Introdução ao Processamento de Imagem Digital (MO443/MC920)

Prof. Alexandre Xavier Falcão

Segundo semestre de 2003

1 Perseguição de bordas

Quando examinamos a intensidade dos pixels em torno da borda de um objeto em uma imagem cinza percebemos que existe uma incerteza com relação à posição exata da borda (abordagem *hard*). Poderíamos transferir nossa incerteza para um conjunto de pixels que forma uma faixa de largura variável em torno da borda e usar este conjunto para representá-la (abordagem *fuzzy*). Considerando k conjuntos S_i, $i = 1, 2, \ldots, k$, de pixels sementes selecionados sobre uma borda, tais que $S_1 = S_k$ possui um único pixel e os pixels em S_i, $i = 2, 3, \ldots, k-1$, pertencem a uma região pequena de incerteza (e.g. uma linha que cruza a borda, uma marca circular sobre a borda), poderíamos escolher por exemplo os n caminhos de menor custo de S_i para S_{i+1}, $i = 1, 2, \ldots, k-1$, (i.e. os n caminhos mais conexos) como parte de um segmento ótimo de borda *fuzzy*, ou um único caminho de menor custo como segmento ótimo de borda *hard*. Observe que a IFT permite ambas abordagens, mas vamos tratar aqui apenas o caso de contornos ótimos.

Uma borda é um contorno ótimo que passa pela seqüência de conjuntos S_i, $i = 1, 2, \ldots, k$, de pixels sementes. O cálculo de caminhos de custo mínimo de S_i até S_{i+1}, $i = 1, 2, \ldots, k-1$, pode ser feito com uma IFT usando adjacência-4 (ou 8) e função de custo de caminho f_{ctrack}.

$$\begin{aligned} f_{ctrack}(\langle q \rangle) &= h(q) \\ f_{ctrack}(\pi \cdot \langle p, q \rangle) &= f_{ctrack}(\pi) + (K - \max\{G(p,q) \cdot \eta(p,q), 0\}), \end{aligned} \tag{1}$$

onde $h(q) = 0$, se $q \in S_1$ na primeira iteração, ou $h(q)$ é o custo final do pixel q na iteração anterior, se $q \in S_i$ e $i > 1$, ou $h(q) = +\infty$ no caso contrário, independente da iteração; $G(p,q)$ é um vetor gradiente estimado no ponto médio do arco (p,q); $\eta(p,q)$ é o arco (p,q) rotacionado de 90 graus no sentido anti-horário; e K é um limite superior para $|G(p,q) \cdot \eta(p,q)|$. O vetor $G(p,q)$ deve ser tal que arcos sobre a borda orientada do objeto possuam valores baixos de custo e os demais valores altos.

Note que a cada iteração, o algoritmo pode parar o cálculo da IFT quando o último pixel de S_{i+1} sai da fila Q. O contorno final pode ser obtido dos respectivos $k-1$ mapas de predecessores $P_{k-1}, \ldots, P_1$, mas também é possível modificar o algoritmo da IFT para usar uma única imagem anotada e um mapa de predecessores auxiliar em todas iterações.

1.1 Aplicações

O método *live-wire* é um exemplo que utiliza esta abordagem de forma interativa. Para segmentar uma borda, o usuário seleciona a primeira semente (pixel em S_1) e o método calcula uma IFT em toda imagem, cuja única árvore terá como raiz a semente selecionada. Para cada posição subseqüente do cursor, o algoritmo mostra na tela o caminho de custo mínimo da raíz até esta posição. O usuário pode mover livremente o cursor sobre a imagem e verificar os caminhos ótimos. Quando o cursor se aproxima da borda desejada, este caminho gruda na borda e o usuário então seleciona a posição atual do cursor (pixel em S_2) para confirmar o segmento ótimo de borda. O processo se repete a partir do pixel selecionado até o usuário decidir fechar o contorno.

O algoritmo da IFT também pode ser modificado para ser executado de forma incremental. O método live-wire-on-the-fly usa este variante. Neste caso nós exploramos três propriedades do algoritmo de Dijkstra.

1. O algoritmo pode parar o cálculo quando o pixel q que define a posição do cursor sai da fila Q.

2. Neste instante, todos os caminhos com custo menor que $C(q)$ já foram calculados, definindo uma região de crescimento em torno da raíz (árvore de caminhos ótimos). Então, se o usuário mover o cursor para qualquer outro pixel desta região, basta mostrar o caminho ótimo correspondente na tela.

3. Quando o pixel q sai da fila Q, a fronteira da região de crescimento que ainda está na fila Q deve ser armazenada junto com os valores de custo e predecessor de cada pixel. Se o usuário move o cursor para fora da região, então o cálculo continua a partir dos pixels de fronteira até encontrar a nova posição do cursor.

O efeito desta otimização é que após selecionar um pixel sobre a borda, o usuário movimenta o cursor e já visualiza o caminho ótimo da semente até a posição atual do cursor, mesmo em imagens grandes.

Já no método 3D-live-wire para seqüências de imagens tomográficas (fatias), o usuário executa o live-wire em cortes ortogonais ao volume criado pelo empinhamento das fatias. Cada live-wire ortogonal gera pelo menos dois pixels sobre a borda do objeto nas fatias de forma tal que obtemos uma seqüência ordenada de sementes ($|S_i| = 1$, $i = 1, 2, \ldots, k$) sobre a borda em cada fatia. Em seguida, os contornos ótimos são calculados automaticamente fatia por fatia. O método é implementado com algumas restrições para tratar mudanças de topologia ao longo das fatias.

2 Exercícios

1. Apresente outras funções de custo de caminho para perseguição de bordas.

2. Implemente uma função para calcular o contorno ótimo que passa por uma seqüência de conjuntos de pixels sementes $\{S_1, S_2, \ldots, S_k\}$, $S_1 = S_k$ possui apenas um único pixel, usando uma única imagem anotada e um mapa de predecessores auxiliar para ir copiando os caminhos obtidos em cada iteração S_i para S_{i+1}, $i = 1, 2, \ldots, k-1$.