IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2021–02–12.
Submitted on 2020–02–27, accepted on 2021–01–13.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.296

Finding the Skeleton of 2D Shape and Contours: Implementation of Hamilton-Jacobi Skeleton

Yuchen He[1], Sung Ha Kang[1], Luis Alvarez[2]

[1] School of Mathematics, Georgia Institute of Technology, US
[2] CTIM (Centro de Tecnologías de la Imagen), University of Las Palmas de Gran Canaria, Spain
(yhe306@gatech.edu, kang@math.gatech.edu, lalvarez@ulpgc.es)

Communicated by Enric Meinhardt-Llopis *Demo edited by* Yuchen He

Abstract

This paper presents the details of the flux-ordered thinning algorithm, which we refer to as the Hamilton-Jacobi Skeleton (HJS). It computes the skeleton of any binary 2D shape. It is based on the observation that the skeleton points have low average outward flux of the gradient of the distance transform. The algorithm starts by computing the distance function and approximating the flux values for all pixels inside the shape. Then a procedure called homotopy preserving thinning iteratively removes points with high flux while preserving the homotopy of the shape. In this paper, we implement the distance transform using a fast sweeping algorithm. We present numerical experiments to show the performance of HJS applied to various shapes. We point out that HJS serves as a multi-scale shape representation, a homotopy classifier, and a deficiency detector for binary 2D shapes. We also quantitatively evaluate the shape reconstructed from the medial axis obtained by HJS.

Source Code

The reviewed source code and documentation for this algorithm are available at the web page of this article[1]. See the `README.txt` file for usage instructions in the archive.

Keywords: 2D shape; skeleton; thinning algorithm; distance transform

1 Introduction

Shape representation plays an important role in feature analysis [33], object recognition [29] as well as classification [34]. A 2D shape is represented either by contour or region, and both can be further classified as structural or global [35]. For example, the chain code [26] is a contour-based structural method; the Fourier descriptor [21] is a contour-based global method; the convex hull is a region-based structure method; and the Zernike moment [32] is a region-based global method.

[1] https://doi.org/10.5201/ipol.2021.296

These representations provide compact information applicable for different purposes. An attractive approach is to encode the shape using simple geometries such as points, curves, or polygons. It allows user-friendly shape manipulation and scalable image rendition [10]. In this paper, we focus on the skeleton of a 2D shape [23, 5, 6, 22, 15, 8], which is a region-based structural representation method.

We implement the flux-ordered thinning algorithm proposed in [30] and refer to it as the *Hamilton-Jacobi Skeleton* (HJS). The first part of the algorithm is to compute the distance function from the boundary of the shape using a Hamiltonian formalism of the Eikonal equation. Then the flux is defined at each point using the gradient vector field of the distance function. In the second part, each boundary point with high flux is examined and removed if the homotopy of the shape remains unchanged. This homotopy preserving thinning procedure continues until no point is removable anymore, then these points constitute the skeleton of the shape.

We organize this paper as follows. After a brief review on the skeleton of 2D shape in Section 2, HJS is described in Section 3. In Subsection 3.1, we present the details of the computation of the distance transform of a shape. In Subsection 3.2, the average outward flux is given. In Subsection 3.3 and 3.4, we discuss the homotopy-preserving criteria and the implementation of the homotopy preserving thinning using a heap data structure to achieve high efficiency. We show the performance of HJS in Section 4 using several examples and conclude the paper in Section 5.

2 Review on Skeleton of 2D Shape

Let $I : \Omega \to \{0, 1\}$ be a binary image, where $\Omega = [0, M] \times [0, N]$ denotes the continuous image domain, and M, N are positive integers. A *2D shape*, A, is a subset of Ω with piecewise analytical boundary. Its *skeleton*, S, is a set of points which has equal shortest distance from two or more boundary points of A [23]. More precisely, the skeleton of a 2D shape is a finite union of C^2 curves that are either closed or ending at a finite set of junctions or endpoints [11]. One of the many equivalent definitions of skeleton [5, 6, 22, 15, 8] is based on the distance transform: the skeleton S of a shape A consists of the singularities of the distance transform restricted to the inside of A [30]. For example, the skeleton of a disk is its center point, and the skeleton of a square is given by its diagonals. Figure 1 shows the skeletons of some simple shapes. A skeleton can be used to reconstruct a shape via the medial axis transform [5]. It defines a coordinate system placed along the skeleton that encodes the distance from a skeleton point to the boundary of the shape. Hence, the contour can be recovered as the envelope of a series of circles centered at the skeleton with radii specified by the distance transform.

In the literature, different methods have been devised to skeletonize 2D shapes from various perspectives. For example, Voronoi diagram based methods [7, 25] focus on characterizing the symmetry axis; continuous transformation based approaches [20, 30, 16, 5, 22, 23, 15, 8] aim at extracting the singularity set of certain evolution of the boundary curve; and most morphology-based techniques [1, 4] look for maximally inscribed balls whose centers compose the skeleton. We refer the readers to [28] for a detailed review. A modified U-net was developed to directly map 2D shapes to their respective skeletons [24].

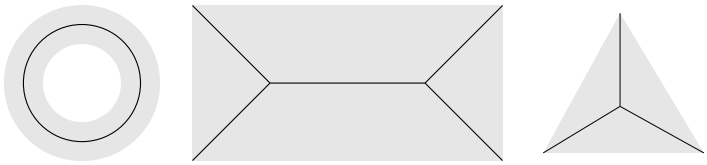


Figure 1: Skeletons (black curves) of some elementary shapes.

3 Hamilton-Jacobi Skeleton Algorithm

As summarized in [30], HJS has two main parts: Part I, distance function and flux computation, and Part II, homotopy preserving algorithm. We present the details in four Subsections. For Part I, the distance function is computed in Subsection 3.1. We propose to use the fast sweeping algorithm [36, 37] for the distance computation. In Subsection 3.2, the flux is defined for each point using the gradient field of the distance function. For Part II, in Subsection 3.3, the homotopy preserving point classification is explained, and in Subsection 3.4, the flux-ordered thinning algorithm is presented.

3.1 Distance Transform Using the Fast Sweeping Algorithm

Let $I : \Omega \cap \mathbb{N}^2 \to \{0, 1\}$ be a discretized binary image, where $\Omega \cap \mathbb{N}^2$ represents a union of square pixels on a grid. We denote by $\mathrm{Int}(\Omega) \cap \mathbb{N}^2 = \{1, 2, \ldots, M-1\} \times \{1, 2, \ldots, N-1\}$ the set of pixels in the interior of the image domain. On the continuous domain, the distance map $D : \Omega \to \mathbb{R}$ to the contour ∂A of the shape A is defined by

$$D(x, y) = \min_{(x',y') \in \partial A} \sqrt{(x - x')^2 + (y - y')^2} \, . \tag{1}$$

The distance D to the contour ∂A is the *unique* viscosity solution [12, 27] of the Eikonal equation

$$\begin{cases} |\nabla D(x, y)| = 1 \, , \\ D(x, y) = 0 \, , (x, y) \in \partial A \, . \end{cases} \tag{2}$$

This is a first-order nonlinear PDE of Hamilton-Jacobi type which does not have classical solutions. To solve D from (2), we employ the *fast sweeping algorithm* [37, 36]. The fast sweeping algorithm is an iterative method that uses an upwind scheme for discretization and Gauss-Seidel iterations with alternating sweeping orders for solving the Eikonal equations [36].

In the discrete setting, a pixel $(i, j) \in \Omega \cap \mathbb{N}^2$ is contained in the given shape A if $I[i, j] = 0$, and outside of A if $I[i, j] = 1$. To trace the contour of A on the discrete domain $\Omega \cap \mathbb{N}^2$, a pixel $(i, j) \in \mathrm{Int}(\Omega) \cap \mathbb{N}^2$ is considered a *boundary point* if it satisfies:

1. $I[i, j] = 0$, and
2. at least one of its 8-neighbors is outside the shape, i.e., $I[i \pm 1, j \pm 1] = 1$.

We denote by $\mathcal{B}$ the set of discrete boundary points.

On a rectangular grid, we discretize (2) using the Godunov upwind difference scheme for the interior points $(i, j) \in \mathrm{Int}(\Omega) \cap \mathbb{N}^2$

$$[(D[i, j] - D_{x,\min})^+]^2 + [(D[i, j] - D_{y,\min})^+]^2 = 1 \, , \tag{3}$$

where $D_{x,\min} = \min\{D[i + 1, j], D[i - 1, j]\}$, $D_{y,\min} = \min\{D[i, j + 1], D[i, j - 1]\}$, and $(z)^+ = z$ if $z > 0$ and $(z)^+ = 0$ otherwise. For pixels on the edge of the image, a one-sided difference scheme is applied. We initialize to $D[i, j] = 0$ every $(i, j) \in \mathcal{B}$, and assign large positive values to the other pixels. Next, we sweep the whole computational domain with four alternating orderings:

1) $i = 0, 1, \ldots, M, \ j = 0, 1, \ldots, N.$ 2) $i = M, M-1, \ldots, 0, \ j = 0, 1, \ldots, N.$
3) $i = M, M-1, \ldots, 0, \ j = N, N-1, \ldots, 0.$ 4) $i = 0, 1, \ldots M, \ j = N, N-1, \ldots, 0.$

As we are at pixel $(i, j) \in \Omega \cap \mathbb{N}^2$ during the sweeping, we compute the solution $\overline{D}[i, j]$ of (3) by

$$\overline{D}[i, j] = \begin{cases} \min\{D_{x,\min}, D_{y,\min}\} + 1 \, , & |D_{x,\min} - D_{y,\min}| \geq 1 \, , \\ \frac{1}{2}(D_{x,\min} + D_{y,\min} + \sqrt{2 - (D_{x,\min} - D_{y,\min})^2}) \, , & |D_{x,\min} - D_{y,\min}| < 1 \, . \end{cases} \tag{4}$$

Then we update $D[i,j]$ with $\min\{\overline{D}[i,j], D[i,j]\}$. Upon completing the four specified sweepings, the fast sweeping algorithm terminates; hence, the total computational cost is $O(MN)$. In practice, we only need to update the values of the pixels in the interior of the shape.

In [36], Zhao proved that the iterative solution by the fast sweeping algorithm converges monotonically to the solution of the discretized system (3). After four iterations, the iterative solution at every pixel is bounded from above by its true distance. Since the numerical Hamiltonian (3) is monotone and first-order consistent, combining with the fact that the numerical solution from a monotone and consistent scheme converges to the viscosity solution [27], Zhao concluded that the solution from the fast sweeping algorithm converges to the distance map. Moreover, by providing a pointwise error estimation, Zhao proved that this solution is optimal in the sense that any other method solving (3) has the same accuracy if not worse. These properties hold in n-dimensional Euclidean space ($n \geq 1$) as well, where the fast sweeping algorithm takes 2^n iterations.

Remark Starting from the continuous PDE setting, specifically, the Eikonal equation (2), we focus on the fast-sweeping algorithm for its efficiency and simplicity as a PDE numerical scheme. Thanks to one of the reviewers of this paper, we would like to acknowledge the advances in Euclidean Distance Transform (EDT) in discrete graph settings. In [13], the authors gave a survey comparing six algorithms proposed between 1994 and 2003. In 2012, Felzenszwalb and Huttenlocher [14] proposed a simple method focusing on the cost function defined on a grid. They formulated the problem as the minimum convolution of two functions and proposed a simple and fast algorithm. We compare the performance in Section 4.4.

Algorithm 1: Distance Computation: The Fast Sweeping Algorithm [36]

Input: I a binary image where a pixel is 0 if it is inside the shape and 1 otherwise.
Compute the set of boundary points $\mathcal{B}$.
Assign $D[i',j'] = 0$ for all $(i',j') \in \mathcal{B}$, and $D[i',j'] = M^2 + N^2$ for all $(i',j') \in (\Omega \cap \mathbb{N}^2) \setminus \mathcal{B}$.
for *i=0,1,...,M-1, j=0,1,...,N-1* **do**
  if $i = M-1$ **then**
    Define $D_{x,\min} = D[i-1,j]$.
  else if $i = 0$ **then**
    Define $D_{x,\min} = D[i+1,j]$.
  else
    Define $D_{x,\min} = \min\{D[i-1,j], D[i+1,j]\}$.
  Define $D_{y,\min}$ along the y-direction similarly.
  Compute $\overline{D}[i,j]$ according to (4).
  Update $D[i,j] = \min\{\overline{D}[i,j], D[i,j]\}$.
Repeat the above procedure for the other 3 sweeping directions, i.e.,
$i = M, M-1, \ldots, 0,\ j = 0, 1, \ldots, N.$
$i = M, M-1, \ldots, 0,\ j = N, N-1, \ldots, 0.$
$i = 0, 1, \ldots M,\ j = N, N-1, \ldots, 0.$
Output: Distance map D for the interior points of the shape in I.

3.2 Computation of Average Outward Flux

The skeleton points can be distinguished from the others by comparing their average outward fluxes derived from the gradient of the distance transform. On the continuous domain Ω, the average outward flux F of ∇D is defined by the outward flux through the boundary of its neighboring region

R, normalized by the Hausdorff measure of ∂R

$$F(x,y) = \frac{\int_{\partial R} \langle \nabla D, \mathcal{N} \rangle \, ds}{|\partial R|} \ , \quad (x,y) \in \Omega \tag{5}$$

where $\mathcal{N}$ is the outward normal along ∂R, and ds is the length element. By the assumption that the boundary of a 2D shape is piecewise analytical (in fact, being smooth suffices), the divergence of ∇D can be regarded as a measure supported by the skeleton of the shape. By the divergence theorem, when (x,y) is not a skeleton point, and R is sufficiently small, $F(x,y) = 0$. At any skeleton point which is not a crossing, up to a translation and a rotation, the distance function D admits a local expansion

$$D(x,y) \approx D_0 - x \text{ for } x \geq 0 \ ; \quad D(x,y) \approx D_0 + x \text{ for } x < 0,$$

where the skeleton point is placed at $(0,0)$, and D_0 denotes the distance from $(0,0)$ to the shape's boundary.

If we consider the disk $R_r(s,0) \equiv \{(x,y) : (x-s)^2 + y^2 \leq r^2\}$, with $r > 0$ then the average outward flux of ∇D is given by

$$\frac{\int_{\partial R_r(s,0)} \langle \nabla D, N \rangle \, ds}{|\partial R_r(s,0)|} = \begin{cases} \frac{-\int_{-\arccos(s/r)}^{\arccos(s/r)} \cos(t) r dt + \int_{\arccos(s/r)}^{2\pi - \arccos(s/r)} \cos(t) r dt}{2\pi r} = -\frac{2}{\pi}\sqrt{1 - \left(\frac{s}{r}\right)^2} & if \quad |s| < r, \\ 0 & if \quad |s| \geq r, \end{cases}$$

therefore the average outward flux of ∇D has a minimum at the skeleton point when we move in the direction orthogonal to the skeleton curve and the function $s \to -\frac{2}{\pi}\sqrt{1 - \left(\frac{s}{r}\right)^2}$ gives us an idea about how the average outward flux varies when approaching the skeleton.

More generally, notice that for any nonnegative test function $\varphi \in C_c^\infty$ with a sufficiently small support R such that $(0,0) \in R$, we have

$$\begin{aligned} \int_{\partial R} \langle \nabla D, \nabla \varphi \rangle \, ds &= \iint_{\mathbb{R}^2} \operatorname{div}(\nabla D) \, \varphi \, dx \, dy \\ &\approx -\iint_{x \leq 0} \varphi_x \, dx \, dy - \iint_{x \geq 0} -\varphi_x \, dx \, dy \\ &= -2 \int_{\mathbb{R}} \varphi(0,y) \, dy \ . \end{aligned}$$

The above calculation explains why, numerically, skeleton points are expected to have a clearly negative outward flux.

We compute the gradient vector field $\nabla D = (\partial_x D, \partial_y D)$ using a finite difference scheme that can be derived from optimization. We approximate $D(x,y)$ using a linear function

$$\widehat{D}(x,y) = D[i,j] + a(x-i) + b(y-j) \ .$$

Comparing to the Taylor expansion, we have $\partial_x D[i,j] \approx a$ and $\partial_y D[i,j] \approx b$. To determine the coefficients a and b explicitly, we fit $\widehat{D}$ to the 3×3 stencil centered at $P = (i,j) \in \operatorname{Int}(\Omega) \cap \mathbb{N}^2$ by minimizing the following weighted squared-error

$$\begin{aligned} \mathcal{E}(a,b) = &(D[i+1,j] - \widehat{D}(i+1,j))^2 + (D[i-1,j] - \widehat{D}(i-1,j))^2 + \\ &(D[i,j+1] - \widehat{D}(i,j+1))^2 + (D[i,j-1] - \widehat{D}(i,j-1))^2 + \\ &\frac{1}{\sqrt{2}}(D[i+1,j+1] - \widehat{D}(i+1,j+1))^2 + \frac{1}{\sqrt{2}}(D[i-1,j-1] - \widehat{D}(i-1,j-1))^2 + \\ &\frac{1}{\sqrt{2}}(D[i+1,j-1] - \widehat{D}(i+1,j-1))^2 + \frac{1}{\sqrt{2}}(D[i-1,j+1] - \widehat{D}(i-1,j+1))^2 \ . \end{aligned}$$

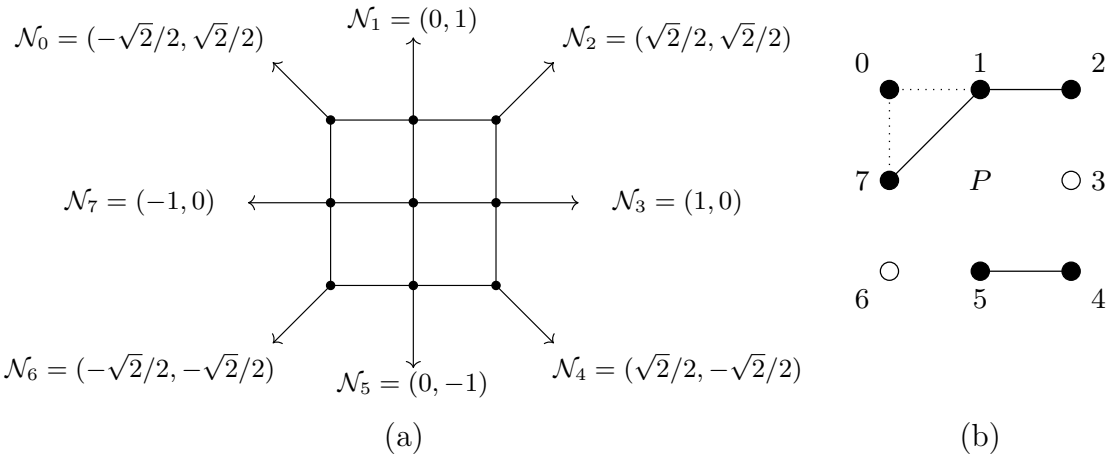


Figure 2: (a) The normal vectors at the neighboring points used for approximating the flux (8) at the central pixel. (b) An example graph G constructed for the pixel P. For any arbitrary pixel, its 8 neighborhoods are indexed as shown here. Neighboring pixels inside the shape (black circles) are the vertices of G, and two vertices are connected if they are 8-neighborhood to each other. We avoid the 3-loops at the corners, e.g., $0-1-7$, by directly connecting the furthest two among them.

,

This provides approximations for partial derivatives at interior points

$$\partial_x D[i,j] = (1-\alpha)\frac{D[i+1,j] - D[i-1,j]}{2} + \alpha\frac{D[i+1,j+1] - D[i-1,j+1] + D[i+1,j-1] - D[i-1,j-1]}{4} \tag{6}$$

$$\partial_y D[i,j] = (1-\alpha)\frac{D[i,j+1] - D[i,j-1]}{2} + \alpha\frac{D[i+1,j+1] - D[i+1,j-1] + D[i-1,j+1] - D[i-1,j-1]}{4} \tag{7}$$

where $\alpha = 2 - \sqrt{2}$. Note that the scalar α is derived from requiring the estimated gradient to have a norm invariant under rotations of 45°.

In the discrete domain, to compute $F[i,j]$ as defined in (5) for the pixel $(i,j) \in \mathrm{Int}(\Omega) \cap \mathbb{N}^2$, we replace R with a square containing the 8-neighboring points of (i,j). We pre-compute the outward normals at the neighboring points, $\mathcal{N}_n = (\mathcal{N}_n^x, \mathcal{N}_n^y)$, $n = 0, 1, \ldots, 7$, as illustrated in Figure 2 (a), and we approximate the integral by a Riemann sum. The formula for computing the average outward flux at (i,j) is

$$F[i,j] = \frac{1}{8}\sum_{n=0}^{7}(\partial_x D[i,j] \times \mathcal{N}_n^x + \partial_y D[i,j] \times \mathcal{N}_n^y) \ . \tag{8}$$

3.3 Point Classification Based on Local Topology

As a thinning algorithm, HJS finds the skeleton of a shape by consecutively removing non-skeleton points. For an appropriate shrinkage, each pixel needs to be examined carefully based on the local topology, i.e., the intensity distribution of its 8-neighboring pixels. In particular, two types of points are critical in the success of HJS.

A point is *simple* if its removal does not affect the topology of the object [30]. This means that removing a simple point does not create a new connected component nor a hole in the original shape.

We construct a graph G for every pixel $P \in \mathrm{Int}(\Omega) \cap \mathbb{N}^2$ based on its 8-neighbors and use the Euler's characteristic of a graph to decide if P is simple. See Figure 2 (b). In G, each vertex corresponds to a neighboring pixel that is inside the shape A, and there is an edge if the associated two pixels are neighbors to each other. Because P is simple if and only if G is a tree [30], it suffices to check if the number of vertices minus the number of edges of G, i.e., the Euler characteristic of the graph, is exactly 1. For convenience, we label the 8-neighboring pixels by integer indices from 0 to 7 in a clockwise orientation starting at the top-left one (see Figure 2 (b)).

The procedure for checking if a point is simple goes as follows. At each point, construct a set V collecting the indices of neighbors that are inside A. Initialize $v = 0$ and $e = 0$ for recording the number of vertices and edges of G, respectively. For $k = 0, 1, \ldots, 7$, if both neighbor k and neighbor $\mathrm{mod}(k+1, 8)$ are found in V, we increase both v and e by 1; otherwise, we increase only v by 1. The graph may contain unnecessary loops of length 3 when all the three vertices on the corners are in V. For example, the neighbors 0, 1, and 7 in Figure 2 (b). To simplify the graph, we delete the shortest two edges in such a loop. In particular, if the neighbor k ($k = 0, 2, 4, 6$) is in V, and if both neighbor $\mathrm{mod}(k-1, 8)$ and neighbor $\mathrm{mod}(k+1, 8)$ are in V as well, we reduce both v and e by 1. Finally, we compute $v - e$; if it is 1, then we mark P as a simple point, otherwise, P is not a simple point.

In addition, endpoints need to be tracked. An *endpoint* corresponds to the end of a 4-connected or 8-connected digital curve [30]. Identifying these points helps to produce robust skeletons and avoid interior points in the final result. A pixel P is an endpoint if the associated vertex set V only has one element, or if V only has two elements whose indices differ by 1 or 7, i.e., they are 4-neighborhood to each other. Otherwise, P is not an endpoint.

3.4 Homotopy Preserving Thinning

The second part of HJS, called the homotopy preserving thinning, sifts through the pixels inside the shape such that the remaining pixels are the skeleton points. The flux computed in Subsection 3.2 as well as the point type discussed in Subsection 3.3 are the keys. To avoid early removals of the pixels that are likely to be skeleton points, the average outward flux ranks the pixels inside the shape from high to low, which are then examined in order. To preserve the homotopy of the shape, only simple points and endpoints with high fluxes are considered removable.

An important part of the implementation is the heap data structure, which allows operations such as `insert`, `top`, and `pop`. Elements stored in a heap are associated with sorting keys. A heap will automatically sort the inserted data so that the first element, retrieved by `top`, always has the maximal key value; and `pop` automatically deletes the top element. It is noted that a heap only allows access to the top element, thus retrieving the element with the maximal key is very efficient. In C++, one may define a heap via a `priority_queue`; and in Python, one can resort to `heapq`.

In our case, we construct a heap of pixels with their average outward fluxes as the sorting keys. To save the storage, we directly update the pixels of the given binary image I using three types of labels: points that are currently in the heap (label 2), removed points (label 1), and candidate skeleton points (label 0).

First, we insert all the boundary points that are simple into a heap H sorted by their fluxes and label them by 2. We sift through the points via iterations. During the updates, some points that are not simple may become simple later, or vice versa; thus it is necessary to test simple points at each iteration. In each iteration, we extract the top element in the heap by applying H.`top()` followed by H.`pop()`. If the top element is an endpoint and its average flux is below a threshold, τ, indicating that it has more potential to be a skeleton point (see Section 3.2), we remove it from H and update its label by 0. Otherwise, we change its label to 1, then we consider its 8-neighbors. If any of them is labeled 0, we insert it in the heap together with its sorting key and label it by 2. Here we propose

a practical formula for the threshold

$$\tau = \min_{(i,j)\in A\cap\mathbb{N}^2} F[i,j]/\gamma \ . \tag{9}$$

This $\gamma > 0$ is a parameter whose default value is $\gamma = 2.5$ in our experiments. The iteration terminates when there are no more points to be removed, i.e., H is empty. The pixels labeled by 0 constitute the skeleton of the shape A.

We display a complete description of HJS in the form of pseudo-code in Algorithm 2. The full algorithm divides into two parts. Part I computes the distance function and the average outward flux of the gradient of the distance transform, and Part II describes the homotopy preserving thinning.

Algorithm 2: Hamilton-Jacobi Skeleton (HJS) [30]

Input: I a binary image which takes value 0 for points inside the shape, and 1 for outside; τ a threshold parameter for flux value.

Part I: Distance Function and Average Outward Flux

Compute the distance map D (11) of the set of boundary points.
Compute the gradient vector field ∇D according to (6) and (7).
Compute the average outward flux of the gradient, F, using (8).

Part II: Homotopy Preserving Flux-ordered Thinning

Initialize an empty heap H. For each point P on the boundary of the shape:
if *P is simple* **then**
    insert $(P, F(P))$ into a heap H with the flux $F(P)$ as the sorting key;
    update $I(P) = 2$
while *H is not empty* **do**
    Let $(P, F(P)) \leftarrow H$.`top()`. Delete P from H via H.`pop()`
    if *P is simple* **then**
        if *P is not an endpoint* ***OR*** $F(P) > \tau$ **then**
            update $I(P) = 1$
            for *neighboring point Q of P* **do**
                if $I(Q) = 0$ **then**
                    if *Q is simple* **then**
                        Insert $(Q, F(Q))$ into H
                        update $I(Q) = 2$.
        else
            update $I(P) = 0$

Output: A binary image I where pixels labeled by 0 represent skeleton points.

4 Numerical Results

We present numerical results to illustrate various interesting aspects of HJS. Throughout the experiments, our default choice was $\gamma = 2.5$. If the image is grayscale with dynamic range $[0, 255]$, we transform it into binary by setting the pixels to be 1 if the intensity > 125, and 0 if the intensity ≤ 125. For an RGB image, we use the same binarization on its lightness obtained via $(R+G+B)/3$.

4.1 Skeletonization of 2D Shapes

In Figure 3, we show the skeletons computed using HJS for various types of shapes. In the first row, variations from a disk shape are shown. Compared to a disk whose skeleton is a single point, these shapes have more complicated skeletons due to continuous modifications on their boundaries. Some features of the shape can be understood from the graph topology of the skeleton. For example, each convex corner of the shape creates a branch; the last shape in the first row of Figure 3 has 16 convex corners, and its skeleton is a tree with 16 branches. In the second row, we applied HJS to shapes of some common objects: a *cup*, an *apple*, a *vase*, and a *hammer*. Both the *cup* and the *vase* which are topologically equivalent to an annulus, provide examples for the fact that a skeleton may contain loops, the number of which is equal to the genus of the shape. We also observe the correspondence between the convex corners of the shape and the branches of the skeleton as mentioned above. This exact property explains instability in computing the skeleton: any perturbation on the boundary of the shape may create a prominent change in the skeleton. Applying HJS to varying shapes of the lizard in the third row, we observe some extraneous branches on the skeleton due to the non-smooth boundaries. For this issue, many techniques, such as skeleton evolution [2] and pruning [3], have been proposed to automatically delete the irrelevant branches.

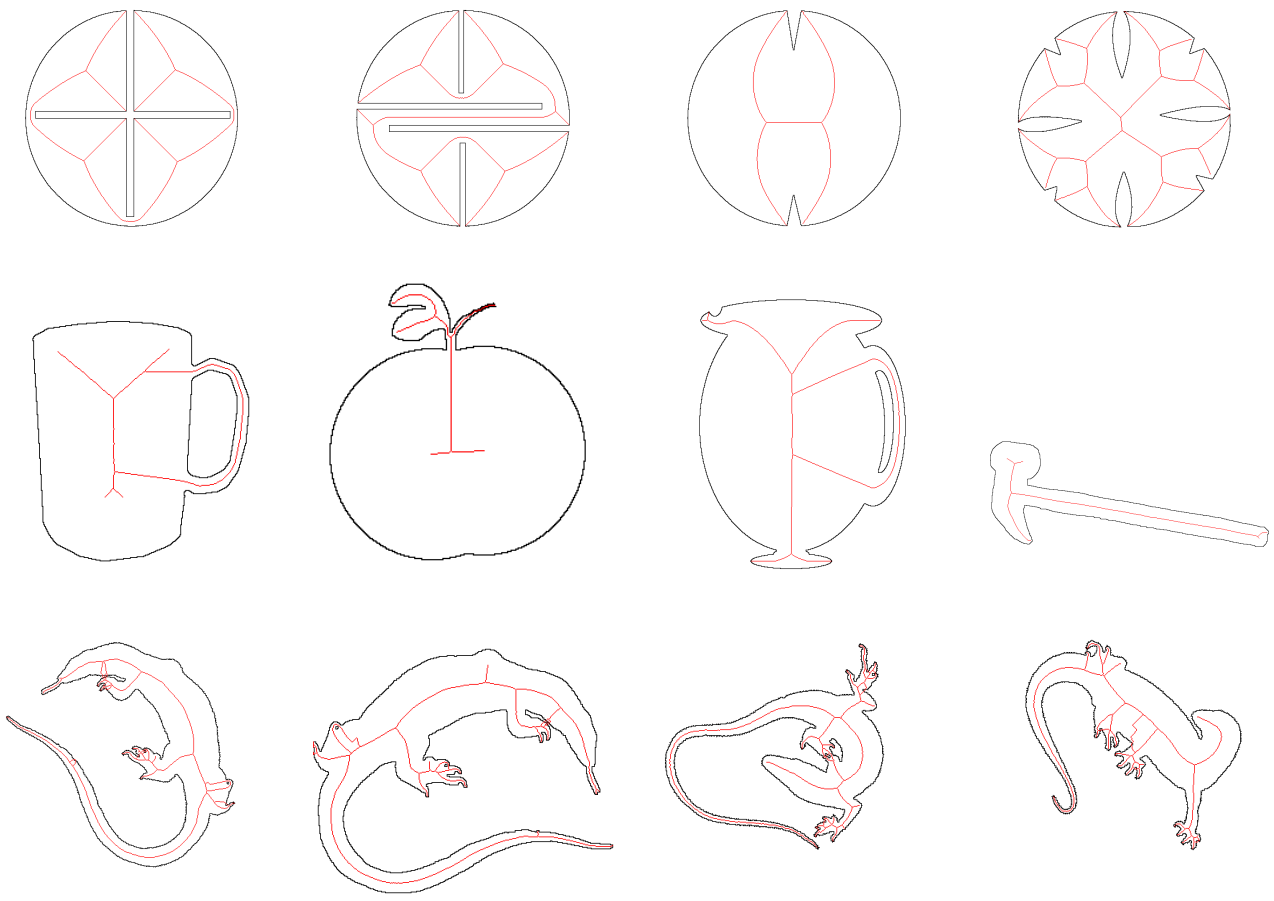


Figure 3: Skeletons (red curves) for various shapes computed by HJS. In all examples above, we used the default parameter $\gamma = 2.5$.

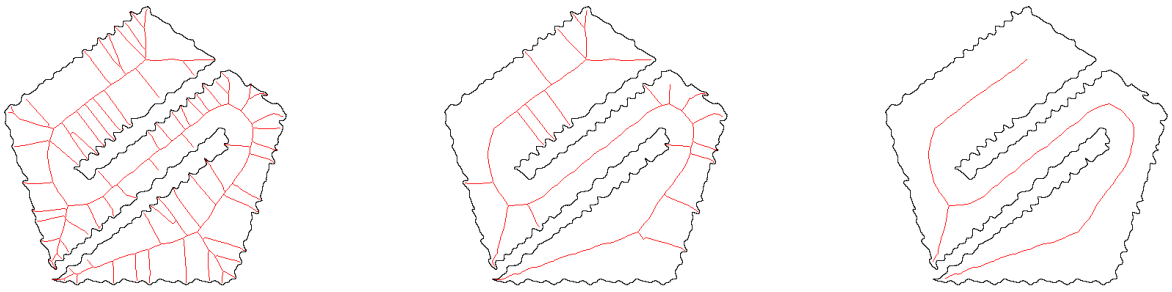
4.2 Effects of the Parameter γ

In HJS, an endpoint is kept if its average flux is greater than a threshold τ. As a consequence of the homotopy preserving, the whole branch attached to that endpoint remains as a part of the skeleton. Therefore, when we increase γ in (9), we expect to see more branches.

In Figure 4, we demonstrate the effects of γ by applying HJS to a shape modified from a pentagon whose boundary is highly perturbed. Figure 4 (a) is produced when $\gamma = 2.5$ (the default choice in this paper); due to the irregularities on the boundary, many branches are created. Observe the length of the skeleton branches in contrast to the size of the perturbations on the boundary: they are not proportional. The local distribution of skeleton branches, rather than individual ones, permits to infer the smoothness of the boundary. For instance in (a), along a segment of the skeleton near the top-left boundary of the pentagon, there are more branches on the lower side compared to the upper side; within that range, the upper-side boundary is smoother than the lower-side boundary.

In (b), with $\gamma = 1.5$, many of the branches in (a) disappear, leaving only those associated with sharp corners. The order of cancellation is determined by the flux. During the homotopy preserving thinning, the threshold τ only acts on endpoints. Consequently, as long as the average flux at the tip of the skeleton is comparatively low, the whole branch is kept as a part of the identified skeleton.

We further reduced γ to 1.2 in (c) and observe that only two major components remain: the S-shape and a single branch caused by the sharp corner at the bottom-left of the pentagon. By considering the medial axis to reconstruct the shape from the skeleton, with a small γ, one can obtain a compact representation of a shape with regularized contour. The evolution from (a) to (b), then to (c) shows that the set of skeletons extracted from different γ's provides a multi-scale representation of the given shape. It is analogous to applying bandpass filters to a signal to separate the low-frequency components of the original sequence from the high-frequency components such as noise. As we gradually decrease γ, we omit the small-scale variations along the boundary, and focus more on capturing the principal shape. A similar approach to shape analysis can be found in [21, 17].

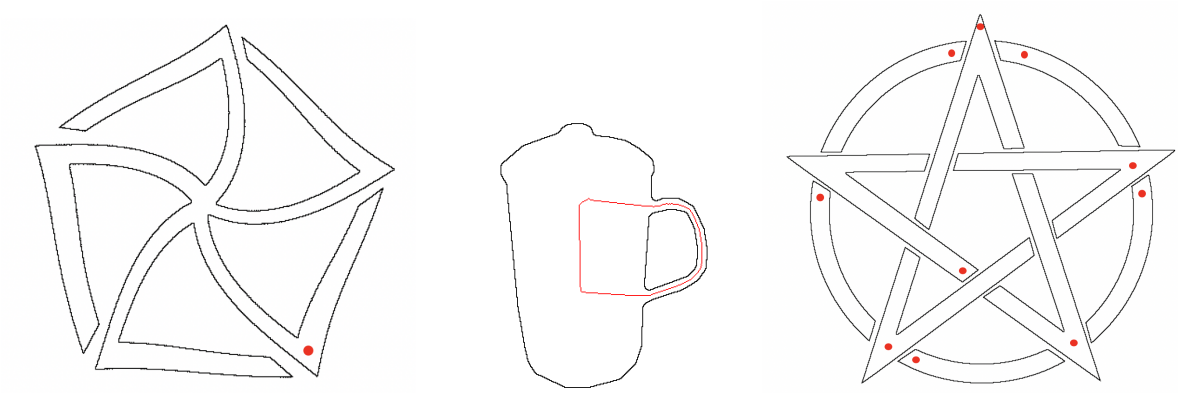


(a) (b) (c)

Figure 4: Multi-scale representation of the shape using skeletons computed by different γ. (a) $\gamma = 2.5$. (b) $\gamma = 1.5$. (c) $\gamma = 1.2$. By choosing a smaller γ, the identified skeleton becomes more robust against boundary perturbation and captures the large-scale shape features.

For a smaller value such as $0 < \gamma < 1$, HJS shows an interesting property. *The shape is simply-connected if and only if the skeleton identified using $\gamma < 1$ is a single point.* This is based on the fact that HJS preserves the homotopy. In Figure 5, we applied HJS with $\gamma = 0.9$ to three different shapes and used this property to check the simple-connectedness. In (a), since the skeleton is a single point (shown at the corner of the right-bottom branch), the shape is simply-connected. The converged skeleton point appears at the minimum of the average fluxes of the entire image, and

depending on the shape, it is not necessarily at the center of mass of the shape. In (b), the skeleton is homeomorphic to a circle, formed by the line segments (the *mug* body) and an arc (the handle). Since genus is a homeomorphic invariant, we infer that this *mug* shape has genus 1. It is worth noting that the number of connected components of the skeleton graph is the same as that of the shape. In (c), the skeleton consists of 10 points, hence the shape has 10 simply-connected components. With the remark that the skeletons identified using any $\gamma \in (0, 1)$ are identical, this experiment shows that HJS with $0 < \gamma < 1$ can be applied as an effective homotopy type detector.



(a) (b) (c)

Figure 5: HJS with $\gamma < 1$ used as a homotopy classifier. (a) The skeleton is a single point, hence the shape is simply-connected. (b) The skeleton is homeomorphic to a circle, hence the shape is not simply-connected and has genus 1. (c) The skeleton consists of 10 points, hence the shape has ten simply-connected components. In (a) and (c), the identified skeleton points are emphasized by red disks for visualization.

Since HJS with $0 < \gamma < 1$ effectively distinguishes simply-connected shapes from the others, we apply it to detect small holes inside the shape which are not immediately visible to humans. Figure 6 (a) shows the non-trivial skeleton identified using $\gamma = 0.9$, which indicates that the perturbed boundary contains loops. When we zoom in the top-left (b) and bottom-right (c) of the original shape, the homotopy type of the shape is indeed modified by the perturbed pixels.

(a) (b) (c)

Figure 6: HJS with $\gamma < 1$ used as a deficiency detector in binary shapes. (a) The given shape and identified non-trivial skeleton using $\gamma = 0.9$. (b) A hole on the boundary of the top-left petal. (c) A hole on the bottom-right pedal. (b) and (c) show the deficiencies inducing the non-trivial skeleton in (a). In all examples here, we keep $\gamma = 0.9$.

4.3 Shape Reconstruction from Medial Axis

Skeleton serves as a compact representation of the original shape. Combined with the distance function evaluated at the skeleton points, a medial axis allows shape reconstruction via taking a union of disks. In particular, from the set of skeleton points, $\mathrm{Sk}(S_o)$, of the discrete shape S_o, our reconstructed discrete shape S_r is computed by

$$S_r = \bigcup_{(i',j')\in \mathrm{Sk}(S_o)} \{(i,j) \in \Omega \cap \mathbb{N}^2 \mid \sqrt{(i-i')^2 + (j-j')^2} \leq D[i',j'] + \varepsilon\} \,. \tag{10}$$

Here, we introduce a *dilation parameter* $\varepsilon > 0$ to address the bias introduced by pixelization. We take $\varepsilon = 1.5$ as the default, which is justified later.

Figure 7 shows the reconstructed shapes from HJS using different values of γ. Recall from the previous discussion that, as $\gamma > 0$ increases, the identified skeleton consists of more branches representing details of the silhouette. Consistent with this feature characterization, from (b) to (d), as γ is increased from 0.9 to 20, more details of the original shape are recovered using the medial axis. Since $\gamma < 1$ for column (b), the reconstructed shape indicates existence of holes or simply connected components in the shapes in (a). In particular, the *sakura* in the first row is simply connected, the *knot* in the second row has 12 simply connected components, the *trophy* in the third row is of genus 6, and the *deer* in the fourth row contains many corrupted pixels (small holes) which are hard to observe at first glance. When $\gamma = 2.5$ (our default value), we recover most parts of the shapes in all cases. The results in column (d) are obtained using $\gamma = 20$, which recovers finer details of the original shapes compared to (c). For example, the sharp tips of the *petals*, the corners and T-junctions in the *knot*, the layers on the bottom of the *trophy*, and the elongated horns of the *deer*.

To quantify the reconstruction results, we compare the original discrete shape $S_o = \{(i,j) \in \Omega \cap \mathbb{N}^2 \mid I(i,j) = 0\}$ with the reconstructed shape S_r by considering the following three measures:

Jaccard Index [19]: $$J = \frac{|S_o \cap S_r|}{|S_o \cup S_r|} \,.$$

Dice Similarity Coefficient [31]: $$DSC = \frac{2|S_o \cap S_r|}{|S_o| + |S_r|} \,.$$

Bpn Bias Estimator [9]: $$Bpn = \frac{|S_r \setminus S_o| - |S_o \setminus S_r|}{|S_o \cap S_r|} \,.$$

Here $|\cdot|$ denotes the cardinality of a set. Higher values of Jaccard index or Dice similarity coefficient indicate higher similarity between S_o and S_r. A positive (negative) Bpn bias estimator signifies that S_r is an over-(under-) coverage of S_o, and a zero value means no bias, i.e., the size of the over-coverage cancels out with the size of the under-coverage. These measures behave differently. For the Jaccard index, every element in the intersection is counted once, whereas for the Dice similarity coefficient, every element in the intersection is counted twice. In fact, the Jaccard index and the Dice similarity coefficient are related via $J = DSC/(2 - DSC)$, hence, compared to the Dice similarity coefficient, the Jaccard index is less sensitive to distinct objects, while more sensitive to similar objects. The Bpn bias estimator provides the additional information for avoiding over-coverage or under-coverage.

In Table 1, we report these measures for the results in Figure 7. In all cases, when we increase γ, the shape reconstructed from the medial axis becomes more similar to the original shape, which is characterized by the increasing J and DSC. Since we are using disks with radius enlarged by 1.5 pixels for reconstruction, Bpn's change their signs from negative (under-coverage) to positive (over-coverage) as the endpoints of the skeleton approach the boundary of the original shapes, respectively. For comparison, we also include these measures when $\varepsilon = 0$. Using $\varepsilon = 1.5$ improves the reconstructions measured by higher J's and DSC's, and it produces less biases quantified by the

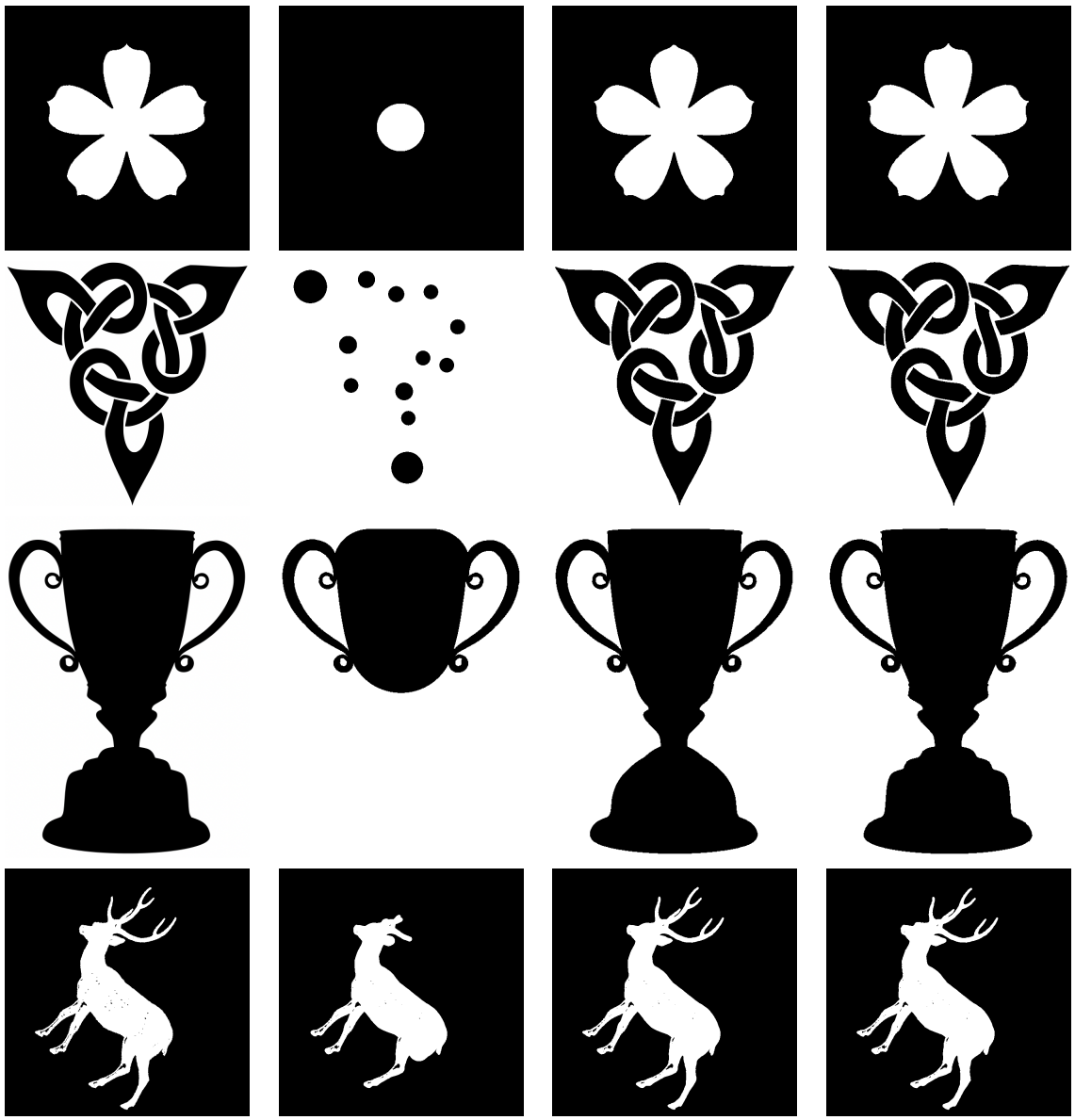


Figure 7: Shape reconstructed from the medial axis transform. (a) Original shapes. (b)-(d) The shapes reconstructed from the HJS using (b) $\gamma = 0.9$, (c) $\gamma = 2.5$, and (d) $\gamma = 20$. Here we fixed $\varepsilon = 1.5$.

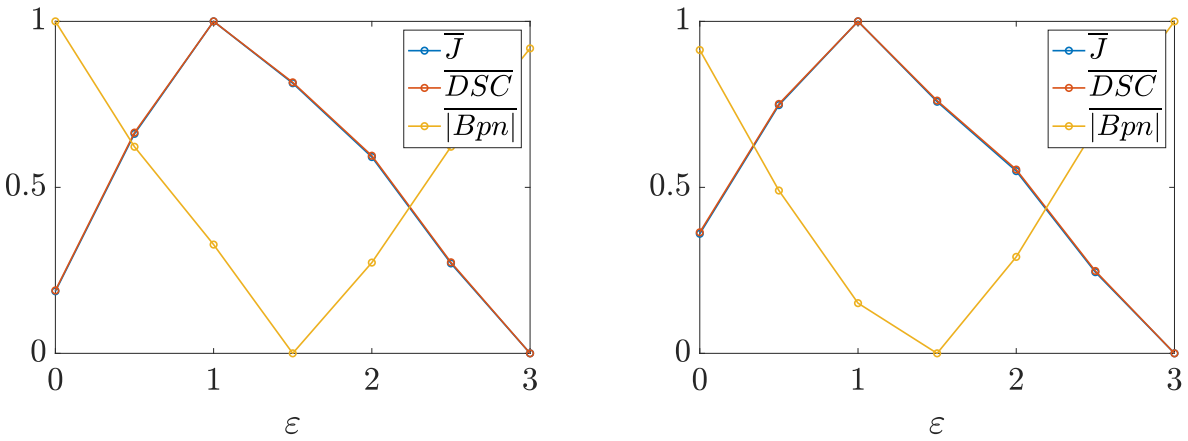
smaller absolute values of Bpn's. We notice that when $\gamma = 0$, Bpn's always remain negative in our examples. Figure 8 provides a further investigation on the effects of varying ε as $\gamma = 2.5$ is fixed. In both (a) and (b), the maximal rescaled values of J and DSC occur around $\varepsilon = 1$, but to acquire the minimal bias, larger values of ε are needed. This is due to the fact that we are using unions of finitely many disks to cover non-convex shapes, and slightly increasing ε allows balancing the over- and under-fitting. Hence, we recommend $\varepsilon = 1.5$, which leads to results with highly accurate shape reconstruction and low bias.

4.4 Performance of Distance Computation

As a natural extension of the Eikonal equation, the fast sweeping algorithm suits our purpose. In this section, we compare the fast sweeping algorithm with a brute-force method, and the celebrated [14], which is based on an optimization model efficiently solved using morphological operations.

Sakura (1^{st} row)	$\gamma = 0.9$		$\gamma = 2.5$		$\gamma = 20$	
	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$
J	0.1267	0.1340	0.9566	0.9770	0.9715	0.9816
DSC	0.2247	0.2363	0.9778	0.9884	0.9855	0.9907
Bpn	−6.8943	−6.4625	−0.0454	0.0006	−0.0294	0.0187
Knot (2^{nd} row)	$\gamma = 0.9$		$\gamma = 2.5$		$\gamma = 20$	
	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$
J	0.1714	0.1911	0.9451	0.9719	0.9436	0.9479
DSC	0.2926	0.3209	0.9718	0.9858	0.9732	0.9831
Bpn	−4.8351	−4.2234	−0.0581	0.0272	−0.0550	0.0342
Trophy (3^{rd} row)	$\gamma = 0.9$		$\gamma = 2.5$		$\gamma = 20$	
	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$
J	0.5576	0.5729	0.9641	0.9787	0.9745	0.9832
DSC	0.7160	0.7285	0.9817	0.9892	0.9871	0.9915
Bpn	−0.7933	−0.7146	−0.0372	0.0043	−0.0261	0.0170
Deer (4^{th} row)	$\gamma = 0.9$		$\gamma = 2.5$		$\gamma = 20$	
	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$	$\varepsilon = 0$	$\varepsilon = 1.5$
J	0.8361	0.8691	0.9324	0.9736	0.9411	0.9662
DSC	0.9108	0.9300	0.9650	0.9866	0.9697	0.9828
Bpn	−0.1960	−0.1153	−0.0725	0.0174	−0.0626	0.0332

Table 1: Comparative measures of the reconstruction results in Figure 7 ($\varepsilon = 1.5$). Higher values of J and DSC indicate higher similarity between S_o and S_r. When Bpn is positive (negative), S_r is an over- (respectively under-) coverage for S_o, and when $Bpn = 0$, there is no bias. We report these measures when $\varepsilon = 0$ for comparison.



(a) Sakura (b) Trophy

Figure 8: Effects of varying ε on the comparative measures. Here we plot the values of the rescaled measures, $\overline{J}$, $\overline{DSC}$ and $\overline{|Bpn|}$, against different values of ε, when HJS ($\gamma = 2.5$) is applied to the *sakura* in the first row and the *trophy* in the third row of Figure 7. Both plots indicate that using slightly dilated disks improves the reconstruction results.

On a discrete image domain, similarly to definition (1), one may define for each pixel $(i, j) \in \Omega \cap \mathbb{N}^2$, its distance to the boundary specified by $\mathcal{B}$ by

$$D[i, j] = \min_{(i',j') \in \mathcal{B}} \sqrt{(i - i')^2 + (j - j')^2} \, . \tag{11}$$

The implementation is straightforward, and we present the pseudo-code in Algorithm 3. If there are

K boundary points, then the cost of the brute-force method is of the order of $O(KMN)$, and it can be reduced by focusing only on the interior points of the shape.

Algorithm 3: Distance Computation: A Brute-force Method

Input: I a binary image which takes value 0 for points inside the shape, and 1 for background points

Compute the set of boundary points $\mathcal{B}$.
Assign $D[i', j'] = 0$ for every boundary point $(i', j') \in \mathcal{B}$.
for $(i, j) \in \Omega \setminus \mathcal{B}$ *with* $I[i, j] = 0$ **do**
  Set $D[i, j] = M^2 + N^2$.
  for $(i', j') \in \mathcal{B}$ **do**
    if $(i - i')^2 + (j - j')^2 < D[i, j]$ **then**
      $D[i, j] = (i - i')^2 + (j - j')^2$.
  Update $D[i, j] \leftarrow \sqrt{D[i, j]}$.

Output: Distance map D for the interior points of the shape in I.

The Felzenszwalb-Huttenlocher (F-H) algorithm [14] views the distance transform as a minimum convolution of two functions. More specifically, the squared Euclidean distance to the boundary of the shape $A \subseteq \Omega$ is obtained via an optimization problem

$$D_{\text{F-H}}(x, y) = \min_{x', y'} \left((x - x')^2 + (y - y')^2 + \chi_{\partial A}(x', y') \right), \tag{12}$$

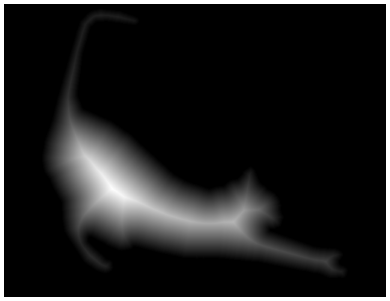
where $\chi_{\partial A}(x', y') = 0$ if $(x', y') \in \partial A$ and $+\infty$ otherwise. The discretized version of (12) is efficiently solved via two main steps in each dimension. First, it is computed the lower envelope of parabolas induced by each grid point's squared Euclidean distance function; at this stage, the minimum convolution is employed. Second, the distance values at grid points are filled in by comparing them with the computed lower envelope. The total computational cost is $O(2MN)$.

We applied theses three methods to the shape of a cat in Figure 9 (a). The distance transform computed using the brute-force method is shown in (b), the distance transform obtained using the fast sweeping algorithm is in (c), and the distance transform by the F-H algorithm is in (d). Noticeably, the computation of both fast sweeping algorithm (141.22 ms) and F-H algorithm (23.16 ms) are much faster than the brute-force method (3424.08 ms), yet all theses methods produce similar results. The skeletons computed based on the distance transform of these methods are respectively displayed in (e), (f), and (g). Notice that compared to (e) and (g), the skeleton in (f) obtained based on the fast-sweeping has fewer short branches and is more robust against small-scale fluctuations on the shape's boundary. This was expected since the fast sweeping algorithm's distance transform approximates a diffusive solution of the Eikonal equation. Hence, (c) can be regarded as a slightly smoothed version of the exact distance transform.

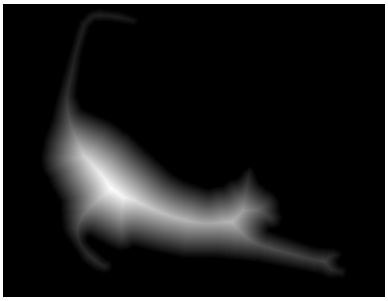
5 Conclusion

In this paper, we considered the flux-ordered thinning algorithm proposed by [30], which we referred to as the Hamilton-Jacobi Skeleton (HJS), and described an implementation of this method for extracting skeletons of 2D shapes from binary images. As a natural extension of the PDE framework, we updated a part of the algorithm computing the distance transformation by the fast sweeping algorithm [36], which improves the efficiency. The robustness of the identified skeleton against boundary perturbations can be adjusted via a single parameter $\gamma > 0$. For arbitrary shapes, we

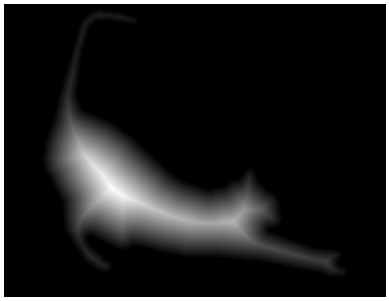
(a)



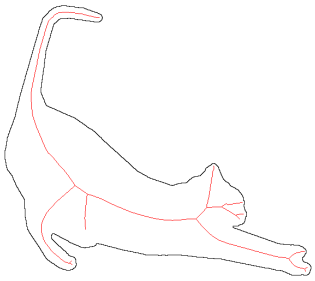
(b) Brute-force (3424.08 ms)



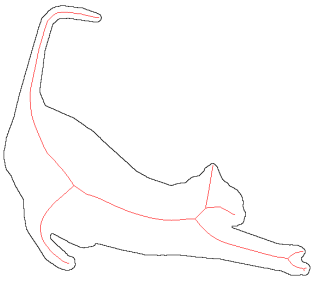
(c) Fast Sweeping (141.22 ms)



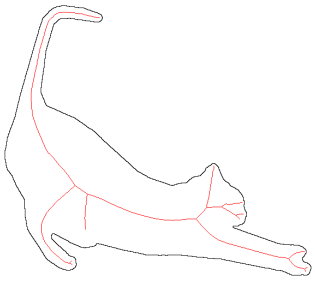
(d) F-H Algorithm (23.16 ms)



(e)



(f)



(g)

Figure 9: (a) Binary image (537×700): a cat silhouette. The distance function is computed by (b) a brute-force method (Algorithm 3), (c) the fast sweeping algorithm (Algorithm 1), and (d) the F-H algorithm [14]. Brighter pixels indicate further distance from the contour. The F-H algorithm is the fastest, then the fast-sweeping, and the brute-force is the slowest. (e) shows the skeleton computed based on the distance transform in (b); (f) shows the skeleton computed based on the distance transform in (c); and (g) shows the skeleton computed from (d). In all cases, we fixed $\gamma = 2.5$.

recommend fixing $\gamma = 2.5$ which produces the principal skeleton component and some branches indicating highly irregular features on the boundary. By applying HJS to a fixed shape using varying values of γ, we can obtain a multi-scale shape representation analogous to the approach considered in frequency component analysis. We investigated the special case where $\gamma < 1$ by exploring its connection to the homotopy type of a given shape and illustrating its usage as a deficiency detector for binary shapes. Moreover, we tested the skeleton identified by HJS as a tool for reconstructing shapes from their medial axes. When γ increases, the reconstruction is more precise.

Acknowledgment

This work is supported by the Chateaubriand Fellowship 2019-2020 awarded to Yuchen He and the Simons Foundation grant (584960) from Sung Ha Kang. We also thank all the reviewers for the constructive comments, especially the reviewer who provided the code for the Felzenszwalb and

Huttenlocher method [14].

Image Credits

 MPEG-7 Core Experiment CE-Shape-1 Test Set[2] [18], provided by Dr. Longin Jan Latecki, Professor, Department of Computer and Information Sciences, Temple University, US

 [3] Cat Stretch Silhouette In Black, CC0 Public Domain K

 [4] [5] [6] svgsilh.com, Creative Commons CC0

References

[1] C. Arcelli and G.S. Di Baja, *A width-independent fast thinning algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1985), pp. 463–474.

[2] J. August, A. Tannenbaum, and S.W. Zucker, *On the evolution of the skeleton*, in IEEE International Conference on Computer Vision (CVPR), vol. 1, IEEE, 1999, pp. 315–322. https://doi.org/10.1109/ICCV.1999.791236.

[3] X. Bai, L.J. Latecki, and W. Liu, *Skeleton pruning by contour partitioning with discrete curve evolution*, IEEE transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 449–462. https://doi.org/10.1109/TPAMI.2007.59.

[4] I. Bitter, A.E. Kaufman, and M. Sato, *Penalized-distance volumetric skeleton algorithm*, IEEE Transactions on Visualization and Computer Graphics, 7 (2001), pp. 195–206. https://doi.org/10.1109/2945.942688.

[5] H. Blum, *A transformation for extracting new descriptors of shape*, Models for the Perception of Speech and Visual Form, 19 (1967), pp. 362–380.

[6] H. Blum and R.N. Nagel, *Shape description using weighted symmetric axis features*, Pattern Recognition, 10 (1978), pp. 167–180.

[7] J.W. Brandt and V.R. Algazi, *Continuous skeleton computation by Voronoi diagram*, CVGIP: Image Understanding, 55 (1992), pp. 329–338.

[8] L. Calabi and W.E. Hartnett, *Shape recognition, prairie fires, convex deficiencies and skeletons*, The American Mathematical Monthly, 75 (1968), pp. 335–342.

[9] M.J. Cardoso, T. Arbel, S.L. Lee, V. Cheplygina, S. Balocco, D. Mateus, G. Zahnd, L. Maier-Hein, S. Demirci, E. Granger, and L. Duong, *Intravascular*

[2] http://www.dabi.temple.edu/~shape/MPEG7/dataset.html

[3] https://www.publicdomainpictures.net/en/view-image.php?image=32201&picture=cat-stretch-silhouette-in-black

[4] https://svgsilh.com/image/152115.html

[5] https://svgsilh.com/image/365843.html

[6] https://svgsilh.com/image/1614530.html

imaging and computer assisted stenting, and large-scale annotation of biomedical data and expert label synthesis, in CVII-STENT and Second International Workshop, LABELS, Springer, 2017.

[10] N. Chapman and J. Chapman, *Digital multimedia*, Wiley Publishing, 2009. ISBN-10 0470512164.

[11] H.I. Choi, S.W. Choi, and H.P. Moon, *Mathematical theory of medial axis transform*, Pacific Journal of Mathematics, 181 (1997), pp. 57–88.

[12] M.G. Crandall and P. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Transactions of the American Mathematical Society, 277 (1983), pp. 1–42.

[13] R. Fabbri, L.F. Costa, J.C. Torelli, and O.M. Bruno, *2D Euclidean distance transform algorithms: A comparative survey*, ACM Computing Surveys (CSUR), 40 (2008), pp. 1–44. https://doi.org/10.1145/1322432.1322434.

[14] P.F. Felzenszwalb and D.P. Huttenlocher, *Distance transforms of sampled functions*, Theory of Computing, 8 (2012), pp. 415–428. http://dx.doi.org/10.4086/toc.2012.v008a019.

[15] A.J. Frank, J.D. Daniels, and D.R. Unangst, *Progressive image transmission using a growth-geometry coding*, Proceedings of the IEEE, 68 (1980), pp. 897–909.

[16] B.B. Kimia, A.R. Tannenbaum, and S.W. Zucker, *Shapes, shocks, and deformations I: the components of two-dimensional shape and the reaction-diffusion space*, International Journal of Computer Vision, 15 (1995), pp. 189–224.

[17] I. Kunttu, L. Lepisto, J. Rauhamaa, and A. Visa, *Multiscale Fourier descriptor for shape classification*, in International Conference on Image Analysis and Processing, IEEE, 2003, pp. 536–541. https://doi.org/10.1109/ICIAP.2003.1234105.

[18] L.J. Latecki, R. Lakamper, and T. Eckhardt, *Shape descriptors for non-rigid shapes with a single closed contour*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, IEEE, 2000, pp. 424–429. https://doi.org/10.1109/CVPR.2000.855850.

[19] M. Levandowsky and D. Winter, *Distance between sets*, Nature, 234 (1971), pp. 34–35.

[20] F. Leymarie and M.D. Levine, *Simulating the grassfire transform using an active contour model*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1992), pp. 56–75. https://doi.org/10.1109/34.107013.

[21] P. Maragos, *Pattern spectrum and multiscale shape representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11 (1989), pp. 701–716.

[22] U. Montanari, *A method for obtaining skeletons using a quasi-Euclidean distance*, Journal of the ACM (JACM), 15 (1968), pp. 600–624.

[23] ——, *Continuous skeletons from digitized images*, Journal of the ACM (JACM), 16 (1969), pp. 534–549.

[24] S. Nathan and P. Kansal, *SkeletonNet: Shape pixel to skeleton pixel*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019. https://doi.org/10.1109/CVPRW.2019.00156.

[25] R.L. Ogniewicz and M. Ilg, *Voronoi skeletons: theory and applications.*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 92, 1992, pp. 63–69. https://doi.org/10.1109/CVPR.1992.223226.

[26] T. Pavlidis, *Algorithms for graphics and image processing*, Springer Science & Business Media, 2012. ISBN 978-3-642-93208-3.

[27] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 867–884. https://doi.org/10.1137/0729053.

[28] P.K. Saha, G. Borgefors, and G.S. di Baja, *A survey on skeletonization algorithms and their applications*, Pattern Recognition Letters, 76 (2016), pp. 3–12. https://doi.org/10.1016/j.patrec.2015.04.006.

[29] T.B. Sebastian and B.B. Kimia, *Curves vs. skeletons in object recognition*, Signal Processing, 85 (2005), pp. 247–263. https://doi.org/10.1016/j.sigpro.2004.10.016.

[30] K. Siddiqi, S. Bouix, A. Tannenbaum, and S.W. Zucker, *Hamilton-Jacobi skeletons*, International Journal of Computer Vision, 48 (2002), pp. 215–231. https://doi.org/10.1023/A:1016376116653.

[31] T. Sørensen, *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons*, København, I kommission hos E. Munksgaard, 1948.

[32] M.R. Teague, *Image analysis via the general theory of moments*, Journal of the Optical Society of America, 70 (1980), pp. 920–930.

[33] J. Toriwaki, T. Saitoh, and M. Okada, *Distance transformation and skeleton for shape feature analysis*, in Visual Form, Springer, 1992, pp. 547–563. https://doi.org/10.1007/978-1-4899-0715-8_52.

[34] X. Yang, X. Bai, D. Yu, and L. Latecki, *Shape classification based on skeleton path similarity*, in International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, Springer, 2007, pp. 375–386. https://doi.org/10.1007/978-3-540-74198-5_29.

[35] D. Zhang and G. Lu, *Review of shape representation and description techniques*, Pattern Recognition, 37 (2004), pp. 1–19. https://doi.org/10.1016/j.patcog.2003.07.008.

[36] H. Zhao, *A fast sweeping method for Eikonal equations*, Mathematics of Computation, 74 (2005), pp. 603–627. https://www.jstor.org/stable/4100081.

[37] H. Zhao, S. Osher, B. Merriman, and M. Kang, *Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method*, Computer Vision and Image Understanding, 80 (2000), pp. 295–314. https://doi.org/10.1006/cviu.2000.0875.