Shape-based Image Representation (Part I)

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Shape-based image representation

- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.

Shape-based image representation

- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.
- In 2D, each contour is a closed, connected, and oriented curve (a Jordan curve).

Shape-based image representation

- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.
- In 2D, each contour is a closed, connected, and oriented curve (a Jordan curve).
  - Closed curve because it separates the interior from the exterior of the object.

Shape-based image representation

- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.
- In 2D, each contour is a closed, connected, and oriented curve (a Jordan curve).
  - Closed curve because it separates the interior from the exterior of the object.
  - Connected curve because it can be represented by a sequence (path) of 8-adjacent pixels (arcs).

Shape-based image representation

- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.

- In 2D, each contour is a closed, connected, and oriented curve (a Jordan curve).

  - Closed curve because it separates the interior from the exterior of the object.

  - Connected curve because it can be represented by a sequence (path) of 8-adjacent pixels (arcs).

  - Oriented curve because when it is followed in clockwise orientation, its interior stays on the right side of the curve.

Shape-based image representation

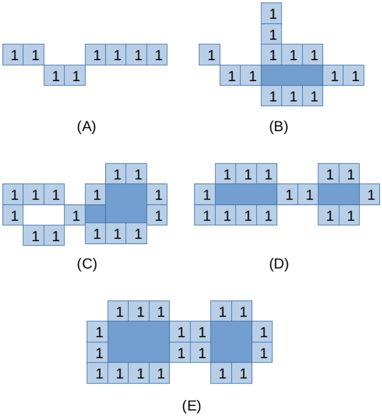
- An image may be represented by the internal and external contours (boundaries) of its objects, being an external contour often referred to as an object's shape.
- In 2D, each contour is a closed, connected, and oriented curve (a Jordan curve).
  - Closed curve because it separates the interior from the exterior of the object.
  - Connected curve because it can be represented by a sequence (path) of 8-adjacent pixels (arcs).
  - Oriented curve because when it is followed in clockwise orientation, its interior stays on the right side of the curve.
- Each contour encodes important shape properties for object description.

Agenda

- Challenges in contour extraction and labeling.
- Algorithm for contour extraction and labeling.
- Geodesic length of a contour.

Challenges in contour extraction and labeling

Noise in binary images 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 $\hat{I} = (D_I, I)$ can reduce some of these problems.

Challenges for contour pixel labeling

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

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

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

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

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

Challenges for contour pixel labeling

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

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

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

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

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

- 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 $\hat{I} = (D_I, I)$, the algorithm must be constrained to the border set $\mathcal{B}$.

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

Contour pixel labeling

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

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

- 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.

Contour pixel labeling

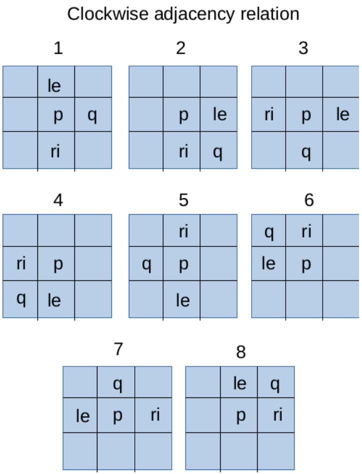
- For a given binary image $\hat{I} = (D_I, I)$, the algorithm must be constrained to the border set $\mathcal{B}$.

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

- 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.

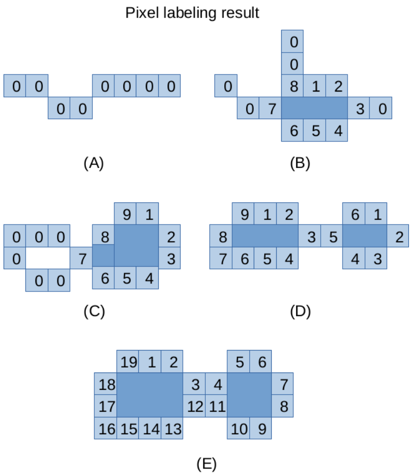
- It must also avoid arcs $\langle p, q \rangle$ which pass through the object/background and take into account the anti-clockwise orientation (i.e., $I(le(p,q)) = 1$ and $I(ri(p,q)) = 0$).

It starts from pixels that define valid arcs and visits the border pixels in depth search using clockwise adjacency relation $\mathcal{A}_{\sqrt{2}}$.



Contour pixel labeling

It starts from pixels that 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 $p \in D_I$, 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 $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid \langle p, q \rangle$ is a valid arc and q could also be a starting pixel.

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 $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid \langle p, q \rangle$ is a valid arc and q could also be a starting pixel.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, $I(le(p, q)) = 1$, and $I(ri(p, q)) = 0$.

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 $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid \langle p, q \rangle$ is a valid arc and q could also be a starting pixel.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, $I(le(p, q)) = 1$, and $I(ri(p, q)) = 0$.

- Let's indicate these events by $Valid(p)$ and $Valid(p, q)$.

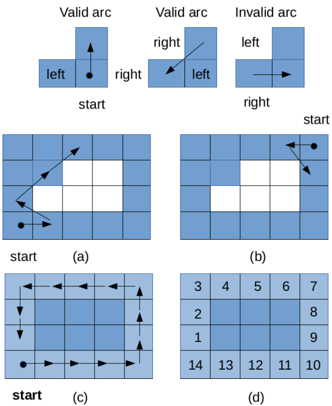
- Two conditions are used for a border pixel p and an arc $\langle p, q \rangle$ be considered a valid starting pixel and a valid arc, respectively.

  - Valid starting pixel p: $p \in \mathcal{B}$, $P(p) = nil$, and $\exists q \in \mathcal{A}_{\sqrt{2}}(p) \mid \langle p, q \rangle$ is a valid arc and q could also be a starting pixel.
  - Valid arc $\langle p, q \rangle$: $q \in \mathcal{A}_{\sqrt{2}}(p)$, $q \in \mathcal{B}$, $P(q) = nil$, $I(le(p, q)) = 1$, and $I(ri(p, q)) = 0$.

- Let's indicate these events by $Valid(p)$ and $Valid(p, q)$.

- Such rules cannot prevent the algorithm to start labeling silly contours, such as $(x, y) \rightarrow (x + 1, y) \rightarrow (x, y - 1)$, $(x, y) \rightarrow (x, y - 1) \rightarrow (x - 1, y)$, etc.

Contour pixel labeling



Silly contour with a last invalid arc at the top. Labels should not be assigned when (a) the path starts but never finishes and (b) it finishes as a silly contour. (c)-(d) A valid contour and its pixel labels.

- Therefore, once a pixel p has the starting pixel q as its neighbor and $P(p) \neq q$, the contour is about to close.

Contour pixel labeling

- Therefore, once a pixel p has the starting pixel q as its neighbor and $P(p) \neq q$, the contour is about to close.

- At this moment, if it is not a silly contour, we can start the labeling process.

Contour pixel labeling

- Therefore, once a pixel p has the starting pixel q as its neighbor and $P(p) \neq q$, the contour is about to close.
- At this moment, if it is not a silly contour, we can start the labeling process.
- A silly contour is detected when $P(P(p))$ is the starting pixel.

- Therefore, once a pixel p has the starting pixel q as its neighbor and $P(p) \neq q$, the contour is about to close.
- At this moment, if it is not a silly contour, we can start the labeling process.
- A silly contour is detected when $P(P(p))$ is the starting pixel.
- Therefore, the algorithm visits pixels in $\mathcal{B}$ in anti-clockwise until the starting pixel is reached again. If the path is not a silly contour, its pixels are labeled clockwise using P.

Contour pixel labeling

```
03. For each s ∈ D_I | Valid(s), do.
04.    Set P(s) ← s and insert s in 𝒬.
05.    While 𝒬 ≠ ∅, do.
06.      Remove p from 𝒬.
07.      For each q ∈ 𝒜_√2(p), do.
08.         If q = s and P(p) ≠ s, then
09.            If P(P(p)) ≠ s, then go to 13, else go to 17.
10.         If Valid(p, q), then.
11.           Set P(q) ← p.
12.           Insert q in 𝒬.
13.    Set i ← 1.
14.    While P(p) ≠ p, do.
15.      Set L_px(p) ← i, p ← P(p) and i ← i + 1.
16.    Set L_px(p) ← i.
17.    Set 𝒬 ← ∅.
```

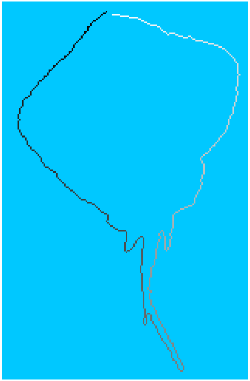
Contour pixel labeling

03. For each $s \in D_I \mid Valid(s)$, 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 $q = s$ and $P(p) \neq s$, then
09. If $P(P(p)) \neq s$, then go to 13, else go to 17.
10. If $Valid(p, q)$, then.
11. Set $P(q) \leftarrow p$.
12. Insert q in $\mathcal{Q}$.
13. Set $i \leftarrow 1$.
14. While $P(p) \neq p$, do.
15. Set $L_{px}(p) \leftarrow i$, $p \leftarrow P(p)$ and $i \leftarrow i + 1$.
16. Set $L_{px}(p) \leftarrow i$.
17. Set $\mathcal{Q} \leftarrow \emptyset$.

One can easily change it to assign a distinct contour label.

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.



Geodesic length of a contour

- Let (p_n, p_1) be a valid arc in a border set $\mathcal{B}$ and $\Pi_{\mathcal{B}}$ be the set of all possible paths in a graph $(\mathcal{B}, \mathcal{A}_{\sqrt{2}})$.

Geodesic length of a contour

- Let (p_n, p_1) be a valid arc in a border set $\mathcal{B}$ and $\Pi_{\mathcal{B}}$ be the set of all possible paths in a graph $(\mathcal{B}, \mathcal{A}_{\sqrt{2}})$.
- Let $\pi^*_{p_1 \rightsquigarrow p_n} = \langle p_1, p_2, \ldots, p_n \rangle$ be the shortest path from p_1 to p_n in $\Pi_{\mathcal{B}}$, excluding arc (p_1, p_n).

Geodesic length of a contour

- Let (p_n, p_1) be a valid arc in a border set $\mathcal{B}$ and $\Pi_{\mathcal{B}}$ be the set of all possible paths in a graph $(\mathcal{B}, \mathcal{A}_{\sqrt{2}})$.
- Let $\pi^*_{p_1 \rightsquigarrow p_n} = \langle p_1, p_2, \ldots, p_n \rangle$ be the shortest path from p_1 to p_n in $\Pi_{\mathcal{B}}$, excluding arc (p_1, p_n).
- The length $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_n})$ of $\pi^*_{p_1 \rightsquigarrow p_n}$ is called geodesic [1].

Geodesic length of a contour

- Let (p_n, p_1) be a valid arc in a border set $\mathcal{B}$ and $\Pi_{\mathcal{B}}$ be the set of all possible paths in a graph $(\mathcal{B}, \mathcal{A}_{\sqrt{2}})$.
- Let $\pi^*_{p_1 \rightsquigarrow p_n} = \langle p_1, p_2, \ldots, p_n \rangle$ be the shortest path from p_1 to p_n in $\Pi_{\mathcal{B}}$, excluding arc (p_1, p_n).
- The length $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_n})$ of $\pi^*_{p_1 \rightsquigarrow p_n}$ is called geodesic [1].
- The geodesic length of a contour in $\mathcal{B}$ is defined as $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_n}) + w(p_n, p_1)$, where

$$
\begin{aligned}
f_{geo}(\pi^*_{p_1 \rightsquigarrow p_n}) &= \min_{\pi_{p_1 \rightsquigarrow p_n} \in \Pi_{\mathcal{B}} \backslash \langle p_1, p_n \rangle} \left\{ \sum_{k=1}^{n-1} w(p_k, p_{k+1}) \right\}, \\
w(p_k, p_{k+1}) &= \begin{cases} 0.9016 & \text{if } \|p_{k+1} - p_k\| = 1, \\ 1.2890 & \text{if } \|p_{k+1} - p_k\| = \sqrt{2}. \end{cases}
\end{aligned}
$$

Geodesic length of a contour

- The shorest path formulation can label each pixel p_k, $k \in [1, n]$, of a contour $\pi^*_{p_1 \rightsquigarrow p_n} \cdot \langle p_n, p_1 \rangle$ by the geodesic length $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_k})$.

Geodesic length of a contour

- The shorest path formulation can label each pixel p_k, $k \in [1, n]$, of a contour $\pi^*_{p_1 \rightsquigarrow p_n} \cdot \langle p_n, p_1 \rangle$ by the geodesic length $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_k})$.
- We will see how useful is this formulation to obtain smooth multiscale skeletons from the shape.

Geodesic length of a contour

- The shorest path formulation can label each pixel p_k, $k \in [1, n]$, of a contour $\pi^*_{p_1 \rightsquigarrow p_n} \cdot \langle p_n, p_1 \rangle$ by the geodesic length $f_{geo}(\pi^*_{p_1 \rightsquigarrow p_k})$.
- We will see how useful is this formulation to obtain smooth multiscale skeletons from the shape.
- As exercise, elaborate an IFT-based algorithm to extract and label all contours in a binary image by the geodesic length assigned to each pixel with respect to an arbitrary pixel p_1 and a valid arc $(p_n, p_1) \in \mathcal{A}_{\sqrt{2}}$.

[1] A.X. Falcão, C. Feng, J. Kustra, and A.C. Telea.
Chapter 2 - multiscale 2d medial axes and 3d surface skeletons by the image foresting transform.
In P.K. Saha, G. Borgefors, and G.S. di Baja, editors, *Skeletonization*, pages 43 – 70. Academic Press, 2017.