Processamento de Imagens usando Grafos

Prof. Alexandre Xavier Falcão

Segundo semestre de 2004

1 Definição original de borda

Seja $\hat{I} = (D_I, I)$ uma imagem binária contendo vários objetos, onde $I(p) = 1$ para pixels de objeto e $I(p) = 0$ para pixels de fundo. Na aula anterior, definimos uma borda de objeto como um subconjunto 8-conexo de D_I, tal que um pixel p pertence a borda quando $I(p) = 1$ e existe um pixel q adjacente-4 de p (i.e. $q \in A_4(p)$) com valor $I(q) = 0$. A representação de um objeto por suas bordas apresenta características topológicas indesejáveis para vários algoritmos de análise de imagens. Alguns exemplos são apresentados abaixo.

(a) Linha

```
   11  1111
     11
```

(b) Contorno com ramos

```
     1
     1
     111
  111   111111
     111
```

(c) Contorno com pixel de estrangulamento

```
       11
  111 1  1
  1  1   1
   11 111
```

(d) Dois contornos que se tocam

```
 111  11
1   11  1
1111  11
```

(e) Contorno que toca o lado oposto

```
  111  111
 1   11   1
 1   11   1
  111  111
```

No caso (a), o objeto é muito fino e todos os seus pixels são considerados pixels de borda (i.e. o objeto é um **esqueleto**). Isto impossibilita uma distinção entre interior e borda, que pode ser desejável em alguns algoritmos. O caso (b) impossibilita a enumeração seqüêncial de pixels 8-adjacentes. Na verdade, os ramos na maioria das vezes não fazem parte do objeto, pois decorrem de erros de segmentação. Os casos de (c-e) geram ambigüidades, pois não sabemos quando temos uma borda de um único objeto, contendo dois contornos, e quando temos duas bordas, contendo um contorno cada, provenientes de dois objetos que se tocam. Também é desejável em muitas aplicações obter uma representação hierárquica das bordas de um objeto, através de uma relação de continência entre bordas.

Para minimizar esses problemas, nós buscamos uma representação mais formal, onde uma **borda** é como definida originalmente um contorno fechado, 8-conexo e orientado (i.e. uma curva de Jordan). Fechado no sentido que separa o interior do exterior do objeto, orientado porque o interior deve ficar sempre à esquerda (ou à direita) do contorno, e 8-conexo porque o contorno deve ser um caminho $\langle p_1, p_2, \ldots, p_n \rangle$ de pixels distintos, tais que (p_i, p_{i+1}), $i = 1, 2, \ldots, n-1$, e (p_n, p_1) são 8-adjacentes.

O problema, portanto, passa a ser a extração desses contornos em $\hat{I}$, tratando todos os casos descritos acima. Uma solução vagamente descrita na literatura é o algoritmo da "bengala", que simula uma pessoa com uma bengala caminhando sobre a borda do objeto. A cada passo, a bengala parte do pixel anterior e gira no sentido horário (ou anti-horário) até encontrar o próximo pixel p da borda do objeto. Não é difícil perceber que esta solução falha nos casos a,b,c, e d. Portanto, uma solução mais robusta é proposta a seguir.

2 Rotulação de pixels de contornos

Seja $\hat{B} = (D_I, B)$ uma imagem de bordas obtida com a definição antiga onde $B(p) = 1$, se $I(p) = 1$ e existe $q \in A_4(p)$ tal que $I(q) = 0$, e $B(p) = 0$ no caso contrário. Desejamos identificar um único contorno por borda de objeto, eliminando ramos e linhas (casos b e a). Bordas com pixels de estrangulamento serão tratadas como contornos com ramos iniciando

nesses pixels (caso c). O caso d será tratado como dois objetos que se tocam e o caso e deve resultar em um único objeto cuja borda se toca em lados opostos.

Nosso objetivo é escrever um algoritmo para enumerar seqüencialmente de 1 até n os pixels subseqüentes de cada contorno em $\hat{B}$. Assume-se que situações específicas, onde os objetos possuem 1 pixel de largura, devam ser tratadas com uma **dilatação** do objeto antes de aplicar o algoritmo. Infelizmente, a forma mais robusta de obter contornos é interpretar os **vértices dos pixels** da imagem como os nós de um grafo cujos arcos são obtidos por uma relação de adjacência-4 entre vértices. Porém, todos os outros algoritmos trabalham na resolução de pixel e isto envolveria modificá-los também.

Para resolver as ambigüidades dos casos (c-e), vamos evitar arcos entre pixels 8-adjacentes que passem **por dentro e por fora** do objeto. Isto é, para cada arco $(p, q) \in A_8$ temos os conceitos de pixel $e(p, q)$ à esquerda e pixel $d(p, q)$ à direita do arco.

```
e       eq
pq      pd
d
```

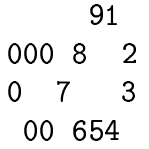
Esses pixels podem ser obtidos facilmente a partir da relação de adjacência A_8. Um arco (p, q) passa por dentro ou por fora de um objeto quando $I(e(p, q)) = I(d(p, q))$. Vamos também definir uma **ordem** para os pixels em A_8 no sentido horário (ou anti-horário) com relação ao pixel central, iniciando no vizinho-4 à esquerda. A estratégia é visitar cada pixel de contorno uma única vez caminhando no sentido anti-horário (horário) através de uma **busca em profundidade** no grafo. A enumeração é feita ao final da busca, percorrendo de volta a lista de predecessores. Note que, devemos escolher um pixel inicial para cada contorno e que este pixel deve possuir ao menos um arco partindo dele, que satisfaça a primeira condição acima. Os resultados esperados para os casos acima são:

```
(a) Linha
                    00  0000
                      00
```

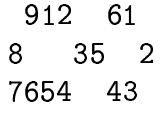
```
(b) Contorno com ramos

                       0
                       0
                       812
                    007   300000
                       654
```

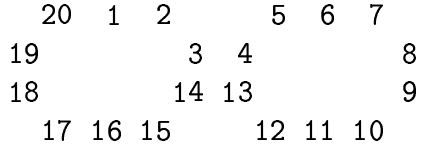
(c) Contorno com pixel de estrangulamento



(d) Dois contornos que se tocam



(e) Contorno que toca o lado oposto



Algoritmo rotulação de pixels de contorno:

Entrada: Imagem binária $\hat{I} = (D_I, I)$.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde cada pixel de um contorno tem um rótulo l, $l = 1, 2, \ldots, n$ e n é o número total de pixels do contorno.
Auxiliares: Fila LIFO Q (i.e. uma **pilha**), adjacência ordenada e irreflexiva A_8. Imagem $\hat{C} = (D_I, C)$, onde a cor $C(p)$ de um pixel $p \in D_I$ pode ser branco, cinza ou preto, dependendo se p nunca foi inserido em Q, está em Q, ou já foi removido de Q, respectivamente. Imagem $\hat{P} = (D_I, P)$ de predecessores, imagem $\hat{B} = (D_I, B)$ de bordas (definição antiga), e variável l.

1. Para todo pixel $p \in D_I$, faça
2. $\quad L(p) \leftarrow 0$, $B(p) \leftarrow 0$ e $C(p) \leftarrow branco$.
3. $\quad$ Se $I(p) = 1$ e existe $q \in A_4(p)$ tal que $I(q) = 0$, então faça $B(p) \leftarrow 1$.
4. Para todo pixel $s \in D_I$, tal que $B(s) = 1$ e $C(s) = branco$, faça
5. $\quad$ Se existe $q \in A_8(s)$, tal que $B(q) = 1$ e $I(e(s,q)) \neq I(d(s,q))$, então
6. $\quad\quad$ Faça $C(s) \leftarrow cinza$, $P(s) \leftarrow nil$, e insira s em Q.
7. $\quad\quad$ Enquanto $Q \neq \emptyset$, faça

8. Remova p de Q e faça $C(p) \leftarrow preto$.

9. Para todo $q \in A_8(p)$, faça

10. Se $q = s$ e $P(p) \neq s$, então vá para a linha 14.

11. Se $B(q) = 1$, $C(q) \neq preto$ e $I(e(p,q)) \neq I(d(p,q))$, então

12. Faça $P(q) \leftarrow p$.

13. Se $C(q) = branco$, então insira q em Q e faça $C(q) \leftarrow cinza$.

14. Faça $l \leftarrow 1$ e esvazie Q.

15. Enquanto $p \neq nil$, faça

16. $L(p) \leftarrow l$, $p \leftarrow P(p)$ e $l \leftarrow l + 1$.

O algoritmo acima tem sido usado com sucesso no cálculo de esqueletos multi-escala, como veremos mais adiante.

3 Rotulação de contornos

Um variante simples do algoritmo anterior permite identificar cada contorno como uma unidade. Suponha que agora desejamos enumerar todos os pixels de um mesmo contorno com o rótulo do contorno, onde todos os k contornos da imagem recebem um rótulo distinto de 1 até k.

Algoritmo rotulação de contornos:

Entrada: Imagem binária $\hat{I} = (D_I, I)$.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde cada contorno tem um rótulo l, $l = 1, 2, \ldots, k$ e k é o número total de contornos.
Auxiliares: Fila LIFO Q (i.e. uma **pilha**), adjacência ordenada e irreflexiva A_8. Imagem $\hat{C} = (D_I, C)$, onde a cor $C(p)$ de um pixel $p \in D_I$ pode ser branco, cinza ou preto, dependendo se p nunca foi inserido em Q, está em Q, ou já foi removido de Q, respectivamente. Imagem $\hat{P} = (D_I, P)$ de predecessores, imagem $\hat{B} = (D_I, B)$ de bordas (definição antiga) e variável l.

1. Para todo pixel $p \in D_I$, faça

2. $L(p) \leftarrow 0$, $B(p) \leftarrow 0$ e $C(p) \leftarrow branco$.

3. Se $I(p) = 1$ e existe $q \in A_4(p)$ tal que $I(q) = 0$, então faça $B(p) \leftarrow 1$.

4. Faça $l \leftarrow 1$.
5. Para todo pixel $s \in D_I$, tal que $B(s) = 1$ e $C(s) = branco$, faça
6. Se existe $q \in A_8(s)$, tal que $I(q) = 1$ e $I(e(s,q)) \neq I(d(s,q))$, então
7. Faça $C(s) \leftarrow cinza$, $P(s) \leftarrow nil$, e insira s em Q.
8. Enquanto $Q \neq \emptyset$, faça
9. Remova p de Q e faça $C(p) \leftarrow preto$.
10. Para todo $q \in A_8(p)$, faça
11. Se $q = s$ e $P(p) \neq s$, então vá para a linha 15.
12. Se $B(q) = 1$ e $C(q) \neq preto$ e $I(e(p,q)) \neq I(d(p,q))$, então
13. Faça $P(q) \leftarrow p$.
14. Se $C(q) = branco$, então insira q em Q e faça $C(q) \leftarrow cinza$.
15. Enquanto $p \neq nil$, faça
16. $L(p) \leftarrow l$ e $p \leftarrow P(p)$.
17. Faça $l \leftarrow l + 1$ e esvazie Q.

O algoritmo acima tem sido usado com sucesso no cálculo de esqueletos por zonas de influência, como veremos mais adiante, e cada *bin* do histograma da imagem $\hat{L}$ representa o perímetro de um contorno.