Processamento de Imagens usando Grafos

Prof. Alexandre Xavier Falcão

Segundo semestre de 2004

1 Partição ótima baseada em sementes

Vimos que quando as sementes em S são propagadas ao mesmo tempo, temos um processo competitivo onde cada semente define uma zona de influência composta por pixels mais conexos com esta semente do que com qualquer outra em S. Se eliminarmos a restrição de κ-conexidade, deixando as sementes propagarem para todos os pixels da imagem, a partição resultante será uma floresta de caminhos de custo mínimo, visto que f deve ser suave. A **Transformada Imagem-Floresta** (IFT-*Image Foresting Transform*) explora esta formulação para resolver problemas de processamento de imagens baseados em conexidade. Essencialmente, a IFT reduz o problema ao cálculo de uma floresta de caminhos de custo mínimo, seguido de uma operação local sobre atributos da floresta: os caminhos ótimos, os custos desses caminhos e as raízes da floresta junto com suas zonas de influência. Esses atributos podem ser calculados nas imagens de predecessores, custos, e raízes. Note que uma semente s pode não se transformar em raiz, caso exista outra semente $s' \in S$ cujo caminho ótimo até s tenha custo menor que $f(\langle s \rangle)$. Neste sentido dizemos que as sementes são apenas candidatas a raízes. Em alguns operadores, por exemplo, $S = D_I$. Quando $S \subset D_I$, $f(\langle s \rangle)$ é normalmente finita se $s \in S$ e infinita no caso contrário. Dizemos então que f^S é a função f restrita ao conjunto de sementes S. Note, porém, que se f é suave, f^S pode não ser. Para facilitar, podemos embutir o conjunto S na definição de f e apresentar o algoritmo da seguinte forma.

Algoritmo geral da IFT:

Entrada: Imagem $\hat{I} = (D_I, I)$, adjacência A, e função f de custo de caminho.
Saída: Imagens $\hat{C} = (D_I, C)$ de custo, $\hat{P} = (D_I, P)$ de predecessores, e $\hat{R} = (D_I, R)$ de raízes.
Auxiliares: Fila Q de prioridades e variável tmp.

1. Para todo pixel $p \in D_I$ faça
2. $\quad C(p) \leftarrow f(\langle p \rangle)$, $P(p) \leftarrow nil$, e $R(p) \leftarrow p$.
3. $\quad$ Se $C(p) < +\infty$, insira p em Q.

4. Enquanto $Q \neq \emptyset$ faça
5. Remova um pixel p de Q cujo custo $C(p)$ é mínimo.
6. Para todo $q \in A(p)$, tal que $C(q) > C(p)$, faça
7. $tmp \leftarrow f(P^*(p) \cdot \langle p, q \rangle)$.
8. Se $tmp < C(q)$ faça
9. Se $C(q) \neq +\infty$, remova q de Q.
10. $C(q) \leftarrow tmp$, $P(q) \leftarrow p$, $R(q) \leftarrow R(p)$, e insira q em Q.

As aplicações deste algoritmo incluem diversos operadores de filtragem, segmentação, e análise de imagens: transformadas de *watershed*, perseguição de bordas, caminhos geodésicos, transformadas de distância, esqueletos multiescala, dimensão fractal multiescala, saliências de formas, reconstruções morfológicas e outras operações conexas.

1.1 Resolvendo empates

Um dos principais problemas da IFT é estabelecer uma regra de desempate, quando a força de conexidade de um pixel é a mesma com relação a duas ou mais sementes. Uma regra normalmente adotada é o desempate FIFO (*First-In-First-Out*) para extrair pixels p com mesmo custo mínimo em Q. Outra forma, com vantagens em algumas aplicações, é o desempate LIFO (*Last-In-First-Out*). Neste caso, o algoritmo deve ser implementado da segunte forma.

Algoritmo da IFT com desempate LIFO:

Entrada: Imagem $\hat{I} = (D_I, I)$, adjacência A, e função f de custo de caminho.
Saída: Imagens $\hat{C} = (D_I, C)$ de custo, $\hat{P} = (D_I, P)$ de predecessores, e $\hat{R} = (D_I, R)$ de raízes.
Auxiliares: Fila Q de prioridade e variável tmp.

1. Para todo pixel $p \in D_I$ faça
2. $C(p) \leftarrow f(\langle p \rangle)$, $P(p) \leftarrow nil$, $R(p) \leftarrow p$, e insira p em Q.
3. Enquanto $Q \neq \emptyset$ faça
4. Remova um pixel p de Q cujo custo $C(p)$ é mínimo.
5. Para todo $q \in A(p)$, tal que $q \in Q$, faça
6. $tmp \leftarrow f(P^*(p) \cdot \langle p, q \rangle)$.
7. Se $tmp \leq C(q)$ então
8. Remova q de Q, faça $C(q) \leftarrow tmp$, $P(q) \leftarrow p$, e $R(q) \leftarrow R(p)$, e insira q em Q.

A principal diferença entre este último algoritmo e o anterior está na linha 8, onde $P(q)$ e $R(q)$ devem ser atualizados, e q deve ser removido e reinserido em Q, mesmo quando tmp é igual a $C(q)$. Com a política FIFO, qualquer conjunto conexo de pixels que pode ser encontrado por duas ou mais raízes através de caminhos ótimos de mesmo custo será particionado entre as respectivas árvores. No caso da poítica LIFO, esses pixels são associados a uma mesma árvore. Infelizmente, ambas não resolvem por completo o problema de ambigüidade das florestas ótimas e regras extras de desempate devem ser implementadas em Q, ou podemos ainda definir f como uma função de custo lexicográfica.

1.2 Variantes

Diversos variantes desses algoritmos podem ser adotados visando uma maior eficiência para determinadas operações de imagem. Os casos mais simples são a propagação simultânea de rótulos, a saturação da função de custo, e a busca por caminhos específicos.

No caso de funções suaves f^S restritas a um conjunto S de pixels sementes, onde cada semente p possui um rótulo $\lambda(p)$, podemos gerar uma imagem de rótulos $\hat{L} = (D_I, L)$ fazendo $L(p) \leftarrow \lambda(p)$ na linha 2 de ambos algoritmos, e $L(q) \leftarrow L(p)$ na linha 10 do primeiro e na linha 8 do segundo. Este variante é muito comum em segmentação de imagens, quando desejamos atribuir um rótulo distinto para cada objeto (incluindo o fundo).

No caso de estarmos interessados em podar as árvores ficando com os nós de custo menor ou igual a um dado limiar, podemos evitar o cálculo da floresta completa fazendo com que a função f retorne $+\infty$ na linha 7 do algoritmo geral. Este variante é útil, por exemplo, em métodos de segmentação de imagens por crescimento de regiões e no cálculo de caminhos geodésicos.

Quando desejamos encontrar um caminho ótimo que chega a um dado pixel (ou a um conjunto de pixels), podemos parar o cálculo quando este pixel (ou o primeiro pixel do conjunto) sai da fila na linha 5 do algoritmo geral. Este variante é útil no cálculo de caminhos geodésicos entre conjuntos de pixels e em algoritmos de perseguição de borda.

Alguns variantes permitem simplificações do algoritmo geral, e neste curso veremos essas simplificações para várias situações específicas.

1.3 Fila de prioridade

A implementação mais fácil para a fila Q usa um *heap* binário. Neste caso os algoritmos acima terão complexidade $O(m + n \log n)$, onde $n = |D_I|$ é o número de nós (pixels) e $m = |A|$ o número de arcos.

Na maioria das aplicações, porém, podemos usar funções de custo de caminho com incrementos de custo inteiros e limitados a uma constante K ao longo do caminho. Isto permite a utilização da fila circular de Dial com $K + 1$ posições. Cada posição i, $i = 0, 1, \ldots, K$, deve armazenar uma lista duplamente ligada de todos os pixels p com custo $i = C(p)\%K$. Como sabemos o tamanho máximo $|D_I|$ do grafo, essas listas podem ser implementadas em uma única matriz A de ponteiros $A.next(p)$ e $A.prev(p)$ com $|D_I|$ elementos. Neste caso, os

algoritmos da IFT terão complexidade $O(m+nK)$. Se a adjacência A definir um grafo esparso $m \ll n$, a IFT levará tempo proporcional ao número $n = |D_I|$ de pixels.

Note que a cada instante existe um valor mínimo $C_{\min}$ e um valor máximo $C_{\max}$ de custo para os pixels armazenados em Q. A diferença $C_{\max} - C_{\min} \leq K$ deve ser mantida para garantir a corretude da fila. Em algumas aplicações sabemos que os incrementos são inteiros e limitados, mas não conhecemos o valor de K. Neste caso, a fila circular inicia com um dado tamanho K, mas antes de inserir um novo pixel devemos verificar a necessidade de realocar ou não mais elementos para a fila.

O código em C com os algoritmos da IFT, FIFO e LIFO, a implementação da fila Q com realocação dinâmica, e alguns exemplos de operadores de imagem estão disponíveis em www.ic.unicamp.br/~afalcao/ift.html.

2 Exercícios

1. Implemente a IFT-FIFO com um *heap* binário e teste seu algoritmo com algumas funções de custo de caminho suaves.

2. Calcule o maior incremento K para a função de custo de caminho $f_{euc}(\pi \cdot \langle p, q \rangle) = d^2(org(\pi), q)$, onde d é a distância Euclideana e $org(\pi)$ é o pixel inicial do caminho π.