

FREIE UNIVERSITÄT BERLIN
INSTITUT FÜR INFORMATIK
TAKUSTR. 9
14195 BERLIN

DIPLOMARBEIT

Multiscale Curvature Matching for Smooth Polylines

Author:
Anne DRIEMEL

Supervisor:
Prof. Helmut ALT

ABSTRACT: The objective of matching polygonal curves is to determine their perceptual similarity, the notion of which is often based upon the similarity transformations. But searching the transformation space is costly and error-prone, depending on the set of transformations that are allowed. Curvature, as defined for smooth curves, is to a large extent invariant under isometries and uniform scaling. At the same time it determines the curve up to rigid motions. The idea to use curvature for matching is therefore imminent. The work in this text includes the analysis of different self-contained definitions for the curvature of polygonal curves, which use the concept of scale space, a metrical distance measure for those curvature functions and an efficient algorithm to compute the respective similarity distance of two polygonal curves under the above mentioned transformations.

31. Januar, 2009

CONTENTS

1. Introduction	1
2. Curvature of Smooth Curves	3
2.1. General Formula	4
2.2. Osculating Circle	5
2.3. Invariance	6
2.4. Uniqueness	8
3. Curvature of Polygonal Curves	9
3.1. Polygonal Curve Model	9
3.2. Curvature Model	11
3.3. Radius of Curvature	13
3.4. Turning Angle Curvature	13
3.5. Norm of the Second Derivative	14
3.6. Direct Curvature	15
3.7. Differentiation using Convolution	16
3.8. Equivalences in the Ideal Case	17
3.9. Examples	20
4. Definition of Similarity/Distance	25
4.1. The Distance Measure	25
4.2. Metric Property	26
4.3. Reflection and Change of Orientation	28
4.4. Normalization	29
5. Analysis of the Objective Function	31
5.1. Continuity	31
5.2. Decomposition	32
5.3. Case Studies in 1D	36
5.4. Singularities	43
5.5. Illustrative Example in 2D	49
6. Computation	52
6.1. Algorithm	53
6.2. Experimental Results	58
7. Conclusions	63
Index	64
References	65
Appendix A. Implementation	67
A.1. CurvatureViewer Project	67
A.2. gdmatch Project	77
A.3. Source Code	78
Appendix B. Detailed Results of the Experiments	89
B.1. Input Data	89
B.2. High Scale Matching	91
B.3. Low Scale Matching	118

1. INTRODUCTION

Researchers agree that curvature is perceptually relevant, [FF94], [MSK95], [AB86], [WS93], and would therefore be useful for the computerized interpretation of image data, both from photography and human-made. Much effort has been put into finding methods for the estimation of the curvature of the pre-digitized curve. A related, maybe the relaxed, problem is that of corner detection. Rosenfeld and Johnston used the cosine of the angle between consecutive edges as a measure for corner strength as early as 1973 [RJ73]. In 1993 Worring and Smeulders did a survey on the existing curvature estimation methods [WS93]. They categorized the methods into three classes, orientation based, path based (curve fitting), and methods based on the osculating circle. They found that almost all suffer from poor precision as a result of the rasterization process, which the curves undergo as the real world images that contain them are being digitized. Some of the methods are also fairly involved in their computation [TC94]. Researchers concluded that exact curvature computation is difficult because of its sensitivity to noise [FF94].

A common practice to eliminate noise in data is the use of Gaussian smoothing. In order to do this, however, one has to decide for an appropriate kernel size which controls the strength of the smoothing. To choose a kernel size automatically is a non-trivial, maybe ill-defined, problem which has led to the evolution of scale space theory. According to the respective entry in the *Encyclopedia of Computer Science and Engineering* [Lin08], from 2008, real world objects have a multi-scale nature and therefore have to be dealt with at multiple scales simultaneously. "A simple example is the concept of a branch of a tree, which makes sense only at a scale from, say, a few centimeters to at most a few meters. It is meaningless to discuss the tree concept at the nanometre or kilometre level". Through the observation perspective the objects are additionally scaled. The article concludes that the notion of scale is fundamental to the understanding of both natural and artificial perception. The Gaussian kernel and its derivatives are the designated central operators of scale space theory, which is motivated by the fact that they resemble closely receptive field profiles registered in neuro-physiological studies [Lin08], [Hil80].

Different concepts exist that use both, the concept of scale space and curvature for the structuring of curves for recognition. Asada and Brady introduced the curvature primal sketch in 1984 [AB86]. Mokhtarian and Mackworth are the authors of the curvature scale space which they introduced in the early 90's [MB03] and which is now part of the MPEG-7 standard for multimedia content description. The noise, however that is due to rasterization is not Gaussian, which is why Matas et al. introduced a technique called median filtered differencing [MSK95], which chooses the median of different angles instead of the weighted sum. This was also already indicated by Worring and Smeulders in their conclusion [WS93]. They write that the anisotropic nature of the grid causes difficulty in the correct parametrization, and introduces discontinuities in the second derivatives and therefore high frequencies in the Fourier transform, which, they say, makes smoothing unreliable. In 2001 another survey was done by Coeurjolly et. al. [CMT01] with the conclusion that discrete arc fitting is possible in optimal time without smoothing. Although Utcke states in 2003 in [Utc03] that accurate curvature estimation under noise is still difficult.

Naturally the common definition of the discretized curve is based on pixels in most of these papers. Another way to approach discrete curves is to define them as polygonal curves, given as a list of vertices. The problem of matching polygonal curves, is widely studied in the field of computational geometry [VH99], [VL06], [AG96], but remains a hard problem. It can be stated as an optimization problem.

Given two polygonal curves, or shapes, find a transformation that minimizes the dissimilarity between the transformed curve and the other curve, where the measure of dissimilarity is yet to be defined. The time complexity of this problem is often dominated by the complexity of the class of transformations that are allowed. We consider the class of isometries together with uniform scaling. In the context of shape matching the curvature is not only valuable because of its perceptual relevance, but also because of its geometric properties. The curvature, as defined for smooth curves, is invariant under rigid motions and varies only by a constant factor under uniform scaling. At the same time it determines the curve uniquely up to those transformations. The idea to match only the curvature function is straightforward, since the only information that is lost about the curve, is the curve's position in space and its direction, which are properties, that can be said to be irrelevant for resemblance.

Another approach for the definition of the curvature of a polygonal curve makes use of the technique of subdivision [DL03], which is similar to convolution, except that in each iteration new vertices are being introduced. Subdivision schemes are used in modern applications of computer graphics. Some subdivision masks (kernels) are known to yield spline curves as limit under the repeated application of the mask. The limit points and therefore the derivatives can be directly computed using the eigenvectors of the subdivision matrix [Uml01]. In the simple case of Bézier curves, one can also use the Bernstein polynomials [GJS07].

For the above stated shape matching problem, we do not need to search for an exact estimation of the curvature of the pre-digitized curve. A measure that has the same property of being invariant under the considered similarity transformations with respect to the polygonal curve and a meaning, such that the dissimilarity measure is justified, would suffice. An example of a distance measure for polygons, proposed by Arkin et. al. in 1990, in [ACH⁺90] uses the so-called turning function. The turning function $\theta(s)$ measures the angle of the counterclockwise tangent as a function of the arc length with respect to some reference orientation. The function $\theta(s)$ is constant on the interval where s traverses a single edge, and therefore piecewise constant for the polygon. Adding a constant ϕ to the function amounts to rotating the curve by the angle ϕ . The distance between polygons A and B is defined as

$$\delta_p(A, B) = \|\theta_A - \theta_B\|_p = \left(\int_0^1 |\theta_A(s) - \theta_B(s)|^p ds \right)^{\frac{1}{p}}.$$

Both curves are scaled such that they have unit length. Now in order to minimize this distance under rotation and the choice of a starting point it is to solve

$$d_p(A, B) = \left(\min_{\phi \in \mathbb{R}, t \in [0,1]} \int_0^1 |\theta_A(s+t) - \theta_B(s) + \phi|^p ds \right)^{\frac{1}{p}}.$$

Let $p = 2$, using the fact that there are global minima where a vertex of A is aligned with a vertex of B and that the objective function is a convex function of ϕ , Arkin et. al. propose a basic algorithm that has complexity $O(mn(n+m))$, which can be improved to $O(mn \log mn)$ using the Fast Fourier Transform. The speed-up however requires that the edges of the polygons have pairwise equal length and their number is the same in A and B . The metric is scale-invariant because it normalizes the scale of the curves. If one wants to apply this to open polylines and take different scales into account, a third parameter is introduced, that has to be optimized [CG97]. The turning function and its associated distance measure is interesting for the work that we are presenting since there are a lot of parallels.

2. CURVATURE OF SMOOTH CURVES

We will first deliver the standard definitions of the basic terms curve, arc length, curvature, and others, that are used throughout the text. We will only be dealing with plane curves, which are the most simple objects in differential geometry. The notation and definitions are mostly taken from [dC76] some of the concepts and ideas are taken from [Spi79].

A *parametrized plane curve* is a map $\alpha : I \rightarrow \mathbb{R}^2$ of a closed interval $I = [a, b] \subset \mathbb{R}$ into the plane. We will refer to parametrized plane curves simply as curves. A curve is usually given as a tuple of component functions $\alpha(t) = (x(t), y(t))$ and is differentiable, or *smooth* if each of the component functions x and y are differentiable.

The derivative vector at a point $\alpha'(t) = (x'(t), y'(t))$ is referred to as the *tangent* at the point. This can be justified by the fact that $x'(t)$ and $y'(t)$ are the slopes of the component functions and therefore $\{\lambda\alpha'(t) + \alpha(t) \mid \lambda \in \mathbb{R}\}$ constitutes the tangent line at $\alpha(t)$. Of course, this is not well-defined if $\alpha'(t) = 0$. Curves that satisfy $\alpha'(t) \neq 0$ everywhere are called *regular*. The discussion in this chapter is restricted to regular curves.

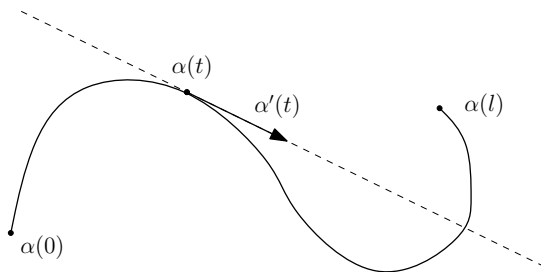


FIGURE 1. The first derivative $\alpha'(t)$ and the tangential line at $\alpha(t)$.

We call the image $\alpha(I)$ the *trace* of α . A strictly increasing function $\phi : D \rightarrow I$, where $D = (a, b), I = (0, l) \subset \mathbb{R}$, is called the *parametrization* of $\alpha \circ \phi$. It is clear that $\alpha \circ \phi : I \rightarrow \mathbb{R}^2$ is also a curve which has the same trace as α , but not necessarily the same domain. The functions ϕ which are strictly decreasing also yield a curve of the same trace. We say that those curves differ by a *change of orientation*. We define the *arc length* function of a curve

$$(1) \quad s(t) = \int_{t_0}^t \|\alpha'(t)\| dt, \quad \text{where } \|\alpha'(t)\| = \sqrt{x'(t)^2 + y'(t)^2}.$$

Sometimes the parameter of the curve is equal to the arc length measured from the beginning of the curve $t_0 = 0$. In this case we speak of an *arc length parametrization* of the curve. It happens when $\|\alpha'(t)\| = 1$ everywhere. Then and only then $s(t) = \int_0^t 1 dt = t$. If the tangent vector $\alpha'(t)$ has unit length, then the norm of the second derivative measures the rate of change of the angle which neighbouring tangents make with the tangent at t , see also Figure 5. $\|\alpha''(t)\|$ gives, therefore, a measure of how rapidly the curve pulls away from the tangent line at t in a neighborhood of t . A common definition of the *unsigned curvature* is therefore

$$(2) \quad \kappa(t) = \|\alpha''(t)\|, \quad t = \text{arc length parameter.}$$

In our special case of plane curves it is possible to give the curvature a sign, such that $\kappa(t) > 0$ if the curve makes a left turn at t and $\kappa(t) < 0$ if the curve turns right, we will further specify this shortly.

The second derivative $\alpha''(t) = (x''(t), y''(t))$ is perpendicular to the unit vector $\alpha'(t)$, since

$$2\langle \alpha'(t), \alpha''(t) \rangle = \frac{d\langle \alpha'(t), \alpha'(t) \rangle}{dt} = \frac{d1}{dt} = 0.$$

Since $|\kappa(t)|$ is the length of $\alpha''(t)$ and $\alpha'(t)$ has length equal to one, clearly $|\kappa(t)|$ is also the area of the rectangle spanned by $\alpha'(t)$ and $\alpha''(t)$. This area is given by the determinant of the matrix that consists of the two vectors as column vectors,

$$(3) \quad \kappa(t) = \det(\alpha'(t), \alpha''(t)) = \begin{vmatrix} x'(t) & x''(t) \\ y'(t) & y''(t) \end{vmatrix} = x'(t)y''(t) - x''(t)y'(t).$$

We define the *sign of the curvature* to be the same as the sign of the determinant. This is consistent with what we have required before, see Figure 2.

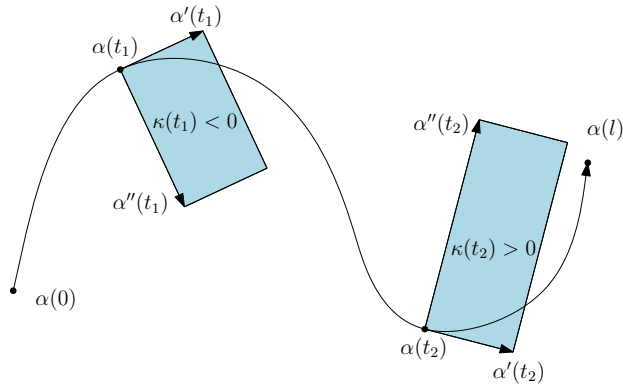


FIGURE 2. Demonstrating the meaning of the determinant in (3). The areas of the rectangles spanned by the first and second derivatives are shaded. The sign of the determinant indicates if the curve is turning right or left. The length of the vectors in the figure are not according to scale.

2.1. General Formula. Now we will derive a general formula for the signed curvature κ that is invariant under reparametrization of the curve. Consider an arbitrary curve $\gamma : (a, b) \rightarrow \mathbb{R}^2$ with length function $s : [a, b] \rightarrow [0, l]$. The curve $\gamma \circ s^{-1} = \alpha$ is a curve parametrized by arc length, which has the same trace and orientation as γ .

We can write the derivatives of α using the derivatives of γ . Since

$$\gamma = \gamma \circ s^{-1} \circ s = \alpha \circ s,$$

we have

$$\begin{aligned} \gamma(t) &= \alpha(s(t)) \\ \gamma'(t) &= \alpha'(s(t))s'(t) \\ \gamma''(t) &= \alpha''(s(t))(s'(t))^2 + \alpha'(s(t))s''(t). \end{aligned}$$

And therefore

$$\begin{aligned} \alpha'(s(t)) &= \frac{\gamma'(t)}{s'(t)} \\ \alpha''(s(t)) &= \frac{\gamma''(t) - \alpha'(s(t))s''(t)}{(s'(t))^2} = \frac{\gamma''(t) - \frac{\gamma'(t)}{s'(t)}s''(t)}{(s'(t))^2} = \frac{\gamma''(t) - c\gamma'(t)}{(s'(t))^2}. \end{aligned}$$

Those derivatives are with respect to the arc length parameter, therefore we can plug them into (3)

$$\begin{aligned}
\kappa(t) &= \det(\alpha'(s(t)), \alpha''(s(t))) \\
&= \frac{\det(\gamma'(t), \gamma''(t) - c\gamma'(t))}{(s'(t))^3} \\
&= \frac{1}{(s'(t))^3} \underbrace{(\det(\gamma'(t), -c\gamma'(t)))}_0 + \det(\gamma'(t), \gamma''(t)),
\end{aligned}$$

and obtain the *general curvature formula*

$$(4) \quad \kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}.$$

2.2. Osculating Circle. There is yet another way to characterize the curvature. It uses the notion of the *osculating circle*. Similarly to the definition of the tangential line, the osculating circle at $\alpha(t)$ is the limit of the circle passing through three points on the curve as they approach $\alpha(t)$, see Figure 3.

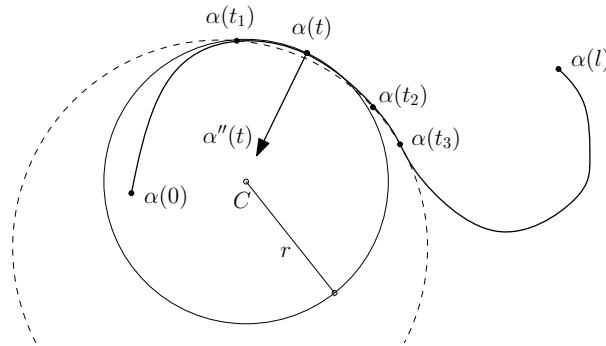


FIGURE 3. The circle through the points $\alpha(t_1), \alpha(t_2)$ and $\alpha(t_3)$ converges to the osculating circle with radius r at $\alpha(t)$ as $t_1, t_2, t_3 \rightarrow t$.

Assume α is parametrized by arc length and that the circle going through $\alpha(t_1), \alpha(t_2)$ and $\alpha(t_3)$ for $t_1 < t_2 < t_3$ exists. Let $C(t_1, t_2, t_3)$ denote the circle center. Consider the function

$$f : t \mapsto \langle \alpha(t) - C(t_1, t_2, t_3), \alpha(t) - C(t_1, t_2, t_3) \rangle.$$

By definition $f(t_1) = f(t_2) = f(t_3) = r^2$, where r is the radius of the circle through $\alpha(t_1), \alpha(t_2)$ and $\alpha(t_3)$. According to the mean value theorem f' must be zero somewhere in the intervals (t_1, t_2) and (t_2, t_3) . Let those values be $\eta_1 \in (t_1, t_2)$ and $\eta_2 \in (t_2, t_3)$,

$$\frac{f'(\eta_i)}{2} = \langle \alpha'(\eta_i), \alpha(\eta_i) - C(t_1, t_2, t_3) \rangle = 0, \quad i \in \{1, 2\}.$$

With the same argument the derivative of the function $g(t) = \frac{f'(t)}{2}$ must be zero at some point $\eta_3 \in (\eta_1, \eta_2)$,

$$g'(\eta_3) = \langle \alpha''(\eta_3), \alpha(\eta_3) - C(t_1, t_2, t_3) \rangle + \langle \alpha'(\eta_3), \alpha'(\eta_3) \rangle = 0.$$

Now, if $C = \lim_{t_1, t_2, t_3 \rightarrow t} C(t_1, t_2, t_3)$ exists, we have the two equalities

$$(5) \quad \langle \alpha'(t), \alpha(t) - C \rangle = 0$$

$$(6) \quad \langle \alpha''(t), \alpha(t) - C \rangle = -\langle \alpha'(t), \alpha'(t) \rangle.$$

We have required α to be parametrized by arc length, therefore $\alpha'(t)$ is perpendicular to $\alpha''(t)$. Since $\alpha'(t)$ is also perpendicular to $\alpha(t) - C$, which is what we have

just derived in (5), it follows that the two vectors are not linearly independent and therefore there exists some factor $c \in \mathbb{R}$, such that

$$(7) \quad \alpha(t) - C = c\alpha''(t).$$

We also know that $\|\alpha'(t)\| = 1$. Substituting in (6) gives

$$c \langle \alpha''(t), \alpha''(t) \rangle = - \langle \alpha'(t), \alpha'(t) \rangle = -1.$$

Therefore $c = -\frac{1}{\|\alpha''(t)\|^2}$. From (7) it also follows that $\|\alpha(t) - C\| = |c|\|\alpha''(t)\|$, where the left hand side is equal to the radius of the circle. We therefore obtain the relationship

$$(8) \quad \frac{1}{r} = \|\alpha''(t)\| = \kappa(t).$$

According to [Spi79, p. 6] the osculating circle exists if $\alpha''(t) \neq 0$. For $\alpha''(t) = 0$ the point $\alpha(t)$ is called a *reflection point* and $\kappa(t) = 0$. This happens whenever the curvature changes its sign. Intuitively the osculating circle can be pictured to change the side of the curve and to become a circle of infinite radius in between. The radius of the osculating circle is sometimes called the *radius of curvature* and the center of the circle is called the *curvature center*.

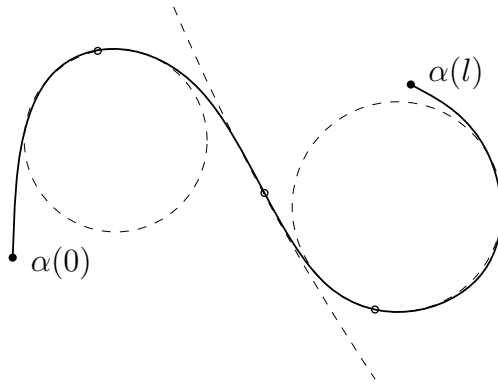


FIGURE 4. The osculating circle switches sides at reflection points.

2.3. Invariance. We motivated the study of curvature for polygonal curves with the property that it is invariant or varies only by a factor under a set of transformations that is relevant for perception. Now we want to examine this property more closely. The proofs are based upon the fact that the application of every similarity transformation, can be separated into the application of a translation part, a rotation part, a reflection and a part for uniform scaling on the data set. This allows us to analyze the transformations of one kind individually, while we retrieve a result that holds for any combination of those transformations. Let $\alpha(t) = (x(t), y(t))$ be a curve parametrized by arc length and T the transformation we are investigating. We can easily see that the curvature is invariant under translations and rotation.

Proof. T is a rigid motion that rotates every point by the angle θ around the origin and translates it by the vector (e, f) . We write the transformed curve using the variables $a = \cos \theta, b = \sin \theta$ as follows,

$$T(\alpha(t)) = \begin{pmatrix} ax(t) - by(t) + e \\ bx(t) + ay(t) + f \end{pmatrix}^T$$

We can differentiate with respect to t and plug the x and y -components into the general curvature formula (4). Since rigid motions preserve the lengths, and we

required that α is parametrized by arc length, the length of the first derivative of the transformed curve $T(\alpha)$ is one everywhere. Therefore we can drop the denominator. We simplify the numerator,

$$\begin{aligned}\kappa_{T(\alpha)}(t) &= (ax'(t) - by'(t))(bx''(t) + ay''(t)) - (ax''(t) - by''(t))(bx'(t) + ay'(t)) \\ &= (a^2 + b^2)(x'(t)y''(t) - x''(t)y'(t)) \\ &= (\cos^2 \theta + \sin^2 \theta)\kappa_\alpha(t) \\ &= \kappa_\alpha(t),\end{aligned}$$

and find that the curvature functions are equal. Therefore the curvature is invariant under translation and rotation. \square

A reflection T is a transformation, such that $T(\alpha)(t) = (-x(t), y(t))$. If we plug the derivatives with respect to t into the general curvature formula (4), we retrieve,

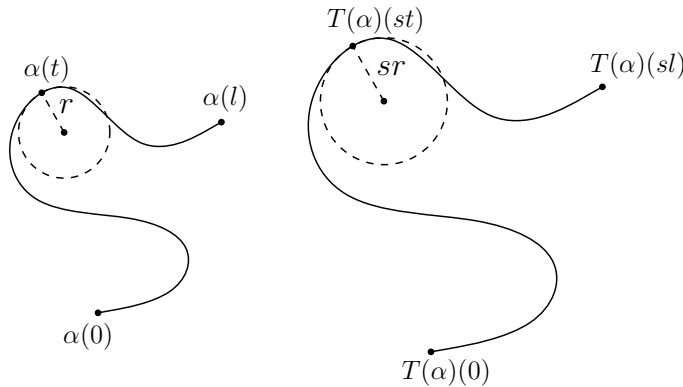
$$\kappa_{T(\alpha)}(t) = -x'(t)y''(t) + x''(t)y'(t) = -\kappa_\alpha(t).$$

Therefore we can express the curvature of the reflection of the curve by reversing the sign of the curvature.

Now let T be a uniform scaling with scaling factor s . We have mentioned that the curvature $\kappa(t)$ is equal to the inverse of the radius r of the osculating circle at $\alpha(t)$. It is clear that if we scale the curve by a factor $s > 0$ the circle scales with the same factor as well as the length of the curve l does, while the sign of the curvature does not change.

$$\kappa_\alpha(t) = \text{sign} \frac{1}{r}, \quad \kappa_{T(\alpha)}(st) = \text{sign} \frac{1}{sr} \quad \Rightarrow \quad \kappa_\alpha(t) = s\kappa_{T(\alpha)}(st)$$

Let $\beta = T(\alpha)$, then $\kappa_{T^{-1}(\beta)}(t) = s\kappa_\beta(st)$. The transformation T^{-1} is a uniform scaling with the scaling factor $\frac{1}{s}$. Therefore we can express the curvature of the scaled curve using the curvature of the original curve and the inverse of the scaling factor.



The integral of the curvature function is invariant under the scaling of the curve. With the formula we have just derived and with the substitution rule for integrals this is easy to see.

$$\begin{aligned}\int_0^l \kappa_\alpha(t) dt &= \int_0^{\frac{l}{s}} s\kappa_\alpha(su) du, \quad su = t, sdu = dt \\ &= \int_0^{\frac{l}{s}} \kappa_{T(\alpha)}(u) du,\end{aligned}$$

if T is a scaling by $\frac{1}{s}$. Indeed the transformed curve has length $\frac{l}{s}$, therefore the integral of the curvature is invariant under all of the above mentioned transformations.

With the exception that the integral of the curvature of the reflection of a curve is the negative integral of the curvature of the curve.

Another invariance we want to mention briefly is the invariance of the curvature under change of orientation of the curve. It is clear that the absolute value of the curvature does not change, since the trace of the curve stays the same. But because of the change of orientation the sign of the curvature changes.

$$\kappa_{\alpha \circ \phi}(t) = -(\kappa_{\alpha} \circ \phi)(t)$$

where ϕ is a strictly decreasing monotonic function. If α is parametrized by arc length then the function $\phi(t) = l - t$ preserves this quality.

2.4. Uniqueness. The curvature function of a curve α is unique with respect to the set of curves that are obtained by applying a rigid motion to α . This can be seen, as a curve can be given in terms of its curvature function κ , a rotation angle ψ and a translation vector (a, b) using the following formula.

$$(9) \quad \alpha(t) = \left(\int \cos \theta(t) dt + a, \int \sin \theta(t) dt + b \right), \text{ with } \theta(t) = \int \kappa(t) dt + \psi.$$

We check if the curvature function is in fact the curvature of α ,

$$\begin{aligned} |\kappa(t)| &= \|\alpha''(t)\| \\ &= \|((\cos \theta(t))', (\sin \theta(t))')\| \\ &= \|(\sin \theta(t)\theta'(t), -\cos \theta(t)\theta'(t))\| \\ &= |\kappa(t)|(\sin^2 \theta(t), \cos^2 \theta(t))^{\frac{1}{2}} \\ &= |\kappa(t)|. \end{aligned}$$

Therefore κ is the unsigned curvature of α , regardless of the values of ψ and (a, b) .

Now we want to see what influence those parameters have on the curve α . Up to now we have pictured the tangent always as a direction at a specific point on the curve. But it is in fact a vector and should be pictured at the origin. Then we can see that the unit tangent vector actually traverses a circle. By the definition of the curvature,

$$\kappa(t) = \frac{d\phi(t)}{dt},$$

where ϕ is the tangential angle of the curve, as labeled in Figure 5.

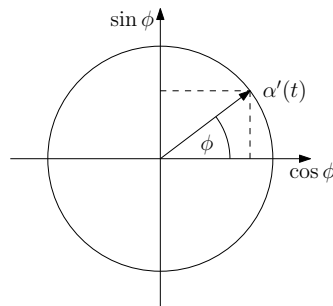


FIGURE 5. The unit tangent vector traverses the unit circle.

Therefore $\int \kappa(t) dt = \phi(t)$. Thus $\theta(t)$ in (9) returns the sum of the tangential angle of the curve α at t and a constant rotation angle. Therefore $x'(t) = \cos(\phi(t) + \psi)$ and $y'(t) = \sin(\phi(t) + \psi)$ are the derivatives of the component functions of the curve α rotated by ψ . Finally $\alpha = (x(t) + a, y(t) + b)$ is the translation of this curve with the vector (a, b) .

3. CURVATURE OF POLYGONAL CURVES

We would like to find a reasonable curvature definition for polygonal curves based on the curvature of smooth curves. Reasonable in the sense that, it should have some of the same properties and a similar meaning, but also be simple and easy to work with. More specifically, the curvature should be invariant under translation and rotation of the original curve, and should behave the same way under reflection and uniform scaling as the curvature of smooth curves does. The meaning of the curvature is more difficult to define. [Spi79] defines the curvature as a measure of how much the curve is "curving" at a point, but the term is very vague. We require, that parts of the curve which have a shape which is more pointy to have higher curvature than parts that appear more shallow or flat. We also want to benefit from the meaning of the sign of the curvature, which distinguishes right from left turns that the curve is taking.

In this chapter we will discuss four different curvature definitions that use the same set of parameters. In the last chapter we have elaborated on the different equivalent ways to define the curvature of smooth curves. Each of the four definitions takes one of the curvature definitions and transfers it to polygonal curves. We will analyze their behaviour under rigid motions and uniform scaling of the polygonal curve. And we will also analyze their relations to each other in an ideal case to see how much equivalency has been maintained.

The parameters used are namely, a parameter r which defines the sampling rate and a parameter ϵ which controls the scale, the curvature should be measured at, what this means exactly will become clearer once we get to the actual definitions. A large ϵ will allow for a smaller sampling rate and therefore a more compact representation of the curvature, but it also changes the shape of the curvature functions, since locally strong features, or noise respectively, are smoothed out.

Some of the introduced definitions or variations of them have been used in other work, for example the Gaussian curvature. But the definitions in this context are self-contained and follow directly from the discussion in the previous chapter. The Gaussian curvature is different to the other definitions, since it does not use the same set of parameters. We mainly listed it for the purpose of comparison, since convolving with the derivative of a Gaussian kernel is central in scale space theory [Lin08]. The motivation to analyse exactly those definitions was further inspired by [Bel99] and [CMT01].

3.1. Polygonal Curve Model. Our model of a polygonal curve shall be the following. A *polygonal curve* p is the union of line segments $\overline{v_{i-1}v_i}$, given by an ordered list of points $v_0, \dots, v_N \in \mathbb{R}^2$, those of which we call the *vertices*, while we call the line segments the *edges* of p . More specifically, we define a set P with respect to those vertices, and that a point is said to be *lying on p* iff it is an element of the closure of P , denoted \overline{P} .

$$(10) \quad P = \bigcup_{i=1}^N \{(1-t)v_{i-1} + tv_i \mid t \in (0, 1)\}.$$

Note that the boundary of P are the vertices of p . And note also that the reversed list of vertices and only that yields the same \overline{P} , if we demand that no three vertices are collinear. Although this condition will not be important for our purposes.

We want to find the continuous curve that has \overline{P} as its trace and is parametrized by arc length. To find this curve we first define a length function λ on \overline{P} with respect

to p , $\lambda : \overline{P} \rightarrow [0, l] \subset \mathbb{R}$.

$$(11) \quad \lambda(v) = \begin{cases} \sum_{i=1}^n \|v_i - v_{i-1}\| & v = v_i, i \in \{0, \dots, N\} \\ \lambda(v_n) + t\|v_{n+1} - v_n\| & v = (1-t)v_n + tv_{n+1}, \quad t \in (0, 1), n < N \end{cases}$$

Observe that this function is not well defined for self-intersecting curves. For an intersection point v the value of n in the expression $(1-t)v_n + tv_{n+1}$, and the value of i respectively, is not unique. This is not so much a problem as we only need the inverse of λ to be a function, as we will see.

We will now show that $\lambda : \overline{P} \rightarrow [0, l]$ is a bijective function if p has no self-intersections, while the inverse $\lambda^{-1} : [0, l] \rightarrow \overline{P}$ is a function in any case.

Proof. To show that λ is surjective, we need to find for every $x \in [0, l]$ a point $v \in \overline{P}$, such that $\lambda(v) = x$. We will find values for t and n and set $v = (1-t)v_n + tv_{n+1}$.

If $x = 0$ we set $n = 0$, $t = 0$ and if $x = l$ we set $n = N$ and $t = 0$. For $x \in (0, l)$, we set

$$(12) \quad n = \max \left\{ m \in \{0, \dots, N-1\} \mid x - \sum_{i=1}^m \|v_i - v_{i-1}\| > 0 \right\}$$

$$(13) \quad t = \frac{x - \sum_{i=1}^n \|v_i - v_{i-1}\|}{\|v_{n+1} - v_n\|}.$$

For those values the condition $\lambda(v) = x$ is clearly satisfied. We need to check that they are well-defined and that they always yield a point on the curve. For the last condition it suffices that $n \in \{0, \dots, N-1\}$ and $t \in [0, 1]$. We know that $x > 0$, therefore we can always set $m = 0$, such that the set in (12) is non-empty, and therefore n is well-defined. It is clear that n is only assigned values that m is allowed to have. Because of the condition in (12) we know that the numerator in (13) is positive, and therefore t is as well. We know that $x < l = \sum_{i=1}^N \|v_i - v_{i-1}\|$. Assume that $t > 1$, then n would not be the maximum, since

$$0 < t - 1 = x - \sum_{i=1}^{n+1} \|v_i - v_{i-1}\|, \quad n < N.$$

To prove that λ is injective, we need to show that every point that lies on p has a distinct length with respect to p . It is clear that the lengths of the vertices differ from each other and from the lengths of rest of the curve. It suffices to show this for the elements of P only. Every element of P is generated from the expression $(1-t)v_n + tv_{n+1}$. Assume there exist two distinct points $p_1, p_2 \in P$ with $\lambda(p_1) = \lambda(p_2)$. Since p_1 and p_2 are distinct, their generating values have to differ, so either $t_1 \neq t_2$ or $n_1 \neq n_2$. For every n , λ takes on values from a distinct subset of the range, more specifically

$$\sum_{i=1}^n \|v_i - v_{i-1}\| < \lambda(v) < \sum_{i=1}^{n+1} \|v_i - v_{i-1}\|, \quad v \in \overline{v_n v_{n+1}} \setminus \{v_n, v_{n+1}\}.$$

Therefore $\lambda(p_1) \neq \lambda(p_2)$ if $n_1 \neq n_2$. Now there is still the possibility that $n_1 = n_2$ and $t_1 \neq t_2$, but this cannot happen, since $t \in (0, 1)$. Both analyses also hold if p has self-intersections. Therefore λ^{-1} is a function. \square

Now we can write P as the trace of the continuous curve $\alpha(s) = \lambda^{-1}(s)$. To prove continuity we need to show that for every $\epsilon > 0$ we can find a $\delta > 0$ such that,

$$\forall s, t \in [0, l] : |s - t| \leq \delta \Rightarrow \|\alpha(s) - \alpha(t)\| \leq \epsilon.$$

Proof. We can set $\delta = \epsilon$. Intuitively we are showing that if we traverse a certain length on the curve then we can get at most this much further away. Let $p = \alpha(s), q = \alpha(t)$ we can show that $|\lambda(p) - \lambda(q)| \leq \delta \Rightarrow \|p - q\| \leq \epsilon$. If the covered part of the curve is a straight line then $|\lambda(p) - \lambda(q)| = \|p - q\|$, otherwise we have passed a finite number of vertices and have traversed the straight edges in between. So we can apply the triangle inequality a finite number of times and obtain $\|p - q\| \leq |\lambda(p) - \lambda(q)| \leq \delta = \epsilon$. \square

3.2. Curvature Model. With the representation of the polygonal curve as a continuous parameterized curve α in the last section it is clear that the original definition of curvature is not reasonable, in the sense we described in the beginning of the chapter. The component functions of α are piecewise linear functions, therefore they are continuous, but obviously they are not differentiable at the vertices. No matter how many vertices we pick to design our polygonal curve and no matter how close we pick them, the curvature function maps almost every point to zero, that is all but a finite set of points, the set of the vertices, where it is undefined. Therefore we need to modify the definition to retain the curvatures meaning. Our curvature definitions use the same set of parameters. Therefore we first discuss the generic curvature definition before we get to the actual definitions.

3.2.1. Sampling Rate. We think of the given polygonal curves as approximations of the smooth curves that they still represent visually. Therefore it would be delusive to try to measure a curvature continuously along the curve and think of the measurement as exact. We will measure the curvature at finitely many points and linearly interpolate between those points, just as it is done with the input curves.

The number of points should be chosen in the order of the number of vertices, but can also be reduced to achieve a more compact, albeit lossy, representation. Another global parameter is therefore the sampling rate $r < 1$. Let κ be the curvature sampled at finitely many points $t_i = ir\lambda(v_N), i \in \{0, \dots, \frac{1}{r}\}$. Then the resulting curvature $\hat{\kappa}$ will be,

$$\hat{\kappa}(t_i + t) = (1 - t)\kappa(t_i) + t\kappa(t_{i+1}), \quad t \in [0, r\lambda(v_N)].$$

3.2.2. Scale of the Curvature. The curvature is a measure of second order, it gives information about how the curve is shaped in the area surrounding the point it is measured at. Since we cannot take the limit we need to define a parameter as for the size of the neighbourhood that this information should be taken from. More precisely, for each curvature measurement at a specific point we will take three samples from the curve, namely the point and two points that have the arc length distance of ϵ from this point. Some features that contribute to the curvature at high scale will be smoothed out if the curvature is measured at a lower scale, i.e. with a larger ϵ . For example in the lower curve of Figure 6 the absolute value of the curvature will always be small when measured at a low scale, but when measured at a high scale it will be at some points very high and therefore accounts for more details. In the upper curve the sign of the curvature will not change along the curve when measured at a low scale, therefore according to this measurement the curve only makes one "turn". But when measured at a higher scale the curvature accounts also for the other two turns the curve makes in between. Therefore the parameter ϵ controls the scale, which the curvature is measured at or the smoothing.

This characterization of the chosen parameter and the examples are supposed to give an idea of the notion of the *scale of the curvature*. We will get more precise once we get to the actual definitions where the parameter is being used. And it will

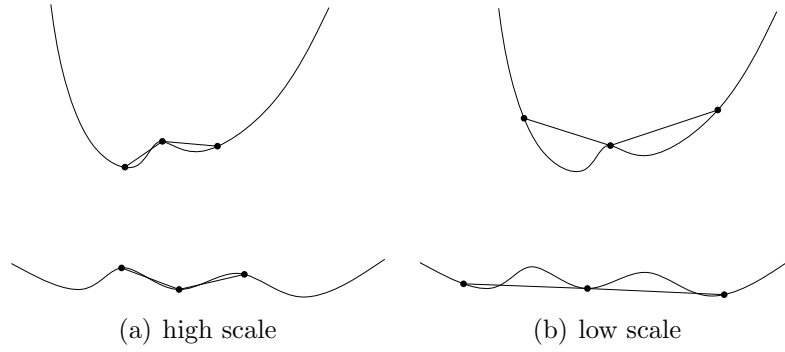


FIGURE 6

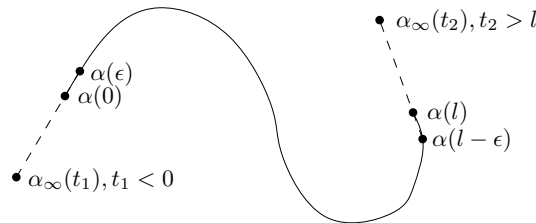
become clear with the examples of curvature functions computed according those definitions.

It is clear that a too small ϵ yields points that lie on a line for a sampling point on an edge and in this case a curvature definition can only be zero, which is undesirable since it leads to a noisy curvature function. If we want to measure the curvature at this scale there are several ways to deal with this problem. If we approach the polygonal curve as the approximation of a differentiable curve, then we can require that the sampling rate should be chosen in a way such that the edge lengths fall below a certain threshold. If the curves are given to us and we have no influence on the angles between consecutive edges, we could still apply a subdivision scheme of our choice for an appropriate number of iterations to retrieve a polygonal curve that is better suitable and still has the same appearance. Or we can simply smoothen the resulting curvature function.

In this context we note the minor problem that the talked about neighbourhood is missing at the ends of the curve. To address this we could shrink the domain of the curvature function. But we are interested in the curvatures being zero at the endings. Therefore we extend the curve endings to fill in the missing information as it is often done in the context of subdivision. The specification is as follows,

$$\alpha_\infty(t) = \begin{cases} \alpha(0) + (-t) \frac{\alpha(0) - \alpha(\epsilon)}{\|\alpha(0) - \alpha(\epsilon)\|} & t < 0 \\ \alpha(t) & 0 \leq t \leq l \\ \alpha(l) + (t - l) \frac{\alpha(l) - \alpha(l - \epsilon)}{\|\alpha(l) - \alpha(l - \epsilon)\|} & t > l \end{cases}$$

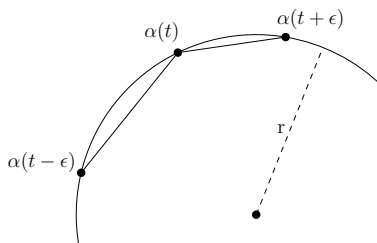
where $l = \lambda(v_N)$.



One could argue that the neighbourhood is not exactly missing. A circle is an osculating circle to a specific point on the curve if curve and circle pass through the point with the same second derivative. This information could also be measured with a one-sided limit. But we have to decide for a direction and therefore the neighbourhood will still be missing at one end of the curve. We will actually do this with the curvature definitions that are based on discrete derivatives.

3.3. Radius of Curvature. We discussed the equivalence of the inverse of the radius of the osculating circle to the curvature. The most simple of our definitions makes use of this equivalence. Let r be the radius of the circle C that passes through $\alpha(t-\epsilon)$, $\alpha(t)$, and $\alpha(t+\epsilon)$, as shown in the figure, then the curvature can be defined as

$$\kappa_o(t) = \text{sign} \frac{1}{r},$$



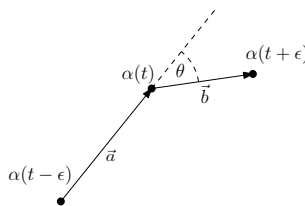
The sign is positive iff $\alpha(t+\epsilon)$ is left of the directed line that goes through $\alpha(t-\epsilon)$ and $\alpha(t)$. For the case that the points are collinear, we define a line to be a circle with infinite radius and the reciprocal of that radius to be zero.

The curvature defined this way is clearly invariant under rigid motions since the three points form a triangle that has C as its circumcircle, which is determined only by an angle and the length of the opposite side. It seems that we could apply the argument for the scale-invariance of the curvature of smooth curves, which we discussed in (2.3), to this definition easily since this also uses the radius of the circle. But in order to keep the same three points in the neighbourhood we also need to scale the parameter ϵ . We defined ϵ to be absolute, we could instead choose ϵ to be relative to the diameter of the bounding box of the curve. However if we insist on an absolute ϵ we have to be aware of the fact that it is not scale-invariant in the sense that the scale the curve is given to us initially is irrelevant. Naturally those considerations also apply to the following definitions.

3.4. Turning Angle Curvature. As mentioned in section (2.4) the curvature can be defined as the derivative of the tangential angle with respect to the arc length. The next definition takes the difference of *turning angles* of two consecutive edges. Where the turning angle is defined as the angle which is spanned by the edge and a reference line. Since we are taking the difference, the reference is irrelevant.

Let $\vec{a} = \alpha(t) - \alpha(t-\epsilon)$, and $\vec{b} = \alpha(t+\epsilon) - \alpha(t)$ as shown in the figure, then the turning angle curvature is defined as

$$\kappa_a(t) = \text{sign} \frac{2|\angle(\vec{a}, \vec{b})|}{\|\vec{a}\| + \|\vec{b}\|}$$



The absolute angle can be computed using the dot product $|\angle(\vec{a}, \vec{b})| = \arccos \left\langle \frac{\vec{a}}{\|\vec{a}\|}, \frac{\vec{b}}{\|\vec{b}\|} \right\rangle$. Observe that although the angle will be a value between zero and π the curvature can still take on values arbitrarily large if the lengths of the edges \vec{a} and \vec{b} are sufficiently small. But depending on the choice of ϵ there is a bound on the value of the curvature. The sign is positive iff $\alpha(t+\epsilon)$ is left of the directed line that goes through $\alpha(t-\epsilon)$ and $\alpha(t)$.

It is clear that the angle difference and the lengths of the edges are invariant under rigid motions, since rigid motions preserve lengths and relative angles. In analogy to the discussion in (2.3), let $T(\alpha)$ be the curve obtained by uniformly scaling α with the scaling factor s . Then we can write with respect to the curvature function

we just defined,

$$\kappa_{T(\alpha)}(st) = \text{sign} \frac{2|\angle(\vec{a}, \vec{b})|}{s(\|\vec{a}\| + \|\vec{b}\|)} = \frac{1}{s} \kappa_\alpha(t).$$

Therefore $s\kappa_{T(\alpha)}(st) = \kappa_\alpha(t)$ or equally $\kappa_{T^{-1}(\alpha)}(t) = \hat{s}\kappa_\alpha(\hat{s}t)$, where $\hat{s} = \frac{1}{s}$. Therefore the curvature defined this way behaves the same way under uniform scaling as the curvature of smooth curves does, except that we, again, have to define ϵ to be a relative parameter, in order for this to work.

3.5. Norm of the Second Derivative. The curvature of smooth curves is equal to the norm of the second derivative, if differentiated with respect to the arc length. This is our next approach to a curvature definition. Let

$$\kappa_s(t) := \text{sign} \frac{\|\alpha(t) - 2\alpha(t - \epsilon) + \alpha(t - 2\epsilon)\|}{\epsilon^2}$$

This curvature definition might for some readers reveal what we mean, when we say that ϵ controls the scale the curvature is measured at. We basically want to measure the curvature with respect to the scale of the curve where the length of the discrete derivative is everywhere equal to one or at least within some small error, see figure (7). The *discrete derivative* of a sequence is the sequence, $f : \mathbb{N} \rightarrow U$, $f'(t) = f(t) - f(t - 1)$.

We assume that the error we make is small, when we set $\|T(\alpha)'(s(t))\| = 1$, since we require the polygonal curve to be sampled with a high sampling rate on a smooth curve, such that the angles between consecutive edges get very small. But naturally the error will be large if we choose ϵ too large.

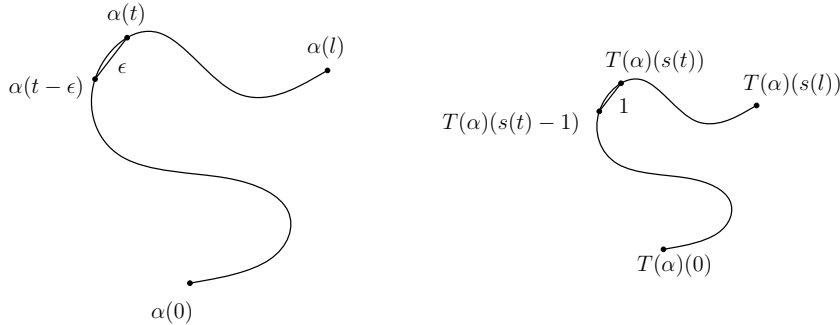


FIGURE 7

We reparametrized α with $s(t) = \frac{t}{\epsilon}$ and differentiated with respect to this parametrization. The discrete derivatives are,

$$\begin{aligned} \alpha'(t) &= \alpha(t) - \alpha(t - 1) \\ \alpha''(t) &= \alpha'(t) - \alpha'(t - 1) = \alpha(t) - 2\alpha(t - 1) + \alpha(t - 2) \end{aligned}$$

Using the chain rule and the product rule we retrieve

$$\begin{aligned} (\alpha(s(t)))'' &= (\alpha'(s(t)))'s'(t) \\ &= \alpha''(s(t))s'(t)^2 + \alpha'(s(t))s''(t) \\ &= \left(\alpha\left(\frac{t}{\epsilon}\right) - 2\alpha\left(\frac{t}{\epsilon} - 1\right) + \alpha\left(\frac{t}{\epsilon} - 2\right) \right) \frac{1}{\epsilon^2} \end{aligned}$$

And therefore

$$((\alpha \circ s)'' \circ s^{-1})(t) = \frac{\alpha(t) - 2\alpha(t - \epsilon) + \alpha(t - 2\epsilon)}{\epsilon^2}$$

which is basically our formula. The sign is positive iff $\alpha(t)$ is left of the directed line that goes through $\alpha(t - 2\epsilon)$ and $\alpha(t - \epsilon)$. The current curvature definition is consistent with the previous two definitions in that it uses the same neighbourhood points on the curve, only shifted by ϵ .

To argue that the curvature defined this way is invariant under rigid motions, we deliver a geometric interpretation of the enumerator. Consider the vectors \vec{a} , \vec{b} and \vec{d} in Figure 8(a).

$$\vec{d} = \vec{b} - \vec{a} = (\alpha(t) - \alpha(t - \epsilon)) - (\alpha(t - \epsilon) - \alpha(t - 2\epsilon))$$

Therefore the absolute value of the curvature in the current definition is equal to the length of the vector \vec{d} divided by ϵ^2 , which is clearly invariant under rigid motion for fixed parameter ϵ .

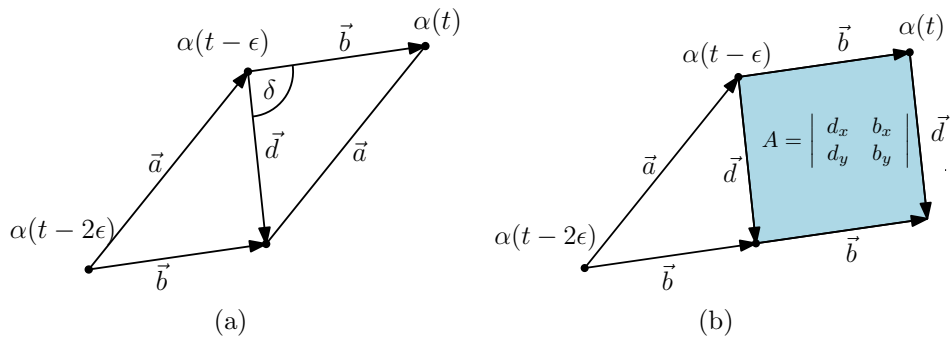


FIGURE 8. While the vectors \vec{b} and \vec{a} are geometric interpretations of the first derivative at t and $t - \epsilon$. The vector \vec{d} is a geometric interpretation of $\alpha''(t)$. Note that δ is not necessarily a right angle, as it would be the case with smooth curves. The area of the spanned parallelogram by \vec{b} and \vec{d} is a geometric interpretation of the enumerator in the direct curvature definition $\alpha''(t)_x \alpha'(t)_y - \alpha'(t)_x \alpha''(t)_y$.

As for the case of uniform scaling we can say that for T being a scaling transformation with factor s ,

$$\kappa_{T(\alpha)}(st) = \text{sign} \frac{s \|\vec{d}\|}{(s\epsilon)^2} = \frac{1}{s} \kappa_\alpha(t),$$

and therefore $s\kappa_{T(\alpha)}(st) = \kappa_\alpha(t)$ or equally $\kappa_{T^{-1}(\alpha)}(t) = \hat{s}\kappa_\alpha(\hat{st})$, where $\hat{s} = \frac{1}{s}$. Therefore the curvature defined this way behaves the same way under uniform scaling as the curvature of smooth curves does, except that we, again, need ϵ to be defined relative to the curve.

3.6. Direct Curvature. If we take the discrete derivatives of a sequence of points sampled on the polygonal curve, we can apply the general formula directly and hope that everything else will be taken care of by the formula, since it is invariant under reparametrization in the case of smooth curves. Let

$$\kappa_d(t) = \frac{\alpha'(t)_x \alpha''(t)_y - \alpha''(t)_x \alpha'(t)_y}{\|\alpha'(t)\|^3},$$

where v_x and v_y denote the x and y -component of $v \in \mathbb{R}^2$ and derivatives are

$$\begin{aligned} \alpha'(t) &= \alpha(t) - \alpha(t - \epsilon) \\ \alpha''(t) &= \alpha'(t) - \alpha'(t - \epsilon) = \alpha(t) - 2\alpha(t - \epsilon) + \alpha(t - 2\epsilon). \end{aligned}$$

The current curvature definition uses the same neighbourhood points on the curve, as the previous definition does.

We can give geometric interpretations of terms in this formula to show the invariance under rigid motions. The enumerator in the formula describes the determinant of the two vectors \vec{d} and \vec{b} in Figure 8(b) and therefore also the area of the parallelogram spanned by these vectors. The denominator is only dependent on the length of \vec{b} . It is clear that this area does not change if we translate or rotate the curve.

Let T be a scaling transformation with factor s ,

$$\kappa_{T(\alpha)}(st) = \text{sign} \frac{s^2 A}{(s\vec{b})^3} = \frac{1}{s} \kappa_\alpha(t),$$

Therefore the general curvature formula behaves in the discrete setting equivalently under uniform scaling as when applied to smooth curves, except that we have to set ϵ relative to the curve.

3.7. Differentiation using Convolution. The *convolution* of the two discrete functions $f : \{0, \dots, n-1\} \rightarrow U$ and $g : \{-m, \dots, m\} \rightarrow \mathbb{R}$, where g is often called the *kernel*, and often $m \ll n$, is defined as

$$(f * g)(t) = \sum_{k=t-m}^{t+m} f(k)g(t-k), \quad t \in \{0, \dots, n-1\},$$

outside the domain of the functions the function values are defined to be zero.

It is known that the derivative of the convolved function can be obtained by convolving the function with the derivative of the kernel. And the same holds for the second derivative, as in $(f * g)''(t) = (f * g'')(t)$. Consider for example the identity kernel $id = (0, 1, 0)$. The derivative of the identity kernel is $id' = (0, 1, -1)$, convolving with id' yields the discrete derivative $(f * id')(n) = -f(n-1) + f(n) = f'(n)$.

We can see the general property as follows,

$$\begin{aligned} (f * g)'(t) &= (f * g)(t) - (f * g)(t-1) \\ &= \sum_{k=t-m}^{t+m} f(k)g(t-k) - \sum_{k=t-m}^{t+m} f(k)g(t-1-k) \\ &= \sum_{k=t-m}^{t+m} (f(k)(g(t-k) - g(t-1-k))) \\ &= \sum_{k=t-m}^{t+m} f(k)g'(t-k) \\ &= (f * g')(t). \end{aligned}$$

One can think of different kernels to obtain derivatives with different properties. Often times a Gaussian kernel is being used. Which can be defined as follows. Recall the definition of the normal distribution with mean zero¹,

$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}},$$

¹The normalization factor can be dropped in the computations, since the kernel will be renormalized.

with σ the standard deviation. We define the one-dimensional Gaussian kernel,

$$G_\sigma : \{-\sigma, \dots, \sigma\} \rightarrow \mathbb{R}$$

$$G_\sigma(t) = \frac{g_\sigma(t)}{\sum_{k=-\sigma}^{\sigma} g_\sigma(k)}$$

The parameter σ is also said to control the *size* of the kernel. An approximation for a kernel with $\sigma = 1$ is, for example $(\frac{3}{11}, \frac{5}{11}, \frac{3}{11})$. The Gaussian kernel and its derivatives are the central operators in scale space theory. It is widely used in computer vision to eliminate noise in images, but the kernel can be defined in any dimension and is also being applied to curves.

If a is a sequence of points on the curve α , then the sequence of points $(a * G_\sigma)$ is also a polygonal curve and describes a smoothed version of α . Furthermore $(a * G'_\sigma)$ and $(a * G''_\sigma)$ are equal to the first and second derivatives of the smoothed curve.

Using the general curvature formula from equation (4) we define the *Gaussian direct curvature*² denoted by κ_g analogously to the direct curvature.

$$\kappa_g(t) = \frac{(a * G'_\sigma)(t)_x (a * G''_\sigma)(t)_y - (a * G''_\sigma)(t)_x (a * G'_\sigma)(t)_y}{\|(a * G'_\sigma)(t)\|^3},$$

The curvature defined this way should be equally invariant under rigid motions for a fixed Gaussian kernel and a fixed sampling rate to generate the sequence of points a .

3.8. Equivalences in the Ideal Case. We have derived the definitions for the curvature of polygonal curves on the basis of equivalent curvature definitions for smooth curves. Now it would be interesting to see how much equivalency between the definitions has been maintained. Consider the case that the curvature is being measured at three equidistant points and that in addition to that the arc length between the points along the curve is equal to their euclidian distance. In the following paragraphs let x denote the length of the vector \vec{x} .

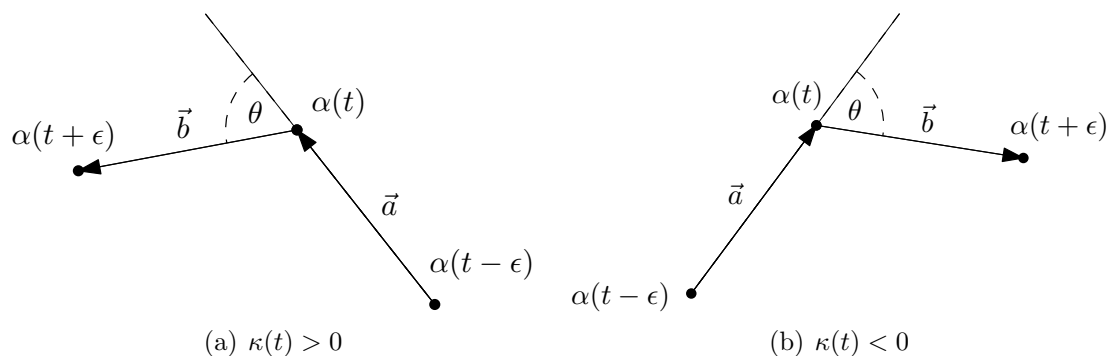


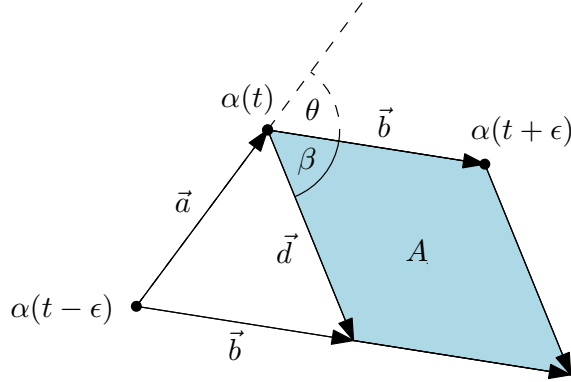
FIGURE 9. Consider the elementary case that $a = b = \epsilon$.

The case we are investigating is the most simple case for the analysis of the proposed curvature definitions. Relations between the curvature definitions can be easily established using basic trigonometric concepts. It is clear that the curvature definitions are consistent in their sign. Therefore we will only analyze the case displayed in (9(b)) and the other case will be analogous.

²Not to be confused with the Gaussian curvature of surfaces.

3.8.1. κ_s and κ_d . We have already delivered simple geometric interpretations for some of the terms in the formula of the length of the second derivative curvature definition (3.5) and the direct curvature definition (3.6). For the considered case we can express the curvatures as follows, labels are with respect to the figure below,

$$\kappa_d(t + \epsilon) = -\frac{A}{b^3}, \quad \kappa_s(t + \epsilon) = -\frac{d}{\epsilon^2}.$$



Since the area A of the parallelogram can also be expressed with the formula $bd \sin \beta$, it follows directly that

$$\kappa_d(t + \epsilon) = -\frac{bd \sin \beta}{b^3} = -\frac{d \sin \beta}{\epsilon^2} = \kappa_s(t + \epsilon) \sin \beta.$$

We can set β in relation to the difference of turning angles of the two edges, θ , as $\pi = 2\beta + \theta$. In our setting $\beta, \theta > 0$ and therefore $0 \leq \beta \leq \frac{\pi}{2}$. We have derived that κ_s and κ_d are nearly equal iff β is near $\frac{\pi}{2}$. This happens when the θ is very small.

3.8.2. κ_o and κ_a . Now we want to analyze the relation of the radius of curvature (3.3) and the turning angle curvature (3.4). We can express the curvatures using the labels in Figure 10(a),

$$\kappa_o(t) = -\frac{1}{r}, \quad \kappa_a(t) = -\frac{\theta}{a}.$$

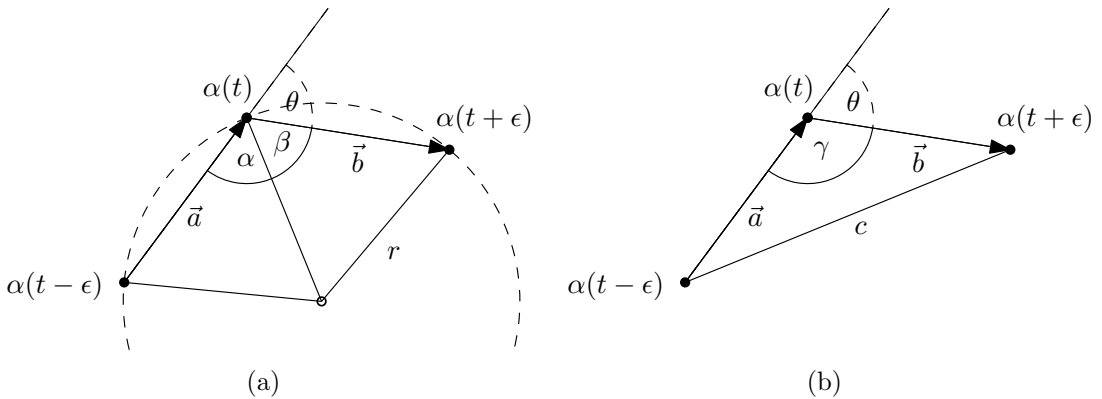


FIGURE 10

Since we required the three points to be equidistant along the curve we can conclude from the rule of cosine, that

$$\cos \alpha = \frac{a^2 + r^2 - r^2}{2ar} = \frac{a}{2r},$$

and since $\pi = \theta + 2\alpha$ and in our setting $\alpha, \theta > 0$

$$\theta = \pi - 2 \arccos \frac{a}{2r} = 2 \arcsin \frac{a}{2r} \Rightarrow \sin \frac{\theta}{2} = \frac{a}{2r}.$$

We obtain

$$(14) \quad \sin \frac{a\kappa_a(t)}{2} = \frac{a\kappa_o(t)}{2},$$

using $\sin(-x) = -\sin(x)$. For positive x it holds that $\sin(x) \leq x$, since the two curvatures always have the same sign, we can directly conclude that $|\kappa_a(t)| \geq |\kappa_o(t)|$.

An upper bound for their difference can be retrieved as follows. The circle through $\alpha(t)$, $\alpha(t - \epsilon)$ and $\alpha(t + \epsilon)$ with radius r is the circumcircle of the triangle defined by these points, and since $\sin \theta = \sin \gamma$, θ and r have the following relationship.

$$2r = \frac{c}{\sin \gamma} \Rightarrow \sin \theta = \frac{c}{2r}$$

Therefore

$$(15) \quad \sin(a\kappa_a(t)) = \frac{c}{2}\kappa_o(t).$$

We want to find an upper bound for the difference, let $\kappa_a(t) < 0$, $\kappa_o(t) < 0$, which is the case we are considering

$$(16) \quad \kappa_o(t) - \kappa_a(t) = \frac{2 \sin(a\kappa_a(t))}{c} - \kappa_a(t).$$

We can find an expression for c with respect to θ using the law of cosine,

$$c = \sqrt{a^2 + b^2 + 2ab \cos \theta} = \sqrt{2a^2(1 + \cos \theta)} = 2a \cos \frac{\theta}{2}.$$

Therefore the right hand side in (16) is

$$\text{rhs} = \frac{2 \sin(-\theta)}{2a \cos \frac{\theta}{2}} - \kappa_a(t).$$

We can rewrite this to express the difference relative to the curvature value $\kappa_a(t)$,

$$\text{rhs} = \left(\frac{\sin -\theta}{\kappa_a(t)a \cos \frac{\theta}{2}} - 1 \right) \kappa_a(t) = \left(1 - \frac{\sin \theta}{\theta \cos \frac{\theta}{2}} \right) (-\kappa_a(t)) = \alpha(-\kappa_a(t)).$$

Now find out how large the factor α can be for $0 \leq \theta \leq \pi$,

$$\alpha = 1 - \frac{\sin \theta}{\theta \cos \frac{\theta}{2}} = 1 - \frac{2 \sin \frac{\theta}{2}}{\theta} \stackrel{(*)}{\leq} 1 - \frac{2(\frac{\theta}{2} - \frac{\theta^3}{2^3 3!})}{\theta} \leq \frac{\theta^2}{24},$$

where (*) follows from the series definition of sine. Therefore we have for negative curvature,

$$|\kappa_o(t) - \kappa_a(t)| \leq \frac{\theta^2}{24} |\kappa_a(t)|.$$

And the case where $\kappa_a(t) > 0$ and $\kappa_o(t) > 0$ is geometrically equivalent. We conclude that for small θ the two curvature functions are very close to each other, but with increasing angle θ they may diverge relatively fast. The latter follows from the initial equivalence we derived for the factor α .

3.8.3. κ_a and κ_s . Now we want to investigate

$$\kappa_s(t + \epsilon) = -\frac{d}{\epsilon^2}, \text{ and } \kappa_a = -\frac{\theta}{a}.$$

We can establish a relationship between the angle θ and the norm of the second derivative d using the law of cosine.

$$d = \sqrt{a^2 + b^2 - 2ab \cos \theta} = \sqrt{2a^2(1 - \cos \theta)} = 2a \sin \frac{\theta}{2}$$

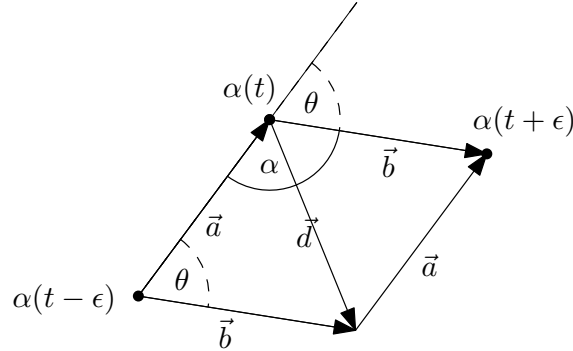


FIGURE 11

We required $\epsilon = a$, therefore

$$\kappa_s(t + \epsilon) = -\frac{2 \sin \frac{\theta}{2}}{a} \Rightarrow \frac{a\kappa_s(t + \epsilon)}{2} = \sin \frac{a\kappa_a(t)}{2}.$$

Therefore the relationship we have derived before for $\kappa_a(t)$ and $\kappa_o(t)$ also holds with respect to $\kappa_s(t + \epsilon)$, see Equation (15). And we can also conclude the surprising direct equality $\kappa_s(t + \epsilon) = \kappa_o(t)$. According to our analysis the norm of the second derivative curvature should yield the same function as the radius of curvature under the same choice of parameters. The special case we are investigating is the ideal case, but it is also an unlikely setting. It will happen only if the polygonal curve is already sampled such that all edges have the same length and that we pick the parameter ϵ as the edge length and the sampling rate such that we sample only the vertices. But this analysis justifies our curvature definitions, since it shows that they are almost equal in an ideal setting. The curves where those curvature definitions deviate in the ideal case should not be considered *smooth*, since the differences in tangential angles of consecutive edges are too large.

3.9. Examples. The following examples show the curvature functions for the displayed curves from high scale to low scale. The vertices of the first example curve are more equally distributed on the curve than the vertices of the second example curve. This is probably why the definitions that are based on derivatives in Figures 17(c), 17(d), and 17(e) seem to respond to noise frequencies, which is undesirable. We have not investigated this any further, we simply suggest to use one of the other two definitions.

The curves in Figures 14 and 16 have local similarities, but they are not similar in their appearance as a whole. We can see this to some extent reflected in their curvature functions. The curvature functions are locally similar at a high scale, but they are very different at a low scale. We conjecture that, if we were to match those two curves, we would have to match them at different curvature scales simultaneously.

The third example curve is in reference to [Spi79]. Spivak introduces the concept of curvature in an interesting way. According to Spivak the curvature is supposed

to be a measure of how much the curve is "curving" at any point. He goes on "No matter how vague, as yet intuitive, the term may be, we will surely all agree that

- (a) a straight line is not curving at all
- (b) a circle of radius $R > r$ is curving less than a circle of radius r ."

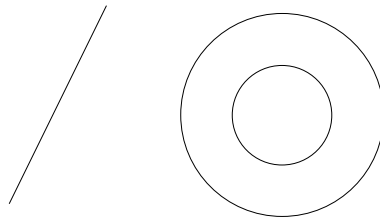


FIGURE 12. Special cases that reveal the concept of curvature according to [Spi79].

He shows that one can build the general definition of curvature upon this. Therefore we have included an example that consists of approximations of circular arcs and straight lines by a polygonal curve in Figure 18. In the last row, which displays the Gaussian curvature it can be observed that the curvature actually increases under smoothing. This is due to the fact that the Gaussian kernel shrinks the curve at high curvature points. Worring and Smeulders write in [WS93] ". . . such Gaussian smoothing of the path causes a shrinking effect on the curve. That is, locally the smoothed curve tends to move towards the local center of curvature."

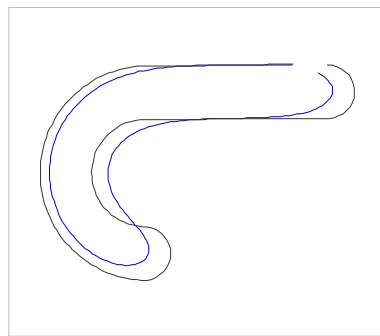


FIGURE 13. Shrinking of the curve under the Gaussian smoothing.

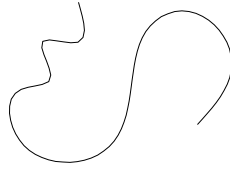
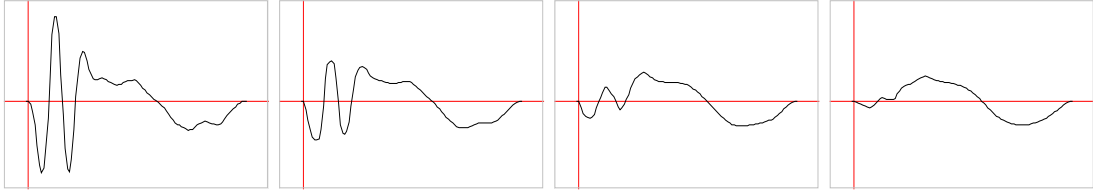
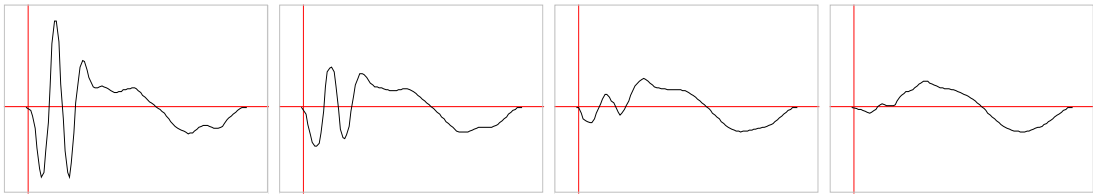


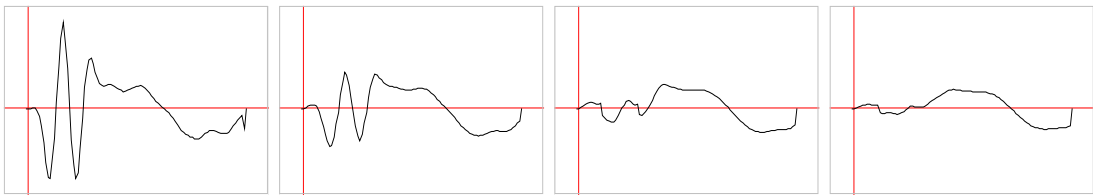
FIGURE 14



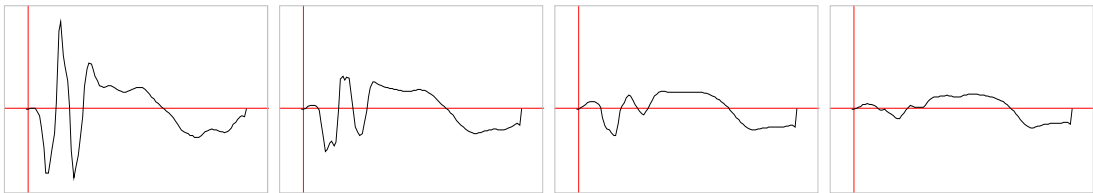
(a) Radius of curvature as defined in (3.3).



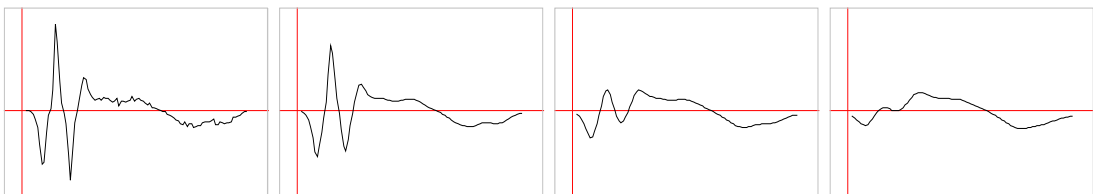
(b) Turning angle curvature as defined in (3.4).



(c) Norm of second derivative curvature as defined in (3.5).



(d) Direct curvature as defined in (3.6).



(e) Gaussian direct curvature as defined in (3.7).

FIGURE 15. Curvature functions of the curve in Figure 14 from left to right measured from high scale to low scale, i.e. in the figures to the left ϵ is smaller than in the figures to the right.

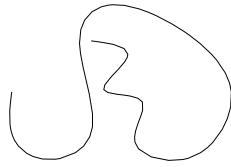
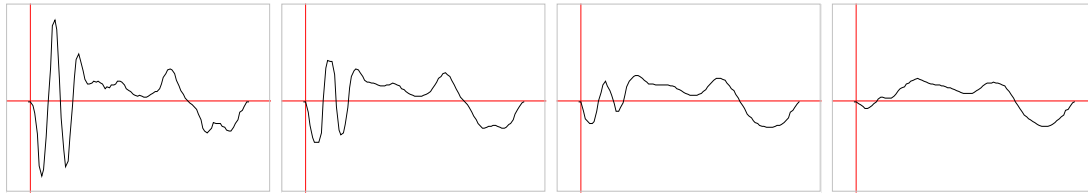
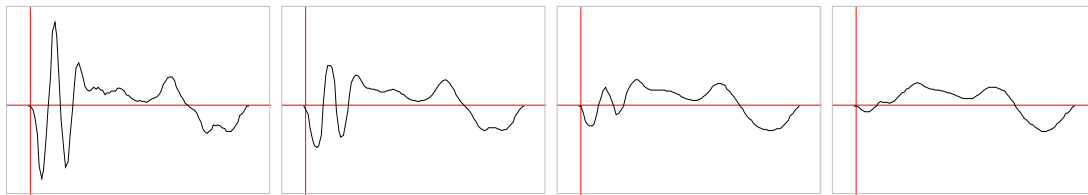


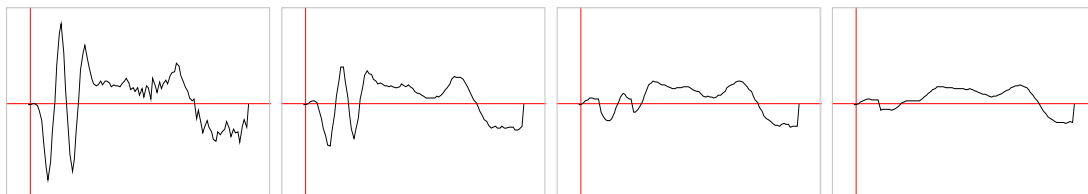
FIGURE 16



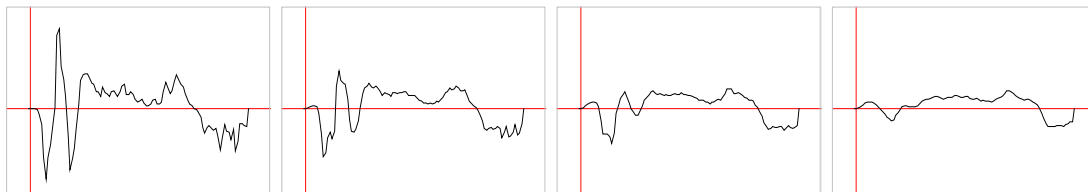
(a) Radius of curvature as defined in (3.3).



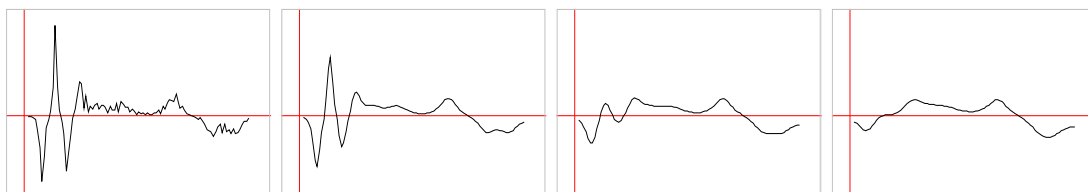
(b) Turning angle curvature as defined in (3.4).



(c) Norm of second derivative curvature as defined in (3.5).



(d) Direct curvature as defined in (3.6).



(e) Gaussian direct curvature as defined in (3.7).

FIGURE 17. Curvature functions of the curve in Figure 16 from left to right measured from high scale to low scale, i.e. in the figures to the left ϵ is smaller than in the figures to the right.

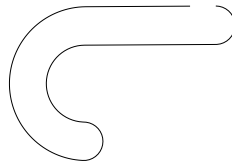
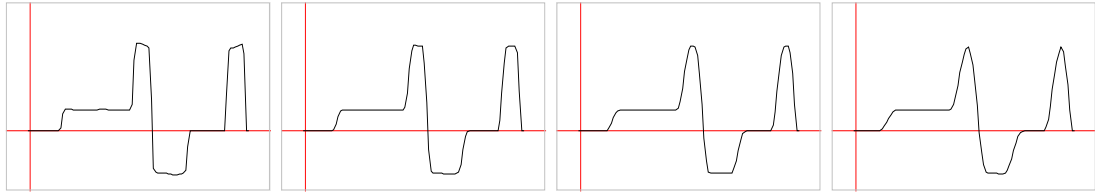
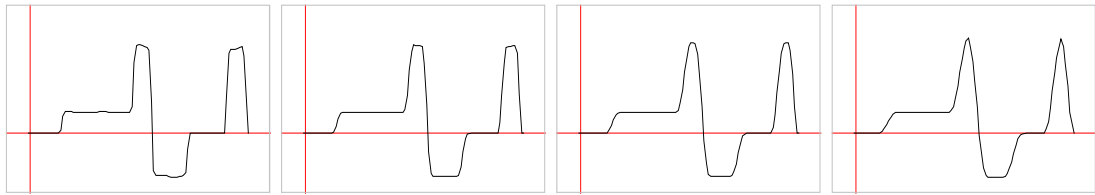


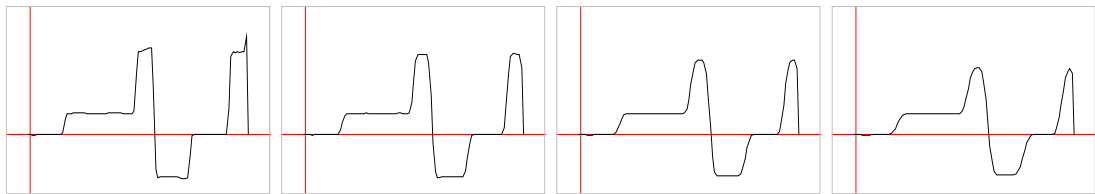
FIGURE 18



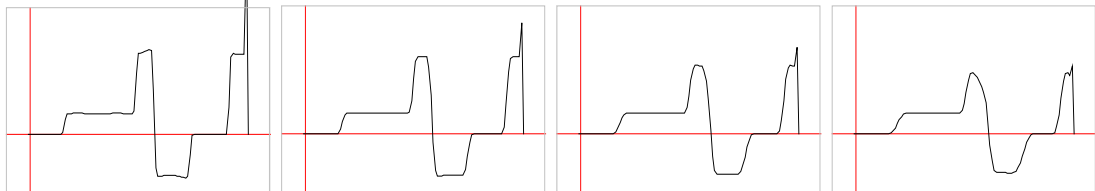
(a) Radius of curvature as defined in (3.3).



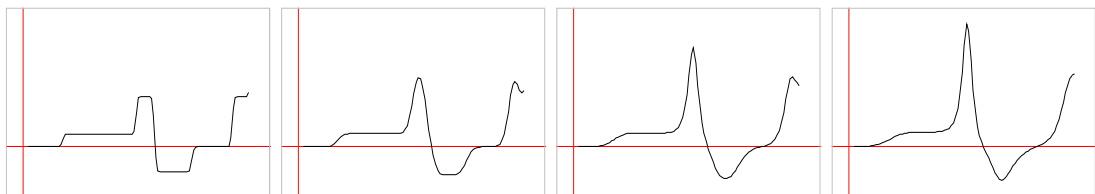
(b) Turning angle curvature as defined in (3.4).



(c) Norm of second derivative curvature as defined in (3.5).



(d) Direct curvature as defined in (3.6).



(e) Gaussian direct curvature as defined in (3.7).

FIGURE 19. Curvature functions of the curve in Figure 18 from left to right measured from high scale to low scale, i.e. in the figures to the left ϵ is smaller than in the figures to the right.

4. DEFINITION OF SIMILARITY/DISTANCE

A prerequisite of determining the similarity of things is a concept of what can make them similar, or dissimilar. A mathematically sound definition of such a concept is a distance measure. A *distance measure* is a function that assigns a non-negative value, the distance, to every two-combination of the objects to be compared. A large distance means the two objects are dissimilar, a small distance means they are similar and a distance of zero can only be assigned if the objects are identical. If the triangle inequality holds for the distance measure it is said to be a metric and the objects together with the distance measure form a metric space.

The concept of similarity of polygonal curves is often based upon the similarity transformations, which we know the curvature is invariant under, see Sections 2.3 and references to this section in the respective subsections of 3. For two polygonal curves to be similar means, that there exists a similarity transformation, that, applied to one of the curves, yields the other curve or a curve that diverges very little in the absolute placement of the vertices from the vertices of the other polygonal curve. The set of objects we want to be able to compare is the set of all polygonal curves, each of which we are given the curvature functions. Each curvature function κ is given as a finite set of assignments

$$\mathcal{A}_\kappa = \{(k_i, v_i) \in \mathbb{R} \times \mathbb{R} \mid k_i < k_{i+1}, i \in \{0, \dots, n\}\},$$

where the differences of two consecutive keys should be equal $k_{i+1} - k_i = k_i - k_{i-1}$.

We want to find a distance measure $\delta : \{\mathcal{A}\}^2 \rightarrow \mathbb{R}^+$, such that $\delta(\kappa_P, \kappa_Q)$ is small iff two polygonal curves P and Q , such that κ_P is the curvature of P and κ_Q is the curvature of Q , are similar. For δ to induce a distance measure on the set of polygonal curves the curvature functions needed to be unique. This is true of the curvature of smooth curves, but we have not shown it for polygonal curves. We conjecture that it is not true. In fact, for the curvature definitions we proposed a curvature function could only be unique with respect to a certain choice of parameters.

Therefore we compare the equivalence classes of polygonal curves that have the same curvature function, and assign each two-combination of the curves the distance between their equivalence classes. We write $[P]$ for the equivalence class of the curves with the curvature function κ_P under the chosen curvature parameters, and we will use δ to denote both, the distance measure for the curvature functions and the induced measure on the equivalence classes of polygonal curves.

4.1. The Distance Measure. An intuitive approach to a distance measure for polygonal curves to map large sets of points along one curve onto a set of points on the other curve with the objective of minimizing some property under specific constraints on the possible mappings. Instead of a taking discrete points on the curve one can also work with continuous parametrizations, and take the integral over the whole domain. In our case we want to minimize the difference of the curvature values. We would want something like,

$$\delta([P], [Q]) = \min_{\phi} \int_D |\kappa_P(\phi(t)) - \kappa_Q(t)| dt,$$

where $\phi : D \rightarrow U$ such that D is the domain of κ_Q and U is the domain of κ_P . But we do not want to take parametrizations into account other than arc length parametrizations, because we would lose the integrity of the curvature function. Since the curvature is unique and for every curvature function there exists a curve, that has this curvature function, a reparametrization amounts to changing the shape of the original curve, at least in the case of smooth curves.

So which domain will we integrate over? We could consider the intersection of the domains, but that is out of question since we may lose important information about the original curves. We could scale one of the curves such that the curves have equal length. But this way we would allow only one mapping. Instead we propose a more radical solution that will allow us to extend the measure in a favorable way. On the subset of the domain, where κ is undefined, we map the subdomains to zero,

$$\tilde{\mathcal{A}}_\kappa = \mathcal{A}_\kappa \cup \{(-\infty, 0), (k_0 - k, 0), (k_n + k, 0), (-\infty, 0)\},$$

where k is the difference between two consecutive keys in the set of assignments. With respect to the original curve this amounts to lengthening each curve-ending to the infinite. Since the curvature of this extension is zero everywhere it should be a line in the direction, in which the curve ending was pointing. This modification is therefore coherent with the modification we took in Section 3.2 in connection with the parameter ϵ .

Now we can take different scales of the polygonal curve into account. From Section 2.3 we know that we can express the curvature of the scaled curve $T(\alpha)$ using the curvature of the original curve α through

$$\kappa_{T(\alpha)}(t) = s\kappa_\alpha(st).$$

where s is the inverse of the scaling factor. We want to introduce s as a parameter into our distance measure such that we minimize over all possible scaling factors. We also want the curvature functions be able to shift against each other, so similar parts of the curve can be traversed with the same parameter and therefore be matched to each other. For reflection see Section 4.3.

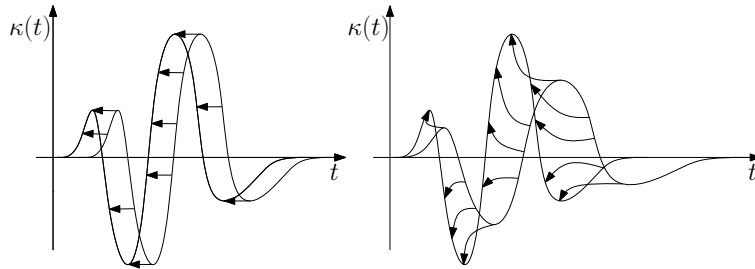


FIGURE 20. Arrows indicate the transformations that would minimize the integral of the difference of the two functions.

The proposed distance measure is the following,

$$(17) \quad \delta([P], [Q]) = \min_{h, s \in \mathbb{R}, s > 0} \int_{-\infty}^{\infty} |s\kappa_P(st - h) - \kappa_Q(t)| dt$$

The parameter h allows one of the curvature functions to be shifted along the x-axis to minimize the integral of the difference. The parameter s allows P to be scaled uniformly by any positive value. We declare curvature functions that differ only through the scale of their original curves and curves that are shifted horizontally to be equal. We call the objective function $d_{P,Q}(h, s) = \int_{-\infty}^{\infty} |s\kappa_P(st - h) - \kappa_Q(t)| dt$ or $d(h, s)$.

4.2. Metric Property. We have already indicated that a metric is a distance measure for which the triangle inequality holds. More specifically, a function $d : X \times X \rightarrow \mathbb{R}$ is called a *metric* on X if it has the following properties,

- (1) $d(x, y) = 0 \iff x = y$ (*identity*)
- (2) $d(x, y) = d(y, x)$ (*symmetry*)

(3) $d(x, y) + d(y, z) \geq d(x, z)$ (*triangle inequality*).

Note that from the above properties it follows that d is non-negative everywhere. We claim that (17) is a metric.

Proof. Clearly those and only those curves have a measure zero which of them we declared to be equal. To show that it is symmetric we can use the rule of substitution for integrals as laid out below. Set $u = st - h$, and substitute $\frac{u+h}{s}$ for t and $\frac{1}{s}du$ for dt . Since $s > 0$ we can pull the factor into the absolute value.

$$\begin{aligned} d_{P,Q}(h, s) &= \int_{-\infty}^{\infty} |s\kappa_P(st - h) - \kappa_Q(t)| dt \\ &= \int_{-\infty}^{\infty} \left| s\kappa_P\left(\frac{u+h}{s}\right) - \kappa_Q\left(\frac{u+h}{s}\right) \right| \frac{1}{s} du \\ &= \int_{-\infty}^{\infty} \left| \kappa_P(u) - \frac{1}{s}\kappa_Q\left(\frac{u+h}{s}\right) \right| du \\ &= d_{P,Q}(h', s'), \quad h' = -\frac{h}{s}, s' = \frac{1}{s} \end{aligned}$$

Since h' and s' are elements of the respective sets of parameters we minimize over, (17) is clearly symmetric.

For the triangle inequality, let a, b, c , and d be the arg-mins of the respective terms.

$$\begin{aligned} \delta([P], [Q]) + \delta([Q], [S]) &= d_{P,Q}(a, b) + d_{Q,S}(c, d) \\ &= \int_{-\infty}^{\infty} |a\kappa_P(at - b) - \kappa_Q(t)| dt + \int_{-\infty}^{\infty} |c\kappa_Q(ct - d) - \kappa_S(t)| dt \end{aligned}$$

We can rewrite the terms under the first integral using the substitution rule with $u = \frac{t+d}{c}$. Substitute $cu - d$ for t and cdu for dt in the first integral. Since c is positive we can pull it into the absolute value.

$$= \int_{-\infty}^{\infty} |ac\kappa_P(a(cu - d) - b) - c\kappa_Q(cu - d)| du + \int_{-\infty}^{\infty} |c\kappa_Q(ct - d) - \kappa_S(t)| dt$$

Now we can join the two integrals into one and apply the triangle inequality of the real euclidian distance for every t .

$$|x - y| + |y - z| \geq |x - z|, \quad x, y, z \in \mathbb{R}$$

$$x = ac\kappa_P(a(ct - d) - b)$$

$$y = c\kappa_Q(ct - d)$$

$$z = \kappa_S(t)$$

We retrieve

$$\begin{aligned} \delta([P], [Q]) + \delta([Q], [S]) &\geq \int_{-\infty}^{\infty} |ac\kappa_P(a(ct - d) - b) - \kappa_S(t)| dt \\ &= \int_{-\infty}^{\infty} \left| \underbrace{ac}_{s'} \kappa_P\left(\underbrace{ac}_{s'} t - \underbrace{(cd + b)}_{h'}\right) - \kappa_S(t) \right| dt \\ &= d_{P,S}(h', s'), \quad h' = cd + b, s' = ac \\ &\geq \delta([P], [S]), \end{aligned}$$

since h' and s' are elements of the respective parameter sets that we minimize over, the triangle inequality holds. Therefore δ is a metric. \square

4.3. Reflection and Change of Orientation. Since we want to compare the appearance of polygonal curves, we actually want to compare the traces of the curves. Therefore we also have to take a change of orientation into account. A change of orientation takes place when we reverse the order of the vertices. We also want the distance measure to be invariant under reflections of the polygonal curve.

We learned in Section 2.3) that under both transformations the curvature only changes the sign, aside from reversing the parameter. We can easily incorporate these invariances as follows, let the actual distance measure be

$$(18) \quad \tilde{\delta}([P], [Q]) = \min \{ \delta(\tilde{\kappa}_P, \kappa_Q) \mid \tilde{\kappa}_P \in \{ \kappa_P, -\kappa_P, -(\kappa_P \circ \phi), \kappa_P \circ \phi \} \},$$

where ϕ reverses the parameter of κ_P , in our case we can write $\phi(t) = -t$, since the domain of the curvature function is infinite. It turns out that those transformations are in fact very similar, since ϕ carries out a reflection in the parameter space, see Figure 21.

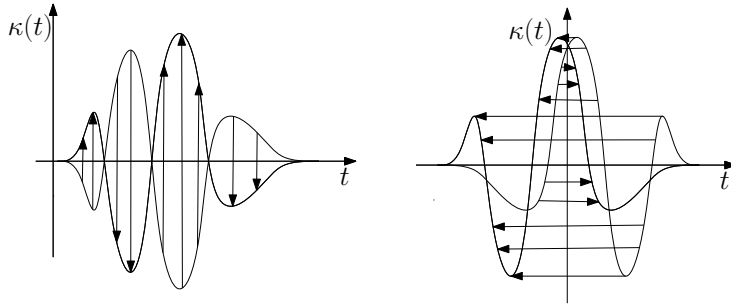


FIGURE 21. Reflection and reflection in the parameter space.

The equivalence classes of polygonal curves that the induced measure compares, are now those of the polygonal curves with the same trace, and which differ only by a combination of reflection, translation, rotation and uniform scaling and which yield the same or a shifted curvature function under a certain choice of parameters. In practice we will only compare curvature functions that were computed with the same method and the same set of parameters.

We claim that $\tilde{\delta}$ is a metric.

Proof. By definition $\tilde{\delta}$ is zero iff the curves are in the same equivalence class.

For symmetry and the triangle inequality consider the following. For a, b curvature functions it holds that,

$$(19) \quad \delta(-a, b) = \delta(a, -b)$$

$$(20) \quad \delta(a \circ \phi, b) = \delta(a, b \circ \phi)$$

Both properties follow from the definition of δ as the integral over the absolute difference of the functions. The absolute value of the difference does not change for arbitrary functions a and b , $|a - b| = |b - a|$, and the integral does not change if we reverse the integration parameter. Therefore it is immediate that $\tilde{\delta}$ is still symmetric.

Let $\tilde{\kappa}_P \in \{ \kappa_P, -\kappa_P, -(\kappa_P \circ \phi), \kappa_P \circ \phi \}$, such that $\delta(\tilde{\kappa}_P, \kappa_Q)$ is minimal. And let $\tilde{\kappa}_Q$ be defined analogously. Then we can write

$$\tilde{\delta}([P], [Q]) + \tilde{\delta}([Q], [S]) = \delta(\tilde{\kappa}_P, \kappa_Q) + \delta(\tilde{\kappa}_Q, \kappa_S).$$

Because of (19) and (20) there exists a $\tilde{\kappa}_S \in \{ \kappa_S, -\kappa_S, -(\kappa_S \circ \phi), \kappa_S \circ \phi \}$, such that $\delta(\tilde{\kappa}_Q, \kappa_S) = \delta(\kappa_Q, \tilde{\kappa}_S)$. The triangle inequality holds for $\tilde{\kappa}_P, \kappa_Q$ and $\tilde{\kappa}_S$ with

respect to δ ,

$$\delta(\tilde{\kappa}_P, \kappa_Q) + \delta(\kappa_Q, \tilde{\kappa}_S) \geq \delta(\tilde{\kappa}_P, \tilde{\kappa}_S)$$

Again, because of (19) and (20) there exists a $\hat{\kappa}_P \in T = \{\tilde{\kappa}_P, -\tilde{\kappa}_P, -(\tilde{\kappa}_P \circ \phi), \tilde{\kappa}_P \circ \phi\}$, such that $\delta(\tilde{\kappa}_P, \tilde{\kappa}_S) = \delta(\hat{\kappa}_P, \kappa_S)$. Without loss of generality we assume that in the beginning the curvature functions were centered in the parameter space, i.e. the parameter $t \in [-\frac{l}{2}, \frac{l}{2}]$, where l is the length of the curve. This way the function $\phi(t) = -t$ is equal to its inverse and therefore it holds for a curvature function a that $a = a \circ \phi \circ \phi$, as well as $a = -(-a)$ in any case. It is also true that the two transformations are commutative, $(-a) \circ \phi = -(a \circ \phi)$. It follows that T is equal to the set $\{\kappa_P, -\kappa_P, -(\kappa_P \circ \phi), \kappa_P \circ \phi\}$. Therefore

$$\delta(\tilde{\kappa}_P, \tilde{\kappa}_S) = \delta(\hat{\kappa}_P, \kappa_S) \geq \min\{\delta(\hat{\kappa}_P, \kappa_S) \mid \hat{\kappa}_P \in T\} = \tilde{\delta}([P], [S]).$$

Therefore $\tilde{\delta}$ is a metric. \square

4.4. Normalization. A normalized distance measure has the image set $[0, 1]$. The output of a normalized distance measure can be more meaningful, since it is not only a value that sets the similarity relations of the objects in relation, but it is often a percentage of how dissimilar the objects are. This is important if one not only wants to measure how dissimilar the objects are, but rather how similar. A measure that gives the similarity of the objects is simply $1 - \delta([P], [Q])$.

In Section 2.3 we have discussed, that the integral of the curvature function is invariant under scaling. Clearly the integral is also invariant under a shift of parameter, reflection and change of orientation. Therefore we could normalize the distance measure with the sum of the two integrals of the curvature functions $\int_{-\infty}^{\infty} \kappa_P + \int_{-\infty}^{\infty} \kappa_Q$, since this is the maximal distance that the two curves can have under the proposed distance measure. The bound is attained for the distance of any curvature function and the curvature of a straight line segment, therefore this is an exact bound.

As it turns out the distance measure modified this way is not a metric anymore. We can give a counter-example of a set of curves where the triangle inequality does not hold. Consider the functions a, b, c in Figure 22. We can show that for any choice of h and s , the normalized integral of the difference between $sa \circ \phi$ and b is larger than the sum of the normalized distances between a and c , and between b and c respectively. And therefore the triangle inequality cannot hold for the normalized distance measure.

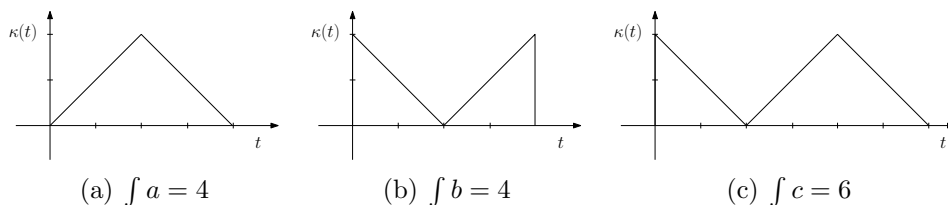


FIGURE 22. A counter example such that the triangle inequality does not hold for the considered normalized distance measure.

We claim that

$$\forall h, s : D = \frac{d(h, s)}{\int a + \int b} > \frac{2}{5} \geq \frac{\tilde{\delta}(a, c)}{\int a + \int c} + \frac{\tilde{\delta}(b, c)}{\int b + \int c},$$

where d is the objective function in (18).

Proof. We can easily find parameters, to prove the second inequality

$$\frac{\tilde{\delta}(a, c)}{\int a + \int c} \leq \frac{d(-2, 1)}{10} = \frac{1}{5}, \quad \text{and} \quad \frac{\tilde{\delta}(b, c)}{\int b + \int c} \leq \frac{d(0, 1)}{10} = \frac{1}{5}.$$

The difficulty lies in proving the first inequality. We define C to be the area of intersection of the areas under the graphs of $sa \circ \phi$ and b . Using the inclusion-exclusion principle we can write

$$D = \frac{\int a + \int b - 2C}{\int a + \int b} = 1 - \frac{C}{4},$$

since the integral of $sa \circ \phi$ is invariant under the choice of h and s . A choice of h and s such that D is minimal, maximizes C .

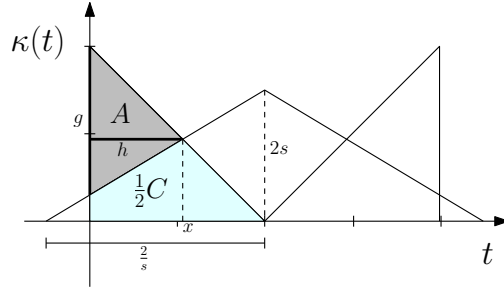


FIGURE 23

We conjecture that a configuration such that C is maximal, covers the triangles of b in equal parts see Figure 24, and we will reinforce this conjecture in an additional analysis in (5.5) where we also demonstrate the decomposition of the objective function which is yet to be derived for the analysis. For now want to determine the minimal value of D in this setting.

Consider the area A of the triangle in Figure 23. We know $\frac{1}{2}C + A = 2$, since it is the triangle from function b and therefore $D = \frac{A}{2}$. We can find the length of the vertical side of the triangle g and the height h with respect to s and substitute into $A = \frac{gh}{2}$. It is $g = 2s^2 - 2s + s$, because of the intercept theorem and $h = x$, where x is the function value of the intersection of the two functions $b_1(x) = -x + 2$ and $a_1(x) = s^2x + 2s - 2s^2$, which is $x = \frac{2s^2 - 2s + 2}{s^2 + 1}$. Plugging those values into the formula for D , we retrieve

$$D \geq \min_s \frac{(s^2 - s + 1)^2}{s^2 + 1} > 0.418587820 > \frac{2}{5}.$$

□

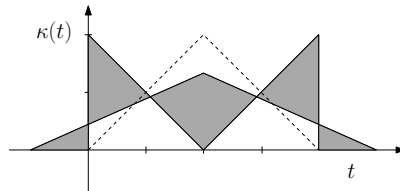


FIGURE 24. Optimal configuration with respect to maximizing the covered area, and such that both triangles are covered in equal parts, $h = 2s - 2$, $s = 0.6823278040$, $D = 0.418587820$

5. ANALYSIS OF THE OBJECTIVE FUNCTION

The distance measure (17), as well as its extended version (18), pose an optimization problem of which we denote the objective function,

$$(21) \quad d_{P,Q}(h, s) = \int_{-\infty}^{\infty} |s\kappa_P(st - h) - \kappa_Q(t)| dt.$$

The global minimum of this function is the distance between P and Q . Naturally $d_{P,Q}$ can be approximated brute force, by sampling values on a grid within reasonable ranges for s and h . The complexity lies in $O(k^2(m + n))$, where k is the number of grid points in one dimension, and m and n are the number of assignments of the curvature functions κ_P and κ_Q . But this does not grant a solution anywhere near the optimum, since the first derivative of the objective function can be arbitrarily high, and therefore the function can get arbitrarily small in a range between grid points. Even if we had a bound for the maximum of the first derivative, for instance with respect to the maximum derivative of the input functions. Then we could refine the grid, but for a large grid size the computation time is unacceptable.

Therefore we need to understand more about the objective function. We will see that the function is continuous everywhere and differentiable almost everywhere and therefore the chapter after this chapter deals with the approximation of the minimum of $d_{P,Q}$ using the stochastic version of the method of the steepest descent.

Since κ_P and κ_Q are defined piecewise, it is unlikely that we can find an exact analytical expression that holds for the whole domain. We will however find for almost every point $p = (h, s)$ in the parameter space an analytical expression that holds locally on a large enough open set such that it is differentiable at p . And we will show that the set of points that are excluded has measure zero. We will in fact analyze what this set looks like in Section 5.4. We will also examine the function bounded on two exemplary subspaces, of which the simpler yields a piecewise quadratic function for $d_{P,Q}|_U$, U being the subspace. In general the derived function is a piecewise function, where the pieces are of finite number and are fractions of polynomials.

5.1. Continuity. We can show that, for functions $f : \mathbb{R} \rightarrow \mathbb{R}$, $g : \mathbb{R} \rightarrow \mathbb{R}$,

$$d(p) = \int_{-\infty}^{\infty} |sf(st - h) - g(t)| dt, \quad p = (h, s) \in \mathbb{R}^2$$

is continuous if f and g are zero outside an interval.

Proof. We have to show that for every $p = (h_1, s_1) \in \mathbb{R}^2$ and for every $\epsilon > 0$ there exists a $\delta > 0$ such that

$$\|p - q\| < \delta \Rightarrow |d(p) - d(q)| \leq \epsilon, \quad q = (h_2, s_2) \in \mathbb{R}^2.$$

We have

$$\begin{aligned} |d(p) - d(q)| &= \left| \int_{-\infty}^{\infty} |s_1 f(s_1 t - h_1) - g(t)| dt - \int_{-\infty}^{\infty} |s_2 f(s_2 t - h_2) - g(t)| dt \right| \\ &= \left| \int_{-\infty}^{\infty} |s_1 f(s_1 t - h_1) - g(t)| - |s_2 f(s_2 t - h_2) - g(t)| dt \right| \\ &\stackrel{(*)}{\leq} \int_{-\infty}^{\infty} |s_1 f(s_1 t - h_1) - s_2 f(s_2 t - h_2)| dt = \int F(t) dt \end{aligned}$$

where $(*)$ follows from the triangle inequality. We required that f is zero outside an interval, let this interval be $[t_{\min}, t_{\max}]$. It follows that this also holds for the

integrand F in the equation above. Let $\phi_i(t) = s_i t - h_i, i \in \{1, 2\}$. We can say that $F(t) = 0$, for either

$$\begin{aligned} t &< \min(\phi_1^{-1}(t_{\min}), \phi_2^{-1}(t_{\min})) = a \\ t &> \max(\phi_1^{-1}(t_{\max}), \phi_2^{-1}(t_{\max})) = b. \end{aligned}$$

Therefore

$$\int F(t) dt = \int_a^b F(t) dt$$

Let $\psi(h, s) = st - h$ be a function defined for some fixed t . The function $sf \circ \psi$ is clearly continuous, therefore there exists a δ' , such that for $\|p - q\| < \delta'$

$$F(t) = |(sf \circ \psi)(p) - (sf \circ \psi)(q)| < \epsilon' = \frac{\epsilon}{b - a}.$$

We choose δ to be the minimum of the δ' over all t . Then

$$|d(p) - d(q)| \leq \int_a^b \frac{\epsilon}{b - a} dt = \epsilon.$$

□

5.2. Decomposition. We can decompose the integral into a finite sum if we know the intersections of the two curves, because of the fact that the integral of the difference over an interval $[a, b]$ is equal to the difference of the integrals, if the functions bounded to $[a, b]$ have no intersections. In other words,

$$(22) \quad \int_a^b |f(t) - g(t)| dt = \left| \int_a^b f(t) dt - \int_a^b g(t) dt \right| \\ = \left| [F(t)]_a^b - [G(t)]_a^b \right|$$

since for $[a, b]$ we have

$$\forall t \in [a, b] : f(t) > g(t) \vee \forall t \in [a, b] : f(t) < g(t).$$

Because of this it is also easy to eliminate the absolute value. We only need to evaluate $f(t) > g(t)$ at an arbitrary point $t \in (a, b)$ to find out which of the two terms $[F(t)]_a^b$ or $[G(t)]_a^b$ will be greater. In our case we can also easily compare the slopes at one of the intersection points. Furthermore, we do not have to take into account whether f or g intersect with the zero axis, clearly (22) also holds if f or g go below zero within the interval $[a, b]$.

Now we can decompose the integral according to the intervals between intersections, given that we have an expression of those with respect to h and s . Let ϕ be a function such that $(sf \circ \phi)(t) = sf(st - h)$. We call a function $\omega : U \subset \mathbb{R} \times \mathbb{R}^+ \setminus \{0\} \rightarrow \mathbb{R}$ an *intersection function* of linear functions $f|_A$ and $g|_B$, where A and B are intervals, if

$$\forall (h, s) \in U : sf(s\omega(h, s) - h) = g(\omega(h, s)),$$

and if its domain U is connected and locally maximal. The maximality condition is supposed to reduce the number of possible intersection functions. We require there is no other intersection function ω' for the same linear functions, such that U is a proper subset of the domain of ω' , which is also a connected set. The condition that U should be connected will increase the number of intersection functions, but we will see that their number is constant, defined this way.

According to this definition then $(\omega(h, s), g(\omega(h, s)))$ is an intersection point of g and $sf \circ \phi$. An expression for $\omega(h, s)$ is easily derived. Let

$$\begin{aligned} f : A \rightarrow \mathbb{R}, \quad f(t) &= at + b \\ g : B \rightarrow \mathbb{R}, \quad g(t) &= ct + d. \end{aligned}$$

We need to solve the following statement for t .

$$\begin{aligned} sf(st - h) &= g(t), \quad h, s \in \mathbb{R}, s > 0 \\ s(a(st - h) + b) &= ct + d \\ t(as^2 - c) &= ash - bs + d \\ (23) \quad t &= \frac{ash - bs + d}{as^2 - c} = \omega(h, s) \end{aligned}$$

Special cases occur where the intersection of two linear functions is equal to the whole domain of the functions, when $as^2 - c = 0$. We will see that, in order for the decomposition to work, we only need to define an intersection function that returns an arbitrary element of the set of intersections in this case.

To picture the meaning of the set of intersection functions for two piecewise linear functions, consider $d_{P,Q}(h, s)|_V$ bounded on the specific one dimensional subspace $V = \{(h, 1) | h \in \mathbb{R}\}$. For this case the domain of an intersection function is an interval. Determining the intersections for a specific choice of h reduces to a query for h in a set of intervals to retrieve the intersection functions, the domain of whose contains h . The functions evaluated at h yield the parameters t_i , such that $(t_i, g(t_i))$ is an intersection point. In the general case we will see that if two edges do intersect for any choice of parameters h and s , then their intersection function is defined in a neighbourhood around (h, s) , iff the intersection point for (h, s) is not an endpoint of the edges. We will analyze what the domain U exactly looks like in Section 5.4.

But it is clear that, given our intersection functions are well defined, the domains of the intersection functions of two piecewise linear functions subdivide the parameter space into a finite number of subdomains, every subdomain being the intersection of the domains it covers, such that in each cell of the partition the intersection functions are preserved.

Assume the intersection functions are well defined, then we can use them to simplify the objective function. Let $\omega_0(h, s) < \omega_1(h, s) < \dots < \omega_n(h, s)$ be the intersections for some h, s . The function values subdivide the domain of the integral, such that in each interval the functions have no intersections, see Figure 25.

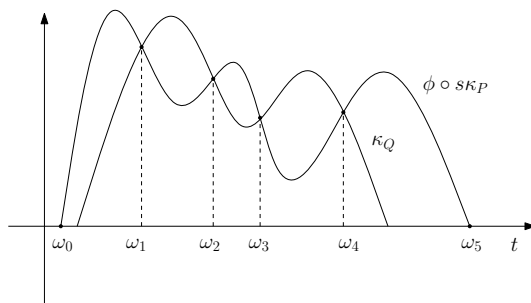


FIGURE 25. $\omega_0(h, s) < \omega_1(h, s) < \dots < \omega_n(h, s)$ subdivide the domain of the integral.

Let U be the maximal subdomain around (h, s) where the intersection functions $\omega_0, \dots, \omega_n$ are preserved. Since we required the domain of an intersection function

to be connected, this is the intersection of their domains

$$U = \bigcap_{i=0}^{i=n} \text{Domain}(\omega_i)$$

Using (22) we can write the integral in $d_{P,Q}(h, s)$ as a finite sum.

$$\begin{aligned} d_{P,Q}(h, s)|_U &= \int_{-\infty}^{\infty} |s\kappa_P(st - h) - \kappa_Q(t)| dt \\ &= \sum_{i=1}^n \int_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} |s\kappa_P(st - h) - \kappa_Q(t)| dt \end{aligned}$$

Now we can further simplify the right hand side

$$\begin{aligned} \text{rhs} &= \sum_{i=1}^n \left| \int_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} s\kappa_P(st - h) dt - \int_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} \kappa_Q(t) \right| \\ &= \sum_{i=1}^n \left| \int_{s\omega_{i-1}(h,s)}^{s\omega_i(h,s)} \kappa_P(u - h) du - \int_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} \kappa_Q(t) \right|, \quad u = st, du = sdt \\ &= \sum_{i=1}^n \left| [\mathcal{K}_P(t)]_{s\omega_{i-1}(h,s)-h}^{s\omega_i(h,s)-h} - [\mathcal{K}_Q(t)]_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} \right| \end{aligned}$$

where \mathcal{K}_P and \mathcal{K}_Q denote the antiderivatives of κ_P and κ_Q .

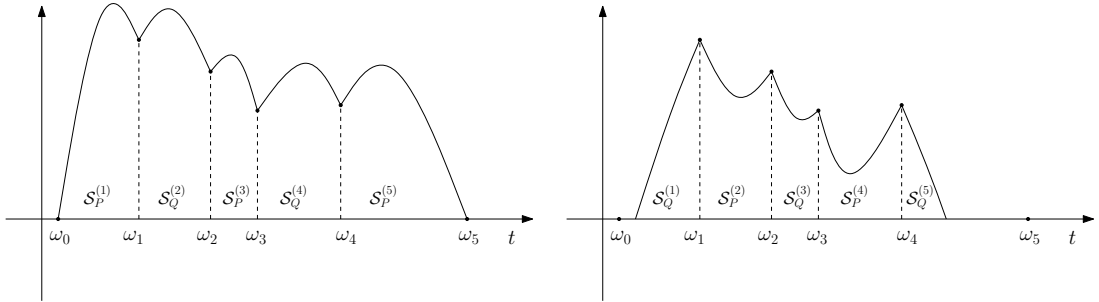


FIGURE 26. In reference to Figure 25. For each interval we add the integral of the function that is larger in this interval and subtract the integral over the function which is smaller in this interval.

We write $\mathcal{S}_P^{(i)}(h, s) = [\mathcal{K}_P(t)]_{s\omega_{i-1}(h,s)-h}^{s\omega_i(h,s)-h}$ and $\mathcal{S}_Q^{(i)}(h, s) = [\mathcal{K}_Q(t)]_{\omega_{i-1}(h,s)}^{\omega_i(h,s)}$ and re-group the terms into two sets of functions \mathcal{S}_+ and \mathcal{S}_- such that

$$\mathcal{S}_+ = \bigcup_{i=1}^n \left\{ \max \left(\mathcal{S}_P^{(i)}, \mathcal{S}_Q^{(i)} \right) \right\}, \quad \mathcal{S}_- = \bigcup_{i=1}^n \left\{ \min \left(\mathcal{S}_P^{(i)}, \mathcal{S}_Q^{(i)} \right) \right\},$$

where the functions min and max act according to

$$\mathcal{S}_P^{(i)} \leq \mathcal{S}_Q^{(i)} \Leftrightarrow \forall (h, s) \in U : \mathcal{S}_P^{(i)}(h, s) \leq \mathcal{S}_Q^{(i)}(h, s).$$

See also Figure 26. And therefore we can write

$$(24) \quad d_{P,Q}(h, s)|_U = \left(\sum_{f \in \mathcal{S}_+} f - \sum_{f \in \mathcal{S}_-} f \right) (h, s)$$

Note that the absolute value has been eliminated. We claim that the nature of the function $d_{P,Q}|_U$ is dominated by the nature of the intersection functions and that it is not a piecewise function.

We can see this with the following considerations. Recall that U is the intersection of the domains of intersection functions it covers. For each of the intersection functions the image set $B = \omega_i(U)$ is a subdomain of κ_Q , such that $\kappa_Q|_B$ is a linear function, because this is how we have designed the intersection functions. Therefore $\mathcal{K}_Q|_B$ is a quadratic function. We can make the analogous case for κ_P . By definition $A = \phi(\omega_i(U))$ is a subdomain of κ_P , such that $\kappa_P|_A$ is a linear function. Therefore $\mathcal{K}_P|_A$ is a quadratic function. Thus $d_{P,Q}|_U$ is a finite sum of terms of the form $\mathcal{K} \circ \phi \circ \omega$, or $\mathcal{K} \circ \omega$, respectively, where ϕ is a linear function and \mathcal{K} is a quadratic function. In the general case the intersection function ω is a polynomial fraction. A quadratic function chained with a polynomial fraction yields a polynomial fraction, the same holds for a linear function chained with a polynomial fraction. Therefore our expression for $d_{P,Q}|_U$ is the finite sum of polynomial fractions, which is a polynomial fraction.

We have already shown that the objective function is continuous. But let us check if this is consistent with the current result. To find out whether a polynomial fraction has discontinuities we have to examine the denominator, which is $s^2a - c$ in our case. Therefore the function could have a discontinuity for each pair of edges at $s = \sqrt{\frac{a}{c}}$. For this specific choice of s the functions $sf \circ \phi$ and g have the same slope. Therefore they only intersect for exact one choice of h . For this choice the set of intersections can be an infinite set of points on the edges, but for the decomposition to work ω only needs to return one point from this set of intersections. We will encounter this case again in the course of this chapter.

In Section 5.4 we will examine the boundaries of the partition of the domain of the subdomains, where the intersection functions are invariant. We will find that they constitute a set of measure zero. And therefore we have found an expression for $d_{P,Q}$ as a piecewise rational function, that consists of polynomial fractions.

A note on the usability of what we have derived so far. For the purpose of computing the complete objective function on a subspace one can compute the whole set of intersection functions between two piecewise linear functions, as it was done in the implementation. But computing the whole set in order to find the minimum of the objective function would not yield an improvement to the brute force algorithm in terms of the complexity since the size of the set of intersection functions is quadratic in the number of assignments. For a worst-case example consider Figure 27. It is not more, since the number of intersection functions for two edges is constant, as we will see later, and the number of two-combinations is quadratic.

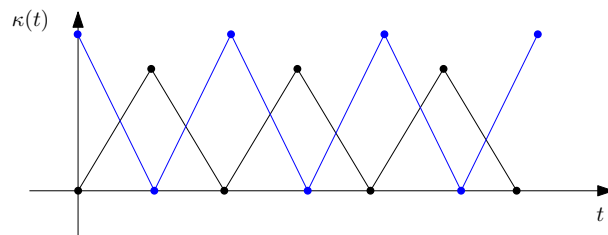


FIGURE 27. Worst case example for the number of intersection functions.

5.3. Case Studies in 1D. In the case studies we computed the entire function bounded on specific one dimensional subspaces of the parameter space. In order to do this we had to compute the entire set of intersection functions for the curvature functions and their domains. Then we computed the subdivision of the domain into intervals such that in each interval the intersection functions are preserved. We computed for each interval $I \subset \mathbb{R}$ the representation of $d_{P,Q}|_I$ as the function in (24).

The implementation included a library for the visualization and the handling of functions and curves, an overview of the Java classes can be found in the appendix. Plots of the resulting objective functions of an example set of curves are attached at the end of each section.

The analysis of the domain of the intersection function and the handling of special cases were the most involved part of work in this. The following two subsections give a detailed account on how it can be done.

5.3.1. Shifting Subspace. We want to investigate the role of the parameter h in the objective function. For this we consider the function bounded on the subspace of the domain where $s = 1$. We call $\hat{\omega} : I \subset \mathbb{R} \rightarrow \mathbb{R}$ an intersection function of the linear functions $f : A \rightarrow \mathbb{R}$ and $g : B \rightarrow \mathbb{R}$, if

$$\forall h \in I : f(\hat{\omega}(h) - h) = g(\hat{\omega}(h))$$

We can show that $\hat{\omega}$ is a linear function by substituting in (23)

$$\hat{\omega}(h) = \omega(h, 1) = \frac{ah - b + d}{a - c}.$$

If $f(A) \cap f(B) = \emptyset$ there are no intersections and therefore no intersection function can be found. There exist two general cases for an intersection function. In the first case the intersection function is increasing, the case when $\frac{a}{a-c} > 0$ in the second case it is decreasing, the case when $\frac{a}{a-c} < 0$. In one case the slope of f is greater than the slope of g and in the other case the slope of g is greater. See Figure 28.

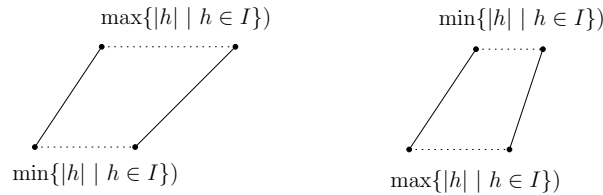


FIGURE 28. Two general cases for the intersection function if $s = 1$.

If the slopes are equal, i.e. if $a = c$, the function $\hat{\omega}$ is not well-defined. Clearly the function, as we defined it would have to return the whole range of intersection. But in order for the function $d_{P,Q}(h, 1)$ to return the integral of the difference it suffices if $\hat{\omega}$ returns at least one arbitrary value from the intersection range.

For the case that $a = c$ and $a \neq 0$ this is easy, because $f(\hat{\omega}(h) - h)$ and g only intersect for $h = -\frac{d-b}{a}$. We define $\hat{\omega}$ in this case to return the minimal value from the intersection range, $\hat{\omega}(h) = \min(\{t \mid t + \frac{d-b}{a} \in A\} \cap B)$.

If $a = c = 0$ the linear functions intersect for any value of h if $b = d$ or none otherwise. We need to compute the domain of $\hat{\omega}$ from the intervals A and B , that the functions are bounded on. We propose to define four different intersection

functions which return the minimal and maximal values of the intersection range.

$$\begin{aligned}\omega_1 &: [\min(B) - \max(A), \min(B) - \min(A)] \rightarrow \mathbb{R}, & \omega_1(h) &= \min(B) \\ \omega_2 &: [\max(B) - \max(A), \max(B) - \min(A)] \rightarrow \mathbb{R}, & \omega_2(h) &= \max(B) \\ \omega_3 &: [\min(B) - \min(A), \max(B) - \min(A)] \rightarrow \mathbb{R}, & \omega_3(h) &= -h + \min(A) \\ \omega_4 &: [\min(B) - \max(A), \max(B) - \max(A)] \rightarrow \mathbb{R}, & \omega_4(h) &= -h + \max(A)\end{aligned}$$

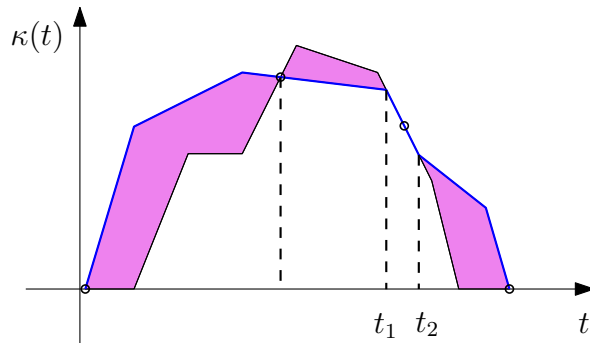


FIGURE 29. Special case that $a = c$. In order for the decomposition into intersection intervals to be well-defined, it suffices for the intersection function to return an arbitrary value from the range $[t_1, t_2]$. The same holds for the general special case that $s^2a = c$.

Remaining special cases occur when exactly one of the edges is horizontal. Although $\hat{\omega}$ is well defined in those cases, the inverses of f and g are no proper functions, therefore we need to derive the domain in some other way. For $a = 0$ we can use

$$\omega : [g^{-1}(b) - \min(A), g^{-1}(b) - \max(A)] \rightarrow \mathbb{R}, \quad \omega(h) = g^{-1}(b) = \frac{b-d}{c}.$$

For $c = 0$ we use

$$\omega : [\min(B) - f^{-1}(d), \max(B) - f^{-1}(d)] \rightarrow \mathbb{R}, \quad \omega(h) = h + \frac{d-b}{a}.$$

Note that we do not have to deal with vertical edges because f and g are functions.

In all the other cases the domain of ω is $\omega^{-1}(g^{-1}(f(A) \cap g(B)))$. This is a closed interval, since the intersection of two closed intervals is a closed interval and both ω and $g|_B$ are linear and not constant, therefore their inverses are linear as well.

We can conclude that the domains of the set of intersection functions for two piecewise linear functions is a set of intervals in the order $O(nm)$, where n and m are the number of assignments of the curvature functions.

We can see that the objective function bounded on the considered subspace is a piecewise quadratic function. The analysis of (24) also holds for the function bounded on the subspace. Therefore we have a finite sum of terms of the form $\mathcal{K} \circ \phi \circ \hat{\omega}$, or $\mathcal{K} \circ \hat{\omega}$, respectively, where \mathcal{K} is a quadratic function, and ω is a linear function and $\phi(t) = t - h$ is also a linear function.

The following Figures show an example of the objective function bounded to the investigated subspace and examples of how the curves are shifted relative to each other at a local extremum.

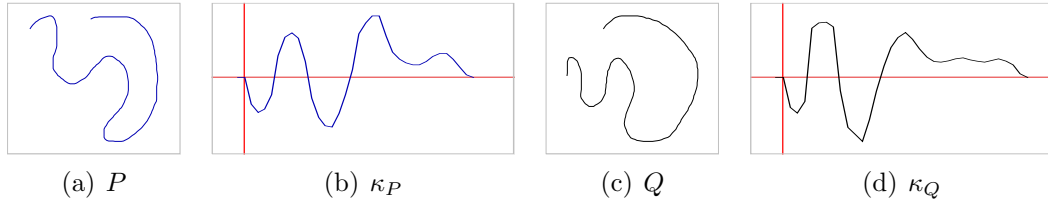


FIGURE 30. Example set of curves for the case studies.

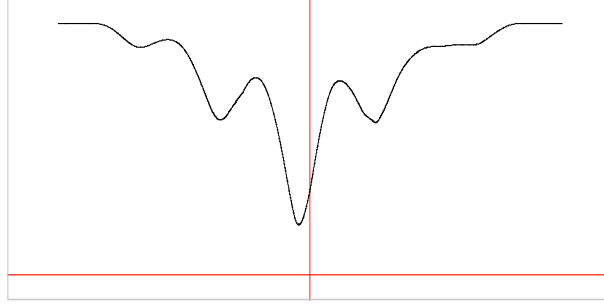


FIGURE 31. A representation of $d_{P,Q}(h, 1)$ as function of h . The y -coordinate of the global minimum of this function is the "shifting distance" between P and Q in Figure 30.

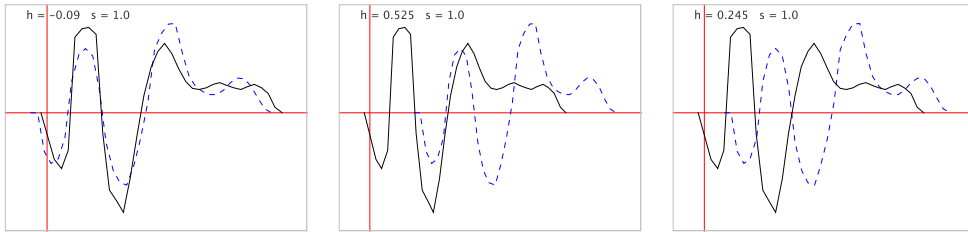


FIGURE 32. Global minimum and examples of a local minimum, and a local maximum.

5.3.2. *Scaling Subspace.* We want to investigate the role of the parameter s in the objective function. For this we consider the function bounded on the subspace of the domain where $h = 0$. An intersection function has to satisfy

$$\forall s \in [a, b] : sf(s\omega(s)) = g(\omega(s)).$$

Obtain a formula for ω by substituting into (23).

$$\omega(0, s) = \frac{-bs + d}{as^2 - c}$$

In general ω is not a monotonous function and therefore not invertible. To find the domain of ω we first introduce the notion of trajectories and trajectory ranges.

Consider the following problem. For each point on the graph of the function f , we want to find component functions x and y , such that

$$(25) \quad sf(sx(s)) = y(s).$$

We call the trace of the resulting curve $\tau(s) = (x(s), y(s))$ the *trajectory* of the point $(x(1), y(1))$. This point lies on the function graph of f , see Figure 33.

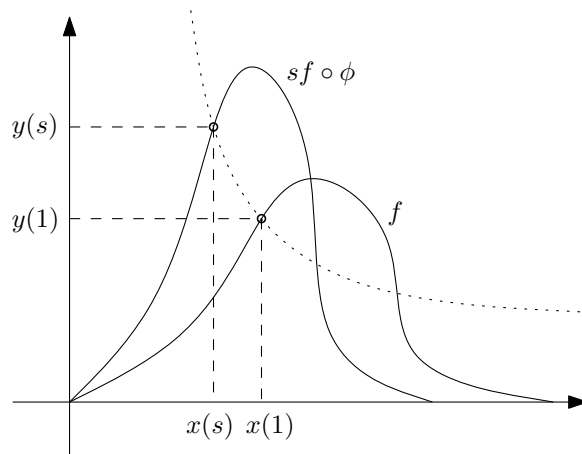


FIGURE 33. The functions f and $sf \circ \phi$, where $\phi(s) = st$.

Possible component functions that satisfy (25) are

$$x(s) = \frac{x_1}{s}, \quad y(s) = sy_1, \quad \text{for } f(x_1) = y_1.$$

The resulting curve is $\tau(s) = \left(\frac{x_1}{s}, sy_1\right)$. The trace of this curve is the trajectory of the point with coordinates $x(1) = x_1$ and $y(1) = y_1$. Since we are only interested in the trace of the curve we can arbitrarily reparametrize τ and even change the orientation.

$$(\tau \circ \alpha)(s) = \left(s, \frac{x_1 y_1}{s}\right), \quad \alpha(s) = \frac{x_1}{s}$$

$\tau \circ \alpha$ has the same trace as τ and, as a set of points, is equal to the graph of the function $\frac{x_1 y_1}{s}$. Thus the set of points on the graph of the function f can be pictured to traverse trajectories of the class of functions $f_a(t) = \frac{a}{t}$. And the trajectory of a single point is the function where $a = xy$. Note that the trajectory of a point on the graph of f is independent of f and that we can compute the trajectory parameter a from every point on the same trajectory.

We want to find the domains of the intersection functions for f and g . The two functions $sf \circ \phi$ and g intersect iff the trajectory of a point on the graph of f intersects with g . But we can just as well compute trajectories of points on the graph of g and intersect them with f .

We can think of f and g as line segments in \mathbb{R}^2 , as they are linear functions bounded on an interval. More specifically, let p_1, p_2 be the endpoints of the line segment of f . We can say $p_1 = (\min(A), f(\min(A)))$ and $p_2 = (\max(A), f(\max(A)))$, then the image set $f(A)$ is equal to the line segment $e = \overline{p_1 p_2}$. We call an interval $I = [a_1, a_2] \subset \mathbb{R}$ the *trajectory range* of e , if the graphs of the functions $\{f_a(t) = \frac{a}{t} \mid a \in I\}$ are the trajectories that the points on the edge traverse. To compute the trajectory range of (p_1, p_2) we need to solve for the maximum and the minimum of the quadratic function $f(t) = xy$, where $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ and

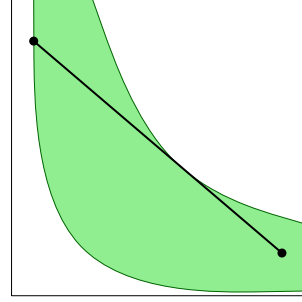
$$(x, y) = (1 - t)p_1 + tp_2, \quad t \in [0, 1]$$

Multiplying the x and y -component, we retrieve

$$\begin{aligned} f(t) &= ((1-t)x_1 + tx_2)((1-t)y_1 + ty_2) \\ &= ut^2 + vt + w \end{aligned}$$

where

$$\begin{aligned} u &= x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 \\ v &= x_1y_2 + x_2y_1 - 2x_2y_2 \\ w &= x_1y_2 + x_2y_1 + x_2y_2. \end{aligned}$$



In the above figure you can see an example of the case that the resulting trajectory range of an edge is not simply equal to the trajectory parameters of the endpoints.

Now we can intersect the trajectory ranges of f and g to check if there are any intersection functions at all. If there are none, we are done. If the intersection is non-empty we have to find ranges of intersection with respect to s .

Our starting point of reasoning is that all intersection points lie on g and that we can model ω with two functions bounded on disjoint domains, each of them invertible. A function $\frac{a}{t}$ can intersect with a linear function g at two, one, or zero points. Therefore we have for each trajectory four values, where each of them could possibly be a boundary value of the domain of an intersection function.

Since there are so many possible cases we will collect those possible boundary values, sort them and test for each interval if the image under ω lies in the respective domains of g and $sf \circ \phi$. We assume

$$\omega([a, b]) \subset B \cap \phi^{-1}(A) \Leftrightarrow \exists s \in [a, b], \omega(s) \in B \cap \phi^{-1}(A),$$

since a and b are consecutive elements in the sorted list of possible boundary values.

We have the following algorithm to compute the connected domains of ω in the general case. We will use the inverse functions of omega

$$\omega_i^{-1}(t) = \pm \sqrt{\frac{4cat^2 + 4dat + b^2}{4a^2t^2}} + \frac{-b}{2at}, \quad i \in \{1, 2\}.$$

```

1  Vector<Double> trajectories = new Vector<Double>();
2
3
4  trajectories.add( A.min() * f.value(A.min()) );
5  trajectories.add( A.max() * f.value(A.max()) );
6  trajectories.addAll( computeTrajectoryRange(f) );
7  trajectories.addAll( computeTrajectoryRange(g) );
8
9  //project all possible x-values of boundary intersection points
10 Vector<Double> intersections = new Vector<Point>();
11 for(double a: trajectories){
12
13     //intersect trajectories with g
14     intersections.addAll(new Function(a/x).intersect(g));
15
16 }
17 //add remaining possible x-values of boundary intersection points
18 intersections.add(B.min())
19 intersections.add(B.max())
20
21 //compute possible inverse values
22 Vector<Double> boundaries = new Vector<Double>();
23 for(double t: intersections){
24
25     double s = omega_1^{-1}.value(t);
26     boundaries.add(s);
27

```

```

28     double s_ =  $\omega_2^{-1}$ .value(t);
29     boundaries.add(s_);
30 }
31 //add zero just in case
32 boundaries.add(0)
33
34 //sort collected possible boundary values
35 Collections.sort(boundaries);
36
37 //test each range separately
38 Vector<Range> result = new Vector<Range>();
39 for(int i = 1; i < boundaries.size(); i++){
40     double min = boundaries.get(i-1);
41     double max = boundaries.get(i);
42     if(min == max) continue;
43
44     //test for some s from the range
45     double s = (min+max)/2.;
46     if(B.contains( $\omega$ (s)) and A.contains( $\phi$ ( $\omega$ (s)))){
47         result.add(new Range(min,max));
48     }
49 }

```

Now we can analyze how many intersection function there will possibly be in the general case for two given functions f and g . The number of trajectories we are collecting in lines 4-7 is at most 6. With the operation in line 14 this number at most doubles and we add another two in lines 18 and 19. Therefore we have at most 14 possible values of boundary intersection intersection points on g . With the computation of the possible values of boundaries in lines 25 and 29 this number at most doubles again. Therefore we have at most 28 boundary values for the domain of ω .

We will see in Section 5.4 that there are actually at most two intersection functions for two edges. As it turns out, the analysis of the number of intersection functions is more complicated in the simplified case.

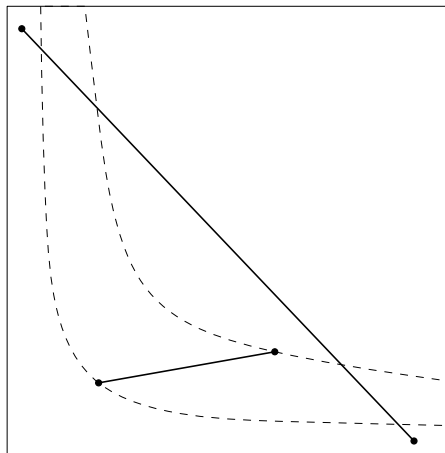


FIGURE 34. Case where two edges intersect for two disconnected intervals in the domain of s .

And we still have not said anything yet about the special cases, and what the special cases are. Let us investigate the cases in which the proposed method fails. Obviously it fails if the denominator of ω is zero for parameters in the intersection range or if the denominator in the inverse functions goes to zero, or if the expression underneath the root goes negative.

We restricted our analysis to the case that both curvature functions have non-zero values only for positive parameters. It is sufficient to handle the following cases, given that two edges intersect at all,

- (1) $a = 0, b = 0, c = 0, d = 0$, is analogous to the same case in the last section.
- (2) $b = 0, d = 0$, and either $a \neq 0$ or $c \neq 0$, the edges only intersect if both intersect with the origin, therefore they intersect for any value of s , $\omega : (0, \infty]$, $\omega(s) = 0$.
- (3) $a = 0, b = 0, c \neq 0, d \neq 0$, $\omega : [-\frac{c}{d} \max(A), -\frac{c}{d} \min(A)]$, $\omega(s) = -\frac{d}{c}$.
- (4) $a \neq 0, b \neq 0, c = 0, d = 0$, $\omega : [-\frac{b}{a \max(B)}, -\frac{b}{a \min(B)}]$, $\omega(s) = -\frac{b}{as}$, the boundaries of the range may be interchanged, depending on their actual values.
- (5) $\sqrt{\frac{a}{c}}b = d$, in this case the two edges have the same slope at the point of their intersection, therefore the set of intersection might be infinitely large. We can either pick $\omega(s) = \min(B)$ or $\omega(s) = \max(B)$, depending on the range of intersection. The supporting lines intersect for $s = \sqrt{\frac{c}{a}}$, this is the only value that is contained in the domain.

The following Figures show an example of the objective function bounded to the investigated subspace and examples of how the curves are scaled relative to each other at a local extremum.

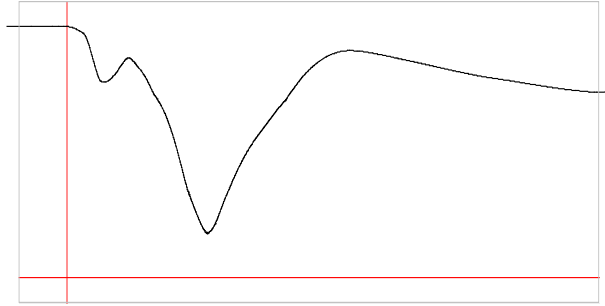


FIGURE 35. A representation of $d_{P,Q}(0, s)$ as a function of s . The y -coordinate of a global minimum of this function is the "scaling distance" between P and Q in Figure 30.

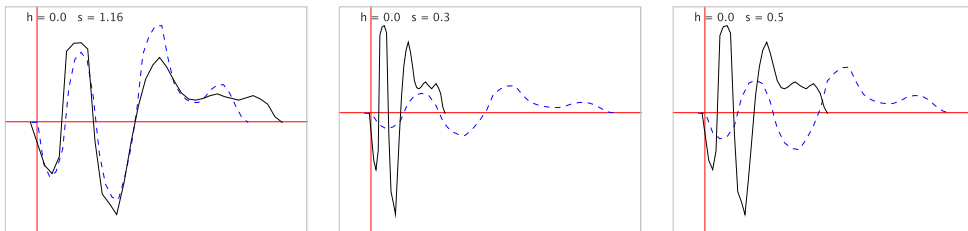


FIGURE 36. Global minimum and examples of a local minimum, and a local maximum (not to scale).

5.4. Singularities. We have found a piecewise expression of the objective functions that consists of polynomial fractions in (24). The pieces of this function can be constructed from the set of intersection functions. The boundaries of the domains of the intersection functions partition the domain in a way that for each cell U , the set of intersection functions, whose domain intersects with U , are preserved within U . In other words, for each (h, s) of the domain, the set of intersection functions defined for (h, s) , is invariant in the containing cell.

$$\text{Domain}(\omega) \cap U \neq \emptyset \Leftrightarrow U \subset \text{Domain}(\omega)$$

On the boundaries of the cells, the objective function is not differentiable. We have to assume this, since nothing forces the one sided derivative of expression in (24) on the boundary of the cell to be the same as the one sided derivative from the neighbouring cell. The partition can be obtained by intersecting the boundaries of the domains of each intersection function of all possible combinations of edges from the curvature functions. Therefore we will analyze the boundaries of the domains of the intersection functions for two edges.

Let the linear functions $f : A \rightarrow \mathbb{R}$ and $g : B \rightarrow \mathbb{R}$ be pieces of our curvature functions as in the last sections. We call them edges, since their image sets $f(A)$ and $g(B)$ are line segments and together with the other edges, defined this way, of the curvature function they constitute a polygonal curve, such that the set of points on this curve is equal to the set of points on the function graph.

For each point on the graph of the function f , we want to find component functions x and y , such that

$$sf(sx(h, s) - h) = y(h, s).$$

Possible component functions that satisfy the above condition are

$$x(h, s) = \frac{x_{01} + h}{s}, \quad y(h, s) = sy_{01},$$

for any fixed x_{01}, y_{01} , such that $f(x_{01}) = y_{01}$. Then we have $x(0, 1) = x_{01}$ and $y(0, 1) = y_{01}$. The resulting function

$$\Psi_p(h, s) = (x(h, s), y(h, s)), \quad \text{where } p = (x_{01}, y_{01})$$

can also be written as a matrix multiplication

$$\Psi_p(h, s) = \begin{pmatrix} \frac{1}{s} & 0 \\ 0 & s \end{pmatrix} \begin{pmatrix} x_{01} \\ y_{01} \end{pmatrix} + \begin{pmatrix} \frac{h}{s} \\ 0 \end{pmatrix}.$$

If we fix the parameters h and s and regard it as a function of p , it is an affine transformation. It is the transformation that can be applied to the function graph of κ_P in our objective function to obtain the function graph of $s\kappa_P \circ \phi$, where $\phi(t) = st - h$. We are now approaching the linear functions $f|_A$ and $g|_B$ as a set of points. Let e_1 be the edge of $f|_A$ and e_2 the edge of $g|_B$, we have

$$(26) \quad e_1 = (1 - t)p_1 + tp_2, \quad t \in [0, 1]$$

where $p_1 = (\min(A), f(\min(A)))$ and $p_2 = (\max(A), f(\max(A)))$

and

$$(27) \quad e_2 = (1 - t)p_3 + tp_4, \quad t \in [0, 1]$$

where $p_3 = (\min(B), g(\min(B)))$ and $p_4 = (\max(B), g(\max(B)))$, say.

Now we can express the set of parameters for which the two edges intersect as the union of pre-images of Ψ functions,

$$C(e_1, e_2) = \bigcup_{p \in e_1} \Psi_p^{-1}(e_2).$$

We want to analyze this set step by step and come up with a possible expression for the boundary of $C(e_1, e_2)$.

We can find the components of Ψ_p^{-1} using the components of Ψ_p . We want,

$$\Psi_p^{-1}(p_I) = (h(x_I, y_I), s(x_I, y_I)), \quad p = (x_{01}, y_{01}).$$

We have

$$\left\{ \begin{array}{l} \frac{x_{01}+h}{s} = x_I \\ sy_{01} = y_I \end{array} \right\}.$$

Solving for s we retrieve

$$s = \frac{y_I}{y_{01}} =: s(x_I, y_I).$$

Solving for h ,

$$h = s(x_I) - x_{01} = \frac{x_I y_I}{y_{01}} - x_{01} =: h(x_I, y_I).$$

We can see that $C(p, e_2) = \Psi_p^{-1}(e_2)$ is a continuous curve in the parameter space, since we can find an expression for this curve with respect to the parameter of a line $l_2(t) = (1-t)p_3 + tp_4$, that supports e_2 . We define

$$\gamma(t) = \Psi_p^{-1}(l_2(t))$$

The image set $l_2([0, 1])$ is the edge e_2 , therefore $\gamma(t)|_{[0,1]} = \Psi_p^{-1}(e_2)$, if the curve exists.

We can substitute into Ψ_p^{-1} , and retrieve $\gamma(t) = (h(t), s(t))$, with

$$s(t) = \frac{y_3 - t(y_4 - y_3)}{y_{01}}$$

and the quadratic function $h(t) = at^2 + bt + c$ with

$$\begin{aligned} a &= \frac{(y_4 - y_3)(x_4 - x_3)}{y_{01}} \\ b &= \frac{y_3(x_4 - x_3) + x_3(y_4 - y_3)}{y_{01}} \\ c &= \frac{y_3 x_3}{y_{01}} - x_{01}. \end{aligned}$$

Those component functions are well-defined unless $y_{01} = 0$. Let us ignore this case for a moment and deal with it later. Note also that, according to the calculations, now it can happen that $s(t) \leq 0$. But we have required that $s > 0$, therefore we have to intersect the curve with the half space where $s > 0$. Since the s -component is linear the curve is still continuous under the intersection.

Now we can analogously derive a curve ϕ , such that $\phi(t)|_{[0,1]} = C(e_1, p_I)$, if such a thing exists. We obtain

$$\phi(t) = \Psi_{l_1(t)}^{-1}(p_I) = (h(t), s(t))$$

with

$$s(t) = \frac{y_I}{y_1 + t(y_2 - y_1)}$$

$$h(t) = \frac{x_I y_I}{y_1 + t(y_2 - y_1)} - x_1 - t(x_2 - x_1).$$

Again, we have a problem if the denominator is zero, i.e. if the edge e_1 crosses the x -axis. Assume for now this does not happen. Then we encounter $s < 0$ iff e_1 and p_I are on opposite sides of the x -axis. Clearly they cannot intersect for $s > 0$ in this case, therefore they have no intersection function. But if both p_I and e_1 are on the same side of the x -axis, ϕ is continuous and well-defined.

Now, we want to find the boundary of $C(e_1, e_2)$. A point p can be said to lie on the *boundary* of a set iff every ϵ -neighbourhood around p contains points from the set as well as points from the complement of the set. We denote the boundary of a set S as $\partial(S)$.

Since $C(e_1, e_2) = \bigcup_{p \in l_1([0,1])} C(p, l_2([0,1]))$ and because of the fact that C is continuous in the parameters of both l_1 and l_2 , it is clear that we can only step outside C at the points where we encounter the boundary of $[0, 1] \times [0, 1]$ with those parameters (t_1, t_2) , i.e. if either one of them is either zero or one. In other words,

$$\partial(C(e_1, e_2)) = \bigcup_{p \in \{l_1(0), l_2(1)\}} \Psi_p^{-1}(e_2) \cup \bigcup_{p \in e_1} \Psi_p^{-1}(\{l_2(0), l_2(1)\}).$$

Therefore we can easily express the boundary using four curves. Namely

$$\begin{aligned} \gamma_1(t) &= \Psi_{p_1}^{-1}(l_2(t)), & \gamma_2(t) &= \Psi_{p_2}^{-1}(l_2(t)), \\ \gamma_3(t) &= \Psi_{l_1(t)}^{-1}(p_3), & \gamma_4(t) &= \Psi_{l_1(t)}^{-1}(p_4), \end{aligned}$$

where the p_i are the endpoints of e_1 and e_2 , as defined in (26),(27). We already know what those curves look like, since we can substitute into the expressions we have derived for γ and ϕ earlier.

We still have to investigate what happens to Ψ_p^{-1} if we encounter $p = (x_{01}, 0)$. What happens, to the objective function at a point (h, s) where two edges intersect at $y_I = 0$? The function as it is defined in (21) is well-defined for the parameters (h, s) .

The y -component function of Ψ_p^{-1} maps every value to zero, therefore the x -component maps the intersection point $(x_I, 0)$ to the line $\{(h, s) \mid h = x_I s - x_{01}\}$ in the parameter space. You can see an example of this line in Figure 45.

As we can see this is not a problem, but we have to be careful with our derivation of the boundary in this case, since we required $s > 0$ in the beginning. To approximate the boundary of two edges where this case occurs, we can pick points $p_\epsilon = (x_{01}, \epsilon)$ and $p_{-\epsilon} = (x_{01}, -\epsilon)$ with $\epsilon > 0$ and sufficiently small.

Now it happens that the line for which two edges intersect at $y = 0$ is part of the boundary. But there are only finitely many edges in the curvature function and less edges for which this situation occurs. Therefore the boundary is still a set of measure zero.

5.4.1. *Examples.* Examples of boundaries are displayed in the proceeding figures, where each of the figures on the left show the respective edges or piecewise functions. In the figures on the right h is on the horizontal axis and s is on the vertical axis. The boundary of the domain of one intersection function can intersect itself once. This happens where a point $p = (h, s)$ with $s = \sqrt{\frac{c}{a}}, h = \frac{bs-d}{as}$, is part of the domain, a and c being the slope of the edges and b and d their y -intercept, see Figure 38 for an example. The boundaries of the domains of the set of intersection functions

partition the domain. We can picture the boundary of this partition as an embedded graph where each face has exactly four sides, and the sides themselves are concrete curves, and where the edges of the planar graph can overlap with each other, see Figure 42 for an example. The cell can also be open as it is the case in Figures 43 and 45.

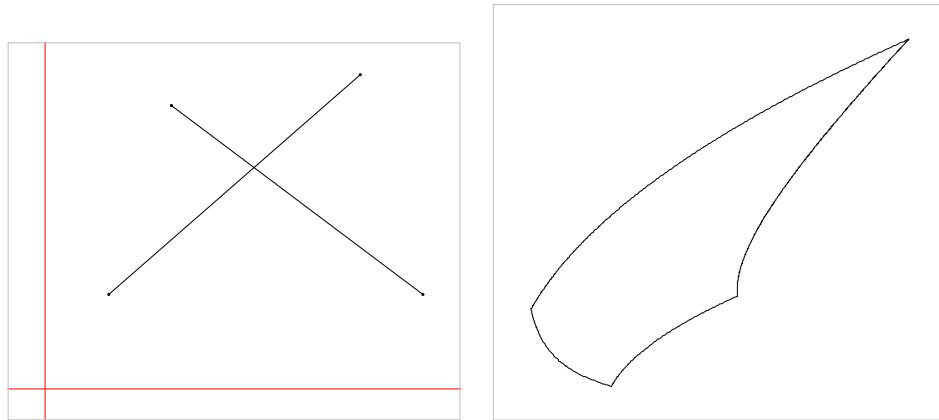


FIGURE 37. Boundary of the domain of one intersection function.

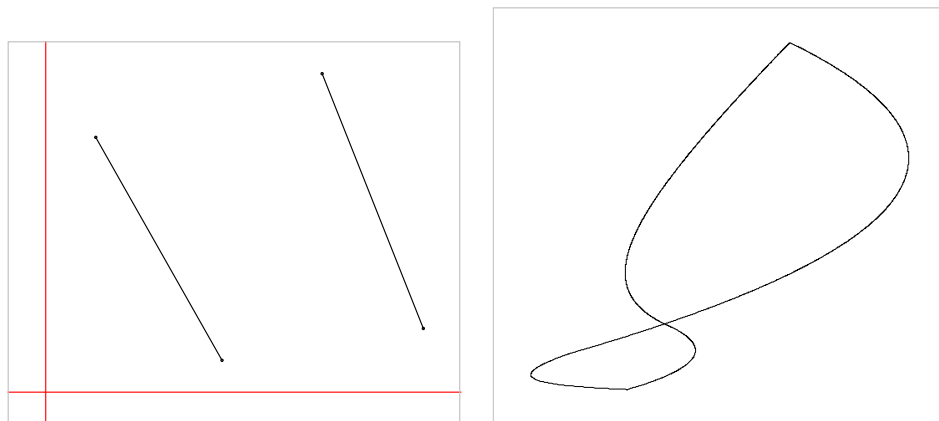


FIGURE 38. Self-intersecting boundary of the domain of one intersection function. Self-intersections can only happen once for $s > 0$.

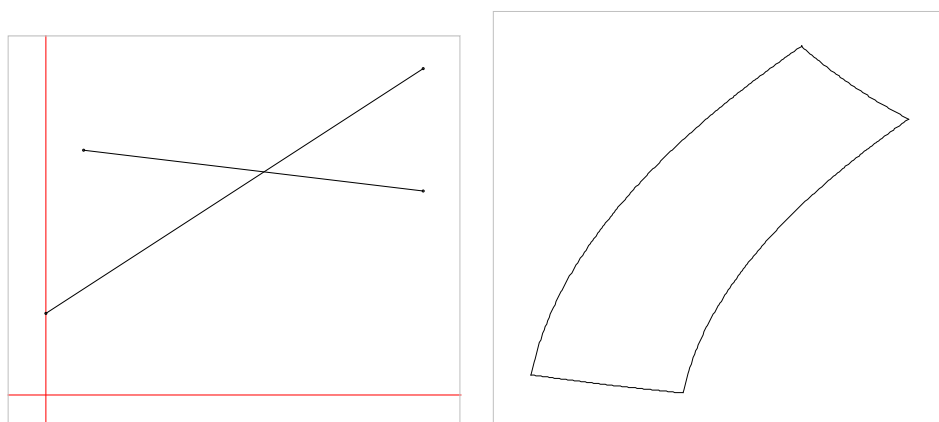


FIGURE 39. Another example for the boundary of the domain of the intersection function of two edges.

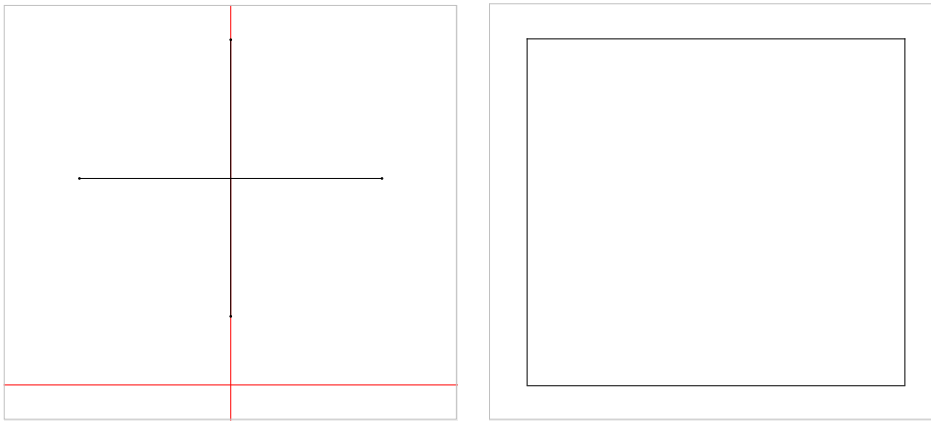


FIGURE 40. Another example, the edge being transformed is the vertical edge. Both edges have equal length and intersect at $(0, 1)$. The resulting boundary is a square.

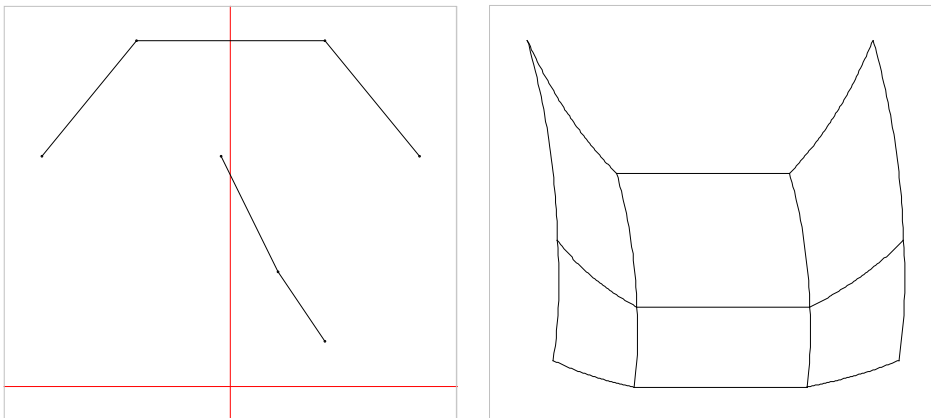


FIGURE 41. Boundary of the domain partition for two piecewise linear functions. The lower function is the function being transformed.

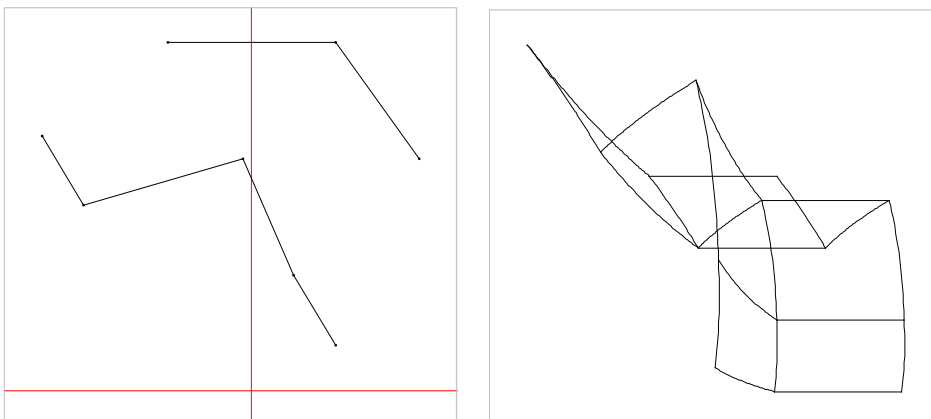


FIGURE 42. Self-intersecting boundary of the domain partition for two piecewise linear functions. The lower function is the function being transformed.

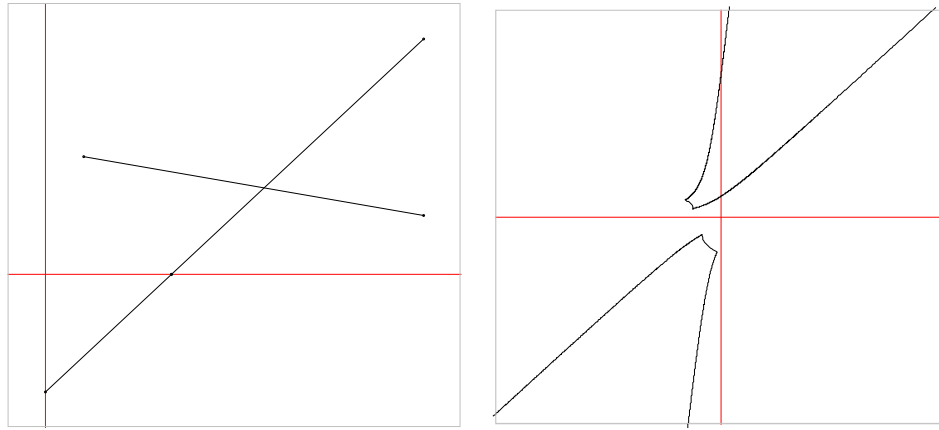


FIGURE 43. Example for the boundary of an open cell. Also showing $s \leq 0$, which is not part of the domain. The function that intersects the x -axis at p is the function being transformed. The cell must be open since p stays on the axis under the transformation Ψ_p .

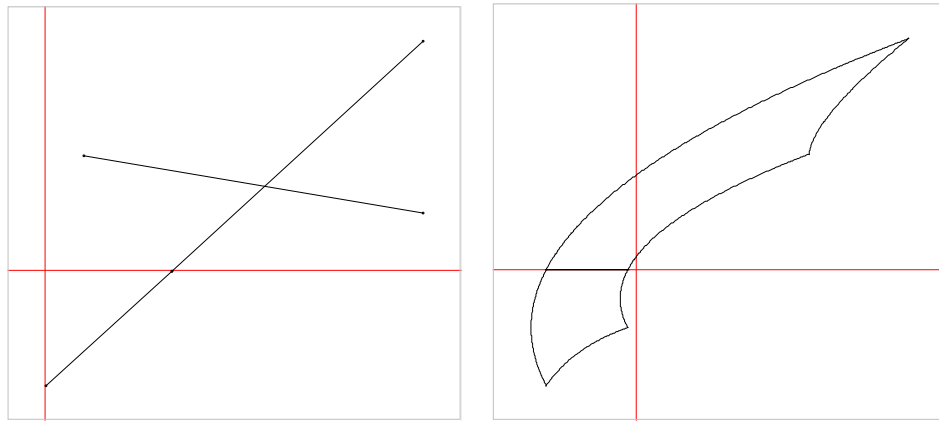


FIGURE 44. Example of the boundary of a closed cell. Also showing $s \leq 0$, which is not part of the domain. The edges are the reversed edges from Figure 43.

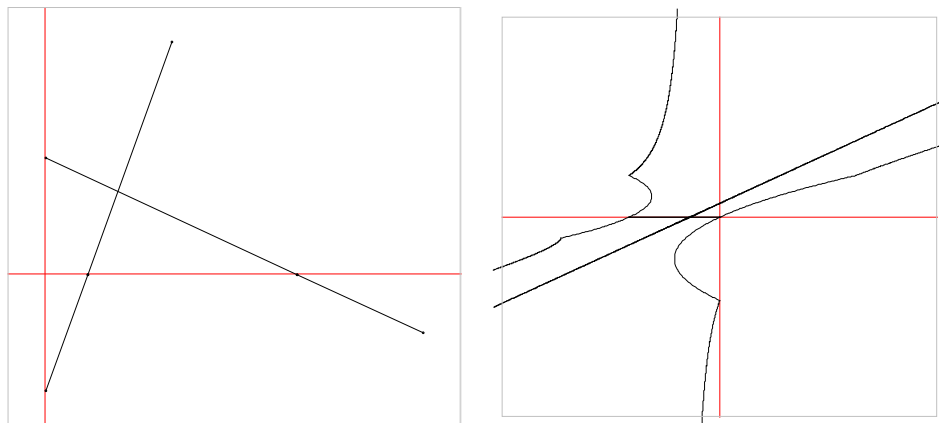


FIGURE 45. Another example for the boundary of an open cell and where the line of the intersection of the edges at $y = 0$ can be seen. Also showing $s \leq 0$, which is not part of the domain.

5.5. Illustrative Example in 2D. Now we want to give a concrete example for the function $d(h, s)$ and its domain partition. The second incentive is to investigate the counter-example from Section 4.4. There we computed the minimum of the normalized distance function of the piecewise functions a and b from Figure 22, bounded to a certain subspace and claimed, that this was a global minimum. Without loss of generality we modify the functions a and b such that they are centered with respect to the parameter t , see Figure 46. For a grid point approximation of the objective function see Figure 47.

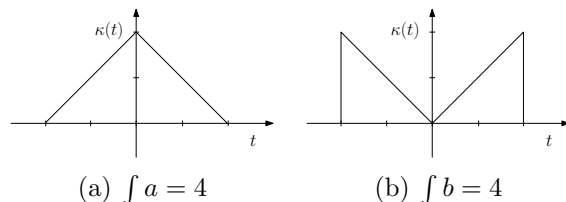


FIGURE 46. Modified functions a and b from Page 29.

Consider the cells of the domain partition displayed in Figure 48. We can sample a point in each cell to find the intersections of the cell. Figure 49 shows a representative from each cell. Since the function is symmetric we will only consider $h > 0$.

One could object that this list of cases is not complete and that there could be a cell such that the second triangle of b is fully covered by the triangle of a , plus that a part of the first triangle of b is covered, similar to case F . If only one of the triangles is fully covered, then the area of intersection is equal to two, therefore the normalized distance would be 0.5, which is greater than a function value we have already found in Section 4.4. It can be seen that the described case where, in addition to this, a part of the other triangle is covered, is geometrically impossible. Therefore the list of considered cases is complete with respect to our objective.

From the grid point approximation of the graph it is clear that a global minimum must lie in cell A . Therefore we will now determine the distance function bounded to A explicitly using the decomposition described in Section 5.2.

We can write the two functions analytically as follows,

$$a(t) = \begin{cases} t + 2, & -2 < t < 0 \\ -t + 2, & 0 < t < 2 \end{cases}$$

$$b(t) = \begin{cases} -t, & -2 < t < 0 \\ t, & 0 < t < 2 \end{cases}.$$

Note that we actually have a little problem here, since the vertical edges are not part of a function. But we can simply introduce an intersection function that is a constant function in this case, therefore this is no problem.

For cell A , we have the following intersection functions

$$\omega_0(h, s) = \frac{sh - 2s}{s^2}, \quad \omega_1(h, s) = -2,$$

$$\omega_2(h, s) = \frac{sh - 2s}{s^2 + 1}, \quad \omega_3(h, s) = \frac{-sh - 2s}{-s^2 - 1},$$

$$\omega_4(h, s) = 2, \quad \omega_5(h, s) = \frac{-sh - 2s}{-s^2}.$$

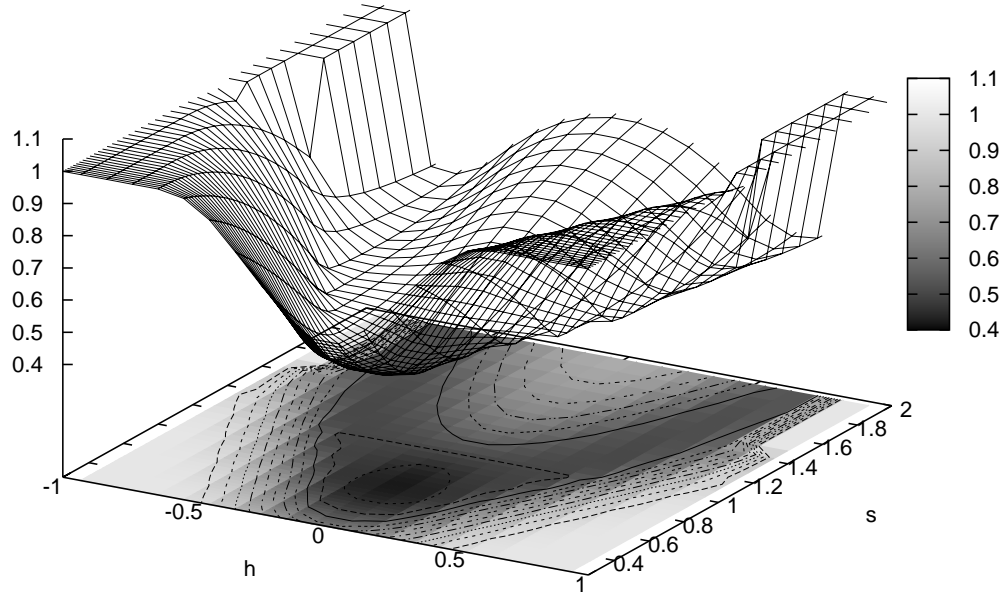


FIGURE 47. Grid point approximation of the function $\frac{d(h,s)}{\int a + \int b}$ with 50 grid points. The graph of the function is symmetric along $h = 0$.

And we have the decomposition,

$$(28) \quad d(h,s)|_A = \sum_{i \in \{1,3,5\}} \left([\mathcal{A}(t)]_{s\omega_{i-1}(h,s)-h}^{s\omega_i(h,s)-h} - [\mathcal{B}(t)]_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} \right) \\ - \sum_{i \in \{2,4\}} \left([\mathcal{A}(t)]_{s\omega_{i-1}(h,s)-h}^{s\omega_i(h,s)-h} - [\mathcal{B}(t)]_{\omega_{i-1}(h,s)}^{\omega_i(h,s)} \right),$$

with the anti-derivatives of a and b ,

$$\mathcal{A}(t) = \begin{cases} 0, & t < -2 \\ \frac{1}{2}t^2 + 2t + 2, & -2 < t < 0 \\ -\frac{1}{2}t^2 + 2t + 2, & 0 < t < 2 \\ 4, & 2 < t \end{cases}, \quad \mathcal{B}(t) = \begin{cases} 0, & t < -2 \\ -\frac{1}{2}t^2 + 2, & -2 < t < 0 \\ \frac{1}{2}t^2 + 2, & 0 < t < 2 \\ 4, & 2 < t \end{cases}.$$

We can calculate and simplify the decomposed function using modern computer algebra systems and retrieve

$$d(h,s)|_A = \frac{2(4s^4 - 8s^3 + s^2h^2 + 12s^2 - 8s + 4)}{s^2 + 1}$$

This function has a unique minimum for the range $U = [-1, 1] \times [0, 1]$, at a point p that is contained in A . Also $A \subset U$ therefore p also minimizes $d(h,s)|_A$, and we have

$$\min_{h,s} \frac{d(h,s)|_A}{\int a + \int b} = \frac{d(0, 0.6823278038)}{\int a + \int b} = 0.4185878202.$$

Note that we had already computed this minimum in Section 4.4 by minimizing over the symmetry axis.

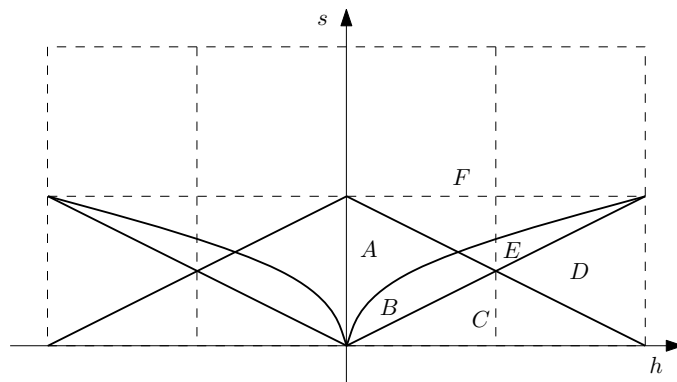


FIGURE 48. Domain partition for the range of the objective function as it is displayed in Figure 47. The dashed grid has unites equal to one.

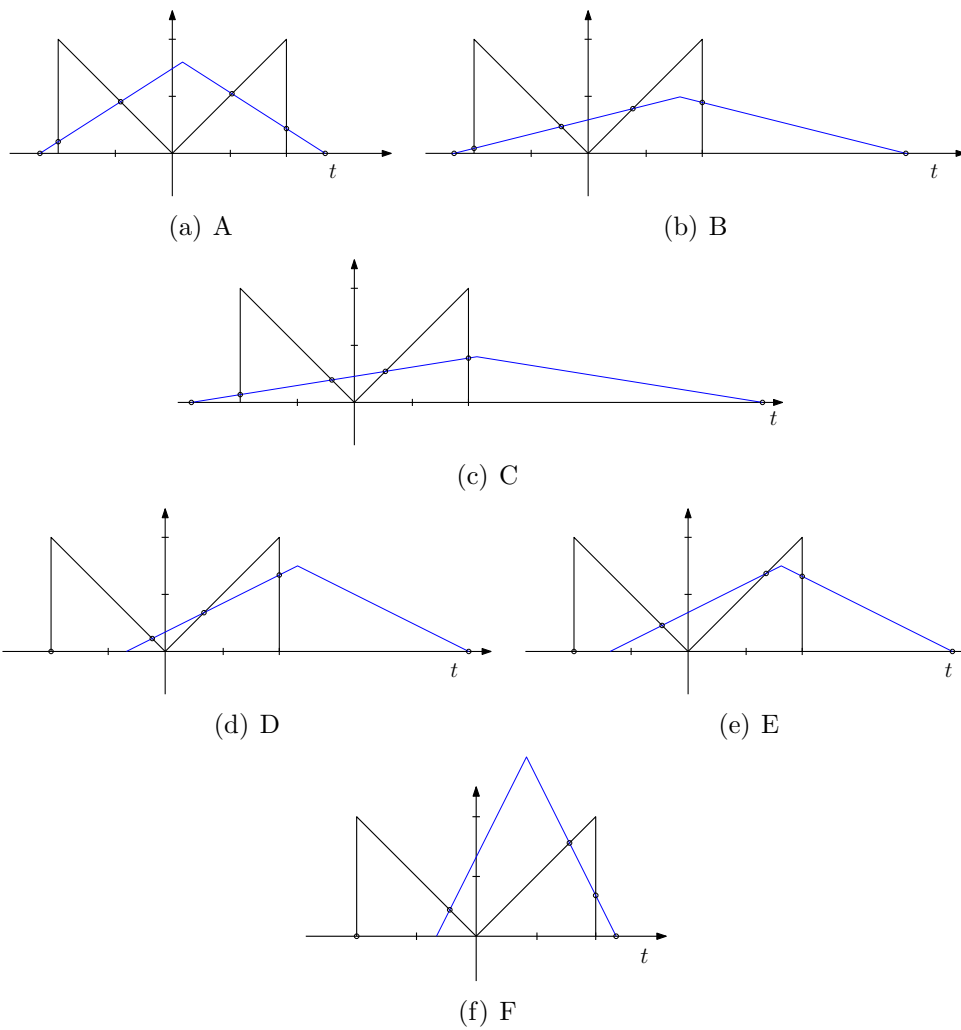


FIGURE 49. Representatives of the intersections for the respective cells in the domain partition as they are labeled in Figure 48.

6. COMPUTATION

Now that we have analyzed the objective function thoroughly in the last chapter, we want to find a way to solve the optimization problem. The summary of gradient based optimization algorithms [Sny05] serves as a basis for the following discussion. According to them we are dealing with an unconstrained non-convex two-dimensional minimization problem. Where the optimization function f as defined in (21) is continuous and almost everywhere differentiable. However, we do not have a global approximation of the function. We have an exact representation as a piecewise polynomial fraction, where the pieces in the worst case are at least quadratic and possibly in the order of $O(n^2m^2)$ in the number of the edges of the input curvature functions. Since we are only interested in the value of f at a global minimum, we do not care if this minimum is unique, less if it is a strong minimum.

A simple class of algorithms for unconstrained minimization are the line search descent algorithms as described in [Sny05]. The general outline of those algorithms is described as follows. At each iteration step a descent direction u_i , in which an improvement of the current position x_i is conjectured, is chosen. Then the optimization function is minimized over the whole subspace in this direction, $\lambda_i = \arg \min_{\lambda} \{d(h, s) \mid (h, s) = x_i + \lambda u_i\}$, using a method to be specified, and the position is set to this minimum $x_{i+1} = x_i + \lambda_i u_i$. The step is repeated until one or all of the following convergence criteria is fulfilled, the improvement of the function value $f(x_i) - f(x_{i+1}) \leq \epsilon_1$, or the step-size $\|x_i - x_{i+1}\| \leq \epsilon_2$, or the norm of the gradient at the current position, which indicates that a local minimum has been reached. The algorithms differ in the way they choose the descent direction and in the way they perform the line search.

The simplest strategy for a descent direction is to choose the steepest descent direction, which is the opposite direction of the gradient. This method has been known already by Cauchy in 1847. Why the gradient gives the steepest descent can be seen with the following considerations, taken from [Sny05, p. 40].

Proof. At x we seek the unit vector u such that for $F(\lambda) = f(x + \lambda u)$, the directional derivative

$$\left. \frac{df(x)}{d\lambda} \right|_u = \frac{dF(0)}{d\lambda} = \langle \nabla f(x), u \rangle$$

assumes a minimum value with respect to all possible choices for the unit vector u .

By Schwartz's inequality:

$$\langle \nabla f(x), u \rangle \geq -\|\nabla f(x)\| \|u\| = -\|\nabla f(x)\| = \text{least value.}$$

For the particular choice $u = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$ the directional derivative at x is given by,

$$\frac{dF(0)}{d\lambda} = -\left\langle \nabla f(x), \frac{\nabla f(x)}{\|\nabla f(x)\|} \right\rangle = -\|\nabla f(x)\| = \text{least value.}$$

Therefore u is the vector at x that points in the direction of the steepest descent. \square

The different methods for the line search step are not relevant for our problem. Since we would in each iteration step optimize over a continuous piecewise rational function, where the number of pieces is in the worst case in the same order than the number of pieces in the original two-dimensional problem. Therefore we will leave out the line search step and instead in each iteration step attempt to go a step of fixed size in the minimizing direction. This will yield a directed search in the parameter space, and will be obviously an improvement to the brute force approach.

The posed problem could almost be reduced to minimizing a set of optimization functions, each of them a piece of our piecewise function, with their subdomains as the respective constraints. Except that the given piecewise function is continuous, which is a property that our method relies heavily upon.

For the same reason the second order Quasi-Newton methods described in [Sny05, p. 51] would not work properly if applied to our problem, since we do not have a global approximation of the objective function. Those methods use an approximation of the Hessian matrix at the current position of the objective function, that is updated in each iteration step. Whereas the original Newton method, which recomputes the Hessian matrix and its inverse at each iteration step, might actually be another option we have to reserve for later work. As well as the method of simulated annealing, which can be applied to almost any optimization problem.

Since our optimization function is not convex the gradient descent method will in most cases not converge to a global minimum. A common practice is to rerun the algorithm with random start positions. The resulting algorithm is sometimes called stochastic gradient descent.

6.1. Algorithm. We have implemented a variation of the steepest descent algorithm which, in each iteration step, attempts to go one step of fixed size in the parameter space in the opposite direction of the gradient at the current position. The source code of the essential Java classes `GD` and `GD.Position`, can be found in the appendix, starting from page 79.

The general outline of the algorithm, as we implemented it, is as follows.

```

1  public static double Main.match(Function kP, Function kQ) {
2
3      double scalingP = kP.length()/2.0;
4      Function kP = computeScaled(kP, 1/scalingP);
5      List<Function> kPs = computeReflections(kP);
6
7      double scalingQ = kQ.length()/2.0;
8      Function kQ = computeScaled(kQ, 1/scalingQ);
9
10     double result = null;
11
12     for(Function kP: kPs){
13
14         GD gd = new GD(kP, kQ);
15         int trials = GD.TRIALS;
16
17         while(trials > 0){
18             gd.initRandomly();
19             result = min( gd.run(GD.PRECISION), result);
20         }
21     }
22
23     return result;
24 }

```

The function `match` is given two curvature functions, κ_P and κ_Q , and returns their distance under reflection, change of orientation and the discussed transformations of the initial curve, as far as the gradient descent was able to explore the objective function. For the gradient descent it is important, that the variables have a similar scale, since otherwise the direction of the gradient is dominated by the larger variable. It is best if the problem should be stated in a way that the probability that the solution is an element of $[0, 1] \times [0, 1]$ is very high and also uniformly distributed on this range.

We assume that the curves have approximately the same scale and that the optimal scaling factor is somewhere in the range $[\frac{1}{3}, 3]$. Therefore we transform the

curvature functions in lines 4 and 8 according to the formula we have derived, such that they measure the curvature of the input curves uniformly scaled by the reciprocal of their arc lengths and shift it to the center. This way the curvature functions are zero outside the range $[0, 1]$ and the optimal shifting value is likely to be in the range $[-1, 1]$ for scaling values near one.

In line 5 we generate the four possible combinations of reflection and change of orientation for κ_P and in line 12 we iterate over those and apply the whole algorithm to each of the curvature functions obtained this way and κ_Q .

The number of trials is a global parameter. For each trial the gradient descent object is initialized with a random choice for (h, s) . The following pseudo code is for the function `GD.run(precision)` which performs the gradient descent from the current position:

```

1 public double GD.run(double precision) {
2
3     while(true){
4
5         try{
6             Position newPos = new Position(currPos - eps * currPos.gradient());
7
8             if(!newPos.isValid())
9                 throw new InvalidPositionException(newPos.toString());
10
11            double gain = currPos.value() - newPos.value();
12
13            if(gain < 0)
14                throw new SteppedBeyondException(""+gain);
15
16            //dot product has the same sign as the cosine
17            if( dot( currPos.gradient(), newPos.gradient()) < 0)
18                throw new SteppedBeyondException("gradient_based");
19
20            currPos = newPos;
21
22            if(gain < precision) break; //success?!
23
24        }catch(InvalidPositionException ex){
25            this.stepSize *=0.5;
26
27        }catch(SteppedBeyondException ex){
28            this.stepSize *=0.5;
29        }
30    }
31
32
33    return currPos.value();
34 }

```

The function is given the `precision` parameter and returns the value of the local minimum it converged to with the given precision. In each iteration step a new position is computed and evaluated. Details on how the gradient is computed can be found in the proceeding subsection. For now let us assume we have computed the gradient for the current position. The new position is a step in the opposite direction of the normalized gradient in line 6. Then we evaluate if the new position is valid in line 8. Obviously we are in an invalid position if $s < 0$, we also declared positions where the supporting sets of the functions have an empty intersection to be invalid. Since there the objective function should be constant and the gradient has no useful information.

We compute how much we would gain in terms of the function value in line 11. If the new position is in fact uphill of the current position we do not take it. We also evaluate in line 17 if the gradients of the new and old positions point in opposite

directions. This also indicates that we have stepped beyond a local minimum. In the case that the evaluation of the new position turned out negative, we decrease the step-size by a factor of one half, in lines 28 and 31 respectively, and start a new iteration with the same gradient.

If the evaluation is positive we accept the new position and also start a new iteration step, given that the gain is not below the threshold. The algorithm as it is implemented therefore also converges at saddle points and in valleys. Note that we reset the step size to the initial value if we take the new position in line 25.

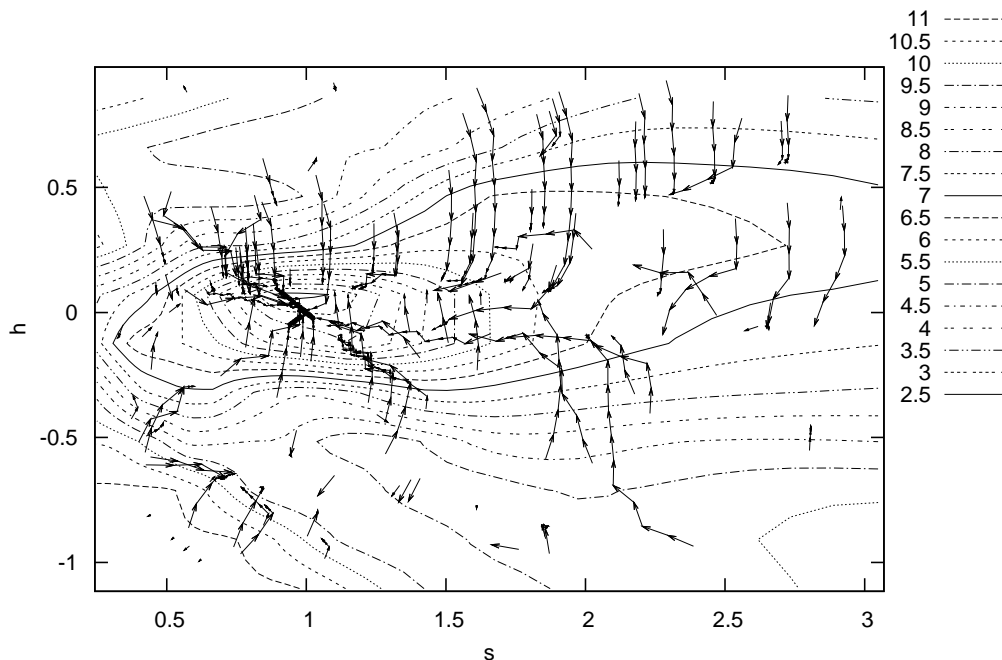


FIGURE 50. Example of the paths in the parameter space which the algorithm traverses. The objective function is displayed via contour lines that were computed from grid point data.

6.1.1. *Computation of the Gradient.* At each position in the parameter space we have a function as in Equation (24), where each term of the sum is one of the form of either

$$(29) \quad \mathcal{K}_P(\omega_i(h, s) - h)|_A = \frac{a}{2} (\Phi_P(h, s))^2 + b\Phi_P(h, s) + C_P$$

$$(30) \quad \mathcal{K}_Q(\omega_i(h, s))|_B = \frac{c}{2} (\Phi_Q)^2 + d(\Phi_Q) + C_Q$$

where C_P and C_Q are constants that will disappear in the differentiation, and where the variables a, b, c, d are the parameters of the respective intersecting edges, $\kappa_P(t)|_A = at + b$, and $\kappa_Q(t)|_B = ct + d$. And where

$$\Phi_P(h, s) = s \left(\frac{ash - bs + d}{as^2 - c} \right) - h = \frac{as^2h - bs^2 + ds}{as^2 - c} - h$$

$$\Phi_Q(h, s) = \frac{ash - bs + d}{as^2 - c}.$$

Because of the linearity of differentiation it suffices to compute the partial derivatives of each of those terms. The sum of the partial derivatives is then the partial derivative of the objective function at the current position.

We compute the partial derivatives for the function in (29),

$$\begin{aligned}\frac{df(h, s)}{dh} &= (a(\Phi_P(h, s)) + b) \frac{d\Phi_P(h, s)}{dh} \\ &\text{with } \frac{d\Phi_P(h, s)}{dh} = \frac{as^2}{as^2 - c} - 1 \\ \frac{df(h, s)}{ds} &= (a(\Phi_P(h, s)) + b) \frac{d\Phi_P(h, s)}{ds} \\ \text{with } \frac{d\Phi_P(h, s)}{ds} &= \frac{(2ahs - 2bs - d)(as^2 - c) - 2as(as^2h - bs^2 - ds)}{(as^2 - c)^2}\end{aligned}$$

And for the function in (30),

$$\begin{aligned}\frac{df(h, s)}{dh} &= (a(\Phi_Q(h, s)) + b) \frac{d\Phi_Q(h, s)}{dh} \\ &\text{with } \frac{d\Phi_Q(h, s)}{dh} = \frac{as}{as^2 - c} \\ \frac{df(h, s)}{ds} &= (a(\Phi_Q(h, s)) + b) \frac{d\Phi_Q(h, s)}{ds} \\ \text{with } \frac{d\Phi_Q(h, s)}{ds} &= \frac{(ah - b)(as^2 - c) - 2as(ash - bs - d)}{(as^2 - c)^2}.\end{aligned}$$

The partial derivatives evaluated at the current position (h, s) yield the s and h -component of the gradient vector at this position. We therefore only have to find the edges that intersect for the current subdomain and substitute for each intersection into the above formulas, as it is being done in the functions `gradientH()` and `gradientS()` in the implementation starting from page 79. The two functions are defined for an `GD.Position.Intersection` object. The actual gradient is being computed in the method `evaluate()`, which is defined for a `GD.Position` object. Since we do not have a sublinear method to predict which edges will intersect for a given choice of parameters (h, s) , this function has the same complexity as determining the actual function value $d_{P,Q}(h, s)$ by iterating over the edges of κ_P and κ_Q .

6.1.2. Comparison to the Brute Force Method. We want to compare this algorithm to the brute force method which performs a grid point approximation. In some respect the methods are similar, since the gradient descent does not grant an optimal solution either and has to be restarted at random positions in the parameter space. Therefore we should be careful with the assumption that the gradient descent performs better on any data set. We have already stated that the complexity of the brute force method is $O(k^2(m + n))$ with n, m being the number of assignments of the curvature functions and k the number of grid points in one dimension. For each grid point we compute $s\kappa_P \circ \phi$, which is linear in the number of assignments and compute the integral of the difference to κ_Q , which is also linear in the number of assignments and there are k^2 grid points. The downside of the method is, that the objective function can get very steep, especially near the global minimum and therefore it is possible that the whole area around the minimum where the function values are significantly smaller falls between gridpoints and is therefore not registered. The gradient descent uses local information to do a directed search and will

therefore not "miss" the minimum if it is already near it, but it also has its downsides. Let us analyze the complexity as far as it is possible to do so. We cannot be sure how many steps the gradient descent takes to converge each time to a local minimum, in fact it is not clear if it converges within a finite number of steps at all. Therefore we have a variable that counts the number of steps and abort the search if a certain threshold for the maximal number of steps is reached. But we do not assume, that the descent will take the maximal number of steps each time. Let a be the average number of steps that the gradient descent needs to converge. And let N be the number of trials the gradient descent is supposed to perform. At each iteration step the gradient descent does the same amount of work as the brute force methods does at a grid point. Therefore we obtain complexity of $O(aN(m+n))$. We probably need N to be at least in the same order as the number of grid points in one dimension to achieve a comparable result. Our experiments show that for 70 trials for the gradient descent and a grid point approximation with 100 points in one dimension for the brute force method we get comparable results for our input data set, which is well-behaved. We do not expect this, but if the convergence is really slow, such that we also need N steps on average, then we have no improvement upon the running time of the brute force method. And a better result is not granted either. See Figure 51 for an example of an objective function, that will be hard to minimize using the proposed method. However for the input data set we used, the algorithm proved to be more efficient than the brute force method.

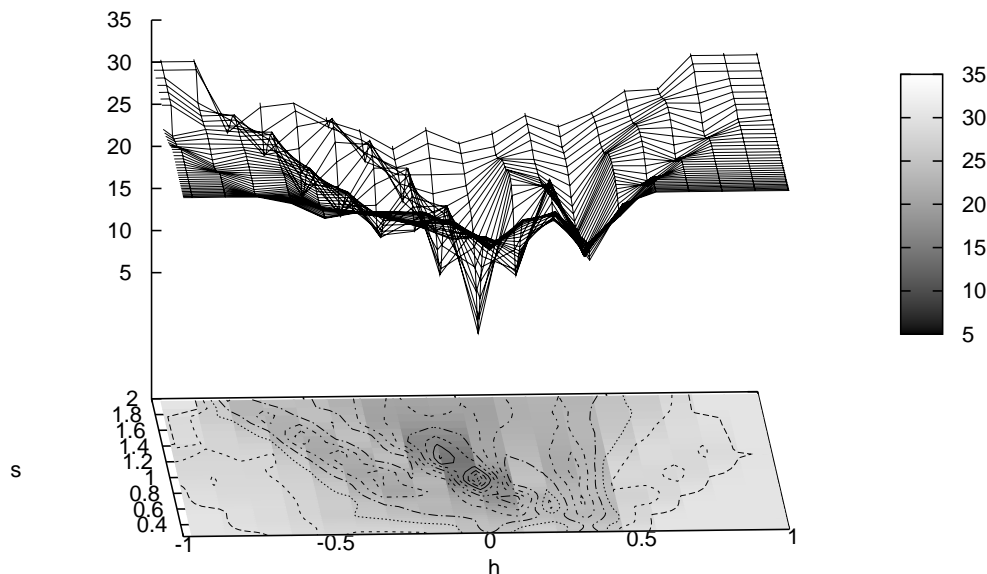


FIGURE 51. Example where the gradient descent method will fail. This is the objective function for the curves P and Q used in the examples for the case studies in Figures 30(a) and 30(d). We conjecture that the reason why the objective function has so many local minima are the similarities of the curves at low scale together with the curves having self-similarities.

6.2. Experimental Results. We deliver and compare the results of two test units. In one test unit we computed the pairwise distance of the set of input curves with respect to a curvature measured at a high scale and in another we used low scale curvature. The results can be found in the appendix. For each shape we display curvature function that was used for matching together with the three best matches in the input set. The set of input curves used can also be found in the appendix. The data was initially retrieved from scans of archaeological fragments, with a focus on what are believed to be the handles. We do not have ground truth for the data therefore we cannot evaluate the method using automated testing to judge the performance by the rate of false-positives and false-negatives. We therefore visualize the three best matches for each shape and additionally cluster the data according to the computed distances.

As parameters for the gradient descent we used 100 trials and a precision parameter of 0.0001. The abort threshold for the maximal number of steps was set very high, to 1000, we conjecture that the gradient descent converged long before this number was reached almost every time. Matching two curves at low scale took on average 0.5 seconds and at high scale on average 1.5 seconds on a standard desktop computer with an AMD Athlon(tm) 64 X2 Dual Core Processor 4800+. The two test units were run simultaneously and took approximately 6 hours for the low scale matching and 10 hours for the high scale matching.

6.2.1. Clustering. For the evaluation of the results we want find clusters in the set of curves according to the retrieved distances. The distance measure gives us a metric space, but no coordinates. We therefore cannot apply any of the standard clustering algorithms for sets of elements in a vector space.

We are given a complete graph with weighted edges. The vertices represent the curves and the weights are the distances between the curves. Now the objective is to find subsets of vertices such that the weights of the edges in the induced subgraph are significantly smaller than the edges between cluster and non-cluster vertices.

$$S \text{ is a cluster} \Leftrightarrow \forall p, q \in S, q' \notin S : \delta(p, q) \ll \delta(p, q')$$

If there exist strong relationships of this kind in the data set such that clusters are easily identified, we can find those clusters using an algorithm for the minimum spanning tree. We run the Kruskal algorithm which builds the MST iteratively from the sorted list of edges, each time testing if the edge added to the tree set would produce a circle. But we stop as soon as we encounter an edge that is significantly larger than the previous edges. The vertices of one component of the MST at this stage are taken as a cluster.³ This method amounts to the same as taking out all the edges above a certain threshold and computing an MST on the remaining set of small edges.

Unfortunately the situation is often not that clear. The algorithm does not yield the expected result if vertices are connected by a chain of small edges, without forming a cluster themselves. On the other hand a subset of the vertices may very well be considered a cluster even if it has some small outgoing edges. Since we can have a situation where a vertex is not considered an element of the cluster because it is not considered similar to the other cluster vertices, but it can still have a strong connection to at least one cluster vertex. This is of course within the bounds of the triangle inequality since the distance measure is a metric, but we conjecture that situations like this can happen. See Figure 52 for an idea of an example.

³A better method would be to check a local condition and continue with the construction of the tree. But we conjecture the argument of chains of small edges also holds here.

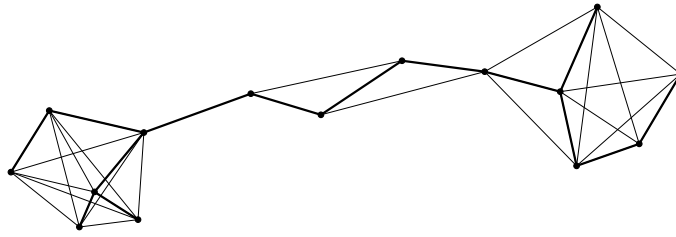


FIGURE 52. Two clusters in the set of small edges can be connected by a chain on the MST.

On the basis of those considerations, we designed the following heuristic clustering algorithm. The idea is to compute the MST as described above and if the diameter of a component exceeds a certain threshold to remove the edge with the largest weight and recompute the MST from the remaining set of small edges. This step is repeated until the diameter of each component is below the threshold.

The thresholds are chosen with respect to the maximum and the median of the distances. We can bound the maximal edge weight within a cluster by the product of the maximal edge weight of the MST component and the diameter.

```

1  public static Clustering(double [][]  $\delta$ ){
2
3
4      clusters = new Vector<Cluster>[5]();
5
6      double bound = min ( max (  $\delta$  ) * 2/3, median(  $\delta$  ) * 1/2);
7
8      for(int diameter = 3; diameter < 8; diameter++){
9
10         double [][]  $\tilde{\delta}$  = Arrays.copyOf(  $\delta$  );
11
12         double threshold = bound/diameter;
13
14         TreeSet trees = new MST(  $\tilde{\delta}$  , threshold);
15
16         while(true){
17             boolean change = false;
18
19             for(Tree t: trees){
20                 if(t.diameter() > diameter){
21
22                     Edge e = t.getHeaviestEdge();
23
24                      $\tilde{\delta}$  [e.u()][e.v()] =  $\infty$ ;
25                      $\tilde{\delta}$  [e.v()][e.u()] =  $\infty$ ;
26
27                     change = true;
28                 }
29             }
30
31             if(!change) break;
32
33             trees = new MST(  $\tilde{\delta}$  , threshold);
34         }
35
36         clusters[diameter] = trees.getComponents();
37     }
38 }

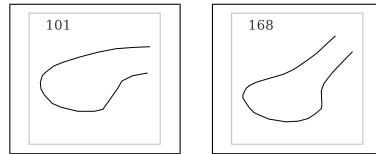
```

The complete results of the clustering can be found in the appendix.

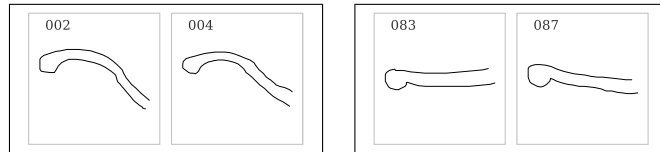
6.2.2. *Observations.* The results are very satisfactory on first sight. But it is hard to evaluate them rigorously since there exists no ground truth data. Our clustering algorithm finds clusters of shapes that could just as well be found by a human, but

a large number of shapes, approximately half the set in the low scale matching and much more in the high scale matching, were not clustered. We conjecture that this is due to the fact that the data is very dense under the distance measure, in the sense that many shapes are similar to other shapes in the set, but the similarity links extend in different "directions" and therefore the shapes do not form a cluster. In the following the scale of the curvature is not to be confused with the scale of the curve, which we refer to as high scale or low scale. We apologize if this naming conflict leads to confusion.

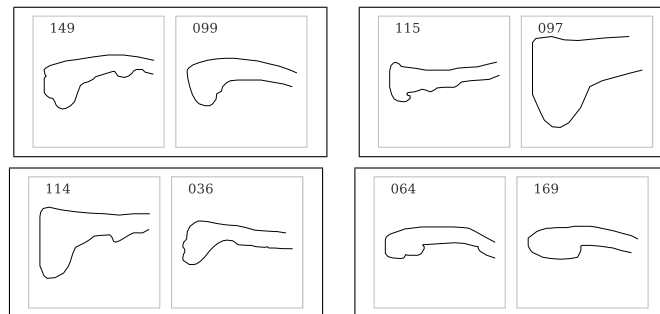
In the third and first cluster in the first clustering round in the high scale matching it can be observed that the distance measure is sensitive to even slight changes in the overall direction of the contour. The main distinction between shapes in those two clusters is that the shapes in the first cluster depict handles that bend downwards, for example 101, and the shapes in the third cluster depict handles that bend upwards, for example 168.



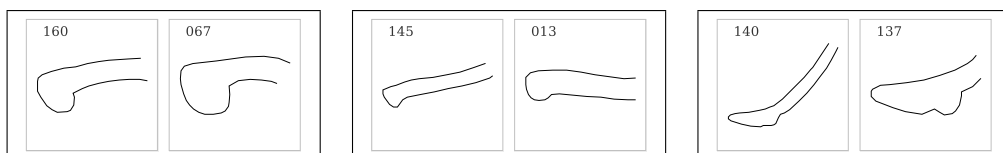
This distinction seems to be very fine, since in the next clustering round the two clusters fall together. Our clustering algorithm does not find the same distinction in the low scale matching results. But it is likely that this is due to the way we choose the threshold. There are differences between the matching at high scale and the matching at low scale, though. Some matches that make sense from the perspective of human perception are only found at low scale, for example

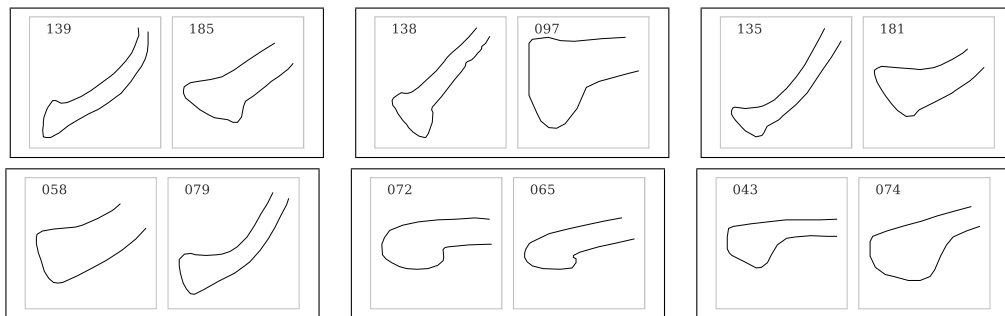


Striking examples where the low scale matching overcomes noise are the matches

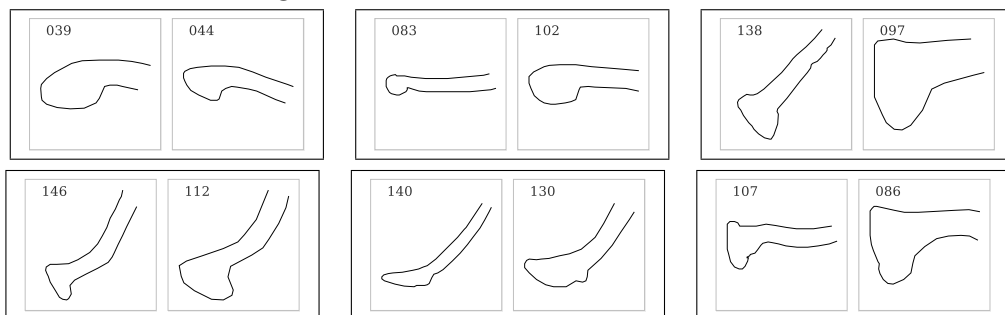


The same shapes, or similar ones, are also matched in the high scale matching (149-43, 114-97, 64-169) but the distances are very large and the reason that those are the best matches could be that no other shape matched the noisy shape better. It would be interesting to see how the order of best matches of one shape changes across the curvature scales, but we have not investigated this further. Note that in the low scale matching 115-97 both, noise and scale differences are being overcome. It seems that the high scale matching is robust to scale changes of the curve, looking at the clustering results and the matches



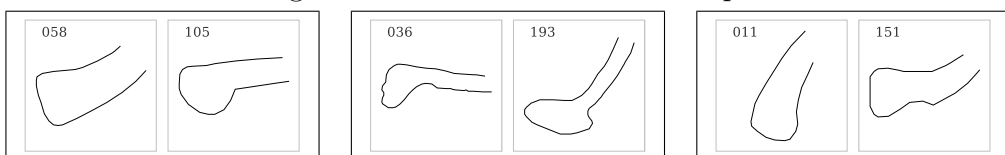


Here in fact our visualization of the input curves is deceptive. The actual size of the matching curve segments is the same in those matches. Therefore their curvature functions are computed at the same scale and should be very similar over the parameter range of the matching segments. But for the gradient descent algorithm the curvature functions are rescaled according to the total length of each curve and this initial correspondence is lost. Therefore this is still a good result. But in order to get faithful results about the scale-invariance we would have to rerun the test units with duplicates of the curves at different scales. In the low scale matching those kind of scale changes can be observed in the matches



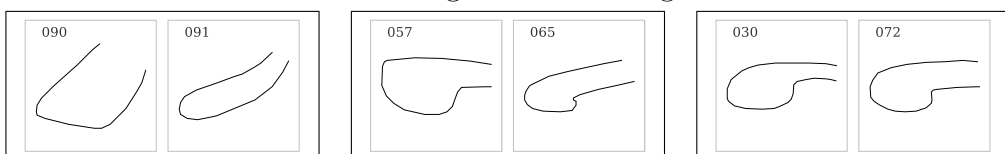
In the clustering of the low scale matching results it seems that the scale of the curve is often not overcome, but this could, again, be due to the way we choose the threshold for a cluster.

Some matches that might be undesirable are for example at low scale



What we have here are zero-crossings that are being ignored by the distance measure. Zero-crossings of the curvature occur at reflection points, where the curvature changes its sign and therefore its "turning direction". Those points are said to be highly perceptive. The curvature scale space, for example, relies entirely on zero-crossings of the curvature across different Gaussian scales. Our distance measure takes the integral of the curvature function and therefore high curvature points and their occurrence with respect to the parameter are more important than what happens near the zero-axis. This explains why those curves are being matched and we can identify it as a drawback of the distance measure.

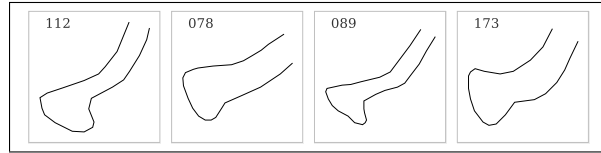
Some matches that occurred at high scale and might be undesirable are



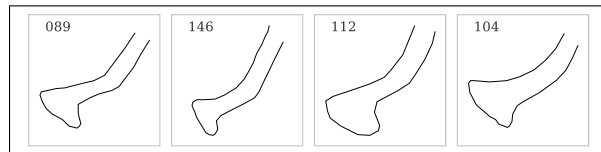
With 90-91 we have a classic example of what might happen, if we use the curvature function at too high a scale. The curvature change that leads to the second

corner of the first shape is registered only locally in the curvature function. It contributes high values to the integral, but only over a small range. In the other two matches we encounter a similar problem. A local change in the curvature propagates into a large difference in the absolute placement of the vertices. We conjecture that this drawback of the curvature function could be overcome if the shapes are matched at different scales simultaneously, as it is the central approach in scale space theory.

Let us draw the attention to one specific example, which is taken from the high scale matching results (best matches of 112),



At low scale we can find the similar matchings (best matches of 89)



In each of those curves the order of high curvature points, zero-crossings and their occurrence with respect to the parameter is very similar, which is why they yield a small distance. But why is it exactly, the reasons are not entirely the same for low and high scale. Their high scale curvature functions are being aligned according to the three high curvature points and the straight line segment at the ending, which makes the curvature tend to zero, see Figure 53(a). At low scale the high curvature points are smoothed, see Figures 53(c) and 53(d), only the ending as such stays as a dominant point. Here they are rather aligned according to long stretches in one turning direction at low scale. It seems that the width of the handle is indirectly being matched this way, which would be desirable, but also surprising.

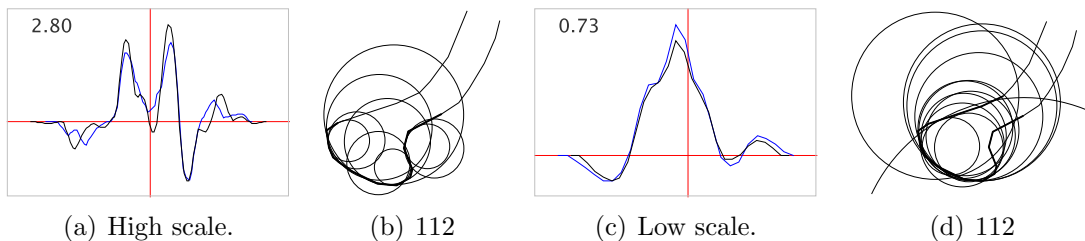


FIGURE 53. Match of 89 and 112 at high and low scale and some of the measured osculating circles. (The curvature functions are not according to scale.)

We conclude that high curvature points control the alignment of the curvature function at high scale, while long stretches that are consistent in the direction of the turning of the curve have a large influence at low scale. This is very interesting, since a similar effect was identified in the studies of the curvature scale space. There the problem of long shallow concavities lead to undesirable matchings and had to be corrected with an additional constraint [AMK00]. Of course this analogy is not completely correct, since we are comparing very different things. But it would be interesting to investigate it further.

7. CONCLUSIONS

We proposed a new theory of multi-scale curvature matching for open polygonal curves, that is yet incomplete. But first results are promising. The central hypothesis is taken from scale space theory and states that objects should be matched at different scales simultaneously. But we are not using Gaussian smoothing, which is the central operator there. Instead the smoothing is done through three point curvature approximations where the three points have variable distance to each other along the curve. Experiments show that this is more faithful to the shape of the curve since it does not implicitly shrink the curve at high curvature points. The proposed distance measure is invariant under rotation, translation, and reflection and we conjecture that it is also to a large extent invariant under scaling and it can be approximated efficiently with the proposed algorithm. It is a semi-metric, according to [VH99], since we are comparing the equivalence classes of curves that have the same curvature function.

It would be interesting to compute the prototypes of those equivalence classes, i.e. the parametrized curves, that have the respective curvature function. According to [Bel99] it is well known, that the curves that match our piecewise constant curvature functions are the class of wooden or mechanical splines. A further analysis of how the curvature smoothing changes the shape of the curve would hopefully also reveal more about possible undesirable matches that the distance measure produces. A list of other questions remain open.

- (1) Can we find examples where the extension to infinity is disadvantageous? A related observation is the one, that a variation of our distance measure for closed polygonal curves is not at all obvious.
- (2) What is the effect of non-uniform noise on the curvature across the scales? Non-uniform noise is a serious problem for any distance measure that uses the curvature, examples include the curvature scale space and the turning function. Exceptions are when the curvature is basically only used as a corner detector to register salient points.
- (3) Can we extend the distance measure to allow non-uniform scaling and affine transformations? The gradient descent algorithm scales easily to k dimensions. The question we have to solve first is the following. What is the effect of non-uniform scaling on the curvature and can it be modelled easily?
- (4) Can we solve the optimization problem exactly in a non-prohibitive time? Can we identify classes of curves such that number of cells of the domain of the objective function is small? It would also be interesting to examine self-similarities, and their effect on the objective function. We observed that they can lead to an objective function that is hard to approximate with the gradient descent method.
- (5) Can we incorporate corners and/or self-intersections into the distance measure? Those are also highly perceptive, but our distance measure does not account for them. Unless self-intersections have an effect on the curvature at some scale.
- (6) The title of this text states that our method is applicable to smooth polylines, but we have not rigorously defined, what is meant by that. Which class of polygonal curves does this method work for?

The last question is linked to the next thing we should do. We need to do more exhaustive experiments with other data to test the scale-invariance and the effect of noise and to compare the distance measure to existing measures.

INDEX

- arc length, 3
- boundary, 45
- convolution, 16
 - kernel, 16
- curvature, 3
 - center, 6
 - general formula, 5
 - invariance, 6
 - radius of, 6
 - sign of, 4
 - uniqueness, 8
- curve
 - polygonal, 9
 - smooth, 3
 - parameterized, 3
- discrete curvature
 - ϵ -parameter, 11
 - direct , 15
 - Gaussian direct, 17
 - osculating circle, 13
 - radius of , 13
 - sampling rate parameter, 11
 - scale of, 11
 - second derivative, 14
 - turning angle , 13
- discrete derivative, 14
- distance measure, 25
- intersection function, 32
- kernel size, 17
- metric, 26
- osculating circle, 5
- parametrization, 3
 - arc length, 3
- parameterized curve, 3
 - change of orientation, 3
 - trace, 3
- polygonal curve, 9
 - α , 10
 - edges, 9
 - point lies on, 9
 - smooth, 20
 - vertices, 9
- tangent, 3
- trajectory, 38
- trajectory range, 38

REFERENCES

- [AB86] H Asada and M Brady. The curvature primal sketch. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):2–14, 1986.
- [ACH⁺90] Esther M. Arkin, L. Paul Chew, David P. Huttenlocher, Klara Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 129–137, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.
- [AG96] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: matching, interpolation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, 1996.
- [AMK00] S. Abbasi, F. Mokhtarian, and J. Kittler. Enhancing css-based shape retrieval for objects with shallow concavities. *Image and Vision Computing*, 18(3):199–211, 2000.
- [Bel99] Alexander Belyaev. A note on invariant three-point curvature approximations. *Surikaiseikikenkyusho Kokyuroku (RIMS, Kyoto)*, 1111:157–164, 1999.
- [CG97] Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. In *In Proceedings of the 8th Annual Symposium on Discrete Algorithms*, pages 777–786, 1997.
- [CMT01] David Coeurjolly, Serge Miguet, and Laure Tougne. Discrete curvature based on osculating circle estimation. In *IWVF-4: Proceedings of the 4th International Workshop on Visual Form*, pages 303–312, London, UK, 2001. Springer-Verlag.
- [dC76] Manfredo P. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1976. Translated from the Portuguese.
- [DL03] Nira Dyn and David Levin. Subdivision schemes for geometric modelling. *Acta Numerica*, 11, 2003.
- [FF94] D. P. Fairney and P. T. Fairney. On the accuracy of point curvature estimators in a discrete environment. *Image and vision Computing*, 12(5):259–265, June 1994.
- [GJS07] Sumanta Guha, Paul Janecek, and Nguyen Duc Cong Song. Simplipoly: Curvature-based polygonal curve simplification. In *GRAPP (GM/R)*, pages 166–171, 2007.
- [Hil80] Ellen C. Hildreth. Theory of edge detection. 207:187–217, 1980.
- [IPE08] <http://tclab.kaist.ac.kr/ipe/>, the ipe extensible drawing editor released on the gnu gpl. Jan 26, 2008.
- [Jav08] <https://javacc.dev.java.net/> javacc is a parser/scanner generator for java, released under the bsd license. Jan 26, 2008.
- [JLI08] <http://jlibeps.sourceforge.net/> jlibeps - a java eps library released on the gnu gpl. Jan 26, 2008.
- [Lin08] Tony Lindeberg. *Wiley Encyclopedia of Computer Science and Engineering*, chapter Scale Space. John Wiley & Sons, 2008.
- [MB03] Farzin Mokhtarian and Miroslav Bober. *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.
- [MSK95] J. Matas, Z. Shao, and J. Kittler. Estimation of curvature and tangent direction by median filtered differencing. In *In 8th International Conference on Image Analysis and Processing*, pages 83–88, 1995.
- [RJ73] A. Rosenfeld and E.G. Johnston. Angle detection on digital curves. *TC*, 22:875–878, 1973.
- [Sny05] Jan A. Snyman. *Practical mathematical optimization*, volume 97 of *Applied Optimization*. Springer-Verlag, New York, 2005. An introduction to basic optimization theory and classical and new gradient-based algorithms.
- [Spi79] Michael Spivak. *A comprehensive introduction to differential geometry. Vol. II*. Publish or Perish Inc., Wilmington, Del., second edition, 1979.
- [Sun08] Dan Sunday. <http://softsurfer.com/algorithms.htm> geometry algorithms web site. Jan 26, 2008.
- [TC94] Du-Ming Tsai and Ming-Fong Chen. Curve fitting approach for tangent angle and curvature measurements. In *Pattern Recognition Volume 27, Issue 5*, pages 699–711, 1994.
- [Uml01] Georg Umlauf. Computing curvature bounds for bounded curvature subdivision. *Computer-Aided Geometric Design*, 18:455–461, 2001.

- [Utc03] Sven Utcke. Error-bounds on curvature estimation. In *Scale-Space*, pages 657–666, 2003.
- [VH99] Remco C. Veltkamp and Michiel Hagedoorn. State-of-the-art in shape matching. Technical report, Principles of Visual Information Retrieval, 1999.
- [VL06] Remco C. Veltkamp and Longin Jan Latecki. Properties and performances of shape similarity measures. In *Content-Based Retrieval*, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.
- [WS93] Marcel Worring and Arnold W. M. Smeulders. Digital curvature estimation. *CVGIP: Image Underst.*, 58(3):366–382, 1993.