Figure 4.8: Convex 2D shape: cut $\mathcal{C}$, dashed: distance-angle pair, eccentric path.

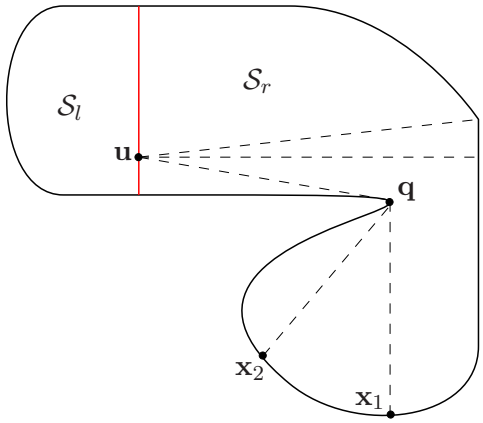


Figure 4.9: Non-convex 2D shape: cut $\mathcal{C}$, dashed: distance-angle pair.

computed in the part or if computed in the whole. These considerations can be applied to any n dimensional convex shape, using the Euclidean distance d_ε.

4.5.4 Non-convex 2D shape (simply connected)

The non-convex 2D shape can be considered as an extension of the convex one (Section 4.5.3). In addition to directions that would intersect $\partial\mathcal{S}$ normal to the tangent at the intersection point, and directions leading to corner points, points on the cut have to consider the additional direction eccentric paths could go to: the tangent to concave parts of the boundary. Paths still end at convex parts of the boundary, whether smooth or corner points. Due to concave parts, from a single point $\mathbf{u} \in \mathcal{C}$ and a single direction, geodesics to more than one boundary point can go. In this case only the longest one has to be considered. Figure 4.9 shows an example: only $\mathbf{ux}_1$ has to be considered, as $\lambda(\mathbf{ux}_2) < \lambda(\mathbf{ux}_1)$.

Figure 4.10 illustrates the notation for the following. Consider a cut $\mathcal{C}$, a point $\mathbf{p} \in \mathcal{S}_l$, and

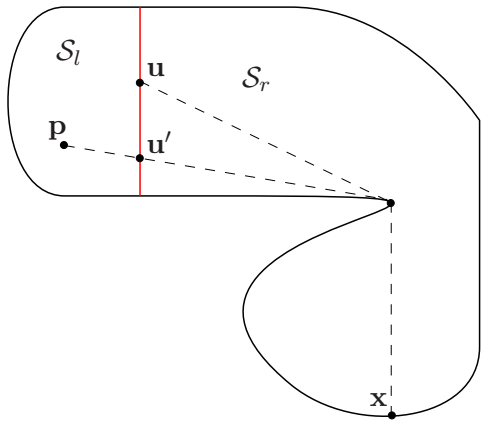
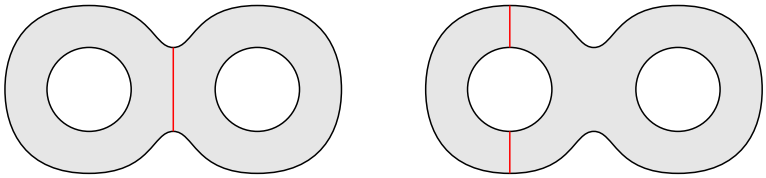


Figure 4.10: Lower bound for distances on the cut: cut $\mathcal{C}$.



a) connected cut b) cut "through" hole

Figure 4.11: Decomposition of 2D shape with holes: cut $\mathcal{C}$.

points $\mathbf{x} \in \partial\mathcal{S}_r \cap \partial\mathcal{S}$, $\mathbf{u} \in \mathcal{C}$, and $\mathbf{u}' = \mathcal{C} \cap \nu(\mathbf{p}, \mathbf{x})$ ($\mathbf{u}$' is the point where the geodesic from $\mathbf{p}$ to $\mathbf{x}$ cuts $\mathcal{C}$). From the triangle inequality we have $d(\mathbf{u}', \mathbf{x}) + d(\mathbf{u}, \mathbf{u}') \geqslant d(\mathbf{u}, \mathbf{x}) \iff d(\mathbf{u}', \mathbf{x}) \geqslant d(\mathbf{u}, \mathbf{x}) - d(\mathbf{u}, \mathbf{u}')$, which basically states that given $d(\mathbf{u}, \mathbf{x})$ we have a lower bound for $d(\mathbf{u}', \mathbf{x})$ as $d(\mathbf{u}, \mathbf{u}')$ is bounded by the length of the cut $\mathcal{C}$.

From the previous we get that, for a given cut $\mathcal{C}$ with end points $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$, and $\mathbf{u} \in \mathcal{C}$ the point with the largest associated distance m, we can ignore all distance-direction pairs associated to points $\mathbf{u}' \in \mathcal{C}$ for which the associated distance is smaller than $m - \max(d(\mathbf{u}, \mathbf{c}_1), d(\mathbf{u}, \mathbf{c}_2))$.

4.5.5 2D shape with holes

Two cases can be considered here (Figure 4.11 shows examples):

- a single connected straight line cut (not always possible),
- the cut goes "through" a hole (disconnected cut).

The first case (single line) is similar to the one in Section 4.5.4 (non-convex shape), as only one geodesic path can go in one direction through one point of the cut. In the second case (disconnected cut) (e.g. the disk with one circular hole in Section 3.5.10) more than one eccentric

path with the same direction can go through a single point of the cut, and the fact that paths do not change direction when crossing the cut is not enough to make sure they are geodesic paths i.e. shortest. Also, because separation lines can intersect the boundary at any point, not just normal or corner, and eccentric points can even lie inside the shape (Property 12 illustrated in Figure 3.5), the number of distance-direction pairs that should be stored could be as high as the number of points of the part.

For shapes where all eccentric points lie on convex parts or at corner points of the boundary, or where knowledge about the presence of eccentric points only in a small subset of $\mathcal{S}$ exists, the problem of multiple paths going to the same point could be solved also by associating the end-point to the distance-direction information. Then the shortest path from all paths ending at the same point has to be maximized.

4.6 Chapter Summary

This chapter considered the computation of the eccentricity transform. First, general computation and efficient approximation have been approached, supported by experiments. A discussion about possible improvements followed. A different approach, divide and conquer, covered the second part of the chapter. Decomposing the shape to reduce computation complexity has been discussed.

5

Example Applications of the Eccentricity Transform

This chapter presents two example applications to motivate the more theoretical part in Chapters 3 and 4. The first one deals with the problem of binary shape matching (Section 5.1) and the second one with shape centered coordinate systems (Section 5.2). The applications themselves are not the central part of this work and thus possible further improvements are proposed as future work.

5.1 Matching 2D and 3D Articulated Shapes using Eccentricity

Parts of this section (Sections 5.1.1, 5.1.2 and 5.1.4) have been previously published in [Ion 07b] (2D shape matching) and [Ion 08a] (3D shape matching).

The recent increase in available 3D models and acquisition systems has seen the need for efficient retrieval of stored models, making 3D shape matching gain attention also outside the computer vision community. Together with its 2D counterpart, 3D shape matching is useful for identification and retrieval in classical vision tasks, but can also be found in Computer Aided Design/Computer Aided Manufacturing (CAD/CAM), virtual reality (VR), medicine, molecular biology, security, and entertainment [Bustos 05].

Shape matching requires to set up a signature that characterizes the properties of interest for the recognition [Veltkamp 06]. The invariance of this signature to local deformations such as articulation is important for the identification of 2D and 3D shapes. Matching can then be carried out over this reduced space of signatures. Most shape descriptors are computed over a transformed domain that amplifies the important features of the shape while throwing away ambiguities such as translation, rotation or local deformations.

For 2D shapes, the *Fourier transform of the boundary curve* [Zahn 72] is an example of such a transformed domain descriptor adapted to smooth shapes. Shape *transformations computed with geodesic distances* [Bronstein 06] lead to signatures invariant to isometric deformations such as bending or articulation. To capture salient features of 2D shapes, local quantities such

as *curvature* [Mokhtarian 92] or *shape contexts* [Belongie 02] can be computed. They can be extended to *bending invariant signatures* using geodesic distances [Ling 07]. More global features include the *Laplace spectra* [Reuter 05] and the *skeleton* [Siddiqi 99]. Some transformations involve the computation of a function defined on the shape, for instance the solution to a linear *partial differential equation* [Gorelick 04] or *geometric quantities* [Osada 02]. Geodesic distance information such as the *mean-geodesic transform* [Hamza 03] could also be used.

Among approaches matching 3D shapes, existing methods can be divided into [Bustos 05]: *Statistical* descriptors, like for example geometric *3D moments* employed by [Elad 01, Paquet 00], and the *shape distribution* [Osada 02, Ip 03]. *Extension-based* descriptors, which are calculated from features sampled along certain directions from a position within the shape [Vranic 02, Vranic 01a]. *Volume-based* descriptors use the volumetric representation of a 3D shape to extract features (examples are *Shape histograms* [Ankerst 99], *Model Voxelization* [Vranic 01b], and point set methods [Tangelder 03]). Descriptors using the *surface geometry* compute curvature measures and/or the distribution of surface normal vectors [Paquet 99, Zaharia 01]. *Image-based* descriptors reduce the problem of 3D shape matching to an image similarity problem by comparing 2D projections of the 3D shapes [Ansary 04, Cyr 04, Chen 03]. Methods matching the *topology* of two shapes (for example *Reeb* graphs, where the topology of the 3D shape is described by a graph structure [Hilaga 01, Shinagawa 91]). *Skeletons* are intuitive shape descriptions and can be obtained from a 3D shape by applying a thinning algorithm on the voxelization of a solid object like in [Sundar 03]. Descriptors using *spin images* work with a set of 2D histograms of the shape geometry and a search for point-to-point correspondences is done to match 3D objects [Johnson 99].

The majority of shape descriptors is quite complex and not invariant to the deformation or articulation of object parts. They require extraction of salient features and local signatures that need to be aligned or registered.

In the context of shape matching the concept of *articulated shape* means that shapes that belong to articulations of the same object, belong to the same class. Assuming that the size of the junctions is very small compared to the size of the parts $\mathcal{O}_i$, it is shown that the variation of the geodesic distance[1] during articulation is small and that geodesic distances are articulation insensitive (Property 3).

Normalized histograms of the eccentricity transform are invariant to changes in orientation, scale, and articulation. We propose eccentricity histograms as descriptors for 2D [Ion 07b] and 3D shape matching [Ion 08a]. They require only a simple representation and can be efficiently matched. A common framework is presented with a study of the properties of the approach, supported by experimental results and detailed discussion. Four variants of the descriptor are

[1]Called "inner-distance" in [Ling 07].

Table 5.1: Types of manifolds used for matching.

Name	input	computing on	$\mathcal{S}$ (d_ε is used)
ECCobj2D	2D	2D: whole shape	4-connected binary 2D shape
ECCobj	3D	3D: whole shape	6-connected 3D voxel shape
ECCborder	3D	3D: border voxels	6 connected voxel surface in 3D, made out of voxels of the shape that are 26 connected to a background voxel
ECCmesh	3D	2D: triangular mesh	connected triangular mesh of the surface of the 3D shape

used, one for 2D shapes and three for 3D shapes (volume, border voxels, mesh) and compared to state of the art methods. To the best of our knowledge, this is the first approach applying eccentricity (furthest point distance) to the problem of shape matching. The presented approach could be fitted to either of the categories *extension-based* or *volume-based*, and it is a *transformation computed with geodesic distances*.

Like the method in [Ling 07], our method does not involve any part models. The articulation model is only for the analysis of the properties of the geodesic distance. Finding the correspondences between all the parts of two shapes is an NP-complete problem in graph theory [Atallah 98] (known also as the "matching" of two graphs) and requires the correct decomposition of the unknown object into parts. A one-to-one mapping is not always possible as some parts might be missing due to, for example, segmentation errors. Decomposition of the shapes into parts is not required by our approach.

This part is organized as follows: Section 5.1.1 discusses used variants of the eccentricity transform. Section 5.1.2 gives the proposed matching method and discusses pros and cons of the descriptor (Section 5.1.3). Section 5.1.4 presents details and discusses the results of the experiments, followed by parameters and improvements in Section 5.1.5.

5.1.1 Eccentricity transform - considerations

The class of $2n$-connected (Definition 7) discrete shapes $\mathcal{S}$ defined by points on a square grid $\mathbb{Z}^n$, $n \in \{2, 3\}$, as well as connected triangular meshes representing the surface of the 6-connected 3D shapes are considered. Table 5.1 shows the types of manifolds used, for which ECC is computed. For ECCobj2D, ECCobj, and ECCborder, paths need to be contained in the area of $\mathbb{R}^n$ defined by the union of the support squares/cubes for the pixels/voxels of $\mathcal{S}$. For ECCmesh, paths need to be contained in the 2D manifold defined by the union of the triangles of the mesh (including the interior of the triangles). The used (approximated) metric is in all cases d_ε. If increasing the resolution of the shapes, ECCborder and ECCmesh converge to the same value.

Figures 5.1 and 5.2 show the eccentricity transform of a 2D, respectively 3D, shape. For the 3D shape, the eccentricity transform is presented for the whole shape (ECCobj), for the

Figure 5.1: *ECC* of example binary shape (point with smallest *ECC* marked).

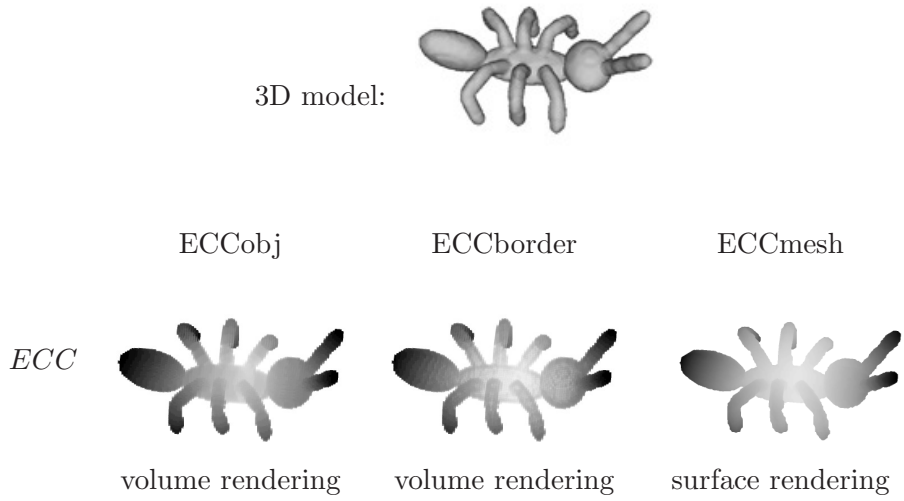


Figure 5.2: Top: 3D model of an ant. Bottom: ECCobj, ECCborder, ECCmesh (darker = higher *ECC* value).

border/boundary voxels (ECCborder), and the surface mesh (ECCmesh). Figure 5.3 shows the difference between ECCobj and ECCborder, both using distances computed in the 3D volume.

5.1.2 Eccentricity histogram matching

To match two shapes we first create a shape descriptor for each of them and then match these descriptors to obtain a similarity measure.

ECC histogram descriptor. The basic building block for our shape descriptor is the histogram $\mathbf{h}$ of the eccentricity transform *ECC* of the shape $\mathcal{S}$. We use k bins for the histogram.

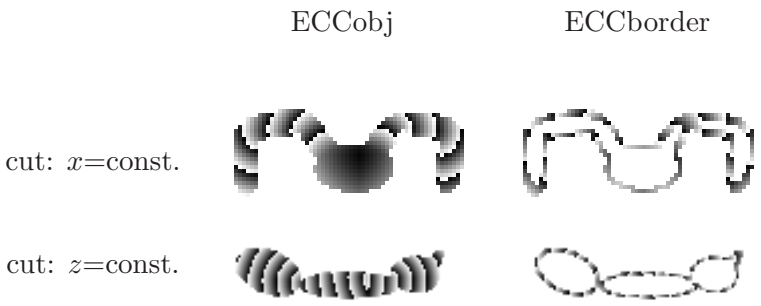


Figure 5.3: Comparison between the two volume computations of ECC: ECCobj and ECCborder.

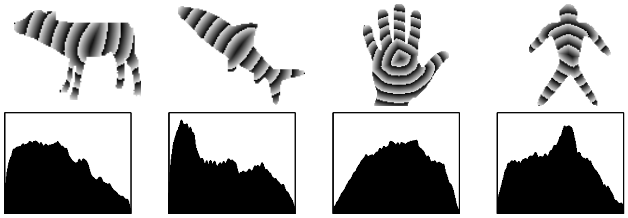


Figure 5.4: Top: ECCobj2D for some 2D shapes. Bottom: corresponding histograms.

The histogram descriptor is the vector $\mathbf{h} \in \mathbb{R}^k$ defined by: $\forall i = 1, \ldots, k$

$$\mathbf{h}(\mathcal{S}, i) = \frac{1}{|\mathcal{S}|} \# \left\{ \mathbf{p} \in \mathcal{S} \mid \frac{i-1}{k} \leqslant \frac{ECC(\mathcal{S}, \mathbf{p}) - m}{M - m} < \frac{i}{k} \right\},$$

where $|\mathcal{S}|$ is the number of pixels/voxels in $\mathcal{S}$, and m and M are the smallest, respectively largest eccentricity values taken over $\mathcal{S}$. A discussion about choosing the number of bins k follows. The obtained histogram contains only bins for the values actually existing in the eccentricity transform i.e. from minimum to maximum eccentricity, and the sum of all bins is 1. Figures 5.4 and 5.5 show examples of histograms for 2D and 3D shapes with different geometric features. We note that the histogram $\mathbf{h}$ is invariant under Euclidean transformations, scaling and isometric bending of $\mathcal{S}$ (Figure 5.6 shows examples).

Comparison of histograms. To match the descriptors of the two shapes $\mathcal{S}$ and $\tilde{\mathcal{S}}$, it is necessary to compute the distance between histograms. Let $\mathbf{h}, \tilde{\mathbf{h}} \in \mathbb{R}^k$ be the two histograms of $\mathcal{S}$ and $\tilde{\mathcal{S}}$ computed as above. We propose to use the simple ℓ^2-norm defined by

$$\delta(\mathbf{h}, \tilde{\mathbf{h}}) \stackrel{\text{def.}}{=} \sqrt{\sum_{i=1}^{k} (\mathbf{h}(\mathcal{S}, i) - \tilde{\mathbf{h}}(\mathcal{S}, i))^2}. \tag{5.1}$$

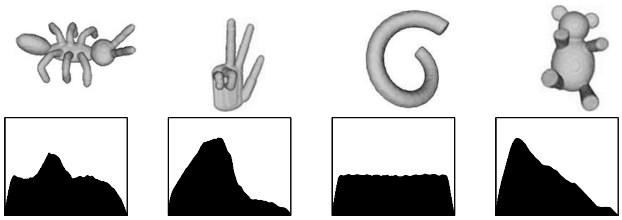


Figure 5.5: Top: example 3D shapes. Bottom: corresponding ECCobj histograms.

One could use more elaborate metrics such as the χ^2 statistic [Belongie 02, Pearson 1900] or those defined in [Osada 02], but we found in numerical experiments that all these metrics give results similar to δ, which is the easiest and fastest to compute (discussion follows in Section 5.1.5).

The dissimilarity $\Delta(\mathcal{S}, \tilde{\mathcal{S}})$ is computed between two shapes $\mathcal{S}$ and $\tilde{\mathcal{S}}$ as the distance of their histogram descriptors:

$$\Delta(S, \tilde{S}) \stackrel{\text{def.}}{=} \delta(\mathbf{h}, \tilde{\mathbf{h}}). \tag{5.2}$$

5.1.3 Characteristics of ECC histograms.

The histogram of the ECC characterizes the compactness of the shape (e.g. a flat histogram characterizes a very elongated shape, a histogram with monotonically decreasing values characterizes a rather compact shape). Figure 5.6 shows examples of eccentricity histograms for basic shapes, with and without holes and articulation.

The histogram of the ECC of a simple open curve[2] $\mathcal{S}_a$ with length $l = d(\mathbf{e}_1, \mathbf{e}_2)$ (Figure 5.7(a)), is flat with a possibly smaller value in the first bin. The continuous formula is:

$$\mathbf{h}(\mathcal{S}_a, i) = \begin{cases} \frac{1}{l} & \text{if } i = \min(ECC(\mathcal{S}_a)) \\ \frac{2}{l} & \text{if } i > \min(ECC(\mathcal{S}_a)) \end{cases},$$

where $\min(ECC(\mathcal{S}_a)) = d(\mathbf{e}_1, \mathbf{c}) = d(\mathbf{e}_2, \mathbf{c})$.

Consider $\mathcal{S}_b$ obtained by adding to $\mathcal{S}_a$ a simple open curve of length $d(\mathbf{q}_1, \mathbf{q}'_3) < l/2$ connected at the point $\mathbf{q}_1$ (Figure 5.7(b)). Let $\mathbf{q}_3 \in \mathcal{S}_b$ s.t. $d(\mathbf{q}_1, \mathbf{q}_3) = d(\mathbf{q}_1, \mathbf{q}'_3)$ and $d(\mathbf{q}_3, \mathbf{e}_1) = d(\mathbf{q}'_3, \mathbf{e}_1)$. For the points with eccentricity between $d(\mathbf{e}_1, \mathbf{q}_1)$ and $d(\mathbf{e}_1, \mathbf{q}_3)$, the eccentricity histogram of $\mathcal{S}_b$ has increased by 50% (there is one additional point having each of the values in the domain). A shape without cycles (e.g. $\mathcal{S}_a$, $\mathcal{S}_b$, $\mathcal{S}_c$) has only one center point (ECC minimum) and the

[2]The term *curve* is used to denote a one dimensional and continuous manifold, and includes both straight and non-straight lines.

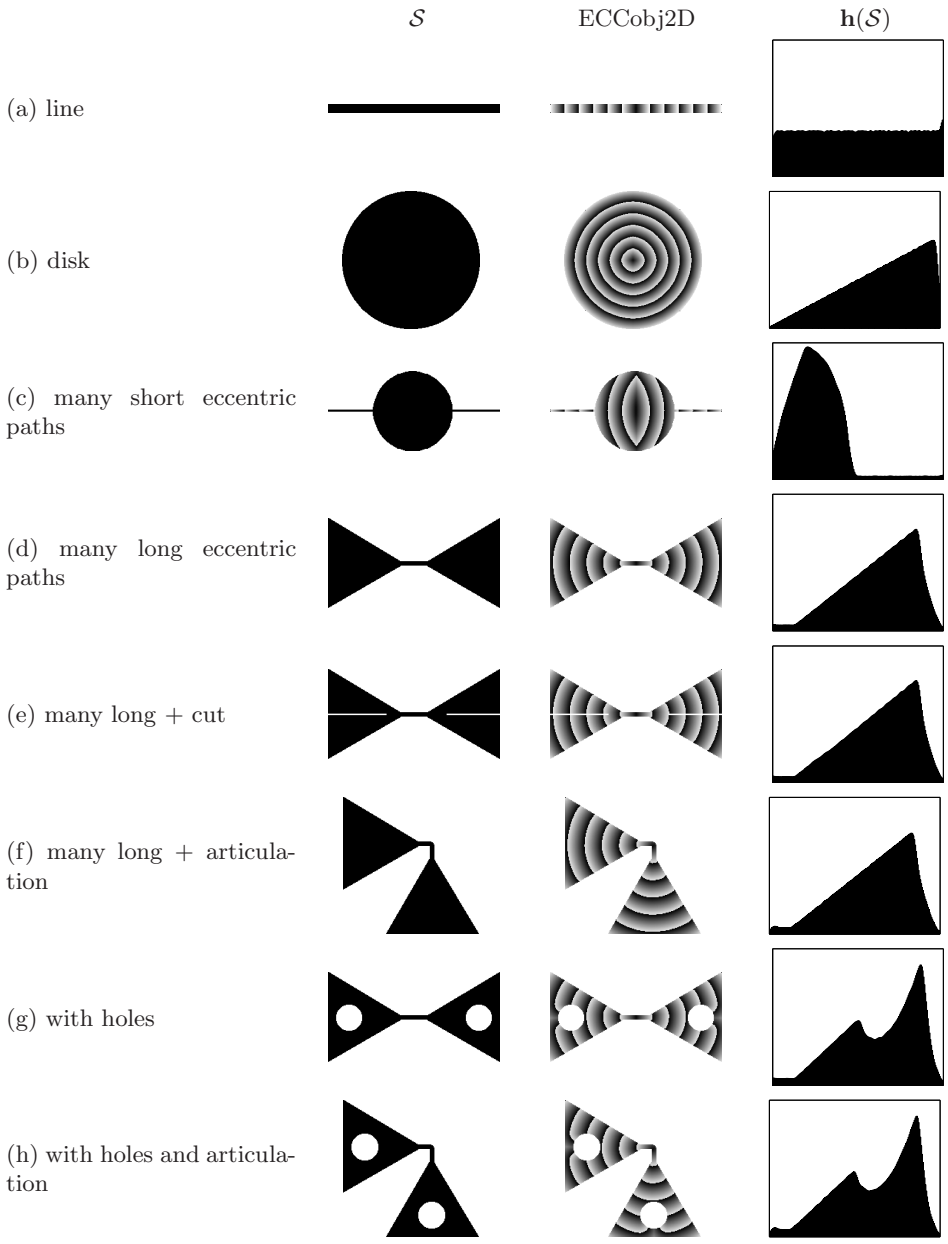


Figure 5.6: Basic shapes and their eccentricity histograms.

histogram value for the center is always one. All other histogram values can be changed by adding branches as above.

A possibility to change the histogram value for the center is to introduce cycles. Consider $\mathcal{S}_d$ obtained by adding a simple open curve $\mathbf{q}_1\mathbf{c}'\mathbf{q}_2$ of length $\lambda(\mathbf{q}_1, \mathbf{q}_2)$ to $\mathcal{S}_a$ (Figure 5.7(d)). The length $d(\mathbf{e}_1, \mathbf{e}_2)$ is kept the same and $\mathbf{q}_1\mathbf{q}_2$ has the same length if going over $\mathbf{c}$ or $\mathbf{c}'$. Also

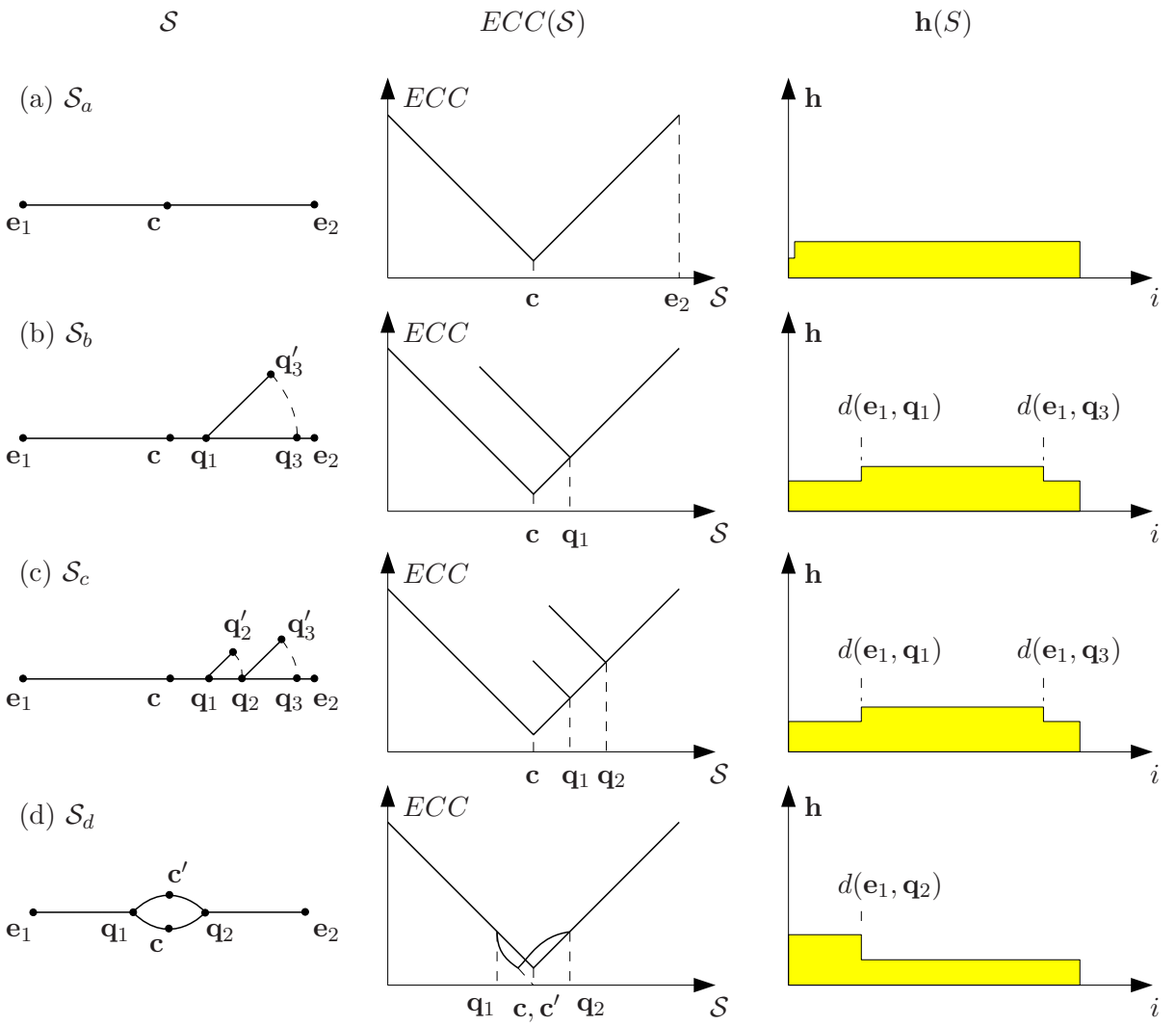


Figure 5.7: Behavior of ECC histogram for basic changes in the shape. Column $\mathcal{S}$: where possible, straight lines where used for illustration, but only the length of the curves is relevant, not whether they are straight or not.

$d(\mathbf{e}_1, \mathbf{c}) = d(\mathbf{e}_1, \mathbf{c}') = d(\mathbf{e}_1, \mathbf{e}_2)/2$. Two center points exist ($\mathbf{c}$ and $\mathbf{c}'$), and for the eccentricity values $[d(\mathbf{c}, \mathbf{e}_1), d(\mathbf{q}_2, \mathbf{e}_1))$ there are two additional points. If $\mathbf{q}_1 \to \mathbf{c}$, $\mathbf{q}_1 \neq \mathbf{c}$ (implies $\mathbf{q}_2 \to \mathbf{c}$, $\mathbf{q}_2 \neq \mathbf{c}$), only one additional point will exist for the eccentricity values $[d(\mathbf{c}, \mathbf{e}_1), d(\mathbf{q}_2, \mathbf{e}_1))$.

For a given histogram, the steps used to create $\mathcal{S}_b$ and $\mathcal{S}_d$, can be iterated to grow the continuous shape (for geodesics computed along thin lines). For discrete shapes, the number of points is finite[3], which limits the number of curves that can be put close to each other and

[3]Depends on the discretization and maximum shape size.

not intersect. With the maximum shape size (number of pixels/voxels) and the number of bins k fixed, not all (real valued) histograms can result as ECC histograms (it can also be seen as a discretization problem, the lower the resolution/maximum size, the higher the dependence between neighboring histogram bins).

A histogram has a smaller dimension (1D) than the shape and a whole class of shapes is projected into the same histogram. Two shapes $\mathcal{S}$ and $\tilde{\mathcal{S}}$ with the same eccentricity histograms satisfy $\Delta(S, \tilde{S}) = 0$, and are thus considered to be the same according to our recognition algorithm. Consider $\mathcal{S}_c$ in Figure 5.7(c) obtained from $\mathcal{S}_a$, similar to $\mathcal{S}_b$, but with two curves s.t. $d(\mathbf{q}_1, \mathbf{q}'_2) = d(\mathbf{q}_1, \mathbf{q}_2)$, $d(\mathbf{q}_2, \mathbf{q}'_3) = d(\mathbf{q}_2, \mathbf{q}_3)$, and $d(\mathbf{q}_1, \mathbf{q}_3)$ is equal in both $\mathcal{S}_b$ and $\mathcal{S}_c$. The two shapes $\mathcal{S}_b$ and $\mathcal{S}_c$ have the same eccentricity histogram and cannot be differentiated using only that. One could say that eccentricity histograms are influenced by the structure of shapes (as new branches change the histogram), but they do not uniquely characterize it.

On the other side, the descriptor is highly compact, which is an advantage for real time retrieving and low memory devices, it is invariant under many natural deformations, it can handle shapes without as well as with holes (Figure 5.6 (g) and (h)), and gives good results comparable to many state of the art methods (experiments follow).

5.1.4 Matching experiments in 2D and 3D

This section shows results on popular benchmarks and comparison with state of the art methods. When comparing the results, keep in mind that the proposed method is simple and matching is fast. An approximation of the ECC can be computed for many shapes with as few as 50 distance propagations (e.g. the average number for the ECCmesh on the McGill database is 54), and determining δ between two computed descriptors (ℓ^2-norm) has practically no CPU time consumption. A single, fixed-length vector as a descriptor can be a very efficient indexing method. The approaches compared with, are more complicated requiring decomposition of shapes, alignment/correspondences of features, etc.

2D shape matching

For the experiments with 2D shapes we have used three shape databases: Kimia 25 [Sharvit 98], Kimia 99 [Sebastian 04] and MPEG7 CE-Shape-1 [Latecki 00].

A shape database is composed of q shapes $\{\mathcal{S}_i\}_{i=1}^{q}$ and each shape $\mathcal{S}_i$ has a label $L(i) \in \{1, \ldots, l_{\max}\}$. Each label value $1 \leqslant l \leqslant l_{\max}$ defines a class of shapes $\mathcal{Q}(l) = \{\mathcal{S}_i \mid L(i) = l\}$. The first columns of the three blocks of Figure 5.8 show the shapes from the Kimia 25 database, ordered by classes (such as fish, planes, rabbits, etc.). Any shape matching algorithm α assigns to each shape $\mathcal{S}_i$ a vector of best matches Φ_i, where $\Phi_i(1)$ is the shape the most similar to

$\mathcal{S}_i$, $\Phi_i(2)$ is the second hit, and so on. Depending on the benchmark, Φ_i contains all shapes including the query shape $\mathcal{S}_i$ (all 2D benchmarks presented), or leaves $\mathcal{S}_i$ out, i.e. the shape $\mathcal{S}_i$ is not matched to itself and Φ_i has $q-1$ elements (all 3D benchmarks presented).

For the Kimia 25 database $l_{\max} = 6$ and $q = 25$, and for the Kimia 99 database, $l_{\max} = 9$ and $q = 99$. The efficiency of various matching algorithms on Kimia databases is measured by the number of correct matches for each ranking position k:

$$\text{Match}_k(\Phi) \stackrel{\text{def.}}{=} \sum_{i=1}^{q} \text{sgn}(|L(\Phi_i(k)) - L(i)|), \tag{5.3}$$

where $\text{sgn}(x)$ is the sign function. Tables 5.2 and 5.3 give the value of Match_k for various shape matching algorithms.

In the case of the MPEG7 database, which contains $l_{\max} = 70$ classes with 20 images each ($q = 70 \times 20 = 1400$), the efficiency of matching algorithms is computed using the standard Bullseye test:

$$\begin{aligned} \text{Bullseye}(\Phi) &\stackrel{\text{def.}}{=} \frac{1}{20q} \sum_{k=1}^{40} \sum_{i=1}^{q} \text{sgn}(|L(\Phi_i(k)) - L(i)|) \\ &= \frac{1}{20q} \sum_{k=1}^{40} \text{Match}_k(\Phi). \end{aligned} \tag{5.4}$$

This test counts the number of correct hits (same class) in the first 40 hits. For each image there can be at most 20 correct hits and a maximum of 20×1400 hits can be obtained during the benchmark and thus $\text{Bullseye}(\Phi) \leqslant 1$. Table 5.4 gives the value of Bullseye for various shape matching algorithms.

The results of the presented approach over both Kimia 25 and Kimia 99, and over MPEG 7 are slightly below the state of the art (Inner Distance Shape Context [Ling 07], Shape Context [Belongie 02], Shock Graphs [Siddiqi 99]).

Case study - Kimia 25: Figure 5.8 shows the retrieval results for Kimia 25. The first column shows the 25 shapes $\mathcal{S}_i$. The following set of shapes forms an array, where the shape at row i and column k is $\Phi_i(k)$, the rank-k shape associated to $\mathcal{S}_i$.

The class with the best results are rabbits, followed by tools, hands, fishes, airplanes and greebles (shapes numbered 5-8 in Figure 5.8). Two questions immediately arise when looking at these results:

(1.) Why are the greebles considered to be more similar to the hands than to other greebles?

Table 5.2: The value of $\text{Match}_k(\Phi)$ (Equation 5.3) for various algorithms on the Kimia 25 database.

Algorithm α	k=1	2	3
Sharvit et. al [Sharvit 98]	23	21	20
ECCobj2D	25	20	16
Gdalyahu and Weinshall [Gdalyahu 99]	25	21	19
Shape Context [Belongie 02]	25	24	22
ID-Shape Context [Ling 07]	25	24	25

Table 5.3: The value of $\text{Match}_k(\Phi)$ (Equation 5.3) for various algorithms on the Kimia 99 database.

Algorithm α	k=1	2	3	4	5	6	7	8	9	10
Shape Context [Belongie 02]	97	91	88	85	84	77	75	66	56	37
ECCobj2D	99	87	74	67	64	49	52	45	38	33
Gen. Model [Tu 04]	99	97	99	98	96	96	94	83	75	48
Shock Edit [Sebastian 04]	99	99	99	98	98	97	96	95	93	82
ID-Shape Context [Ling 07]	99	99	99	98	98	97	97	98	94	79

Table 5.4: The value of Bullseye(Φ) (Equation 5.4) for various algorithms on the MPEG 7 database.

Algorithm α	Bullseye(Φ)
random	2.86%
ECCobj2D	44.28%
Shape Context [Belongie 02]	64.59%
ID-Shape Context [Ling 07]	68.83%

(2.) Why does a rabbit appear in so many cases when the matching has failed?

For the first question, consider the histograms of the greebles and the unoccluded hands (Figure 5.9). *The histograms are similar even though the shapes are of different classes*, e.g. the histogram of the first greeble (Figure 5.9 top-left) looks more similar to the hands, than the second and third greeble. This is due to the abstraction of a 2D shape to a 1D histogram, which, in our case, disregards certain structural properties of distances/paths (studied in detail in Section 5.1.3).

For the second question, consider the shapes in Figure 5.10 (a rabbit - $\mathcal{S}_{19}$, and two tools -

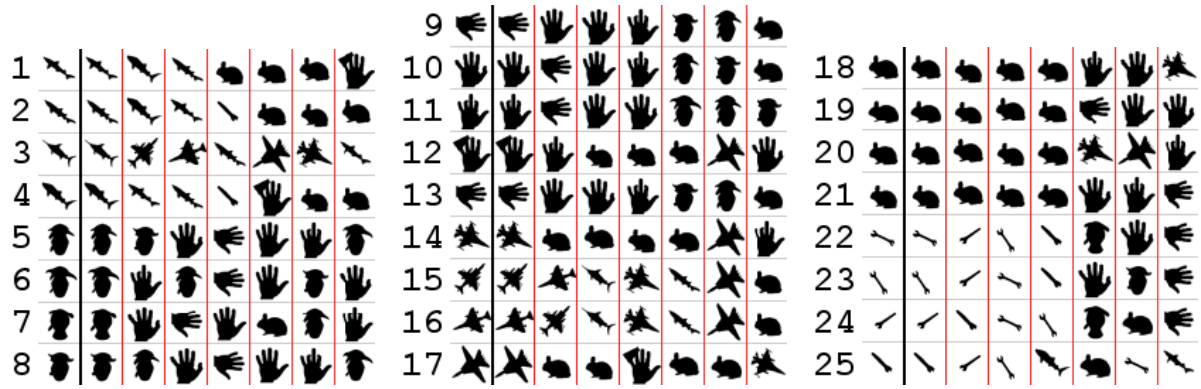


Figure 5.8: Retrieval results for the single scale descriptor on the Kimia 25 database.

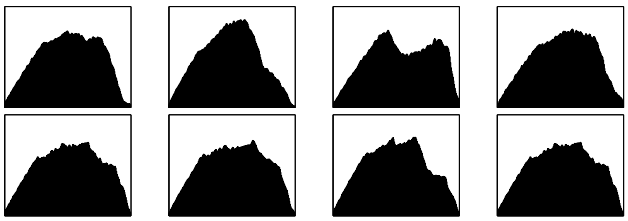


Figure 5.9: Histograms for: top: greebles, bottom: unoccluded hands.

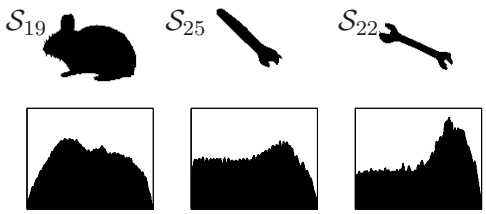


Figure 5.10: Three shapes from the Kimia 25 database and their eccentricity histograms.

$\mathcal{S}_{25}$ and $\mathcal{S}_{22}$), and the results, Φ_{25}, in row 25 of Figure 5.8. When matching $\mathcal{S}_{25}$, the rabbit has a better score than $\mathcal{S}_{22}$, even though one might say that the histograms of $\mathcal{S}_{25}$ and $\mathcal{S}_{22}$ reveal more similar distance characteristics than the histogram of $\mathcal{S}_{19}$ (see Figure 5.10). Both $\mathcal{S}_{25}$ and $\mathcal{S}_{22}$ have more long distances than medium, and short, while $\mathcal{S}_{19}$ has a peak in the medium. This is due to *typical histogram matching methods*, which are inherently low level and *fail to capture the high level context of the task*. Discussion follows in Section 5.1.5.

Geometrical properties of the shapes are well captured by our low-dimensional descriptors. For instance, elongated shapes are well separated from more compact shapes. However, more

advanced geometrical features, such as intricate structural properties are thrown away by our signature extraction. This is for instance why the class "greebles" is not separated enough from the class "hands".

3D articulated shape matching

One widely used 3D object retrieval databases is the Princeton Shape Benchmark [Shilane 04]. It contains 1,814 3D object models organized by class and is effective for comparing the performance of a variety of methods. However, the majority of the models corresponds to rigid, man-made objects. Only a limited number of shapes in the database have articulated parts. As one of the main advantages of using eccentricity is its robustness with respect to articulation, we have turned to the McGill Shape Benchmark [Zhang 05]. It contains several models from the Princeton repository and others added by the authors. The main advantage of this benchmark is that from its 455 3D shapes, 255 have significant part articulation. We show the results on the $q = 255$ shapes grouped into the $l_{\max} = 10$ classes of articulated shapes (Figure 5.11). Shapes are not matched to themselves and so Φ_i contains $q - 1$ shapes.

Three ECC based descriptors were used (Figure 5.2):

(1.) **ECCobj** - eccentricity of the whole shape (all object voxels);

(2.) **ECCborder** - eccentricity of the border/boundary voxels;

(3.) **ECCmesh** - eccentricity of the triangular mesh of the surface of the shape.

ECCborder is obtained by computing the eccentricity transform $ECC(\partial_6\mathcal{S})$, where $\partial_6\mathcal{S}$ is the 6 connected voxel boundary of $\mathcal{S}$. ECCmesh is computed on the 2D manifold defined on the boundary of the 3D shapes. ECCborder uses distance computation in the 3D volume, ECCmesh in the 2D surface. If the resolution of the shapes is increased, ECCborder and ECCmesh converge to the same value. For a similar resolution, ECCmesh needs less memory, as cells not part of the boundary do not have to be stored (e.g. interior of the shape), and it can be more accurate when approximating the eccentricity of the surface, as the computation is done on the surface itself, not on an approximation volume.

Experimental results: In the following, results of the three variants on shapes of the 10 articulated classes of the McGill Shape Benchmark are given. The notation from Section 5.1.4 is used. The following measures are considered.

$$\text{Recall}(\Phi_i, t) = \frac{1}{|\mathcal{Q}(L(i))| - 1} \sum_{k=1}^{t} \text{sgn}(|L(\Phi_i(k)) - L(i)|).$$

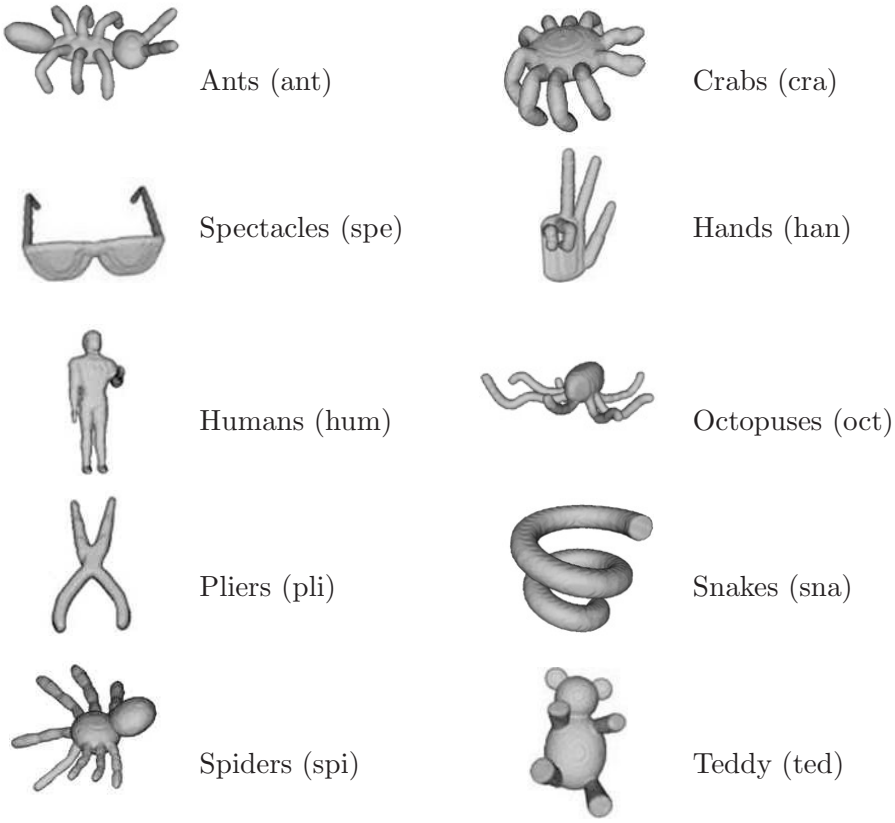


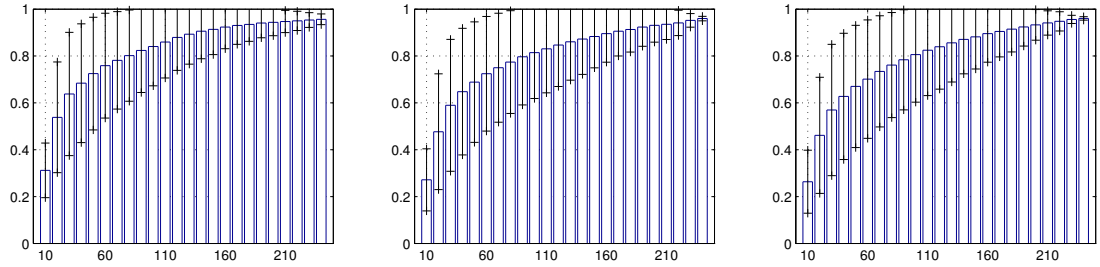
Figure 5.11: The object classes from the McGill 3D shape database having significant part articulation.

The recall computes the ratio of models in the database in the same category as the query, with indexing rank $\leqslant t$, to the total number of shapes in the same category (never including the query itself). The average results and standard deviation for several rank thresholds ($t = 10, 20, \dots$), over all classes, are given in Figure 5.12.

$$\text{AvgRank}(\Phi_i) = \frac{1}{|\mathcal{Q}(L(i))| - 1} \sum_{k=1}^{q-1} k \, \text{sgn}(|L(\Phi_i(k)) - L(i)|).$$

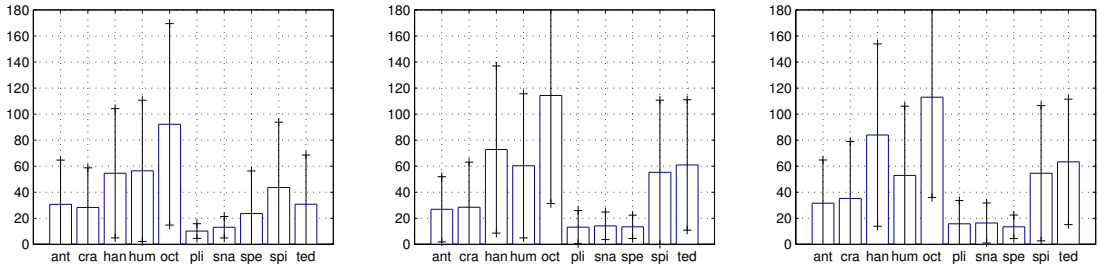
For all queries in a class, the average of the ranks of all other shapes in that class are computed. Figure 5.13 shows the average and the standard deviation of the ranks for each class (lower average is better).

Table 5.5 shows the average score for all pairs of classes. Each shape in the database is matched against all other shapes and each cell shows the average of the score (Equation 5.2)



ECCobj ECCborder ECCmesh

Figure 5.12: Recall for several rank thresholds



ECCobj ECCborder ECCmesh

Figure 5.13: Average ranks for each class. The first three letters of each class name are printed.

between all combinations of shapes of the two classes defined by the row and column.

$$\text{Precision}(\Phi_i, t) = \frac{1}{t}\sum_{k=1}^{t} \text{sgn}(|L(\Phi_i(k)) - L(i)|).$$

Precision refers to the ratio of the relevant shapes retrieved, to the total number retrieved. Figure 5.14 shows the *precision-recall* curves for each of the 10 classes. Precision-recall curves are produced by varying the parameter t. Better results are characterized by curves closer to the top, i.e. recall = 1 for all values of precision. Precision and recall are common in information retrieval for evaluating retrieval performance. They are usually used where static document sets can be assumed. However, they are also used in dynamic environments such as web page retrieval [Fawcett 06].

As can be seen in Figures 5.12, 5.13, and 5.14, and Table 5.5, ECCobj does in most cases a better job than ECCborder and ECCmesh. The recall of the three methods is comparable, with slightly better results from ECCobj. With respect to the average ranks, ECCobj does better with the hands, octopus, pliers, snakes, spiders, teddy, is worse then one of ECCborder and ECCmesh with the ants, crabs, humans, and slightly worse than both other methods with

Table 5.5: Average matching results multiplied by 100 (smaller means more similar). For each row, the first and second smallest value are printed in bold.

ECCobj	ants	crabs	hands	humans	octopus	pliers	snakes	spectacles	spiders	teddy
ants	**1.75**	5.33	3.72	3.53	7.20	2.95	**2.91**	7.25	5.43	3.72
crabs	5.33	**1.55**	3.73	3.50	3.67	3.02	4.36	3.99	3.43	**2.53**
hands	3.72	3.73	**2.30**	3.04	5.60	2.71	3.76	6.19	4.53	**2.51**
humans	3.53	3.50	3.04	**2.19**	5.03	**2.13**	3.15	5.04	3.52	2.62
octopus	7.20	**3.67**	5.60	5.03	**3.90**	5.02	6.22	4.04	4.06	4.58
pliers	2.95	3.02	2.71	2.13	5.02	**0.55**	**1.82**	4.97	3.64	2.11
snakes	2.91	4.36	3.76	3.15	6.22	**1.82**	**0.80**	5.73	4.83	3.55
spectacles	7.25	3.99	6.19	5.04	4.04	4.97	5.73	**2.24**	**3.97**	5.13
spiders	5.43	**3.43**	4.53	3.52	4.06	3.64	4.83	3.97	**2.25**	3.50
teddy	3.72	2.53	2.51	2.62	4.58	**2.11**	3.55	5.13	3.50	**1.46**

ECCborder	ants	crabs	hands	humans	octopus	pliers	snakes	spectacles	spiders	teddy
ants	**1.00**	2.45	2.16	**1.60**	3.09	1.61	2.42	5.62	2.47	1.66
crabs	2.45	**1.41**	**2.30**	3.01	3.42	2.88	3.64	6.92	3.08	2.38
hands	**2.16**	2.30	**1.94**	2.65	2.98	2.48	3.49	6.05	2.78	2.29
humans	1.60	3.01	2.65	**1.57**	3.19	**1.54**	2.16	5.12	2.54	1.93
octopus	3.09	3.42	2.98	3.19	2.97	**2.72**	3.82	4.80	**2.69**	2.80
pliers	1.61	2.88	2.48	1.54	2.72	**0.65**	1.75	4.51	2.00	**1.47**
snakes	2.42	3.64	3.49	2.16	3.82	**1.75**	**0.85**	4.86	3.21	2.44
spectacles	5.62	6.92	6.05	5.12	4.80	**4.51**	4.86	**1.67**	4.69	5.26
spiders	2.47	3.08	2.78	2.54	2.69	**2.00**	3.21	4.69	**1.76**	2.07
teddy	1.66	2.38	2.29	1.93	2.80	**1.47**	2.44	5.26	2.07	**1.45**

ECCmesh	ants	crabs	hands	humans	octopus	pliers	snakes	spectacles	spiders	teddy
ants	**0.97**	2.34	1.94	1.66	2.46	**1.40**	2.36	4.78	1.93	1.47
crabs	2.34	**1.47**	2.52	3.05	3.09	2.88	3.67	6.41	2.75	**2.34**
hands	**1.94**	2.52	**1.98**	2.69	2.62	2.34	3.40	5.30	2.40	2.17
humans	1.66	3.05	2.69	**1.48**	3.03	**1.60**	1.91	4.56	2.48	2.07
octopus	2.46	3.09	2.62	3.03	2.61	2.39	3.54	4.65	**2.33**	**2.34**
pliers	**1.40**	2.88	2.34	1.60	2.39	**0.70**	1.78	3.92	1.71	1.44
snakes	2.36	3.67	3.40	1.91	3.54	**1.78**	**0.98**	4.00	2.95	2.56
spectacles	4.78	6.41	5.30	4.56	4.65	**3.92**	4.00	**1.66**	4.49	4.71
spiders	1.93	2.75	2.40	2.48	2.33	1.71	2.95	4.49	**1.50**	**1.68**
teddy	1.47	2.34	2.17	2.07	2.34	**1.44**	2.56	4.71	1.68	**1.37**

the spectacles. None of the three variants produces an average class rank higher than 50%. All three methods have the smallest average class distance (highest similarity) correct for 8 out of 10 classes, with ECCobj having the correct class as the second smallest one for the other two, the humans and octopus (see Table 5.5). A discussion considering the differences between the three ECC variants follows at the end of this section.

Figure 5.14 shows comparative precision-recall results of ECCobj, ECCborder and ECCmesh,

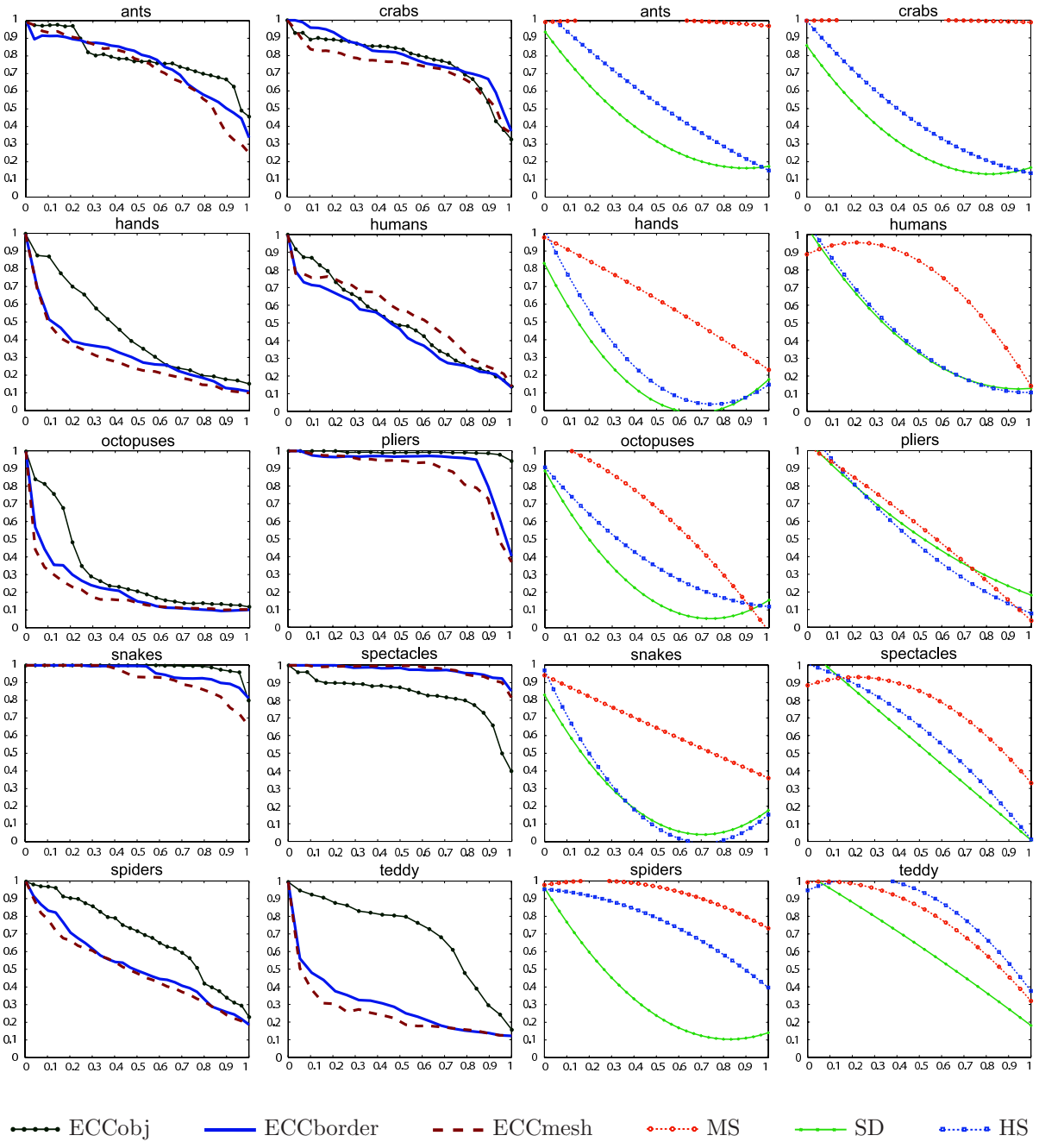


Figure 5.14: Precision-recall for the ten classes. Left two columns: ECCobj, ECCborder, ECCmesh. Right two columns (image taken from [Siddiqi 07], with kind permission of Springer Science and Business Media): results of three other methods on the same database: medial surfaces (MS) [Siddiqi 07], harmonic spheres (HS) [Kazhdan 03], and shape distributions (SD) [Osada 02]. Precision: horizontal axis, recall: vertical axis. (Best visualized in color.)

and three other methods:

- medial surfaces (MS) [Siddiqi 07];
- harmonic spheres (HS) [Kazhdan 03];
- shape distributions (SD) [Osada 02].

ECCobj, ECCborder, and ECCmesh are comparable, except for the teddy bears, where ECCobj is superior to the other two. The best results (higher precision vs. recall) are reached by the *ECC* variants for the snakes, by MS for the ants, and HS and SD for teddy. For these best results the MS has the best precision-recall followed by the *ECC* based methods, followed by HS and SD. The worst results are achieved by ECCobj, ECCborder, and ECCmesh for the octopus, MS for the pliers, and HS and SD for the hands. The results of MS for the pliers are superior to ECCobj, ECCborder and ECCmesh for the octopus, which are in turn superior to the HS and SD for the hands. In comparison to all other three methods (MS, HS, SD), the eccentricity based methods score better on the pliers, spectacles and snakes.

The differences in the results of ECCobj vs. ECCborder and ECCmesh, can be linked to the compactness of the shapes and the width of the joints. The variation of the geodesic distances is larger when computed on the "skin" (boundary) compared to computed inside a joint (smaller). In the case of 2D shapes the eccentricity of the boundary is a constant. In 3D it manages to capture some of the properties of the shape, but it looks more unstable. The eccentricity transform of a simply connected volume has in most of the cases a single stable center (minimum), while the eccentricity transform of its boundary will have a disconnected center or at least one with a more complex structure.

Discussion: 2D and 3D

The computed shape similarities are *robust with respect to scaling, rotation, and part articulation*. The matching results are good, especially when considering the *straightforward approach*. In contrast, the most efficient shape matching algorithms [Ling 07, Siddiqi 07] are more complicated and require extraction of salient features and local signatures that need to be aligned or registered.

The *major current limitations* of our approach include: (1) Eccentricity *histograms do not capture the topology* of the shape and thus histograms of different shapes can be very similar. (2) *Histogram "matching"* (whether using the ℓ^2-norm or more sophisticated methods) is inherently low level and *does not consider the higher level context* in which it is applied.

Connectivity of the isoheight lines/surfaces of the eccentricity transform does capture the

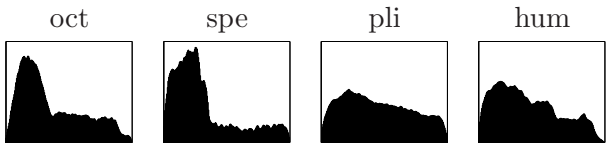


Figure 5.15: Similar ECCobj histograms corresponding to 3D shapes (objects) of different classes.

part structure of a shape [Ion 08e], but the histograms "throw away" this information. (Figure 5.15 shows two pairs of similar histograms belonging to 3D objects of different classes.)

Compared to other approaches (e.g. [Siddiqi 07]), one can identify the aspects discussed above (see Figure 5.14 and Table 5.5). For classes with simple topology (e.g. snakes and spectacles), the results are very good. For classes where part decomposition and structure play an important role (e.g. octopus v.s. spiders and crabs), the discrimination capabilities are reduced.

5.1.5 Parameters and improvements

The number of bins for the histogram: The approach has only one parameter, the number of bins k, of the histograms $\mathbf{h}$. In experiments, we used $k = 200$, which was chosen based on a few initial trials on a smaller set of shapes.

As the shapes are discrete, the number of distance values of the ECC is finite. Let $\mathbf{h}^c$ be the ordered set of eccentricity values computed for a shape $\mathcal{S}$, i.e. each distinct value that exists in the ECC of the discrete shape $\mathcal{S}$. We have $\min(\mathbf{h}^c)$ equal to the ECC value of the center (minimum ECC) and greater or equal to half the diameter of the shape ($\max(\mathbf{h}^c) = \max(ECC)$). The largest distance between two neighboring (grid) points is equal to one (shapes are required to be 4 respectively 6 connected). For the ECC histogram of a shape not to contain any empty bins, the number of bins k has to satisfy:

$$k \leqslant \max(ECC(\mathcal{S})) - \min(ECC(\mathcal{S})).$$

Depending on the shape, k could be much higher and still have no empty bins in $\mathbf{h}$, e.g. for $\mathcal{S}$ a disk with radius r in $\mathbb{Z}^2$ and the Euclidean distance, there are more distinct values than r (consider the discrete approximation of the Euclidean circle). An absolute upper bound is $k = |\mathcal{S}|$. If this number is exceeded, there will be empty bins in $\mathbf{h}$.

As k decreases, the description capability of the histogram also decreases. In the extreme case, **a single bin** would just contain $|\mathcal{S}|$, and for the normalized histogram it would contain the value 1. **Two bins** can give the equivalent of a simple compactness measure (similar to the *circularity ratio*, which relates the area of the shape to the area of the circle with the

same diameter). **Three bins** could be considered as a relative measure for short/long/medium distances and can characterize more than the simple compactness measure. A higher number of bins increases the dimension of the space in which distances are computed and gives more flexibility in the relations, e.g. in 2D there can be maximum 3 points s.t. they are pairwise at the same distance (equilateral triangle), and this number increases to 4 in 3D (regular tetrahedron).

Assuming that shapes from the same class have similar histograms, given the number of classes (vertices) and the required relations (weighted edges), a lower bound for the number of bins is equal to the smallest dimension in which the classes can be embedded s.t. the weights of the edges corresponding to the distance between the vertices. If the variation inside classes increases, the number of classes that can be discriminated will decrease.

Describing topology: One of the problems identified in Section 5.1.3 and during the experiments (Sections 5.1.4 and 5.1.4) is that the histograms do not capture the exact structure of the shape. Classical methods to describe the topology of a shape (e.g. Reeb graphs, [Reeb 64], and homology generators, [Munkres 93]) fail to capture the geometrical aspects. An approach to deal with this problem is presented in [Aouada 08]. To describe a shape, two descriptors are used: a geometric one, based on the *Global Geodesic Function* (GCF), which is defined for a point as the sum of the geodesic distances to all points of the shape multiplied by a factor, and a topological one, the Reeb graph of the shape using the GCF as the Morse function.

Initial steps in combining the eccentricity transform with Reeb graphs have been presented in [Ion 08e].

A better histogram matching: The problem of having a matching function that is aware of the context in which it is applied can be approached in two ways: use expert knowledge about the context to create an algorithm that considers the proper features, or learn the important features by giving a set of representative examples. In [Yang 06a, Yang 06b], a survey of current distance metric learning methods is given. The purpose of distance metric learning is to learn a distance metric for a space, from a given collection of pairs of similar/dissimilar points. The learned distance is supposed to preserve the distance relation among the training data. Example training data would be: $\mathcal{S}_1$ is more similar to $\mathcal{S}_2$ than to $\mathcal{S}_3$. The result is a distance function that would replace the ℓ^2-norm in Equation 5.1 with a new measure which is adapted to the task of computing the distance of eccentricity histograms as given by the training examples.

Higher dimensional data: 4D data has started to be available in the medical image processing community (e.g. 3D scans of a beating heart, over time). The presented method is general and should work in any metric space. This includes 4D, but also gray scale images (e.g. gray

values can determine the distance propagation speed in the respective cells).

5.1.6 Conclusion

We have presented a method for matching 2D and 3D shapes using the eccentricity transform. Descriptors are normalized histograms of the eccentricity transform, compact, and easy to match. The method is straight-forward but still efficient, with experimental results comparable to more complex state of the art methods. Experimental results on popular 2D and 3D shape matching benchmarks are given, with computation on binary 2D images, binary 3D voxel shapes, and 3D triangular meshes. The experiments are preceded by a detailed analysis of the properties of the descriptor and followed by in depth discussion of results, parameters, and improvement possibilities.

5.2 A Non-rigid Object Centered Coordinate System

This section presents a second application. It uses the eccentricity transform to map a coordinate system to an articulated shape, with the purpose of addressing the corresponding point (or a close one) in other instances of the same shape. It is mainly motivated by observations like: "one might change his aspect, alter his pose, but the wristwatch is still located in the same place on the hand". The results in this section have been previously published in [Ion 08b, Ion 08c].

Most shape matching methods output a similarity value (e.g. [Felzenszwalb 03, Ozcanli 07] [Ion 07b, Felzenszwalb 07]), some also give correspondences of the used signature, usually border points/parts [Ling 07, Belongie 02, Siddiqi 99], but finding all point correspondences based on the obtained information is in most of the cases not straightforward.

For correspondences of all points of the shape, the task is similar to the non-rigid registration problem used in the medical image processing community [Crum 04]. Differences include the usage of gray scale information to compute the deformation vs. the usage of a binary shape and, the registration of a whole image (in most cases) vs. the registration of a (in this approach) connected 2D shape. In [Felzenszwalb 03], a triangulation of the shape is used as a model, which could be used to find corresponding points, but an a priori known model is still needed. In the surface parametrization community a coordinate system for shapes is defined [Brechbühler 95], but articulation is not considered. In [Kambhamettu 94], for small variations, correspondences between points of 3D articulated shapes are found. Recently shape matching has also moved toward decomposition and part matching, e.g. [Ozcanli 07], mainly due to occlusions, imperfect segmentation or feature detection.

We use the eccentricity transform (Chapter 3) as a basis for a 2D polar like coordinate system. To support the coordinate mapping, shapes are decomposed into connected parts. The

following sections describe the proposed methods, with experiments given, and finish with a short discussion.

5.2.1 ECC isoheight lines - decomposition

The *level set* (Section 3.4.3) of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, corresponding to a value h, is the set of points $\mathbf{p} \in \mathbb{R}^n$ s.t. $f(\mathbf{p}) = h$. If $n = 2$ and $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, the connected components of the level sets of f form one dimensional manifolds called *isolines*. If $n = 3$ and $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, they are called *isosurfaces*.

A *level set* of the ECC of $\mathcal{S}$ is the set $LS(e) = \{\mathbf{q} \in \mathcal{S} \mid ECC(\mathcal{S}, \mathbf{q}) = e\}$, with $e \in [\min\{ECC(\mathcal{S}, \mathbf{p})\}, \max\{ECC(\mathcal{S}, \mathbf{p})\}]$. For $\mathcal{S} \in \mathbb{R}^2$, $LS(e)$ can be a closed curve or a set of disconnected open curves. The connected components of $LS(e)$ are called *isoheight lines*, $IL \subseteq LS(e)$, IL connected.

$HD(\mathcal{S}) = \{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ is a *a decomposition of $\mathcal{S}$ based on the connectivity of the ECC isoheight lines* (Figure 5.17) if: HD is a partition of $\mathcal{S}$ into simply connected regions; $\forall \mathbb{R}_i$ and $\forall e \in [\min\{ECC(\mathcal{S}, \mathbf{p})\}, \max\{ECC(\mathcal{S}, \mathbf{p})\}] \Rightarrow \mathbb{R}_i \cap LS(e)$ is connected; the number n of regions is **minimal**. $HD(\mathcal{S})$ exists for any connected shape $\mathcal{S}$.

The top level G_t of the graph pyramid (Section 2.1.10) created by Algorithm 9 is a region adjacency graph describing the topology of the decomposition $HD(\mathcal{S})$. Edges of G_t are oriented from regions with lower eccentricity to regions with higher eccentricity. Each vertex contains the length of the longest isoheight line in its receptive field.

If $\mathcal{S}$ is simply connected, the obtained region adjacency graph (top level of the pyramid) is a tree (Theorem 7.9 in [Klette 04]), with the receptive field of the root vertex containing the (unique) center pixel. Such a decomposition can be done for other transforms also (e.g. the $DT(\mathcal{S}, \mathbf{p})$). The eccentricity transform is used because its center is a robust starting point [Kropatsch 06].

5.2.2 The non-rigid coordinate system

A system of *curvilinear coordinates* [Weisstein 02] is a system composed of intersecting surfaces. If all intersections are at angle $\pi/2$, then the coordinate system is called *orthogonal* (e.g. polar coordinate system). If not, a *skew* coordinate system is formed. To define a planar system of curvilinear coordinates, two classes of curves need to be defined - one for each coordinate. Any defined coordinates identify one curve of each class which intersect at a unique point.

The proposed coordinate system is intuitively similar to the polar coordinate system, but forms a skew coordinate system. We focus on simply connected shapes and their properties. The decomposition of non simply connected shapes is much more complex (general graph with

Algorithm 9 HD - Decompose $\mathcal{S}$ based on the ECC isolines

Input: Discrete shape $\mathcal{S}$.

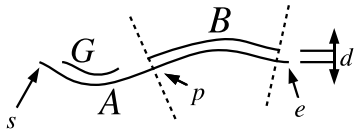
1: $iECC = \lfloor ECC(\mathcal{S}) \rfloor$ /*at least 8 connected isolines*/
2: $G_0 \leftarrow$ oriented neighborhood graph of $iECC$ /* pixels with same iECC connected, G_0 planar, orient from small to high iECC*/
3: $k \leftarrow 0$
4: $\forall v \in V_0$ **do**
$v.maxlength \leftarrow 1$, $v.ecc \leftarrow [ECC(v), ECC(v)]$ /* init max length of isolines and ecc. interval*/
5: **repeat**
6: $A \leftarrow \{e = (v, w) \in E_k \mid v.ecc = w.ecc\}$ /* merge isoheight line parts*/
7: $A \leftarrow A \cup \{e = (v, w) \mid \text{out-deg}(v) = \text{in-deg}(w) = 1 \text{ and closed}(v)=\text{closed}(w)\}$
/* closed(v)=true **iff** receptive field of **v** contains only closed isolines*/
8: **if** $|A| > 0$ **then**
9: $K \leftarrow$ CK as subset of A
/*choose optimal subset of A with e.g. MIS [Kropatsch 05] */
10: $G_{k+1} \leftarrow \text{contract}(G_k, K)$ /* contract and simplify*/
11: $\forall v \in V_{k+1}$ **compute** $v.maxlength$, $v.ecc$ from G_k /* use reduction window*/
12: $k \leftarrow k + 1$
13: **end if**
14: **until** $|A| = 0$
15: $t \leftarrow k$

Output: Graph Pyramid $P = \{G_0, \ldots, G_t\}$.

cycles, etc.) and more complex algorithms are required. Note that θ is not really an angle, just denoted intuitively so. The *radial coordinate*

$$\boldsymbol{r}(\mathbf{p}) = ECC(\mathcal{S}, \mathbf{p}) - \min\{ECC(\mathcal{S}, \mathbf{p})\} \tag{5.5}$$

is a linear mapping from the eccentricity value and the *angular coordinate* θ is mapped to the isoheight lines of the ECC, based on the structure of the shape.



The figure above shows three adjacent isoheight lines (A, B, G) of different regions. A has eccentricity e, and B, G have $e + k$. If $k \to 0$ then $d \to 0$, and maximum smoothness of θ is achieved when each point of B has the same θ as his projection on A. This assumption puts the values θ for A and B into relation. An approximation is to project the endpoints of B onto A, to find their θ values, and interpolate along B:

$$\theta_1' = \theta_1 + \frac{(\theta_2 - \theta_1) \int_s^p dl}{\int_s^e dl} \tag{5.6}$$

Algorithm 10 *CtoP* - Assign θ to $\forall v \in G$

Input: $G = (V, E)$ from Algorithm 9, vertex v, interval $[\theta_1, \theta_2]$.

1: $v.\theta_1 \leftarrow \theta_1, v.\theta_2 \leftarrow \theta_2$
2: $A \leftarrow$ isoheight line of v with highest ECC.
3: **for all** $e = (v, v_o) \in E$ /**all edges oriented away**/ **do**
4: $B \leftarrow$ isoheight line of v_o with lowest ECC.
5: $[\theta'_1, \theta'_2] \leftarrow$ project B to A and compute from $[\theta_1, \theta_2]$ (Equation 5.6)
6: call $CtoP(G, v_o, [\theta'_1, \theta'_2])$
7: **end for**

Output: G, with θ intervals $[v.\theta_1, v.\theta_2]$ for each region

The obtained relation can be used to control the smoothness of θ along region boundaries (having θ for the "last" isoline of a region, determine θ for the first isoline of the adjacent region).

The root vertex of G_t from Section 5.2.1, contains only closed isoheight lines and is the only such vertex. Its associated θ interval is 2π. Other vertices have an "input interval" and 0 or more "output intervals" (edge orientation in G). Smoothness along region boundaries is assumed as above, and *intervals of θ inside each region are kept constant*. Algorithm 10 assigns the θ intervals to each vertex. The parameters are the top level of the pyramid from Algorithm 9, the root vertex of G_t, and $[0, 2\pi]$. This approach works only with real valued θ, as two isoheight segments of the same region can contain a different number of pixels and still get the same interval assigned.

For the origin of θ, a path connecting the center (minimum eccentricity) with a point having the maximum eccentricity can be used. This path is called the *zero path*. (the zero path does not have to be a part of the diameter, as the diameter does not always pass through the center). It is used in the inner most region (root vertex of G_t) to set the 0 for the θ of each isoheight line. Outside this region, linear interpolation is used (Equation 5.6). The point with maximum ECC can be selected using any shape orientation method (e.g. [Zunic 06]) - taking into consideration the possible deformations would be optimal.

5.2.3 Experiments

Figure 5.17 shows the results of Algorithm 9 and 10, and Equation 5.5 and 5.6 for the two hands. The jagged isoheight lines of θ are due to the smoothness/roughness of the shape boundary i.e. curvature of the shape boundary at the endpoints of isoheight lines, and partly due to the simple implementation (point projection by closest point search and integral along line estimation by sum of line segment lengths for Equation 5.6, etc.).

To get a feeling of the "stability" of the mapping w.r.t. articulation we have applied the algorithms on the shapes in Figure 5.16. A pattern was laid on each hand - the *source*, and copied to the other one - the *destination*, by finding for each pixel $p_d(r_d, \theta_d)$ of the *destination*

Figure 5.16: Shapes used in experiments and their eccentricity transform.

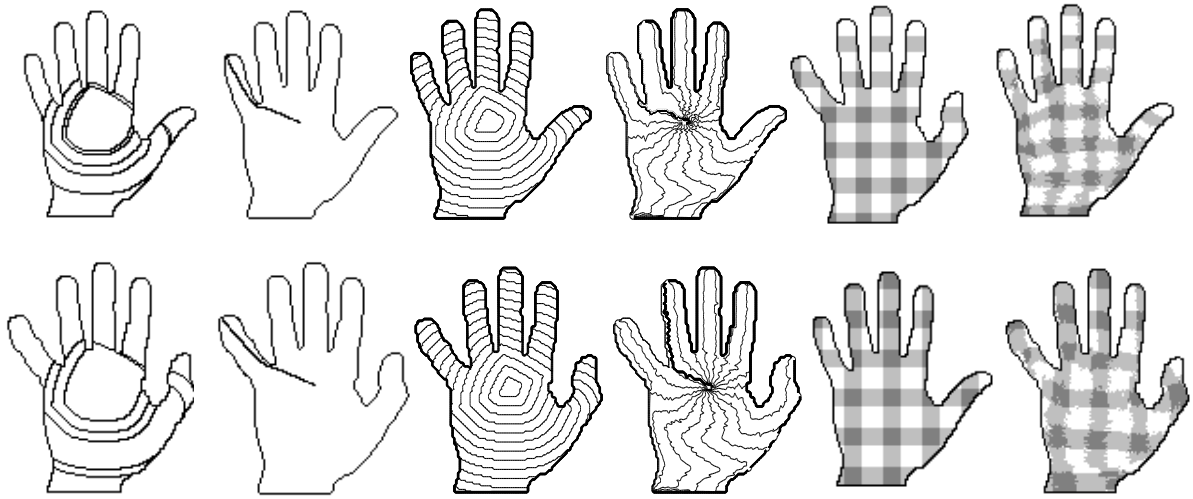


Figure 5.17: Results for the shapes in Figure 5.16.

the "closest" pixel $p_s(r_s, \theta_s)$ in the *source*.

The local variation of θ is not constant over the whole shape, making the ℓ^2-norm between point coordinates not the best option for finding the closest pixel to a given point $p_d(r_d, \theta_d)$. To avoid compensating for this variation, a two step approach is used. First, normalize r in both shapes to $[0, 1]$. This makes finding $ecc_d \rightarrow r \rightarrow ecc_s$ a linear scaling problem. $L \leftarrow (ecc_s \leq ECC(source) < ecc_s + 1)$ gives at least 8 connected isoheight lines of r. Second, the pixel of L which minimizes $|\theta_d - \theta_s|$ is chosen. The results are promising (see Figure 5.17) with the texture of the "articulated" finger being nicely copied from one shape to the other i.e. points are copied to their corresponding region in the articulated version of the shape.

The noise like errors on the pattern are due to the approximations mentioned above and to using "nearest point" for finding the color of each pixel when copying the pattern (instead of interpolating gray values). Errors on the boundaries of fingers are due to certain coordinates not existing in both shapes. The more global perturbation (palm of the hands in Figure 5.17) is

mainly due to the slightly different position of the centers and isoheight line shape. Improvements can be made by considering both shapes when mapping the coordinates to them, or by a more complex method for finding corresponding points. Finding a matching between the regions of the decomposition of the two shapes is an important step and is planned in the future.

5.2.4 Conclusion

This section presented a framework for using the eccentricity transform to map a polar-like coordinate system onto a non-rigid binary shape and find corresponding points between two shapes. Promising initial results were shown. More global decisions will provide smoother angular isoheight lines, and additional correspondences between part structures can help to solve failed correspondences.

5.3 Chapter Summary

This chapter presented two example applications to motivate the study of the eccentricity transform in Chapters 3 and 4. First, a shape descriptor is build from the histogram of the eccentricity transform of the respective shapes. The obtained shape descriptors are used to match 2D and 3D shapes. A second application, a shape-centered coordinate system for shapes undergoing articulation was given. The mapped coordinate system builds on the eccentricity transform and allows addressing corresponding points in different poses of the same simply connected 2D shape.

6 Epilogue

6.1 Conclusion

As a property of an object, *shape* characterizes the objects spatial form and identifies the points that are part of the object. Shape is both complex and structured, and allows an object to be identified [Pizlo 08]. Image transforms are used to extract high abstraction level information from the low abstraction level information contained in an image. The purpose is to extract significant information at higher abstraction levels, while also reducing the amount of data.

The *eccentricity transform (ECC)* is part of a class of image transforms that associate to each point of the shape a function of the distance to other points of the shape. In the case of the ECC, this function is the maximum over all points of the shape, and the distance is a geodesic distance i.e. the length of the shortest path entirely contained in the shape. The eccentricity transform bridges the concepts of *eccentricity* from graph theory, *furthest point* from computational geometry, and *propagation function* from mathematical morphology. It is robust with respect to noise (Salt and Pepper i.e. random missing points in the shape, and minor segmentation errors), and it is quasi invariant with respect to articulation of the shape.

For planar shapes with less than two holes, eccentric points are always located on the boundary of the shape. For planar simply-connected shapes they lie on convex parts of the boundary. The geodesic center of a planar simply connected shape is a single point. For shapes with holes, or non planar 2D manifolds, it can be a disconnected set of points.

A study of the eccentricity transform of several basic shapes has motivated the presented computation approaches. Efficient approximation algorithms have been derived, and properties showing the possibility to decompose a shape for parallel computing have been formulated. Previous computations of the geodesic distance function over a shape, could provide a stopping criteria for future computations over the same shape.

The histogram of the eccentricity transform is a simple yet powerful descriptor of 2D and 3D shapes and has shown good results in comparative experiments. The application of the

eccentricity transform is also shown in the context of a shape centered coordinate system. This coordinate system, once mapped to different poses of the same articulated 2D shape, allows addressing by means of coordinates, the corresponding points in the different poses.

6.2 Outlook

The following is a list of open questions that have emerged during the research presented in this thesis, and are proposed as future work:

The eccentricity transform:

- For non-planar 2D manifolds and 3D shapes, shadows and separation lines can exist also if the shape is simply connected (has no holes). How do these cases affect the position of eccentric points?
- Certain 2D manifolds (e.g. the surface of an ellipsoid) can be seen as the limit of a 3D shape with a hole of increasing size. Can these two concepts be unified, the 2D manifold and the 3D shape with a hole?
- How can the eccentricity transform of shapes with grayscale information be formulated? What are its properties and how can it be applied?
- What can be obtained from the formulation as a Hausdorff distance? Which properties of Hausdorff distances can be useful for the eccentricity transform?
- Is it possible to further relate the eccentricity transform and topology? Can topological information (e.g. generators) help to efficiently overcome the complications introduced by holes?

Computation of the eccentricity transform:

- The properties required for a cut to act as a stopping point for the computation of the geodesic distance function was given. What is a good strategy to choose candidate cuts?
- The divide et impera approach is closely related to computation using a graph pyramid. If for every point the direction of an eccentric path is known, can the eccentricity transform be efficiently computed using a graph pyramid? Is there a good approximation of the eccentricity transform that can be computed using a graph pyramid, without any additional knowledge?

The eccentricity transform and shape matching:

- Given a grayscale image and the histogram of the eccentricity transform of a shape known to be present in the image, can the segmentation be guided to a better result?
- Instead of building a single descriptor (histogram) for the whole shape, can building descriptors for parts of the shape help increase matching results in the presence of occlusion?

The eccentricity transform and the shape centered coordinate system:

- How can the coordinate system be extended for non-planar 2D shapes, and 3D shapes? (In such cases the geodesic center can be a disconnected set, and isolines can disconnect and connect back again in different configurations.)
- Given a 3D model of an (articulated) object and the color of its surface, can the coordinate system be used to find the pose of the object in the image?

Index

List of Figures

List of Tables

Bibliography

[Aleksandrov 04] Aleksandr Danilovich Aleksandrov. Selected works part ii: Intrinsic geometry of convex surfaces. Chapman & Hall/CRC, 1st edition, May 2004.

[Andres 94] Eric Andres. *Discrete circles, rings and spheres.* Computers & Graphics, vol. 18, no. 5, pages 695–706, 1994.

[Andres 97] Eric Andres & Marie-Andrée Jacob. *The Discrete Analytical Hyperspheres.* IEEE Transactions on Visualization and Computer Graphics, vol. 3, no. 1, pages 75–86, 1997.

[Ankerst 99] Mihael Ankerst, Gabi Kastenmller, Hans-Peter Kriegel & Thomas Seidl. *3D shape histograms for similarity search and classification in spatial databases.* In 6th International Symposium on Advances in Spatial Databases, pages 201–226, London, UK, 1999. Springer.

[Ansary 04] Tarik Filali Ansary, Jean-Philippe Vandeborre, Said Mahmoudi & Mohamed Daoudi. *A Bayesian framework for 3D models retrieval based on characteristic views.* 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004, pages 139–146, 6-9 Sept. 2004.

[Aouada 08] Djamila Aouada, David W. Dreisigmeyer & Hamid Krim. *Geometric modeling of rigid and non-rigid 3D shapes using the global geodesic function.* In NORDIA workshop in conjunction with IEEE International Conference on Computer Vision and Pattern Recognition (CVPR08), Anchorage, Alaska, USA, June 2008. IEEE.

[Asano 87] Tetsuo Asano & Godfried T. Toussaint. *Computing the Geodesic Center of a Simple Polygon.* In D.S. Johnson, A. Nozaki, T. Nishizeki & H. Willis, editors, Perspectives in Computing: Discrete Algorithms and Complexity, Proceedings of Japan-US Joint Seminar, pages 65–79, Boston, 1987. Academic Press.

[Atallah 98] Mikhail J. Atallah, editor. Algorithms and theory of computation handbook. CRC-Press, 1st edition, 1998.

[Aurenhammer 96] Franz Aurenhammer & Rolf Klein. *Voronoi Diagrams.* Technical report, Fern Universität Hagen, Department of Computer Science, Germany, 1996.

[Belongie 02] Serge Belongie, Jitendra Malik & Jan Puzicha. *Shape Matching and Object Recognition Using Shape Contexts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pages 509–522, 2002.

[Berge 91] Claude Berge. Graphs, volume 6 (1) of *North-Holland Mathematical Library*. North-Holland, 3rd revised edition, 1991.

[Borgefors 86] Gunilla Borgefors. *Distance transformations in digital images*. Computer Vision, Graphics, and Image Processing, vol. 34, no. 3, pages 344–371, 1986.

[Borgefors 99] Gunilla Borgefors, Ingela Nyström & Gabriella Sanniti Di Baja. *Computing skeletons in three dimensions*. Pattern Recognition, vol. 37, no. 7, pages 1225–1236, 1999.

[Bouttier 03] J. Bouttier, P. Di Francesco & E. Guitter. *Geodesic Distance in Planar Graphs*. Nuclear Physics B, vol. 663, page 535, 2003.

[Brechbühler 95] Christian Brechbühler, Guido Gerig & Olaf Kübler. *Parametrization of Closed Surfaces for 3-D Shape-Description*. Computer Vision and Image Understanding, vol. 61, no. 2, pages 154–170, 1995.

[Bresenham 77] Jack Bresenham. *A Linear Algorithm for Incremental Digital Display of Circular Arcs*. Commun. ACM, vol. 20, no. 2, pages 100–106, 1977.

[Breu 95] Heinz Breu, Joseph Gil, David Kirkpatrick & Michael Werman. *Linear time Euclidean distance transform algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, pages 529–533, 1995.

[Bronshtein 97] I.N. Bronshtein & K.A. Semendyayev. Handbook of mathematics. Springer, 3rd edition, May 1997.

[Bronstein 06] Alexander M. Bronstein, Michael M. Bronstein, Alfred M. Bruckstein & Ron Kimmel. *Matching two-dimensional articulated shapes using generalized multidimensional scaling*. In Proceedings of the Conference on Articulated Motion and Deformable Objects, pages 48–57, 2006.

[Brun 06] Luc Brun & Walter G. Kropatsch. *Contains and inside relationships within combinatorial pyramids*. Pattern Recognition, vol. 39, no. 4, pages 515–526, 2006.

[Buckley 90] Fred Buckley & Frank Harary. Distances in graphs. Addison-Wesley Publishing Company, 1990.

[Bustos 05] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck & Dejan V. Vranić. *Feature-based similarity search in 3D object databases.* ACM Comput. Surv., vol. 37, no. 4, pages 345–387, 2005.

[Chen 03] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te She & Ming Ouhyoung. *On visual similarity based 3D model retrieval.* In Eurographics, pages 223–232, Granada, Spain, 2003.

[Coeurjolly 02] David Coeurjolly. *Algorithmique et géométrie discrète pour la caractérisation des courbes et des surfaces.* PhD thesis, Université Lumière Lyon 2, Bron, Laboratoire ERIC, December 2002.

[Crum 04] William R. Crum, Thomas Hartkens & Derek L. G. Hill. *Non-rigid image registration: theory and practice.* The British Journal of Radiology, vol. 77 Spec No 2, 2004.

[Cuisenaire 99] Olivier Cuisenaire & Benoit M. Macq. *Fast Euclidean distance transformation by propagation using multiple neighborhoods.* Computer Vision and Image Understanding, vol. 76, no. 2, pages 163–172, 1999.

[Cyr 04] Christopher M. Cyr & Benjamin B. Kimia. *A Similarity-Based Aspect-Graph Approach to 3D Object Recognition.* International Journal of Computer Vision, vol. 57, no. 1, pages 5–22, 2004.

[Damiand 05] Guillaume Damiand, Martine Dexet-Guiard, Pascal Lienhardt & Eric Andres. *Removal and contraction operations to define combinatorial pyramids: application to the design of a spatial modeler.* Image and Vision Computing, vol. 23, no. 2, pages 259–269, 2005.

[Danielsson 80] Per-Erik Danielsson. *Euclidean distance mapping.* Computer Vision, Graphics, and Image Processing, vol. 14, pages 227–248, 1980.

[Day 94] Jane M. Day. Plato's meno in focus. Routledge, 1st edition, January 1994.

[Diestel 97] Reinhard Diestel. Graph theory. Number 173 in Graduate Texts in Mathematics. Springer, 1997.

[Elad 01] Michael Elad, Ayellet Tal & Sigal Ar. *Content based retrieval of VRML objects: an iterative and interactive approach.* In Eurographics Workshop on Multimedia, pages 107–118, New York, 2001. Springer.

[Fabbri 08] Ricardo Fabbri, Luciano Da F. Costa, Julio C. Torelli & Odemir M. Bruno. *2D Euclidean distance transform algorithms: A comparative survey.* ACM Computing Surveys, vol. 40, no. 1, pages 1–44, 2008.

[Fawcett 06] Tom Fawcett. *An introduction to ROC analysis.* Pattern Recognition Letters, vol. 27, no. 8, pages 861–874, 2006.

[Felzenszwalb 03] Pedro F. Felzenszwalb. *Representation and Detection of Deformable Shapes.* In IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2003.

[Felzenszwalb 07] Pedro F. Felzenszwalb & Joshua D. Schwartz. *Hierarchical Matching of Deformable Shapes.* In CVPR, 2007.

[Flanitzer 06] Thomas Flanitzer. *The eccentricity transform (computation).* Technical Report PRIP-TR-107, PRIP, TU Wien, 2006.

[Fouard 05] Celine Fouard & Gregoire Malandain. *3-D chamfer distances and norms in anisotropic grids.* Image and Vision Computing, vol. 23, no. 2, page 143158, February 2005.

[Gdalyahu 99] Yoram Gdalyahu & Daphna Weinshall. *Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchical Classification of Silhouettes.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 12, pages 1312–1328, 1999.

[Goodman 04] Jacob E. Goodman & Joseph O'Rourke, editors. Handbook of discrete and computational geometry. Chapman & Hall/CRC, 2nd edition, 2004.

[Gorelick 04] Lena Gorelick, Meirav Galun, Eitan Sharon, Ronen Basri & Achi Brandt. *Shape Representation and Classification Using the Poisson Equation.* In IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pages 61–67, 2004.

[Guan 98] Weiguang Guan & Songde Ma. *A List-Processing Approach to Compute Voronoi Diagrams and the Euclidean Distance Transform.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 7, pages 757–761, 1998.

[Hamza 03] A. Ben Hamza & Hamid Krim. *Geodesic Object Representation and Recognition.* In DGCI: International Conference on Discrete Geometry for Computer Imagery, 2003.

[Hausdorff 05] Felix Hausdorff & John R. Aumann. Set theory. American Mathematical Society (Chelsea), 2005. (Translation from German).

[Haxhimusa 02] Yll Haxhimusa, Roland Glantz, Maamar Saib, Georg Langs & Walter G. Kropatsch. *Logarithmic Tapering Graph Pyramid*. In Luc van Gool, editor, Proceedings of 24th DAGM (German Association for Pattern Recognition) Symposium, pages 117–124, Swiss, 2002. Springer Verlag LNCS 2449.

[Haxhimusa 03] Yll Haxhimusa & Walter G. Kropatsch. *Hierarchy of Partitions with Dual Graph Contraction*. In Bernd Michaelis & Gerald Krell, editors, Pattern Recognition, 25th DAGM (German Association for Pattern Recognition) Symposium, volume 2781 of *Lecture Notes in Computer Science*, pages 338–345, Magdeburg, Germany, September 2003. Springer.

[Hazewinkel 89] Michiel Hazewinkel, editor. Encyclopaedia of mathematics (4), volume 4. Springer, 1st edition, August 1989.

[Hilaga 01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura & Tosiyasu L. Kunii. *Topology matching for fully automatic similarity estimation of 3D shapes*. In SIGGRAPH '01: Proc. of the 28th conf. on Computer graphics and interactive techniques, pages 203–212, New York, USA, 2001. ACM.

[Hirata 96] Tomio Hirata. *A unified linear-time algorithm for computing distance maps*. Information Processing Letters, vol. 58, no. 3, pages 129–133, 1996.

[Huang 94] C. Tony Huang & Owen Robert Mitchell. *A Euclidean Distance Transform Using Grayscale Morphology Decomposition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 4, pages 443–448, 1994.

[Ion 07a] Adrian Ion, Samuel Peltier, Yll Haxhimusa & Walter G. Kropatsch. *Decomposition for Efficient Eccentricity Transform of Convex Shapes*. In Walter G. Kropatsch, Martin Kampel & Allan Hanbury, editors, The 12th International Conference on Computer Analysis of Images and Patterns (CAIP), volume 4673 of *Lecture Notes in Computer Science*, pages 653–660, Vienna, Austria, August 2007. Springer.

[Ion 07b] Adrian Ion, Gabriel Peyré, Yll Haxhimusa, Samuel Peltier, Walter G. Kropatsch & Laurent Cohen. *Shape Matching Using the Geodesic Eccentricity Transform - A Study*. In C. Beleznai W. Ponweiser M. Vincze, editor,

The 31st annual workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR), pages 97–104, Schloss Krumbach, Austria, May 2007. OCG.

[Ion 08a] Adrian Ion, Nicole M. Artner, Gabriel Peyré, Salvador B. López Mármol, Walter G. Kropatsch & Laurent Cohen. *3D Shape Matching by Geodesic Eccentricity*. In Workshop on Search in 3D (in conjunction with CVPR 2008), Anchorage, Alaska, June 2008. IEEE.

[Ion 08b] Adrian Ion, Yll Haxhimusa, Walter G. Kropatsch & Salvador B. López Mármol. *A Coordinate System for Articulated 2D Shape Point Correspondences*. In Proceedings of 19th International Conference on Pattern Recognition (ICPR). IAPR, IEEE, 2008.

[Ion 08c] Adrian Ion & Walter G. Kropatsch. *Mapping a Coordinate System to a Non-rigid Shape*. In Arjan Kuijper, Bettina Heise & Leila Muresan, editors, The 32nd Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR), volume 232 of *books@ocg.at*, pages 169–178, Linz, Austria, May 2008. OCG.

[Ion 08d] Adrian Ion, Walter G. Kropatsch & Eric Andres. *Euclidean Eccentricity Transform by Discrete Arc Paving*. In David Coeurjolly, Isabelle Sivignon, Laure Tougne & Florent Dupont, editors, 14th IAPR International Conference on Discrete Geometry for Computer Imagery (DGCI), volume LNCS 4992 of *Lecture Notes in Computer Science*, pages 213–224, Lyon, France, April 2008. Springer.

[Ion 08e] Adrian Ion, Samuel Peltier, Sylvie Alayrangues & Walter G. Kropatsch. *Eccentricity based Topological Feature Extraction*. In Sylvie Alayrangues, Guillaume Damiand, Laurent Fuchts & Pascal Lienhardt, editors, Workshop on Computational Topology in Image Context, Poitiers, France, June 2008.

[Ip 03] Cheuk Yiu Ip, William C. Regli, Leonard Sieger & Ali Shokoufandeh. *Automated learning of model classifications*. In 8th ACM Symposium on Solid Modeling and Applications, pages 322–327, New York, 2003. ACM Press.

[Johnson 99] Andrew E. Johnson & Martial Hebert. *Using spin images for efficient object recognition in cluttered 3D scenes*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 21, no. 5, pages 433–449, May 1999.

[Jolion 94] Jean-Michel Jolion & Azriel Rosenfeld. A pyramid framework for early vision: Multiresolution computer vision. Kluwer, 1994.

[Kambhamettu 94] Chandra Kambhamettu & Dmitry B. Goldgof. *Curvature-Based Approach to Point Correspondence Recovery in Conformal Nonrigid Motion.* Computer Vision, Graphics and Image Processing: Image Understanding, vol. 60, no. 1, pages 26–43, 1994.

[Kazhdan 03] Michael Kazhdan, Thomas Funkhouser & Szymon Rusinkiewicz. *Rotation invariant spherical harmonic representation of 3D shape descriptors.* In SGP '03: Proc. of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 156–164. Eurographics Association, 2003.

[Klette 04] Reinhard Klette & Azriel Rosenfeld. Digital geometry. Morgan Kaufmann, 2004.

[Kropatsch 95] Walter G. Kropatsch. *Building Irregular Pyramids by Dual Graph Contraction.* IEE Proceedings - Vision, Image and Signal Processing, vol. 142, no. 6, pages 366–374, December 1995.

[Kropatsch 05] Walter G. Kropatsch, Yll Haxhimusa, Zygmunt Pizlo & Georg Langs. *Vision pyramids that do not grow too high.* Pattern Recognition Letters, vol. 26, no. 3, pages 319–337, 2005.

[Kropatsch 06] Walter G. Kropatsch, Adrian Ion, Yll Haxhimusa & Thomas Flanitzer. *The Eccentricity Transform (of a Digital Shape).* In 13th International Conference on Discrete Geometry for Computer Imagery (DGCI), pages 437–448, Szeged, Hungary, 25–27, October 2006. Springer.

[Kropatsch 07] Walter G. Kropatsch, Adrian Ion & Samuel Peltier. *Computing the Eccentricity Transform of a Polygonal Shape.* In Luis Rueda, Domingo Mery & Josef Kittler, editors, 12th Iberoamerican Congress on Pattern Recognition (CIARP 2007), volume LNCS 4756 of *Lecture Notes in Computer Science*, pages 291–300, Viña del Mar / Valparaiso, Chile, November 2007. Springer.

[Latecki 97] Longin Jan Latecki. *3D Well-Composed Pictures.* CVGIP: Graphical Model and Image Processing, vol. 59, no. 3, pages 164–172, 1997.

[Latecki 00] Longin Jan Latecki, Rolf Lakämper & Ulrich Eckhardt. *Shape Descriptors for Non-Rigid Shapes with a Single Closed Contour.* In IEEE International

Conference on Computer Vision and Pattern Recognition (CVPR), pages 1424–1429. IEEE Computer Society, 2000.

[Lienhardt 94] Pascal Lienhardt. *N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds*. International Journal of Computational Geometry and Applications, vol. 4, no. 3, pages 275–324, 1994.

[Ling 07] Haibin Ling & David W. Jacobs. *Shape Classification Using the Inner-Distance*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 2, pages 286–299, 2007.

[Maisonneuve 89] Francis Maisonneuve & Michell Schmitt. *An efficient algorithm to compute the hexagonal and dodecagonal propagation function.* Acta Stereologica, vol. 8, no. 2, pages 515–520, September 1989.

[Marsden 86] Jerrold E. Marsden & Alan Weinstein. Calculus. Springer, 1986.

[Martin 01] David R. Martin, Charless Fowlkes, Doron Tal & Jitendra Malik. *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics*. In Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), volume 2, pages 416–425, July 2001.

[Maurer Jr. 03] Calvin R. Maurer Jr., Rensheng Qi & Vijay V. Raghavan. *A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 2, pages 265–270, 2003.

[Meer 89] Peter Meer. *Stochastic Image Pyramids*. Computer Vision, Graphics, and Image Processing, vol. 45, no. 3, pages 269–294, March 1989. Also as UM CS TR-1871, June, 1987.

[Meijster 00] A. Meijster, J.B.T.M. Roerdink & W. H. Hesselink. Mathematical morphology and its applications to image and signal processing, chapter A general algorithm for computing distance transforms in linear time, pages 331–340. Kluwer, 2000.

[Mer 03] Merriam-webster's collegiate dictionary. Merriam-Webster, 11th edition, 2003.

[Mokhtarian 92] Farzin Mokhtarian & Alan K. Mackworth. *A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 8, pages 789–805, August 1992.

[Mora 03] F. Mora, G. Ruillet, Eric Andres & R. Vauzelle. *Pedagogic discrete visualization of electromagnetic waves.* In Eurographics, Granada, Spain, January 2003.

[Mullikin 92] James C. Mullikin. *The vector distance transform in two and three dimensions.* CVGIP: Graphical Models and Image Processing, vol. 54, no. 6, pages 526–535, 1992.

[Munkres 93] James R. Munkres. Elements of algebraic topology. Addison-Wesley, 1993.

[Narici 97] Lawrence Narici & Edward Beckenstein. *The Hahn-Banach theorem: the life and times.* Topology and its Applications, vol. 77, no. 2, pages 193–211, June 1997.

[Newman 00] James R. Newman, editor. The world of mathematics, volume 3. Dover Publications; Unabridged edition, 2000.

[Ogniewicz 95] Robert L. Ogniewicz & Olaf Kübler. *Hierarchic Voronoi Skeletons.* Pattern Recognition, vol. 28, no. 3, pages 343–359, March 1995.

[Osada 02] Robert Osada, Thomas Funkhouser, Bernard Chazelle & David Dobkin. *Shape distributions.* ACM Transactions on Graphics, vol. 21, no. 4, pages 807–832, 2002.

[Ozcanli 07] Ozge C. Ozcanli & Benjamin B. Kimia. *Generic Object Recognition via Shock Patch Fragments.* In British Machine Vision Conference, pages 1030–1039. Warwick Print, 2007.

[Papadopoulos 03] Athanase Papadopoulos. Metric spaces, convexity and nonpositive curvature. European Mathematical Society, December 2003.

[Paquet 99] Eric Paquet & Marc Rioux. *Nefertiti: A tool for 3-D shape databases management.* SAE transactions, vol. 108, pages 387–393, 1999.

[Paquet 00] Eric Paquet, Anil Murching, Thumpudi Naveen, Ali Tabatabai & Marc Rioux. *Description of shape information for 2-D and 3-D objects.* Signal Pro-

cessing: Image Communication, vol. 16, no. 1–2, pages 103–122, September 2000.

[Paragios 06] Nikos Paragios, Yunmei Chen & Olivier Faugeras. Handbook of mathematical models in computer vision., chapter 6, pages 97–111. Springer, 2006.

[Pearson 1900] Karl Pearson. *Mathematical Contributions to the Theory of Evolution. VII. On the Correlation of Characters not Quantitatively Measurable.* Philosophical Transactions of the Royal Society of London, vol. 195, pages 1–47, 1900.

[Piper 87] Jim Piper & Erik Granum. *Computing distance transformations in convex and non-convex domains.* Pattern Recognition, vol. 20, no. 6, pages 599–615, 1987.

[Pizlo 08] Zygmunt Pizlo. 3d shape: Its unique place in visual perception. The MIT Press, first edition, April 2008.

[Pollack 89] Richard Pollack, Micha Sharir & Günter Rote. *Computing the geodesic center of a simple polygon.* Discrete & Computational Geometry, vol. 4, no. 6, pages 611–626, 1989.

[Qian 97] Kai Qian, Siqi Cao & Prabir Bhattacharya. *Skeletonization of gray-scale images by gray weighted distance transform.* In S. K. Park & R. D. Juday, editors, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, volume 3074 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 224–228, July 1997.

[Reeb 64] Georges Reeb. *Sur les points singuliers d'une forme de Pfaff complément intégrable ou d'une fonction numérique.* Annales de l'institut Fourier, 14 no. 1, vol. 14, no. 1, pages 37–42, 1964.

[Reuter 05] Martin Reuter, Franz-Erich Wolter & Niklas Peinecke. *Laplace-spectra as fingerprints for shape matching.* In L. Kobbelt & V. Shapiro, editors, Symp. on Solid and Physical Modeling, pages 101–106. ACM, 2005.

[Reveillès 91] Jean-Pierre Reveillès. Géométrie discrète, calcul en nombres entiers et algorithmique (in french). University Louis Pasteur of Strasbourg, 1991.

[Rosenfeld 66] Azriel Rosenfeld & John L. Pfaltz. *Sequential Operations in Digital Picture Processing.* Journal of the ACM, vol. 13, no. 4, pages 471–494, 1966.

[Rosenfeld 68] Azriel Rosenfeld & John L. Pfaltz. *Distance functions on digital pictures.* Pattern Recognition, vol. 1, no. 1, pages 33–61, 1968.

[Rosenfeld 83] A. Rosenfeld. *A Note on 'Geometric Transforms' of Digital Sets.* Pattern Recognition Letters, vol. 1, no. 4, pages 223–225, 1983.

[Rouy 92] Elisabeth Rouy & Agnès Tourin. *A viscosity solutions approach to shape-from-shading.* SIAM Journal on Numerical Analysis, vol. 29, no. 3, pages 867–884, 1992.

[Rutovitz 68] D. Rutovitz. *Data structures for operations on digital images.* In G.C. Cheng, R.S. Ledley, D.K. Pollok & A. Rosenfeld, editors, Pictorial Pattern Recognition, pages 105–133. Thompson Book Company, 1968.

[Rutovitz 78] D. Rutovitz. *Expanding picture components to natural density boundaries by propagation methods, the notions of fall-set and fall-distance.* In 4th Int. Joint Conf. Pattern Recognition, pages 657–664, Kyoto, Japan, 1978.

[Saito 94] Toyofumi Saito & Jun-Ichiro Toriwaki. *New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications.* Pattern Recognition, vol. 27, no. 11, pages 1551–1565, November 1994.

[Schmitt 93] Michell Schmitt. *Propagation Function: Towards Constant Time Algorithms.* Acta Stereologica: Proceedings of the 6th European Congress for Stereology, September 7-10, Prague, vol. 13, no. 2, December 1993.

[Sebastian 04] Thomas B. Sebastian, Philip N. Klein & Benjamin B. Kimia. *Recognition of Shapes by Editing Their Shock Graphs.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 5, pages 550–571, 2004.

[Sethian 99] James A. Sethian. Level sets methods and fast marching methods. Cambridge Univ. Press, 2nd edition, 1999.

[Sharvit 98] Daniel Sharvit, Jacky Chan, Hüseyin Tek & Benjamin B. Kimia. *Symmetry-based Indexing of Image Databases.* In IEEE Workshop on Content-based Access of Image and Video Libraries, pages 56–62, 1998.

[Shi 00] Jianbo Shi & Jitendra Malik. *Normalized Cuts and Image Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pages 888–905, 2000.

[Shih 92] Frank Y. Shih & Yi-Ta Wu. *Optimization on Euclidean Distance Transformation Using Grayscale Morphology.* Journal of Visual Communication and Image Representation, vol. 3, pages 104–114, 1992.

[Shilane 04] Philip Shilane, Patrick Min, Michael M. Kazhdan & Thomas A. Funkhouser. *The Princeton Shape Benchmark.* In Int. Conf. on Shape Modeling and Applications (SMI), Genova, Italy, pages 167–178. IEEE Computer Society, 2004.

[Shinagawa 91] Y. Shinagawa, T.L. Kunii & Y.L. Kergosien. *Surface coding based on Morse theory.* Computer Graphics and Applications, IEEE, vol. 11, no. 5, pages 66–78, September 1991.

[Siddiqi 99] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson & Steven W. Zucker. *Shock Graphs and Shape Matching.* International Journal of Computer Vision, vol. 30, pages 1–24, 1999.

[Siddiqi 07] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, R. Chen & Sven J. Dickinson. *Retrieving Articulated 3-D Models Using Medial Surfaces.* Machine Vision and Applications, 2007.

[Soille 94] Pierre Soille. *Generalized geodesic distances applied to interpolation and shape description.* In Jean Serra & Pierre Soille, editors, Mathematical Morphology and Its Applications to Image Processing, Computational Imaging and Vision, pages 193–200. Kluwer/Springer, 1994.

[Soille 02] Piere Soille. Morphological image analysis. Springer, 2nd edition, 2002.

[Sundar 03] H. Sundar, Deborah Silver, Nikhil Gagvani & Sven J. Dickinson. *Skeleton based shape matching and retrieval.* Shape Modeling International, 2003, pages 130–139, 12-15 May 2003.

[Suri 87] Subhash Suri. *The All-Geodesic-Furthest Neighbor Problem for Simple Polygons.* In Symposium on Computational Geometry, pages 64–75, 1987.

[Tangelder 03] Johan W. H. Tangelder & Remco C. Veltkamp. *Polyhedral model retrieval using weighted point sets.* In Shape Modeling International, pages 119–129, Seoul, Korea, 2003. IEEE.

[Thulasiraman 92] K. Thulasiraman & M. N. S. Swamy. Graphs: Theory and algorithms. Wiley-Interscience, 1992.

[Toivanen 96] Pekka J. Toivanen. *New geodesic distance transforms for gray-scale images.* Pattern Recognition Letters, vol. 17, no. 5, pages 437–450, 1996.

[Tu 04] Zhuowen Tu & Alan L. Yuille. *Shape Matching and Recognition - Using Generative Models and Informative Features.* In Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III, volume 3023 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2004.

[Veltkamp 06] Remco C. Veltkamp & Longin Jan Latecki. *Properties and Performance of Shape Similarity Measures.* In Proc. of IFCS 2006, 2006.

[Verwer 91] Ben J. H. Verwer. *Local distances for distance transformations in two and three dimensions.* Pattern Recognition Letters, vol. 12, no. 11, pages 671–682, 1991.

[Voronoï 1907] Georgy Feodosevich Voronoï. *Nouvelles applications des paramètres continusàla théorie des formes quadratiques.* Journal für die Reine und Angewandte Mathematik, vol. 133, pages 97–178, 1907.

[Vranic 01a] Dejan V. Vranic & Dietmar Saupe. *3D model retrieval with spherical harmonics and moments.* In 23rd DAGM-Symposium (German Association for Pattern Recognition), pages 392–397, London, UK, 2001. Springer.

[Vranic 01b] Dejan V. Vranic & Dietmar Saupe. *3D shape descriptor based on 3D fourier transform.* In EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services, pages 271–274. Comenius University, 2001.

[Vranic 02] Dejan V. Vranic & Dietmar Saupe. *Description of 3D-shape using a complex function on the sphere.* In International conference on Multimedia and Expo, pages 177–180. IEEE, 2002.

[Weisstein 02] Eric W. Weisstein. Crc concise encyclopedia of mathematics. Chapman & Hall/CRC, 2nd edition, December 2002.

[Yang 06a] Liu Yang & Rong Jin. *Distance Metric Learning: A Comprehensive Survey.* Technical report, Department of Computer Science and Engineering, Michigan State University, 2006. http://www.cse.msu.edu/˜yangliu1/frame_survey_v2.pdf.

[Yang 06b] Liu Yang, Rong Jin, Rahul Sukthankar & Yi Liu. *An Efficient Algorithm for Local Distance Metric Learning*. In Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI'06). AAAI Press, 2006.

[Zaharia 01] Titus Zaharia & Francoise Prêteux. *3D shape-based retrieval within the MPEG-7 framework*. In SPIE Conference on Nonlinear Image Processing and Pattern Analysis XII, pages 133–145, 2001.

[Zahn 72] C. T. Zahn & R. Z. Roskies. *Fourier Descriptors for Plane Closed Curves*. IEEE Trans. Computer, vol. 21, no. 3, pages 269–281, March 1972.

[Zhang 05] Juan Zhang, Kaleem Siddiqi, Diego Macrini, Ali Shokoufandeh & Sven J. Dickinson. *Retrieving Articulated 3-D Models Using Medial Surfaces and Their Graph Spectra*. In 5th Int. Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition, EMMCVPR 2005, volume 3757 of *LNCS*, pages 285–300. Springer, 2005.

[Zunic 06] Jovisa D. Zunic, Paul L. Rosin & Lazar Kopanja. *On the Orientability of Shapes*. IEEE Transactions on Image Processing, vol. 15, no. 11, pages 3478–3487, 2006.

Curriculum Vitae

Personal Data

Name	Adrian Ion
Date of Birth	24.05.1978
Place of Birth	Timisoara – Romania
Citizenship	Romanian

Education

2004 – 2009	Doctoral study at the Pattern Recognition and Image Processing Group, Faculty of Informatics, Vienna University of Technology.
1996 – 2001	Dipl.-Ing. in Computer Science at the "Politehnica" University Timisoara, Computer Science Faculty.
1992 – 1996	Informatics High School "Grigore Moisil" Timisoara.

Career History

2004 –	Research assistant at the Pattern Recognition and Image Processing Group, TU Vienna, working at the Austrian Cognitive Vision project. Prof. Kropatsch.
2003 – 2004	C++ software developer at Innovative Systems SRL, Timisoara - image processing, Web application development.
1999 – 2003	C++ software developer at CD Nelson Technology Group SRL, Timisoara – CAD/CAM/CNC software development.
1998 – 1999	C++ software developer at Sedona Software SRL, Timisoara.

Awards

2006	Best Student Paper Award of the Austrian Association for Pattern Recognition (ÖAGM) with paper: Thomas Illetschko, Adrian Ion, Yll Haxhimusa & Walter G. Kropatsch. *Collapsing 3D Combinatorial Maps.* Proceedings of the 30th annual workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR), pages 85–93, Obergurgl, Austria, 2006. OCG.

2000, 1999, 1997	3^{rd}, 5^{th}, and 4^{th} place at the "ACM International Collegiate Programming Contest", Southeastern European Stage.
1999, 1998, 1997	2^{nd}, 3^{rd}, and 2^{nd} prize at The Annual Programming Contest of the Computer Science Faculty, "Politehnica" University Timisoara.
1998	1^{st} prize at Annual Communications Session on robotics organized by the Mechanics Faculty, "Politehnica" University Timisoara.
1996	2^{nd} prize at The National Informatics Communications Session, Romania.
1996, 1995	3^{rd} and 2^{nd} prize at The National Informatics Olympics, Romania.
1995	3^{rd} prize at the International Informatics Contest held in Iakutsk (Russian Federation).
1995	Member of the Romanian Extended National Informatics Team (14 high school pupils from the whole country).

Memberships AAPR, IAPR TC-15

Publications

33 refereed publications in scientific journals (3) and books (2), conferences (11), and workshops (17).

Research Interests

Graph based representations and methods, shape representation and recognition, visual abstraction, abstraction in the human language, human problem solving.

Nothing shocks me. I'm a scientist.
$\sim$ Harrison Ford, as Indiana Jones