# 3
# Estéreo e Cortes de Grafo

Muitos problemas de visão computacional podem ser naturalmente formulados como um problema de minimização de energia a fim de solucionar o problema de atribuição de rótulos (*labels*). Especificamente o problema de estéreo pode ser visto como um problema de atribuição de rótulos onde temos um conjunto de $n$ pixels $\mathcal{P}$ e de $k$ disparidades (rótulos) $\mathcal{L}$. O problema consiste em encontrar uma atribuição de rótulos $f$, ou seja, um mapeamento de $\mathcal{P}$ em $\mathcal{L}$, que minimize alguma função de energia. Uma forma típica de função de energia é

$$\begin{aligned} E(f) &= E_{data}(f) + E_{smooth}(f) \\ &= \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{p,q \in \mathcal{N}} V_{p,q}(f_p, f_q) \end{aligned} \tag{3-1}$$

onde $\mathcal{N} \subset \mathcal{P} \times \mathcal{P}$ é uma vizinhança de pixels do pixel $p$. $D_p(f_p)$ é uma função que mede o custo de se atribuir o rótulo $f_p$ a $p$. $V_{p,q}(f_p, f_q)$ define o custo de se atribuir os rótulos $f_p, f_q$ aos pixels adjacentes $p$ e $q$ e é usado para penalizar mudanças de disparidade nas regiões de descontinuidade. Nas bordas dos objetos pixels adjacentes em geral têm rótulos muito diferentes e é importante que $E$ não super penalize esses rótulos. Dessa forma é necessário que $V$ seja uma função não convexa de $|f_p - f_q|$.

Funções de energia como a da equação (3-1) são extremamente difíceis de se minimizar, já que são funções não convexas num espaço com milhares de dimensões. Tradicionalmente essas funções são minimizadas aplicando técnicas de minimização genéricas (como "simulated annealing") que são capazes de minimizar qualquer função de energia. Como consequência da generalidade, essas técnicas possuem custo exponencial e são muito lentas, tornando seu uso inviável na prática.

Nos últimos anos vários algoritmos eficientes baseados em Cortes de Grafo têm sido desenvolvidos para solucionar o problema de minimização de energia. Neste capítulo vamos discutir o método de cortes de grafo e sua utilização para resolver problemas de minimização de energia, abordando o

algoritmo de expansão-$\alpha$ e troca-$\alpha\beta$. Descreveremos também como o problema de estéreo pode ser mapeado em um problema de minimização de energia, e consequentemente, num problema de atribuição de rótulos, e assim mostrar como resolvê-lo utilizando cortes de grafo.

## 3.1 Cortes de Grafo

A técnica de otimização discreta por Cortes de Grafo é utilizada para resolver problemas de minimização de energia. Trata-se de utilizar algoritmos específicos para a construção de um grafo de maneira que, ao calcular o corte mínimo neste grafo, estamos também minimizando a energia do sistema. É especialmente útil quando aplicada a problemas que podem ser modelados como um problema de minimização de energia, e.g. o problema de correspondência por estéreo.

Seja $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ um grafo direcionado com arestas de peso não negativos que possui dois vértices especiais (terminais) chamados $s$ (source) e $t$ (sink). Um corte-$s$-$t$, ao qual chamaremos simplesmente de corte, $C = S, T$ é uma partição dos vértices em $\mathcal{V}$ em dois conjuntos disjuntos $S$ e $T$ tais que $s \in S$ e $t \in T$. A Figura 3.1 ilustra à esquerda o grafo $\mathcal{G}$ e à direita, em verde, o que seria um corte em $\mathcal{G}$. O custo total do corte é a soma dos pesos de todas as arestas que ligam os conjuntos distintos $S$ e $T$ e é definido como:

$$c(S,T) = \sum_{u \in S, v \in T, (u,v) \in \mathcal{E}} c(u,v),$$

onde $c(u,v)$ é o custo da aresta que liga o nó $u$ ao nó $v$. O problema do corte-$s$-$t$ mínimo consiste em encontrar um corte $C \subset \mathcal{E}$ com o menor custo. Ford e

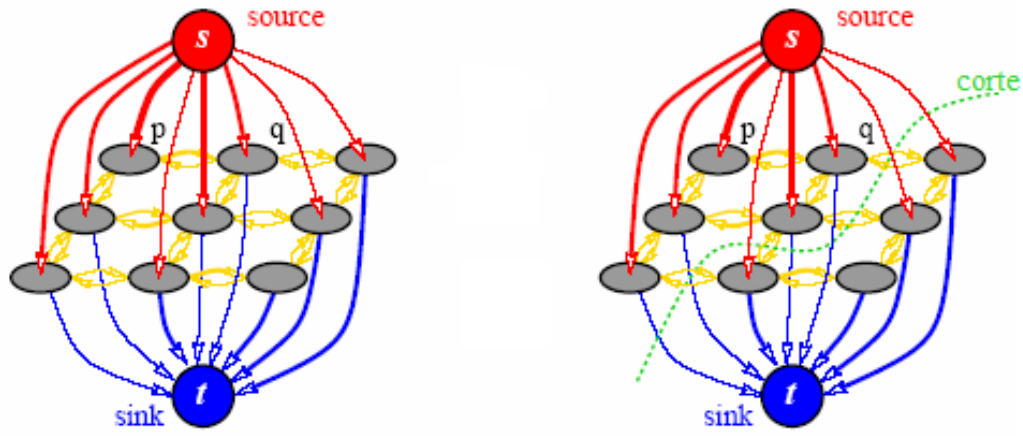


Figura 3.1: Grafo (à esquerda) com 9 pixels a serem classificados e corte em um grafo (à direita). (Reproduzida de (05)).

Fulkerson mostram em (06) que encontrar esse corte é equivalente a encontrar o fluxo máximo do nó terminal $s$ ao nó terminal $t$. Existem vários algoritmos que resolvem o problema do fluxo máximo em tempo polinomial. No nosso trabalho vamos utilizar o algoritmo para fluxo máximo otimizado apresentado por Boykov e Kolmogorov em (03).

Um corte $C = (S, T)$ é uma atribuição de rótulos $f$ que mapeia o conjunto de vértices $\mathcal{V}-\{s, t\}$ em {0,1}, onde $f(v) = 0$ implica em $v \in S$ e $f(v) = 1$ implica em $v \in T$. Ou seja, um corte é uma partição binária de um grafo e que define uma atribuição de rótulos bi-valorada. Existem algumas propostas para resolver o problema do corte mínimo com mais de dois terminais mas todas elas são NP-difíceis.

Para minimizar $E$ usando o método de cortes de grafo é necessário criar um grafo específico de maneira que o corte mínimo no grafo também minimize $E$. A forma do grafo depende da forma exata de $V$ (termo de suavidade), que determina o tipo de algoritmo que pode ou deve ser utilizado, e do número de rótulos a serem atribuídos. Os nós do grafo representam os pixels da imagem enquanto os terminais representam todos os rótulos possíveis de serem atribuídos. Os pesos das arestas entre nós não-terminais (pixels) são definidos pelo termo de suavidade $E_{smooth}(f)$, enquanto os pesos entre nós terminais e não-terminais são definidos pelo termo de dados $E_{data}(f)$.

Como no problema abordado nessa dissertação temos um grafo com vários nós terminais, precisamos dividir o grafo em vários grafos com dois terminais e depois determinar qual configuração tem menor energia para cada uma dessas divisões iterativamente. Uma vez que cada pixel está associado a apenas um terminal, atribuímos esse rótulo (ou disparidade) àquele pixel.

Para o problema de estéreo os rótulos são disparidades e o termo de dados $D_p(f_p)$ pode ser definido como uma função da diferença de intensidade entre o pixel $p$ na imagem de referência e o pixel $p + f_p$ na outra imagem. Nesse tipo de problema especificamente o algoritmo de expansão-$\alpha$ e outros algoritmos de minimização de energia apresentam, segundo Kolmogorov e Zabih (09), resultados muito bons (ainda que não viabilizem a sua execução em tempo real).

Na próxima seção vamos discutir duas possíveis abordagens definidas como movimentos que podem ser utilizadas no algoritmo de cortes de grafo de maneira a mudar a atribuição de rótulos, visando a minimização de energia.

## 3.2
## Algoritmos de Expansão e Troca

Boykov (04) aborda o problema de minimização de energia com cortes de grafo como a determinação de um mínimo local e apresenta dois algoritmos aproximativos baseados em dois tipos diferentes de movimento: o movimento de expansão-$\alpha$ e o movimento de troca-$\alpha\beta$. Boykov prova que o algoritmo de expansão-$\alpha$ produz resultados que distam do mínimo global até um fator multiplicativo conhecido, desde que o termo de suavidade $V$ seja uma métrica (veja seção 3.2.2). Esse fator é no mínimo 2 e depende unicamente de $V$. Por outro lado o algoritmo de troca-$\alpha\beta$ não exige que $V$ seja uma métrica e dessa forma, em geral, é utilizado com funções de energia mais genéricas.

A Figura 3.2 ilustra os possíveis movimentos que podem ser realizados num algoritmo de cortes de grafo. A figura da esquerda mostra a atribuição de rótulos inicial. A figura do meio mostra a nova configuração dos rótulos após um movimento de troca-$\alpha\beta$ e a figura da direita mostra a nova configuração dos rótulos após um movimento de expansão-$\alpha$.

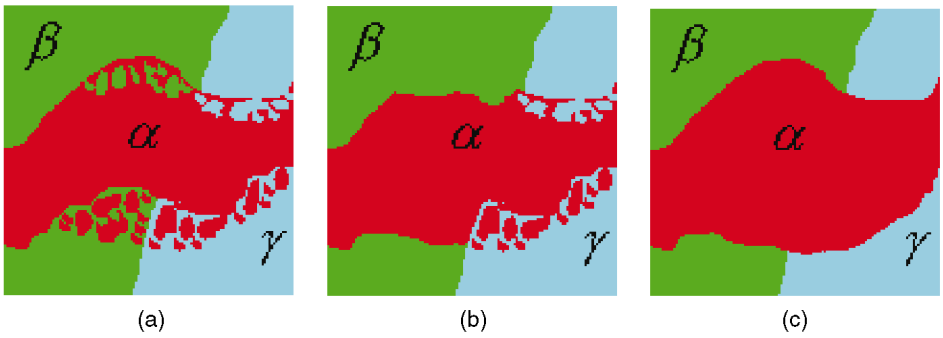


Figura 3.2: Tipos de movimento possíveis para uma atribuição de rótulos (a) com três rotulos possíveis: vermelho, verde e azul. De (a) para (b) temos uma troca-$\alpha\beta$ e de (a) para (c) temos uma expansão-$\alpha$. (Extraída de (04)).

Ambos os algoritmos são muito semelhantes na sua estrutura. O algoritmo de troca encontra um mínimo local quando somente movimentos de troca são permitidos e o algoritmo de expansão encontra um mínimo local quando somente movimentos de expansão são permitidos. Neste capítulo vamos nos concentrar nos detalhes do algoritmo de expansão-$\alpha$, pois é o algoritmo que utilizamos em nosso método. Daremos, entretanto, uma visão geral do algoritmo de troca.

Antes de apresentar mais detalhadamente os movimentos, vamos definir o conceito de partição, pois nos será útil ao definir os algoritmos. Qualquer atribuição de rótulos $f$ pode ser representada por uma partição de pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, onde $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ é um subconjunto de pixels aos quais o rótulo $l$ foi atribuído.

### 3.2.1 Algoritmo de Troca

Dado um par de rótulos $\alpha$ e $\beta$, um movimento de uma partição $\mathbf{P}$ (atribuição de rótulos $f$) para uma partição $\mathbf{P'}$ (atribuição de rótulos $f'$) é chamado de uma troca-$\alpha\beta$ se $\mathcal{P}_l = \mathcal{P'}_l$ para qualquer rótulo $l \neq \alpha, \beta$. Isto implica em que a única diferença entre $\mathbf{P}$ e $\mathbf{P'}$ é que alguns pixels que tinham o rótulo $\alpha$ em $\mathbf{P}$ agora tem rótulo $\beta$ em $\mathbf{P'}$, e alguns pixels que tinham rótulo $\beta$ em $\mathbf{P}$ agora tem rótulo $\alpha$ em $\mathbf{P'}$. Um exemplo de um movimento de troca-$\alpha\beta$ é mostrado na Figura 3.2b.

### 3.2.2 Algoritmo de Expansão

Um dos algoritmos mais eficientes para minimização de energia é o algoritmo de expansão-$\alpha$, que foi introduzido em (04). Esse algoritmo pode ser usado sempre que $V$ é uma métrica no espaço de rótulos, ou seja, $V$ satisfaz as três condições abaixo:

$$\begin{aligned} V(\alpha,\beta) = 0 \quad &\Leftrightarrow \quad \alpha = \beta, \\ V(\alpha,\beta) \quad &= \quad V(\beta,\alpha) \geq 0, \\ V(\alpha,\beta) \quad &\leq \quad V(\alpha,\gamma) + V(\gamma,\beta), \end{aligned} \tag{3-2}$$

para quaisquer rótulos $\alpha, \beta, \gamma \in \mathcal{L}$.

Para um determinado rótulo $\alpha$, um movimento de uma partição $\mathbf{P}$ (atribuição de rótulos $f$) para uma nova partição $\mathbf{P'}$ (atribuição de rótulos $f'$) é chamado de uma expansão-$\alpha$ se $\mathcal{P}_\alpha \subset \mathcal{P'}_\alpha$ e $\mathcal{P} \subset \mathcal{P'}$ para qualquer rótulo $l \neq \alpha$, ou seja, ${f'}_p \neq \alpha$ implica em ${f'}_p = f_p$. Em outras palavras, um movimento de expansão-$\alpha$ permite que qualquer conjunto de pixels da imagem alterem seus rótulos para $\alpha$. Um exemplo de um movimento de expansão-$\alpha$ é mostrado na Figura 3.2c.

Em cada ciclo, o algoritmo itera sobre todos os rótulos $\alpha$ em uma ordem que pode ser fixa ou randômica e procura, a partir da atribuição de rótulos atual, a expansão-$\alpha$ de menor energia. Um ciclo é bem sucedido se uma atribuição de rótulos melhor é encontrada em qualquer ciclo. Se nenhuma atribuição de rótulos em um mesmo ciclo apresentar energia menor o algoritmo encerra, já que não é possível minimizar ainda mais a energia. No final temos uma atribuição de rótulos que é um mínimo local da função de energia em relação a movimentos de expansão. Dessa forma um ciclo no algoritmo de expansão-$\alpha$ possui $|\mathcal{L}|$ iterações.

A técnica descrita aqui é baseada na computação de uma atribuição de rótulos correspondente ao corte mínimo num grafo $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$. A ideia por trás do algoritmo é que pixels são temporariamente vistos como tendo o rótulo $\alpha$ ou $\bar{\alpha}$. Após encontrar o corte mínimo no grafo, ou os pixels irão continuar com seus rótulos atuais ou alterarão seus rótulos para $\alpha$, dependendo do que proporcionar a menor quantidade de energia.

A estrutura do grafo em cada ciclo é determinada pela partição corrente $\mathbf{P}$ e pelo rótulo $\alpha$. O grafo muda dinamicamente após cada iteração. Boykov em (04) atribui pesos às arestas inserindo nós auxiliares entre pixels vizinhos contendo rótulos diferentes. Essa abordagem foi posteriormente aprimorada por Kolmogorov em (10) que reformulou as equações e fez com que esses nós auxiliares não fossem mais necessários. A construção de grafo utilizada para a implementação do nosso método foi a apresentada por Kolmogorov em (10), utilizando o tratamento para oclusões apresentado em (09).

## 3.3 Minimização de Energia, Corte Mínimo e Fluxo Máximo

Em (03) Boykov e Kolmogorov apresentam um novo algoritmo que aprimora o desempenho de técnicas padrão de determinação de caminhos de aumento em grafos (*augmenting path*). Algoritmos de caminhos de aumento anteriores usavam busca em largura quando todos os caminhos de uma determinada largura já haviam sido pesquisados, requerendo a construção de uma nova árvore de busca. Este processo envolve varrer a maioria dos pixels na imagem, tornando-se computacionalmente caro quando repetido várias vezes.

O novo algoritmo resolve este problema construindo duas árvores de busca, uma para o nó terminal $s$ (source) e outra para o nó terminal $t$ (sink), e reutilizando-as. Elas nunca são reconstruídas. O resultado é um algoritmo que é muito mais rápido em todas as aplicações onde grafos consistem em grids 2D. Entretanto existem desvantagens. Não há garantia que o caminho de aumento encontrado é necessariamente o melhor.

A ideia por trás do algoritmo é manter duas árvores de busca de nós interconectados por arestas para encontrar um caminho de aumento da raiz da árvore de $s$ para a raiz da árvore de $t$, como ilustrado na Figura 3.3. Na árvore $S$ todas as arestas partindo de um nó para seus filhos não estão saturados, enquanto nas arestas da árvore $T$ os nós dos filhos para um determinado nó é que não estão saturados.

O algoritmo possui três tipos de nós: livre, ativo ($A$) e passivo ($P$). Nós que não estão na árvore $S$ ou $T$ são nós livres. Nós ativos são os nós da borda de cada árvore que permitem o crescimento da árvore adquirindo novos filhos

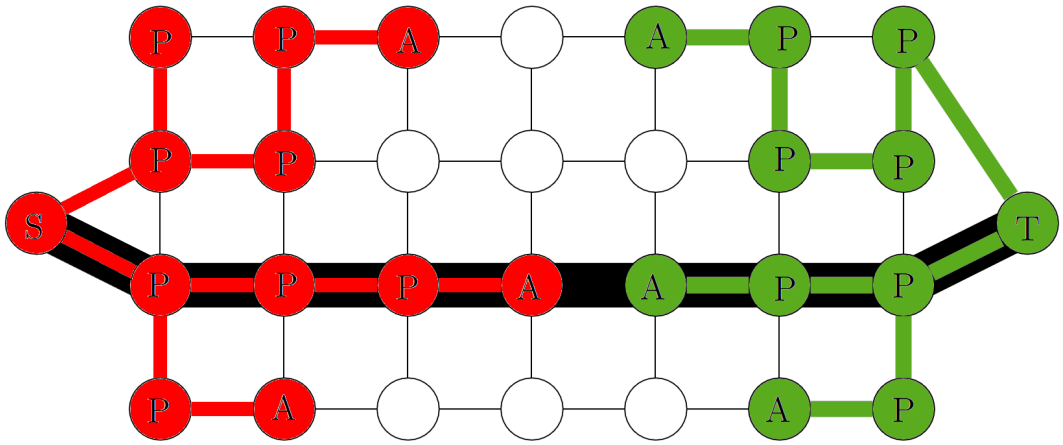


Figura 3.3: Exemplo de árvores de busca $S$ (em vermelho) e $T$ (verde) contendo nós ativos A e nós passivos P. Os nós brancos restantes são os nós livres. O caminho encontrado está em preto (reproduzido de (03)).

de um conjunto de nós livres. Nós passivos são os nós internos que eram nós ativos anteriormente.

O processamento do algoritmo é iterativo e está dividido em três estágios: crescimento, aumento e adoção. Durante o estágio de crescimento, ambas as árvores tentam crescer adquirindo nós de um conjunto de nós livres. O conjunto de nós ativos de uma árvore explora arestas adjacentes não saturadas adquirindo novos filhos do conjunto de nós livres, que se tornam nós ativos. Após ter explorado todos os nós vizinhos, um nó ativo se torna um nó passivo, indicando que não consegue ir mais longe. O estágio de crescimento termina quanto um nó ativo encontra um nó ativo de outra árvore, significando a criação de um caminho de aumento.

O estágio de aumento estende o caminho encontrado com o maior fluxo possível, algumas vezes criando arestas saturadas e, subsequentemente, nós órfãos. Enquanto isso, o estágio de adoção encontra pais para cada órfão, garantindo que o pai esteja conectado através de uma aresta não saturada e pertença à mesma árvore que o órfão. Se não conseguimos achar um nó pai, o nó órfão se torna um nó livre. Esse estágio termina quando não há mais nós órfãos.

O algoritmo executa até que as árvores de busca $S$ e $T$ não possam mais crescer, ou seja, não há mais caminhos de aumento por não haverem mais nós ativos, e as árvores estejam separadas por arestas saturadas. Este é o ponto onde o fluxo máximo é atingido. O corte mínimo resultante é a borda entre $S$ e $T$ onde todas as arestas estão saturadas.