Introdução ao Processamento de Imagem Digital (MO443/MC920)

Prof. Alexandre Xavier Falcão

Segundo semestre de 2003

1 Algoritmos e estruturas de dados para a IFT

O algoritmo geral da IFT é essencialmente o Algoritmo de Dijkstra estendido para múltiplas fontes e funções de custo de caminho suaves. Note que podem existir várias florestas P de caminhos de custo mínimo que satisfazem um dado problema, apenas o mapa C de custos ótimos é único. Esta ambigüidade é parcialmente resolvida quando decidimos pelo caminho de menor custo que encontra um dado pixel primeiro. O algoritmo resultante é apresentado abaixo.

Algoritmo geral da IFT:
Entrada: Imagem $\hat{I} = (D_I, I)$, relação de adjacência A, e função f de custo de caminho suave.
Saída: Imagens $\hat{C} = (D_I, C)$ de custo, $\hat{P} = (D_I, P)$ de predecessores, e $\hat{L} = (D_I, L)$ de raízes.
Auxiliares: Fila Q de prioridade e variável c.

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

Aplicando a regra de desempate acima, inclusive para pixels distintos que são encontrados por caminhos ótimos de mesmo custo, aquele que entrou na fila Q primeiro é o primeiro a sair. Neste caso dizemos que a fila Q segue a política *First-In-First-Out* (FIFO) de desempate. Outra forma de resolver parcialmente o problema de ambigüidade das florestas ótimas é implementar a fila Q com política *Last-In-First-Out* (LIFO) de desempate. Este variante é apresentado abaixo.

Algoritmo da IFT com política de desempate LIFO:
Entrada: Imagem $\hat{I} = (D_I, I)$, relação de adjacência A, e função f de custo de caminho suave.
Saída: Imagens $\hat{C} = (D_I, C)$ de custo, $\hat{P} = (D_I, P)$ de predecessores, e $\hat{L} = (D_I, L)$ de raízes.
Auxiliares: Fila Q de prioridade e variável c.

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

A principal diferença entre este último algoritmo e o anterior está na linha 8, onde $P(q)$ e $L(q)$ devem ser atualizados, e q deve ser removido e reinserido em Q, mesmo quando c é igual a $C(q)$. Com a política FIFO, qualquer conjunto conexo de pixels que poderiam ser encontrados por duas ou mais raízes por caminhos ótimos de mesmo custo serão particionados entre as respectivas árvores. No caso da poítica LIFO, esses pixels são associados a uma mesma árvore. A política LIFO é útil em algumas situações, como no cálculo do número de mínimos regionais de uma imagem, mas a política FIFO satisfaz melhor as expectativas do usuário com relação à partição da imagem (e.g. na segmentação). 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.

1.1 Alguns 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 usar $L(p) \leftarrow \lambda(p)$ na linha 2 em ambos algoritmos

e eles propagarão um mapa L de rótulos em vez de raízes. 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 pode ser 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.

1.2 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.