

Operações morfológicas, descritores de forma

PROF. CESAR HENRIQUE COMIN

Transformada Hit-or-miss

- A transformada hit-or-miss é utilizada para encontrarmos pixels na imagem cuja vizinhança segue um padrão especificado
- Ela é definida como

$$A \star B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

onde B_1 e B_2 são dois elementos estruturantes

Transformada Hit-or-miss

Suponha que queremos encontrar o seguinte padrão em uma imagem A :

onde x significa “o valor nesse pixel não importa”

x	1	x
1	0	1
x	1	x

Para fazer isso, primeiro definimos os seguintes elementos estruturantes:

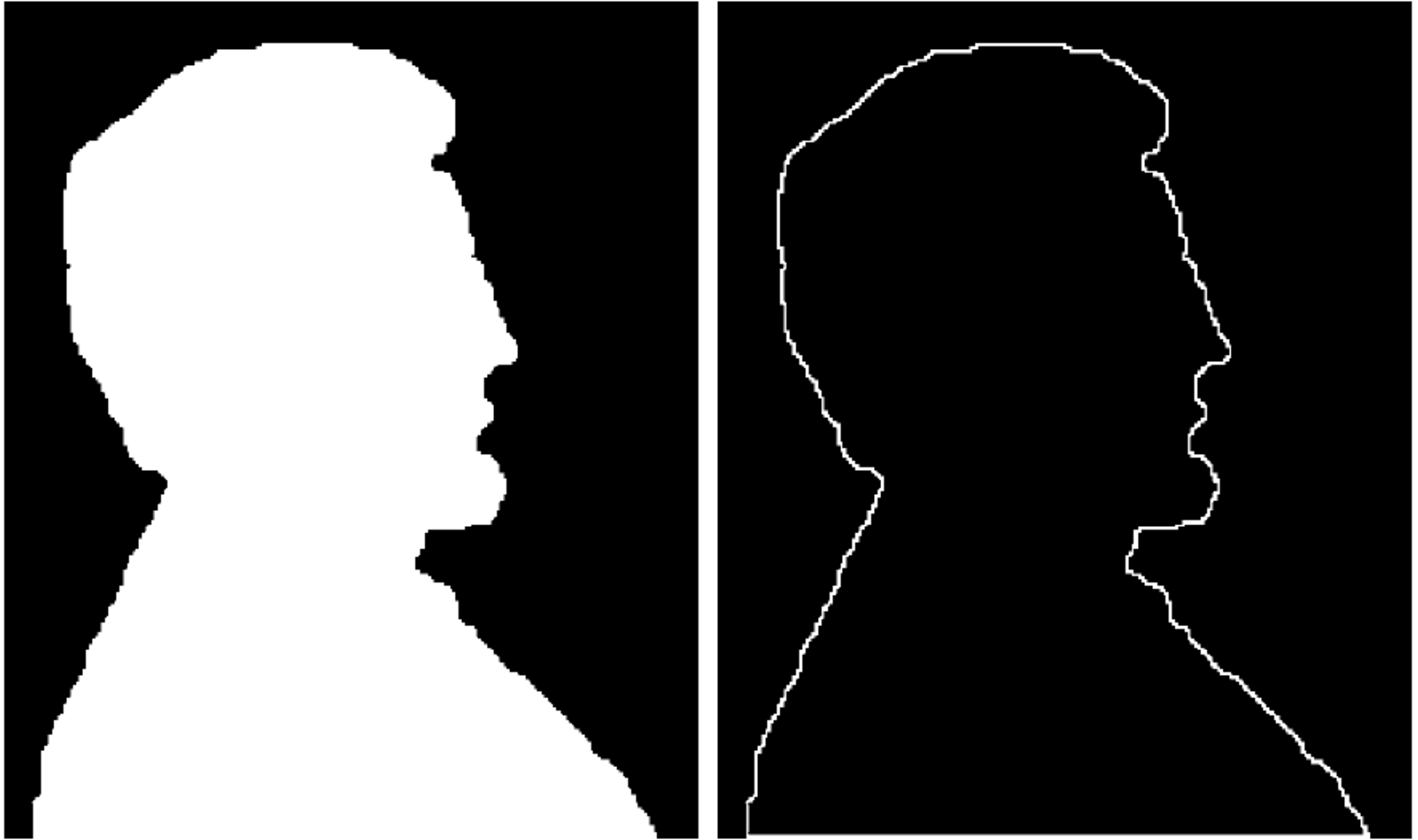
$$B_1 = \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \quad \begin{array}{l} 1 \text{ indica} \\ \text{pixels que} \\ \text{devem ser 1} \\ \text{na imagem} \end{array}$$

$$B_2 = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad \begin{array}{l} 1 \text{ indica} \\ \text{pixels que} \\ \text{devem ser 0} \\ \text{na imagem} \end{array}$$

Em seguida, aplicamos a transformada hit-or-miss:

$$A \star B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

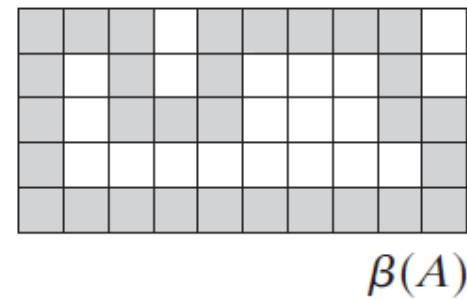
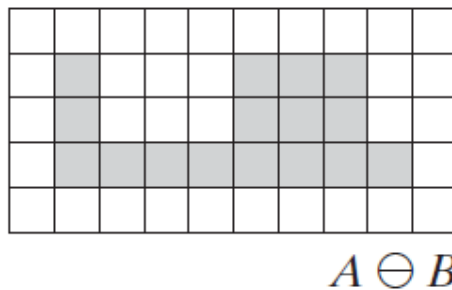
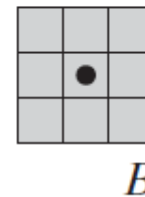
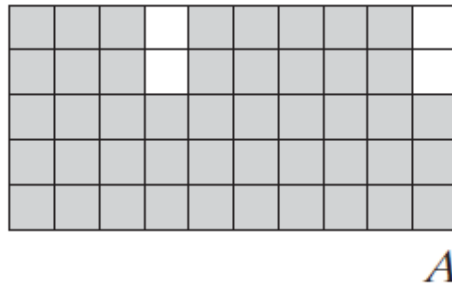
Extração de borda



Extração de borda

Basta calcularmos

$$\beta(A) = A - A \ominus B$$



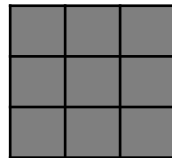
Extração de componentes conexos

- Uma operação comum em imagens binárias é a identificação dos componentes conexos na imagem
- Um componente conexo é formado por um conjunto de pixels conectados na imagem. Dois pixels i e j estão conectados se for possível “partir” do pixel i e “chegar” no pixel j passando apenas por pixels de mesma cor que i e j
- A conectividade entre pixels depende do tipo de vizinhança utilizada

Vizinhança-4

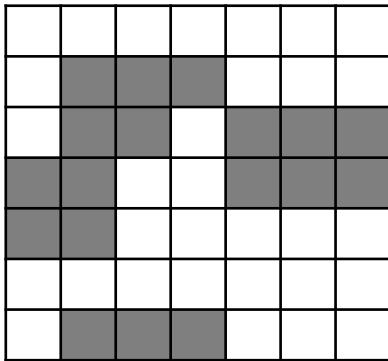


Vizinhança-8



Extração de componentes conexos

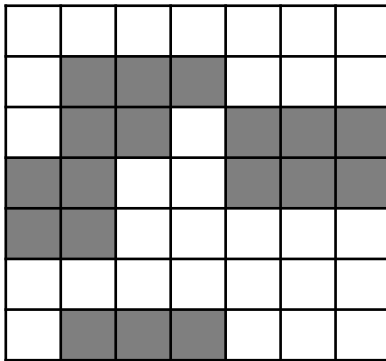
- Uma operação comum em imagens binárias é a identificação dos componentes conexos na imagem
- Um componente conexo é formado por um conjunto de pixels conectados na imagem. Dois pixels i e j estão conectados se for possível “partir” do pixel i e “chegar” no pixel j passando apenas por pixels de mesma cor que i e j
- A conectividade entre pixels depende do tipo de vizinhança utilizada



Quantos componentes para vizinhança-4 e vizinhança-8?

Extração de componentes conexos

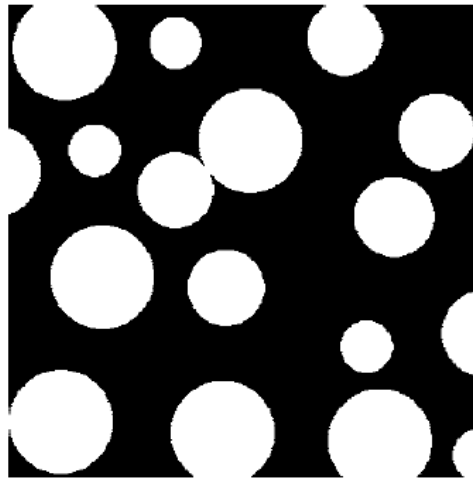
- Uma operação comum em imagens binárias é a identificação dos componentes conexos na imagem
- Um componente conexo é formado por um conjunto de pixels conectados na imagem. Dois pixels i e j estão conectados se for possível “partir” do pixel i e “chegar” no pixel j passando apenas por pixels de mesma cor que i e j
- A conectividade entre pixels depende do tipo de vizinhança utilizada



Temos 3 componentes se for utilizado vizinhança-4 e 2 no caso de vizinhança-8

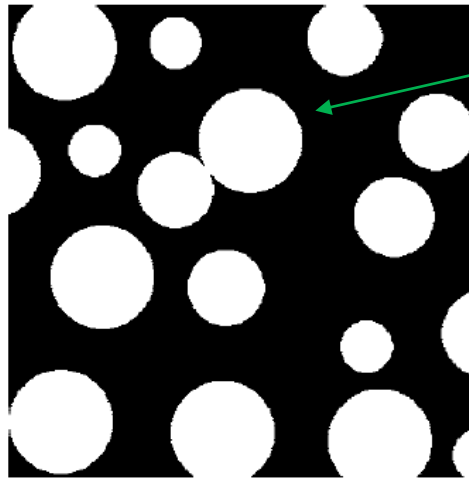
Extração de componentes conexos

- Tendo identificado os componentes, eles podem ser caracterizados de forma individual



Extração de componentes conexos

- Tendo identificado os componentes, eles podem ser caracterizados de forma individual



Cuidado!
Precisamos de uma
operação de
abertura aqui

Extração de componentes conexos

Algoritmo para identificação dos componentes conexos:

- Percorra cada linha da imagem, da esquerda para a direita;
- Para cada pixel com valor 1, verifique os valores dos pixels à esquerda e ao norte;
- Chame o valor do pixel à esquerda de p_e e o valor do pixel ao norte de p_n ;

	p_n
p_e	1

Extração de componentes conexos

Algoritmo para identificação dos componentes conexos:

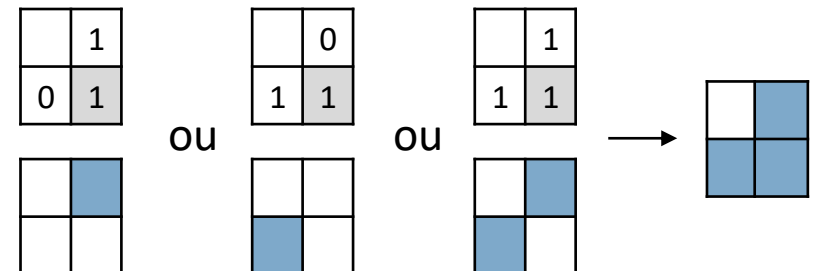
- Percorra cada linha da imagem, da esquerda para a direita;
- Para cada pixel com valor 1, verifique os valores dos pixels à esquerda e ao norte;
- Chame o valor do pixel à esquerda de p_e e o valor do pixel ao norte de p_n ;
- Se $p_e = 0$ e $p_n = 0$, associe um novo rótulo ao pixel atual;

	0
0	1

Extração de componentes conexos

Algoritmo para identificação dos componentes conexos:

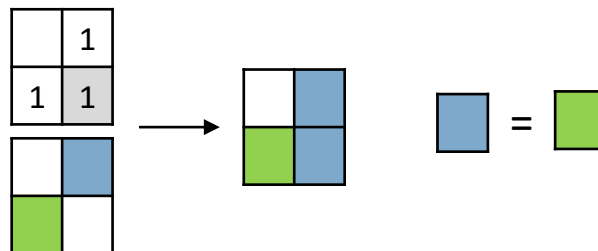
- Percorra cada linha da imagem, da esquerda para a direita;
- Para cada pixel com valor 1, verifique os valores dos pixels à esquerda e ao norte;
- Chame o valor do pixel à esquerda de p_e e o valor do pixel ao norte de p_n ;
- Se $p_e = 0$ e $p_n = 0$, associe um novo rótulo ao pixel atual;
- Se ($p_e = 1$ e $p_n = 0$) ou ($p_e = 0$ e $p_n = 1$) ou ($p_e = 1$ e $p_n = 1$ e ambos os pixels possuem o mesmo rótulo), associe ao pixel atual o mesmo rótulo do pixel com valor 1



Extração de componentes conexos

Algoritmo para identificação dos componentes conexos:

- Percorra cada linha da imagem, da esquerda para a direita;
- Para cada pixel com valor 1, verifique os valores dos pixels à esquerda e ao norte;
- Chame o valor do pixel à esquerda de p_e e o valor do pixel ao norte de p_n ;
- Se $p_e = 0$ e $p_n = 0$, associe um novo rótulo ao pixel atual;
- Se $(p_e = 1 \text{ e } p_n = 0)$ ou $(p_e = 0 \text{ e } p_n = 1)$ ou $(p_e = 1 \text{ e } p_n = 1 \text{ e ambos os pixels possuem o mesmo rótulo})$, associe ao pixel atual o mesmo rótulo do pixel com valor 1
- Se $p_e = 1$ e $p_n = 1$ e os pixels possuem rótulos diferentes, associe ao pixel atual o menor rótulo dentre os dois e armazene a equivalência entre os dois rótulos



Extração de componentes conexos

Percorra novamente a imagem, substituindo cada rótulo pelo menor valor de rótulo equivalente

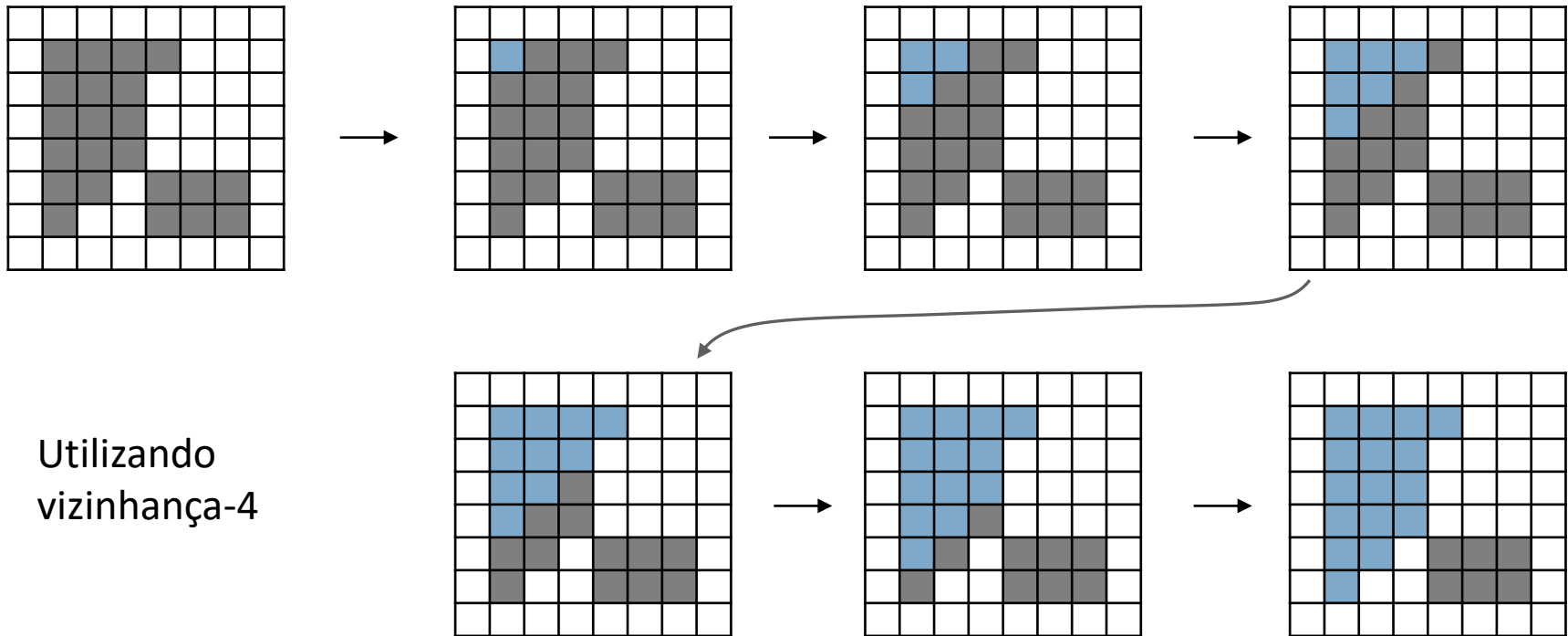
O resultado é uma imagem contendo, para cada pixel, o rótulo do respectivo componente.

Extração de componentes conexos

Notebook “**Componentes conexos**”

Extração de componentes conexos

Outra forma de extrairmos componentes é através do algoritmo *flood fill*, que consiste em fazer uma busca em largura na imagem.



Fecho convexo

- O menor polígono convexo contendo o objeto
- Podemos imaginar uma banda de borracha esticada ao redor do objeto



Transformada distância

- Amplamente utilizada em imagens binárias
- Para cada pixel do objeto, é calculada a distância entre ele e o pixel de borda mais próximo
- Diferentes tipos de distâncias podem ser utilizadas (Euclidiana, city-block, etc)

Transformada distância

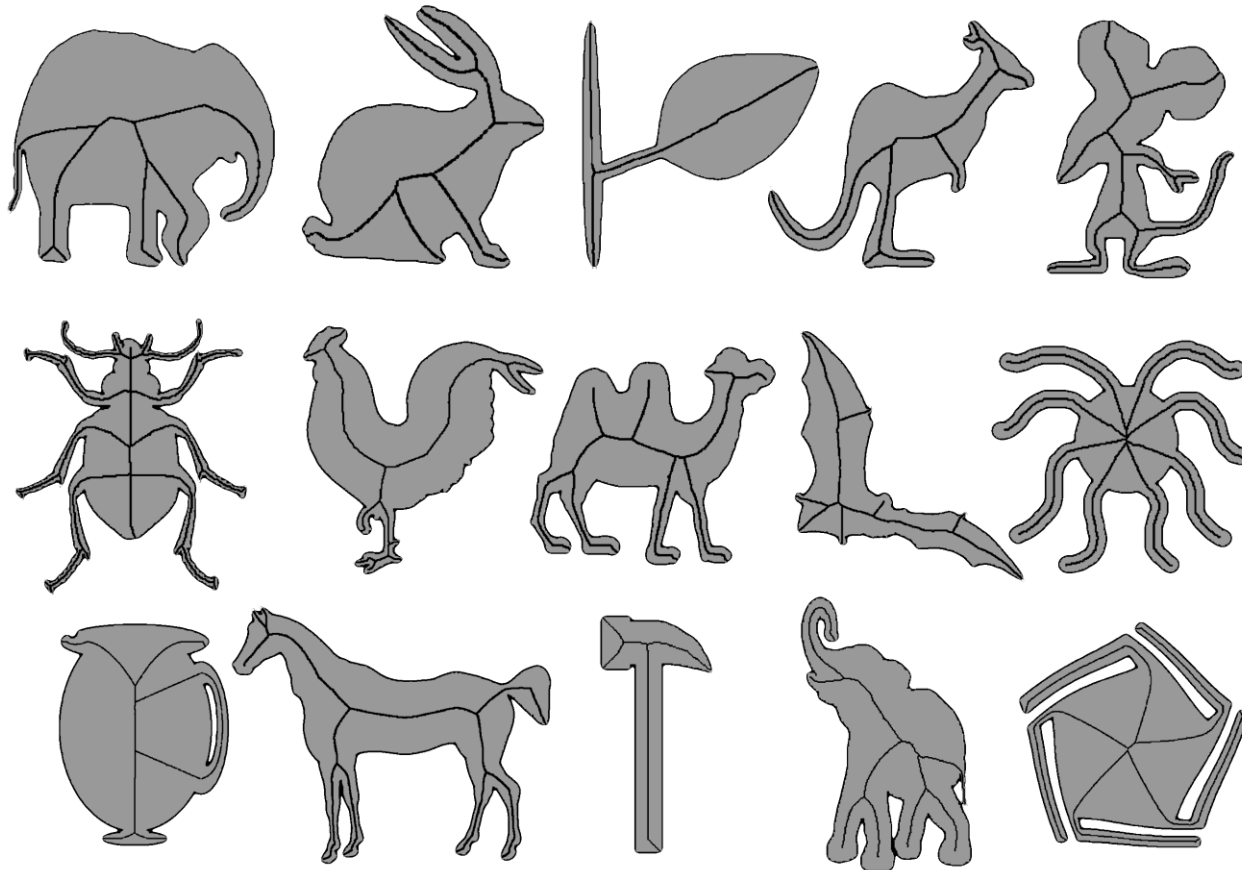


Esqueleto

- Esqueletonização é a transformação de uma forma binária em uma cadeia de pixels possuindo largura 1 e que representa corretamente a topologia da forma original.

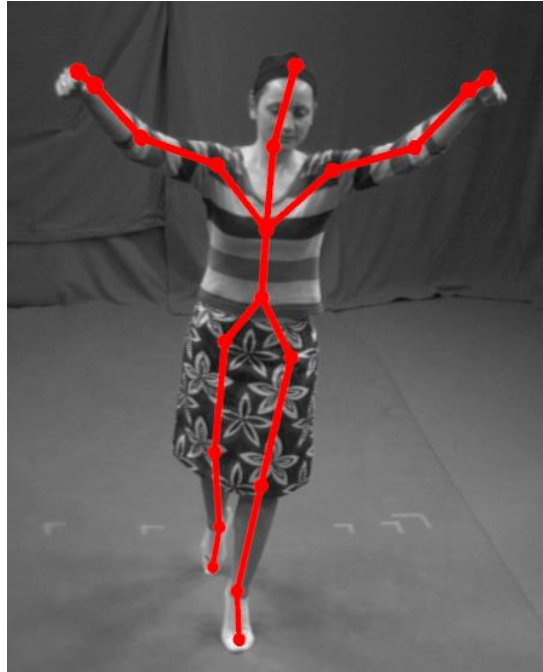
Esqueleto

- Essa técnica permite uma descrição simples e eficiente de formas



Esqueleto

- Essa técnica permite uma descrição simples e eficiente de formas



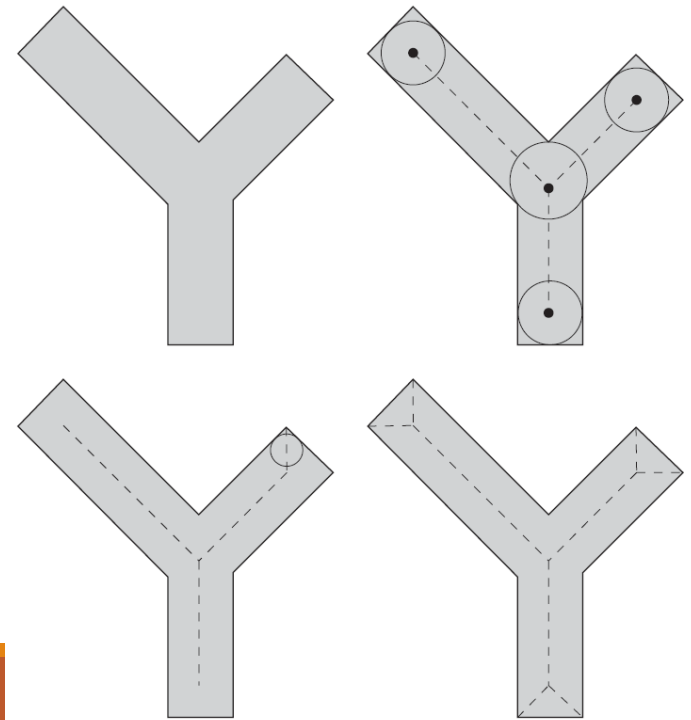
- Além de caracterizar o objeto, o esqueleto pode ser utilizado para outras técnicas de processamento de imagens como identificação da posição de pessoas

Esqueleto

Algoritmo intuitivo:

1. O centro de um disco é posicionado em um pixel pertencente ao objeto;
2. Se o disco estiver completamente no interior do objeto, e ele **tocar a borda do objeto em pelo menos 2 pontos**, o ponto central é adicionado ao conjunto S;
3. Os passos 1 e 2 são repetidos para todos os possíveis raios de disco e para todos os pixels do objeto.

O conjunto final S é o esqueleto do objeto



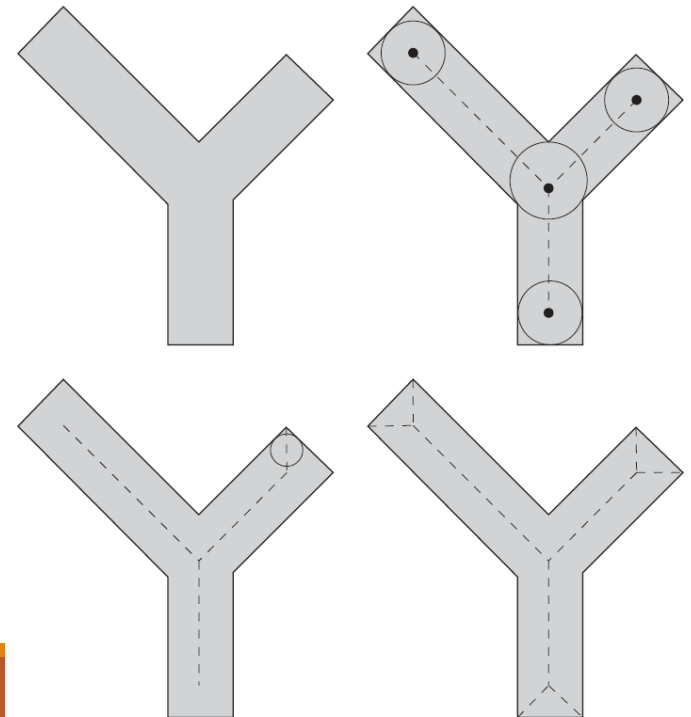
Esqueleto

Algoritmo intuitivo:

1. O centro de um disco é posicionado em um pixel pertencente ao objeto;
2. Se o disco estiver completamente no interior do objeto, e ele tocar a borda do objeto em pelo menos 2 pontos, o ponto central é adicionado ao conjunto S ;
3. Os passos 1 e 2 são repetidos para todos os possíveis raios de disco e para todos os pixels do objeto.

O conjunto final S é o esqueleto do objeto

Se armazenarmos o raio do disco associado a cada ponto de S podemos reconstruir o objeto a partir do esqueleto



Esqueleto

- Algoritmo alternativo para o cálculo do esqueleto:
 - Para cada ponto p do objeto, encontre o ponto de borda mais próximo de p .
Se existir mais de um ponto mais próximo, p pertence ao esqueleto
- Ambas as definições apresentadas são custosas de serem implementadas
- Uma forma alternativa de encontrar o esqueleto é utilizando operações morfológicas

Esqueleto

Algoritmo para calcular o esqueleto utilizando morfologia:

Considere que a vizinhança de um pixel p_1 está nomeada da seguinte maneira:

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Passo 1: Para cada pixel do objeto, remova o pixel (mude seu valor para 0) se todas essas condições forem satisfeitas:

- (a) $2 \leq N(p_1) \leq 6$
- (b) $T(p_1) = 1$
- (c) $p_2 p_4 p_6 = 0$
- (d) $p_4 p_6 p_8 = 0$

onde $N(p_1)$ é o número de vizinhos não nulos de p_1 e $T(p_1)$ é o número de transições 0-1 na sequência ordenada $(p_2, p_3, \dots, p_8, p_9, p_2)$.

Esqueleto

Passo 1: Para cada pixel do objeto, remova o pixel (mude seu valor para 0) se todas essas condições forem satisfeitas:

- (a) $2 \leq N(P_1) \leq 6$
- (b) $T(P_1) = 1$
- (c) $p_2 p_4 p_6 = 0$
- (d) $p_4 p_6 p_8 = 0$

Passo 2: Para cada pixel do objeto, remova o pixel (mude seu valor para 0) se todas essas condições forem satisfeitas:

- (a) Mesmo que no passo 1
- (b) Mesmo que no passo 1
- (c) $p_2 p_4 p_8 = 0$
- (d) $p_2 p_6 p_8 = 0$

Os passos 1 e 2 são repetidos até que não haja mais modificação do objeto (nenhum pixel segue os critérios de remoção)

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Esqueleto

Em cada passo, os pixels não são modificados até que todos os pixels tenham sido checados para remoção

Essencialmente, o algoritmo faz erosões sucessivas do objeto evitando

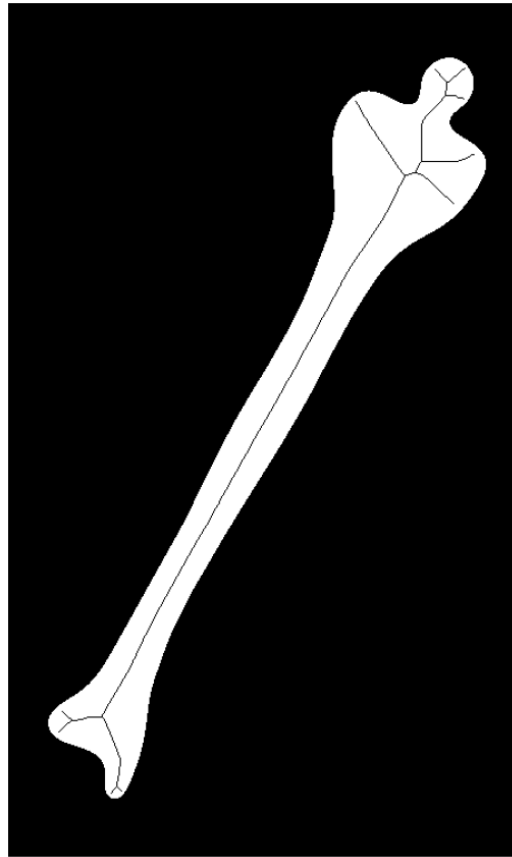
- A remoção de pontos de terminação (“pontas” do objeto)
- A quebra do objeto em componentes desconexos
- A remoção de pontos onde a largura do objeto é 1.

Por causa disso, esse algoritmo também é chamado de afinamento

Note que apenas pixels de borda precisam ser checados para remoção

Esqueleto

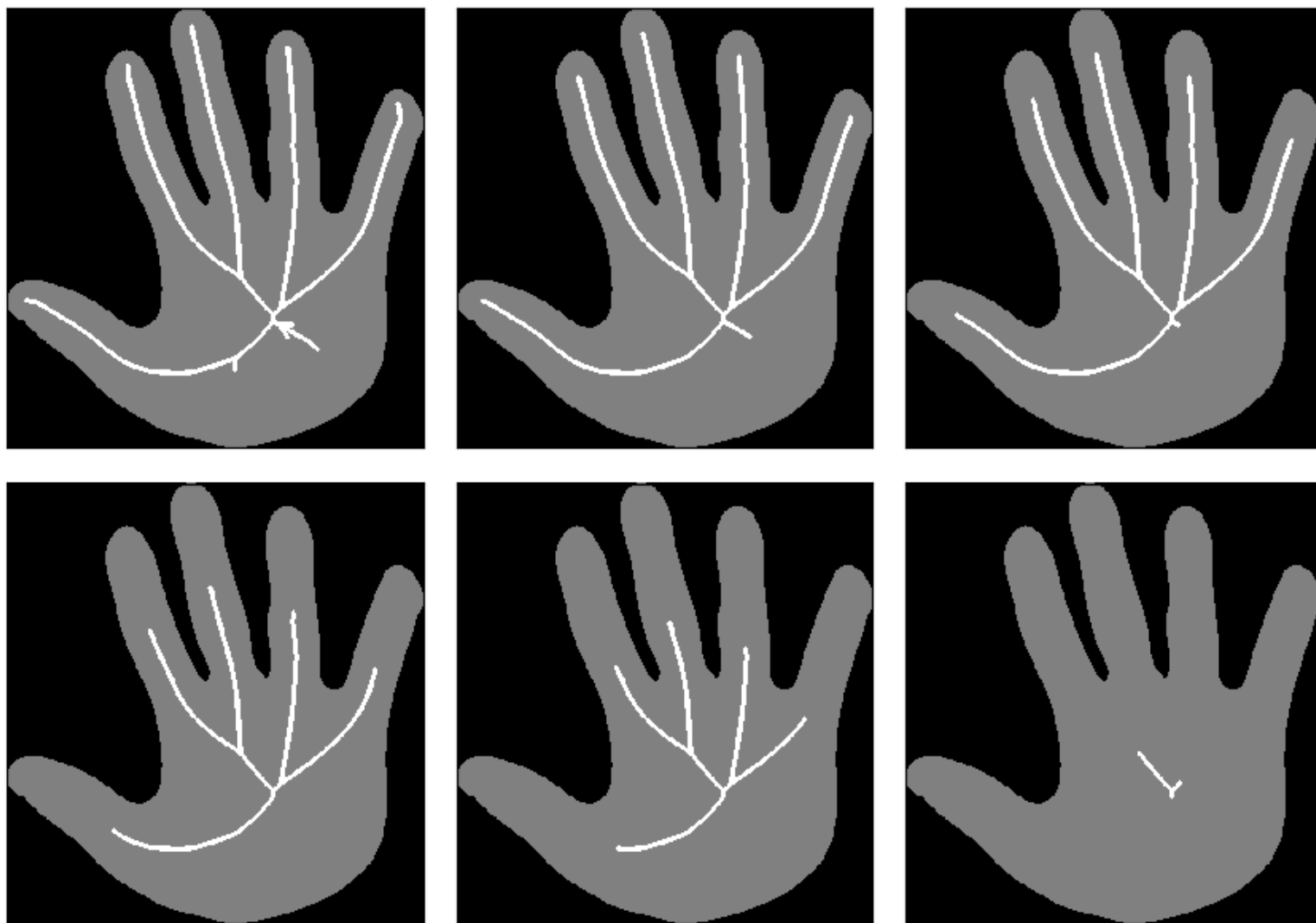
Infelizmente, o esqueleto é sensível a pequenas variações no contorno do objeto



Esqueleto

- Existem muitos algoritmos para o cálculo do esqueleto em escalas específicas. Nesse caso, detalhes menores do que a escala considerada não influenciam o esqueleto.
- Alternativamente, é possível suavizarmos o contorno do objeto antes de aplicarmos o processo de afinamento

Esqueleto multiescala



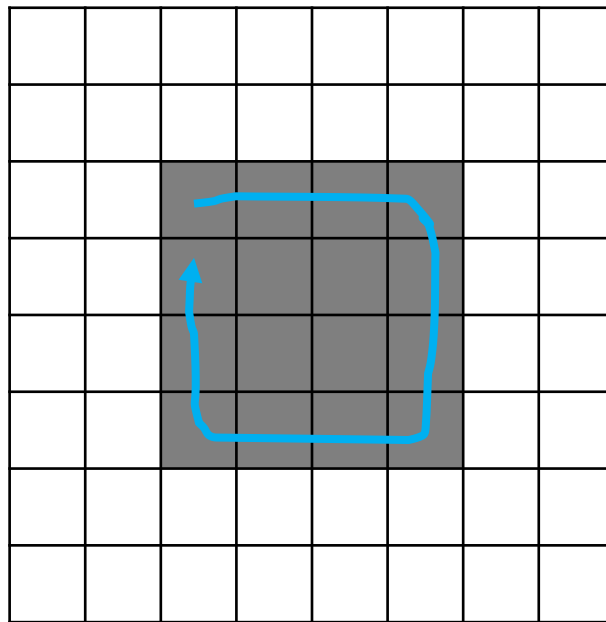
Representação de formas

Contorno paramétrico

O contorno paramétrico é uma sequência **ordenada** de pixels representando a borda de um objeto.

Contorno paramétrico

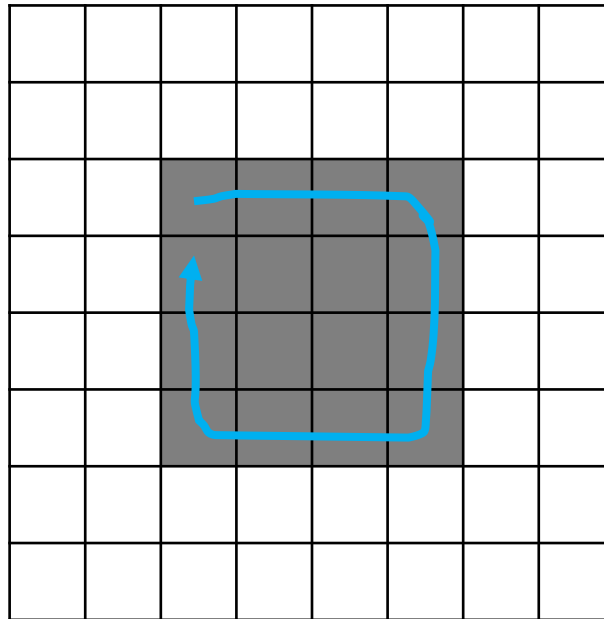
O contorno paramétrico é uma sequência **ordenada** de pixels representando a borda de um objeto.



Contorno paramétrico

No caso da imagem abaixo, um possível contorno paramétrico para o objeto é

$$C = [(2,2), (2,3), (2,4), (2,5), (3,5), (4,5), (5,5), (5,4), (5,3), (5,2), (4,2), (3,2)]$$



Contorno paramétrico

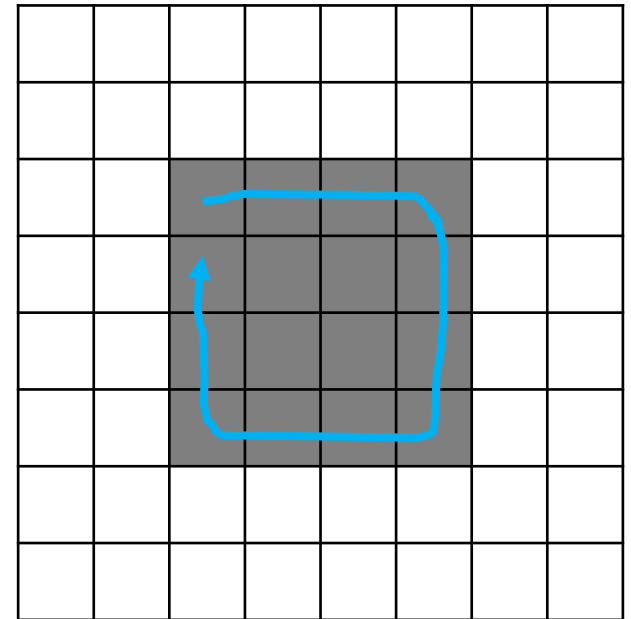
No caso da imagem abaixo, um possível contorno paramétrico para o objeto é

$$C = [(2,2), (2,3), (2,4), (2,5), (3,5), (4,5), (5,5), (5,4), (5,3), (5,2), (4,2), (3,2)]$$

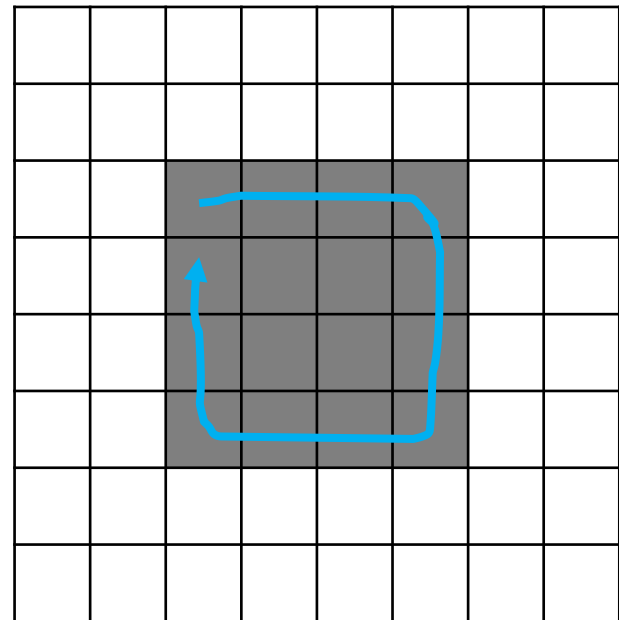
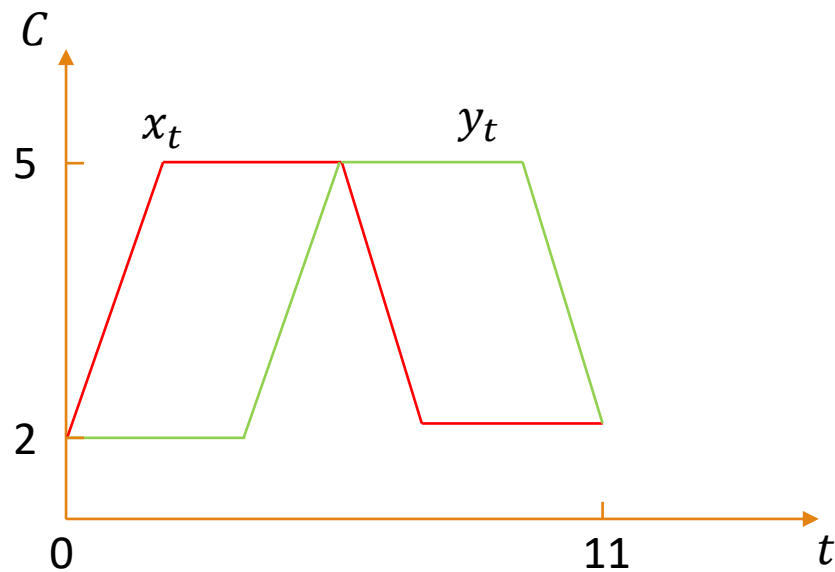
Podemos representar esse contorno na forma de dois sinais:

$$x = [2,3,4,5,5,5,5,4,3,2,2,2]$$

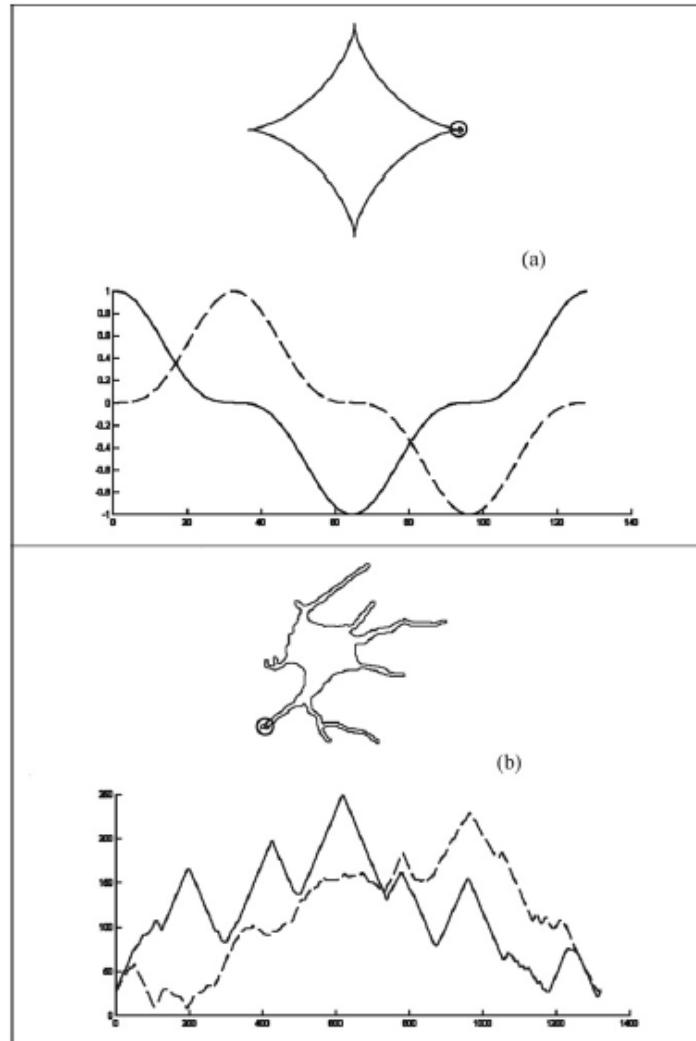
$$y = [2,2,2,2,3,4,5,5,5,5,4,3]$$



Contorno paramétrico



Exemplos de contornos paramétricos



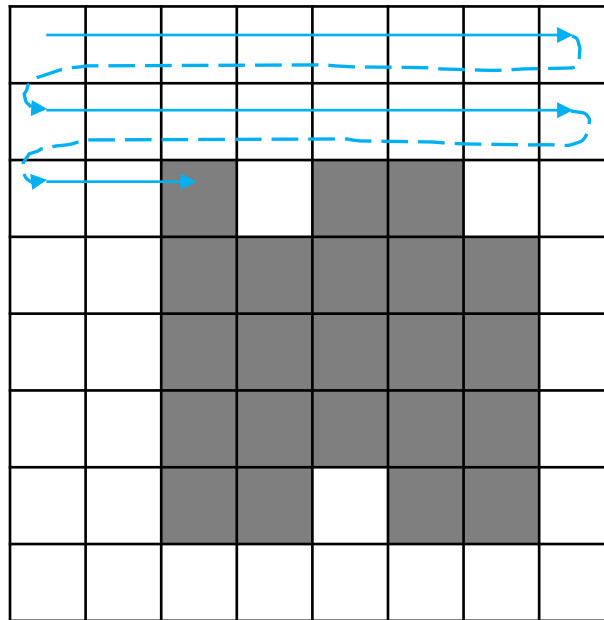
Contorno Paramétrico

- O contorno paramétrico é uma forma de **representação de imagens**
- A partir do contorno paramétrico, podemos reconstruir exatamente o objeto
- O contorno paramétrico pode ser utilizado na obtenção de diversas propriedades de imagens, e também para a aplicação de técnicas de processamento morfológico

Obtenção do contorno paramétrico

Algoritmo de traçado de contorno (Moore-Neighbor tracing algorithm)

Percorra cada linha da imagem de cima para baixo e da esquerda para a direita até encontrar o primeiro pixel do objeto



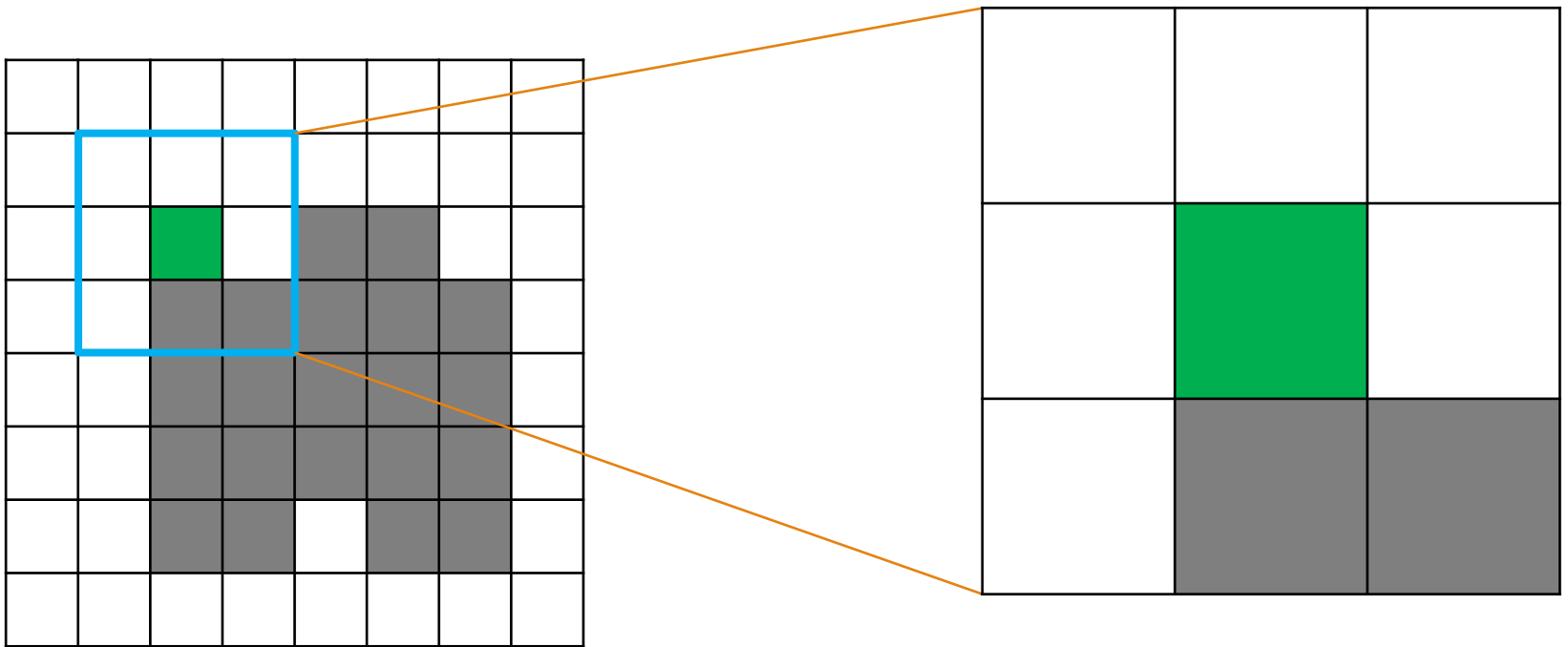
Obtenção do contorno paramétrico

Enumere a vizinhança de um dado pixel P de acordo com o seguinte padrão:

7	0	1
6	P	2
5	4	3

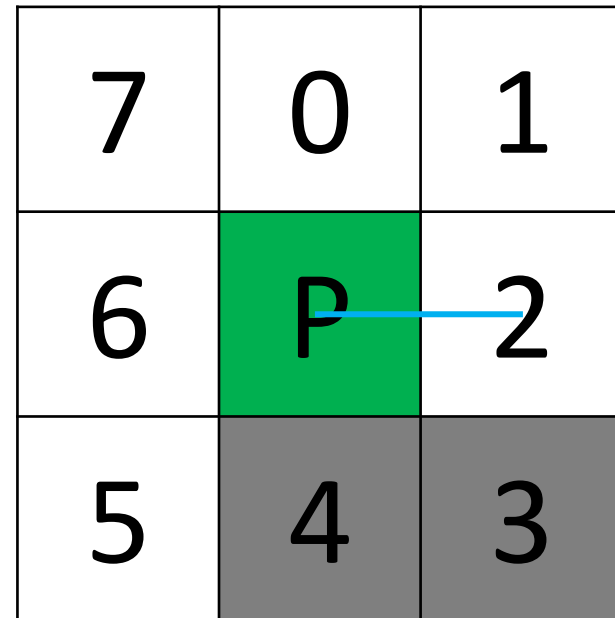
Obtenção do contorno paramétrico

Vamos olhar mais de perto a vizinhança do primeiro pixel encontrado



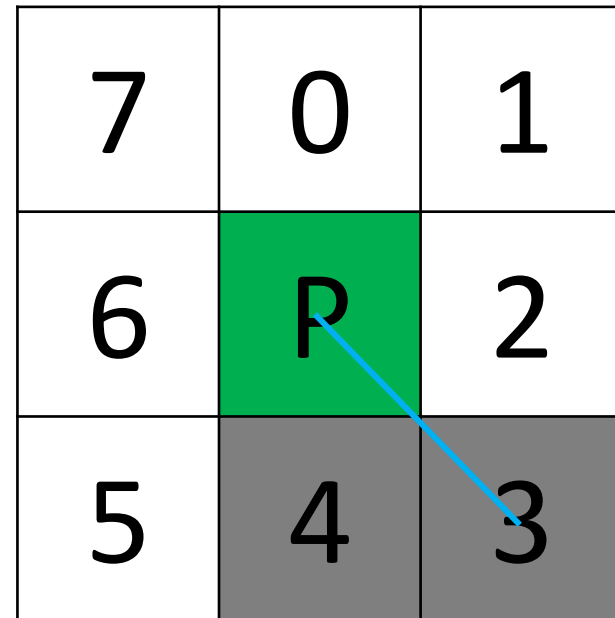
Obtenção do contorno paramétrico

- Já sabemos que os vizinhos 0, 1, 6 e 7 não podem ser do objeto, pois já percorremos esses pixels.
- Começando pelo pixel de índice 2, e seguindo no sentido horário, procuramos pelo próximo pixel pertencente ao objeto



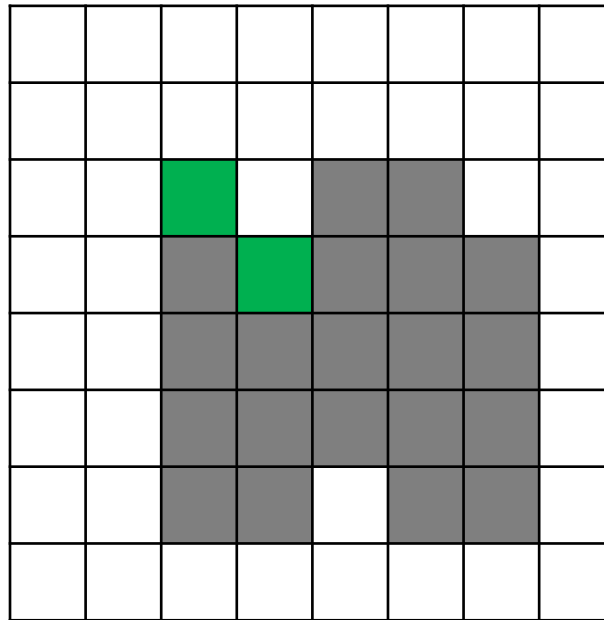
Obtenção do contorno paramétrico

- Já sabemos que os vizinhos 0, 1, 6 e 7 não podem ser do objeto, pois já percorremos esses pixels.
- Começando pelo pixel de índice 2, e seguindo no sentido horário, procuramos pelo próximo pixel pertencente ao objeto
- O pixel de índice 3 é encontrado



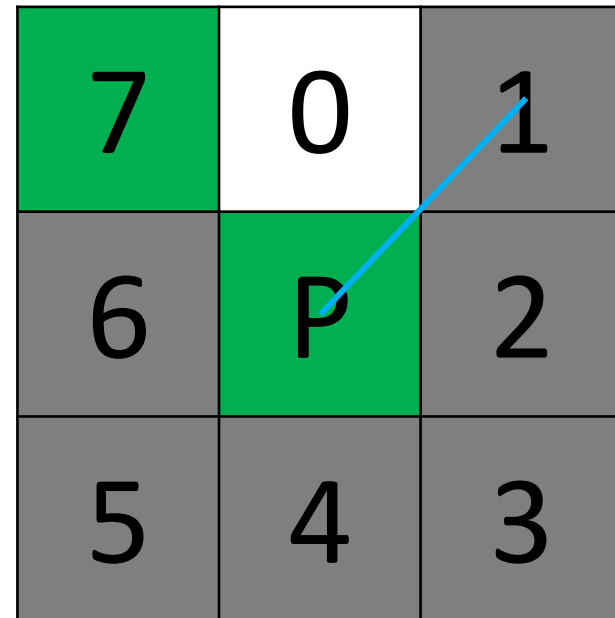
Obtenção do contorno paramétrico

Repetimos a busca na vizinhança do pixel recentemente encontrado



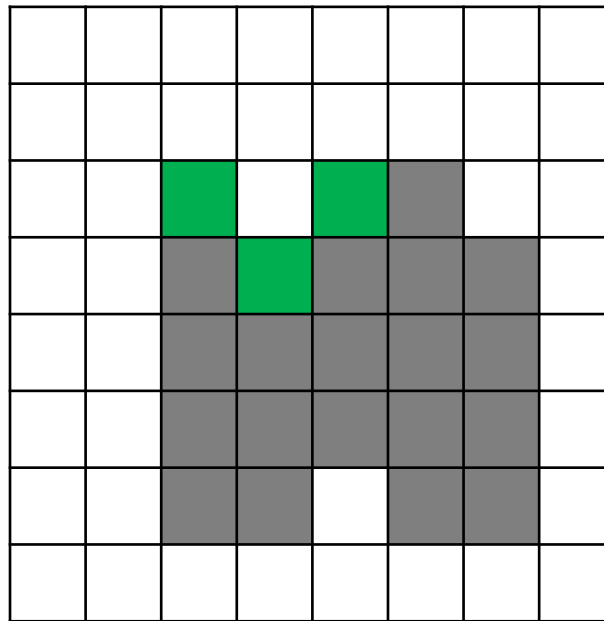
Obtenção do contorno paramétrico

- Importante! **O primeiro vizinho a ser verificado é o vizinho seguinte ao último pixel visitado anteriormente**
- No caso ao lado, o vizinho 0 já foi verificado na iteração anterior, então o próximo pixel a ser verificado é o pixel de índice 1



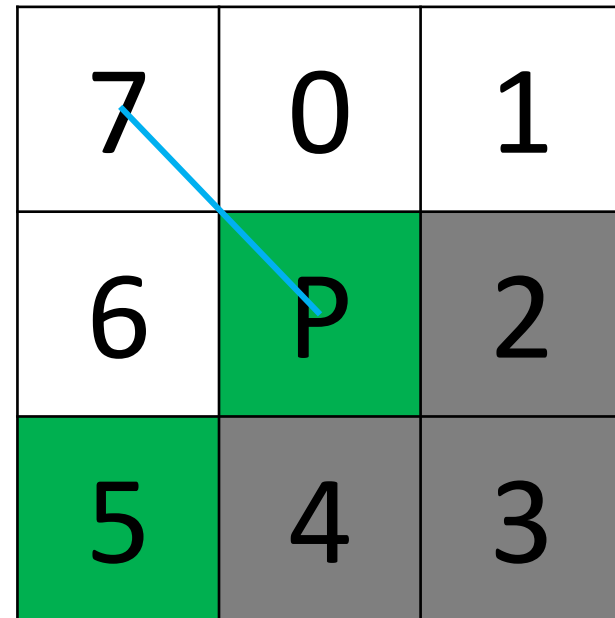
Obtenção do contorno paramétrico

Repetimos a busca na vizinhança do pixel recentemente encontrado



Obtenção do contorno paramétrico

- No caso ao lado, o vizinho 6 já foi verificado, então o próximo pixel a ser verificado é o pixel de índice 7

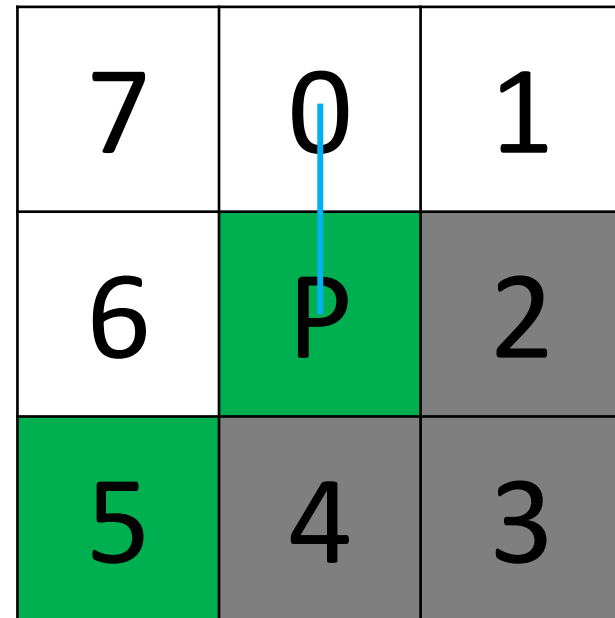


7	0	1
6	P	2
5	4	3

The diagram shows a 3x3 grid of pixels. The central pixel is labeled 'P' and is green. The pixels are indexed 0 through 7 in a row-major order: 0 is top-center, 1 is top-right, 2 is middle-right, 3 is bottom-right, 4 is bottom-center, 5 is bottom-left, 6 is middle-left, and 7 is top-left. A blue arrow points from pixel 6 to pixel 7. The pixels are colored: white for indices 0, 1, 2, 6, 7; green for indices 5 and P; and gray for indices 3, 4, and 2.

Obtenção do contorno paramétrico

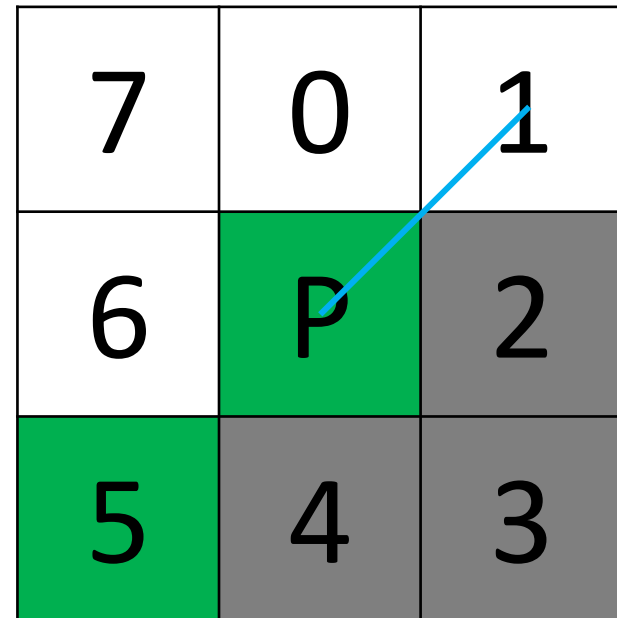
- No caso ao lado, o vizinho 6 já foi verificado, então o próximo pixel a ser verificado é o pixel de índice 7



7	0	1
6	P	2
5	4	3

Obtenção do contorno paramétrico

- No caso ao lado, o vizinho 6 já foi verificado, então o próximo pixel a ser verificado é o pixel de índice 7



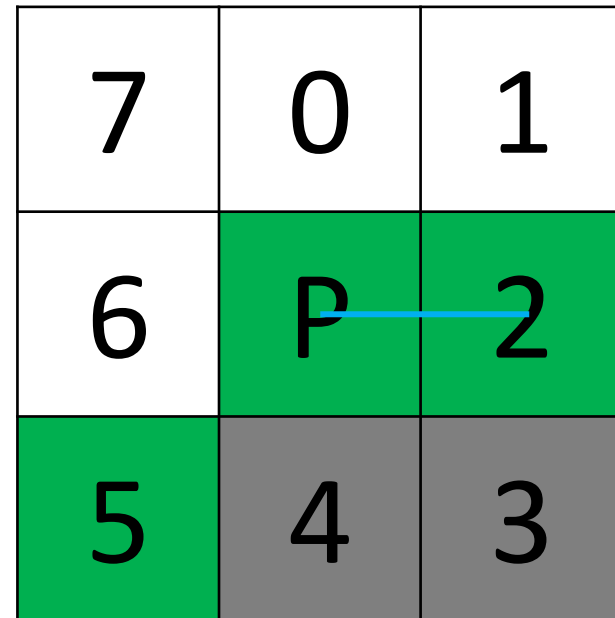
7	0	1
6	P	2
5	4	3

The diagram shows a 3x3 grid of pixels. The central pixel is labeled 'P' and is green. The pixels are indexed 0 through 7 in a row-major order: 0 is at (0,1), 1 at (0,2), 2 at (1,2), 3 at (2,2), 4 at (2,1), 5 at (2,0), 6 at (1,0), and 7 at (0,0). A blue line connects pixel 1 to pixel P. The pixels at (1,2), (2,2), (2,1), and (2,0) are shaded gray, while the others are white.

Obtenção do contorno paramétrico

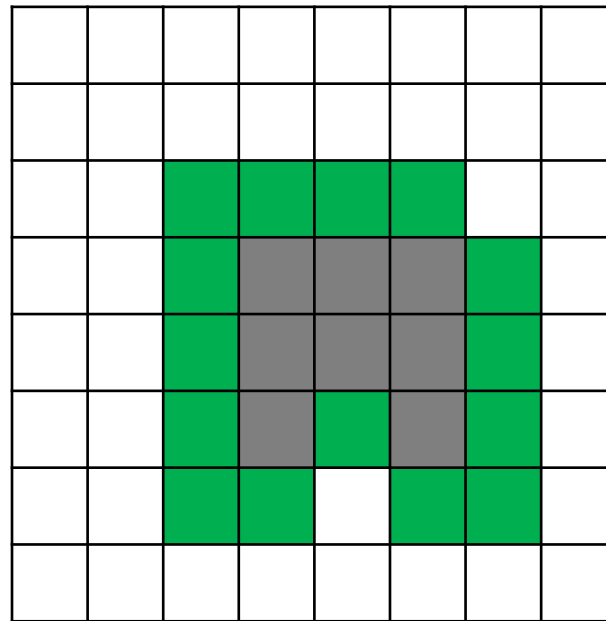
- No caso ao lado, o vizinho 6 já foi verificado, então o próximo pixel a ser verificado é o pixel de índice 7

7	0	1
6	P	2
5	4	3



Obtenção do contorno paramétrico

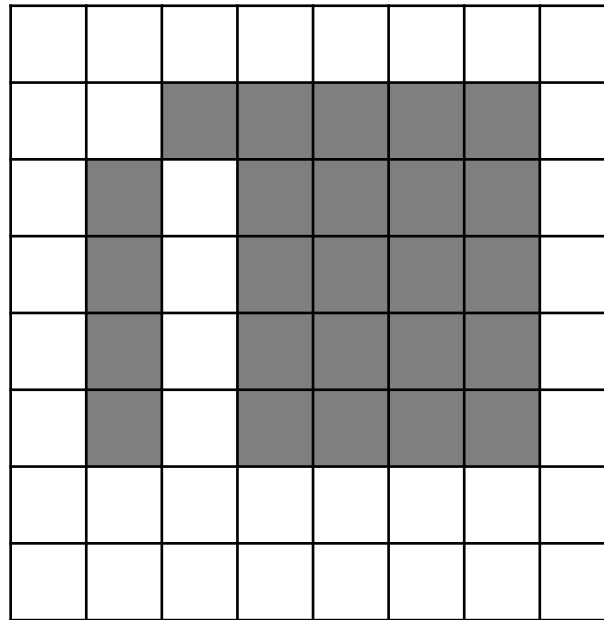
O processo é repetido até que o ponto inicial seja visitado novamente



Obtenção do contorno paramétrico

O critério de parada pode causar alguns problemas

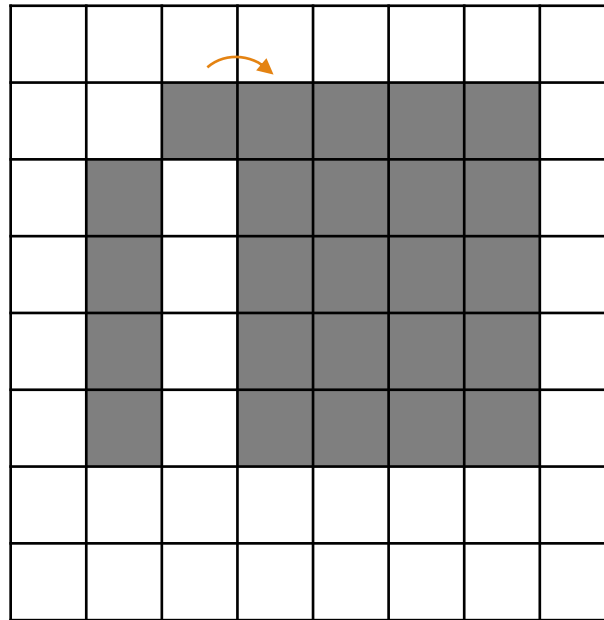
No caso abaixo, a região esquerda do objeto não será identificada



Obtenção do contorno paramétrico

Outro possível critério de parada:

Pare quando visitarmos novamente o segundo ponto de contorno a partir do primeiro ponto



Obtenção do contorno paramétrico

Detalhe do algoritmo:

- Dado o índice do vizinho que será o próximo ponto de contorno, sabemos qual o índice que precisamos começar a procurar quando estivermos na próxima iteração do algoritmo

		7	0	1
		6		2
		5	4	3

O vizinho de índice 5
é o próximo ponto
de contorno



	7	0	1	
	6		2	
	5	4	3	

Portanto, começamos
a procurar pelo vizinho
de índice 3, pois
sabemos que o 2 já foi
verificado

Obtenção do contorno paramétrico

Detalhe do algoritmo:

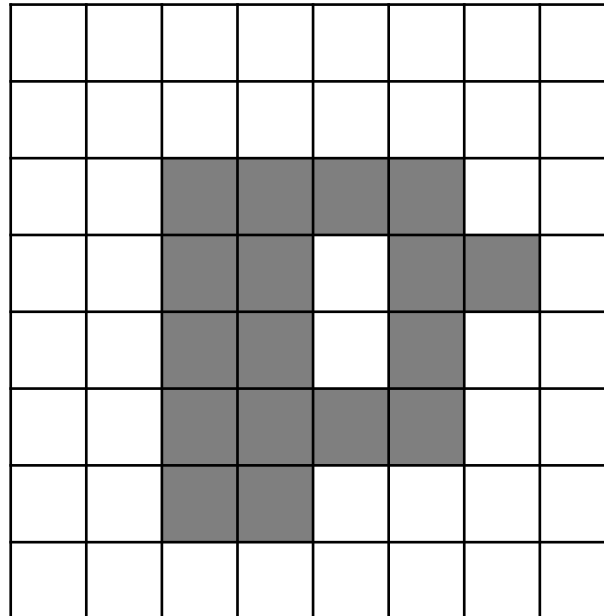
- Dado o índice do vizinho que será o próximo ponto de contorno, sabemos qual o índice que precisamos começar a procurar quando estivermos na próxima iteração do algoritmo

Relação de vizinho
inicial a ser procurado
dado o último vizinho
encontrado

Vizinho encontrado	Vizinho inicial
0	7
1	7
2	1
3	1
4	3
5	3
6	5
7	5

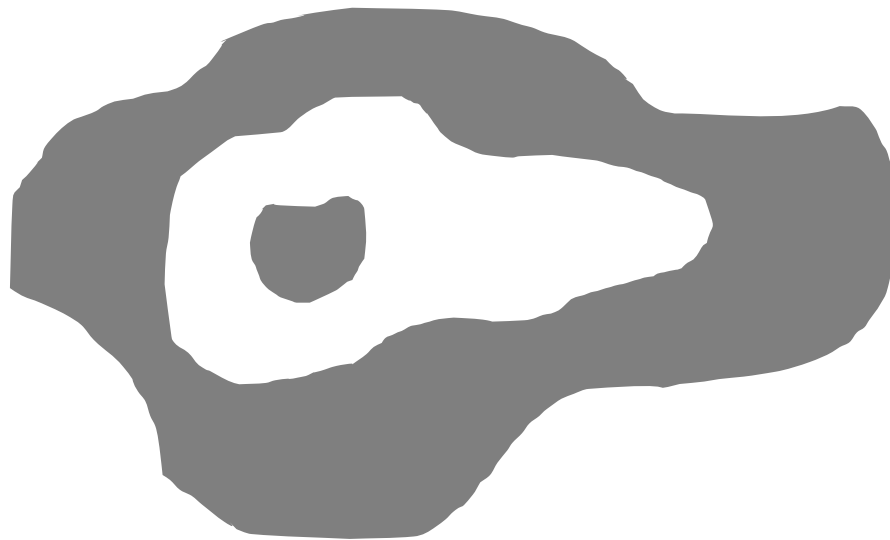
Obtenção do contorno paramétrico

O algoritmo pode ser generalizado para identificar o contorno de buracos no objeto...



Obtenção do contorno paramétrico

...ou até mesmo identificar hierarquias de objetos dentro de outros objetos

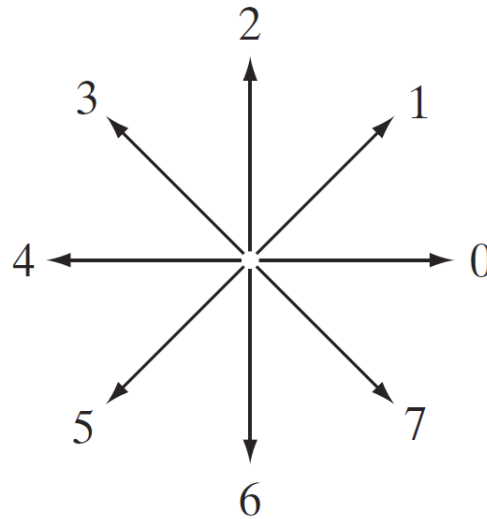


Obtenção do contorno paramétrico

Notebook “**Extração de contorno paramétrico**”

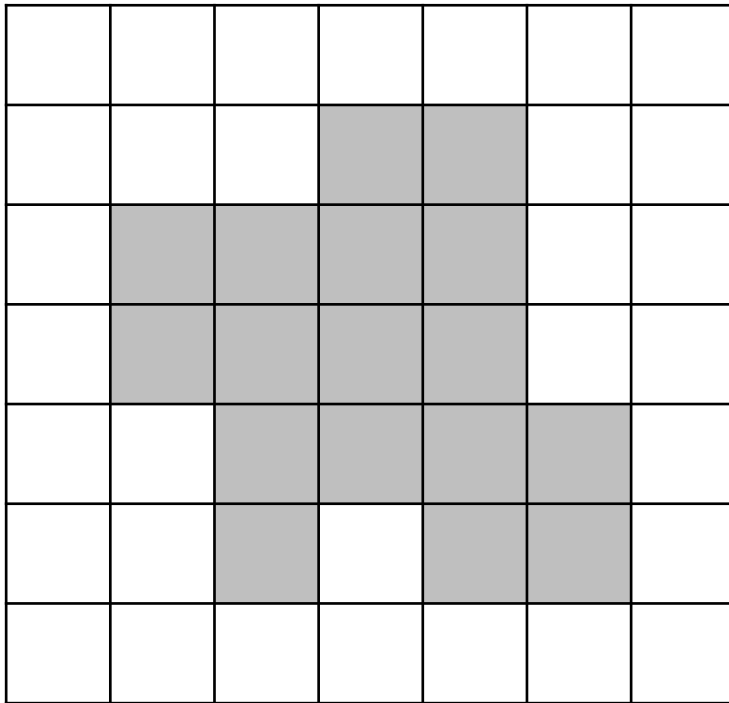
Código da cadeia de Freeman

- O código da cadeia é uma forma compacta de representar um contorno, e que pode facilitar o cálculo de algumas propriedades que veremos a seguir
- Para criar o código, primeiro definimos uma enumeração para os vizinhos de um dado pixel. Usualmente, o seguinte padrão é utilizado:



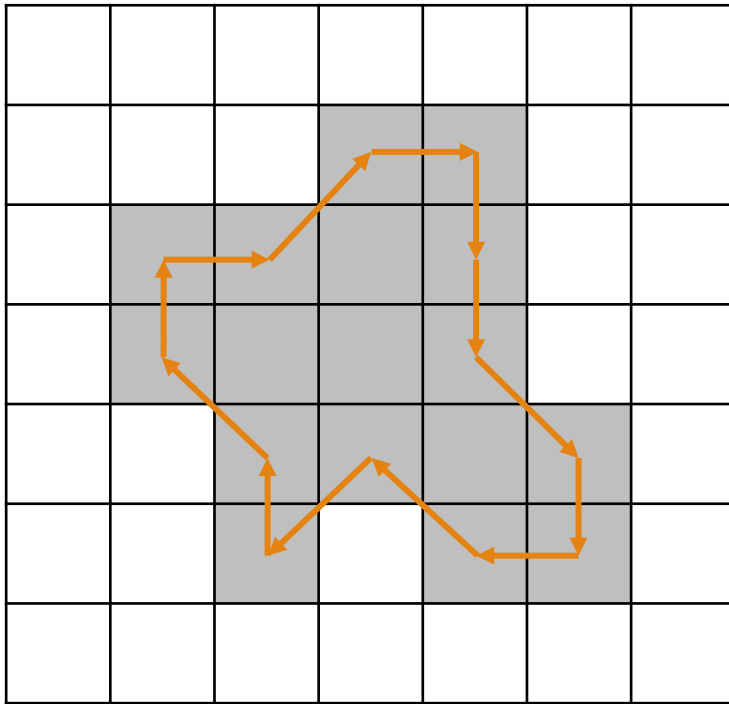
Código da cadeia de Freeman

- Em seguida, encontramos o contorno paramétrico de um dado objeto



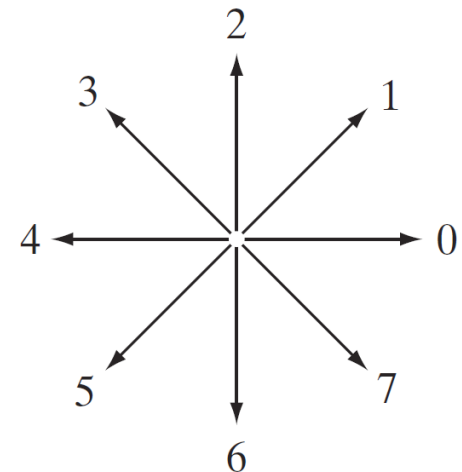
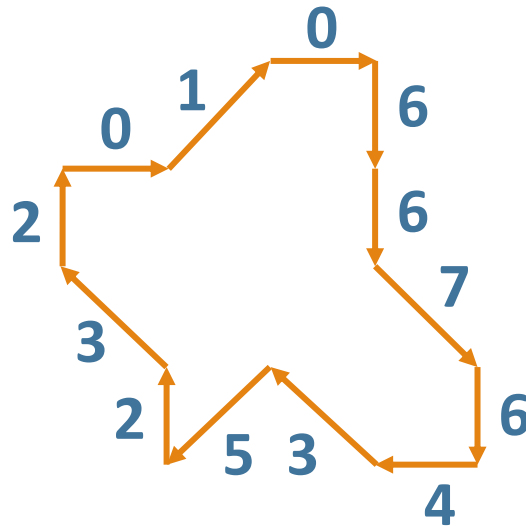
Código da cadeia de Freeman

- Em seguida, encontramos o contorno paramétrico de um dado objeto



Código da cadeia de Freeman

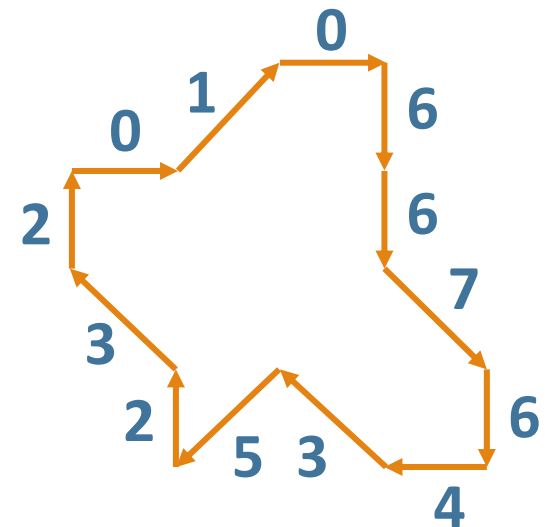
- Para cada transição entre pontos de borda do contorno, associamos o número referente à direção da transição de acordo com o nosso código



Código da cadeia de Freeman

- Para cada transição entre pontos de borda do contorno, associamos o número referente à direção da transição de acordo com o nosso código
- O objeto pode então ser representado pela sequência calculada. No caso ao lado, o objeto é dado pela sequência

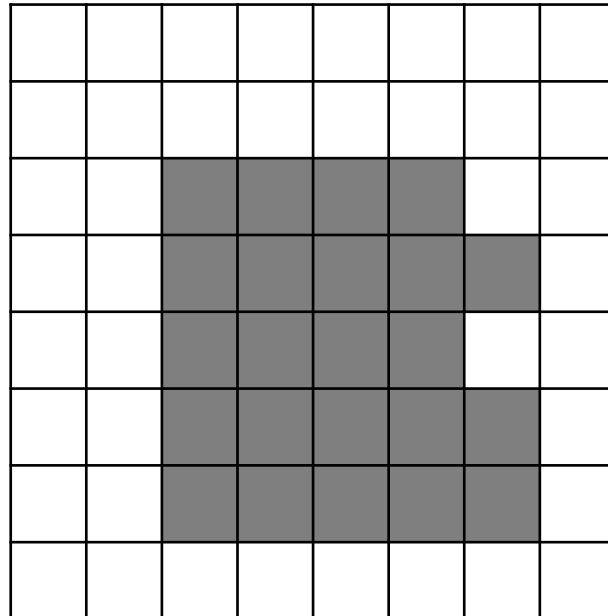
0667643523201



Algunas propiedades básicas de formas

Algumas propriedades básicas de formas

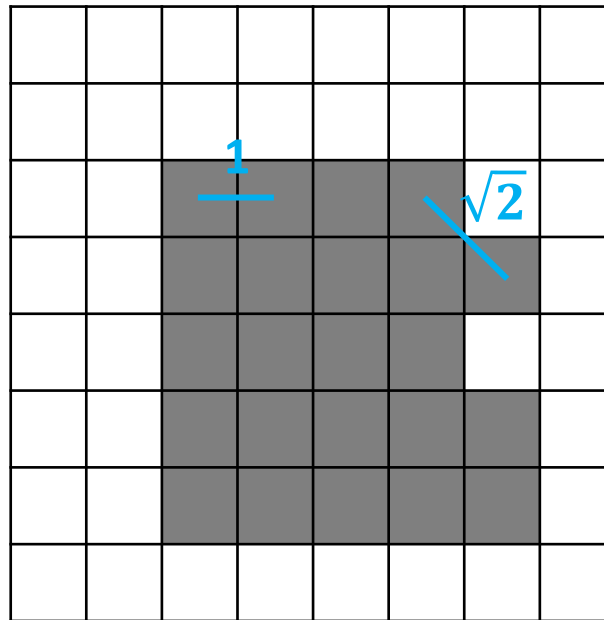
Perímetro: número de pixels no contorno



Algumas propriedades básicas de formas

Uma melhor medida de perímetro: **comprimento de arco** entre cada ponto adjacente de borda.

Nesse caso, vizinhos verticais ou horizontais possuem distância 1, já vizinhos diagonais possuem distância $\sqrt{2}$



* O perímetro pode ser facilmente obtido a partir do código de Freeman

Algumas propriedades básicas de formas

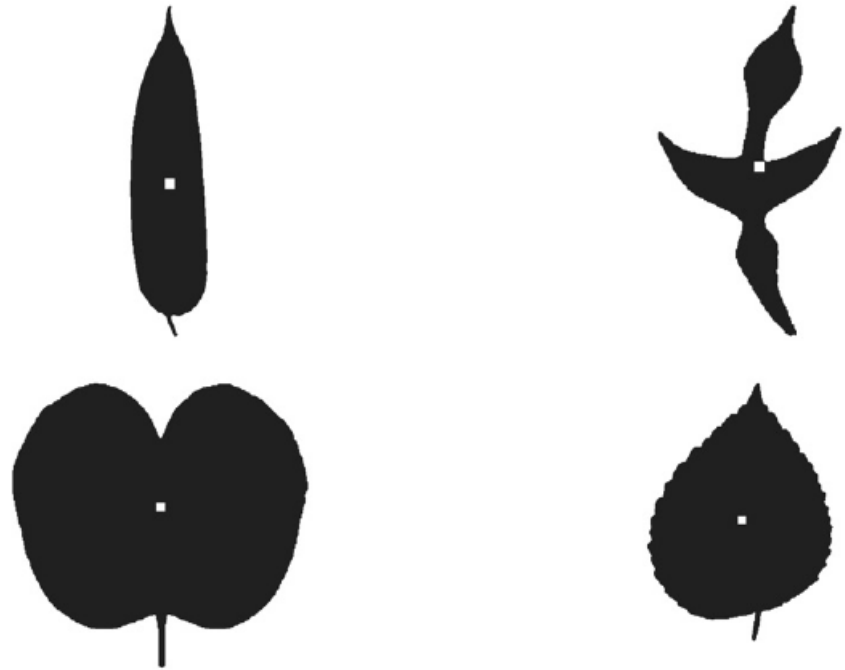
Área: número de pixels pertencentes ao objeto. Outras abordagens podem ser utilizadas para a obtenção de valores mais precisos.

Algumas propriedades básicas de formas

Centróide: média das posições dos pixels de contorno

$$C_x = \frac{\sum_{i=1}^N x(i)}{N}$$

$$C_y = \frac{\sum_{i=1}^N y(i)}{N}$$

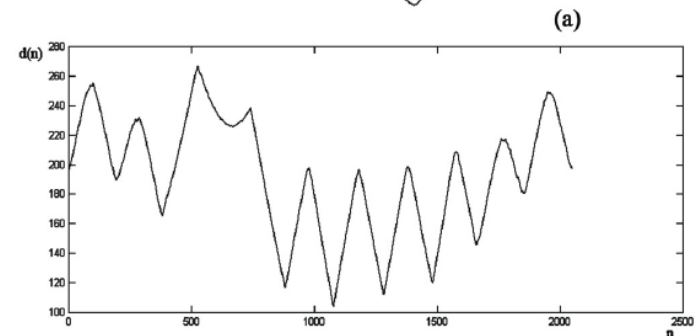
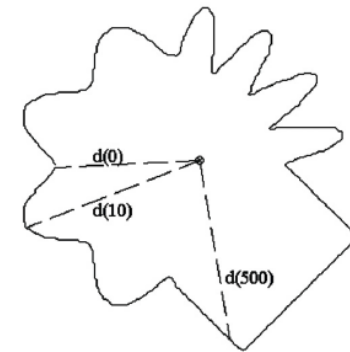


Algumas propriedades básicas de formas

Distância ao centróide: distância entre cada ponto de contorno e o centróide

$$d(t) = \sqrt{(x(t) - C_x)^2 + (y(t) - C_y)^2}$$


- Outras propriedades podem ser derivadas de $d(t)$, como por exemplo o máximo, mínimo e valor médio de $d(t)$.
- Também podemos calcular a razão entre o máximo e mínimo de $d(t)$



Algumas propriedades básicas de formas

Diâmetro: maior distância entre dois pontos de contorno

Definição alternativa: o raio de um disco possuindo a mesma área que o objeto:

$$D = 2 \sqrt{\frac{A}{\pi}}$$


Área do objeto

Área de um disco:

$$A = \pi r^2$$

Algumas propriedades básicas de formas

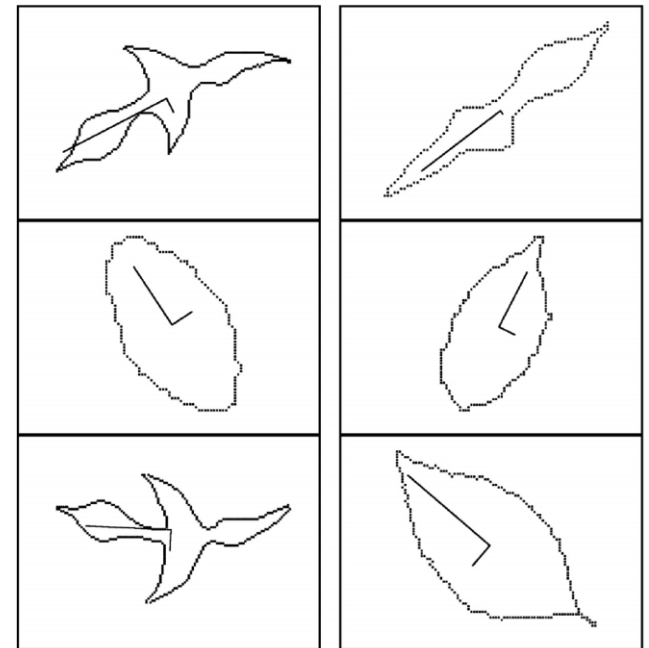
Eixo maior e eixo menor: estão associados com a direção de maior dispersão do objeto (direção na qual o objeto é mais “longo”)

Primeiramente, a matriz de covariância de tamanho 2×2 é calculada. Cada elemento dessa matriz é dado por:

$$C_{xx} = \frac{1}{N} \sum_{i=1}^N (x(t) - C_x)(x(t) - C_x)$$

$$C_{yy} = \frac{1}{N} \sum_{i=1}^N (y(t) - C_y)(y(t) - C_y)$$

$$C_{xy} = C_{yx} = \frac{1}{N} \sum_{i=1}^N (x(t) - C_x)(y(t) - C_y)$$

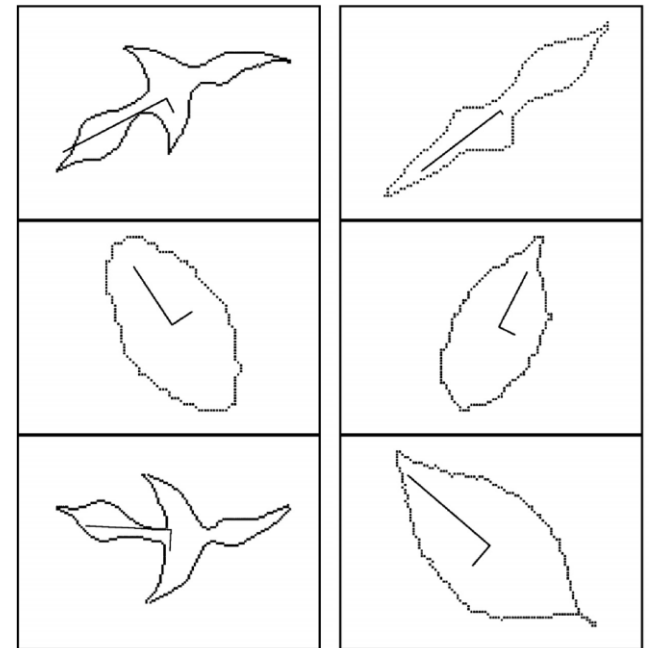


Algumas propriedades básicas de formas

Eixo maior e eixo menor: estão associados com a direção de maior dispersão do objeto (direção na qual o objeto é mais “longo”)

- Em seguida, são calculados os autovalores e autovetores da matriz de covariância
- Os autovetores indicam a direção dos eixos maiores e menores, e os respectivos autovalores indicam o “tamanho” do objeto na direção dos autovetores

Elongação do objeto: a razão entre o tamanho do eixo maior e menor



Algumas propriedades básicas de formas

Circularidade: quantifica quão circular é o objeto

$$C = \frac{4\pi A}{P^2}$$

onde A é a área e P o perímetro

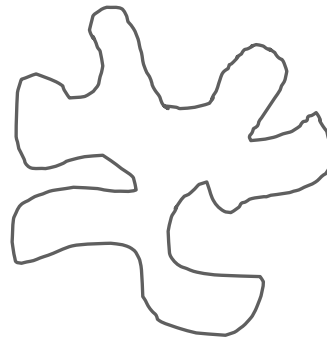
Para um círculo:

$$A = \pi r^2$$

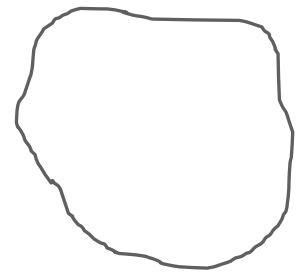
$$P = 2\pi r$$

$$C = 1$$

Baixa circularidade



Alta circularidade



Algumas propriedades básicas de formas

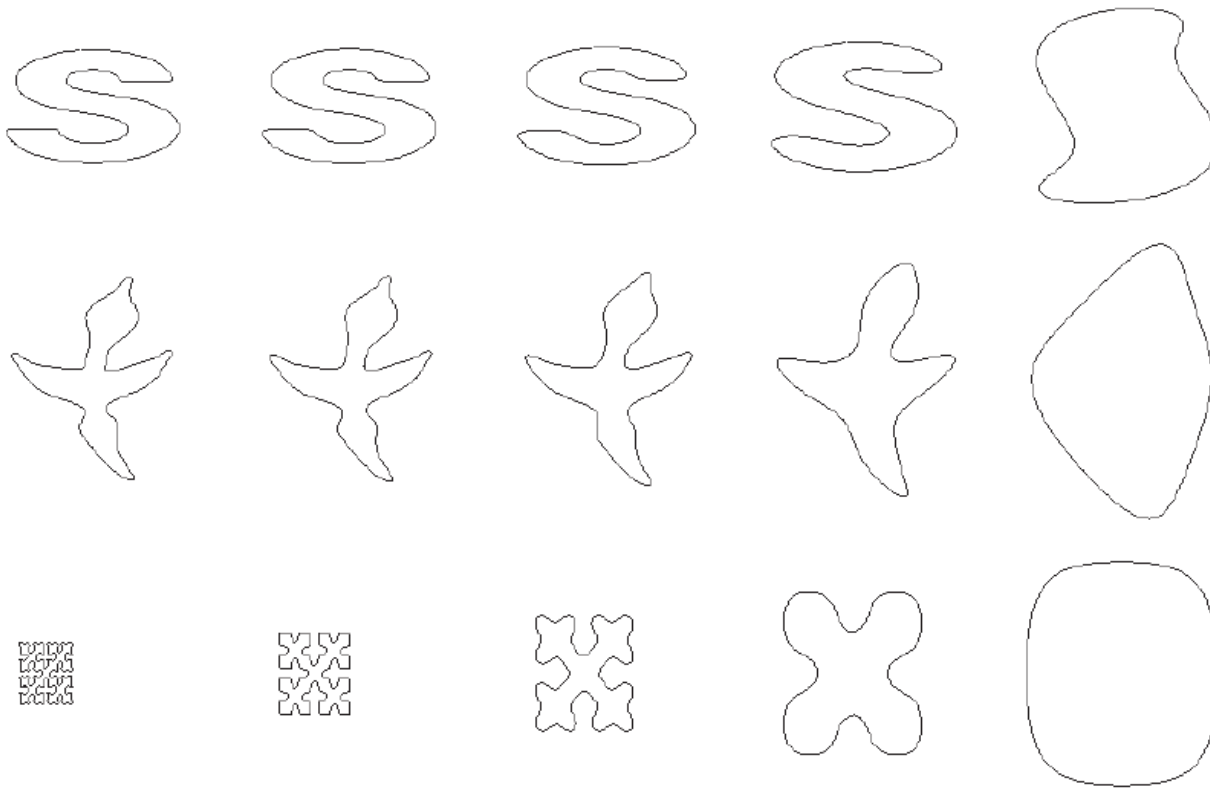
Curvatura: quantifica o grau de tortuosidade do contorno

$$k(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}}$$

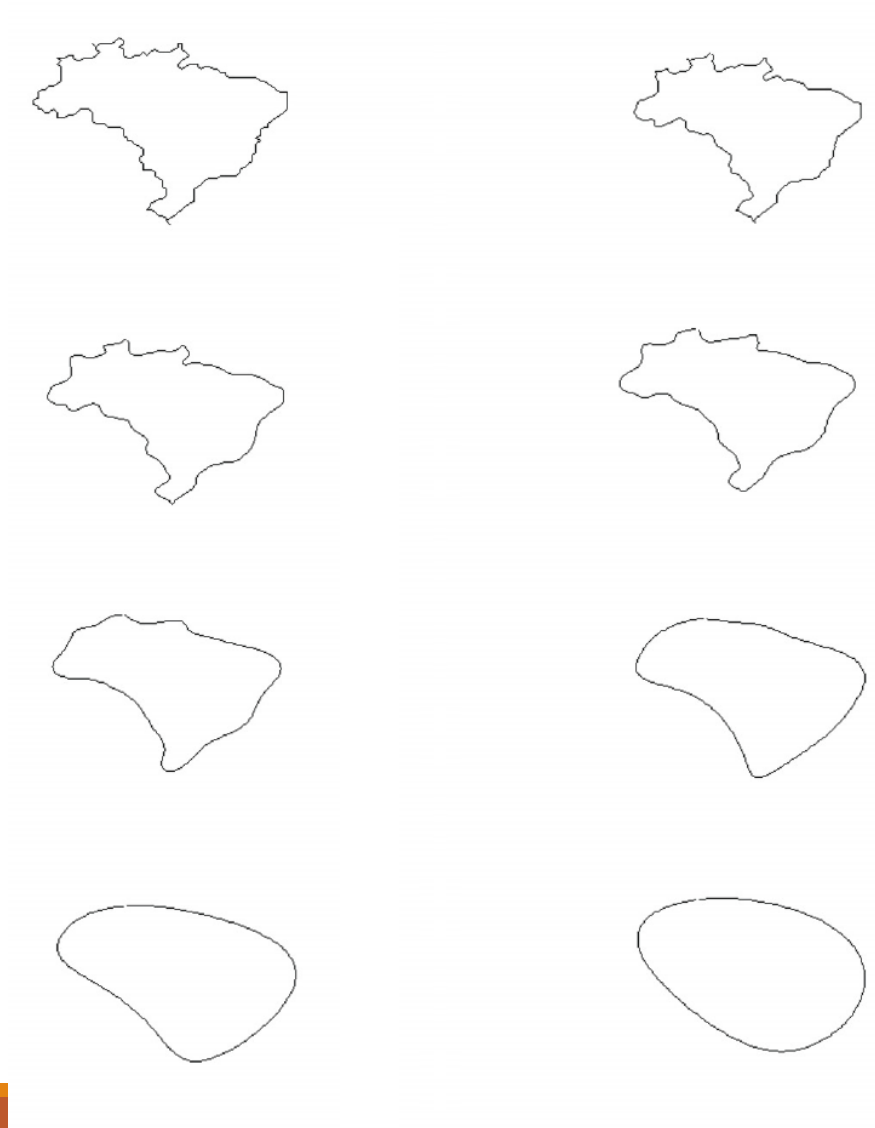
onde \dot{x} indica a primeira derivada e \ddot{x} a segunda derivada do sinal $x(t)$

Caracterização multiescala de formas

- É possível suavizar o contorno de um dado objeto
- Isso é feito através da convolução das coordenadas $x(t)$ e $y(t)$ do contorno com uma gaussiana



Caracterização multiescala de formas



Idealmente, medidas de forma devem possuir as seguintes características:

- **Identificabilidade:** formas que são percebidas de maneira similar por humanos devem levar a valores similares da medida

Idealmente, medidas de forma devem possuir as seguintes características:

- **Identificabilidade:** formas que são percebidas de maneira similar por humanos devem levar a valores similares da medida
- **Invariância à translação, rotação e escala:** a localização, o ângulo de rotação e a escala da forma na imagem não deve afetar os valores da medida

Idealmente, medidas de forma devem possuir as seguintes características:

- **Identificabilidade:** formas que são percebidas de maneira similar por humanos devem levar a valores similares da medida
- **Invariância à translação, rotação e escala:** a localização, o ângulo de rotação e a escala da forma na imagem não deve afetar os valores da medida
- **Robustez a ruído:** os valores da medida devem mudar pouco se houver ruído na imagem

Idealmente, medidas de forma devem possuir as seguintes características:

- **Identificabilidade:** formas que são percebidas de maneira similar por humanos devem levar a valores similares da medida
- **Invariância à translação, rotação e escala:** a localização, o ângulo de rotação e a escala da forma na imagem não deve afetar os valores da medida
- **Robustez a ruído:** os valores da medida devem mudar pouco se houver ruído na imagem
- **Invariância a ocultação:** se partes do objeto estiverem escondidas, devido a outros objetos ou pelo fato do objeto estar parcialmente fora da imagem, o valor da medida não deve mudar muito

Idealmente, medidas de forma devem possuir as seguintes características:

- **Identificabilidade:** formas que são percebidas de maneira similar por humanos devem levar a valores similares da medida
- **Invariância à translação, rotação e escala:** a localização, o ângulo de rotação e a escala da forma na imagem não deve afetar os valores da medida
- **Robustez a ruído:** os valores da medida devem mudar pouco se houver ruído na imagem
- **Invariância a oclusão:** se partes do objeto estiverem escondidas, devido a outros objetos ou pelo fato do objeto estar parcialmente fora da imagem, o valor da medida não deve mudar muito
- **Independência estatística:** é desejado que diferentes medidas de forma não estejam fortemente correlacionadas

Projeto 3

- Não utilizar funções prontas para implementar o principal conceito associado ao tema. Na dúvida, pergunte que funções/bibliotecas podem ser utilizadas no projeto.
- Entregáveis:
 - Código produzido (a organização do código também será avaliada!)
 - Um artigo escrito em Latex (~6 páginas em coluna dupla ou ~10 em coluna única) contendo:
 - Resumo
 - Introdução
 - Motivação do uso do método (porque usar? Em que situações ele é importante?)
 - Objetivos
 - O que será analisado sobre o método?
 - Metodologia
 - Explicação sobre a teoria do método
 - Explicação sobre a parte mais importante do código
 - Resultados
 - Conclusões
- Data de entrega: 28/02

Projeto 3 – Tema 1

- Morfologia em nível de cinza
- Implementar dilatação, erosão, abertura, fechamento e transformada top-hat em nível de cinza
- Utilizar um elemento estruturante uniforme “flat”
- Capítulo 9.6 do livro Processamento Digital de Imagens, Gonzalez e Woods 3ª Edição.
- Mostrar aplicações para cada tipo de operação

Projeto 3 – Tema 1

Erosão:

$$[f \ominus b](x, y) = \min_{(s, t) \in b} \{f(x + s, y + t)\} \quad \text{Filtro de mínimo}$$

Dilatação:

$$[f \oplus b](x, y) = \max_{(s, t) \in b} \{f(x - s, y - t)\} \quad \text{Filtro de máximo}$$

Abertura:

$$f \circ b = (f \ominus b) \oplus b$$

Fechamento:

$$f \bullet b = (f \oplus b) \ominus b$$

Top-hat:

$$T_{\text{hat}}(f) = f - (f \circ b)$$

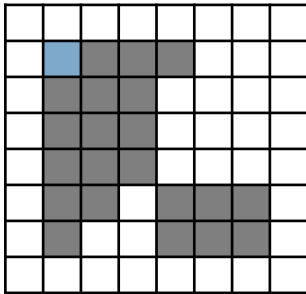
- Não utilizar funções prontas para fazer a filtragem de mínimo/máximo

Projeto 3 – Tema 2

- Detecção de componentes conexos utilizando flood fill
- Dado o pixel inicial pertencente a um componente, basta fazer uma busca em largura na imagem

Pixels visitados: [(1,1)]

Fila dos próximos pixels a serem visitados: $[(1,2),(2,1)]$



Projeto 3 – Tema 3

- Implementação da transformada distância
- Descrita nos slides anteriores
- Ideia geral do algoritmo:
 - Defina uma imagem de “borda externa” como $I_b = I \oplus B - I$, onde B é um elemento estruturante de tamanho 3x3 contendo o valor 1
 - Para cada pixel p do objeto, encontre o pixel de I_b mais próximo ao pixel p . Armazene a distância na posição do pixel p
- Utilizar dois tipos de distâncias: euclidiana e city block
- Calcular a transformada distância de algumas formas (imagens) distintas.

Euclidiana:

$$p_1 = (5, 8)$$

$$p_2 = (9, 10)$$

$$d_{12} = \sqrt{(p_1[0] - p_2[0])^2 + (p_1[1] - p_2[1])^2}$$

City block:

$$d_{12} = |p_1[0] - p_2[0]| + |p_1[1] - p_2[1]|$$

Projeto 3 – Tema 4

- Implementação do cálculo de esqueleto de um objeto em uma imagem binária
- Algoritmo explicado nos slides acima
- Calcular o esqueleto de algumas formas (imagens) distintas
- Discutir problemas que ocorrem no esqueleto. Em especial, ramificações espúrias.

Projeto 3 – Tema 5

- Suavização de contorno paramétrico
- Dado uma imagem binária contendo um objeto, o contorno paramétrico do objeto é suavizado utilizando um filtro gaussiano
- O contorno suavizado é então utilizado para desenhar o objeto em uma nova imagem
- Utilizar diferentes valores de sigma e fazer experimentos para diferentes tipos de objetos em imagens (redondo, alongado, irregular, etc)
- Plotar gráficos mostrando como área, perímetro e circularidade variam de acordo com a suavização

Projeto 3 – Tema 6

- Cálculo da curvatura ao longo do contorno de objetos
- Utilizar a fórmula da curvatura apresentada nos slides acima
- Note que um valor de curvatura é calculado para cada ponto do contorno
- Identificar picos positivos e negativos de curvatura
 - É recomendado que a curvatura seja suavizada para identificação dos picos
- Definir 5 propriedades para caracterizar objetos a partir da curvatura (valor máximo, etc)