Processamento de Imagens usando Grafos

Prof. Alexandre Xavier Falcão

Segundo semestre de 2004

1 Componente κ-conexo a um conjunto de sementes

Um **componente κ-conexo** a um conjunto $S \subset D_I$ de pixels sementes é um subconjunto de D_I, onde todos os pixels p são κ-conexos a pelo menos uma semente $s \in S$. Note que esta definição corresponde à união de todas as árvores de caminhos de custo mínimo com raízes em S, onde os custos dos caminhos são menores ou iguais a κ.
Algoritmo de extração de componente κ-conexo com S:

Entrada: Imagem $\hat{I} = (D_I, I)$, adjacência A, sementes S, função f de custo de caminho e limiar κ.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde $L(p) = 1$ se p é κ-conexo a S, e $L(p) = 0$ no caso contrário. Inicialmente $L(p) = 0$ para todo $p \in D_I$
Auxiliares: Fila de prioridades Q, Imagem de custos $\hat{C} = (D_I, C)$, Imagem de predecessores $\hat{P} = (D_I, P)$ e variável tmp.

1. Para todo pixel $s \in S$, faça
2. $\quad C(p) \leftarrow +\infty$ para todo $p \in D_I$.
3. $\quad C(s) \leftarrow f(\langle s \rangle)$, $P(s) \leftarrow nil$ e insira s em Q.
4. $\quad\quad$ Enquanto $Q \neq \emptyset$, faça
5. $\quad\quad\quad$ Remova p de Q tal que $C(p)$ seja mínimo e faça $L(p) \leftarrow 1$.
6. $\quad\quad\quad$ Para todo $q \in A(p)$ tal que $C(q) > C(p)$, faça
7. $\quad\quad\quad\quad$ $tmp \leftarrow f(P^*(p) \odot \langle p, q \rangle)$.
8. $\quad\quad\quad\quad$ Se $tmp < C(q)$ e $tmp \leq \kappa$ então
9. $\quad\quad\quad\quad\quad$ Se $C(q) \neq +\infty$, remova q de Q.
10. $\quad\quad\quad\quad\quad$ $C(q) \leftarrow tmp$, $P(q) \leftarrow p$, e insira q em Q.

1.1 Aplicações

Uma aplicação para o algoritmo acima é a segmentação de imagens, onde um objeto é definido como um componente κ-conexo com sementes selecionadas no seu interior. Neste caso, podemos usar $f_{\max}$ ou f_{sum} com $h(q) = 0$ para $q \in S$, e $h(q) = +\infty$ no caso contrário. A dissimilaridade $\delta(p,q)$ pode levar em conta dois aspectos importantes:

1. A homogeneidade local no interior do objeto,

$$\delta_1(p,q) = 1 - \exp^{-\frac{(I(p)-I(q))^2}{2\sigma_1^2}}, \tag{1}$$

   onde σ_1 é uma constante.

2. A afinidade entre (p,q) e o objeto representado por S.

$$\begin{aligned} \delta_2(p,q) &= 1 - \exp^{-\frac{(\frac{I(p)+I(q)}{2}-\mu_2)^2}{2\sigma_2^2}}, && (2) \\ \delta_3(p,q) &= 1 - \exp^{-\frac{(\frac{I(p)+I(q)}{2}-I(s))^2}{2\sigma_s^2}}, && (3) \\ \delta_4(p,q) &= \min_{\forall s \in S}\{\delta_3(p,q)\}, && (4) \end{aligned}$$

onde μ_2 é a média de brilho em S, σ_2 é proporcional ao desvio padrão do brilho dos pixels em S, σ_s é proporcional ao desvio padrão do brilho dos pixels em uma vizinhança de $s \in S$, e s é o pixel inicial do caminho ótimo com término em p. Note que podemos usar também uma combinação ponderada dessas funções de dissimilaridade e um valor κ_s diferente para cada semente $s \in S$.

Outra aplicação é a dilatação de um conjunto S de pixels (e.g. o contorno de um objeto) por um disco de raio κ. Na verdade, existem algoritmos mais eficientes para se fazer isso. Porém, vamos nos ater a formulação do problema por enquanto. Seja $\hat{I}$ uma imagem binária, tal que $I(p) = 1$ se $p \in S$, e $I(p) = 0$ no caso contrário. Neste caso, $\hat{L}$ representa o resultado da dilatação para função de custo f_{euc}:

$$\begin{aligned} f_{euc}(\langle q \rangle) &= \begin{cases} 0, & \text{se } q \in S \\ +\infty, & \text{no caso contrário} \end{cases} \\ f_{euc}(\pi \cdot \langle p,q \rangle) &= d_{euc}(org(\pi), q), && (5) \end{aligned}$$

onde d_{euc} é a distância Euclideana entre dois pontos.

1.2 Eficiência

A eficiência do algoritmo depende das cardinalidades de A e de S, da implementação de Q, e da implementação de f. A fila Q pode ser uma estrutura *heap* balanceada ou uma fila circular baseada em *bucket sort*, conforme discutiremos mais adiante. No caso do *bucket sort*, todas as funções de custo devem ser expressas com valores inteiros, e considerando $|S| \ll |D_I|$ e

$|A| \ll |D_I|$, o algoritmo pode ser executado em tempo proporcional ao número de pixels. Um valor inteiro máximo $K_{\max}$ pode ser usado para $f_{\max}$ e f_{sum}, tal que a linha 7 do algoritmo fica modificada para: $tmp \leftarrow \max\{C(p), K_{\max}\delta_i(p,q)\}$, $i = 1,2,3,4$, no caso de $f_{\max}$, e $tmp \leftarrow C(p) + K_{\max}\delta_i(p,q)$, no caso de f_{sum}. No caso de f_{euc}, a linha 7 fica $tmp \leftarrow d^2_{euc}(s,q)$. Note que em todos os casos não precisamos da imagem de predecessores.

Uma questão importante é se podemos ou não propagar as árvores das sementes ao mesmo tempo, tornando a eficiência do algoritmo independente da cardinalidade de S. Este variante pode ser implementado trocando-se as linhas 1,2, e 3 do algoritmo por

1. Para todo pixel $p \in D_I$, faça $C(p) \leftarrow +\infty$,

2. Para todo pixel $s \in S$, faça $C(s) \leftarrow 0$, $P(s) \leftarrow nil$, $R(s) \leftarrow s$, e insira s em Q;

e acrescentando-se na linha 10 a propagação das raizes $R(q) \leftarrow R(p)$ (pixels iniciais dos caminhos), que serão usadas no cálculo de f_{euc} e de f_{max} e f_{sum} com δ_3 ou δ_4.

Uma ressalva, porém, é que a propagação de caminhos com competição entre sementes pode levar o algoritmo a decidir por caminhos não-ótimos, pois a árvore de uma semente s_1 pode bloquear a propagação de outra s_2, impedindo s_2 de atingir pixels mais conexos com s_2 do que com s_1. Considere, por exemplo, $C(p) = 0.10$ e $\delta_3(p,q) = 0.15$ para o caminho ótimo que atinge p vindo de s_1, e $C(p) = 0.14$ e $\delta_3(p,q) = 0.10$ para o caminho ótimo que atinge q vindo de s_2. Se o pixel q for tal que a única forma de atingí-lo é via $\langle p,q\rangle$, então o algoritmo com propagação simultânea vai escolher erroneamente o caminho de s_1 a q via p em $f_{\max}$ e f_{sum}. Exemplo similar pode ser obtido com f_{euc} e três sementes. Isto nos leva a considerar condições que garantam a corretude desses algoritmos.

1.3 Corretude

Os algoritmos discutidos acima são uma adaptação do algoritmo de Dijkstra. Obviamente estamos assumindo que $f(\langle s\rangle) \leq \kappa$ para todo $s \in S$ e que f satisfaz condições que guarantam a corretude do algoritmo de Dijkstra (i.e. $P^*(p)$ deve ser ótimo quando p sai de Q). Isto é, para todo pixel $q \in D_I$, deve existir um caminho ótimo π terminando em q tal que, ou $\pi = \langle q\rangle$ é trivial, ou tem a forma $\tau \cdot \langle p,q\rangle$ onde

(C1) $f(\tau) \leq f(\pi)$,

(C2) τ é ótimo,

(C3) Para qualquer caminho ótimo τ' terminando em p, $f(\tau' \cdot \langle p,q\rangle) = f(\pi)$.

Note que essas condições são necessárias apenas para caminhos ótimos. Funções que satisfazem a essas condições são chamadas **suaves**. Portanto, essas condições são satisfeitas pelas funções $f_{\max}$ e f_{sum} com $\delta(p,q)$ fixo para toda aresta (p,q) (e não-negativo no caso de f_{sum}), e de uma forma geral, por funções monotônicas-incrementais (**MI**):

$$\begin{aligned} f(\langle q\rangle) &= h(q), \\ f(\pi \cdot \langle p,q\rangle) &= f(\pi) \odot (p,q), \end{aligned} \tag{6}$$

onde $h(q)$ é arbitrário e $\odot : V \times A \rightarrow V$ é uma operação binária que satisfaz as condições

(M1) $x' \geq x \Rightarrow x' \odot (p,q) \geq x \odot (p,q)$,

(M2) $x \odot (p,q) \geq x$,

para quaisquer $x, x' \in V$ e qualquer $(p,q) \in A$, onde $\odot$ depende apenas do custo de π e não de qualquer outra propriedade de π.

Note que, muito embora a função f_{euc} não seja MI, o algoritmo com propagação simultânea é ótimo para $|S| \leq 2$, e para $|S| > 2$, sua corretude dependerá de A.

2 Exercícios

1. Prove a corretude do algoritmo de Dijkstra para as condições (C1)–(C3).

2. Implemente o algoritmo acima e compare os seus resultados para diferentes funções de custo de caminho e relações de adjacência, considerando um conjunto S fixo de sementes.