Processamento de Imagens usando Grafos

Prof. Alexandre Xavier Falcão

Aula 02

1 Relação de conexidade

Um **caminho** π no grafo $G = (D_I, A)$ é uma seqüência de pixels distintos $< p_1, p_2, \ldots, p_n >$, onde $(p_i, p_{i+1}) \in A$, $i = 1, 2, \ldots, n-1$. O pixel p_1 é a origem $org(\pi)$ do caminho, p_n é o destino $dst(\pi)$, e o caminho π é dito **trivial** se $\pi =< p_1 >$.

Seja π um caminho que termina em um pixel p e $(p, q) \in A$, então $\pi \odot \langle p, q \rangle$ é dito o caminho resultante da concatenação de π e $\langle p, q \rangle$ com as duas instâncias de p se fundindo em uma.

Um pixel q é dito **conexo** a um pixel p se existir um caminho de p a q em $G = (D_I, A)$.

2 Componente Conexo

Um **componente conexo** em G é um subconjunto de D_I, onde todos os pares (p, q) de pixels são conexos (i.e. existe um caminho de p a q e um caminho de q a p, que não necessariamente são os mesmos).

3 Partição

Uma **partição** de G é um conjunto de componentes conexos e disjuntos, cuja união é o conjunto D_I. Em muitas situações desejamos particionar uma imagem, identificando individualmente cada componente conexo. Por exemplo, podemos enumerar todos os componentes 1's de uma imagem binária ou definir componentes conexos em uma imagem cinza a partir de uma relação de dissimilaridade local entre pixels.

4 Rotulação de componentes conexos

Considere o problema de obter uma partição da imagem, onde cada componente conexo recebe um rótulo $l = 1, 2, \ldots, c$ e o número c de componentes depende da relação de adjacência A. Uma solução simples é aplicar a seguinte busca em largura no grafo G.

Algoritmo de rotulação de componentes conexos:

Entrada: Imagem $\hat{I} = (D_I, \vec{I})$ e relação de adjacência A.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde $L(p) = 0$ inicialmente $\forall p \in D_I$.
Auxiliares: FIFO Q e variável inteira $l = 1$.

1. Para todo pixel $p \in D_I$, tal que $L(p) = 0$, faça
2. $L(p) \leftarrow l$ e insira p em Q.
3. Enquanto $Q \neq \emptyset$ faça
4. Remova p de Q.
5. Para todo $q \in A(p)$, tal que $L(q) = 0$, faça
6. $L(q) \leftarrow l$ e insira q em Q.
7. $l \leftarrow l + 1$.

Observe que a segmentação resultante vai depender da definição de A. O caso particular de uma imagem binária é muito usado na prática.

5 Rotulação por conjuntos disjuntos

O problema de rotulação de componentes conexos também pode ser visto como um problema de manutenção de conjuntos disjuntos (algoritmo *union-find*). Neste caso, porém, vamos considerar A uma relação que depende apenas das posições relativas entre os pixels. A rotulação pode levar em conta propriedades mais globais dos componente, como o brilho médio, por exemplo.

Algoritmo de rotulação por conjuntos disjuntos:

Entrada: Imagem cinza $\hat{I} = (D_I, I)$, relação de adjacência A e limiar T.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$.
Auxiliares: Imagem de representantes $\hat{R} = (D_I, R)$ de cada componente, imagem $\hat{S} = (D_I, S)$ onde cada representante guarda a soma dos brilhos dos pixels do componente, imagem $\hat{N} = (D_I, N)$ onde cada representante guarda o número de pixels de seu componente, e variável inteira $l = 1$.

1. Para todo pixel $p \in D_I$, faça $R(p) \leftarrow p$, $S(p) \leftarrow I(p)$, e $N(p) \leftarrow 1$.
2. Para todo pixel $p \in D_I$, faça

3. $r_p \leftarrow Representante(\hat{R}, p)$.
4. Para todo pixel $q \in A(p)$, faça
5. $r_q \leftarrow Representante(\hat{R}, q)$.
6. Se $r_q \neq r_p$, então
7. Se $|\frac{S(r_p)}{N(r_p)} - \frac{S(r_q)}{N(r_q)}| \leq T$, então
8. $Junte(\hat{R}, r_p, r_q, \hat{S}, \hat{N})$.
9. Para todo pixel $p \in D_I$, faça
10. $R(p) \leftarrow Representante(\hat{R}, p)$.
11. Se $R(p) = p$, então $L(p) \leftarrow l$ e $l \leftarrow l + 1$.
12. Para todo pixel $p \in D_I$, faça $L(p) \leftarrow L(R(p))$.

Algoritmo para encontrar o representante com compressão:

$Representante(\hat{R}, p)$

1. Se $R(p) = p$, então
2. retorne p.
3. Caso contrário,
4. retorne $R(p) \leftarrow Representante(\hat{R}, R(p))$.

Algoritmo de união de componentes com otimização:

$Junte(\hat{R}, r_p, r_q, \hat{S}, \hat{N})$

1. Se $N(r_p) \geq N(r_q)$, então
2. $N(r_p) \leftarrow N(r_p) + N(r_q)$, $S(r_p) \leftarrow S(r_p) + S(r_q)$, e $R(r_q) \leftarrow r_p$.
3. Caso contrário,
4. $N(r_q) \leftarrow N(r_q) + N(r_p)$, $S(r_q) \leftarrow S(r_q) + S(r_p)$, $R(r_p) \leftarrow r_q$, e $r_p \leftarrow r_q$.

Observe que r_p pode ser atualizado na linha 4 da função Junte, portanto deve ser passado por referência.

6 Tarefa

Escreva um programa para rotular componentes conexos usando um dos algoritmos acima e as relações de adjacência que você já implementou.