Representation

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Introduction

- The border set of a segmented object may be represented by multiple contours/surfaces in 2D/3D, which could also be organized in a hierarchical representation.

Introduction

- The border set of a segmented object may be represented by multiple contours/surfaces in 2D/3D, which could also be organized in a hierarchical representation.

- In the case of a contour, our focus, the representation is often closed, connected, and oriented — i.e., a Jordan curve.

Introduction

- The border set of a segmented object may be represented by multiple contours/surfaces in 2D/3D, which could also be organized in a hierarchical representation.
- In the case of a contour, our focus, the representation is often closed, connected, and oriented — i.e., a Jordan curve.
- Each contour defines a shape whose properties can be measured and stored in a feature vector.

Introduction

- The distance between shapes must take into account their size and orientation, being the pair feature extraction algorithm and distance function the constituent parts of the shape descriptor.

Introduction

- The distance between shapes must take into account their size and orientation, being the pair feature extraction algorithm and distance function the constituent parts of the shape descriptor.

- The shapes and the subimage around the object, with or without background, can also be represented in the frequency domain.

Introduction

- The distance between shapes must take into account their size and orientation, being the pair feature extraction algorithm and distance function the constituent parts of the shape descriptor.

- The shapes and the subimage around the object, with or without background, can also be represented in the frequency domain.

- The understanding about space-frequency transforms and multiscale filtering is crucial for the design of shape and image descriptors.

Agenda

This module will then cover the following topics related to the descriptors presented in the next module.

Agenda

This module will then cover the following topics related to the descriptors presented in the next module.

- Shape representation as a Jordan curve.

Agenda

This module will then cover the following topics related to the descriptors presented in the next module.

- Shape representation as a Jordan curve.
- Multiscale shape representations: contours and skeletons.

Agenda

This module will then cover the following topics related to the descriptors presented in the next module.

- Shape representation as a Jordan curve.
- Multiscale shape representations: contours and skeletons.
- Salience points on the shape and tensor scale of its internal points.

Agenda

This module will then cover the following topics related to the descriptors presented in the next module.

- Shape representation as a Jordan curve.
- Multiscale shape representations: contours and skeletons.
- Salience points on the shape and tensor scale of its internal points.
- Space-frequency transforms and multiscale filtering: From Fourier to Wavelets.

Shape representation as a Jordan curve

- Recall that live wire segmentation naturally extracts a shape as a Jordan curve.

Shape representation as a Jordan curve

- Recall that live wire segmentation naturally extracts a shape as a Jordan curve.

- When this is not the case, we need a contour pixel labeling algorithm to extract from the border set $\mathcal{B}$ each shape.

Shape representation as a Jordan curve

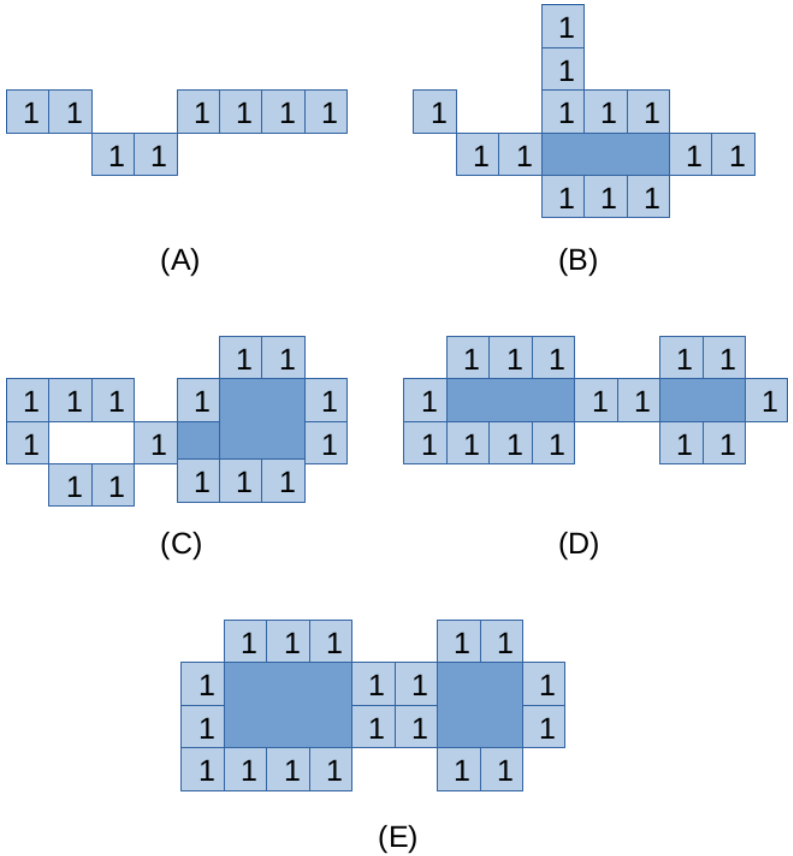
- Recall that live wire segmentation naturally extracts a shape as a Jordan curve.
- When this is not the case, we need a contour pixel labeling algorithm to extract from the border set $\mathcal{B}$ each shape.
- This algorithm must assign to each pixel in the shape a subsequent integer number, while circumscribing the shape in clockwise (or anti-clockwise) orientation.

Shape representation as a Jordan curve

- Recall that live wire segmentation naturally extracts a shape as a Jordan curve.
- When this is not the case, we need a contour pixel labeling algorithm to extract from the border set $\mathcal{B}$ each shape.
- This algorithm must assign to each pixel in the shape a subsequent integer number, while circumscribing the shape in clockwise (or anti-clockwise) orientation.
- The task is actually challenging since undesirable situations might appear when visiting border pixels.

Challenges for contour pixel labeling

Noise from segmentation might create (A) an open curve, (B) a contour with branches, and (C) a contour with bottleneck pixels. (D) Two contours might touch each other and (E) a contour might touch its opposite side.



Challenges for contour pixel labeling

- A morphological dilation of the object in the binary image $\hat{I} = (D_{\hat{I}}, I)$ can reduce some of these problems.

Challenges for contour pixel labeling

- A morphological dilation of the object in the binary image $\hat{I} = (D_{\hat{I}}, I)$ can reduce some of these problems.

- The dilation creates an image $\hat{J} = (D_{\hat{I}}, J)$ such that

$$J(p) = \max_{\forall q \in \mathcal{A}_r(p)} \{I(q)\},$$

where $\mathcal{A}_r$ is from now on defined as

$$\mathcal{A}_r : \{(p,q) \in D_{\hat{I}} \times D_{\hat{I}} \mid \|q - p\| \leq r\}.$$

Challenges for contour pixel labeling

- A morphological dilation of the object in the binary image $\hat{I} = (D_{\hat{I}}, I)$ can reduce some of these problems.

- The dilation creates an image $\hat{J} = (D_{\hat{I}}, J)$ such that

$$J(p) \quad = \quad \max_{\forall q \in \mathcal{A}_r(p)} \{I(q)\},$$

where $\mathcal{A}_r$ is from now on defined as

$$\mathcal{A}_r \quad : \quad \{(p,q) \in D_{\hat{I}} \times D_{\hat{I}} \mid \|q - p\| \leq r\}.$$

- In any case, the algorithm must eliminate open curves, branches, ears at bottleneck pixels, and assign labels to pixels of touching contours and of contours that touches their opposite side.

Contour pixel labeling

- For a given binary image $\hat{I} = (D_{\hat{I}}, I)$, the algorithm must be constrained to the border set $\mathcal{B}$.

$$\mathcal{B} \quad : \quad \{p \in D_{\hat{I}} \mid I(p) = 1 \text{ and } \exists q \in \mathcal{A}_1(p) \mid I(q) = 0\}.$$

Contour pixel labeling

- For a given binary image $\hat{I} = (D_{\hat{I}}, I)$, the algorithm must be constrained to the border set $\mathcal{B}$.

$$\mathcal{B} \quad : \quad \{p \in D_{\hat{I}} \mid I(p) = 1 \text{ and } \exists q \in \mathcal{A}_1(p) \mid I(q) = 0\}.$$

- For each contour, the algorithm must return a path $\pi_{p_1 \to p_n}$, wherein $(p_i, p_{i+1}) \in \mathcal{A}_{\sqrt{2}}$, $i = 1, 2, \ldots, n-1$, being n the size of the contour.

Contour pixel labeling

- For a given binary image $\hat{I} = (D_{\hat{I}}, I)$, the algorithm must be constrained to the border set $\mathcal{B}$.

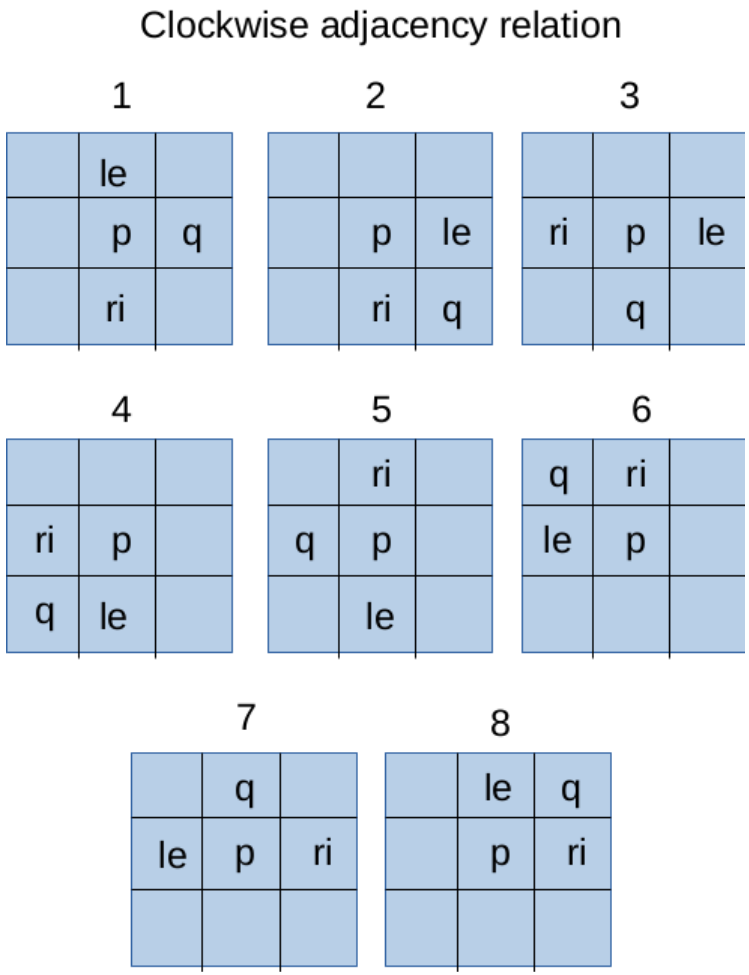
$$\mathcal{B} \quad : \quad \{p \in D_{\hat{I}} \mid I(p) = 1 \text{ and } \exists q \in \mathcal{A}_1(p) \mid I(q) = 0\}.$$

- For each contour, the algorithm must return a path $\pi_{p_1 \rightarrow p_n}$, wherein $(p_i, p_{i+1}) \in \mathcal{A}_{\sqrt{2}}$, $i = 1, 2, \ldots, n-1$, being n the size of the contour.

- It must also avoid path arcs $\langle p, q \rangle$ which pass through the object/background. That is, the left $le(p, q)$ and right $ri(p, q)$ pixels of the arc must be such that $I(le) \neq I(ri)$.

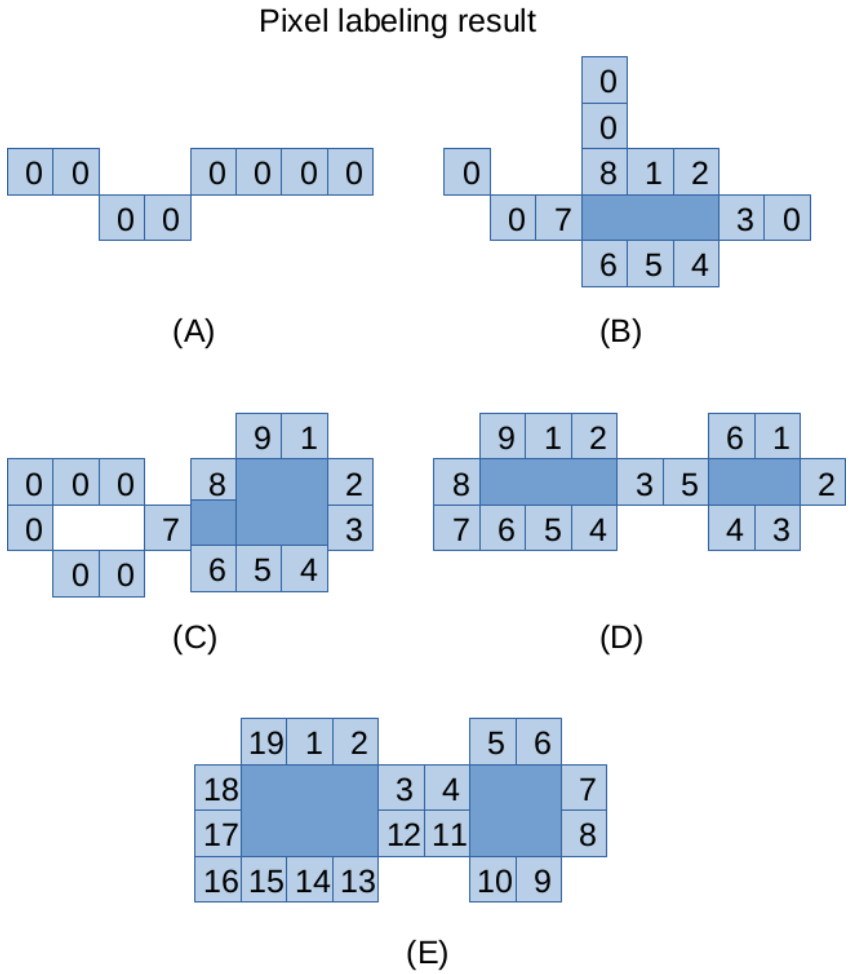
Contour pixel labeling

It avoids starting pixels that do not define valid arcs and visits the border pixels in depth search using clockwise adjacency relation $\mathcal{A}_{\sqrt{2}}$.



Contour pixel labeling

It avoids starting pixels that do not define valid arcs and visits the border pixels in depth search using clockwise adjacency relation $\mathcal{A}_{\sqrt{2}}$.



Contour pixel labeling

The first two lines of the algorithm are dedicated to define the border set $\mathcal{B}$ (initially empty), initialize a predecessor map P, and a label map L_{px}.

Contour pixel labeling

The first two lines of the algorithm are dedicated to define the border set $\mathcal{B}$ (initially empty), initialize a predecessor map P, and a label map L_{px}.

1. For each $p \in D_{\hat{I}}$, set $P(p) \leftarrow nil$ and $L_{px}(p) \leftarrow 0$.

2. If $I(p) = 1$ and $\exists q \in \mathcal{A}_1(p) \mid I(q) = 0$, insert p in $\mathcal{B}$.

Contour pixel labeling

- Two conditions are used for a border pixel p and an arc $\langle p, q \rangle$ be considered a valid starting pixel and a valid arc, respectively.

  - Valid starting pixel p: $p \in \mathcal{B}$, $P(p) = nil$, and $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid q \in \mathcal{B}$, $I(le(p,q)) \neq I(ri(p,q))$.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, and $I(le(p,q)) \neq I(ri(p,q))$.

Contour pixel labeling

- Two conditions are used for a border pixel p and an arc $\langle p, q \rangle$ be considered a valid starting pixel and a valid arc, respectively.
  - Valid starting pixel p: $p \in \mathcal{B}$, $P(p) = nil$, and $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid q \in \mathcal{B}$, $I(le(p,q)) \neq I(ri(p,q))$.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, and $I(le(p,q)) \neq I(ri(p,q))$.
- Let's indicate these events by $Valid(p)$ and $Valid(p,q)$.

Contour pixel labeling

- Two conditions are used for a border pixel p and an arc $\langle p, q \rangle$ be considered a valid starting pixel and a valid arc, respectively.
  - Valid starting pixel p: $p \in \mathcal{B}$, $P(p) = nil$, and $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid q \in \mathcal{B}$, $I(le(p, q)) \neq I(ri(p, q))$.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, and $I(le(p, q)) \neq I(ri(p, q))$.
- Let's indicate these events by $Valid(p)$ and $Valid(p, q)$.
- The subsequent lines visit each contour in the border set $\mathcal{B}$ in anti-clockwise until the starting pixel is found again. Then they follow the contour backwards in a predecessor map P to label its pixels in clockwise.

Contour pixel labeling

03. For each $s \in D_{\hat{I}} \mid Valid(s)$, do.
04. Set $P(s) \leftarrow s$ and insert s in $\mathcal{Q}$.
05. While $\mathcal{Q} \neq \emptyset$, do.
06. Remove p from $\mathcal{Q}$.
07. For each $q \in \mathcal{A}_{\sqrt{2}}(p)$, do.
08. If $q = s$, $P(p) \neq s$, and $P(p) \neq q$, then go to 12.
09. If $Valid(s, q)$, then.
10. Set $P(q) \leftarrow p$.
11. Insert q in $\mathcal{Q}$.
12. Set $P(q) \leftarrow p$, $p \leftarrow q$, $i \leftarrow 1$, and $\mathcal{Q} \leftarrow \emptyset$.
13. While $P(p) \neq p$, do.
14. Set $L_{px}(p) \leftarrow i$, $p \leftarrow P(p)$ and $i \leftarrow i + 1$.
15. Set $L_{px}(p) \leftarrow i$.

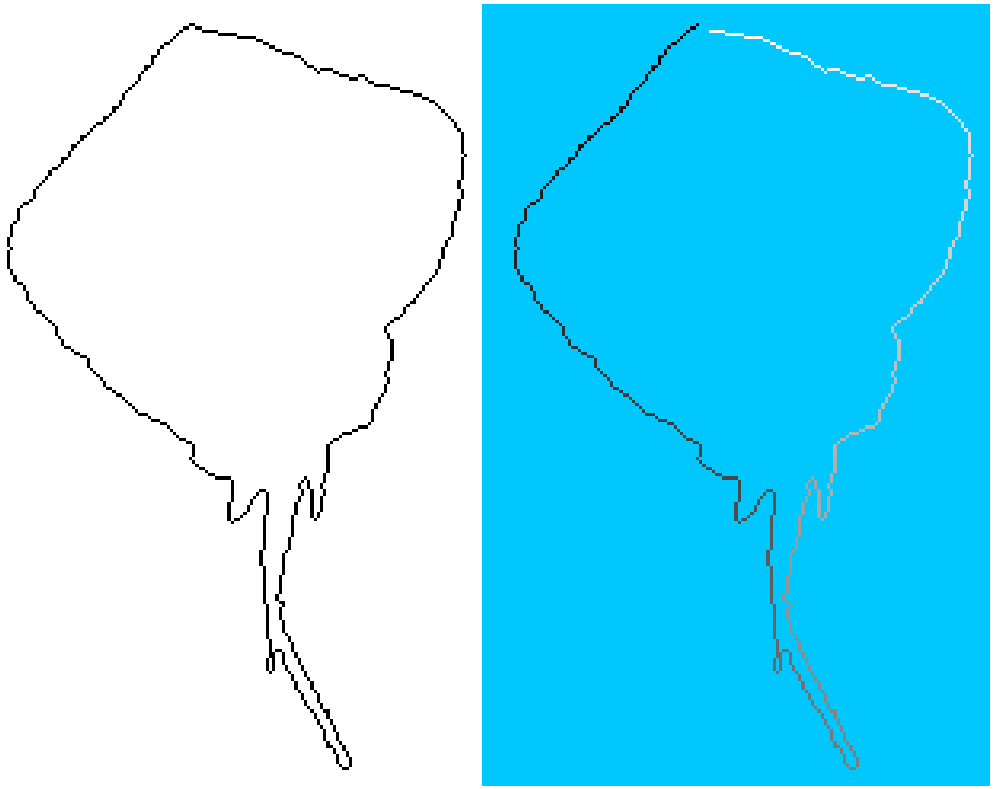
Contour pixel labeling

03. For each $s \in D_{\hat{I}} \mid Valid(s)$, do.
04. Set $P(s) \leftarrow s$ and insert s in $\mathcal{Q}$.
05. While $\mathcal{Q} \neq \emptyset$, do.
06. Remove p from $\mathcal{Q}$.
07. For each $q \in \mathcal{A}_{\sqrt{2}}(p)$, do.
08. If $q = s$, $P(p) \neq s$, and $P(p) \neq q$, then go to 12.
09. If $Valid(s, q)$, then.
10. Set $P(q) \leftarrow p$.
11. Insert q in $\mathcal{Q}$.
12. Set $P(q) \leftarrow p$, $p \leftarrow q$, $i \leftarrow 1$, and $\mathcal{Q} \leftarrow \emptyset$.
13. While $P(p) \neq p$, do.
14. Set $L_{px}(p) \leftarrow i$, $p \leftarrow P(p)$ and $i \leftarrow i + 1$.
15. Set $L_{px}(p) \leftarrow i$.

One can easily change it to assign a distinct number $L_{ct}(p)$ to the pixels of each contour in $\mathcal{B}$.

Contour pixel labeling

An example of the shape of a fish (left) and its contour pixels labeled from 1 to n in clockwise orientation (right) — brighter the pixel lower is the label.



Multiscale shape representation: Contours

- Let $\mathcal{S} \subset \mathcal{B}$ be a set of contour pixels extracted from the border set $\mathcal{B}$, such that $L_{px}(p)$ and $L_{ct}(p)$ are known for each $p \in \mathcal{S}$.

Multiscale shape representation: Contours

- Let $\mathcal{S} \subset \mathcal{B}$ be a set of contour pixels extracted from the border set $\mathcal{B}$, such that $L_{px}(p)$ and $L_{ct}(p)$ are known for each $p \in \mathcal{S}$.

- Let $(D_{\hat{I}}, \mathcal{A}_r)$ be a non-weighted graph for the binary image $\hat{I} = (D_{\hat{I}}, I)$ and adjacency relation $\mathcal{A}_r$.

Multiscale shape representation: Contours

- Let $\mathcal{S} \subset \mathcal{B}$ be a set of contour pixels extracted from the border set $\mathcal{B}$, such that $L_{px}(p)$ and $L_{ct}(p)$ are known for each $p \in \mathcal{S}$.

- Let $(D_{\hat{I}}, \mathcal{A}_r)$ be a non-weighted graph for the binary image $\hat{I} = (D_{\hat{I}}, I)$ and adjacency relation $\mathcal{A}_r$.

- Let f_{edt} be a connectivity function defined as

$$\begin{aligned} f_{edt}(\langle p \rangle) &= \begin{cases} 0 & \text{if } p \in \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases} \\ f_{edt}(\pi_p \cdot \langle p, q \rangle) &= \|q - R(p)\|^2, \end{aligned}$$

  where $R(p) \in \mathcal{S}$ is the root pixel of π_p.

Multiscale shape representation: Contours

- Let $\mathcal{S} \subset \mathcal{B}$ be a set of contour pixels extracted from the border set $\mathcal{B}$, such that $L_{px}(p)$ and $L_{ct}(p)$ are known for each $p \in \mathcal{S}$.

- Let $(D_{\hat{I}}, \mathcal{A}_r)$ be a non-weighted graph for the binary image $\hat{I} = (D_{\hat{I}}, I)$ and adjacency relation $\mathcal{A}_r$.

- Let f_{edt} be a connectivity function defined as

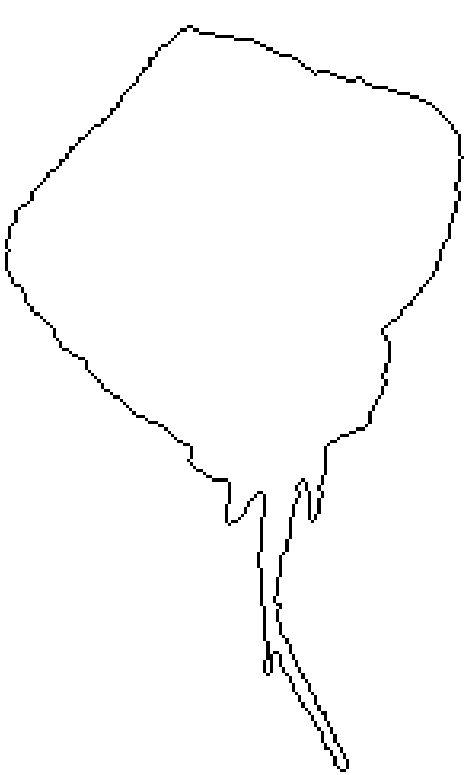
$$\begin{aligned} f_{edt}(\langle p \rangle) &= \begin{cases} 0 & \text{if } p \in \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases} \\ f_{edt}(\pi_p \cdot \langle p, q \rangle) &= \|q - R(p)\|^2, \end{aligned}$$

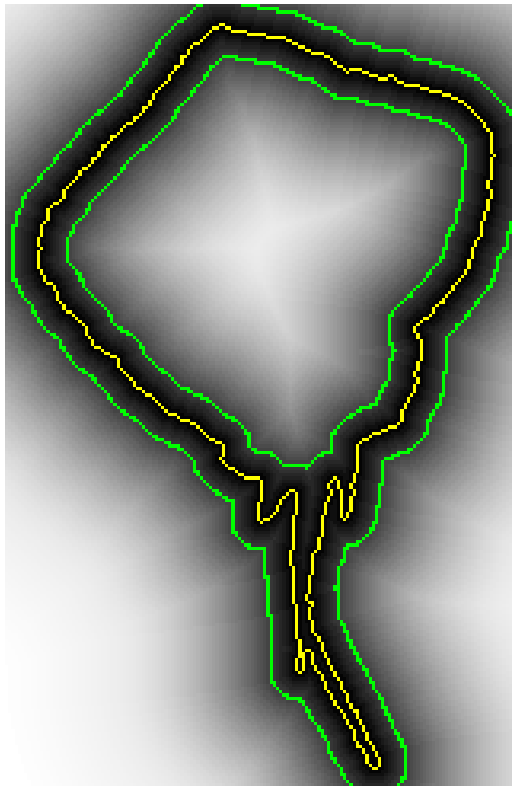
  where $R(p) \in \mathcal{S}$ is the root pixel of π_p.

- For some $r \geq \sqrt{2}$, the minimization of the connectivity map $V(p) = \min_{\forall \pi_p \in \Pi}\{f_{edt}(\pi_p)\}$ assigns to each pixel $p \in D_I$ the squared Euclidean distance to its closest pixel $R(p) \in \mathcal{S}$.

Multiscale shape representation: Contours

This transformation is named Euclidean distance transform of $\mathcal{S}$.





When $\mathcal{S}$ contains a single contour, it creates in the connectivity map V multiscale contours (iso-contours) by subsequent exact dilations and erosions of $\mathcal{S}$ [**?**].

Euclidean distance transform (EDT)

The IFT algorithm for f_{edt} can execute in $O(|D_{\hat{I}}|)$ for small values of $r \geq \sqrt{2}$ due to the use of squared Euclidean distances.

Euclidean distance transform (EDT)

The IFT algorithm for f_{edt} can execute in $O(|D_{\hat{I}}|)$ for small values of $r \geq \sqrt{2}$ due to the use of squared Euclidean distances.

Input:	Binary image $\hat{I} = (D_{\hat{I}}, I)$, seed set $\mathcal{S}$, and adjacency relation $\mathcal{A}_r$.
Output:	Connectivity map V.
Auxiliary:	Priority queue $\mathcal{Q}$ based on bucket sort, root map R, and variable tmp.

Euclidean distance transform (EDT)

The IFT algorithm for f_{edt} can execute in $O(|D_{\hat{I}}|)$ for small values of $r \geq \sqrt{2}$ due to the use of squared Euclidean distances.

Input:	Binary image $\hat{I} = (D_{\hat{I}}, I)$, seed set $\mathcal{S}$, and adjacency relation $\mathcal{A}_r$.
Output:	Connectivity map V.
Auxiliary:	Priority queue $\mathcal{Q}$ based on bucket sort, root map R, and variable tmp.

For contours and surfaces, $r = \sqrt{2}$ and $r = \sqrt{3}$ are enough, respectively.

Euclidean distance transform (EDT)

01. For each $p \in D_{\hat{I}}$, do.
02. Set $V(p) \leftarrow +\infty$.
03. If $p \in \mathcal{S}$ then set $V(p) \leftarrow 0$ and $R(p) \leftarrow p$.
04. Insert p in $\mathcal{Q}$.
05. While $\mathcal{Q} \neq \emptyset$, do.
06. Remove p from $\mathcal{Q}$ such that $p = \arg\min_{\forall q \in \mathcal{Q}}\{V(q)\}$.
07. For each $q \in \mathcal{A}_r(p)$ such that $V(q) > V(p)$, do.
08. Set $tmp \leftarrow \|q - R(p)\|^2$.
09. If $tmp < V(q)$, then.
10. Set $V(q) \leftarrow tmp$ and $R(q) \leftarrow R(p)$.

Euclidean distance transform (EDT)

01. For each $p \in D_{\hat{I}}$, do.
02. Set $V(p) \leftarrow +\infty$.
03. If $p \in \mathcal{S}$ then set $V(p) \leftarrow 0$ and $R(p) \leftarrow p$.
04. Insert p in $\mathcal{Q}$.
05. While $\mathcal{Q} \neq \emptyset$, do.
06. Remove p from $\mathcal{Q}$ such that $p = \arg\min_{\forall q \in \mathcal{Q}}\{V(q)\}$.
07. For each $q \in \mathcal{A}_r(p)$ such that $V(q) > V(p)$, do.
08. Set $tmp \leftarrow \|q - R(p)\|^2$.
09. If $tmp < V(q)$, then.
10. Set $V(q) \leftarrow tmp$ and $R(q) \leftarrow R(p)$.

Note that, we may use the values in I to constrain computation inside the object, outside it, and/or up to a threshold $V(p) \leq T$.

Multiscale shape representation: Skeletons

- Each contour is related to its internal and external skeletons (medial axes)— point sets with at least two equidistant pixels in the contour.

Multiscale shape representation: Skeletons

- Each contour is related to its internal and external skeletons (medial axes)— point sets with at least two equidistant pixels in the contour.

- By painting a ball of radius $\sqrt{V(p)}$ for each skeleton pixel p, one can reconstruct the shape. Filtered skeletons imply filtered shapes.

Multiscale shape representation: Skeletons

- Each contour is related to its internal and external skeletons (medial axes)— point sets with at least two equidistant pixels in the contour.

- By painting a ball of radius $\sqrt{V(p)}$ for each skeleton pixel p, one can reconstruct the shape. Filtered skeletons imply filtered shapes.

- From each contour in $\mathcal{S}$, one can create internal and external multiscale skeletons which are one-pixel-wide and connected in all scales.

Multiscale shape representation: Skeletons

- Each contour is related to its internal and external skeletons (medial axes)— point sets with at least two equidistant pixels in the contour.

- By painting a ball of radius $\sqrt{V(p)}$ for each skeleton pixel p, one can reconstruct the shape. Filtered skeletons imply filtered shapes.

- From each contour in $\mathcal{S}$, one can create internal and external multiscale skeletons which are one-pixel-wide and connected in all scales.

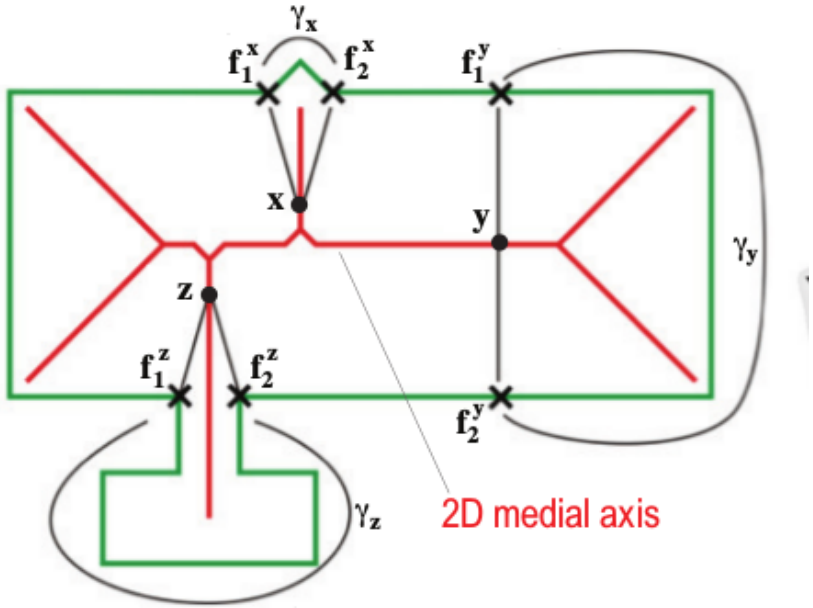
- The multiscale skeletons of each contour can be merged at the skeleton by influence zones (SKIZ — a point set with equidistant pixels in at least two contours).

Multiscale shape representation: Skeletons

- Let n_k be the number of pixels in a contour $\mathcal{S}_k \in \mathcal{S}$, $k = \{1, 2, \ldots, K\}$.

Multiscale shape representation: Skeletons

- Let n_k be the number of pixels in a contour $\mathcal{S}_k \in \mathcal{S}$, $k = \{1, 2, \ldots, K\}$.
- A multiscale skeleton $\hat{S} = (D_{\hat{I}}, S)$ is an image where $S(p)$ is the maximum geodesic length $\gamma(f_1, f_2)$ between $f_1, f_2 \in \mathcal{S}_k$ — feature (root) points equidistant to $p \in D_{\hat{I}} \backslash \mathcal{S}$ [**?**].



Multiscale shape representation: Skeletons

The geodesic length between two feature points $f_1 \in \mathcal{S}_k$ and $f_2 \in \mathcal{S}_k$ can be better estimated as

$$\gamma(f1, f2) = \min_{\forall \pi_{f_1 \to f_2} \in \Pi_{\mathcal{S}_k}} \{ \sum_{i=1}^{n_k - 1} w(p_i, p_{i+1}) \},$$

$$w(p_i, p_{i+1}) = \begin{cases} 0.9016 & \text{if } \|p_{i+1} - p_i\| = 1, \\ 1.2890 & \text{if } \|p_{i+1} - p_i\| = \sqrt{2}, \end{cases}$$

where $p_1 = f_1$, $p_n = f_2$, $\Pi_{\mathcal{S}_k}$ is the set of the two possible paths from f_1 to f_2 such that $p_i \in \mathcal{S}_k$ and $(p_i, p_{i+1}) \in \mathcal{A}_{\sqrt{2}}$, $i = 1, 2, \ldots, n_k - 1$.

Multiscale shape representation: Skeletons

After EDT computation, alternatively, and more efficiently, one can approximate geodesic lengths from the differences between $L_{px}(f_1)$ and $L_{px}(f_2)$ as follows (Lines 4 and 5 define the SKIZ).

Multiscale shape representation: Skeletons

After EDT computation, alternatively, and more efficiently, one can approximate geodesic lengths from the differences between $L_{px}(f_1)$ and $L_{px}(f_2)$ as follows (Lines 4 and 5 define the SKIZ).

1. For each $p \in D_{\hat{I}} \backslash \mathcal{S}$, do.
2. Set $S(p) \leftarrow 0$.
3. For each $q \in \mathcal{A}_1(p)$ do.
4. If $L_{ct}(R(q)) > L_{ct}(R(p))$, then.
5. Set $S(p) \leftarrow +\infty$.
6. Else if $L_{ct}(R(q)) = L_{ct}(R(p))$, then.
7. Set $\Delta \leftarrow L_{px}(R(q)) - L_{px}(R(p))$.
8. Set $S(p) \leftarrow \max\{S(p), \min_{k=L_{ct}(R(p))}\{\Delta, n_k - \Delta\}\}$.

Multiscale shape representation: Skeletons

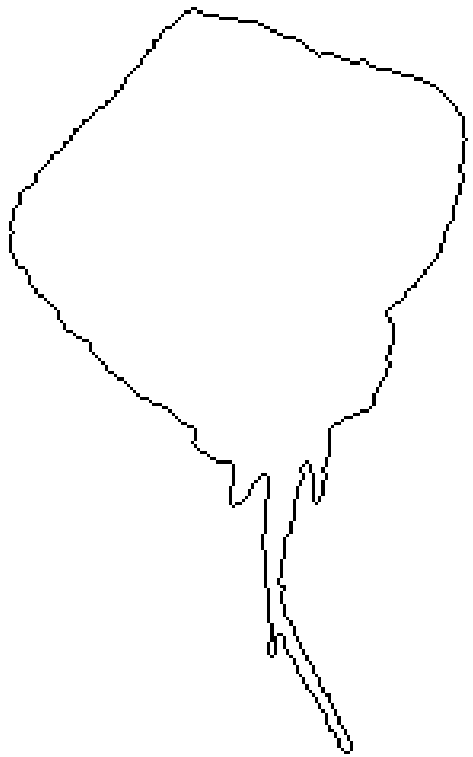
After EDT computation, alternatively, and more efficiently, one can approximate geodesic lengths from the differences between $L_{px}(f_1)$ and $L_{px}(f_2)$ as follows (Lines 4 and 5 define the SKIZ).

1. For each $p \in D_{\hat{I}} \backslash \mathcal{S}$, do.
2. Set $S(p) \leftarrow 0$.
3. For each $q \in \mathcal{A}_1(p)$ do.
4. If $L_{ct}(R(q)) > L_{ct}(R(p))$, then.
5. Set $S(p) \leftarrow +\infty$.
6. Else if $L_{ct}(R(q)) = L_{ct}(R(p))$, then.
7. Set $\Delta \leftarrow L_{px}(R(q)) - L_{px}(R(p))$.
8. Set $S(p) \leftarrow \max\{S(p), \min_{k=L_{ct}(R(p))}\{\Delta, n_k - \Delta\}\}$.

Similarly, one can store $\Gamma(p) = \gamma(f_0, f)$ for each $f \in \mathcal{S}_k$ and starting pixel f_0 and substitute L_{px} by Γ in the algorithm above.

Multiscale shape representation: Skeletons

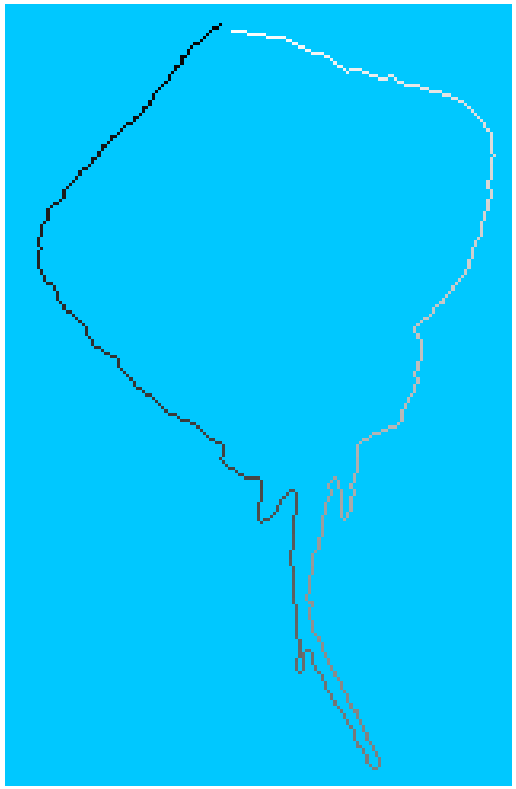
By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.



Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.



Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

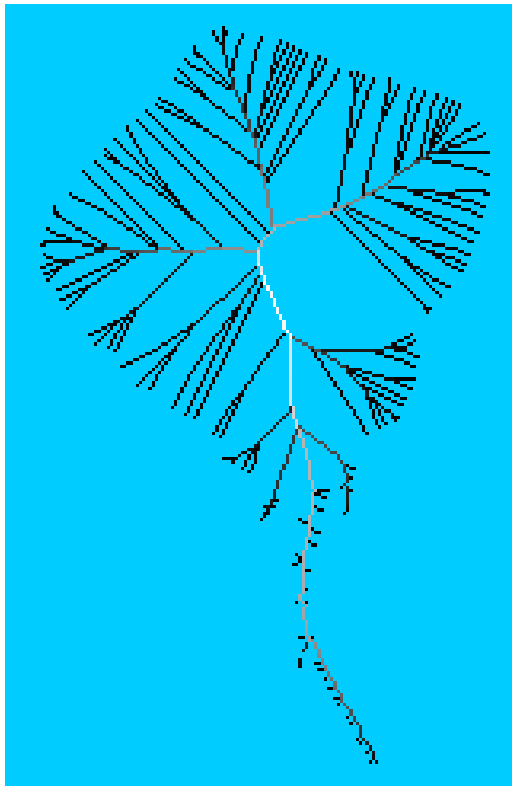
Multiscale shape representation: Skeletons

By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.

Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

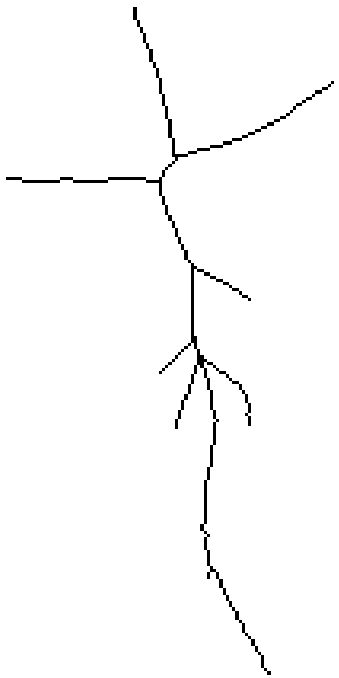
By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.



Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

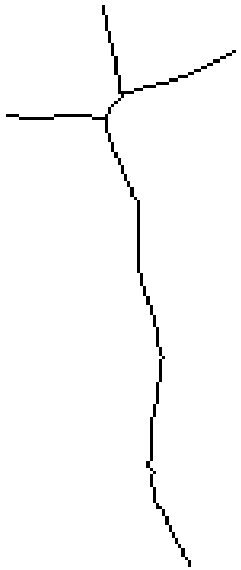
By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.



Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.



Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

By thresholding, $S(p) \geq T \ \forall p \in D_{\hat{I}}$, at a given scale value $T > 0$, one obtains an one-pixel-wide and connected skeleton. Higher is T, more simplified are the skeletons.

Note that connected skeletons are guaranteed only for $\mathcal{A}_{\sqrt{2}}$ in the EDT algorithm.

Multiscale shape representation: Skeletons

An example with multiple contours.

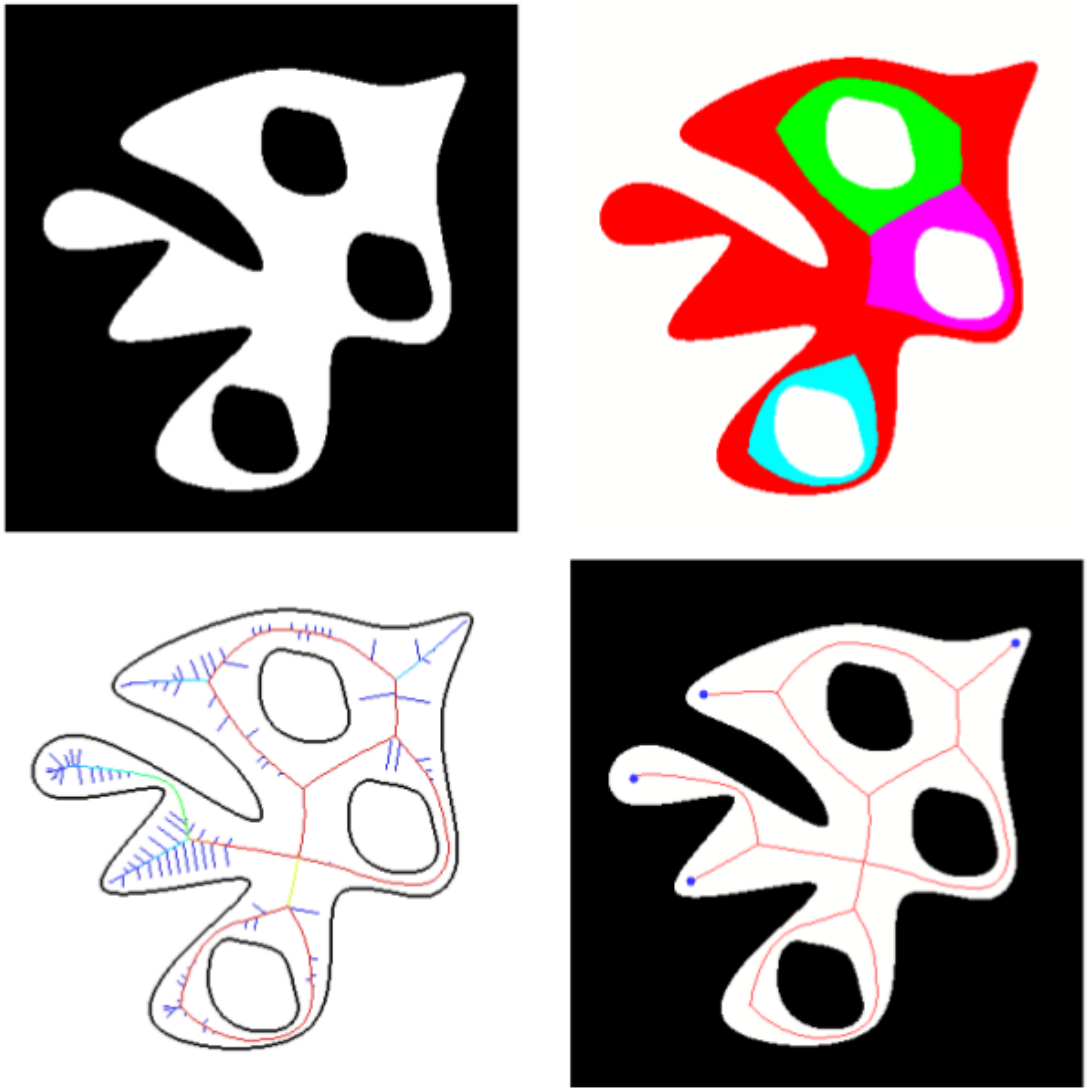


Figure from [2]

Multiscale shape representation: Skeletons

3D surface skeletons can be obtained by the direct extension of the geodesic length on surfaces as importance measure [**?**].

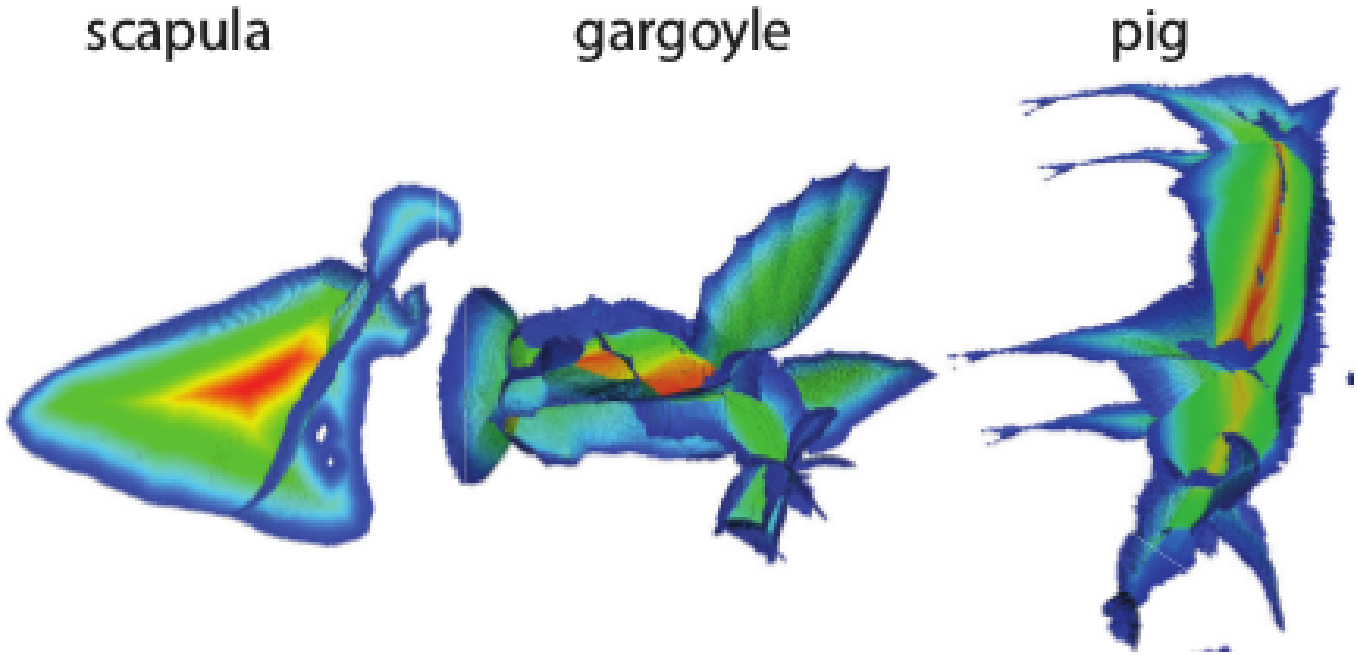


Figure from [**?**].

Multiscale shape representation: Skeletons

3D surface skeletons can be obtained by the direct extension of the geodesic length on surfaces as importance measure [**?**].

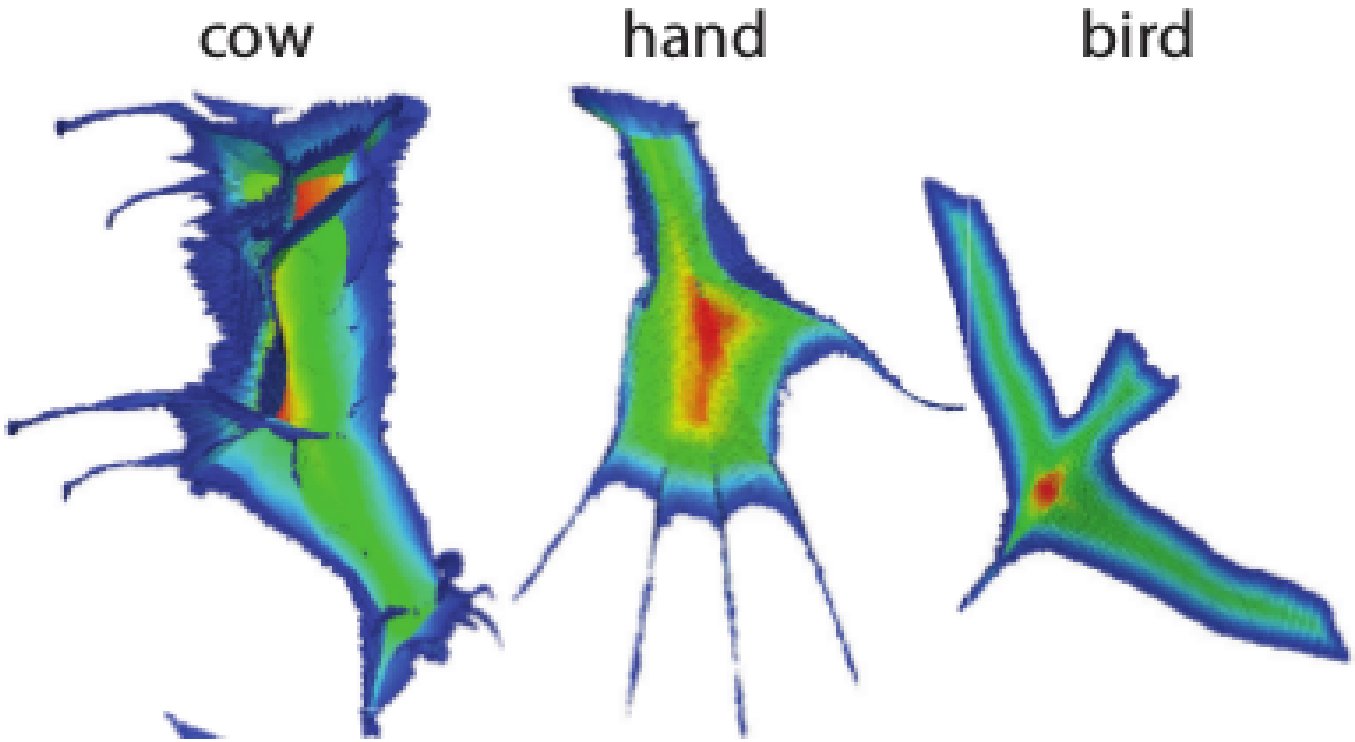
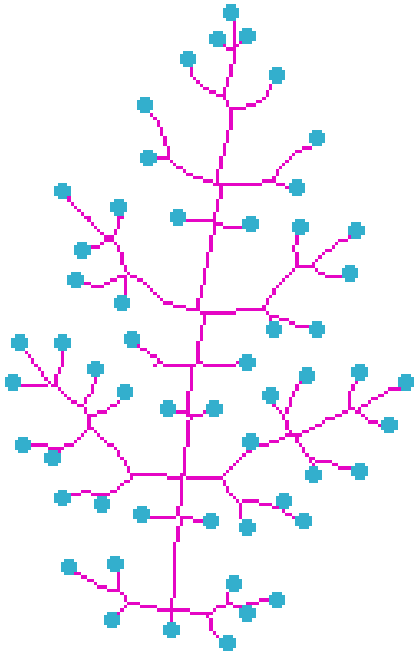


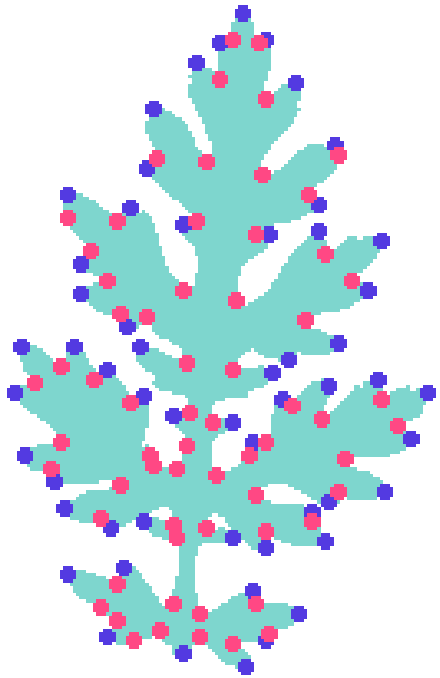
Figure from [**?**].

Salience points



Terminal points of the internal and external skeletons can be directly related to convex and concave salience points on the contour, respectively [**?**].

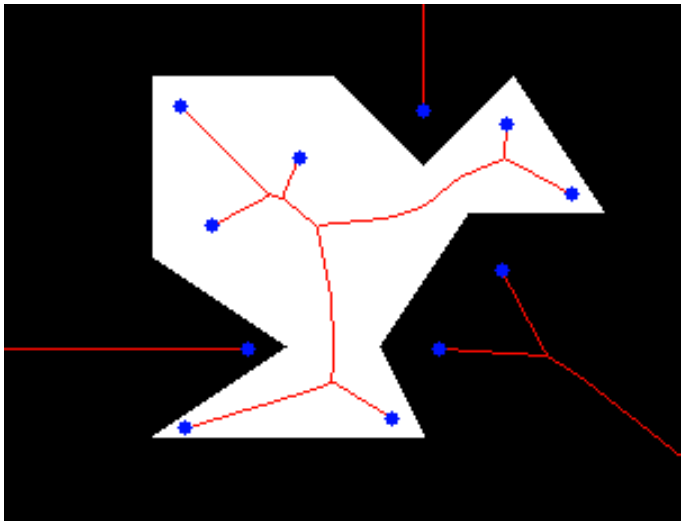
Salience points



Terminal points of the internal and external skeletons can be directly related to convex and concave salience points on the contour, respectively [**?**].

Salience points

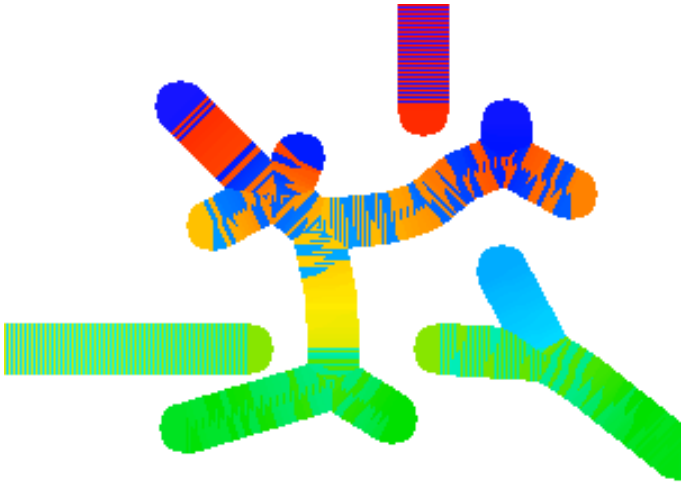
Let $\hat{Sk} = (D_{\hat{I}}, Sk)$ be a binary image, where $Sk(p) = 1$ when $p \in D_I$ is a skeleton point and $Sk(p) = 0$ otherwise. Terminal points of the skeleton can be defined as pixels $p \in D_{\hat{I}}$ with exactly one $q \in \mathcal{A}_{\sqrt{2}}(p) \backslash \{p\}$ such that $Sk(p) = Sk(q) = 1$.



Such a definition might fail if exists $q_1, q_2 \in \mathcal{A}_{\sqrt{2}}(p) \backslash \{p\}$, such that $Sk(p) = Sk(q_1) = Sk(q_2) = 1$, $q_1 \neq q_2$, and $(q_1, q_2) \in \mathcal{A}_1$.

Salience points

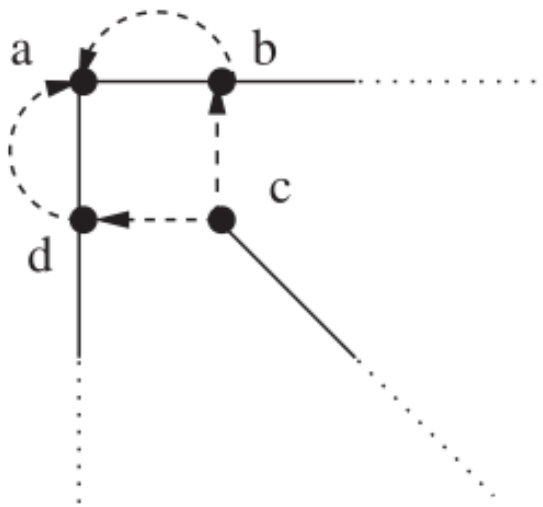
An alternative is to slightly change the scale threshold used to obtain the skeleton. Another alternative is to compute the influence zones of the skeleton points within a small distance r (e.g., 10) to it.



The area A and aperture angle θ of the influence zones are higher for terminal points. By measuring $A = \frac{\theta r^2}{2}$, it is safe to select points with θ above a threshold (e.g., $\theta > 70^o$).

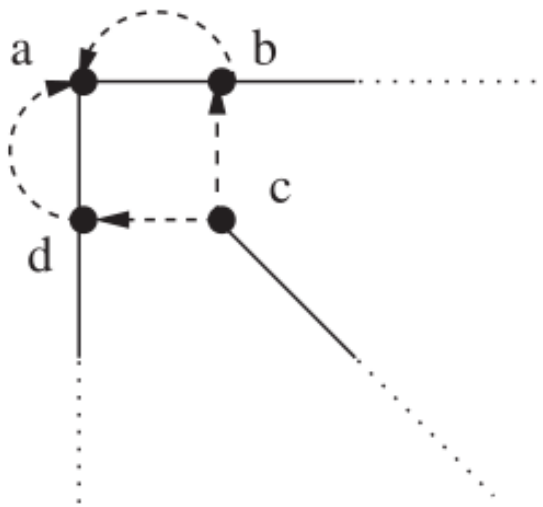
Salience points

Now, to determine which salience point a on a contour of perimeter n corresponds to a terminal point c of the skeleton, we must determine if the root $R(c) = b$ or $R(c) = d$.



Salience points

Now, to determine which salience point a on a contour of perimeter n corresponds to a terminal point c of the skeleton, we must determine if the root $R(c) = b$ or $R(c) = d$.



Let $q \in \mathcal{A}_1(c)$ be the pixel used to set $S(c)$ in the multiscale skeletonization algorithm. Then, either

(1) $R(q) = d$ and $R(c) = b$, or

(2) $R(q) = b$ and $R(c) = d$.

Salience points

- For clockwise-labeled contour pixels,
  (1) If $R(c) = b$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(b) - \frac{S(c)}{2}$.
  (2) If $R(c) = d$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(d) + \frac{S(c)}{2}$.

Salience points

- For clockwise-labeled contour pixels,
  (1) If $R(c) = b$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(b) - \frac{S(c)}{2}$.
  (2) If $R(c) = d$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(d) + \frac{S(c)}{2}$.

- For $\Delta = L_{px}(q) - L_{px}(c)$, case (1) occurs when $\Delta \geq n - \Delta$, and case (2) occurs otherwise.

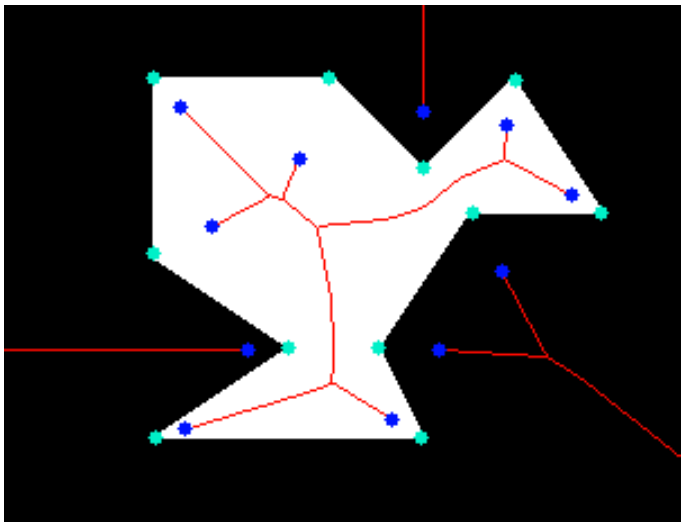
Salience points

- For clockwise-labeled contour pixels,
  (1) If $R(c) = b$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(b) - \frac{S(c)}{2}$.
  (2) If $R(c) = d$, then a is the point with $L_{px}(a)$ obtained from $L_{px}(d) + \frac{S(c)}{2}$.

- For $\Delta = L_{px}(q) - L_{px}(c)$, case (1) occurs when $\Delta \geq n - \Delta$, and case (2) occurs otherwise.

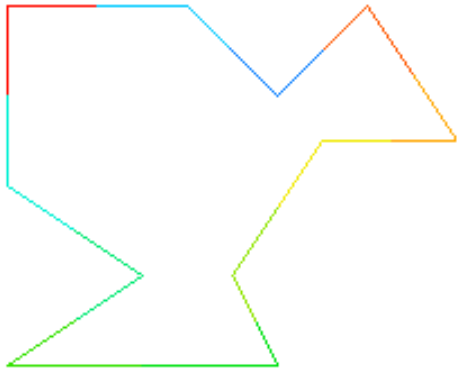
- We use a sign $s \in \{-1, 1\}$ to indicate (1) and (2), and find a by computing $\delta \leftarrow L_{px}(R(c)) + s\frac{S(c)}{2}$ as the point with
  - $L_{px}(a) = \delta$, when $\delta \in [1, n]$,
  - $L_{px}(a) = \delta - n$, when $\delta > n$, and
  - $L_{px}(a) = \delta + n$, when $\delta \leq 0$.

Salience points



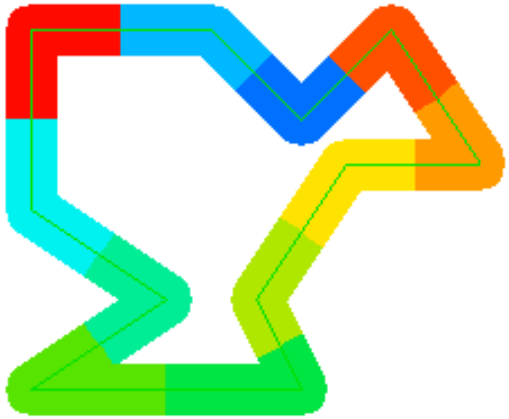
Once salience points on the contour are determined, their influence zones on the contour define segments, which have influence zones using a small distance (e.g., 10) higher outside than inside the shape when they are convex, and the other way around when they are concave.

Salience points



Once salience points on the contour are determined, their influence zones on the contour define segments, which have influence zones using a small distance (e.g., 10) higher outside than inside the shape when they are convex, and the other way around when they are concave.

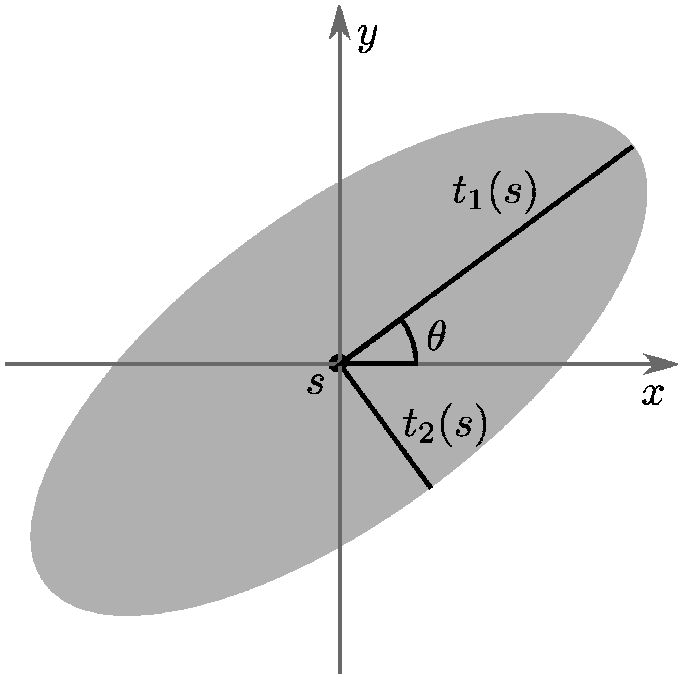
Salience points



Once salience points on the contour are determined, their influence zones on the contour define segments, which have influence zones using a small distance (e.g., 10) higher outside than inside the shape when they are convex, and the other way around when they are concave.

Tensor Scale

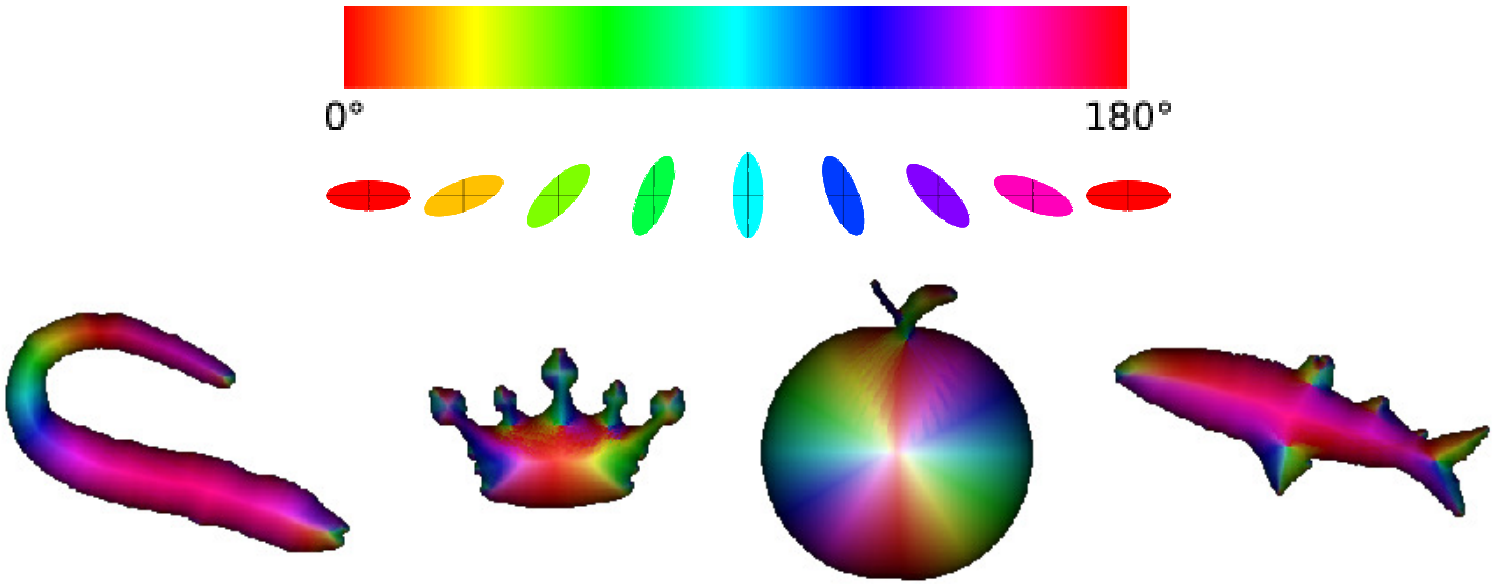
A shape can also be represented by the largest ellipse (tensor scale) centered at each pixel, creating a region-based shape representation.



- $Orientation(s)$ = angle between $t_1(s)$ and the horizontal axis.
- $Anisotropy(s) = \sqrt{1 - \frac{|t_2(s)|^2}{|t_1(s)|^2}}$.
- $Thickness(s) = |t_2(s)|$.

Tensor Scale

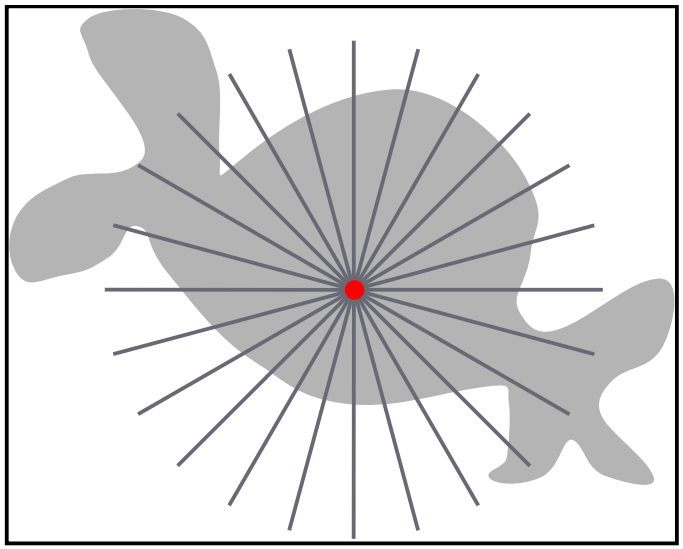
By using the HSI color space, the tensor orientation at each pixel is represented by a distinct color.



The region-based representation stores orientation and anisotropy at each pixel.

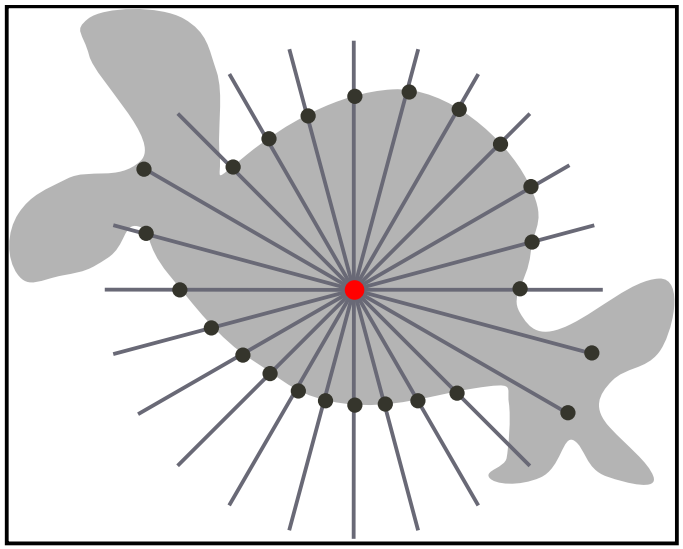
Tensor scale computation

As proposed by Saha [**?**], the tensor scale at s may be computed by tracing sample lines, finding edge points in each line, and fitting the largest ellipse through these points.



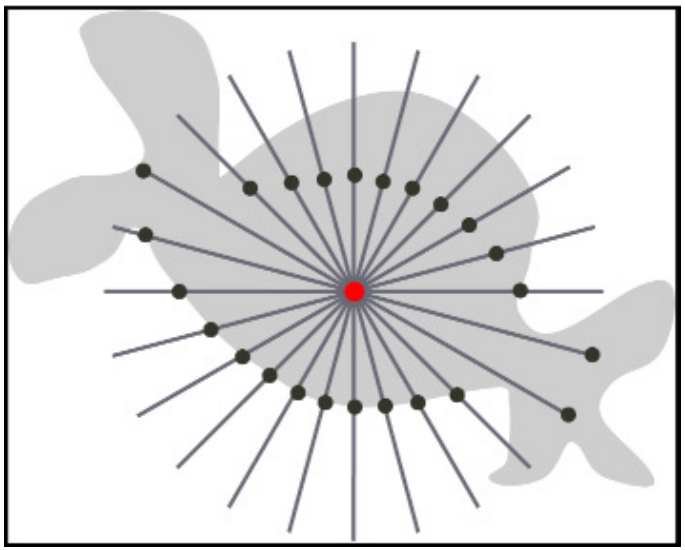
Tensor scale computation

As proposed by Saha [**?**], the tensor scale at s may be computed by tracing sample lines, finding edge points in each line, and fitting the largest ellipse through these points.



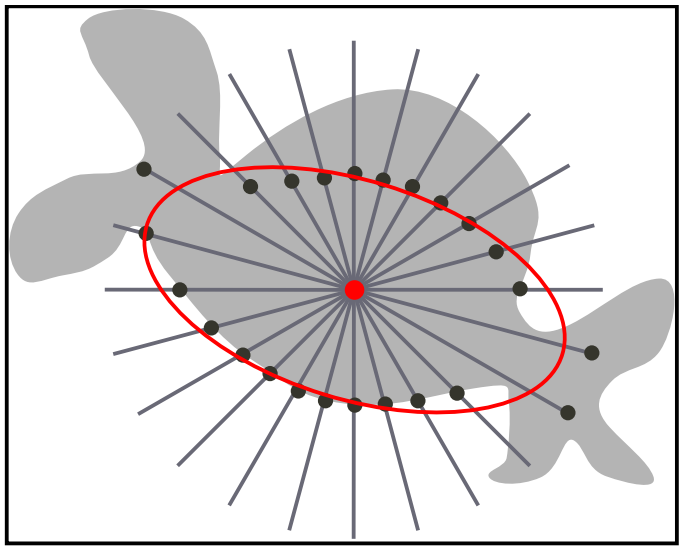
Tensor scale computation

As proposed by Saha [**?**], the tensor scale at s may be computed by tracing sample lines, finding edge points in each line, and fitting the largest ellipse through these points.



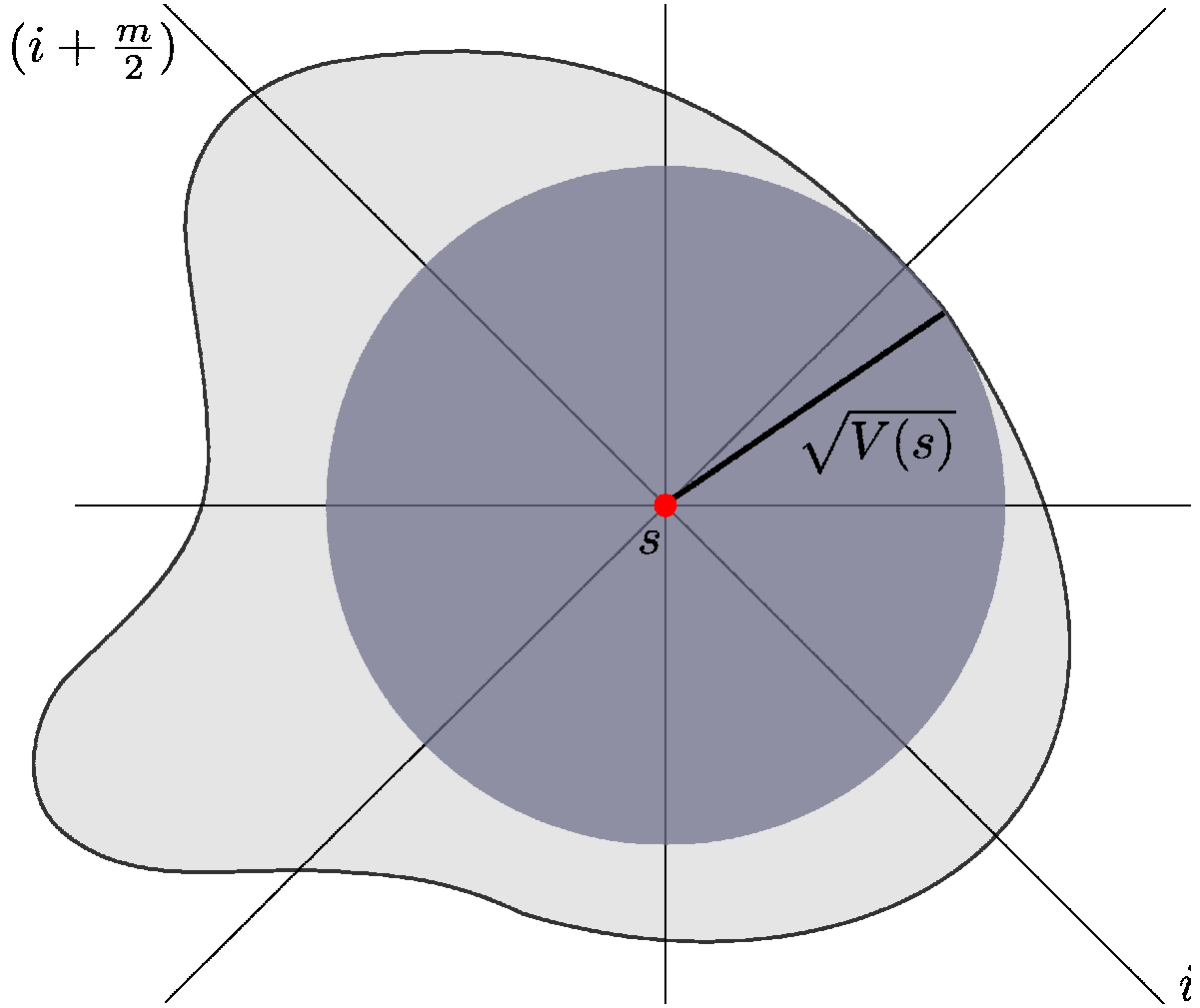
Tensor scale computation

As proposed by Saha [**?**], the tensor scale at s may be computed by tracing sample lines, finding edge points in each line, and fitting the largest ellipse through these points.



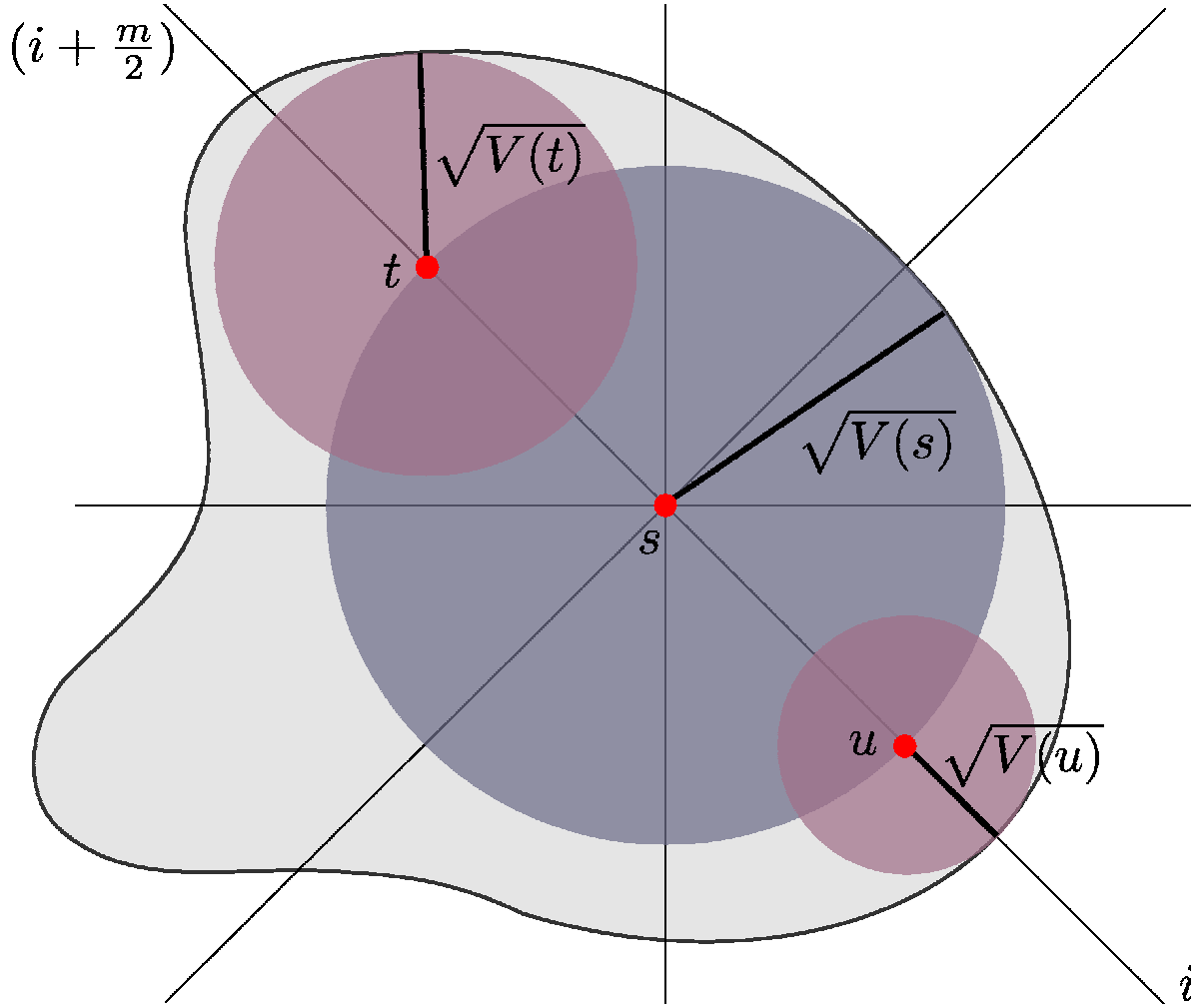
Tensor scale computation

The Euclidean IFT speeds up the search for each pair of edge points by exploiting the values in $V(s)$ [**?**].



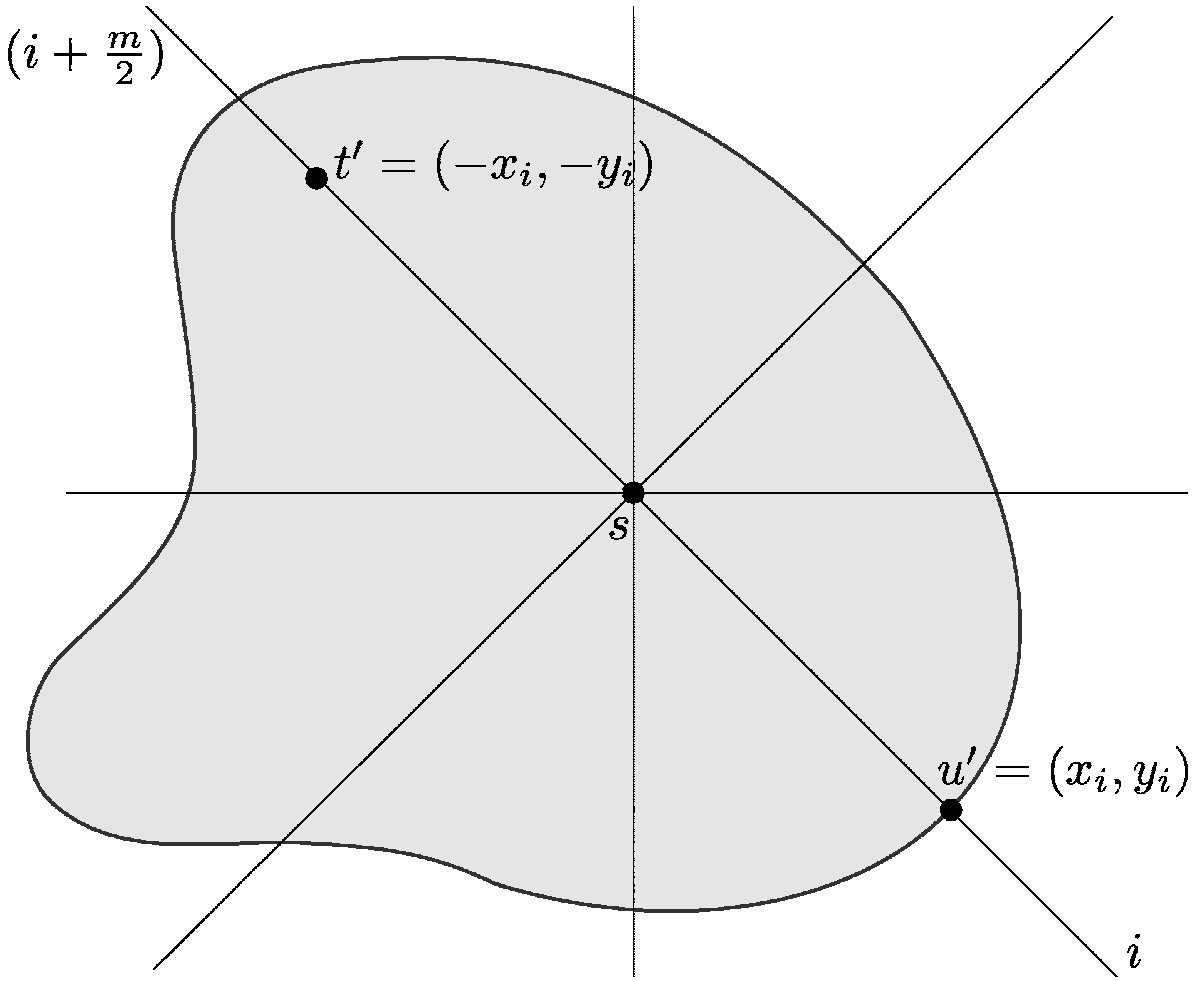
Tensor scale computation

The Euclidean IFT speeds up the search for each pair of edge points by exploiting the values in $V(s)$ [**?**].



Tensor scale computation

The Euclidean IFT speeds up the search for each pair of edge points by exploiting the values in $V(s)$ [**?**].



Tensor scale computation

The ellipse orientation is obtained by PCA from the value γ^* of γ that minimizes function g below [**?**, **?**].

$$g(\gamma) = \sum_{i=1,2,\ldots,m} [x_{i_\gamma}^2 - y_{i_\gamma}^2]$$

where

m is the number of sample lines, (x_i, y_i) are the relative coordinates of the edge points with respect $s = (x_s, y_s)$, and $(x_{i_\gamma}, y_{i_\gamma})$ are the coordinates after rotation by angle γ.

$$\begin{aligned} x_{i_\gamma} &= x_i \cos(\gamma) - y_i \sin(\gamma) \\ y_{i_\gamma} &= x_i \sin(\gamma) + y_i \cos(\gamma) \end{aligned}$$

The lengths $t_1(s)$ and $t_2(s)$ of the major and minor axes of the ellipse, respectively, are obtained from $(x_{i_{\gamma^*}}, y_{i_{\gamma^*}})$ as described in [**?**].

Space-frequency transforms

- A 2D and grayscale image $\hat{I} = (D_{\hat{I}}, I)$ is a space-domain function $I(x, y)$: $D_{\hat{I}} \subset Z^2 \rightarrow \Re$.

Space-frequency transforms

- A 2D and grayscale image $\hat{I} = (D_{\hat{I}}, I)$ is a space-domain function $I(x, y)$: $D_{\hat{I}} \subset Z^2 \rightarrow \Re$.

- Let $T(u, v) = \sum_{y=0}^{n_y-1} \sum_{x=0}^{n_x-1} I(x, y) G(x, y, u, v)$, where n_x and n_y are the number of pixels along x and y, be a general transform from the coordinates $(x, y) \in D_{\hat{I}}$ to coordinates $(u, v) \in D_T$.

Space-frequency transforms

- A 2D and grayscale image $\hat{I} = (D_{\hat{I}}, I)$ is a space-domain function $I(x, y)$: $D_{\hat{I}} \subset Z^2 \rightarrow \Re$.

- Let $T(u, v) = \sum_{y=0}^{n_y-1} \sum_{x=0}^{n_x-1} I(x, y) G(x, y, u, v)$, where n_x and n_y are the number of pixels along x and y, be a general transform from the coordinates $(x, y) \in D_{\hat{I}}$ to coordinates $(u, v) \in D_T$.

- For each $(u, v) \in D_T$ there is a basis function $G(x, y, u, v)$. Whenever they are periodic with frequencies u and v, $T(u, v)$ is said a space-frequency transform.

Space-frequency transforms

- A 2D and grayscale image $\hat{I} = (D_{\hat{I}}, I)$ is a space-domain function $I(x, y)$: $D_{\hat{I}} \subset Z^2 \rightarrow \Re$.

- Let $T(u, v) = \sum_{y=0}^{n_y-1} \sum_{x=0}^{n_x-1} I(x, y)G(x, y, u, v)$, where n_x and n_y are the number of pixels along x and y, be a general transform from the coordinates $(x, y) \in D_{\hat{I}}$ to coordinates $(u, v) \in D_T$.

- For each $(u, v) \in D_T$ there is a basis function $G(x, y, u, v)$. Whenever they are periodic with frequencies u and v, $T(u, v)$ is said a space-frequency transform.

- Such functions usually allow the inverse transform $I(x, y) = \sum_{v=0}^{n_v-1} \sum_{u=0}^{n_u-1} T(u, v)H(x, y, u, v)$, such that $n_u = n_x$ and $n_v = n_y$.

Space-frequency transforms

- The most popular example is the discrete Fourier Transform.

$$
\begin{aligned}
G(x,y,u,v) &= \exp\left[-j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right], \\
H(x,y,u,v) &= \frac{1}{n_x n_y}\exp\left[j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right],
\end{aligned}
$$

where $j=\sqrt{-1}$ and the transform $T(u,v)$ is usually complex (i.e., $\mathbf{T}(u,v)$ in our terminology).

Space-frequency transforms

- The most popular example is the discrete Fourier Transform.

$$G(x,y,u,v) = \exp\left[-j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right],$$

$$H(x,y,u,v) = \frac{1}{n_x n_y}\exp\left[j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right],$$

  where $j=\sqrt{-1}$ and the transform $T(u,v)$ is usually complex (i.e., $\mathbf{T}(u,v)$ in our terminology).

- Such general transforms can be applied to any nD function, such as the image of an object, the $x(s)$ and $y(s)$ coordinates of its external contour for $s=1,2,\ldots,n$, etc.

Space-frequency transforms

- The most popular example is the discrete Fourier Transform.

$$G(x,y,u,v) = \exp\left[-j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right],$$

$$H(x,y,u,v) = \frac{1}{n_x n_y}\exp\left[j2\pi\left(\frac{ux}{n_x}+\frac{vy}{n_y}\right)\right],$$

  where $j=\sqrt{-1}$ and the transform $T(u,v)$ is usually complex (i.e., $\mathbf{T}(u,v)$ in our terminology).

- Such general transforms can be applied to any nD function, such as the image of an object, the $x(s)$ and $y(s)$ coordinates of its external contour for $s=1,2,\ldots,n$, etc.

- The coefficients $T(u,v)$ can then be used for image/shape description.

Space-frequency transforms

It is also common that $G(x, y, u, v)$ and $H(x, y, u, v)$ are separable and symmetric,

$$\begin{aligned} G(x, y, u, v) &= G'(x, u)G'(y, v), \\ H(x, y, u, v) &= H'(x, u)H'(y, v), \end{aligned}$$

which allows to decompose nD forward (and inverse) transforms into 1D transforms.

$$\begin{aligned} T(u, v) &= \sum_{y=0}^{n_y-1} \sum_{x=0}^{n_x-1} I(x, y)G(x, y, u, v) \\ T(u, v) &= \sum_{y=0}^{n_y-1} \left[\sum_{x=0}^{n_x-1} I(x, y)G'(x, u) \right] G'(y, v) \\ T(u, v) &= \sum_{y=0}^{n_y-1} T(u, y)G'(y, v) \end{aligned}$$

Examples of transforms

For example, for $n_x = 2^n$ and $b_k(z)$ being the k-th bit in the binary representation of z, we have:

- The Walsh Transform

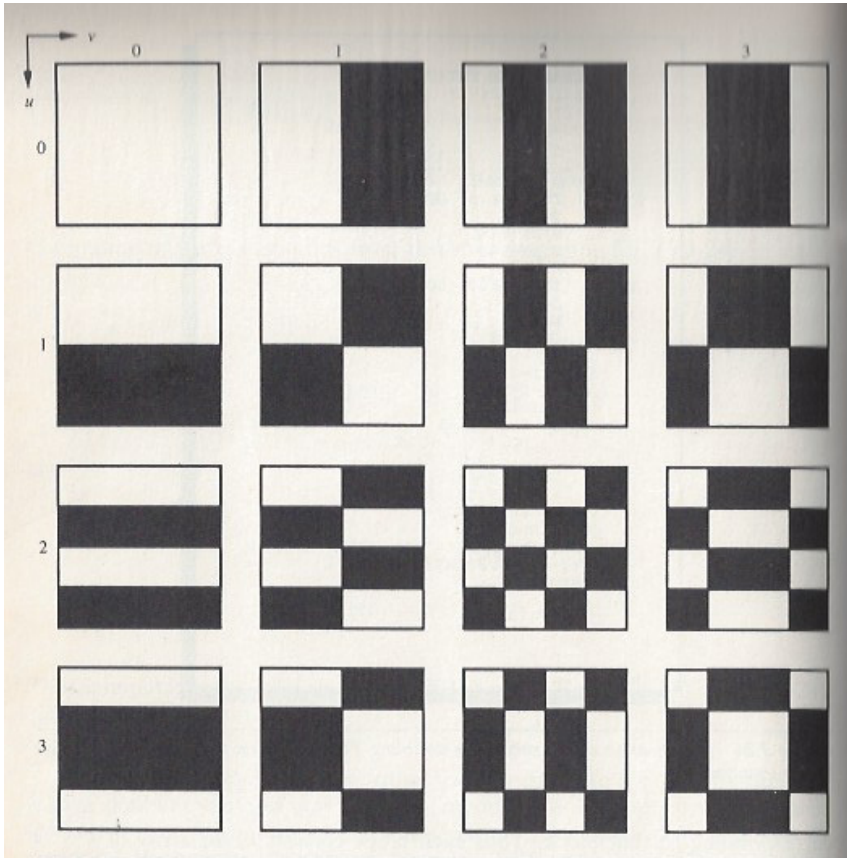
$$\begin{aligned} G'(x,u) &= \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)}, \\ H'(x,u) &= \frac{1}{n_x} \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)}. \end{aligned}$$

- The Hadamard transform

$$\begin{aligned} G'(x,u) &= (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)}, \\ H'(x,u) &= \frac{1}{n_x} (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)}, \end{aligned}$$

where the summation in the exponent is performed in mod 2.

Example of Walsh basis functions



Walsh basis functions $G(x, y, 0, 0)$, $G(x, y, 0, 1)$, ..., $G(x, y, 0, 3)$, $G(x, y, 1, 0)$, $G(x, y, 1, 1)$, ..., $G(x, y, 3, 3)$, where $x, y \in \{0, 1, 2, 3\}$, white denotes $+1$, and black denotes -1 [**?**].

Such discrete transforms stem from the Fourier series (Joseph Fourier, 1807) for periodic and continuous signals $I(x)$ and then from the Fourier transform (1822) for aperiodic signals.

$$T(u) = \int_{-\infty}^{+\infty} I(x)G^*(x)dx,$$

where $G^*(x)$ is the complex conjugate of $G(x) = \exp[j2\pi ux]$, assuming that $\int_{-\infty}^{+\infty} |I(x)|^2 dx < +\infty$.

From Fourier to Wavelets

In order to analyze frequencies u within Gaussian windows of the space (time) domain around locations $x = \tau$, Dennis Gabor (1946) proposed the short-time Fourier Transform.

$$T(u) = \int_{-\infty}^{+\infty} I(x) G_\tau^*(x) dx,$$

where $G_\tau(x) = g(x - \tau) \exp[j2\pi ux]$ and $g(x - \tau)$ is a Gaussian window around $x = \tau$.

From Fourier to Wavelets

In order to analyze frequencies u within Gaussian windows for distinct locations and scales, one can use, for instance, Morlet (Gabor) wavelet transform (Morlet, 1970 and Grossmann and Morlet, 1984).

$$T(u) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} I(x) G^*_{\tau,s}(x) dx,$$

where s is the scale and

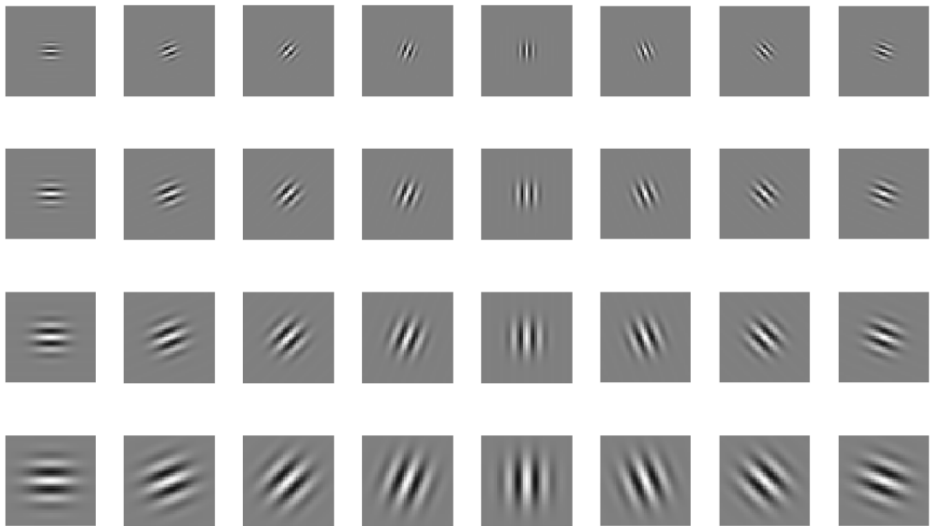
$$G_{\tau,s}(x) = \exp\left[-\pi \frac{(x-\tau)^2}{s^2}\right] \exp\left[j2\pi u\,(x-\tau)\right],$$

being the mother wavelet defined by $\tau = 0$ and $s = 1$.

From Fourier to Wavelets

In 2D, the basins functions (Gabor wavelets) become

$$G_{\tau_x,\tau_y,s_x,s_y}(x,y) = \exp\left\{-\pi\left[\frac{(x-\tau_x)^2}{s_x^2}+\frac{(y-\tau_y)^2}{s_y^2}\right]\right\} \exp\left\{j2\pi\left[u\left(x-\tau_x\right)+v\left(y-\tau_y\right)\right]\right\}.$$



The real part of the basis functions for distinct orientations and scales.

Multiscale Image Filtering

- Recall that the Fourier transform of the convolution between image and kernel (linear filtering) is the product between their Fourier transforms.

Multiscale Image Filtering

- Recall that the Fourier transform of the convolution between image and kernel (linear filtering) is the product between their Fourier transforms.

- One can use a filter bank with Gabor wavelets, in multiple scales and orientations, as kernels for linear filtering (feature extraction).

Multiscale Image Filtering

- Recall that the Fourier transform of the convolution between image and kernel (linear filtering) is the product between their Fourier transforms.

- One can use a filter bank with Gabor wavelets, in multiple scales and orientations, as kernels for linear filtering (feature extraction).

- This strategy creates a multiband image as representation and has been used as first convolutional layer in deep learning.

Multiscale Image Filtering

- Recall that the Fourier transform of the convolution between image and kernel (linear filtering) is the product between their Fourier transforms.
- One can use a filter bank with Gabor wavelets, in multiple scales and orientations, as kernels for linear filtering (feature extraction).
- This strategy creates a multiband image as representation and has been used as first convolutional layer in deep learning.
- Likewise, kernel banks can be composed by other basis functions (e.g., the first wavelets by Alfred Haar, 1909).