Processamento de Imagens usando Grafos

Prof. Alexandre Xavier Falcão

Segundo semestre de 2004

1 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.

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. $\quad r_p \leftarrow Representante(\hat{R}, p)$.
4. $\quad$ Para todo pixel $q \in A(p)$, faça
5. $\quad\quad r_q \leftarrow Representante(\hat{R}, q)$.
6. $\quad\quad$ Se $r_q \neq r_p$, então
7. $\quad\quad\quad$ Se $|\frac{S(r_p)}{N(r_p)} - \frac{S(r_q)}{N(r_q)}| \leq T$, então
8. $\quad\quad\quad\quad Junte(\hat{R}, r_p, r_q, \hat{S}, \hat{N})$.
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.

2 Imagem de predecessores

Uma **imagem de predecessores** $\hat{P}$ é um par (D_I, P) onde o mapa P é uma função que associa a cada pixel $q \in D_I$ ou outro pixel $P(q) = p \in D_I$, $(p, q) \in A$, ou uma marca $P(q) = nil \notin D_I$. No segundo caso, q é dito ser uma **raiz** do mapa.

Uma **floresta** é uma imagem de predecessores que não contém ciclos— i.e., aquela que leva todo pixel para *nil* em um número finito de iterações. Para qualquer pixel $q \in D_I$, a

floresta P define um caminho $P^*(q)$ recursivamente como $\langle q \rangle$, se $P(q) = nil$, e $P^*(p) \cdot \langle p, q \rangle$ se $P(q) = p \neq nil$. Note que uma **árvore** é uma floresta com uma única raiz.

O valor de dissimilaridade $\delta(p, q)$ representa o peso do arco $(p, q) \in A$. Uma **árvore de peso mínimo** é aquela onde a soma dos pesos dos arcos é mínima, quando comparada a todas possíveis árvores obtidas de G. Suponha, por exemplo, que desconhecemos o limiar T no problema de rotulação de componentes conexos e desejamos escolher T interativamente. Para cada valor de T, eliminamos os arcos com peso maior que T, gerando uma família de rolulações hierárquicas (i.e. **florestas de peso mínimo**, onde cada árvore é um componente conexo).

3 Rotulação hierárquica de componentes conexos

Considere, por exemplo, $(p, q) \in A$ se q é vizinho-4 de p. A estratégia é gerar uma árvore de peso mínimo, onde $\delta(p, q) = |I(p) - I(q)|$, e depois aplicar um algoritmo de rotulação para cada valor de T selecionado interativamente.

Algoritmo para gerar árvore de peso mínimo:

Entrada: Imagem cinza $\hat{I} = (D_I, I)$ e relação de adjacência A.
Saída: Imagens de pesos $\hat{C} = (D_I, C)$ e de predecessores $\hat{P} = (D_I, P)$, onde $\hat{P}$ representa a árvore e $C(q)$ é o peso $\delta(p, q)$ do arco (p, q), quando $P(q) = p$, e $C(q) = 0$ para $P(q) = nil$. Inicialmente, $C(p) = +\infty$ e $P(p) = nil$, $\forall p \in D_I$.
Auxiliares: Fila de prioridades Q, variável tmp, e arranjo $cor(p)$ indicando $branco$ quando p nunca foi inserido em Q, $cinza$ quando $p \in Q$, e $preto$ quando p já foi processado. Inicialmente $cor(p) = branco$, $\forall p \in D_I$.

1. Escolha um pixel p qualquer em D_I, $C(p) \leftarrow 0$, insira p em Q, e $cor(p) \leftarrow cinza$.
2. Enquanto $Q \neq \emptyset$ faça
3. Remova p de Q tal que $C(p)$ é mínimo e faça $cor(p) \leftarrow preto$.
4. Para todo $q \in A(p)$, tal que $cor(q) \neq preto$, faça
5. $tmp \leftarrow \delta(p, q)$.
6. Se $tmp < C(q)$, então
7. Se $cor(q) = cinza$, então remova q de Q.
8. $C(q) \leftarrow tmp$, $P(q) \leftarrow p$, insira q em Q e $cor(q) \leftarrow cinza$.

Algoritmo de rotulação hierárquica:

Entrada: Imagens de pesos $\hat{C} = (D_I, C)$ e de predecessores $\hat{P} = (D_I, P)$, e limiar T.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$.
Auxiliares: Imagem rotulada $\hat{R} = (D_I, R)$ com as raízes da floresta e variável l. Inicialmente, $\hat{R}$ é uma cópia de $\hat{P}$ e $l = 1$.

1. Para todo pixel $q \in D_I$, faça
2. $\quad p \leftarrow P(q)$.
3. $\quad$ Se $p \neq nil$, então
4. $\quad\quad$ Se $C(q) > T$, então
5. $\quad\quad\quad l \leftarrow l + 1$, $L(q) \leftarrow l$, e $R(q) \leftarrow q$.
6. $\quad$ Caso contrário, $L(q) \leftarrow 1$ e $R(q) \leftarrow q$.
7. Para todo pixel $p \in D_I$, faça
8. $\quad R(p) \leftarrow Raiz(\hat{R}, p)$.
9. $\quad L(p) \leftarrow L(R(p))$.

Algoritmo $Raiz(\hat{R}, p)$ é igual a $Representante(\hat{R}, R)$.

Note que todos os componentes conexos rotulados têm arcos com pesos menores ou iguais a T, assim como no algoritmo não hierárquico. O algoritmo apresentado é essencialmente uma adaptação do algoritmo de Prim.

4 Força de conexidade

Uma **função de custo de caminho** f é um mapeamento que atribui um valor a qualquer caminho no grafo, dentro de um conjunto V de valores ordenados, onde $-\infty$ e $+\infty$ indicam os valores mínimo e máximo em V, respectivamente. Normalmente, os valores atribuídos dependem de propriedades locais dos pixels ao longo do caminho, tais como brilho, gradiente, e posição. Exemplos:

$$\begin{aligned} f_{sum}(\langle q \rangle) &= h(q), \\ f_{sum}(\pi \cdot \langle p, q \rangle) &= f_{sum}(\pi) + \delta(p, q), \end{aligned} \tag{1}$$

onde $(p, q) \in A$, π é qualquer caminho terminando em p, $h(q)$ é um custo inicial fixo para qualquer caminho iniciando em q, e $\delta(p, q)$ é um peso não negativo associado ao arco (p, q).

Um outro exemplo é a função de custo $f_{\max}$, a qual satisfaz

$$\begin{aligned} f_{\max}(\langle q \rangle) &= h(q), \\ f_{\max}(\pi \cdot \langle p, q \rangle) &= \max\{f_{\max}(\pi), \delta(p, q)\}, \end{aligned} \tag{2}$$

onde $h(q)$ e $\delta(p, q)$ são fixos mas arbitrários.

Um caminho π é **ótimo** com relação à f quando $f(\pi) \leq f(\tau)$ para qualquer caminho τ em G, tal que $dst(\tau) = dst(\pi)$ independente do pixel de origem.

O valor de custo $f(\pi)$ de um caminho ótimo está associado à **força de conexidade** de $dst(\pi)$ com relação a $org(\pi)$, onde quanto mais baixo é o custo mais alta é a força de conexidade. A força de conexidade de um pixel q com relação a um pixel p, porém, é o custo do caminho ótimo de p a q. Neste caso, estamos considerando apenas os caminhos que partem em p e chegam em q.

5 Relação de κ-conexidade

Definimos que um pixel q é **κ-conexo** a um pixel p, se existir um caminho ótimo π de p a q tal que $f(\pi) \leq \kappa$.

Um **componente κ-conexo** na imagem é um subconjunto de D_I, onde todos os pares (p, q) de pixels são κ-conexos.

Uma **partição κ-conexa** da imagem é um conjunto de componentes κ-conexos e disjuntos, cuja união é o conjunto D_I.

Note que as partições obtidas na rotulação hierárquica são κ-conexas, para $\kappa = T$, com relação à $f_{\max}$ (Equação 2), onde $h(q) = 0$.

6 Exercícios

1. Implemente o algoritmo de rotulação por conjuntos disjuntos e compare os resultados com o algoritmo da aula anterior usando a mesma função de dissimilaridades.

2. Implemente o algoritmo de rotulação hierárquica para a melhor função de dissimilaridade $\delta(p, q)$ encontrada no exercício da aula anterior. Compare os resultados obtidos com os dois algoritmos.

3. Supondo que sabemos o valor de T *a priori*, modifique o algoritmo que gera a árvore de peso mínimo para gerar a imagem rotulada $\hat{L}$ durante sua execução.