

PRIP-TR-144

July 16, 2018

Students' SS 2018 contributions:

- (A) 183.151 Selected Chapters of Image Processing and
- (B) 186.837 Seminar of Computer Vision and Pattern Recognition

(A) *Christian Brändle, Bernhard Langer*

(B) *Bernhard Langer, Maximilian Langer,*

Julian Strohmayer, Anna Gostler,

Timon Höbert

edited by Walter G. Kropatsch and Jiří Hladůvka

Institute of Visual Computing and HCT

Pattern Recognition and Image Processing Group 193-03

Abstract

This technical report presents a collection of selected papers, submitted by students in the courses “Selected Chapters of Image Processing” (VU 183.151) and “Seminar of Computer Vision and Pattern Recognition” (SE 186.837) of the Pattern Recognition and Image Processing group during summer term 2018. “Selected Chapters of Image Processing” was organized as presentations followed by discussions lead by an opponent.

Contents

Christian Brändle	
2D-Warping of Space-Filling Curves	2
Bernhard Langer	
2D-Warping of Space Filling Curves – Discussion	18
Bernhard Langer	
Hyperbolic Medial Axis	20
Christian Brändle	
Hyperbolic Medial Axis – Discussion	31
Bernhard Langer	
Elliptic Voronoi Diagram	34
Maximilian Langer	
Artificial Video Dataset for Articulated Joint-Movement	44
Julian Strohmayer	
Improved Features for Hypocotyl/Root Transition Detection in Arabidopsis Thaliana	52
Anna Gostler	
Tracking Golden-Collared Manakins in the Wild	62
Timon Höbert	
Vascular Structure Skeletonization: A Survey	66

2D-Warping of Space-Filling Curves

Christian Brändle
TU-Wien, Technical University of Vienna
Wien, Karlsplatz 13, 1040 Wien

Abstract—We evaluate certain possibilities of warping *space-filling* curves to different 2D domains. The domains are described by their contours and are preprocessed to establish a topology equal to a disk. We compare different possibilities according to their accuracy and other qualities of the mappings. Discrete as well as continuous mapping schemes are evaluated.

I. INTRODUCTION

The goal of this review is to find a warping scheme between two 2D domains defined by their corresponding outer border. The actual warp should then transfer a *space-filling* curve from one domain to the other.

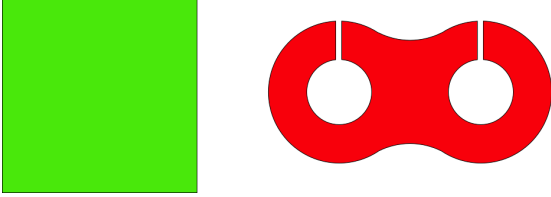


Figure 1. Target domain (red) is cut to be topologically equivalent to fixed source domain.

If a domain has holes, and as such more boundaries, it is cut in a way that only an outer boundary remains. That means we make the domain topologically equivalent to a disk. To make two 2D domains topologically equivalent to a disk we introduce cuts in the target domain. How to optimally cut a surface into a disk can be read for instance in [EHP04]. We just illustrate here a possible cut of such a domain, see Figure 1.

Further we want to define a mapping between the two domains that is continuous and bijective. The two illustrated patches, that are forming the domains respectively, are referenced to as the *domain* or *source domain* for the green patch on the left and *co-domain*, *target domain* or *image* of our mapping for the red patch on the right.

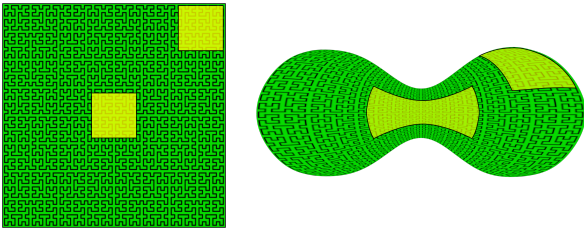


Figure 2. The received deformation of a local patch in the target domain can be significantly different depending on its location.

Further we want that local deformations, such as stretch in the surrounding of certain points of the domain and the co-domain, are as low as possible. This is easiest formulated by further constraints on the domain transformation as we will see later. However, as illustrated in Figure 2, we have to accept that local deformations in different parts of the mapping (domain or co-domain) are different and inhomogeneous in comparison to each other.

Our domain is fixed, it is the unit-square that contains a space-filling curve, like for instance a *Hilbert* curve. We will call this domain in short *unit-domain*. Once again our co-domain consists of a patch cut topologically to a disk that should contain the warped space-filling curve.

We define then a coarse model consisting of a quad-grid, a triangulation or any other model that captures certain data, like points, curves, areas and the like, from the unit-domain and transfers it into the co-domain.

The fine model consists of samplings on the space-filling curve itself, not limited on specific data points of the grid or tessellation used in the coarse model anymore. The reachable sampling points are continuous from their nature, as we can define a continuous parameter value on the curve, find a point in the unit-domain and map that point into the co-domain. This mapping is either established by certain interpolation schemes between data elements of the coarse model or it is defined directly if the established model captures both, the coarse and the fine representation.

II. SPACE-FILLING CURVES

Here we will only give a very brief description of space-filling curves and especially of the *Hilbert* curve to provide a setup for warping such curves from their constructive domain to an arbitrary co-domain in 2D.

A. Definitions

According to Bader et al. [Bad12] to describe a space-filling curve we first need to introduce a sequential order on a d -dimensional array of elements, a so called *mapping*. The mapping maps the range of array indices $\{1, \dots, n\}^d$ to sequential indices $\{1, \dots, n^d\}$ and vice versa.

To change from the *discrete* to a *continuous* mapping the sequential index set becomes a parameter interval, for instance the one dimensional unit interval $[0, 1]$ or the multi-dimensional unit interval $[0, 1]^d$.

We can define a *curve* then

Definition II.1. Curve. We can define a curve $f : I \rightarrow \mathbb{R}^n$ as a continuous mapping of a compact set $I \subset \mathbb{R}$ into \mathbb{R}^n .

Usually I is the unit interval $[0, 1]$ and we want to find a curve that visits each point in the unit square without self intersections. Such a continuous and surjective curve is called a *space-filling curve*. It cannot be injective as several parameter points can be mapped to the same image point.

B. Graphical Construction of Space-Filling Curves

As a space-filling curve is a sort of fractal curve it is based on self-similar elements. To construct such a curve we can rely on a graphical iteration scheme that will be illustrated with a certain curve, the *Hilbert curve*. It is a recursive procedure, that resembles a quadtree generation, see Bader et al. [Bad12]:

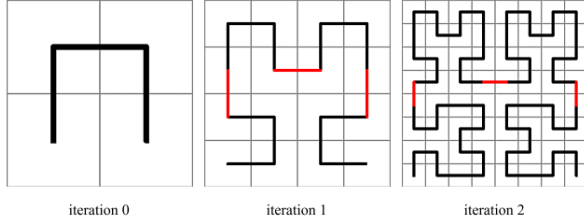


Figure 3. The first three iterations of the Hilbert curve [mit].

- The square target domain, starting with the unit square, is subdivided into four subsquares.
- A space-filling curve is found for each subsquare. It is obtained as a scaled-down, rotated or reflected version of the original curve.
- The reflection and rotation is chosen in such a way that the four partial curves can be connected.

From each iteration to the next all existing subsquares are subdivided and connected by a pattern that is obtained by rotation and/or reflection of the *fundamental pattern*. From each iteration to the next, four copies of existing iterations are connected. The copies are rotated and reflected such that their start and end points can be connected. This is illustrated for the first iterations in Figure 3. Encoding of the curve could also be done with the *RULI chain code* [FK94].

C. Evaluation of Space-Filling Curves

The finite iterations provide an approximate solution to the Hilbert curve. But we want to compute the corresponding Hilbert mapping, that means to find an image point to the corresponding parameter value of the curve.

If we think of the basic properties of the Hilbert curve we see that a given parameter is approximated by *nested intervals*. Each interval is one of the four quarters of its predecessors, starting with the parameter interval I . Further, the target is approximated by nested subsquares. Each subsquare is a scaled, transposed and rotated Hilbert curve section, see Bader et al. [Bad12].

We consider the boundaries of nested intervals in *quaternary* representation:

$$\begin{bmatrix} 0, & \frac{1}{4} \\ \frac{2}{4}, & \frac{3}{4} \end{bmatrix} = [0_4.0, 0_4.1], \quad \begin{bmatrix} \frac{1}{4}, & \frac{2}{4} \\ \frac{3}{4}, & 1 \end{bmatrix} = [0_4.1, 0_4.2], \quad (1)$$

$$\begin{bmatrix} 0, & \frac{1}{4} \\ \frac{2}{4}, & \frac{3}{4} \end{bmatrix} = [0_4.2, 0_4.3], \quad \begin{bmatrix} \frac{1}{4}, & \frac{2}{4} \\ \frac{3}{4}, & 1 \end{bmatrix} = [0_4.3, 0_4.0],$$

When we construct the Hilbert curve, each of the four identical subcurves is filled in one of the four subsquares. To map the parameter of the curve in the 2D domain we have to compute its relative position in the respective subsquare and repeat till we get the desired resolution.

For each subsquare we require a transformation operator that maps the unit square in the correct subsquare by operators H_0, H_1, H_2 and H_3 . The quaternary digits tells us the intervals and therefore the subsquare as well as the operator that has to be applied, see Bader et al. [Bad12].

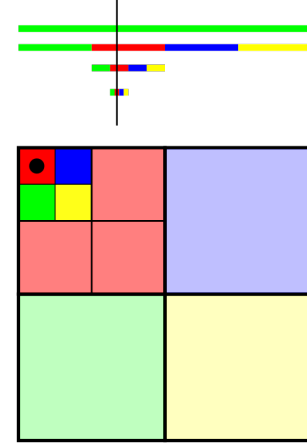


Figure 4. Detection of subsquare corresponding to given interval.

The operators H_i first scale and then performs a combination rotations, reflections and translations.

- H_0 performs reflection on the main diagonal
- H_1 translates by $\frac{1}{2}$ in y-direction
- H_2 translates by $\frac{1}{2}$ in both x- and y-direction
- H_3 performs a reflection in both x- and y-direction

The function value $h(t)$ of the Hilbert curve is then calculated via three steps:

- Compute the quaternary representation of parameter $t = 0_4.q_1q_2q_3q_4 \dots$. If the parameter lies in the q_1 -th interval $h(t)$ is in the q_1 -th subsquare.
- In the q_1 -th subsquare parameter t corresponds to local parameter $\tilde{t} = 0_4.q_2q_3q_4 \dots$. We compute $h(\tilde{t})$, which is the scaled down rotated Hilbert curve in the q_1 -th subsquare.
- Transform the local $h(\tilde{t})$ into the original square by applying operator H_{q_1} to $h(\tilde{t})$.

Repeated in a recursive fashion we can calculate the 2D-coordinates of a point on the Hilbert curve with respect to it's 1D parameter value t with arbitrary precision. Figure 4 illustrates the tracking of the corresponding subsquare of an example interval.

III. WARPING

Given a source image I and the correspondence between the original position $p_i = (u_i, v_i)^T$ of a point in I and its

desired new position $q_i = (x_i, y_i)^T$ for indices $i = 1, \dots, n$, generate a warped image I' such that $I'(q_i) = I(p_i)$ for each point i . Here I and I' represent the intensities, colors or any arbitrary data of the images, see Kheng et al. [Khe08].

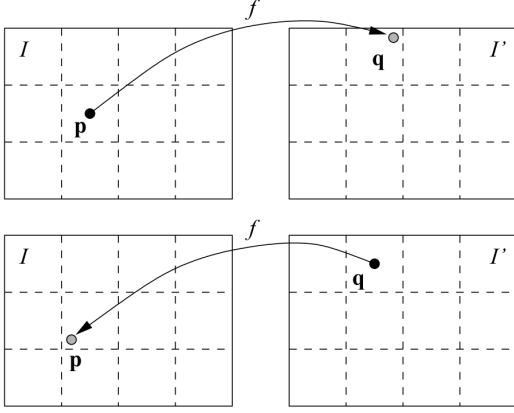


Figure 5. Forward and backward mapping between domain and co-domain [Khe08].

There are two different strategies, namely *forward mapping* and *backward mapping* that map a data point from the domain to the co-domain or vice versa, see Figure 5.

While in continuous space it doesn't make a difference in which direction we transform, in discrete space this is an issue. With forward mapping, if we start from a grid point in the source domain, we won't hit an grid point in the target domain exactly. With backward mapping it is the same issue in the other direction. For digital images we prefer a backward mapping, as sampling in the target domain can be done in desired resolution and corresponding interpolation is done in the source domain.

IV. POINT-BASED WARPING

In a point-based warping scheme we identify so called corresponding *landmarks* \bar{c}_i in the source and \bar{c}'_i in the target domain that are used for alignment. Data points in between are interpolated in a certain fashion according to the warping scheme applied. The mapping of the landmarks themselves can be seen strict or soft depending how important other constraints, like smoothness of transition, are in comparison.

The basic formulation is a *polynomial transformation*, in generic form [Khe08]

$$u = \sum_k \sum_l a_{kl} x^k y^l \quad (2)$$

$$v = \sum_k \sum_l b_{kl} x^k y^l \quad (3)$$

The functional representation $f(\bar{x}) = f(\bar{c})$ of the Transformation T with k parameters $\beta = (\beta_1, \beta_2, \dots, \beta_k)$ is an ordinary system of linear equations where we search a β such that

$$T(\beta, \bar{c}_i) = \bar{c}'_i; i = 1, \dots, N \quad (4)$$

is satisfied. If the system of equations is overconstrained ($K < 2N$) then we solve for the minimum squared error

$$\arg \min_{\beta} \left[\sum_{i=1}^N (\bar{c}'_i - T(\beta, \bar{c}_i))^2 \right] \quad (5)$$

instead. If the system is underconstrained, we can impose some extra constraints and find a regularization and a solution that optimizes for that criterion.

$$\begin{pmatrix} 1 & c_{x,1} & c_{y,1} & c_{x,1}c_{y,1} & c_{x,1}^2 & c_{y,1}^2 \\ 1 & c_{x,2} & c_{y,2} & c_{x,2}c_{y,2} & c_{x,2}^2 & c_{y,2}^2 \\ & & & \vdots & & \\ 1 & c_{x,N} & c_{y,N} & c_{x,N}c_{y,N} & c_{x,N}^2 & c_{y,N}^2 \end{pmatrix} \begin{pmatrix} \beta_{00} \\ \beta_{10} \\ \beta_{01} \\ \beta_{11} \\ \beta_{20} \\ \beta_{02} \end{pmatrix} = \begin{pmatrix} c'_{x,1} \\ c'_{x,2} \\ \vdots \\ c'_{x,N} \\ c'_{y,1} \\ \vdots \\ c'_{y,N} \end{pmatrix}$$

Figure 6. Polynomial Transformation for k parameters β [Whi17].

If polynomials are used, then the constraint points c_i are used with their respective powers $c_{i,j}, c_{i,j}^2, \dots, c_{i,j}^k$ which leads to a transition matrix T that is not well conditioned, see Figure 6.

Because of that an inversion of the matrix with an ordinary *pseudoinverse*, like in equation (7), is better replaced by a *singular value decomposition*, like in equation (10).

$$T^T T \beta = T^T c \quad (6)$$

$$\beta = (T^T T)^{-1} T^T c \quad (7)$$

$$T = U W V^T \quad (8)$$

$$T^{-1} = V W^{-1} U^T \quad (9)$$

$$\beta = T^{-1} c \quad (10)$$

A. Radial Basis Function - based Warping

One possibility of warping between two different sets of landmarks is given by *radial basis functions*, see Whitaker et al. [Whi17]. The RBF formulation is a *polynomial transformation* where the functional representation $f(\bar{x})$ is a weighted sum of basis functions

$$f(\bar{x}) = \sum_{i=1}^N k_i \Phi_i(\bar{x}) \quad (11)$$

$$\Phi_i(\bar{x}) = \Phi(|\bar{x} - \bar{x}_i|) \quad (12)$$

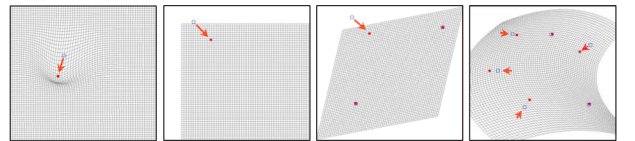


Figure 7. Comparison result between different polynomial degree about moving the control-points. From left to right: without polynomial, with constant, with linear, with quadratic [KLYL09].

As a sum of radial basis functions cannot capture an affine transformation of control points exactly we have to add an additional polynomial to capture such transformations. Further the polynomial is used to relax the exact mapping of controlpoints in favor of a lower bend energy on the domain, see Arad et al. [ADRY94]

$$f(\bar{x}) = \sum_{i=1}^N k_i \Phi_i(\bar{x}) + p_2 y + p_1 x + p_0 \quad (13)$$

$$(14)$$

We then define $T^x(\bar{x})$ and $T^y(\bar{x})$ respectively

$$T^x(\bar{x}) = \sum_{i=1}^N k_i^x \Phi_i(\bar{x}) + p_2^x y + p_1^x x + p_0^x \quad (15)$$

$$T^y(\bar{x}) = \sum_{i=1}^N k_i^y \Phi_i(\bar{x}) + p_2^y y + p_1^y x + p_0^y \quad (16)$$

$$(17)$$

Finally, we impose constraints and solve the linear system to find k' 's and p' 's to fit the data points

$$T^x(\bar{x}_i) = x'_i \quad T^y(\bar{x}_i) = y'_i \quad (18)$$

$$\sum_{i=1}^N k_i^x = 0 \quad \sum_{i=1}^N k_i^y = 0 \quad (19)$$

$$\sum_{i=1}^N k_i^x \bar{x}_i = 0 \quad \sum_{i=1}^N k_i^y \bar{x}_i = 0 \quad (20)$$

$$\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} k_1^x \\ k_2^x \\ \vdots \\ k_N^x \\ k_1^y \\ k_2^y \\ \vdots \\ k_N^y \end{pmatrix} = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_N \\ y'_1 \\ y'_2 \\ \vdots \\ y'_N \end{pmatrix} \quad B = \begin{pmatrix} \phi_1(\bar{x}_1) & \phi_2(\bar{x}_1) & \dots & \phi_N(\bar{x}_1) \\ \phi_1(\bar{x}_2) & \phi_2(\bar{x}_2) & \dots & \phi_N(\bar{x}_2) \\ \vdots & \vdots & \dots & \vdots \\ \phi_1(\bar{x}_N) & \phi_2(\bar{x}_N) & \dots & \phi_N(\bar{x}_N) \end{pmatrix}$$

Figure 8. Solve RBFs for landmarks as constraints [Whi17]

The resulting equation system is seen in Figure 8. The effect of using different polynomials in addition to the radial basis function can be seen in Figure 7 of Kwon et al. [KLYL09]. Finally we can use different *kernels* in the RBF like *Gaussian* kernels or *thin plate spline* kernels, see equation (21).

$$\Phi_i(x) = |x - x_i|^2 \lg(|x - x_i|) \quad (21)$$

The problem with this deformation is that the shape undergoes undesirable local non-uniform scaling and shearing, see Schaefer et al. [SMW06].

V. MESH-BASED WARPING

Here we will discover certain possibilities to generate a mapping from our domain in our co-domain with different meshing algorithms for the given domain. By their nature the algorithms discretize our domain and co-domain in such a fashion, that certain points on the boundary as well as inside the domain are picked to represent vertices in the corresponding meshes.

For the mapping itself we explore different strategies. We can find a parametrization that transforms our domain in our co-domain or we can find an inverse parametrization that does the opposite.

In either case we will need to do an interpolation of points that are not in the set of selected domain or co-domain points.

For the first strategy we take a look at meshings out of *self organizing maps* [Koh98], for the latter we will explore the *stretch-minimization parametrization* [YBS04].

A. Co-domain reconstruction with Self-Organizing Maps

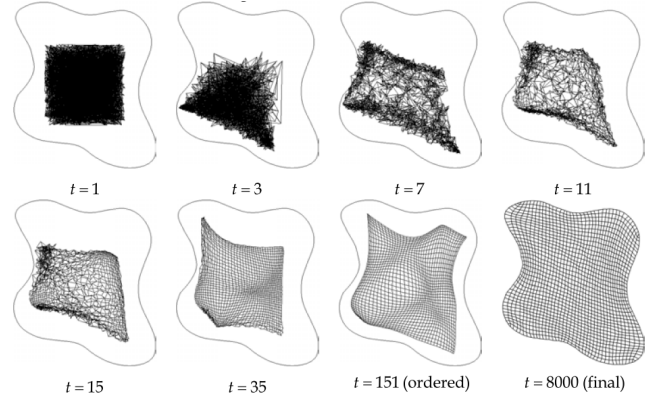


Figure 9. The ordering stage of SOM learning and final mesh in 2D case [Nec10].

This forward-mapping is based on the idea that the grid that covers our domain can be directly transformed in such a way that it covers our co-domain. Figure 9 illustrates this.

According to Nechaeva et al. [Nec10] we define the co-domain as G in Euclidean space R_G^n with coordinates $x = (x^1, \dots, x^n)$ and the domain as Q , also known as computational domain, in Euclidean space R_Q^k with coordinates $q = (q^1, \dots, q^k)$, where in general $k \leq n$ is required, but in our case it is identical, i.e. $k = n$.

The domain is represented by our fixed rectangular shape and the mesh $Q_N = \{q_1, \dots, q_N\}$ is fixed over Q , defining a four-connected grid. So the mesh Q is fixed and uniform with an equal distance d_Q between all pairs of nodes in Q .

The adaptive mesh $G_N = \{x_1, \dots, x_N\}$ is constructed over G , which will be our desired mapping.

Let $B_\gamma(q)$ define the neighborhood around point q with radius γ . Then the goal is to find a mapping from Q onto G that transforms Q_N into G_N .

The SOM model $\langle M, H, Alg \rangle$ consists of a map of neurons $M = \{e_1, \dots, e_N\}$, a training set H and a learning

algorithm *Alg*. The single neuron $e_i = (q_i, x_i)$ consists of the vertices in the domain and co-domain respectively.

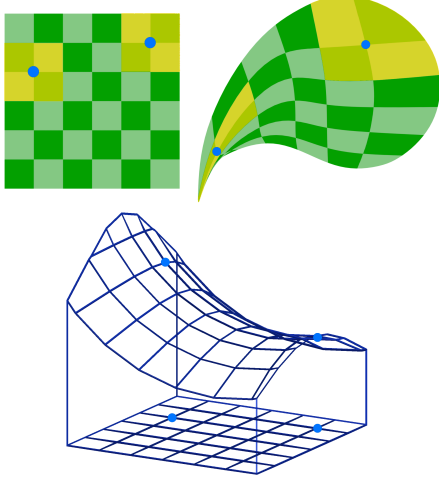


Figure 10. Corresponding weight distribution to fulfill the equidistribution principle, i.e. $\forall a, b : w_a A_a = w_b A_b$ where $a, b \in [1, \dots, N]$ and $a \neq b$.

The desired density of the adaptive mesh G is defined by a *mesh density function* $w : G \rightarrow R^+$, which is based on the equidistribution principle, which states that all the products of each cell area A_i and its corresponding weight w_i should be equal, see Figure 10 for illustration.

To obtain the desired distribution of x_i over G that is based on an equidistribution principle, we have to generate a training set $H = \{y_1, \dots, y_T\}$ that is generated by sampling a *probability distribution* $p(x)$ equal to the normalized mesh density function $w(x)$. Here T is the size of the set and $y_i \in G$:

$$p(x) = \frac{w(x)}{\int_G w(x) dx} \quad (22)$$

The process of 'stretching out' the mesh in the co-domain is as follows:

- 1) Set arbitrary initial locations of mesh nodes $x_i(0)$ for all $i = 1, \dots, N$.
- 2) Repeat the following operations at each iteration $t = t_{st}, \dots, t_{fin}$:
- 3) a) Take the next vector y_t from the training set H .
- b) Calculate the Euclidean distances $d(\cdot, \cdot)$ between y_t and all nodes $x_i(t)$ and choose the *winner* node $x_m(t)$ which is closest to y_t

$$m = m(y_t) = \arg \min_{i=1, \dots, N} d(y_t, x_i(t)) \quad (23)$$

- c) Adjust locations of mesh nodes using the following rule:

$$x_i(t+1) = x_i(t) + \theta_{q_m}(t, q_i)(y_t - x_i(t)) \quad (24)$$

- 4) adjust weights w_i according to the new cell areas following the equidistribution principle

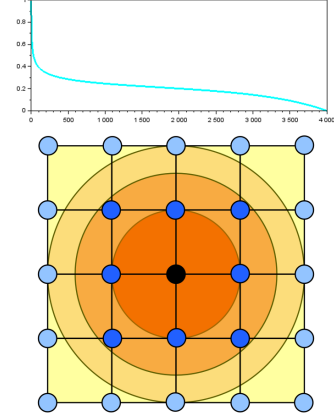


Figure 11. Illustration of learning step function $\delta(t)$ (top) and local neighborhood function $\eta_{q_m}(t, q_i)$ (bottom).

The term $\theta_{q_m}(t, q_i)$ is the so called *learning rate*. It lies between $[0, 1]$ and is the product of the *learning step function* $\delta(t)$ and *local neighborhood function* $\eta_{q_m}(t, q_i)$.

$$\theta_{q_m}(t, q_i) = \delta(t) \eta_{q_m}(t, q_i) \quad (25)$$

where $\delta(t)$ of equation (29) is a time-depended decreasing function and $\eta_{q_m}(t, q_i)$ of equation (31) defines the strength of lateral connections between neuron q_m and q_i , see Figure 11 for illustration.

$$t = 1, \dots, T \quad (26)$$

$$\chi(t) = 1 - e^{5(t-T)/T} \quad (27)$$

$$\delta(t) = t^{-0.2} \chi(t) \quad (28)$$

$$(29)$$

$$r(t) = r(T) + \chi(t)(r(1)0.05^{t/T} - r(T))t^{-0.25} \quad (30)$$

$$\eta_{q_m}(t, q_i) = s^{\left(\frac{d(q_m, q_i)}{r(t)}\right)^2} \quad (31)$$

The *winner* neuron will take the maximum displacement $\Delta x \in G$ at each iteration. The greater the distance between a neuron and the winner in the computational domain Q , the less the displacement of this neuron within the physical domain G . When s is small, only neurons in range of $r(t)$ need to be concerned.

The values of $r(1)$ and $r(T)$ are selected in such a fashion that at the first iteration all neurons in the neighborhood q_i will be affected, whereas at the last iteration T only the closest neighbors of q_i are affected.

As we transform a quad-mesh in our co-domain, we have to define how we transform points that are not corresponding to grid nodes. Possibilities ranges from normal bilinear interpolation to more smooth bicubic interpolation [Key81].

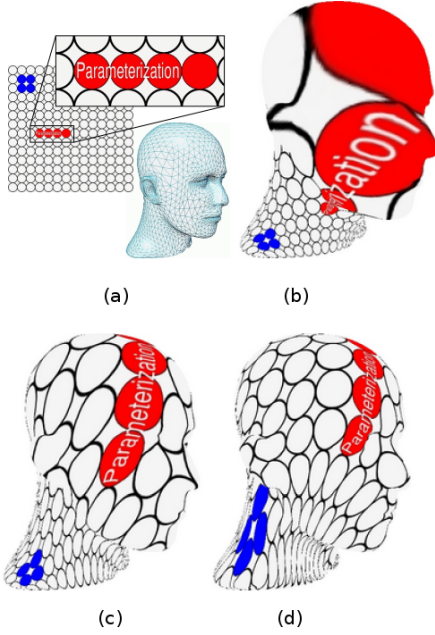


Figure 12. From left to right: 2D mapping domain and model (a), initial barycentric parameterization (b), single step optimization result (c), optimal low-stretch parameterization (d). [YBS04].

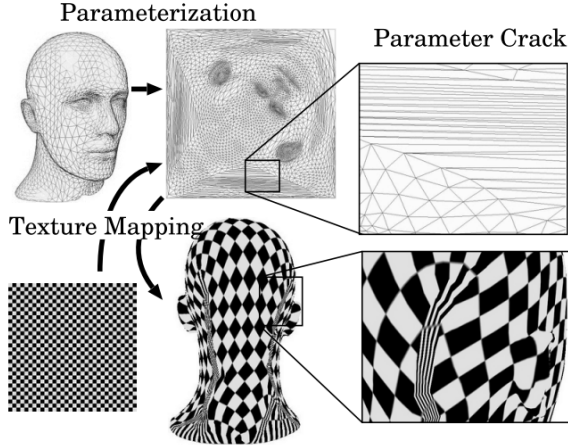


Figure 13. Parameter cracks on textured Mannequin Head model without regularization term [YBS04].

B. Domain reconstruction with Stretch-Minimization Parameterization

Another possibility to represent our warp is with a parametrization. In peculiar we want a parametrization that preserves length in a mapping. Yoshizawa et al. [YBS04] does this via an iterative approach. Starting from a *barycentric* parametrization on a *convex* boundary (i.e. [Flo97]) the parametrization is improved gradually. The barycentric parametrization is defined by minimizing the energy between neighbor vertices u_i and u_j

$$E(u_i) = \sum_j w_{ij} (u_j - u_i)^2 \quad (32)$$

with fixed vertices on the convex outer boundary by solving for

$$\sum_j w_{ij} (u_j - u_i) = 0 \quad (33)$$

The stretch $\sigma(U)$ of a single triangle T in *mesh* space transformed to triangle U in *parameter* space is defined by the minimal and maximal *Eigenvalues* of the transformation

$$\sigma(U) = \sqrt{(\Gamma^2 + \gamma^2)/2} \quad (34)$$

and the stretch of vertex u_i in parameter space depends on the stretch and area $A(T_j)$ of its neighbor triangles

$$\sigma_i = \sqrt{\sum A(T_j) \sigma(U_j)^2 / \sum A(T_j)} \quad (35)$$

The parametrization is optimized iteratively by minimizing the weighted quadratic energy function of equation (32), where the weights w_{ij} are chosen in order to minimize the parametric stretch. This is done by redistributing the local stretches with new weights

$$w_{ij}^{new} = w_{ij}^{old} / \sigma_j \quad (36)$$

and solving for new positions of vertices $\{u_i\}$ at each iteration. We can see the vertices $\{u_i\}$ and corresponding energies in terms of a *mass-spring* system. For area preservation, if a high (low) stretch is observed at u_i , that is $\sigma_i > 1$ ($\sigma_i < 1$), we relax (strengthen) the springs connected with u_i by solving for new weights.

Given the boundary is convex the parametrization guarantees that no triangle flips occur in the mapping. This provides a continuous mapping domain, see Figure 12. To avoid so called parameter cracks in the parametrization, as shown in Figure 13, an additional regularization term is added to the stretch energy.

As our goal is to map from a rectangular domain to a topologically equivalent co-domain the stretch minimization parametrization supplies a good choice. Again we have to think about the interpolation between the vertices in the case that a transformed point does not match a vertex. Here we could use barycentric interpolation as well as higher order or generalized barycentric interpolation to reach a more soft transition between neighboring triangles, see for instance Langer et al. [LS08] and Floater et al. [Flo15].

VI. CONTOUR-BASED WARPING

The idea here is to warp objects with arbitrary shapes seeking contours of correspondence. Figure 14 illustrates this.

The advantage of this idea is that in principle this can be done in continuous space, which solves interpolation issues of discretized mappings and gives a precise transformation of arbitrary points in the domain according to the underlying transformation concept.

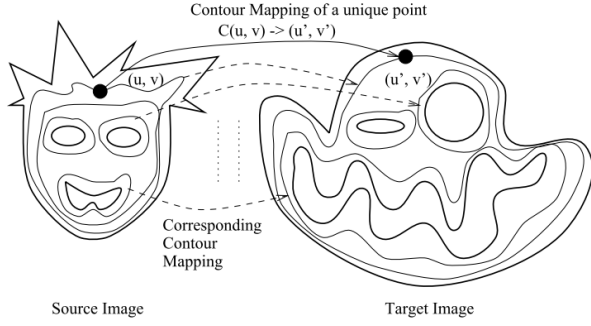


Figure 14. Illustration of contour mapping to warp 2D objects [CL98].

We want to establish a mathematical model that map each point of the source to a corresponding point in the target domain. In the continuous domain we want to transfer a point on a specific contour curve C_i to the corresponding point on the corresponding curve C'_i in the target domain. The curves themselves are continuous parametric functions that map their parameter value within range $[0, 1)$ to a corresponding point on the curve in \mathbb{R}^2 .

So we have a (possibly infinite) set of curve pairs that are bound together, with a designated starting point in each closed curve

$$\{(C_1, C'_1), (C_2, C'_2), \dots, (C_n, C'_n)\} \quad (37)$$

The *level set method* [OF01] can be used in continuous space to define a function that generates these curves with arbitrary distance from the border. To generate the set of curves in discrete space we can for instance utilize *mathematical morphology* [Ser83].

In either case we want a function $(u', v') = \text{warp}(u, v)$ such that $(u', v') \in C'_i \rightarrow (u, v) \in C_i$, where $i \in [1, n]$.

With a given *morphological operation* we define a relation between the image object I and a *structuring element* S . The structuring element is applied to each spatial point of the object with the specified relation.

Here we want to use the two specified operations *erosion* and *dilation*, which we will define next.

We consider an image as a function $F : E \rightarrow T$, where E is the set of image points and T the set of possible values of F and denote the set of functions from E to T by T^E . To align binary images and graylevel images we define a threshold of F at level t with:

$$X_t(F) = \{p \in E | F(p) \leq t\} \quad (38)$$

to decouple the direct value of the corresponding image point from further processing. Then we define *structuring elements* B as special sets known a priori. Their shape can for instance be a cross, like in the middle of Figure 15. Other popular shapes are stars, squares and lines assembled out of useful cell values like 0 or 1 in the binary case. Erosion is defined in equation (39) and dilation is defined in equation

(41), in the case of binary images we can relate to the lower simpler definition, see [NT13][CL98].

$$\epsilon_B(X) = X \oplus B = \bigcap_{b \in B} X_{-b} \quad (39)$$

$$= \{p \in E | B_p \subseteq X\}$$

$$\epsilon_B(E) = \{x | (x + b) \in E, \forall b \in B\} \quad (40)$$

$$\delta_B(X) = X \oplus B = \bigcup_{b \in B} X_b = \bigcup_{x \in X} B_x \quad (41)$$

$$= \{x + b | x \in X, b \in B\}$$

$$\delta_B(E) = \{x | x = i + b, \forall i \in E, \forall b \in B\} \quad (42)$$

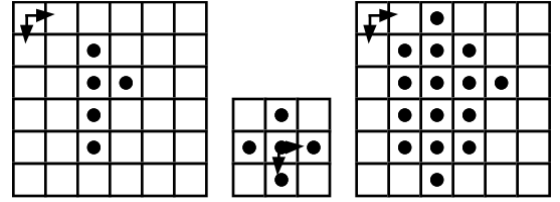


Figure 15. Applied dilation on binary image with given structuring element. From left to right: binary source image, structuring element and destination image [BMT94].

Illustrated with a binary image we can imagine that the structuring element hovers over the original image. Everywhere where the defined origin of the structuring element overlaps with the corresponding pixel in the image, we will apply the desired operation on the neighborhood defined by the structuring element and update the value of the pixel in our destination image at the current location of the origin of the structuring element.

In the simple case of a binary image for dilation we just search if any active pixel in the structuring element overlaps with any active pixel the defined neighborhood of the source image. If this is the case we set the pixel at the current position of the hovering structure element in the destination image active.

For erosion the procedure is approximately the reverse. Here all active pixels of the structuring element must overlap with active pixels of the source image. For situations like the discrete binary example in Figure 15 the binary erosion can be seen as the dual to binary dilation, while this is not true in the general case.

We will stick now to the discrete interpretation and define the *Peel-and-Resample* method of Chan et al. [CL98].

The idea is simple, namely to adapt the steps of a skeletonization process to produce the searched corresponding contours in source and target domain. The image is partitioned in layers derived from the steps of skeletal thinning similar to peeling an object. It is a three pass process where the found layer, i.e contour, is transformed to a rectangular, i.e.

normalized, form. Then this rectangular shape is transformed back in the output space and finally to the output image.

We can relate here the continuous case again with the discrete one.

$$p' = \text{Erode}(p) = E_d^{-1}(R_d^{-1}(R_s(E_s(p)))); \quad (43)$$

discrete case

$$p' = C'_i(C_i^{-1}(p)); \quad (44)$$

continuous case

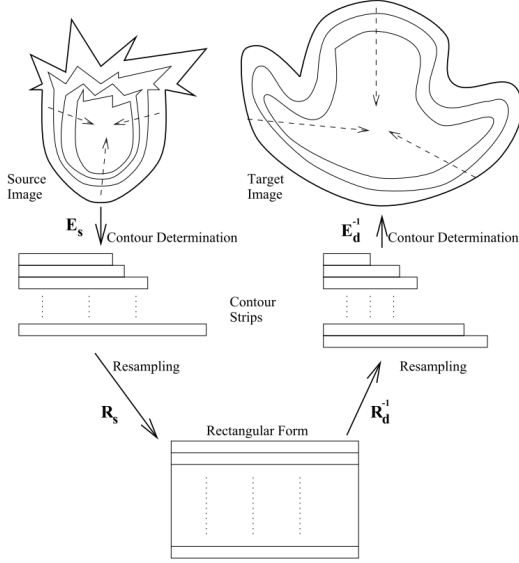


Figure 16. Peel and Resample Method [CL98].

where $p = (u, v) = C_i(t)$ defines a parametric curve in source domain and $p = (u, v) = C'_i(t)$ a curve in target domain respectively. Figure 16 illustrates this for the discrete case.

The process of skeletonization can be done in various ways. For instance we can use the *medial axis transform* [Blu67].

The distance transform measures the distance of each point from the nearest boundary. There are several ways of calculating approximations including *thinning*, *Voronoi diagram*, *distance field* and *hybrid methods* [XT09].

A very basic approach is the so called *grassfire transform* [Blu67] that is a morphological approach, which is described in subsection VI-A. The shape of the structuring element determines the distance metric that is applied, see Bailey et al. [Bai04] for details on different metrics like L_1 , L_2 or L_∞ . A hybrid metric can be approximated by altering different structuring elements in successive iterations. Better however is to use grayscale morphology and bigger structuring elements. The downside is the higher runtime complexity when the size of the structuring element increases and the fact that structures smaller than the size of the structuring element will be overseen. To reduce the computational cost we can exploit the fact that a structuring element can be decomposed

in several smaller structuring elements. When successively applied these chain of structuring elements has the same effect as the original one, but computation time can be reduced significantly.

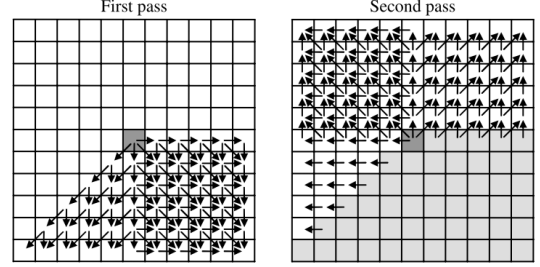


Figure 17. Propagation of distances for a single center pixel. Arrows indicate where the minimum value come from. If the true minimal value path cannot be covered by those chained arrows the determined minimum will deviate from the real minimum.

The *chamfer distance transform* [Bor86] replaces the iterations of successively applied morphological filters with two passes through the image. This is based on the observation that distances are calculated only by propagation of distances of the neighbor pixels. According to Bailey et al. [Bai04] the first pass is from top-left to bottom-right corner and uses only values that are correctly initialized or calculated from the upper left part of the transition window to propagate values. The second pass is from bottom-right to top-left and use only the lower right part of the transition window. It has to be mentioned that we can get an incorrect minimum distance in certain situations. Image the propagation fronts caused by a single pixel in the center of an image like shown in figure 17. As we can see in the bottom left octant there is less redundancy for passing information from neighbor pixels than in other octants. Bailey et al. [Bai04] notes that this lack of redundancy means that every pixel on the propagation path must be closer to the original background point than any other background pixel. If not, we will retrieve an incorrect minimum distance. Situations like this can arise easily in figures of certain shape like for instance spirals. The interested reader is referred to Bailey et al. [Bai04] or Borgefors et al. [Bor86] for more details.

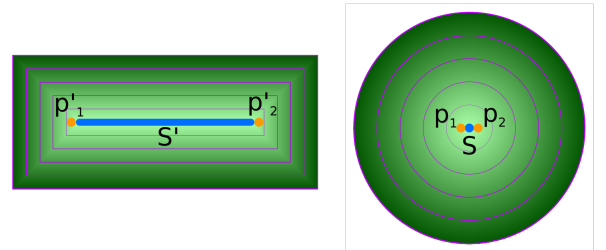


Figure 18. Violation of locality between pairs of points in vicinity of skeleton S and S' . The points p_1 and p_2 in the source domain are locally close but the mapped points p'_1 and p'_2 in the target domain are far from each other.

But this kind of skeletonization method suffers from the

fact, that it can easily violate the property of locality, namely that close points in the domain stay close in the co-domain. Imagine two skeletons S and S' , where the one can be reduced to a point s_1 , whereas the other forms a long line (or another structure) made out of a set of s'_i . A point p_1 mapped close to the point s_1 has a good chance to be mapped to a point p'_1 close to the point s'_1 at the one end of the line whereas another point p_2 close to s_1 could be easily mapped to a point p'_2 close to the point s'_2 at the other end of the line, see Figure 18 for explanation.

So it is maybe more desirable either to have skeletons of similar structure and extent or to use another object to which the contours should converge. As mentioned earlier we could just use a single point inside of each object and let the curves in both domains evolve to that single point. In such a case we can expect that local points stay more local and that undesired artifacts around both skeletons disappear. Because of that we will refer to the *level set method* [OS88] that can be used for such a purpose.

A. Morphological Contour Interpolation

A discrete method that follows a similar idea but generalizes the process of finding intermediate contour lines can be found for instance in Barrett et al. [BMT94]. It interpolates between an outer and a definable inner contour. As the innermost contour can be defined freely, we can rely on contours that are surrounding skeletons as well as on innermost contours that are constituted by just a single point, see again figure 18 for explanation. Finding interpolated contours is based on medial axis transform. For our purpose, we imagine that our outline forms a three dimensional surface that is subject to reconstruction.

This discrete image space algorithm depends on the previously defined morphological transforms named erosion and dilation and interimage operations XOR and union. Upon initial registration it is possible to interpolate between nested, overlapping, non overlapping or branching contours. The contour values can be generated simultaneously due to the nature of the morphological operators and result in a generated elevation map.

Definition VI.1. Isocontour. In 3D, an isocontour here is a 2D contour embedded into an intersection plane in 3D space. The plane itself intersects a 3D surface which yields the contour. Normally this plane is perpendicular to the z-axis and the surface is closed or has openings aligned with the slicing plane. This yields closed contours in the regular case. While isocontours can rely on more general definitions we relate here to the one given.

The process describes a search of *isocontours* in the provided dataset. The isocontours are extracted on a given height in the 3D elevation map and are used as contours in our 2D domain without their height component. The problem of finding a set of contour points, i.e. contours, at the same level of height and the problem of contour interpolation due to topological changes and/or overlap will be addressed by the

medial line generation, which is closely related to the *medial axis*.

Definition VI.2. Medial Line. We define the medial line or midline C' to be the medial axis C without any spines [BMT94].

The key idea is to find the midline between two contours and use it to split the intercontour space into two halves. This is repeated recursively till the intercontour space is exhausted. The algorithm

- can handle any number of contours of any shape including branching or overlapping geometries
- respect local shape by erosion and dilation as well as global shape through midlines
- can be performed in parallel in image space.

We use binary dilation r_b and erosion s_b with X as binary object and B as structuring element. Similar to convolution kernel we describe B_x as the translation of B by x .

The binary dilation of X by B is done by passing B over X and ORing B to the output image whenever the origin of B is over a pixel $x \in X$

$$Xr_bB = \{x | B_x \cap X \neq \emptyset\} \quad (45)$$

The complementary binary erosion is the dual of the binary dilation

$$Xs_bB = \{x | B_x \subset X\} \quad (46)$$

But as we want to establish a height map that can capture interpolation values we need a grayscale erosion and dilation which are defined by minima and maxima over grayscale image pixels I covered by a grayscale structuring element G_x .

The grayscale dilation is defined as

$$I r_g G = \{x | x = \max_{z \in G_x \cap I} (I(z) + G_X(z))\} \quad (47)$$

and the grayscale erosion as

$$I s_g G = \{x | x = \min_{z \in G_x \cap I} (I(z) - G_X(z))\} \quad (48)$$

In the trivial case the values of G are zero and add no bias.

We want to generate a medial line here, which is a modification of a *medial axis* without the spine, see Figure 19. According to Blum et al. [Blu73] the medial axis of a space-filling region R in 2D is the locus of all points $x \in R$ that have two or more points on the boundary of R which are equidistant from x . In our case, the medial line or midline C' will always be a connected line as it stems from an interpolation between two closed curves.

The goal of contour interpolation is to find the height values of points in between initially labeled height contours and producing a continuous interpolated grid of contour values that represent a discrete height surface between the two contours. The basic idea is to expand the contours into the intercontour space until they collide [BMT94]. The collision front defines the medial line of the intercontour space. Finally

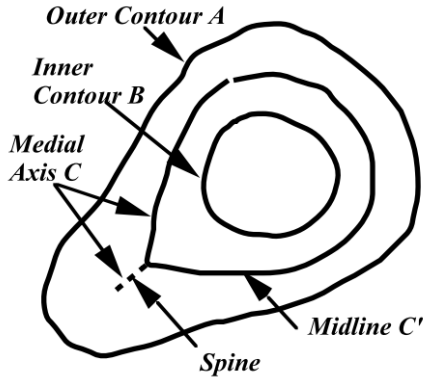


Figure 19. Illustration of medial line as the medial axis between two contours without spine [BMT94].

the intercontour line is labeled with a height value in between the two contour labels.

The steps are repeated till all the intercontour pixels are labeled accordingly. The process is visualized in Figure 20 and an example result of a generated height map is seen in Figure 21. It has to be noted that the procedure produces a staircase effect along ridges that can have negative impact on successive processing, see figure 22 for that.

To extract a corresponding contour we can filter for a certain height value in the final image and let the touching pixels evolve to the closed contour polygon through all points as illustrated in Figure 23. The more robust alternative is just to save the generated contours while they are produced. This solves situations where successive contours may override parts of already generated contours. The other problem shown in figure 23 are regions where several points form a cluster. We have to estimate a proper path there without the knowledge of the real situation.

B. Level Set Method for Contour Generation

To represent the evolution of our contours in a more general and a more continuous fashion we can utilize the level set method. In that case our contours are represented as closed 2D curves that are planar cuts of a 3D function Φ . It has to be noted that in the continuous case the function Φ has to be continuous as well as smooth, whereas in the discrete case we can deal with corners in a more natural way.

According to Persson et al. [Per05] the evolution of those boundaries or interfaces are represented implicitly and their propagation is modeled using partial differential equations. The boundary itself is given by the level sets of a function $\Phi(x)$.

Definition VI.3. Interface. The interface is implicitly defined by the zero level set of a function $\Phi(x) = 0$ [Per05]. In 3D, the interface can represent a set of 3D isocontours, each with its own 3rd coordinate, while propagating. The 3D isocontours are converted into 2D contours by dropping the third component. Propagation is provided by an additional parameter for Φ , typically $\Phi(t, x)$, where t represent the time

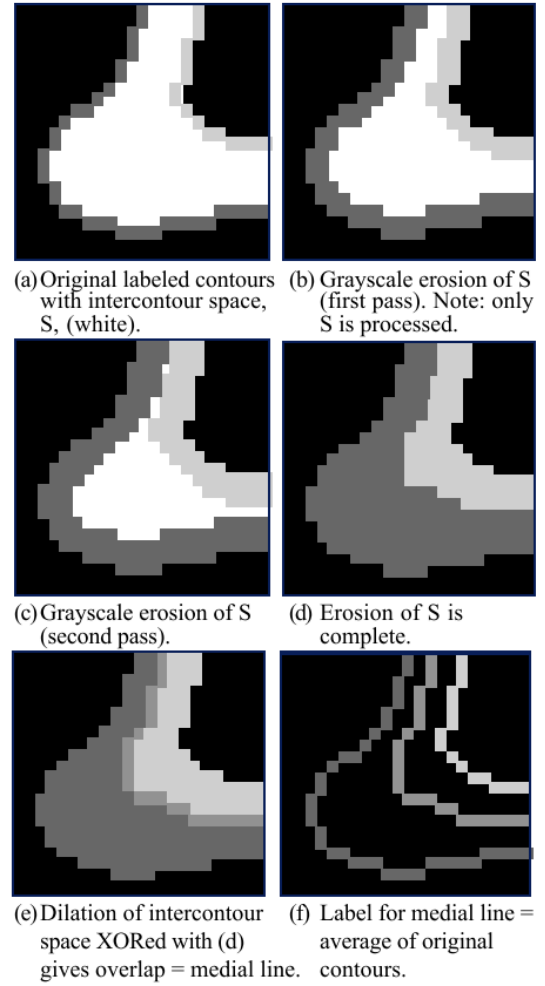


Figure 20. Steps of generating the medial line between two contours [BMT94].

passed. So interfaces can be seen to have a velocity v and move in time.

Suppose we have a boundary and a velocity field v , which depends on space and time properties of the boundary, such as normal direction and curvature, as well as on indirect or external dependencies, as shown in Figure 24.

The objective is to model the evolution of the boundary with the given velocities v . The interface is implicitly defined by the zero level set of a function $\Phi(x) = 0$. For numerical approximation Φ can be discretized over a grid, as shown in Figure 25. Nevertheless the interface should only be accessed implicitly by operating on Φ . We want to have Φ represented as *signed distance function* with the property of $|\nabla \Phi| = 1$ with different signs at the two sides of the interface. Because approximations get inaccurate if Φ has big variations in the gradient it will be frequently reinitialized to stay close to a signed distance function.

Geometric attributes of our curve are directly computed from Φ without representing the interface explicitly. So we get the normal vector and the curvature with

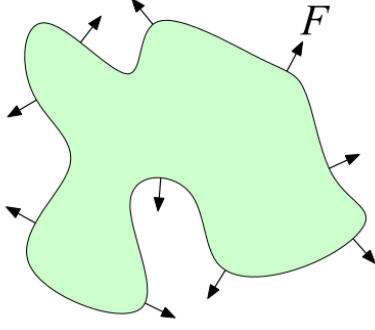


Figure 24. Evolving interface according to given speed function F [Per05].

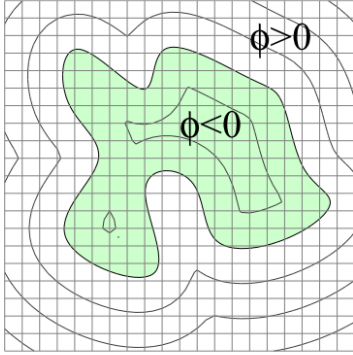


Figure 25. Discretization of signed distance function Φ on a grid [Per05].

over a short period of time, the other is to explicitly update the nodes close to the boundary by extracting curve segments and computing the distance to the grid nodes and use the *fast marching method* for the remaining nodes.

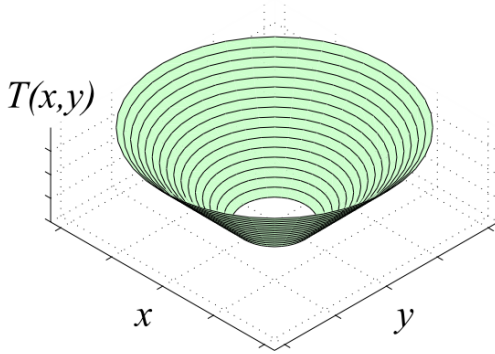


Figure 26. Boundary value formulation where $T(x, y)$ gives the arrival time of the interface at point (x, y) [Per05].

The level set equation is an initial value problem, where we track the zero level set $\Phi = 0$ in time. If the speed $F > 0$ we can see the evolution as an arrival function T , see Figure 26, where $T(x)$ is the time of the interface to reach x from

its initial location Γ . As speed times time equals distance we can derive the boundary value problem (Eikonal equation)

$$|\nabla T|F = 1, T = 0 \text{ on } \Gamma \quad (57)$$

If $F = 1$ equation (57) can be used to compute the distance functions for boundary Γ .

As the convection equation (51) before we can discretize the *Eikonal equation* (57) with

$$\left[\begin{array}{l} \max(D_{ijk}^{-x}T, -D_{ijk}^{+x}T, 0)^2 \\ \max(D_{ijk}^{-y}T, -D_{ijk}^{+y}T, 0)^2 \\ \max(D_{ijk}^{-z}T, -D_{ijk}^{+z}T, 0)^2 \end{array} \right]^{1/2} = \frac{1}{F_{ijk}} \quad (58)$$

As the front propagates outward from Γ nodes with higher value of T will never affect nodes with lower values. So the *fast marching method* considers only neighbor nodes of already 'known' initial boundary value nodes, update them with equation (58) and put them in a priority queue where the node with the smallest unknown value is removed and marked as 'known' as well. Its neighbors are then updated and inserted into the queue until all nodes are marked as 'known'.

With this framework we have the possibility to deal with contours in a more continuous fashion. For more details we refer to [Per05], [LK05], [OF01] or [Par02].

VII. DEFORMATION-BASED WARPING

Deformation-based Warping relates the source domain to the target domain based on *deformation energy* E . First we have to define the deformation energy between the source and target constellation, which can include points, lines, contours, patches or whatever we like to compare. Given that energy function we want to minimize the energy under certain constraints. Typical constraints are the exact mapping of control-structures, like control points, control lines and the like.

Closely related to parametrization, deformation-based methods can also be used in a similar fashion. Here however we will focus on 2D warps of contour outlines and try to find solutions that are discretized as less as possible to provide an analytic correct matching at an arbitrary resolution to transform sampled curve points from a space-filling curve to the warped domain as exact as possible.

The defined deformation function f will map points from the undeformed domain in the deformed co-domain.

A. Deformation using Moving Least Squares

Schaeffer et al. [SMW06] let the user select some handles, i.e. points or line segments, to control the deformation. There are source handles p_i and corresponding target handles q_i to define the deformation. The deformation function f should show certain properties:

- Interpolation: handle p should map to handle q ,
 $f(p_i) = q_i$
- Smoothness: f should produce smooth deformations
- Identity: if p is identical to q , i.e. $p = q$, then f should be the identity transform, i.e. $f(v) = v$

Undesired local non-uniform scaling and shearing are minimized by an as 'rigid-as-possible' deformation. No triangulation of the domain is required and deformations are created by solving small linear systems (2x2) at each point in closed form. As such no general linear solver is required. Further the method extends to line segments.

Given a point v , moving least squares solves for the best affine transformation $l_v(x)$ that minimizes

$$\sum_i w_i |l_v(p_i) - q_i|^2 \quad (59)$$

with the weights w_i

$$w_i = \frac{1}{|p_i - v|^{2\alpha}} \quad (60)$$

Because the weights w_i depend on v , $l_v(x)$ is different for each v . Since $l_v(x)$ is an affine transformation it consists of two parts, a transformation M and a translation T .

$$l_v(x) = xM + T \quad (61)$$

If we insert equation (61) into equation (59) it is possible to replace T by the fact that the minimum is where the derivatives of the free variables are zero. If we rearrange for T we get

$$T = q_* - p_*M \quad (62)$$

where q_* and p_* are weighted centroids

$$p_* = \frac{\sum_i w_i p_i}{\sum_i w_i} \quad (63)$$

$$q_* = \frac{\sum_i w_i q_i}{\sum_i w_i} \quad (64)$$

$$(65)$$

We can express $l_v(x)$ then by substituting T into equation (61)

$$l_v(x) = (x - p_*)M + q_* \quad (66)$$

and rewrite the energy minimization in equation (59) as

$$\sum_i w_i |\hat{p}_i M - \hat{q}_i|^2 \quad (67)$$

with \hat{p}_i and \hat{q}_i as

$$\hat{p}_i = p_i - p_* \quad (68)$$

$$\hat{q}_i = q_i - q_* \quad (69)$$

The matrix M can be further constrained to be an affine transformation, a similarity transformation or a rigid-body transformation. Each restriction will remove undesired artifacts. Since affine transformations can contain non-uniform scale and shear, we can remove the shear by restricting M to similarity transformations that only contains translation, rotation and uniform scaling. Further restriction of M towards

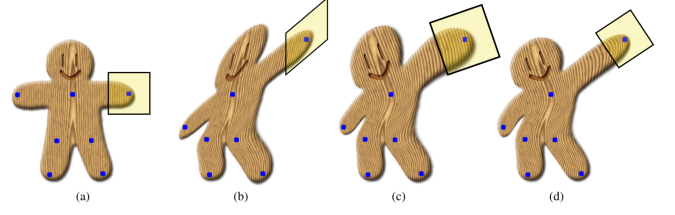


Figure 27. Moving least squares deformation on image (a) with affine transformations (b), similarity transformations (c) and rigid transformations (d), local deformation is illustrated with yellow patches [SMW06].

rigid-body transformations removes unwanted scaling as well. Additional details can be looked up at [SMW06].

For a precise control over curves, points may be insufficient to specify the deformation. While the naive approach of representing curves with dense sets of points will work it will be computationally inefficient. A better approach is to generalize the moving least squares deformations to arbitrary curves. We define $p_i(t)$ as the i -th control curve and $q_i(t)$ as the corresponding deformed curve. A generalization of equation (59) is done by integrating over each control curve $p_i(t)$ with $t \in [0, 1]$

$$\sum_i \int_0^1 w_i(t) |p_i(t)M + T - q_i(t)|^2 \quad (70)$$

with weights

$$w_i(t) = \frac{|p'_i(t)|}{|p_i(t) - v|^{2\alpha}} \quad (71)$$

where the factor $|p'_i(t)|$ makes the integral independent of the parametrization used. Translation T can still be expressed as

$$T = q_* - p_*M \quad (72)$$

with p_* and q_* as

$$p_* = \frac{\sum_i \int_0^1 w_i(t) p_i(t) dt}{\sum_i \int_0^1 w_i(t) dt} \quad (73)$$

$$q_* = \frac{\sum_i \int_0^1 w_i(t) q_i(t) dt}{\sum_i \int_0^1 w_i(t) dt} \quad (74)$$

$$(75)$$

Rewriting equation (70) in terms of M provides us

$$\sum_i \int_0^1 w_i(t) |\hat{p}_i(t)M - \hat{q}_i(t)|^2 \quad (76)$$

with \hat{p}_i and \hat{q}_i as

$$\hat{p}_i(t) = p_i(t) - p_* \quad (77)$$

$$\hat{q}_i(t) = q_i(t) - q_* \quad (78)$$

$$(79)$$

for arbitrary curves. Restricted to line segments the curves $\hat{p}_i(t)$ and $\hat{q}_i(t)$ can be represented in matrix form

$$\hat{p}_i(t) = (1 - tt) \begin{pmatrix} \hat{a}_i \\ \hat{b}_i \end{pmatrix} \quad (80)$$

$$\hat{q}_i(t) = (1 - tt) \begin{pmatrix} \hat{c}_i \\ \hat{d}_i \end{pmatrix} \quad (81)$$

$$(82)$$

where $\hat{a}_i(t)$ and $\hat{b}_i(t)$ are the endpoints of $\hat{p}_i(t)$ and $\hat{c}_i(t)$ and $\hat{d}_i(t)$ are the endpoints of $\hat{q}_i(t)$. Here we again can derive a closed form for equation (70) with a replacement of T and a minimizer that finds the optimal M , for details see Schaefer et al. [SMW06].

A result of the corresponding transformations based on affine transformation, similarity transformation and rigid-body transformation can be seen in Figure 27.

If we could manage to integrate cubic bsplines or bezier curves $p_i(t)$, $q_i(t)$ into equation (70) we would provide a smooth analytic shape that defines the outline more accurately and therefore provide a better warp in the vicinity of the border.

B. Locally injective Mappings

Schüller et al. [SKPSH13] defines an algorithm that modifies any deformation energy to guarantee local injective mappings. This is done by a barrier term in combination with a solver strategy that provides interactive manipulation and robustly handles extreme deformations.

As local injectivity requires that the Jacobian of a mapping is always positive we can guarantee that all elements will have positive area. the mapping f between arbitrary shapes is again defined by an *energy functional* $E(f)$ that measures the desired properties. The mapping is then computed as the minimizer of this energy E under certain constraints.

The approach works by augmenting an arbitrary deformation energy with a barrier term, that grows to infinity as the area of the deformed elements approach zero. Since such nonlinear energies are hard to minimize as the gradient and Hessian become ill-conditioned when elements degenerate, solutions in the vicinity of the near infinity energy areas has to be treated differently.

The minimization of a deformation energy is only of interest if a term competes with it, for instance as soft constraint. The energy depending on the position v can then be minimized by

$$\arg \min_v E(v) + \alpha |Cv - d|^2 \quad (83)$$

This energy can be modified by adding a barrier term which grows to infinity as the elements approach zero area. To prevent an ill-conditioned Hessian because of near infinity values in the barrier term a smooth series of intermediate optimization results is generated.

First we have to measure the area of the j -th element, i.e. the triangle in configuration v , by a fraction of the determinant. We define our constraint function as

$$c_j(v) = \lambda_j(v) - \epsilon; \epsilon = 10^{-5} \quad (84)$$

and forcing that elements cannot invert by requiring $c_j(v) > 0$. The energy function is then augmented with the soft constraint as well as with the barrier term and we minimize then

$$\arg \min_v E(v) + \alpha |Cv - d|^2 + \beta \sum_{j \in \mathcal{E}} \Phi_j(c_j(v)) \quad (85)$$

which guarantees that no element will invert. But the barrier function should affect the energy only when elements are close to degenerate. So we want a barrier function that smoothly approaches zero when the area of the deformed element reach a certain fraction $s_j = s\lambda_j(v_0)$ of the area in the initial rest-pose. Using a spline with the desired properties as barrier function Φ_j we got

$$\Phi_j(x) = \begin{cases} \infty, & x \leq 0 \\ \frac{1}{g_j(x)} - 1, & 0 < x < s_j \\ 0, & s_j \end{cases} \quad (86)$$

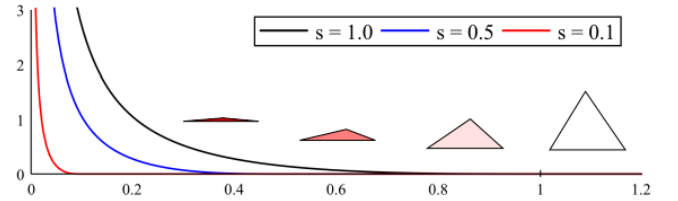


Figure 28. Increase of barrier function while degeneration of triangle [SKPSH13].

where the parameter s determines when the barrier starts to interfere to prevent flips, see Figure 28 for that. An improved barrier function will be a log-barrier function, augmented with a linear term to alter the energy when elements increase their area.

The minimization is done by a *Levenberg-Marquardt* algorithm. The barrier term offers a closed form of the gradient and Hessian $\nabla^2 E_{bar}(v)$ where the Hessian is used to quadratically approximate the energy by determining the direction of decreasing energy to a local minimum.

The iterative vertex update is done by

$$p_i = (\nabla^2 E_{bar}(v_i) + \mu_i I)^{-1} \nabla E_{bar}(v_i) \quad (87)$$

$$v_{i+1} = v_i - \sigma_i p_i \quad (88)$$

and the stepsize σ is adaptively computed by a backtracking line search, where σ_i is halved until a decrease in energy is found. Identically the value of σ_{i-1} is doubled for the i -th iteration, if the energy is still decreased. The Hessian in equation (88) is regularized by $\mu_i I$ where μ_i is chosen as

small as possible to make the Hessian invertible. The value of μ_i is chosen with the same strategy as σ_i .

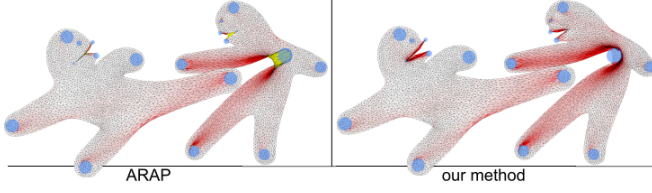


Figure 29. Improvement of ARAP-deformation energy such that inversions are infinitely penalized [SKPSH13].

As this strategy cause small step sizes and doesn't always work for extreme deformations the strategy is altered when the Hessian is close to singular. The key idea is to relax the positional constraints to allow the deformation energy and the barrier term to improve the shape of the degenerate elements. So we replace d in the positional constraint with intermediate targets t_i that are updated iteratively.

$$\arg \min_v E(v) + \alpha |Cv - t_i|^2 + \beta \sum_{j \in \mathcal{E}} \Phi_j(c_j(v)) \quad (89)$$

If the Hessian is invertible $t_i = d$, else depending on the amount of regularization needed, t_i is moved further from d according to

$$t_i = Cv_{i-1} + \frac{1}{1 + \mu_i^2} (d - Cv_{i-1}) \quad (90)$$

This adaptive substepping allows the barrier to improve the shape of the elements. Finally, as the positional constraint should always dominate the energy in a certain radius r , α is defined as

$$\gamma = \frac{r}{|Cv - d|^2} (E(v) + \beta \sum_{j \in \mathcal{E}} \Phi_j(c_j(v))) \quad (91)$$

$$\alpha_{i+1} = \min(\max(\alpha_i, \gamma), t) \quad (92)$$

Some results can be seen in Figure 29.

The system of locally injective mappings can be used to facilitate a lot of possible energies for deformation purpose while guaranteeing local injectivity of the resulting mapping.

VIII. CONCLUSION

We saw that there are many possibilities to warp two different 2D domains from one into the other. We saw that discretizations of the problem can be done at various places, starting from the problem definition, the intermediate data representation, the calculations executed to the final representation of the mapping of defined data.

As we define our model in terms of outer boundaries of a certain domain in a analytic fashion we favor solutions that could represent this outer boundary exactly while transform a point determined on a space-filling curve in a form as close as

possible to the actual mapping equations. Interpolations and discretizations of the domain are not such a good choice with respect to the nature of a space-filling curve, which does not align to any discrete structure in the domain that could be mapped straight forward to another discrete structure in the co-domain.

That's why we rate mappings based on the level set method higher than mappings based on its discrete 'counterpart', the discrete morphological contour interpolation. As the level set method has the potential to define a contour in a more analytic fashion we could derive a precise description of our outline.

The same is true for deformation methods like moving least squares because of their beautiful mapping of an arbitrary precise point from the domain into the co-domain. If this mapping would also handle analytic outlines, for instance defined by cubical curves, it may be our favorite choice.

Point-based warpings are not rated well as by their nature they could only approximate an outline by sampling it with many points. Moreover the RBF-warps can have degenerated parts in presence of high distortions.

The SOM-principle is nice in the way that it is easy to implement but we think that it suffers from a reliable flat convergence of the mesh and only will get robust results if a lot of effort is done by tuning the parameters.

Triangulation based approaches like the stretch-minimization are fine in combination with advanced interpolation algorithms, but again the proper definition of the outline is distracting. If selected, we would prefer the locally injective mappings, as they handle the cases of fold-overs in the parametrization as well but can utilize parametrizations superior to stretch-minimization.

REFERENCES

- [ADRY94] ARAD N., DYN N., REISFELD D., YESHURUN Y.: Image warping by radial basis functions: Application to facial expressions. *CVGIP: Graphical models and image processing* 56, 2 (1994), 161–172.
- [Bad12] BADER M.: *Space-filling curves: an introduction with applications in scientific computing*, vol. 9. Springer Science & Business Media, 2012.
- [Bai04] BAILEY D. G.: An efficient euclidean distance transform. In *International workshop on combinatorial image analysis* (2004), Springer, pp. 394–408.
- [Blu67] BLUM H.: A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms, 1967* (1967), 362–380.
- [Blu73] BLUM H.: Biological shape and visual science (part i). *Journal of theoretical Biology* 38, 2 (1973), 205–287.
- [BMT94] BARRETT W., MORTENSEN E., TAYLOR D.: An image space algorithm for morphological contour interpolation. In *Graphics Interface* (1994), CANADIAN INFORMATION PROCESSING SOCIETY, pp. 16–16.
- [Bor86] BORGEFORS G.: Distance transformations in digital images. *Computer vision, graphics, and image processing* 34, 3 (1986), 344–371.
- [CL98] CHAN K. H., LAU R. W.: Contour-based warping. *Graphical models and image processing* 60, 5 (1998), 331–348.
- [EHP04] ERICKSON J., HAR-PELED S.: Optimally cutting a surface into a disk. *Discrete & Computational Geometry* 31, 1 (2004), 37–59.
- [FK94] FERMÜLLER C., KROPATSC W.: Multiresolution shape description by corners. In *Shape in Picture*. Springer, 1994, pp. 539–548.

- [Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.* 14, 3 (Apr. 1997), 231–250.
- [Flo15] FLOATER M. S.: Generalized barycentric coordinates and applications. *Acta Numerica* 24 (2015), 161–214.
- [Key81] KEYS R.: Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing* 29, 6 (1981), 1153–1160.
- [Khe08] KHENG L. W.: Image warping and morphing. <https://www.comp.nus.edu.sg/~cs4340/lecture/imorph.pdf>, 2008.
- [KLYL09] KWON J. H., LEE B. G., YOON J., LEE J. J.: Image deformation using radial basis function interpolation.
- [Koh98] KOHONEN T.: The self-organizing map. *Neurocomputing* 21, 1-3 (1998), 1–6.
- [LK05] LARSEN J., KRISTENSEN T. G.: An overview of the implementation of level set methods, including the use of the narrow band method. <http://www.daimi.au.dk/~geko/courses/LevelSet/LevelSet.pdf> (2005).
- [LS08] LANGER T., SEIDEL H.-P.: Higher order barycentric coordinates. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 459–466.
- [mit] Tutorial: Hilbert curve coloring. <http://www.kerrymitchellart.com/tutorials/hilbert/hilbert-tutorial.html>.
- [Nec10] NECHAEVA O.: Using self organizing maps for 3d surface and volume adaptive mesh generation. In *Self-Organizing Maps*. InTech, 2010.
- [NT13] NAJMAN L., TALBOT H.: *Mathematical morphology: from theory to applications*. John Wiley & Sons, 2013.
- [OF01] OSHER S., FEDKIW R. P.: Level set methods: an overview and some recent results. *Journal of Computational physics* 169, 2 (2001), 463–502.
- [OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics* 79, 1 (1988), 12–49.
- [Par02] PARAGIOS N.: Level set methods in medical image analysis: Segmentation. <http://web.cse.ohio-state.edu/~parent.1/classes/694A/Lectures/Material/levelset.ppt>, 2002.
- [Per05] PERSSON P.-O.: The level set method. <http://math.mit.edu/classes/18.086/2007/levelsetnotes.pdf>, 2005.
- [REM] RICHARD C., EUGENE I., MINA R.: On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics* 5, 3, 243–255.
- [Ser83] SERRA J.: *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.
- [SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 125–135.
- [SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *ACM transactions on graphics (TOG)* (2006), vol. 25, ACM, pp. 533–540.
- [Whi17] WHITAKER R.: Geometric transformations and image warping. http://www.eng.utah.edu/~cs6640/geometric_trans.pdf, 2017.
- [XT09] XIA H., TUCKER P. G.: Distance solutions for medial axis transform. In *Proceedings of the 18th International Meshing Roundtable*. Springer, 2009, pp. 247–265.
- [YBS04] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of the Shape Modeling International 2004* (Washington, DC, USA, 2004), SMI '04, IEEE Computer Society, pp. 200–208.

1 Warping of Space Filling Curves

Discussion Protocol

1.1 Can we construct an injective space filling curve?

No, not if it should be continuous. The connection elements between the fundamental patterns cause it to have overlaps in the limit.

1.2 Can interpolation cause self intersections of the space filling curve?

No, not with a valid interpolation scheme. However we cannot provide any guaranteed distances between iterations any more.

1.3 May discretisation cause self intersections of the space filling curve?

If we come below the sampling frequency of the Nyquist-Theorem, yes.

1.4 Red segments part of the curve? If no still continuous? If yes how can we assure parameterisation?

Theoretically we have the problem that with the connection elements between the fundamental patterns we either have a continuous but surjective curve. Without these elements the curve is bijective but not continuous anymore. Practically this is not a problem, as we only iterate to a certain level and are not interested in the true space filling curve at the limit.

Radial Basis Function: Why the constraint that $\sum(k_i) = 0$ and $\sum(k_i * x_i) = 0$? ($i = 1..N$)

This provides necessary constraints to represent affine transformations by the polynomial part.

1.5 Self Organising Maps: Why neurons? Related to Neural Network or just similar terminology?

Historically Kohonen defined this term as the concept comes from the artificial intelligence community. With this he does dimensionality reduction of high dimensional data into a two dimensional net-structure. Somehow the 2D net folds itself the best way into the high dimensional data field.

1.6 Stretch Minimization: What can go wrong on non-convex (concave) boundaries?

With concave boundaries we can get foldovers of triangles in the mapping. So the triangle invert their orientation or even can fold over each other.

1.7 All warping methods: If warping method is inexact, can we still ensure that curve is space filling and non self intersecting?

We can assure that the curve fills the specified domain but there may be uncaptured areas at the border. The curve will still be without self intersections but as stated before we cannot guarantee a precise distance between non-overlapping curve-segments.

1.8 How is deformation based warping with point handles different to point based warping?

It is related in the sense that deformation based warping can utilize points for defining the warp. But it can also use other elements of correspondence, like line segments, patches or other geometrical elements that should establish the relation.

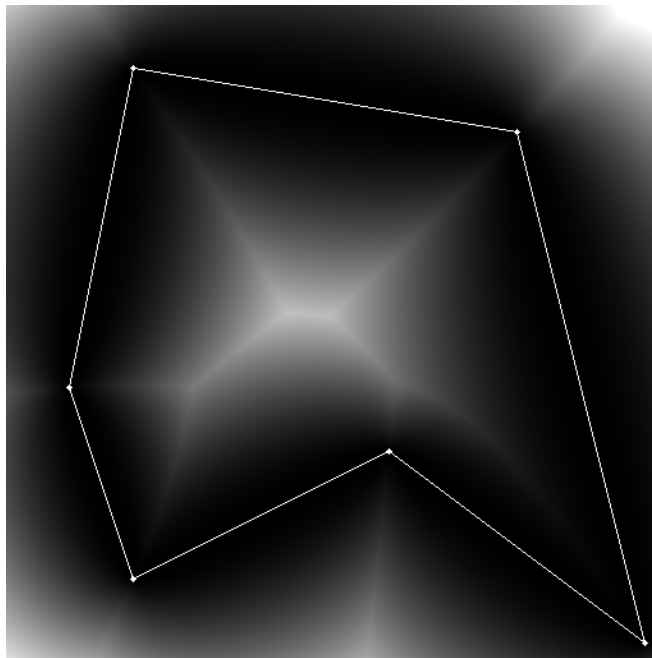
1.9 DBW: Include translation part T into matrix M by using homogeneous coordinates?

No, the Transformation T is encoded in a clever way in Matrix M by exploiting constraints we want to have fulfilled.

Hyperbolic Medial Axis

Bernhard Langer (00427400)
183.151 Selected Chapters in Computer Vision S2018

July 6, 2018



Abstract

The *Medial Axis Transform* (*MAT*) defines a skeleton as a simple shape representation of a binary image region. With the *hyperbolic medial axis* we present an alternative to the classic medial axis on polygonal regions. The hyperbolic medial axis uses an alternative distance measure called the *Confocal Elliptic Distance* (*CED*) to define a distance between an image point and a line segment of the region's boundary. This extension allows to define an exoskeleton for convex regions. Additionally we present some geometric properties of the hyperbolic medial axis and take a look at the underlying algorithm and its parallelisation.

1 Medial Axis Transform

The classic *medial axis* is a simplified representation of a binary image region. Most notably based on the works of Harry Blum who laid the foundation of the medial axis as a shape descriptor [6].

The classic definition of the medial axis using Euclidean distances for a given binary image region R is the set of points that are centers of maximal inscribable circles or *medial discs* C_i that touch the region boundary in at least two points without intersecting it. This concept can be applied to 3 dimensions by exchanging the medial discs with maximal inscribable spheres [6].

An alternative definition of the medial axis that can be incorporated into an algorithm are the local maxima in at least one direction (*lmod*) of the minimal Euclidean distances between each point of the region to the closest point on the boundary [6]:

$$MAT(R) = lmod(\min(\overline{XB}), B \in boundary(R)), X \in R \quad (1)$$

Figure 1 shows an example of the classic medial axis including a few exemplary maximal inscribable circles.

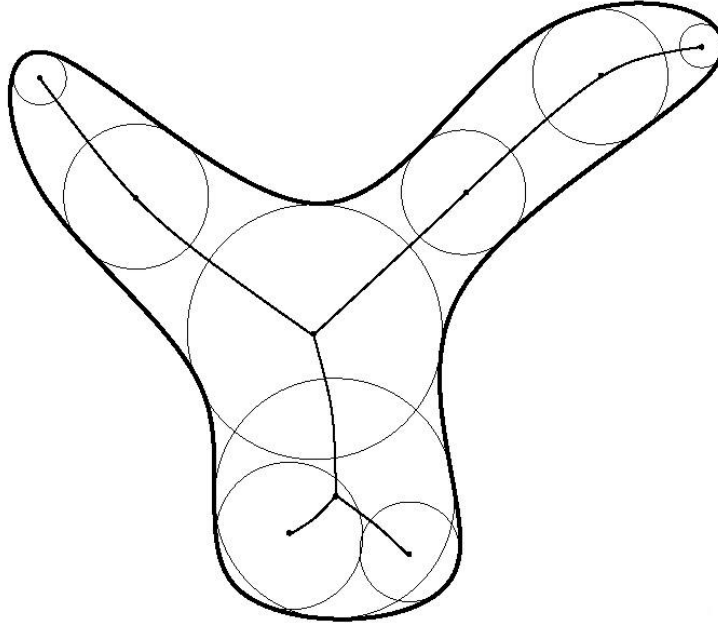


Figure 1: Example of the medial axis transform as seen in Gupta2006 [3].

1.1 Medial Axis of Polygonal Region Shapes

Given a convex region of polygonal shape (i.e. bound by line segments) without any holes, the classic medial axis itself consists only of line segments too. Each line segment of the medial axis is then part of the angular bisector of two (usually consecutive) line segments of the image region boundary.

The medial axis is only defined on the inside of convex regions, since medial disks touching the region boundary from the outside can grow to infinite size. The alternative definition of the *lmod* on the other hand works in theory but fails due to the fact that points within an area around the joint of two line segments have an equal minimal distance to both line segments, therefore the *lmod* does not exist in that region.

Figure 2 shows an example where the points in the blue region have equal distance to two different line segments.

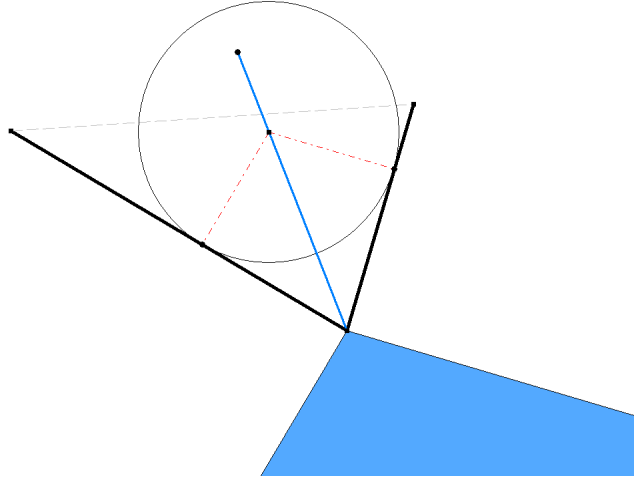


Figure 2: Example of a region around the line segment joint, where every point within the region has identical minimal distance to both line segments and a local maximum of minimal distances cannot be found.

This problem however is related to the fact that we use Euclidean distances where the isolines of two separate line segments may partially overlap. This can be overcome if we use a different distance measure. In the next section we will present an alternative distance on which we can define a medial axis outside of the region.

2 Confocal Elliptic Distance

For a given line segment s and point P , we consider the distance $d(s, P)$ between them. So far we assigned the distance between P and its closest point on s ($\overline{XP} | X \in s$). This definition is closely related to the *Hausdorff distance* defined on arbitrary sets [4].

However there are multiple ways to define a distance between a line segment and a point, one of which is the *Confocal Elliptic Distance* or $CED(s, P)$. The Confocal Elliptic Distance takes the endpoints F_1 and F_2 of a line segment s to compute a distance value that is zero if P lies on the line segment s and has a positive value otherwise. The formula is given by:

$$CED(s, P) = \frac{1}{2} \cdot (\overline{PF_1} + \overline{PF_2} - \overline{F_1F_2}) \quad (2)$$

The definition is very closely related to the implicit formula of the ellipse given by two focal points F_1 and F_2 , where the ellipse is defined as the union of points X with $\overline{XF_1} + \overline{XF_2} = 2 \cdot a$. The subtraction of the constant factor $\overline{F_1F_2}$ ensures $CED(s, P) = 0 | P \in s$ and the factor $\frac{1}{2}$ ensures that $CED(Q, P) = d(Q, P)$ for two points P and Q where Q is interpreted as a line segment of zero length. These adjustments however do not change the geometric structure of the isolines, which means that all points P for a given $CED(s, P)$ value and given line segment s all lie on an ellipse with F_1 and F_1 as its focal points.

Different isolines for a fixed line segment s form a set of confocal ellipses, since they all share the same line segment endpoints as their focal points. These confocal ellipses are all intersection-free, because if such an intersection point X on two different isolines would exist, it would have two different confocal elliptic distances to the line segment s which is not possible since the CED is a function that returns only a single value.

3 Hyperbolic Medial Axis

By applying the Confocal Elliptical Distance on every point of the plane for a given line segment s , we get its *Confocal Elliptical Field* $CEF(s)$. Given a polygonal region R with line segment s_i forming its boundary, every border segment s_i creates its own Confocal Elliptical Field $CEF(s_i)$. We can now create a Confocal Elliptical Field Distance Transform $CEF-DT$ of the polygon by combining all the individual Confocal Elliptical Fields of the polygon segments by assigning each point its minimal Confocal Elliptical Distance to each of the line segments s_i :

$$CEF-DT(s_1, \dots, s_n, X) = \min_{i=1, \dots, n} CEF(s_i, X), X \in \mathbb{R}^2 \quad (3)$$

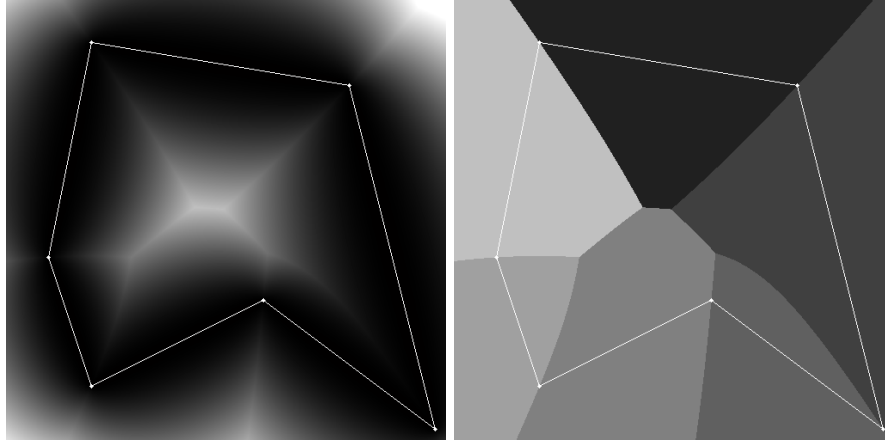
Similar to the alternative definition of the classic medial axis as the *lmod* of minimal Euclidean distances to the pixels on the boundary, we define the *Hyperbolic Medial Axis* as the *lmod* of the minimal confocal elliptical distances

to the boundary segments. The *CEF-DT* already assigns each point the minimal distance to each border segment of the image region R , and therefore the Hyperbolic Medial Axis consists of the *lmod* of the combined Confocal Elliptical Field.

$$HMAT = lmod(\min(CEF(s_i, P)), P \in \mathbb{R}^2, s_i \in \text{boundary}(R)) \quad (4)$$

$$= lmod(CEF-DT(s_1, \dots, s_n, P)) \quad (5)$$

Figure 3 shows an example of the Hyperbolic Medial Axis and the corresponding combined distance field.



(a) The combined CEF-DT formed as the minimum of multiple Confocal Elliptic Distance Fields (b) Segments illustrating each contributing Distance Field, the cell boundaries form the Hyperbolic Medial Axis

Figure 3: Example of a Hyperbolic Medial Axis for a given polygonal region.

4 Geometric Properties of the Hyperbolic Medial Axis

4.1 Branches of the Hyperbolic Medial Axis defined by Consecutive Line Segments

Along the boundary polygon of the image region, each pair of consecutive line segments shares one of their endpoints with each other. Let's consider the two distance fields CEF_{12} and CEF_{23} defined by each line segment F_1F_2 and F_2F_3 respectively. If we take the pixel-wise minimum of each distance field, the branch of the Hyperbolic Medial Axis is the separation curve of the regions of the contributing distance fields CEF_{12} and CEF_{23} as previously shown in figure

3b, thus having equal Confocal Elliptic Distance to each line segment F_1F_2 and F_2F_3 .

$$\begin{aligned} CED(F_1F_2, P) &= CED(F_2F_3, P) \\ \overline{F_1P} + \overline{F_2P} - \overline{F_1F_2} &= \overline{F_2P} + \overline{F_3P} - \overline{F_2F_3} \\ \overline{F_1P} - \overline{F_3P} &= \overline{F_1F_2} - \overline{F_2F_3} \end{aligned} \quad (6)$$

Now we consider the hyperbola which is defined by all points X of the implicit formula $|\overline{F_1X} - \overline{F_2X}| = 2a$. The hyperbola can be split into two separate branches $\overline{F_1X} - \overline{F_2X} = +2a$ and $\overline{F_1X} - \overline{F_2X} = -2a$. Now we can see that the branch of the Hyperbolic Medial Axis defined by two consecutive line segments is a single hyperbola branch with focal points F_1 and F_3 . Furthermore the branch goes through F_2 where $CED(F_1F_2, P) = CED(F_2F_3, P) = 0$.

Figure 4 shows an example of two consecutive line segments and the hyperbolic branch of the medial axis created by them.

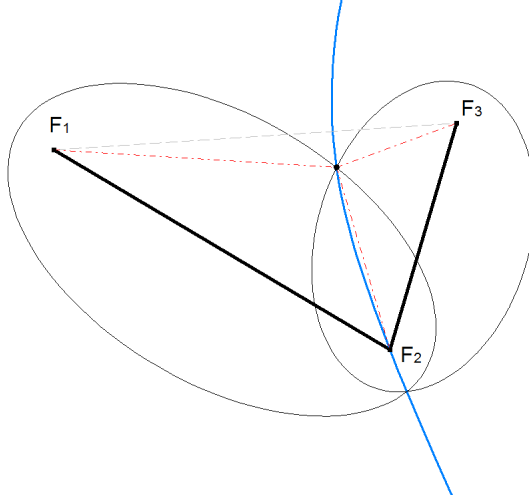


Figure 4: Two consecutive line segments F_1F_2 and F_2F_3 of the image region boundary polygon. The Hyperbolic Medial Axis of the Polygon creates a hyperbolic branch through F_2 with focal points F_1 and F_3 .

4.2 Effects of Polygon Sampling at Different Levels

Since the Confocal Elliptic Distance incorporates the complete line segment, changes in length influences the distance value. This may cause some unwanted effects if the polygon boundary line segments are modified. One example to illustrate the issue are the effects of different sampling resolutions of the boundary (e.g. if derived from a non polygonal source). Figure 5 illustrates this for

a simple triangle. In the first case 5a the triangle segment on the left creates a unified distance field, the colors only show the regions of influence but not the value, the separation curves form the Hyperbolic Medial Axis. In the second case 5b the left triangle side is divided into two separate line segments, each spawning its own distance field. Not only is an additional branch introduced to the Hyperbolic Medial Axis, but also the original branches are effected either by simple changes in curvature (bottom left branch) or even by changes in continuity. Even denser sampling as shown in figures 5c and 5d introduces further artifacts, since the dark region on the left has a lower Confocal Elliptic Distance to the right triangle segment caused by the relative length difference of the line segments.

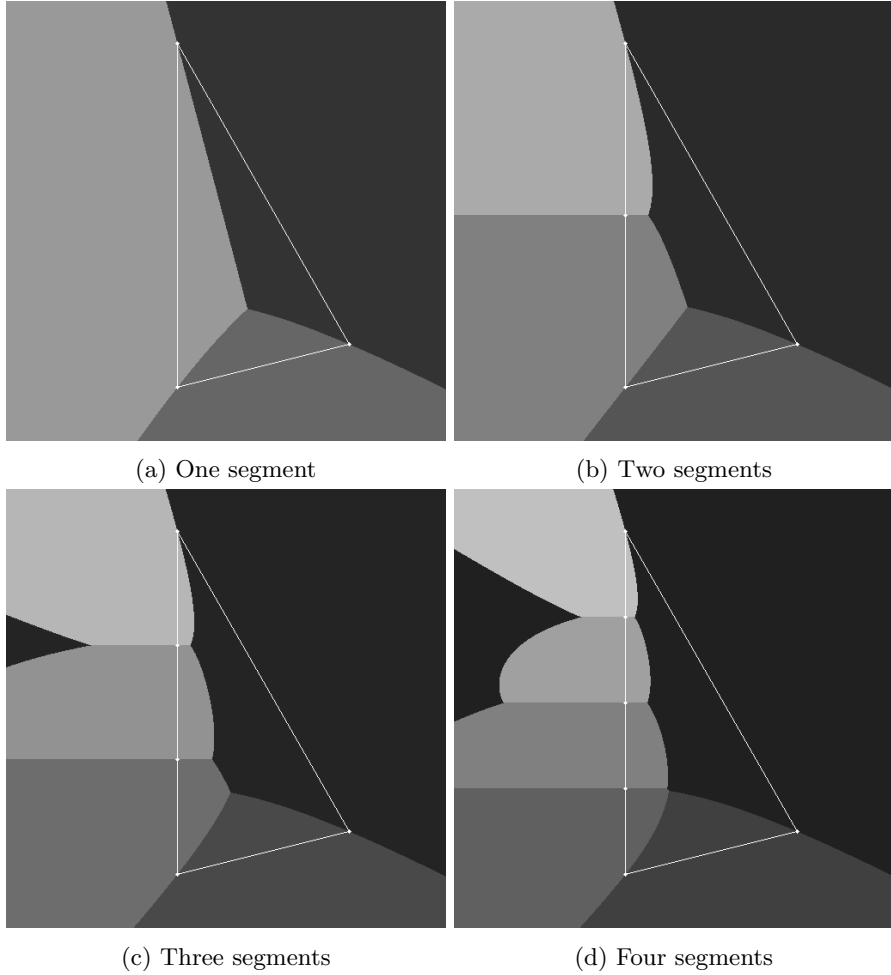


Figure 5: Hyperbolic Medial Axis for different sampling resolution

5 Algorithms and Parallelisation

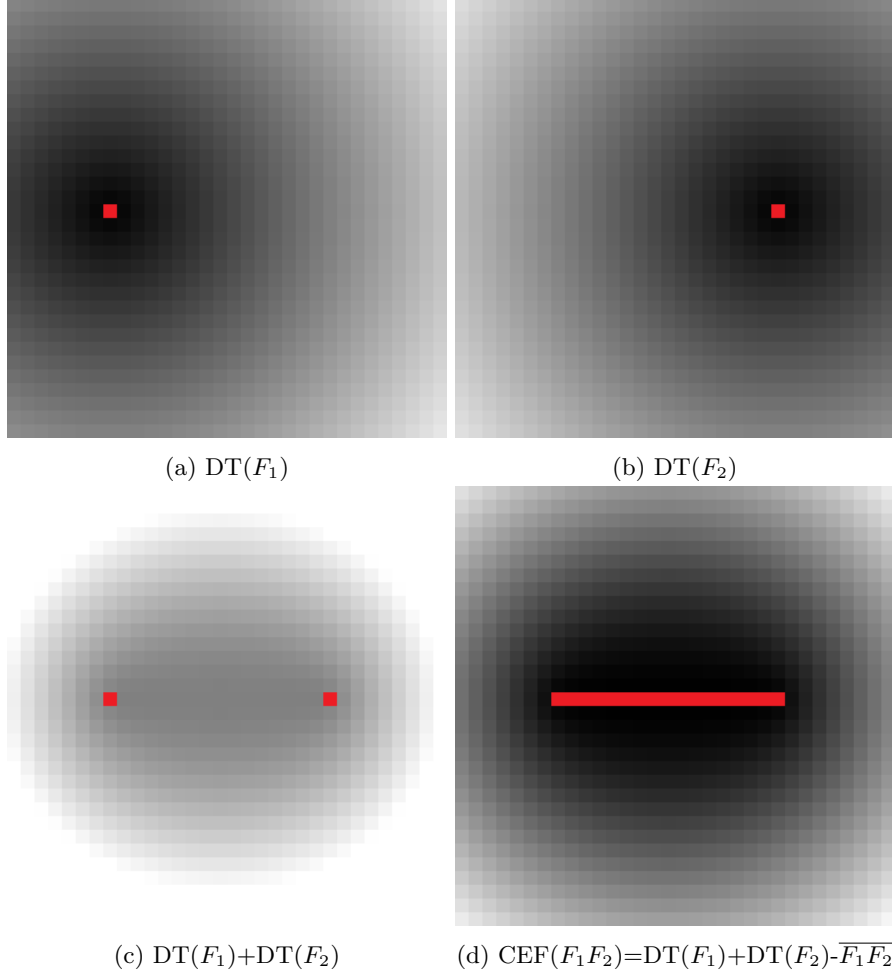


Figure 6: Construction of the Confocal Elliptic Field of a line segment F_1F_2 as a combination of two distinct Euclidean distance fields $DT(F_1)$ and $DT(F_2)$.

In the previous chapters we talked about Confocal Elliptic Distance and the Hyperbolic Medial Axis from a rather analytical perspective set in continuous space. However if we apply these concepts to image regions, we will have to operate in discrete image space.

The classic *Medial Axis Transform (MAT)* can be implemented by several different algorithms. One possible approach to approximate the medial axis in linear time is the *chamfer algorithm*. It is based on a 2-pass image sweep that propagates distances, usually in the first step from upper left to lower right forward and in the second step in the opposite direction backwards [1] [2].

For the algorithm that creates the hyperbolic medial axis we will take a different approach. Contrary to the Euclidean *MAT* on arbitrarily shaped image regions where the distance field uses every pixel on the boundary of the image regions, the Hyperbolic *MAT* on polygonal image regions only requires the line segment endpoints. Therefore we can create a Confocal Elliptic Distance Field (*CEDF*) for every line segment of the regions boundary and use them to create the overall distance field as the minimum of each *CEDF* for each individual pixel as presented previously.

Let's take a look again on the formula of the Confocal Elliptic Distance: $\frac{1}{2}(\overline{XF_1} + \overline{XF_2} - \overline{F_1F_2})$. Apart from the scalar factor $\frac{1}{2}$ and the subtracted constant factor $\overline{F_1F_2}$, the Confocal Elliptic Field is a simple combination of two separate Euclidean distance fields $DT(F_1)$ and $DT(F_2)$.

Since no distance propagation is used in this algorithm, we can compute the distance field for each pixel independently. Actually we can compute the distance field for a single pixel and apply the result to pixels at different locations by shifting, since the distance transform is location invariant. This means that the algorithm can be subject to efficient parallelisation.

Algorithms based on distance propagation cannot be applied in a straight forward way without adaptations, since the isolines of the *CED* are not necessarily equidistant.

6 Related and Future Work

6.1 Excentricity Transform on Confocal Elliptic Distance

The concepts shown in this report can easily be applied to the *eccentricity transform* [5] of regions of polygonal shape without any holes. However if we want to apply it to regions containing holes, future work would involve a definition of the shortest path between two points also known as the *geodesic distance*. One possible solution might be to take the thread construction of an ellipse into account and wind the threads around the hole, in other words the two Euclidean distances $\overline{F_1X}$ and $\overline{F_2X}$ are exchanged with two geodesic distances.

6.2 Arbitrarily Shaped Image Region Boundary

So far, we have only defined the Hyperbolic Medial Axis for image regions of polygonal shape. Arbitrarily shaped region boundaries might be approximated by polygons, but variations in sampling resolution highly influence the result.

However we can turn the concept upside down, by creating a medial axis only consisting of line segments that approximate the regions piecewise with regards to the Confocal Elliptic Distance from the line segments of the linear medial axis to pixels on the boundary. This shape representation could be seen as an arrangement of best fitting ellipses.

6.3 Extend CED to Full Metric

The Confocal Elliptic Distance $CED(s, P)$ as presented defines a distance between a line segment s and a point P . It is compatible with the Euclidean distance between two points if one point is interpreted as a line segment of zero length such that $CED(s = (FF), P) = d(F, P)$. The next step would be to define a distance $CED^*(s_1, s_2)$ between two separate line segments s_1 and s_2 that is compatible with our presented Confocal Elliptical Distance such that $CED^*(s_1, s_2 = (FF)) = CED(s_1, F)$.

Furthermore I would suggest to construct a full metric fulfilling the identity and subadditivity condition.

6.4 Parallel Framework

As shown the algorithm to construct the hyperbolic medial axis is highly parallelisable and thus one of the next steps would be to create a complete framework using a parallel environment such as *CUDA* or *OpenCL*.

7 Conclusion

We created an alternative to the classic *medial axis* on regions of polygonal shape called the *Hyperbolic Medial Axis (HMA)* that allows us to define an exoskeleton for regions of convex shape. The *HMA* incorporates the *Confocal Elliptic Distance (CED)*, a distance measure between a line segment of the polygon and a point in the plane.

We presented some geometric properties of the *HMA* and effects of sampling a line segment at different resolutions by splitting the line segments into multiple subsegments of varying length.

We showed the algorithmic advantage of the *HMA* in terms of parallelisation and gave an exemplary approach how to compute the *Confocal Elliptic Distance Field (CEDF)* as a combination of Euclidean distance fields of their endpoints.

Finally we gave a short overview of related topics and future work on the *Hyperbolic Medial Axis* and the concept of the *Confocal Elliptic Distance* in general.

References

- [1] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer vision, graphics, and image processing*, 27(3):321–345, 1984.
- [2] Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986.
- [3] Ashish Gupta. Writer dependent handwriting synthesis. 07 2006.
- [4] F. Hausdorff. *Mengenlehre*, pages 41–351. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [5] Walter G Kropatsch, Adrian Ion, Yll Haxhimusa, and Thomas Flanitzer. The eccentricity transform (of a digital shape). In *International Conference on Discrete Geometry for Computer Imagery*, pages 437–448. Springer, 2006.
- [6] Jayanta Mukhopadhyay, Partha Pratim Das, Samiran Chattopadhyay, Partha Bhowmick, and Biswa Nath Chatterji. *Digital geometry in image processing*. Chapman and Hall/CRC, 2016.

Hyperbolic Medial Axis - Questionnaire

Christian Brändle
TU-Wien, Technical University of Vienna
Wien, Karlsplatz 13, 1040 Wien

July 4, 2018

1 Why restriction on line segments (p.6)? Can we have a more general shape?

The Confocal Elliptic Distance for now is only defined on line segments and therefore I restrict the region boundaries to polygonal shape only consisting of line segments. However the CED can be extended in future work to work on arbitrary curvature segments. One idea would work as follows: If we consider a line segment s and its elliptic isoline e for a given CED value d , we will find that the classic medial axis of e happens to be the original line segment s . Using this property we can map the medial disks of e by their centers on s to any curve segment c and derive a distance value $CED^*(c,P)$.

2 Why no holes, is it possible to adapt the scheme to holes? Or to cut the domain like I do with the warping?

The holes aren't actually a restriction for the classical medial axis, but only a requirement for the medial axis to consist only of line segments. Also the Hyperbolic medial axis can be applied to regions with holes, as long as these holes are also of polygonal shape.

3 Explanation of undefined region (p.15)

The requirement for the medial axis is that points on it are a local maximum in at least one direction. The undefined regions however do not consist of these directional local maxima although the first derivation in at least one direction is zero. This stems from the fact that isolines of euclidean minimal distance for two consecutive line segments overlap partially. The elliptic isolines of the confocal elliptic distance on the other hand always intersect in four countable points (real or complex) unless they are identical. However it is possible to extend the definition and for example take the intersection point in the direction of the biggest curvature.

4 How does the properties relate to a good or desireable transform or how they can be used related to that (p.20ff)?

Some properties of the transform can be derived from these geometric properties such as for example the location the intersection point of the hyperbola branches (Equal Detour Point) always in the triangle formed by the incenter and the shortest line segment of ABC . From my perspective the geometrical aspects of the transform are primarily of use/interest for the future research and the general topic of the Confocal Elliptic Distance rather than the special application of the Hyperbolic Medial Axis.

5 Resampling of boundary: it this not a bad property that the transform depends on the sampling of the boundary? Can this fact may be used in a positive way?

It is something to be aware of to prevent unwanted effects. One aspect we can observe for sampling of the higher resolution is that the medial axis moves closer to the subsampled line segment and in fact we can move as close to it as we need by sampling in a sufficiently high resolution.

6 CED-DT using EDT components: What does this transformation actually show/say now? What is the source image that cause this transform picture?

The source of the two EDT components are a single point each corresponding to the two line segment endpoints and they are created with a simple euclidean distance transform. The advantage of this is that we can compute the distance field for a single point in the plane and use the result for any point of the plane simply by shifting the result image with a vector corresponding to the point position.

7 Can we use others seeds instead of line segment to define another type of voronoi diagram or distance transform (F1)?

Yes, we can apply the concept of the confocal elliptic distance to any geometric simplex. Since an n -simplex contains n vertices V_n we can use the vertex/point distances $d(V_n, P)$ for our computation.

8 Can you explain the Hausdorff distance in more detail? With a figure/picture (slide 2)?

See explanation picture at https://en.wikipedia.org/wiki/File:Hausdorff_distance_sample.svg.

9 Is a reconstruction of an original domain possible by an Confocal Elliptic Distance Transform like with the Medial Axis Distance Transform (slide 3)?

From geometrical point of view we can analytically (maybe even with a construction) derive the focal points for a given hyperbola from a given branch segment, but I fear that this approach is very sensible to the input and at this points its hard to give an upper bound of the reconstruction error, therefore it should be treated as a theoretical possibility rather than a practical one.

10 Colinear line segments: Voronoi cell of s_i is completely swallowed by s_j . Can this feature be useful? In what for a situation (slide 6)?

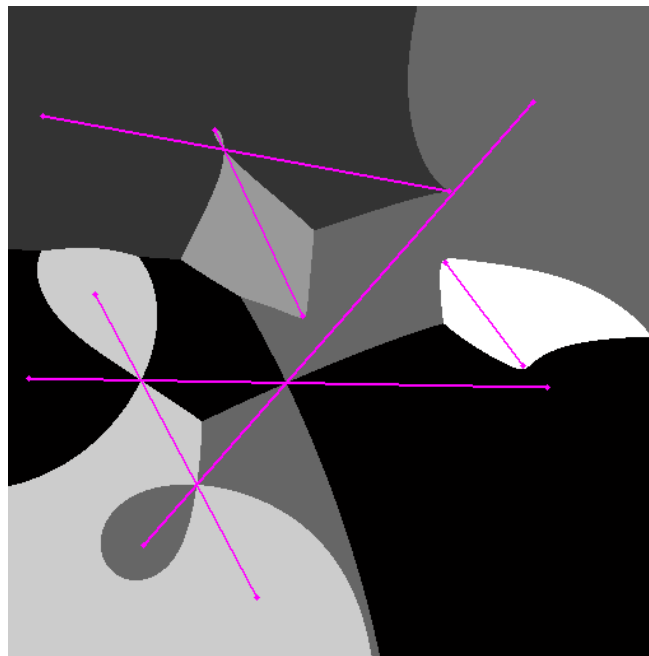
For the Voronoi Diagram seed line segments may coincide with parts of bigger line seeds, in this case it can be a useful property that the shorter line segments do not spawn additional Voronoi Cells. For the hyperbolic medial axis this case will not occur though. However errors in the discretization stage may lead to the case that line segments that should be sharing one endpoint end up creating a very short additional line segments (only a few pixels long, maybe one or two), in this case the artifacts created by this additional line segments are attenuated by this property.

Elliptic Voronoi Diagram

Bernhard Langer (00427400)

186.837 Seminar aus Computer Vision und Mustererkennung S2018

July 6, 2018



Abstract

The *Voronoi Diagram* is a partitioning of the Euclidean plane into Voronoi Cells defined by a seed set consisting of points only. We present an extension of the Voronoi Diagram on a seed set of line segments for which we present an alternative distance measure called the *Confocal Elliptic Distance* to define a distance between a line segment seed and a point of the Euclidean plane. The extended Voronoi Diagram will also work on a seed set including point seeds and will return the classic Voronoi Diagram if applied to a seed set consisting of points only. Furthermore we take a look at various geometric properties and in detail the case where three line segment form a triangle.

1 Classic Voronoi Diagram

The Classic Voronoi Diagram on a point set \mathbb{P} is a partitioning of the Euclidean Space into Voronoi Cells. For each point $P_i \in \mathbb{P}$ the corresponding Voronoi Cell VC_i contains all points $Q \in VC_i$ that are closer to P_i than to any other point in \mathbb{P} (with regards to the Euclidean Distance). The points in \mathbb{P} are also called seeds for the Voronoi Cells. Points in the plane that have the same distance to two or more points in \mathbb{P} are part of the separation lines of the Voronoi Cells and therefore lie on the cell borders. [1]

Figure 1 shows an example of the Classic Voronoi Diagram.

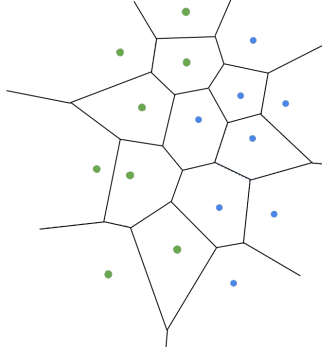


Figure 1: Voronoi Diagram for a point set using Euclidean distances

2 Voronoi Diagram on Line Sets

Now we consider a more general seed set \mathbb{S} for which we extend the point set to line segments of finite length. To create a Voronoi Diagram for line segments, we have to define some kind of non-negative distance measure $d(s, P)$ for a given line segment s and a point P . Furthermore we want this distance measure to be compatible with the Euclidean distance for points interpreted as line segments of length zero and for points lying on the line segment we want the distance to be zero.

$$\begin{aligned} d(s, P) &\geq 0 \\ d(s, P) &= 0, P \in s \\ d(s = (QQ), P) &= \overline{PQ} \end{aligned}$$

3 Hausdorff Distance and its Application on Line Voronoi Diagrams

One simple way to define a distance between a line segment s and a point P is to use an assymmetric variation $d_{\tilde{H}}$ of the *Hausdorff Distance* d_H [2]. The definition of the Hausdorff Distance between two sets X and Y is given by:

$$d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\} \quad (1)$$

To apply this distance function to our line segments, we consider only a part of d_H and define our variation for a given line segment seed s and a given point P as:

$$d_{\tilde{H}}(s, P) = \sup_{P \in \{P\}} \inf_{Q \in s} d(P, Q) = \inf_{Q \in s} d(P, Q) \quad (2)$$

In other words this distance function returns the shortest Euclidean distance between P and its closest point Q on the line segment s . Its isolines are equidistant ovals around the line segment.

Figure 2a shows an example of an isline for a given distance including an example point on the isline.

However if we use this distance measure to create a distance field on a seed set containing two separate line segments that share a common endpoint, the isolines of both line segments overlap partially. Points in this region around the shared endpoint have the same minimal Euclidean distance to both incident line segments and therefore are not assignable to a Voronoi Cell and can be interpreted as separation areas.

Figure 2b shows an example of such an unassignable area.

In the next section we will discuss a different approach, that will always create separation curves of zero width.

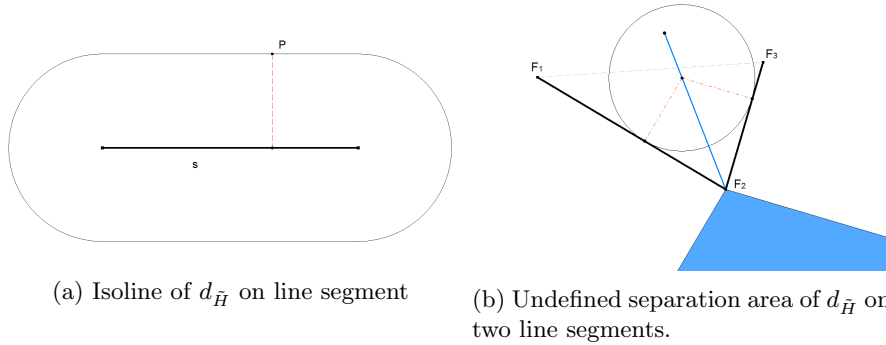


Figure 2: Examples for the distance metric $d_{\tilde{H}}$ derived from Hausdorff Distance.

4 Confocal Elliptic Coordinates

Another possible way to define such a distance measure would be to consider the line endpoints F_1 and F_2 of a given line segment s . The *Confocal Elliptic Coordinates* (or *Confocal Ellipsoidal Coordinates*) identify a point in the plane using two conic sections [3]. For a given line segment s and its endpoints we consider all ellipses E_s with the line endpoints of s as their focal points F_1 and F_2 . Different ellipses with the same focal points are called confocal. For every point P in \mathbb{R}^2 we can find an ellipse that goes through P , thus E_s covers the whole plane. Furthermore, not more than one ellipse from E_s can go through P , in other words the ellipses in E_s are intersection-free.

For the second coordinate we consider the set of intersection-free hyperbolas H_s with the same focal points. Every intersection of a hyperbola and an ellipse from these sets has the property that their tangents in the intersection points are orthogonal [3].

Figure 3 shows the Confocal Elliptic Coordinates for a given line segment.

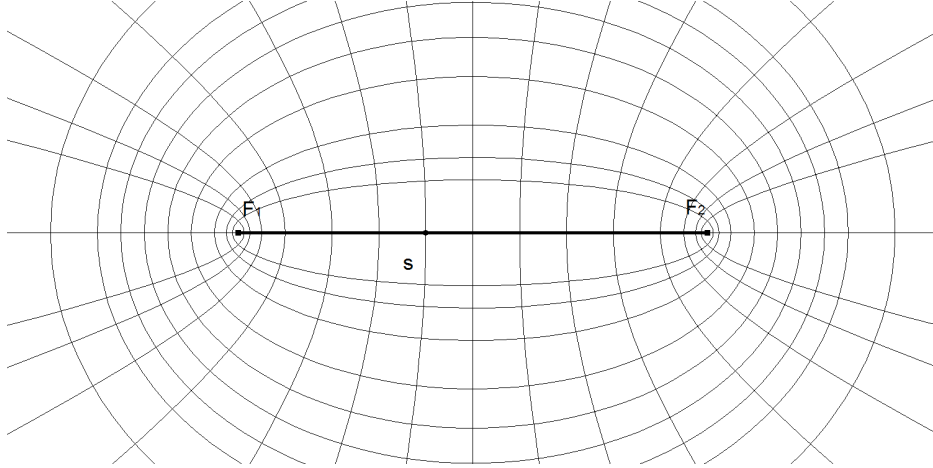


Figure 3: Confocal Elliptic Coordinates for the line segment s .

5 Confocal Elliptic Distance

The *Confocal Elliptic Distance* (CED) for a line segment s and a point P takes the distances between F_1 , F_2 and P into consideration: Based on the formula of an ellipse $\overline{F_1X} + \overline{F_2X} = 2a$ with a constant factor a we define

$$CED(s, P) = \frac{1}{2} \cdot (\overline{F_1P} + \overline{F_2P} - \overline{F_1F_2}) \quad (3)$$

The subtraction of the constant factor $\overline{F_1F_2}$ ensures, that points on the line segment have distance zero. The factor $\frac{1}{2}$ is multiplied for the compatibility with the Euclidean distance such that $CED(P, Q) = d(P, Q)$ for points P, Q .

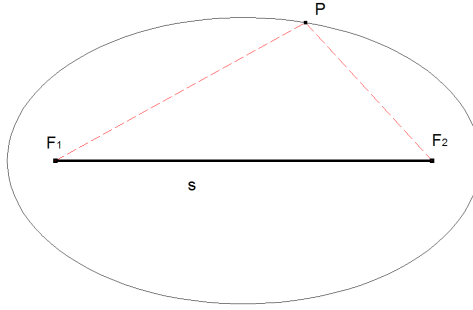


Figure 4: Isoline Ellipse for fixed Confocal Elliptic Distance on line segment s .

6 Confocal Elliptic Voronoi Diagram

In the following we will refer to the Voronoi Diagram on Confocal Elliptic Distances as the *Confocal Elliptic Voronoi Diagram* (*CEVD*). To create the *CEVD* on a seed set \mathbb{S} of points and line segments, we define the Voronoi Cells VC_i for a given seed s_i as the set of points P in the plane with the confocal elliptic distance to s_i smaller than to any other seed

$$VC_i = \{P \in \mathbb{R}^2 : CED(s_i, P) < CED(s_j, P), s_j \in \mathbb{S}\} \quad (4)$$

similar to the definition of the Voronoi Cells in the classic Voronoi Diagram. The union of points with $CED(s_i, P) = CED(s_j, P)$ for two different seeds s_i, s_j are the separation curves of the *CEVD*.

7 Properties of the Separation Curves

7.1 Two symmetric line segments

We consider two symmetric line segments s_i ($i = \{1, 2\}$) with endpoints F_{i1}, F_{i2} , where one can be transformed into the other by reflection on an axis α . The points P on the axis have the property that the Euclidean distance to two separate endpoints is identical. $d(F_{11}, P) = d(F_{21}, P)$ and $d(F_{12}, P) = d(F_{22}, P)$. Therefore $CED(s_1, P) = CED(s_2, P)$ and the reflection axis α becomes the separation line. Figure 5a shows an example of two symmetric line segments and their separation line.

However as shown in figure 9a, in some cases of two intersecting line segments two distinct reflections using two different axes can be used to transform one line segment into the other and thus both axes are separation lines of the $CEVD$.

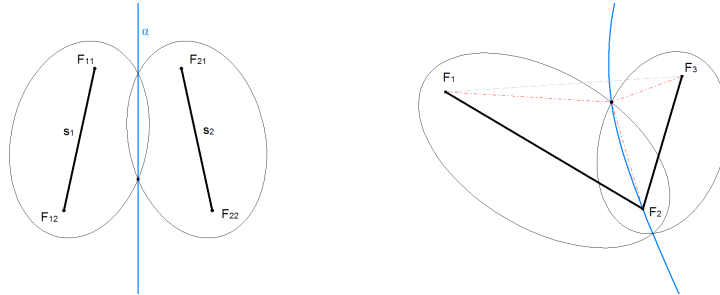
7.2 Two Line Segments Sharing One Endpoint

In the next case two line segments $s_1 = (F_1F_2)$ and $s_2 = (F_2F_3)$ share one identical endpoint. The points on the separation curve can be described as

$$\begin{aligned} CED(s_1, P) &= CED(s_2, P) \\ \frac{1}{2} \cdot (\overline{F_1P} + \overline{F_2P} - \overline{F_1F_2}) &= \frac{1}{2} \cdot (\overline{F_2P} + \overline{F_3P} - \overline{F_2F_3}) \\ \overline{F_1P} - \overline{F_3P} &= \overline{F_1F_2} - \overline{F_2F_3} \end{aligned} \quad (5)$$

As the right side of the equation is constant, using the hyperbola equation $|\overline{XF_1} - \overline{XF_3}| = 2a$, we can see that the separation curve is a single branch of a hyperbola with focal points F_1 and F_3 .

Figure 5b shows an example of the hyperbolic separation curve.



(a) Two symmetric line segments and (b) Two line segment sharing an endpoint in the $CEVD$.

Figure 5: Examples of the separation curves between line segments.

7.3 Triangle

Let ABC be a triangle formed by three line segments. The separation curves of such a triangle hold multiple interesting properties. As shown in section 7.2 the separation curve of two consecutive line segments is a hyperbola branch, thus the triangle is separated by three different hyperbola branches through each corner A, B, C intersecting the opposite line segments in K, L, M .

The hyperbola branches meet in a common point, known in the literature as the *Equal Detour Point* (EDP) [4]. If we consider the complementary branches of the hyperbolas, they too meet in a common point, known as the *Isoperimetric Point* (IP) [4], shown in figure 6a.

The hyperbola tangents in A, B, C are the angular bisectors of their incident line segments and all meet in the incenter I of the triangle [4]. The hyperbola tangents in K, L, M intersect the respective line segment in a right angle and also meet in the incenter I , shown in figure 6b [4].

The hyperbola chords AK, BL and CM meet in a common point known as the *Gergonne Point* (G) shown in figure 6c [4].

The four point EDP, IP, I and G are harmonically collinear, see figure 6d [4].

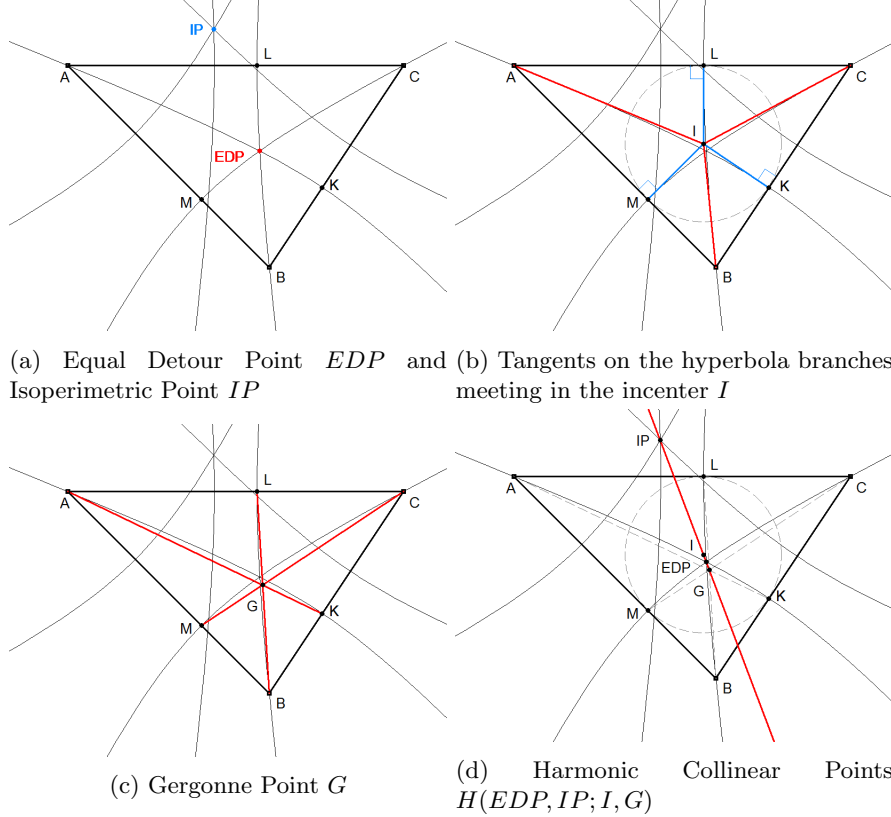


Figure 6: Geometric properties of $CEVD$ on a triangle of three line segments.

8 Other Properties of the CEVD

8.1 Two Line Segments of Considerable Difference in Length

If we take the distance field of two line segments in close proximity where one line segment is significantly longer than the other, the Voronoi Cell of the shorter line segment will be a closed area. This means that points that are in front of the short line segment and are perceived closer to it than to the long line segment in terms of the minimal Euclidean distance, might still have a smaller confocal elliptic distance to the long segment. This property might for example be useful for collision detection of robots, where larger obstacles in the background are a bigger threat than smaller obstacles right in front of the robot.

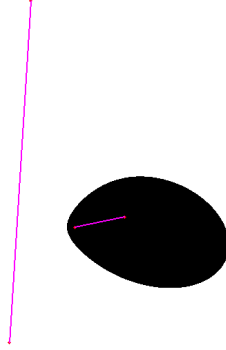


Figure 7: Voronoi Cells of two line segments of considerable length difference.

8.2 Collinear Line Segments

Considering two line segments $s_i = F_{i1}F_{i2}$ and $s_j = F_{j1}F_{j2}$ where s_i lies completely within the line segment s_j , every point $P \notin s_i$ has a smaller confocal elliptical distance to the enclosing line segment s_j , or $CED(s_j, P) < CED(s_i, P)$. The Voronoi Cell of s_i is completely swallowed by the cell of s_j . The same happens for multiple consecutive collinear line segments. This property has an effect on the Confocal Elliptic Fields for different segmentations of a line segment. The same effect can occur on non-collinear line-segments, if polygons are created by sampling arbitrary curves at different resolutions.

Figure 8 shows an example of a line segment arrangement and their isolines of equal distance value as well as the isoline for the complete combined line segment. The isoline ellipses of the inner segments always lie within the isoline ellipse of the complete segment, thus points in the plane will always have smaller confocal elliptic distance to the complete segment than to any of the subsegments.

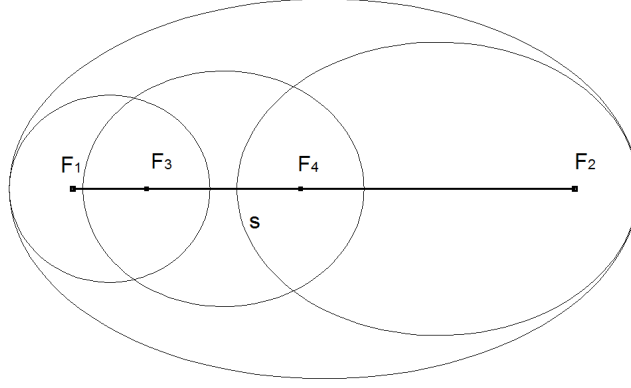


Figure 8: Isolines for specific distance of consecutive line segments inside of isoline of combined segment.

8.3 Intersecting Line Segments

If two line segments s_i and s_j intersect each other, the resulting separation curve(s) will go through the intersection point. Depending on the length difference of the line segments, the separation curve(s) will either resemble an x-shape, ∞ -shape or α -shape.

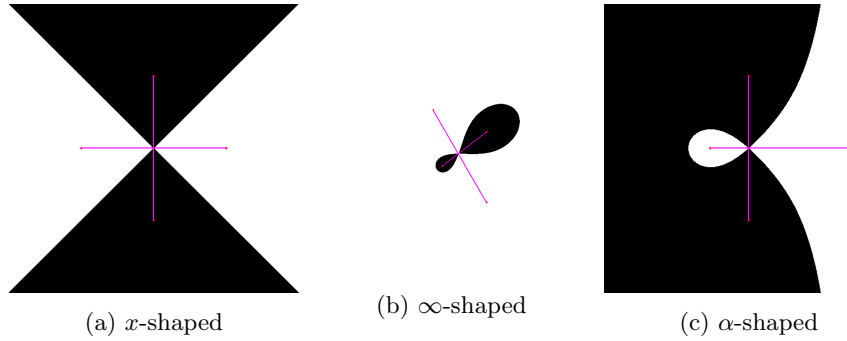


Figure 9: Shape examples of separation curves of intersecting line segments.

9 Conclusion

Using the *Confocal Elliptic Distance*, we created an extension of the *Voronoi Diagram* for seed sets \mathbb{S} consisting of points and/or line segments called the *Confocal Elliptic Voronoi Diagram (CEVD)*. The resulting *CEVD* is compatible with the classic Voronoi Diagram such that if applied to a seed set \mathbb{P} of points only, the *CEVD*(\mathbb{P}) is identical to the classic Voronoi Diagram *VD*(\mathbb{P}) on the same seed set using Euclidean distances.

We gave an overview of the concept of the *Confocal Elliptic Coordinates* as a theoretical basis for the *CED* and an overview of the *Hausdorff Distance* and its derivations as an alternative distance measure to the *CED* and showed the advantage of the *CED* if applied to line segments sharing an endpoint.

Finally we showed several geometric properties of the *CEVD* and its separation curves in particular and we gave an in depth view on the geometric properties of the *CEVD* created by a seed set containing three line segments forming a triangle.

References

- [1] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.
- [2] Felix Hausdorff. *Mengenlehre*, pages 41–351. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [3] David Hilbert and Stephan Cohn-Vossen. *Anschauliche Geometrie*, chapter 1, page 5. Springer-Verlag, 2011.
- [4] GR Veldkamp. The isoperimetric point and the point(s) of equal detour in a triangle. *The American Mathematical Monthly*, 92(8):546–558, 1985.

Artificial Video Dataset for Articulated Joint-Movement

Maximilian Langer (1226991)

July 1, 2018

Abstract

There exist numerous datasets for pose estimation for human action, but general joint datasets are in short supply. The proposed dataset contains different layers of abstraction for temporal, articulated joint-movement: simple binary abstract objects, abstract objects and realistic 3D objects. The different layers allow for evaluation of methods on different stages of generalization.

Additionally the dataset was used to train a deep neural network and evaluate whether it is possible to learn the joint position in a video context.

1 Introduction

Deep Neural Networks getting a lot of attention in recent years and show promising results in various areas of pattern recognition and artificial intelligence. For those algorithms to give good results vast amounts of training data is needed and not readily available for a lot of tasks. There exists a lot of data for human joint annotation that was already successfully applied using deep learning [4, 5, 8, 12] but all these methods take only single images into account and are fine tuned to work on humans only. For many tasks it would be beneficial to be able to use a more general joint articulation that is not fixed on a single skeleton or shape. A lot of other pose estimation problems (animals, robots) could be based on such generalized data.

This work introduces a new dataset that exists to train computer vision algorithms to find articulated joint movement in different degrees of abstraction.

1.1 Articulated Joint Movement

Articulated joint movement occurs when two rigid bodies are connected at a common point that allows for angular change between those bodies. The connection point (joint) does not move relative to all points on the two bodies, it is in the rigid system of both.

The term *articulated* is used to empathize on the rigidity of the attached bodies, the movement of the bodies is constrained by the connecting point. These bodies cannot be infinitely small compared to a soft body joint (e.g. a worm).

There exists methods taking articulated joint movement into account to solve pose estimation [1, 13] but those methods rely on a model of the shape the pose estimation is given for or are tailored to human pose estimation.

2 Dataset

The dataset is divided into three levels of generalization and can be used with human pose video datasets.

2.1 Human Pose Video Datasets

There exist two public datasets that may be used with the introduced dataset to add additional data. The datasets only consist of human actions and not only articulated joints are marked (e.g. hands).

Penn Action The Penn Action dataset [15] consists of around 2300 image sequences in 15 different actions. They are mainly sport shots and contain one or more persons.

JHMDB The JHMDB dataset [9] consists of around 1000 video sequences in 21 different actions. Some categories are not useful for articulated joint detection (e.g. *chew* action).

2.2 Layered dataset

The proposed dataset consists of three layers: *simple*, *artificial* and *animal*.

2.2.1 Simple

The *simple* part of the dataset is a compilation of white stick figures on black background. Points are connected into a tree and some points are defined as joints. One joint is the tree root and all branches are rotated smoothly. This rotation is also done on all other joint-marked points further down the tree. Additionally, the whole structure may be moved and rotated globally. See Figure 1 for a sample sequence.

By including connections that look like joints given only one image, the video data is needed to derive a valid result. The data is very basic, no texture or other information is given to confuse a learning algorithm. The additional global rotation and movement avoids to only look for static points in the image. To learn the correct joint labeling, an algorithm has to look to its local neighborhood to learn which parts are rigid components.

This part consists of 1000 black and white image sequences (800x800x1) with 80 frames each. Thickness of the lines and number of points and joints vary.

2.2.2 Artificial

The *artificial* part of the dataset consists of 3D rendered simple forms that sometimes have texture, complex non-articulated backgrounds or objects in the foreground or a combination of these attributes. The objects were produced using the 3D-Software Blender ¹ and all articulated joints were derived from 3D-armature joints. See Figure 2 for an example sequence.

¹<http://blender.org>

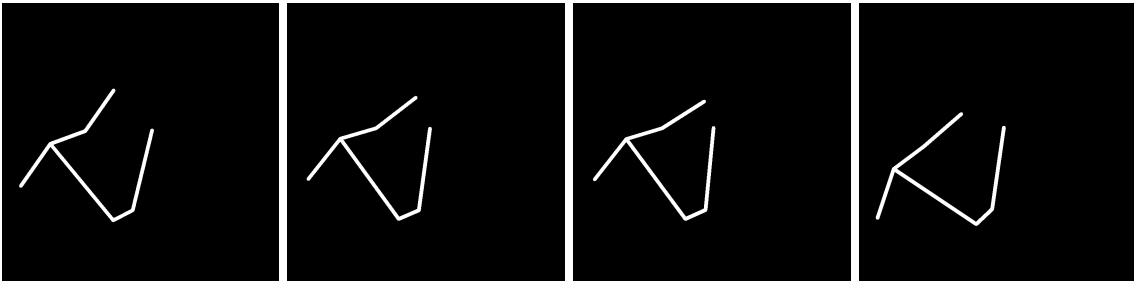


Figure 1: Example for a sequence of images in the *simple* part of the dataset. Images are spaced five time steps apart. Note how only two corners have articulated joint movement.

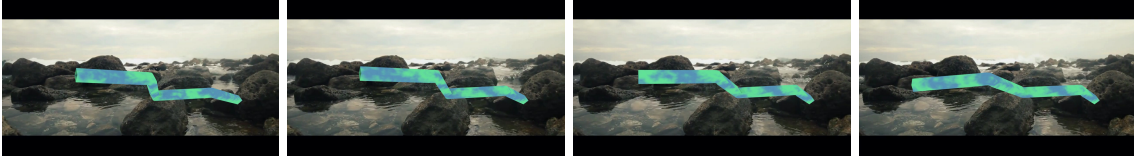


Figure 2: Example for a sequence of images in the *artificial* part of the dataset. Images are spaced five time steps apart.

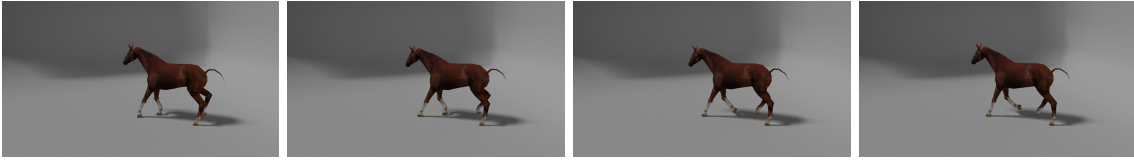


Figure 3: Example for a sequence of images in the *animal* part of the dataset. Images are spaced five time steps apart.

Articulated joints are easily recognized by humans in this part of the dataset. Some sequences include multiple objects. Backgrounds have been downloaded from the Internet ² licensed under CC0 ³.

This part consists of 36 color image sequences (1280x720x3) with 140 frames each where 3 image sequences are multiple shots from different views on the same object and 15 image sequences have complex video backgrounds.

2.2.3 Animal

The *animal* part of the dataset shows different 3D-rendered animals performing a walk cycle. In contrast to the *artificial* part the *animal* part has no complex backgrounds and occlusion happens solely by the object itself. On the other hand are joints harder to see, because they happen on a bone level and might be off-centered (flesh/muscles). See Figure 3 for an example sequence.

This layer of generalization is used to learn more realistic occurrences of articulated joint movement in nature. Models have been downloaded from the Internet ⁴ and licensed under CC0 and CC-BY⁵.

This part consists of 15 color image sequences (1280x720x3) with 120-240 frames each where 3 image sequences are multiple shots from different views on the same object.

3 Experiments

The experiments were performed using two different neural networks: U-Net and a simple convolutional neural network (CNN). Both methods use a heatmap to train the joint location as likeliness at a given pixel and are trained on Keras [6] using Mean Squared Error as loss function and Adam as optimizer [10]. The dataset was split into 80% for training and 20% for validation and testing.

²<https://videos.pexels.com>

³<https://creativecommons.org/publicdomain/zero/1.0/>

⁴<https://www.blendswap.com>

⁵<https://creativecommons.org/licenses/by/3.0/>

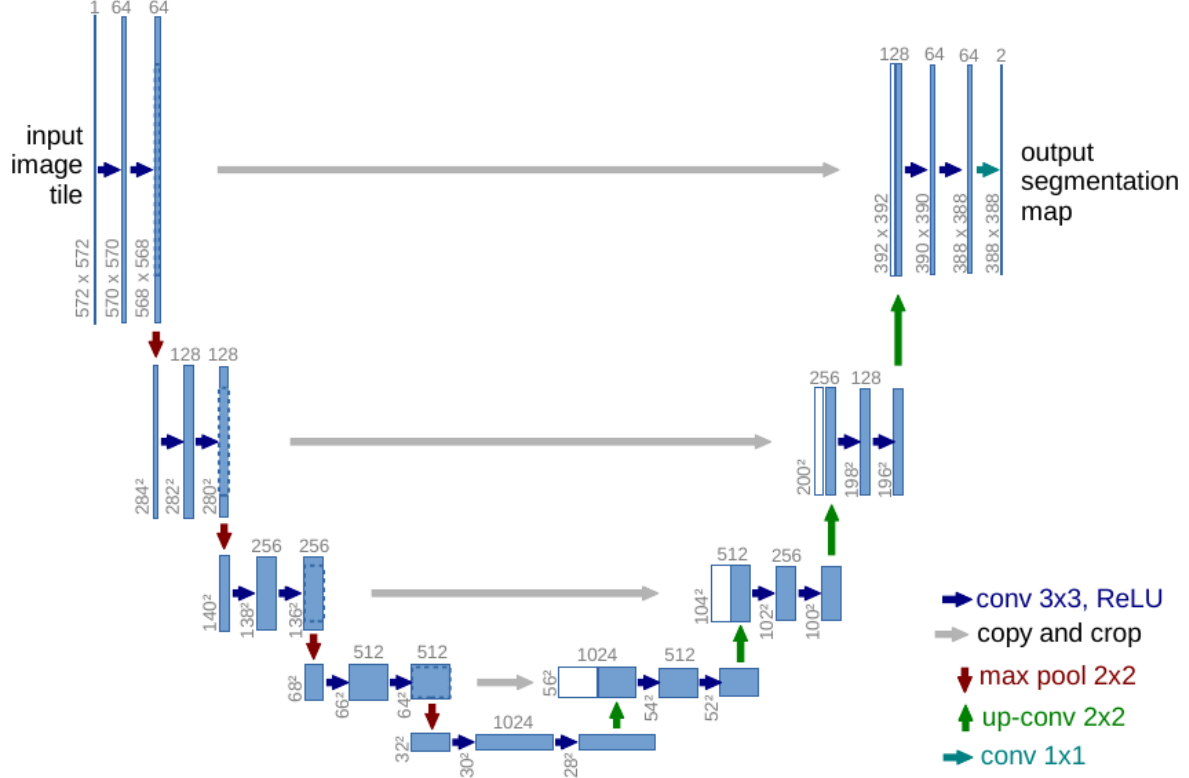


Figure 4: U-Net architecture. Image taken from [14].

3.1 U-Net

As a first architecture a U-Net implementation was used [14]. U-Nets are a special CNN that upscales lower levels and combining them with upper levels again to obtain a same-sized output from an input image (see Figure 4). The model was modified slightly to be able to work with image sequences through two different methods: 3D convolution and 2D convolution with time distributed overlay (i.e. 2D convolution for each frame). The input is temporal centered around a frame f_t , taking after a pyramid fashion frames $f_{t-10}, f_{t-5}, f_{t-3}, f_{t-1}, f_t, f_{t+3}, f_{t+5}$ and f_{t+10} . The number of frames (8) has to be even for proper upscaling in the U-Net framework. In the end a convolution over the time axis is performed to get a final heatmap for the centered frame. Images were resized to 128x128 pixels beforehand.

3.2 CNN

As second architecture a simple CNN was used, that uses 3D convolution and up-sampling to obtain a one-image-sized output image from an input image sequence. The same input sequence and resolution as with the U-Net was used. See Figure 5 for an architecture overview.

3.3 Results

The following experiments were undertaken:

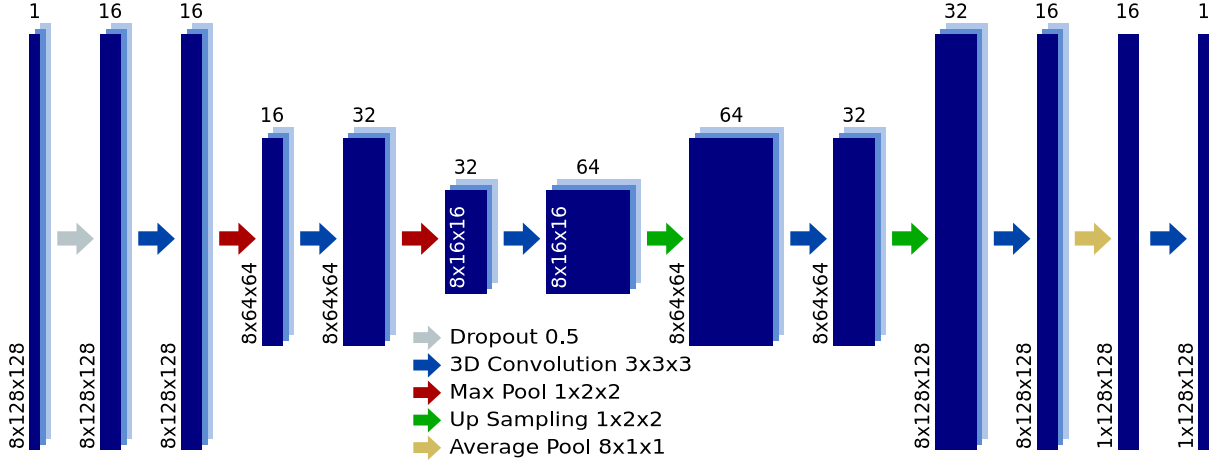


Figure 5: CNN architecture used for experiments.

1. U-Net 3D Conv, 3D parts of dataset and Penn Action
2. U-Net 3D Conv, 3D parts of dataset
3. U-Net 3D Conv, *Artificial* part of dataset
4. U-Net 3D Conv, Subset of *artificial* part of dataset
5. U-Net 3D Conv, *Simple* part of dataset 256x256
6. U-Net Time Distributed, *Artificial* part of dataset
7. U-Net Time Distributed, *Artificial* part of dataset 512x512
8. CNN, 3D parts of dataset and Penn Action
9. CNN, *Artificial* part of dataset

Results for most experiments are similar: The network was not able to learn joints of the image sequences. In experiment 4 only 15 out of the 36 image sequences were used for training. The result did not improve with all image sequences.

Training only with the artificial part of the dataset improves the result compared to adding Penn Action, concentrating the heatmap on the object and not outside although peaks are not on the joint positions in any case.

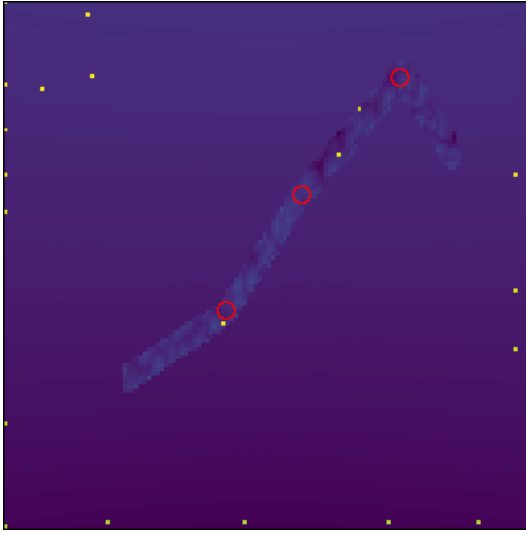
Experiment 5 returns in all cases corner points as highest value on the heatmap.

See Figure 6 for example results of the test set.

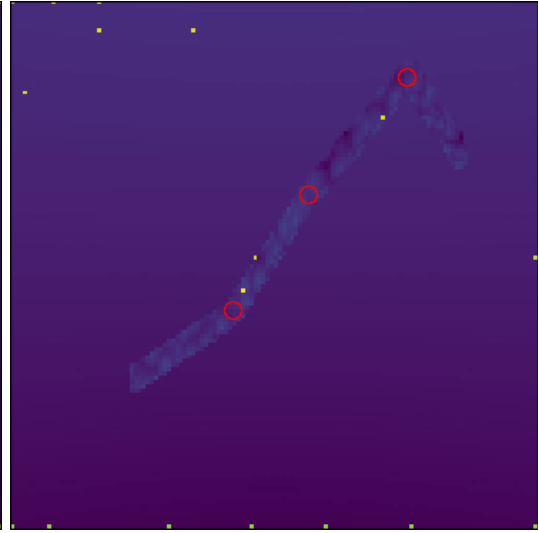
4 Future Research

Although the results are not promising, the work raises the question which - if any - neural network can solve the dataset. Especially the *simple* part of the dataset poses an interesting challenge because it cannot be solved with single frames (as opposed to the rest of the dataset). The network would have to consider the local neighborhood in a temporal sequence of images and relate the structure of the object.

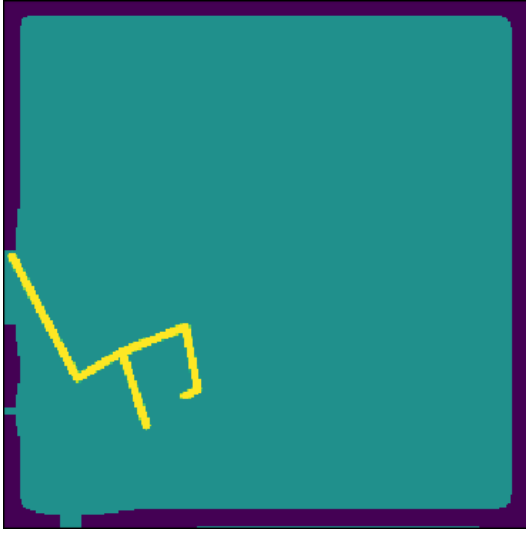
For solving the other parts of the dataset (more application relevant) other approaches including neural networks (Transfer learning [11], LSTM [7]) and other methods (random forest [3], triangulation [2]) can be utilized. Also the use of still images instead of videos can be considered, as current state-of-the art methods use still images only [4, 5, 8, 12].



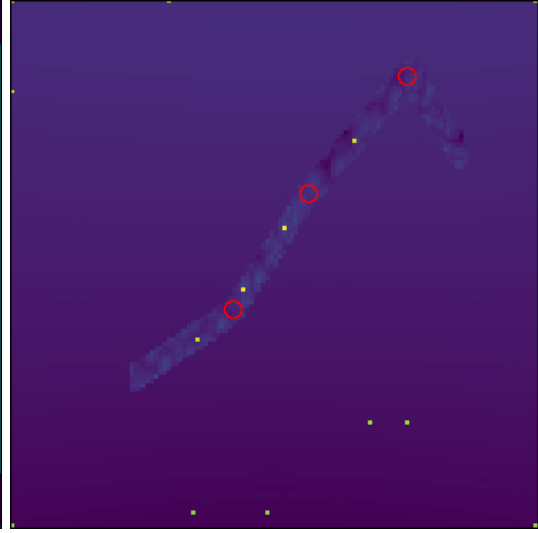
(a) Experiment 1



(b) Experiment 3



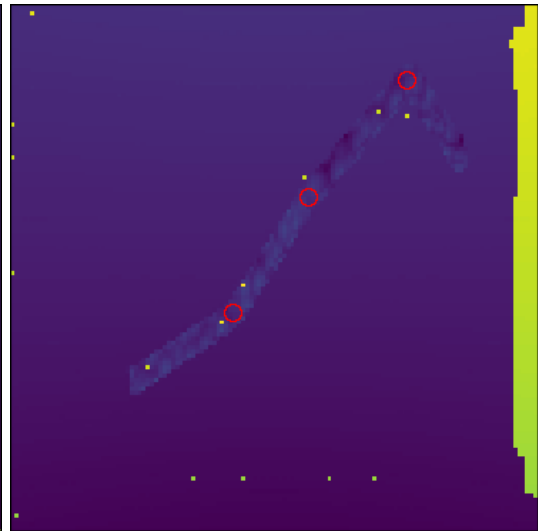
(c) Experiment 5



(d) Experiment 6



(e) Experiment 7



(f) Experiment 9

Figure 6: Example results for some of the experiments. Yellow points correspond to predicted joints (local maxima in heatmap) and red rings to ground truth. In Experiment 5 all cyan pixel and yellow pixel have the same value ($=0$) in the center.

References

- [1] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.
- [2] Nicole M. Artner, Adrian Ion, and Walter G. Kropatsch. Hierarchical spatio-temporal extraction of models for moving rigid parts. *Pattern Recognition Letters*, 32(16):2239–2249, 2011.
- [3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, volume 1, page 7, 2017.
- [5] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4733–4742, 2016.
- [6] François Chollet et al. Keras. <https://keras.io>, 2015.
- [7] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.
- [8] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepcut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.
- [9] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *International Conf. on Computer Vision (ICCV)*, pages 3192–3199, December 2013.
- [10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [11] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [12] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016.
- [13] Deva Ramanan. Learning to parse images of articulated bodies. In *Advances in neural information processing systems*, pages 1129–1136, 2007.
- [14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [15] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2248–2255, 2013.

Improved Features for Hypocotyl/Root Transition Detection in Arabidopsis Thaliana

Julian Strohmayr

Abstract

This work proposes two new features for the detection of the hypocotyl/root transition point in Arabidopsis Thaliana seedlings for phenotyping applications. The proposed features exploit color and intensity variations along the plant body to capture the gradual transition from hypocotyl to root. The performance of both features is evaluated using data, produced by an automatic phenotyping pipeline.

1 Introduction

In order to analyze the process of plant growth under controlled environmental conditions and to monitor the response of a plant to certain compounds in the soil, large scale plant phenotyping experiments are conducted. These endeavors are common practice in the field of agricultural plant breeding, where crop plants are engineered to be more efficient. Growth under challenging conditions, robustness against pests or increased yield are common goals. A typical plant phenotyping experiment is conducted by growing a large number of seedlings on agar plates, which are photographed by an image acquisition setup to acquire a high resolution image of the plates. Plate images are then processed by a software stage that measures/computes characteristics of individual seedlings, which are the basis of further analysis. Because of the large amount of data that has to be processed in these experiments nowadays, automatic phenotyping pipelines [4][3][2] have been developed that require little to no human intervention. These automatic phenotyping pipelines offer significantly higher throughput than manual methods, since the human bottle neck is eliminated. However, the benefits of these automatic phenotyping pipelines only show when the produced measurements are correct, which is not always possible. There is still a need for robust detection/segmentation methods for certain problems in the field of phenotyping. One of these problems is the detection of the hypocotyl/root transition point, discussed in this work. The hypocotyl/root transition point is the interface between the stem and the root of the plant embryo. This measurement has to be as precise as possible, because it is the basis for subsequent length measurements along the plant axis. The detection of this point is not a trivial task due to many confounding factors, such as bad segmentation, abnormal plant growth or the presence of foreign objects on the plate. This work presents two new features, capable of improving the hypocotyl/root transition detection performance of the approach described in [5].

2 Previous Work

This is a continuation of previous work [5], in which a novel method for the detection of the hypocotyl/root transition point in *Arabidopsis Thaliana* seedlings was presented. The described approach uses features, derived along the plant axis of a seedling, strongly correlated with the transition from hypocotyl to root to detect the transition point. It is based on the assumption that the hypocotyl/root transition point is the point on the plant axis where these correlated features diverge the most. A more detailed description of the approach is given in Section 4, as well as in the original work.

3 Data

The performance of the proposed features is evaluated using data from the GMI phenotyping pipeline, presented by Slovak et al. [4]. The data set consists of individual agar plate scans with a corresponding measurement file that contains all BRAT-measurements for the image. The entries of this file are 5-dimensional row vectors of the form $(objectId, x, y, width, pixelType)$, where *objectId* is a sequence number for the detected object, *x* and *y* are the plant axis coordinates on the image, *width* is the diameter at position (x,y) and *pixelType* defines the type of segmentation. The data is comprised of two separate sets of plate scans, which will be referred to as data set *A* and *B*. Data set *A* is a time series of a single agar plate with 24 Arabidopsis Thaliana seedlings, scanned once a day, over a period of five days. The resulting .tiff-images have a resolution of 5612x6088 pixels (1200 dpi) and a file size of about 100MB. The plate scans of data set *A* are given by Figure 1 (a)-(e). Data set *B* consists of two different plates with 24 Arabidopsis Thaliana seedlings, both scanned at day 1. The resulting .tiff-images have a resolution of 6608x6614 pixels (400 dpi) and a file size of about 125MB. The plate scans of data set *B* are given by Figure 1 (f)-(g). The combination of both data sets amounts to 167 seedlings in total, which are used for the evaluation (one seedling was not detected by the BRAT-software). For each of these seedlings, ground truth annotation for the hypocotyl/root transition point from 8 independent experts is available.

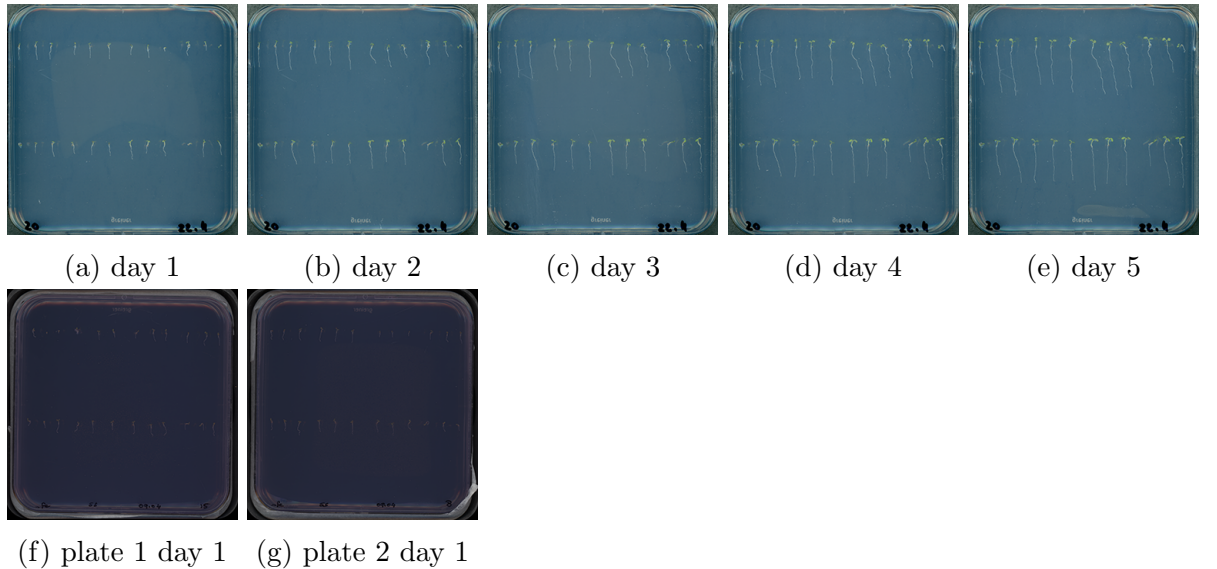


Figure 1: (a)-(e) Data set *A* - Scanned agar plate with 24 Arabidopsis Thaliana seedlings over a series of 5 days. (f)-(g) Data set *B* - Two different agar plates with 24 Arabidopsis Thaliana seedlings at day 1. Slovak et al. [4].

4 Methodology

The detection method for the hypocotyl/root transition point, described in [5], is based on one key assumption. Namely, that there exist some features, derivable along the plant axis of a seedling, which are strongly correlated with the transition from hypocotyl to root. This means that the feature values along the plant axis of such features change abruptly within the transition zone. This abrupt change in value effectively encodes the position of the hypocotyl/root transition point and can therefore be used for detection purposes. An example of such a correlated feature is given by Figure 2 (c), which shows the CIELAB a^* component (red/green-difference). This feature effectively shows the change in greenness along the plant axis. It is easy to see that the hypocotyl is on average much greener than the root, which is both visible directly on the seedling, as well as on the graph. The abrupt drop in greenness coincides exactly with the hypocotyl/root transition point, which was the assumption. This example shows the desirable characteristics of a feature, usable for the detection of the hypocotyl/transition point. In order to find the actual location of the hypocotyl/root transition point on the plant axis, atleast two features that satisfy the before mentioned requirements are needed. When using n features for the detection, a seedling with a plant axis of length m can be seen as n -dimensional multivariate data set of size m . According to the assumption is the hypocotyl/root transition point the point on the plant axis where feature divergence is maximal. Therefore, finding the break point in the covariance structure of this multivariate data set, that is the seedling, should reveal the location of the hypocotyl/root transition point. For the break point detection, the method proposed by Aue et al. [1] is used. The result of the break point detection for a seedling from data set A is given by Figure 2 (c), which shows that the global maximum (green line) coincides with the hypocotyl/root transition point.

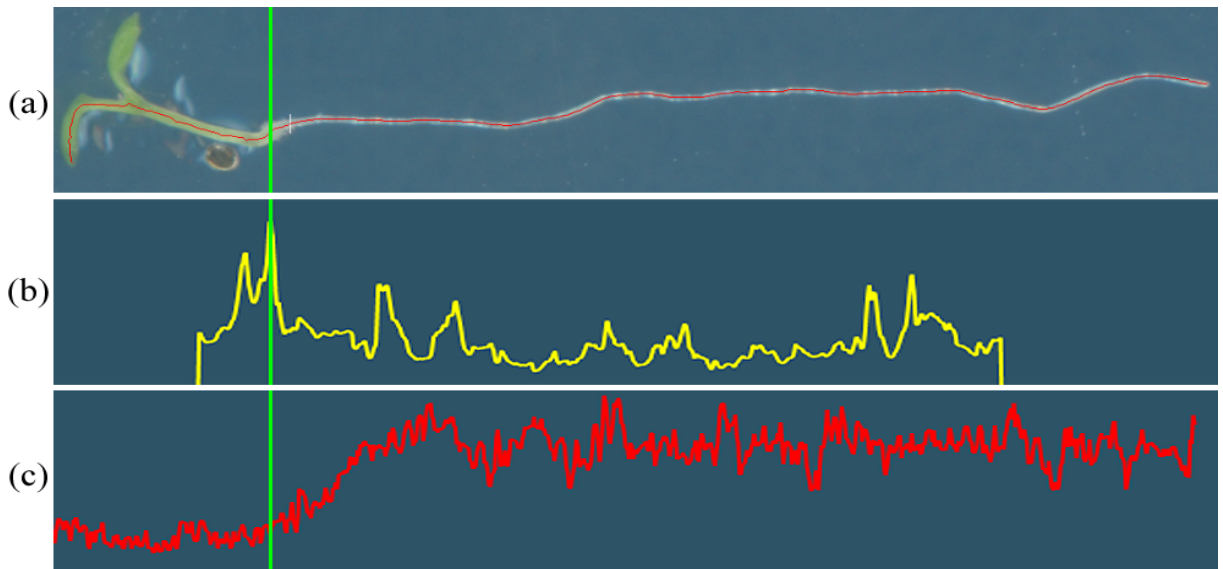


Figure 2: (a) Arabidopsis Thaliana seedling from data set A at day 3. (b) Result of the break point detection and (c) feature (CIELAB a^*), correlated with the hypocotyl/root transition.

5 Features

5.1 Mean "Greenness"

The greenness as feature, given by the CIELAB a^* component performed well in previous experiments [5]. For this reason, a modified version of this feature is presented in this work, which is less noisy and is easier to interpret. This feature can also be interpreted as greenness along the plant axis, however, it is not simply the CIELAB a^* component. The feature value at a given position on the plate scan is computed by negating the CIELAB a^* component and adding the CIELAB b^* component at the position. Negating the CIELAB a^* component simply makes interpretation easier. It is more intuitive to have positive values describing the greenness, because the graph is reflecting what a human observer sees on the image of the seedling. The CIELAB b^* component is added to reward yellow and penalize blue hues along the plant axis. This is based on the observation that the hypocotyl contains large amounts of yellow, whereas the root contains large amounts of blue. Since the negative CIELAB b^* component corresponds to blue, adding it to the negated CIELAB a^* component will increase feature values along the hypocotyl and decrease feature values along the root, effectively making the drop in value at the hypocotyl/root transition point more extreme. The resulting value of this computation is referred to as greenness from now on.

Three different variants of the mean greenness feature were tested, each computing the arithmetic mean of greenness across differently shaped sampling regions to reduce noise. The first variant, given by Figure 3 (b), computes the mean greenness along a horizontal sampling line. The mean greenness value for a point p on the plant axis with image coordinates (p_x, p_y) and width p_w is the arithmetic mean of greenness of all pixels with image coordinates $([p_x - \lfloor \frac{p_w}{2} \rfloor, p_x + \lfloor \frac{p_w}{2} \rfloor], p_y)$. The second variant, given by Figure 3 (c), computes the mean greenness within a sampling window. The mean greenness value for a point p on the plant axis with image coordinates (p_x, p_y) and width p_w is the arithmetic mean of greenness of all pixels in $([p_x - \lfloor \frac{p_w}{2} \rfloor, p_x + \lfloor \frac{p_w}{2} \rfloor], [p_y - \lfloor \frac{p_w}{2} \rfloor, p_y + \lfloor \frac{p_w}{2} \rfloor])$. Finally, the third version, given by Figure 3 (d) samples greenness values within a certain interval along the plant axis and computes the arithmetic mean. The mean greenness value for a point p on the plant axis with plant axis index p_i and width p_w is the arithmetic mean of greenness of all plant axis points with index $[p_i - \lfloor \frac{p_w}{2} \rfloor, p_i + \lfloor \frac{p_w}{2} \rfloor]$. As given by Figure 3, all variants perform very similar in this example, with the window and axis based versions being less noisy by an insignificant amount. As expected, the drop in mean greenness coincides with the hypocotyl/root transition point.

Considering that all variants perform equally well, computing the window based version only wastes computational resources, since this variant has to sample p_w^2 pixels, instead of p_w . Another drawback of the horizontal and window based variants is the lack of robustness against foreign objects in the image. This problem is given by Figure 4. Horizontal and window based variants both sample pixels outside of the plant axis, which can lead to dips/spikes in greenness in the presence of foreign objects with strongly diverging color. The axis based variant is not affected by this problem, because pixels are only sampled along the plant axis. Therefore, the axis based variant is superior.

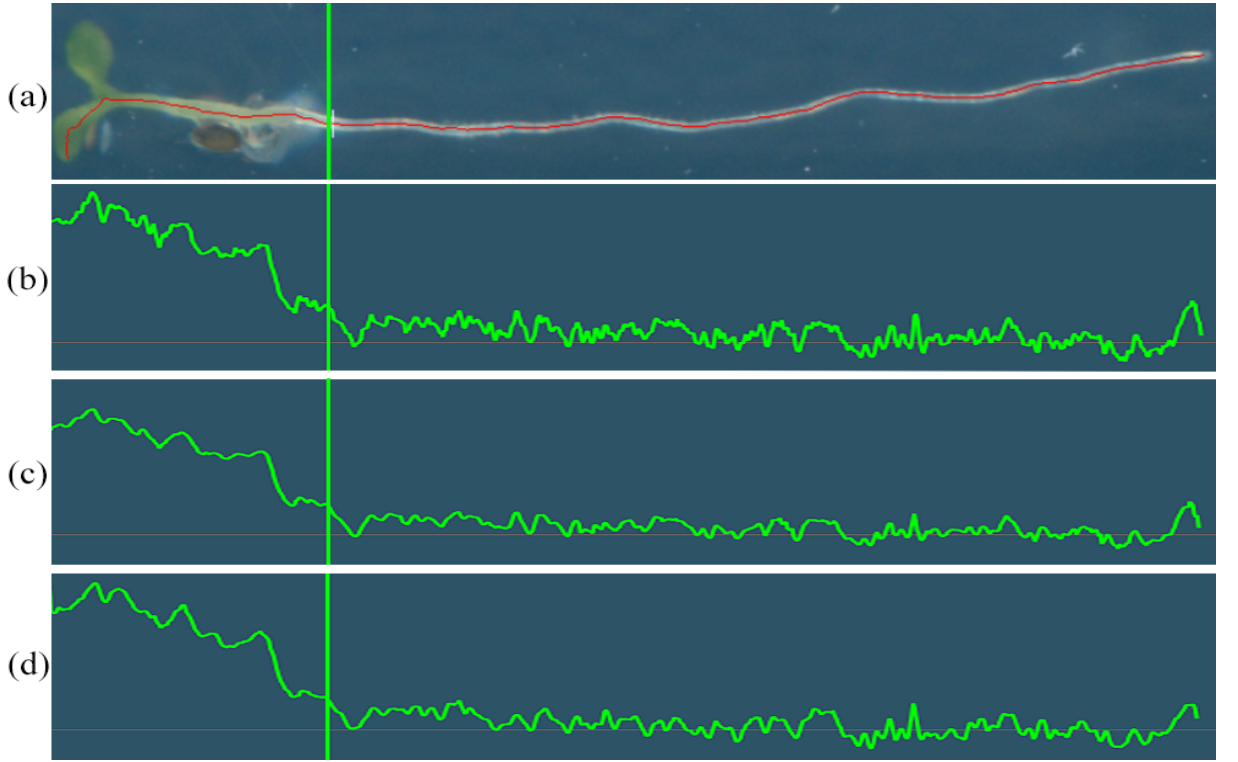


Figure 3: (a) *Arabidopsis Thaliana* seedling from data set A at day 3 (b) mean greenness (horizontal) (c) mean greenness (window) (d) mean greenness (axis)

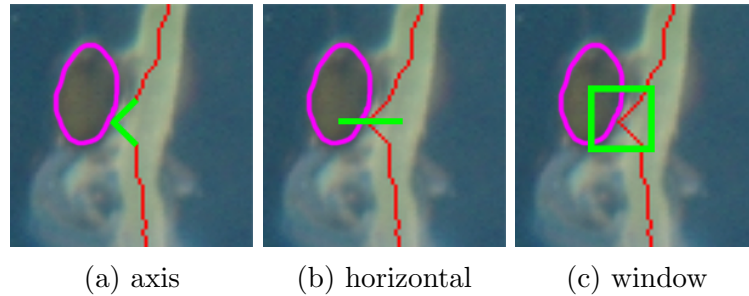


Figure 4: Example, showing the superiority of (a) the axis based variant of the mean greenness feature in the presence of foreign objects, over the (b) horizontal and (c) window based variants.

5.2 Smoothed Standard Deviation of Intensity (SSDI)

The standard deviation of intensity (CIELAB L*) within the hypocotyl is lower than it is within the root. Therefore, this phenomenon can be exploited for detection purposes. However, previous experiments have shown that the standard deviation of intensity along a horizontal line is not a suitable feature for the detection of the hypocotyl/root transition point due to heavy value fluctuations, caused by large intensity variations along the root [5]. These fluctuations effectively create unwanted breaks in the multivariate data set, which if large enough, could result in a global maximum that does not coincide with the hypocotyl/root transition point. This problem can be avoided by simply convolving the raw standard deviation values with a one-dimensional Gaussian kernel. The kernel size is dynamically chosen with respect to the length of the plant axis. A plant axis of length n is convolved with a kernel of size $s = \lfloor \frac{n}{10} \rfloor$. Again, three different variants of the SSDI, with the same sampling regions (axis, horizontal and window) as used by the already mentioned mean greenness feature were tested. Typical SSDI responses for all three variants are given by Figure 5. Comparison of all three graphs reveals that the axis based SSDI variant produces a flatter response than the other variants. This is due to the fact that the root has a characteristic drop in intensity at the center of its horizontal intensity profile, which the hypocotyl does not have. By sampling only along the plant axis, this characteristic cannot be captured which leads to a feature less correlated with the transition from hypocotyl to root. The horizontal SSDI is for this reason and the already mentioned performance argument against the window based variant superior.

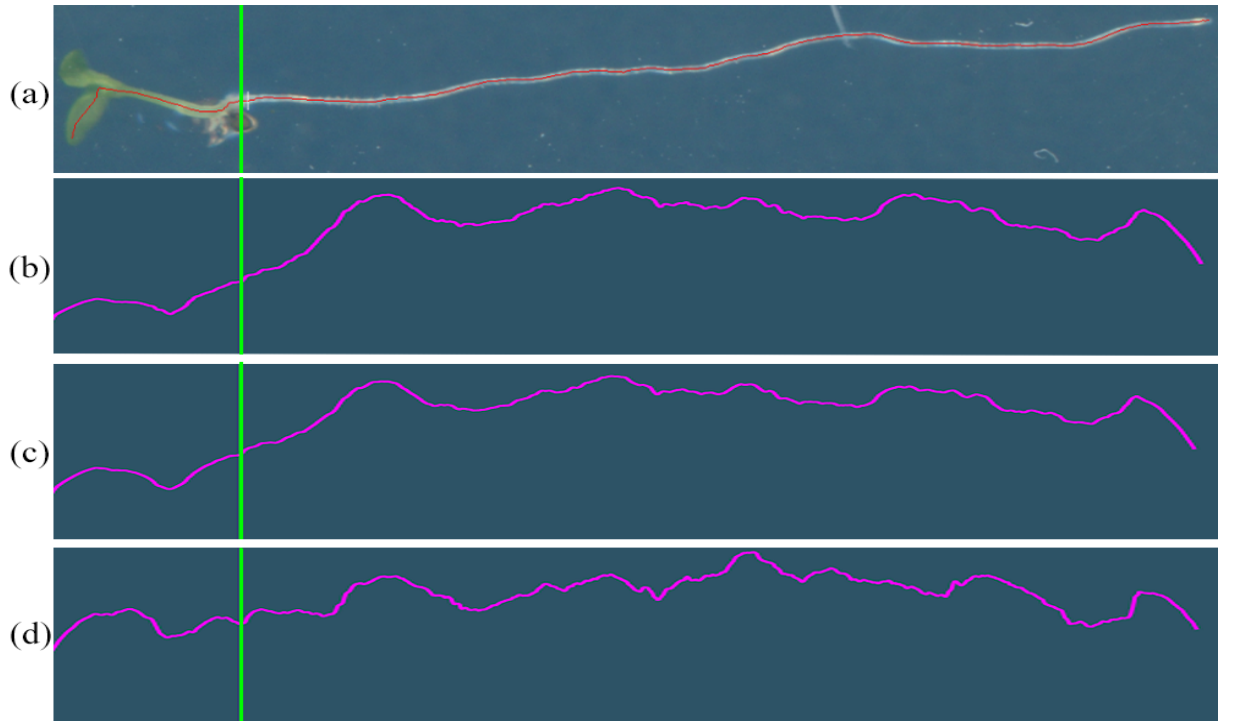


Figure 5: (a) Arabidopsis Thaliana seedling from data set A at day 3 (b) SSDI (horizontal) (c) SSDI (window) (d) SSDI (axis)

6 Evaluation Results

The axis based variant of the *mean greenness* and the horizontal variant of the *SSDI* were chosen for the final evaluation, due to the already mentioned drawbacks of the other variants. The evaluation was carried out using the visual analytics framework PlateViewer [5], which implements the detection method described in Section 4. The detection capability of the used method, combined with the proposed features, is compared to BRAT, the software stage of the automatic phenotyping pipeline, presented by Slovak et al. [4]. BRAT supports automatic detection of the hypocotyl/root transition point as well and can therefore be used as benchmark. The performance metric in this evaluation is the minimal absolute distance along the plant axis, between the detected hypocotyl/root transition point and the ground truth label. The method, being closest to the ground truth label (i.e. having a smaller error), is considered superior for a given case. To get a performance baseline for the evaluation, all features from [5] were used. These are plant *width*, *CIELAB a**, *CIELAB b** and *shape*. The evaluation results for these features, given by Table 1, shows that the used method outperformed BRAT in 55.1% of all cases. In order to rule out harmful interference between the proposed features, both were evaluated separately by adding them to the baseline feature set. The results for the axis based *mean greenness* are given by Table 2. Adding this feature to the feature set improved the detection performance by 4.8%. The horizontal SSDI performed even better. As given by Table 3, the used feature set could outperform BRAT in 64.1% of all cases, which amounts to an increase in detection performance by 9%.

Table 1: Baseline evaluation results for the features *width*, *CIELAB a**, *CIELAB b** and *shape* described in [5].

	Number of Seedlings	PlateViewer	BRAT
<i>data set A, day 1</i>	24	12	12
<i>data set A, day 2</i>	24	11	13
<i>data set A, day 3</i>	24	13	11
<i>data set A, day 4</i>	24	12	12
<i>data set A, day 5</i>	24	17	7
<i>data set B, plate 1</i>	23	13	10
<i>data set B, plate 2</i>	24	14	10
sum	167	92	75
		55.1%	44.9%

Table 2: Evaluation results for the proposed *mean greenness (axis)*, combined with the features *width*, *CIELAB a**, *CIELAB b** and *shape* described in [5].

	Number of Seedlings	PlateViewer	BRAT
<i>data set A, day 1</i>	24	13	11
<i>data set A, day 2</i>	24	12	12
<i>data set A, day 3</i>	24	18	6
<i>data set A, day 4</i>	24	16	8
<i>data set A, day 5</i>	24	11	13
<i>data set B, plate 1</i>	23	15	8
<i>data set B, plate 2</i>	24	15	9
sum	167	100	67
		59.9%	40.1%

Table 3: Evaluation results for the proposed *SSDI (horizontal)*, combined with the features *width*, *CIELAB a**, *CIELAB b** and *shape* described in [5].

	Number of Seedlings	PlateViewer	BRAT
<i>data set A, day 1</i>	24	13	11
<i>data set A, day 2</i>	24	14	10
<i>data set A, day 3</i>	24	20	4
<i>data set A, day 4</i>	24	15	9
<i>data set A, day 5</i>	24	15	9
<i>data set B, plate 1</i>	23	16	7
<i>data set B, plate 2</i>	24	14	10
sum	167	107	60
		64.1%	35.9%

7 Discussion

The evaluation results in Section 6 clearly show that the proposed features increased detection performance by a significant amount. Both the *mean greenness*, as well as the *SSDI* are well suited features for the detection of the hypocotyl/root transition point in *Arabidopsis Thaliana* seedlings. However, further testing is necessary to confirm these results. The data set used for the evaluation is limited in terms of size and variety of nutrient solutions. It already contains two different types of nutrient solutions, distinguishable by the blue/brownish background color. Nevertheless, a wider variety of nutrient solutions need to be tested, in order to determine the robustness of the *mean greenness* against smaller color variations, caused by differently colored backgrounds. The root of a seedling is slightly transparent, which means the color of the nutrient solution in the background will affect the color composition slightly. This could lead to detection problems, because the drop in greenness within the hypocotyl/root transition zone will be smaller and the feature less descriptive. The *SSDI* also requires some optimization. Currently, the chosen kernel size of the smoothing kernel is set to 10% of the plant axis length. This approach is somehow dynamic, but still a choice based on observations, which is obviously not ideal.

A solution for finding the optimal kernel size for a given plant axis length could possibly further improve the performance of the SSDI as a feature.

8 Conclusions

In this work, the *mean greenness* and the *smoothed standard deviation of intensity (SSDI)* were presented as new features for the detection of the hypocotyl/root transition point in *Arabidopsis Thaliana* seedlings. The detection capabilities of both features were evaluated using a set of plate scans with ground truth labeling for the hypocotyl/root transition point for each seedling on the plate. Evaluation results showed that the *mean greenness* improved detection performance by 4.8% and the *SSDI* improved detection performance by 9%, when compared to the previously used feature set.

References

- [1] A. Aue, S. Hörmann, L. Horvth, and M. Reimherr. Break detection in the covariance structure of multivariate time series models. *Ann. Statist.*, 37(6B):4046–4087, 12 2009.
- [2] A. Hartmann, T. Czauderna, R. Hoffmann, N. Stein, and F. Schreiber. Htphen: An image analysis pipeline for high-throughput plant phenotyping. *BMC Bioinformatics*, 12(1):148, May 2011.
- [3] A. Kicherer, K. Herzog, M. Pflanz, M. Wieland, P. Rger, S. Kecke, H. Kuhlmann, and R. Tpfer. An automated field phenotyping pipeline for application in grapevine research. *Sensors*, 15(3):4823–4836, 2015.
- [4] R. Slovak, C. Göschl, X. Su, K. Shimotani, T. Shiina, and W. Busch. A scalable open-source pipeline for large-scale root phenotyping of arabidopsis. *The Plant Cell*, 26:2390–2403, 2014.
- [5] J. Strohmayer. A visual analytics approach to hypocotyl/root transition detection in *arabidopsis thaliana*, Jan. 2018.

Tracking Golden-Collared Manakins in the Wild

Anna Gostler, Nicole M. Artner and Walter G. Kropatsch
Pattern Recognition and Image Processing Group
TU Wien, Austria
E-mail: anna_g@prip.tuwien.ac.at

Leonida Fusani
Department of Cognitive Biology
University of Vienna, Austria

I. INTRODUCTION

The golden-collared manakin (*Manacus vitellinus*) is a small tropical bird, which lives in the Panama forest. The males perform elaborate, acrobatic displays to court mates [1]. During its courtship dance the male demonstrates its physical strength by jumping between saplings, producing loud wing snaps mid-flight. Mating success seems to be related to superior motor skills [2], which allow the male to execute its dance faster and more precisely. However, it is not fully clear yet how exactly the courtship dance has to be performed to impress a female. To gain more knowledge about their dance, biologists recorded the birds in the wild with high-speed cameras at 60 fps. One of the videos can be found at ¹. Manually annotating the male bird in every frame of the videos to enable analyzing their behavior is a tedious process. We propose a novel approach for automatic visual tracking of the male golden-collared manakin, combining a convolutional neural network, background subtraction and a Kalman filter.

The following properties of the videos make tracking the birds challenging:

Speed: While jumping, the bird moves very quickly (avg. 28 px per frame; avg. bounding box size: 113x95 px; frame size: 1928x1208 px).

Motion blur: Strong motion blur can make the bird hard to recognize as it loses most of its local features.

Size and shape change: The bird’s bounding box changes in size and shape, e.g. when the bird opens its wings, turns, or moves away from the camera.

Occlusion: The bird can be partly or fully occluded by saplings and leaves.

Out of frame: The bird frequently leaves the camera’s field of view (18 times in 78 videos).

Trajectory: The bird starts and stops abruptly and typically changes direction when starting a new jump. On average, the bird makes 3.7 (max. 10) jumps per video.

Background color: The forest is colored mostly green, yellow and brown – similar to the male bird, which has a green body, black head and a yellow neck.

Background motion: Leaves and branches move in the background. Saplings often move when the bird lands on them.

Tracking is made easier, however, by the static camera setup, i.e. **absence of camera motion**.



Figure 1: Close-ups of male (top row) and female (bottom row) golden-collared manakins

II. RELATED WORK

Most computer vision methods to analyze the behavior of birds worked on videos that were recorded inside custom-built arenas [3]–[7] or on videos of birds flying against the sky or distant landscapes [8], [9]. Therefore, background segmentation and tracking was not a particular concern for these studies. A method to get more precise information about the bird’s position is to equip them with body markers [6]. This could potentially modify the bird’s behavior during courtship, which the biologists wanted to avoid with the manakins.

In 2017, Oliva et al. [10] developed a visual tracker for golden-collared manakin males for videos that were recorded in 2016 by the same team of biologists with a different setup than the videos in this paper. Oliva’s tracker detects foreground blobs using Mixture of Gaussians (MOG) [11] and finds the male manakins among these blobs based on the yellow color and saturation of their neck or, if it cannot find the bird this way,

¹<https://github.com/anna-gostler/ManakinTracker>

predicts its location with a linear Kalman filter. This method relies on a distinct color difference between the background and the bird, which is not present in most of the current videos.

As we aim to track birds that strongly and abruptly change their appearance in videos recorded against a highly cluttered background, we have evaluated the top performing trackers of the VOT2016 challenge [12], which deal with visual tracking under similarly challenging conditions: TCNN [13] and C-COT [14].

Nam et al. [13] developed TCNN, a tracker that uses Convolutional Neural Networks (CNNs) arranged in a tree structure, where a CNN in a child node is a fine-tuned version of the CNN in its parent node. By keeping multiple models of the target object the tracker can handle appearance changes. The CNNs are based on a CNN pre-trained on ImageNet, but are adapted to output only two scores: a target and a background score.

Danelljan et al.’s tracker C-COT [14] is also based on a CNN pre-trained on ImageNet. C-COT extracts feature maps, which consist of the input image patch and convolutional layers, from the CNN and learns continuous convolution filters. The feature maps are convolved with the filters to obtain a continuous confidence score for every location in an area centered on the previous location of the target. This approach uses the CNN as a feature extractor. However, there are not many consistent local features (such as points and edges) in our target, mainly due to motion blur.

Both TCNN and C-COT do not model target movement, but instead search the target around its previous location, so they might not be able to follow a fast moving target such as the manakin.

III. MANAKINTRACKER

In this paper, we propose the ManakinTracker, which

- detects moving objects with a Mixture Of Gaussians model (MOG) [11],
- decides if a candidate location visually resembles the male bird with a fine-tuned CNN,
- and estimates the location of the target using a Kalman filter [15] for frames without reliable visual cues (see Fig. 2).

A. Blob Detection

As the videos were recorded with stationary cameras, we can use a method based on background subtraction to segment the foreground. We chose Mixture Of Gaussians (MOG) because it can handle small movements in the background. For every frame, MOG generates a foreground mask from which we extract a set of moving objects, called blobs (Fig. 3). Out of these candidate blobs, we aim to select the ones that contain the target.

B. CNN architecture

To decide which candidate blobs contain the target, we use a CNN, as CNNs have shown top performance in object classification in images. Our CNN is based on the CNN AlexNet [16], which is pre-trained on ImageNet.

We transfer the pre-trained layers of AlexNet to our CNN, except for the last 3 layers, which we replace with a new fully connected layer, a new softmax-layer and a new output layer to match our two classes: target (i.e. male golden-collared manakin) and background (i.e. Panama forest). The output of our CNN is a background score and a target score in $[0, 1]$. We fine-tune this new CNN with image patches of male golden-collared manakins and of background cropped from a set of videos in our dataset.

C. Kalman Filter

We use a linear Kalman Filter to predict the location of the bird if we could not obtain a reliable estimation of the location from III-A and III-B. In addition, the Kalman Filter’s location estimation is used if we find more than one blob. In such cases, we select the blob that is closest to the Kalman Filter’s location estimation.

D. Bird Tracking

The male golden-collared manakin’s location is initialized in the first frame with the ground truth bounding box. For each following frame, moving foreground blobs are detected in the scene. To find the blobs that contain the male bird, the blobs are classified with the fine-tuned CNN. We keep the blobs that receive a high target score, and discard the others. If there is only one such blob, its position is selected as the current target location.

In case there is more than one blob, the one that is closest to the location predicted by the Kalman filter is selected as the main blob. Since the bird can be partly occluded (e.g. by the sapling it sits on) it can consist of more than one blob. Thus, we add blobs to the main blob that received a high target score by our CNN and that are close to the main blob. If we find a blob or combination of blobs, that fit these criteria, it becomes the current target location (Fig. 4).

If we find no blobs in a frame or none that fulfill the conditions described above we search the bird in the region around its previous location. This usually happens when the bird is sitting and thus not recognized as a foreground blob. We shift the bounding box from the previous position to its left, right, top, bottom and diagonal neighborhood, crop image patches at these candidate locations and classify them with the CNN. All candidate locations that receive a high target score are averaged, and selected as the current target location (Fig. 5).

In frames where the bird is not recognized by the CNN the Kalman filter is used if the bird is predicted to be flying. Otherwise, we use the bird’s previous location as the current target location.

If the bird leaves the scene, candidate locations are placed along the edges of the frame to detect the bird when it re-enters the scene. To avoid false positive detections while the bird is outside the frame, only blobs are considered for detecting re-entering birds.

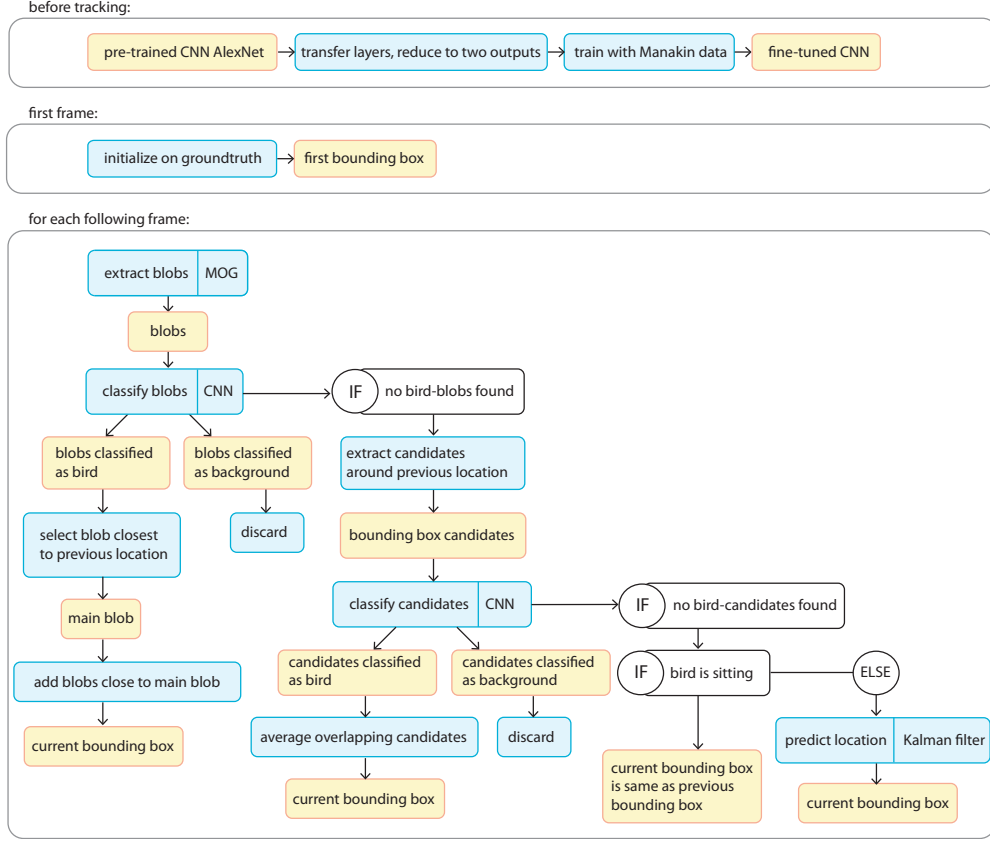


Figure 2: Flowchart of ManakinTracker.



Figure 3: Foreground mask (right) generated by MOG model of frame (left). Blue box: extracted blob.

The ManakinTracker uses a Kalman Filter and MOG, and thus does not rely only on the visual information in a single frame but detects motion based on multiple frames.

It uses a CNN, that is fine-tuned specifically to recognize male golden-collared manakins (including highly blurry and partly occluded) with high accuracy.

In most cases (particularly during jumps) blobs lead to very accurate bounding boxes and no further correction of the bounding boxes' dimensions is necessary. We can track the bird efficiently in most frames by classifying only a limited number of image patches extracted from blobs (usually 1-4 per frame).

In some cases the tracker recognizes the female bird as the target, even though the CNN was only trained on

male birds. This suggests that the CNN does not rely only on the male bird's yellow neck for classification. The downside of this is, that if the male and female bird are both present in a frame the two have to be distinguished by the tracker. One solution would be to train a CNN on images of female birds also, which would require ground truth bounding boxes for the female birds. Currently, we handle this issue by choosing the blob that is closest to the location predicted by the Kalman filter if there are multiple blobs that get a high target score.

IV. EVALUATION

To evaluate the performance of the ManakinTracker, we compared it to two of the trackers presented in Section II: TCNN and C-COT. A fair comparison with Oliva's tracker was not possible, because this tracker can neither be initialized nor re-started with a ground truth bounding box. Additionally, this tracker relies strongly on thresholds that were determined based on the dataset it was trained on, and outputs only bounding boxes for the male bird's neck, which keeps accuracy low even in case of successful tracking.

We assess the trackers' performance based on accuracy and number of re-starts. Accuracy is measured with the Jaccard index. A tracker is re-started at the next frame that has a ground truth annotation if the pre-

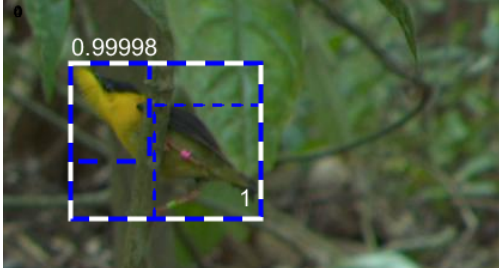


Figure 4: Two small blobs (small blue bounding boxes) are combined into a bigger blob (big blue bounding box). The white text indicates the blobs’ target scores.

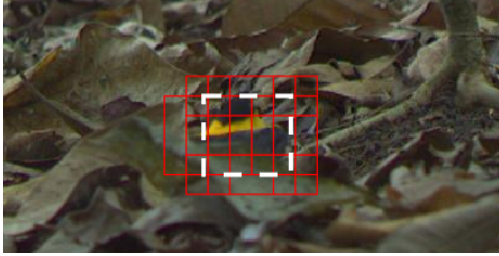


Figure 5: Bird is sitting and no blob was found (candidate locations with high target score (red boxes), final bounding box (white box))

dicted bounding box has zero overlap with the ground truth bounding box.

Our test dataset consists of 78 video sequences that show male golden-collared manakins performing their courtship dance. All videos were recorded with stationary high-speed cameras at 60 fps in the Panama forest. Every frame has a ground truth annotation (bounding box enclosing the male bird) provided by biologists. For testing, we split the dataset in half and train the CNN on one half and run the tracker on the other half.

Tracker	Avg. accuracy	Avg. # re-starts
ManakinTracker	58.3093%	1.2051
C-COT	49.0720%	4.6795
TCNN	53.3405%	7.1154

Table I: Trackers’ performance on test dataset.

V. RESULTS

Table I shows that the ManakinTracker performed the best out of the three trackers, both in terms of accuracy (58.31% average overlap) and robustness (1.21 re-starts per sequence on average). TCNN achieves higher accuracy (53.34%) than C-COT (49.07%), but needs about 1.5 times more re-starts on average.

TCNN and C-COT both use CNNs pre-trained on ImageNet. The performance of networks trained on the ImageNet dataset, which consists of still images, decreases strongly if images are blurry [17]. In contrast, the ManakinTracker’s CNN was trained also on blurry images, extracted from videos similar to the ones it was tested on. For a more detailed evaluation see [18].

VI. CONCLUSION

The ManakinTracker achieved better accuracy and needed less re-starts than two state-of-the-art trackers. Keeping the number of re-starts low was our main goal as we aim to minimize user input during tracking. Using a CNN trained on similar videos as the test set led to a high accuracy in detecting and tracking the male golden-collared manakin.

REFERENCES

- [1] M. J. Fuxjager, L. Fusani, F. Goller, L. Trost, A. T. Maat, M. Gahr, I. Chiver, R. M. Ligon, J. Chew, and B. A. Schlinger, “Neuromuscular mechanisms of an elaborate wing display in the golden-collared manakin (*manacus vitellinus*),” *Journal of Experimental Biology*, 2017.
- [2] J. Barske, B. A. Schlinger, M. Wikelski, and L. Fusani, “Female choice for male motor skills,” *Proceedings of the Royal Society of London B: Biological Sciences*, 2011.
- [3] D. Kress, E. van Bokhorst, and D. Lentink, “How lovebirds maneuver rapidly using super-fast head saccades and image feature stabilization,” *PLOS ONE*, vol. 10, no. 6, pp. 1–24, 06 2015.
- [4] D. Eckmeier, B. R. Geurten, D. Kress, M. Mertes, R. Kern, M. Egelhaaf, and H.-J. Bischof, “Gaze strategy in the free flying zebra finch (*taeniopygia guttata*),” *PLoS One*, vol. 3, no. 12, p. e3956, 2008.
- [5] D. L. Altshuler, E. M. Quicazán-Rubio, P. S. Segre, and K. M. Middleton, “Wingbeat kinematics and motor control of yaw turns in anna’s hummingbirds (*calypte anna*),” *Journal of experimental biology*, vol. 215, no. 23, pp. 4070–4084, 2012.
- [6] I. G. Ros, L. C. Bassman, M. A. Badger, A. N. Pierson, and A. A. Biewener, “Pigeons steer like helicopters and generate down- and upstroke lift during low speed turns,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 50, pp. 19990–19995, 2011.
- [7] N. Ota, M. Gahr, and M. Soma, “Tap dancing birds: The multimodal mutual courtship display of males and females in a socially monogamous songbird,” *Scientific Reports*, vol. 5, 2015.
- [8] D. Song and Y. Xu, “Monocular vision-based detection of a flying bird,” *Texas A & M University*, 2008.
- [9] W. Li and D. Song, “Automatic video-based bird species filtering using periodicity of salient extremities,” *Department of Computer Science and Engineering, Texas A&M University, Tech. Rep. TR2012-08-2*, 2012.
- [10] L. Oliva, A. Saggese, N. M. Artner, W. G. Kropatsch, and M. Vento, “From trajectories to behaviors: an algorithm to track and describe dancing birds,” *Proceedings of the 22nd Computer Vision Winter Workshop, Retz, A*, p. 1–9, 2017.
- [11] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” p. 252 Vol. 2, 02 1999.
- [12] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder *et al.*, *The Visual Object Tracking VOT2016 Challenge Results*. Cham: Springer International Publishing, 2016, pp. 777–823.
- [13] H. Nam, M. Baek, and B. Han, “Modeling and propagating cnns in a tree structure for visual tracking,” *CoRR*, vol. abs/1608.07242, 2016.
- [14] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in *ECCV*, 2016.
- [15] G. Welch and G. Bishop, “An introduction to the kalman filter,” 1995.
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
- [17] S. F. Dodge and L. J. Karam, “Understanding how image quality affects deep neural networks,” *CoRR*, vol. abs/1604.04004, 2016.
- [18] A. Gostler, “Tracking golden-collared manakins in the wild,” PRIP, TU Wien, Tech. Rep. PRIP-TR-141, 2018.

Vascular Structure Skeletonization: A Survey

Timon Höbert

Abstract

Spatial and geometric features of vessel structures are important criteria to identify anomalies. The detection of the center-lines of these structures, the skeleton, is a necessary preprocessing step. This paper highlights the main properties of skeletons and gives an overview of different skeletonization techniques and their applications. Seven techniques are identified and compared according to centeredness, robustness, performance and parallelization.

1 Introduction

The structure of tree-like topologies, such as nerves, vascular, and bronchial trees is significant in diagnosis and therapy. Even slightly deviating shapes or branching patterns in these structures cause the need for different therapies. Thus, an exact analysis and visualization of different geometric features are important in the decision-making process for different treatments.

The detection of various anomalies such as narrowings (stenosis) or dilated parts (aneurysms) as well as plaque and thrombus formation is one typical field of application [10]. Via virtual colonoscopy, an examination of a patient's colon is simulated via rendering of CT/MR data through vessels. This technique is used for early detection of colonic polyps which are the major cause of colon cancer, the second leading cause of cancer deaths in the USA [7].

These analysis and visualization methods highly depend on geometric features such as the diameter and center-line of the vessel. The estimation process of these vessel center-lines is called "skeletonization".

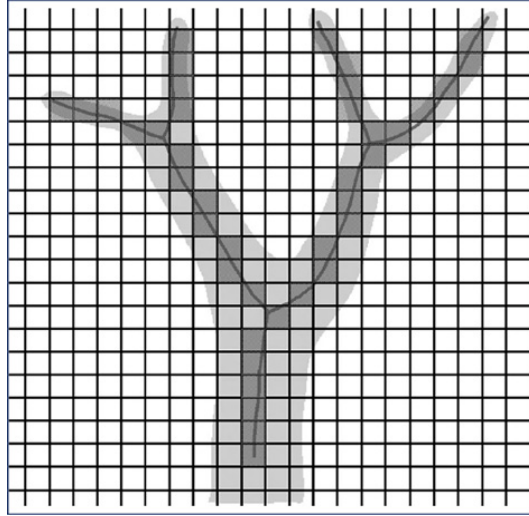


Figure 1: Continuous and discrete skeleton: The dark line represents the continuous skeleton of the gray vessel. The dark gray pixels represent the discrete skeleton. (From: [10]).

2 Properties

The skeletonization of an object reduces its dimensionality to a medial axis while preserving topologic and geometric properties [5]. It is common to differentiate between discrete skeletons in discrete space and continuous ones in the continuous domain. A continuous skeleton is a set of space curves or surfaces representing the centers of maximum enclosing circles in 2D or the maximum enclosing spheres in 3D. A discrete skeleton is a one or two voxel wide line in the center of the corresponding volumetric object [10]. Figure 1 visualizes both types of skeletons.

Beside the definition of the maximum enclosing balls, a skeleton can be defined in two other ways as well. The usage of maximal inscribed balls (MIB) leads to a related definition, but different results since the maximum enclosing ball touches the boundary only on two or more disjoint locations. One benefit of this method is the possibility of an exact reconstruction of the shape by using the location and radius of the MIBs. The firefront-definition [2] uses a grassfire-transformation, where the object shape can be imagined as a field of dry grass with a fire lit up synchronously along the boundary. The skeleton represents the set of quench points, where two independent firefronts meet based on a uniform propagation velocity.

In discrete space, a binary/grayscale 2D-shape is reduced to a curve of 1D structures. In 3D, a volume is either reduced to a surface of 2D-structures or to curves of 1D-structures. Figure 2 visualizes the difference of curve (a,b) and surface-skeletons (c,d). This paper focuses on 3D-data because of the generally used procedure, which produces 3D-data, but most of the figures are in 2D for a simpler visualization.

Figure 2 also shows the difference between symmetric (a,c) and asymmetric skeletons (b,d). The ideal line width of one voxel of an (asymmetric) skeleton lacks rotational invariance if the related continuous skeleton lies between two voxels. This rotational invariance is achieved via the definition of symmetric skeletons, where a width of two voxels is allowed for these cases. An analogous definition can be generalized to surface skeletons as well, as shown in figure 2(d).

As mentioned in [12, 10], the quality of a skeletonization method is dependent on the following properties:

Centeredness The skeleton stays away from the walls of the vessel as much as possible.

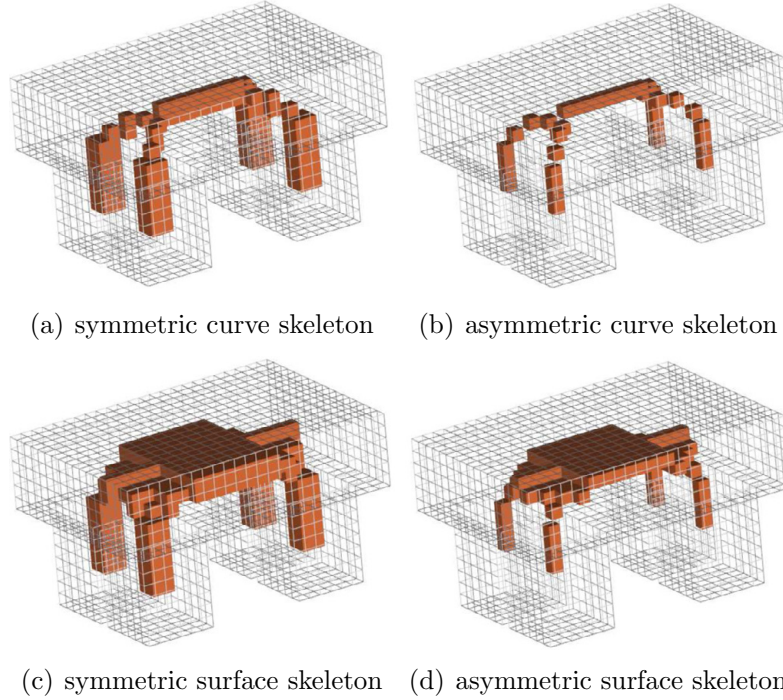


Figure 2: Different discrete skeleton definitions. Symmetric/asymmetric skeletons in the columns and curve/surface skeletons in the rows (From: [10]).

Robustness Structural changes of a vessel do not change the skeleton: A separated segment with branches and loops should yield equivalent results as a perfectly prepared tubular segment.

Invariant against affine transformation The skeleton stays consistent in case of translated, rotated and scaled structures.

Performance The skeletonization method must be efficient enough, to compute and visualize the skeleton in real time.

Parallelization The computation of the skeletonization should be possible in multiple parallel subcomputations.

Connectivity The parts of the skeleton should be connected. In discrete space, this is achieved via a defined 6-, 18- or 26-connected neighborhood.

3 Different techniques of skeletonization

Seven different main techniques of skeletonization are identified and compared in the following chapters.

3.1 Thinning

The simplest skeletonization method is a thinning of previously segmented vessel structures. It is accomplished via continuous morphological erosion in 3D-space of the initial structure until one-voxel wide lines result. Thereby, all voxels are removed, which are not terminal voxels (with only one neighbor) and not branching voxels (with more than two voxels) [7].

The morphological peeling operation is simple, and a topological correctness is ensured. However, the thinning of large objects results in numerous iterations of the process, even with accelerations [7]. Additionally, this method heavily depends on the quality of the segmented input vessels.

3.2 Wave-front propagation

Wave-front propagation operates analogously to iterative thinning. The boundary-voxels are used as an initial wave, which is iteratively expanded to the inside of the given shape. Collisions of opposite wave-fronts indicate the skeleton-voxels. This approach leads to disconnected skeleton-parts in any narrowing of the shape [6]. In some cases, disconnected skeletons are sufficient, for example for the vessel registration of multiple volumes. But for most cases, an additional merging step is needed. This merging can be done by linking the collision points computed in wave iteration i with their nearest point at distance i [3]. Similarly to iterative thinning this method needs as many iterations as the number of voxels at the thickest point of the object, but a parallel computation is possible.

3.3 Distance Transform

Via distance transforms a representation is calculated where every voxel indicates the distance to the closest boundary voxel. Figure 3 visualizes the calculated distance transform of a 2D-Image in 3D where the height encodes the distance. Based on these distances, the path along the maximally valued voxels is generated, which represents the center-lines. It is also possible to

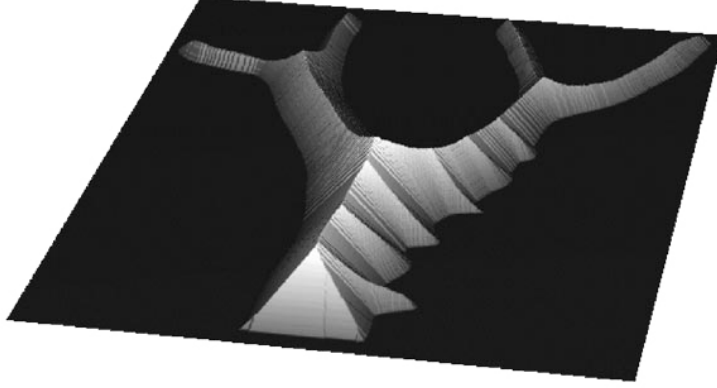


Figure 3: Distance Transform: The computed distance transform of the segmented vessel tree from figure 1 illustrated as a 3-D visualization where the height encodes the calculated distances (From: [11]).

approximate the distance transform, to minimize the computational costs. To avoid the square root of the exact Euclidean distance approximated metrics with integer values are preferred. Instead of distances of 1, $\sqrt{2}$ and $\sqrt{3}$ for face-, edge- or vertex-connected voxels, the values 1, 2 and 3 [15], 3, 4 and 5 [7] and 10, 14 and 17 [12] are approximated. The computational costs increase with more precise metrics, but the precision of the final results do as well. In general, the precision of skeletons using distance transforms is accurate enough to stay within the object; still it is not perfectly centralized, especially in corner areas.

3.4 Voronoi diagram

Voronoi diagrams can be used for skeleton extraction as well. Hereby, the voxels along the boundary are used as seed-points to calculate the diagram. The resulting cell-boundaries which are not associated with two neighboring seed points represent the final skeleton, as shown in figure 4(a,b). This skeleton generation matches the previously mentioned definition of skeletonization via maximum inscribed balls [1]. The union of all inscribed balls along the skeleton leads to a reconstruction of the original shape, as shown in figure 4(c).

There are efficient algorithms to compute 3D-Voronoi diagrams with an experimental time complexity of $\mathcal{O}(n \log n)$ [8]. Additionally, parallelization

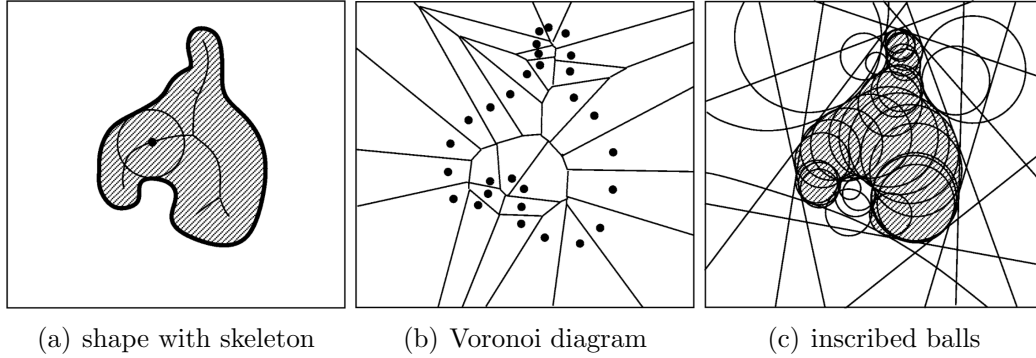


Figure 4: Example of a two-dimensional shape and its skeletonization via a Voronoi diagram. The original shape with its generated skeleton and one inscribed ball (a). (From: [1]).

is possible by separate cell-based computing ¹. However, the computation of the Voronoi diagram becomes very complex if the object is noisy, which leads to discontinuities and parasitic branches.

3.5 Direct Skeletonization

The skeletonization method directly applied to the scanned data - without any pre-segmentation of the vessels - is called "direct skeletonization". Thereby a starting point in the center of the vessel and a direction is selected by the user, and the procedure follows the vessel skeleton. Here the algorithm is continuously computing following vessel centers as circle centers on the perpendicular planes. This is accomplished via radial ray-casting to measure the distance to the wall of the vessel [13].

This method is fast and efficient because of its local operation, which includes a minimum of computational steps. Additionally, the skipped segmentation process also improves overall computational costs. The drawback of this method is the precision, especially in highly bent segments or near branchings [10], where some vessels are ignored or imprecise centers computed. Therefore, an additional manual correction or adjustment of parameters are required.

¹<http://math.lbl.gov/voro++>

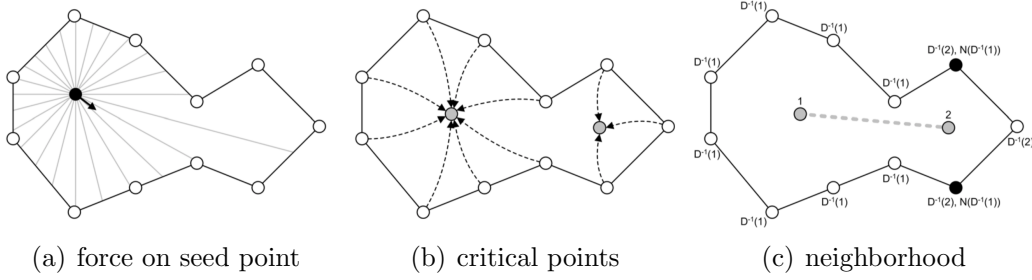


Figure 5: Repulsive field of force visualization. Example seed point with the resulting force towards the center (a). The critical points represent the local force minimum of the boundary voxels (b). Linear connection of critical points where the original boundary points are adjacent (c). (From: [14])

3.6 Repulsive Field of force

The repulsive field of force skeletonization uses the idea of a force field inside of the object via "charging" the object boundary. This force field pushes any seed point placed away from the charged boundary towards the center of the shape, as visualized in figure6.

The repulsive field for every point x can be calculated as

$$RF(x) = \sum f(\|v_i - x\|_2) \cdot \overrightarrow{(v_i - x)} \quad (1)$$

where v_i represents the boundary voxels, and $f(r) = r^{-2}$ the force function with an order of 2 for the Newtonian force.

After the computation of the force field, each boundary voxel is put into the force function resulting in a set of local minimum points, the *critical points*. Since the neighborhood of the original boundary voxels is known, the neighborhood of the resulting critical points can be derived. The neighboring critical points can be connected either with simple lines [14] or with curves via integration along the force field [4].

The computational complexity of the force field algorithm is dependent on the number of object voxels to calculate the force field and the boundary voxels to calculate the critical points. It can be further optimized via only using the visible voxels in the force field computation [14]. The total complexity is $O(no * nb)$, where no is the number of object voxels and nb the number of boundary voxels [4].

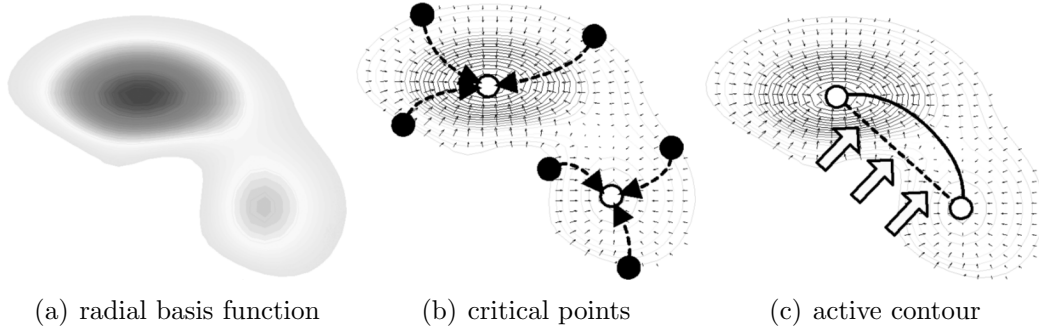


Figure 6: Radial Basis function skeletonization. Resulting force field of the shape via radial basis function (a). The critical points represent the local force minimum of the boundary voxels (b). Active contour model of critical points where the original boundary points are adjacent (c). (From: [9])

3.7 Radial Basis Functions

Similar to the repulsive field of force is the skeletonization method based on the radial basis function (RBF). The RBF level set, as shown in figure 6(a), is locally derived to result in a gradient vector field, as shown in figure 6(b). Again, the critical points are the centers of the resulting level set, which are the local maxima of the gradient vector field. The same neighborhood criterion as of the repulsive field of force is used to connect adjacent critical points. Here a curved connection between two critical points is calculated via an active contour model. This model minimizes an energy function which is dependent on the distance field and intrinsic properties of the curve, such as its length and curvature. The radial basis function can be efficiently computed sequentially and in parallel, but the optimization of the active-contour model is responsible for a reduced performance [9].

4 Comparison

A comparative overview of the previously discussed methods is shown in figure 4 for the criteria centeredness, robustness, performance and parallelization. There is no ideal method fulfilling all listed criteria. Each method strikes a compromise between performance (centeredness and robustness) and efficiency (sequential performance and parallelization).

Method	Centeredness	Robustness	Performance	Parallelization
Thinning	+	+	-	+
Wave Front Propagation	+	+	-	+
Distance Transform	+	-	+	+
Voronoi Diagram	+	-	+	+
Direct Skeletonization	-	-	+	-
Visible Repulsive Force	+	+	-	+
Radial Basis Function	+	+	-	+

Figure 7: Comparison of the discussed skeletonization methods based on centeredness, robustness, (sequential) performance, and parallelization.

5 Conclusion

To sum up, the discussed methods differ in performance and efficiency. Depending on the use case, different trade-offs have to be considered. In addition to the issue that the definition of skeletonization in general is rather ambiguous, which makes a comparison of different skeletonization methods difficult, currently, no practical evaluation of the discussed methods is available. This evaluation conducted on one data set represents a topic for further research.

References

- [1] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2-3):127–153, 2001.
- [2] H. Blum. A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms, 1967*, pages 362–380, 1967.
- [3] R. Cárdenes and J. Ruiz-Alzola. Skeleton extraction of 2d objects using shock wavefront detection. In *International Conference on Computer Aided Systems Theory*, pages 392–397. Springer, 2005.
- [4] N. D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21(11):945–955, 2005.
- [5] M. Couprie and G. Bertrand. Asymmetric parallel 3d thinning scheme and algorithms based on isthmuses. *Pattern Recognition Letters*, 76:22–31, 2016.
- [6] A. A. Farag, M. S. Hassouna, R. Falk, and S. Hushek. Reliable fly-throughs of vascular trees. *Department of Electrical and Computer Engineering, University of Louisville, Tech. Rep*, 2004.
- [7] Y. Ge, D. R. Stelts, and D. J. Vining. 3d skeleton for virtual colonoscopy. In *Proceedings of the 4th International Conference on Visualization in Biomedical Computing, VBC '96*, pages 449–454, London, UK, UK, 1996. Springer-Verlag.
- [8] H. Ledoux. Computing the 3d voronoi diagram robustly: An easy explanation. In *Voronoi Diagrams in Science and Engineering, 2007. ISVD'07. 4th International Symposium on*, pages 117–129. IEEE, 2007.
- [9] W.-C. Ma, F.-C. Wu, and M. Ouhyoung. Skeleton extraction of 3d objects with radial basis functions. In *Shape Modeling International, 2003*, pages 207–215. IEEE, 2003.
- [10] B. Preim and C. P. Botha. *Visual Computing for Medicine, Second Edition: Theory, Algorithms, and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2013.

- [11] D. Selle, B. Preim, A. Schenk, and H. O. Peitgen. Analysis of vasculature for liver surgical planning. *IEEE Transactions on Medical Imaging*, 21(11):1344–1357, Nov 2002.
- [12] M. Wan, Z. Liang, Q. Ke, L. Hong, I. Bitter, and A. Kaufman. Automatic centerline extraction for virtual colonoscopy. *IEEE Transactions on Medical Imaging*, 21(12):1450–1460, Dec 2002.
- [13] O. Wink, W. J. Niessen, and M. A. Viergever. Fast delineation and visualization of vessels in 3-d angiographic images. *IEEE Transactions on Medical Imaging*, 19(4):337–346, April 2000.
- [14] F.-C. Wu, W.-C. Ma, P.-C. Liou, R.-H. Laing, and M. Ouhyoung. Skeleton extraction of 3d objects with visible repulsive force. In *Computer Graphics Workshop*, pages 124–131. Citeseer, 2003.
- [15] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Trans Vis Comput Graph*, 5(3):196–209, 1999.