UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

OSCAR PAESI DA SILVA

Redimensionamento de Imagens Preservando a Proporção dos Objetos

Trabalho de Graduação.

Prof. Dr. Manuel Menezes de Oliveira Neto
Orientador

Porto Alegre, Dezembro de 2009

AGRADECIMENTOS

Agradeço à minha mãe, Elides, por sempre ter apoiado meus sonhos e feito tudo a seu alcance para que eu os concretizasse. Por priorizar e sempre buscar o melhor para a educação e formação de seus filhos. Por ser um grande exemplo e ser a pessoa a quem eu mais admiro.

Agradeço ao restante de minha família e aos meus amigos, por serem pessoas maravilhosas e grandes companheiros. Por proporcionarem tantas alegrias e pelo conforto nos momentos difíceis. Por sempre torcerem pelo meu sucesso e se orgulharem das minhas conquistas.

Sou muito grato à equipe da Endeeper Rock Knowledge Systems, por ter me proporcionado um ótimo ambiente de trabalho, onde pude aprender e crescer muito como profissional. Pela troca de experiência e conhecimento em diversos aspectos.

Por fim, agradeço ao meu orientador, Prof. Manuel Menezes de Oliveira Neto, pelo apoio, incentivo e compreensão. Por ter me direcionado no caminho certo diante das dificuldades e por ter me passado tanto conhecimento.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
GPU	**G**raphics **P**rocessing **U**nit (Unidade de processamento gráfico)
ROI	**R**egion of **I**nterest (Região de interesse)

LISTA DE FIGURAS

RESUMO

O redimensionamento de imagens digitais para que sejam exibidas em diferentes dispositivos é uma necessidade crescente, especialmente com a popularização de dispositivos móveis que podem armazenar e exibir grandes quantidades de imagens. As técnicas tradicionais de redimensionamento não são satisfatórias para este fim, uma vez que distorcem o conteúdo das imagens. Diversos trabalhos têm buscado contornar este problema, identificando regiões de importância na imagem e preservando-as durando o redimensionamento. Dentre estas técnicas, destaca-se o *Seam Carving*, que altera as dimensões da imagem através da remoção ou inserção de caminhos de baixa importância visual, ou *seams*. Apesar de apresentar bons resultados finais, o processo é custoso e bastante demorado. Isto torna a tarefa tediosa, especialmente quando é necessário editar grandes conjuntos de imagens. Neste trabalho apresentamos uma implementação do algoritmo de redimensionamento de imagens por *Seam Carving* e avaliamos os seus resultados quanto à preservação do conteúdo das imagens.

Palavras-chave: Redimensionamento de imagens, Manipulação de imagens preservando o conteúdo.

Image Resizing Preserving Object Proportions on the Scene

ABSTRACT

Resizing digital images to fit different devices is a growing need, as today there is a large diversity of mobile devices which can store and display a great number of images. Standard resizing techniques are not suitable for this purpose, as they distort the image content. Several works have been trying to address this issue by searching for regions of interest in the image and protecting them during rezising. One of the most impressive of these techniques is *Seam Carving*. It changes the size of the image by repeatedely carving out or inserting seams of low visual importance. Although it achieves great results, the process takes a significant time to conclude, what makes the resizing process tedious, specialy when dealing with large sets of images. In this work we present an implementation of the *Seam Carving* algorithm for content-aware image resizing and evaluate its suitability regarding content preservation.

Keywords: Image resizing, Image retargeting, Image seams, Content-aware image manipulation .

1 INTRODUÇÃO

Atualmente existem muitos dispositivos para a exibição de imagens, como televisores, monitores de computador e telas de celulares. Ajustar as imagens para que sejam exibidas adequadamente em cada dispositivo é um problema enfrentado constantemente por designers gráficos, que por vezes precisam gerar a mesma imagem em diferentes resoluções e proporções. Ao alterar a proporção de uma imagem, pode-se causar a distorção dos objetos presentes nela, fazendo, por exemplo, com que uma pessoa pareça mais alta ou mais baixa do que realmente é.

A fim de minimizar a distorção de objetos durante o redimensionamento de uma imagem, Avidan e Shamir (2007) propuseram uma técnica que procura manter a proporção dos objetos presentes na cena, chamada *Seam Carving*. Algumas aplicações bastante populares, como o Adobe Photoshop e o GIMP já utilizam o *Seam Carving*. Para diminuir as dimensões de uma imagem, *Seam Carving* remove caminhos de pixels conectados que agregam pouca importância visual a ela, eleitos através do cálculo da energia dos pixels. Rubinsteing et al. (2008) aperfeiçoou a técnica, considerando a energia introduzida na imagem após o redimensionamento, ao invés da energia presente antes de efetuar o processo, alcançando resultados mais realistas. A aplicação da técnica é voltada para pré-processar uma imagem antes de exibí-la e demanda um custo computacional elevado para tal.

1.1 Objetivo

Neste trabalho objetivamos implementar a técnica de redimensionamento de imagens por *Seam Carving* a fim de avaliar os resultados que ela proporciona e a sua viabilidade como ferramenta de edição de imagens preservando o conteúdo.

1.2 Estrutura do Texto

Este trabalho está organizado da seguinte forma:

O **Capítulo 2: Trabalhos relacionados** apresenta os trabalhos mais recentes para redimensionamento de imagens preservando seu conteúdo.

O **Capítulo 3: Seam Carving** detalha as etapas envolvidas na técnica de *Seam Car-*

ving e as contribuições propostas por Avidan e Shamir que serão utilizadas para o desenvolvimento deste trabalho.

O **Capítulo 4: Implementação e resultados** apresenta a aplicação desenvolvida durante o trabalho, para melhor compreensão do método de redimensionamento por *Seam Carving* e verificação dos resultados alcançados.

O **Capítulo 5: Conclusões e trabalhos futuros** discute as melhorias e novas funcionalidades que poderiam ser agregadas ao trabalho.

2 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns dos trabalhos mais recentes e significativos para redimensionamento de imagens preservando seu conteúdo. A maioria deles se motiva pela necessidade de adaptar imagens grandes às pequenas telas de dispositivos móveis, como celulares.

Para adaptar uma imagem grande a dimensões menores utiliza-se, tradicionalmente, uma combinação de recortes e escalas, operações comuns disponíveis nas principais ferramentas de edição de imagens. Porém, este é um processo manual e tedioso, especialmente quando há um conjunto grande de imagens a serem redimensionadas. Além disso, recortes simples não apresentam bons resultados quando a imagem possui mais de um objeto de importância, e escalas distorcem regiões importantes (SETLUR et al., 2005).

SETLUR et al. (2005) propuseram um método automático e não-fotorealístico para a redução de imagens para exibição em telas de dispositivos móveis. O método identifica as regiões de interesse (ROI) da imagem e verifica se é possível incluí-las na imagem final utilizando somente um recorte. Caso não seja possível, as regiões de interesse são recortadas da imagem e os buracos são preenchidos para gerar um plano de fundo. O plano de fundo é reduzido para o tamanho final e os objetos importantes são adicionados. Cada objeto é escalado de acordo com o seu grau de importância antes de ser incluído na imagem final (Figura 2.1). As regiões de interesse da imagem podem ser detectadas automaticamente ou podem ser especificadas pelo usuário.

Figura 2.1: Redimensionamento preservando o realismo funcional ao invés de fotorealismo, utilizando a técnica de SETLUR et al. (2005). (a) Imagem original contendo 3 regiões de importância, os dois garotos e a bola. (b) Imagem redimensionada para ser exibida em uma tela de celular.

LIU; GLEICHER (2005) identificam a região de interesse da imagem e escala cada porção dela de acordo com sua importância, simulando o efeito do uso de uma câmera com lente *olho de peixe*. Primeiro o método determina a ROI através da construção de mapas de saliência e detectores de faces. Então, utiliza uma função de distorção olho de peixe cujo centro é a ROI. A função mapeia cada porção da imagem de entrada para a imagem final, dando ênfase à região de interesse. Diferentemente de SETLUR et al. (2005), nenhuma região da imagem é descartada por completo. Isto permite que o usuário tenha noção de todos os elementos presentes na cena, percebendo com mais clareza aqueles de maior importância (Figura 2.2).

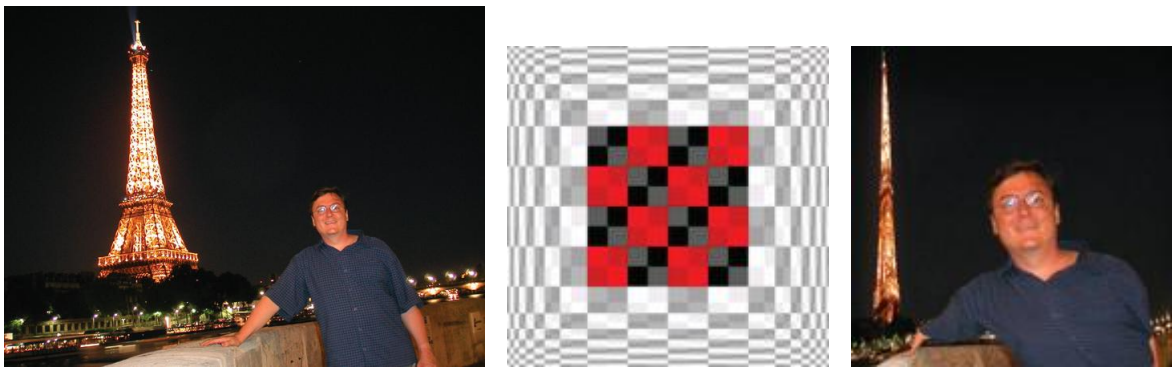


Figura 2.2: Redimensionamento automático com distorção olho de peixe. À esquerda está a imagem original, ao centro uma ilustração da função utilizada e à direita a imagem final.

SANTELLA et al. (2006) considera os movimentos dos olhos do usuário para definir as regiões de importância da imagem. Através de um conjunto de regras de composição, avalia cada possível recorte e seleciona o que melhor de adapta à expectativa do usuário.

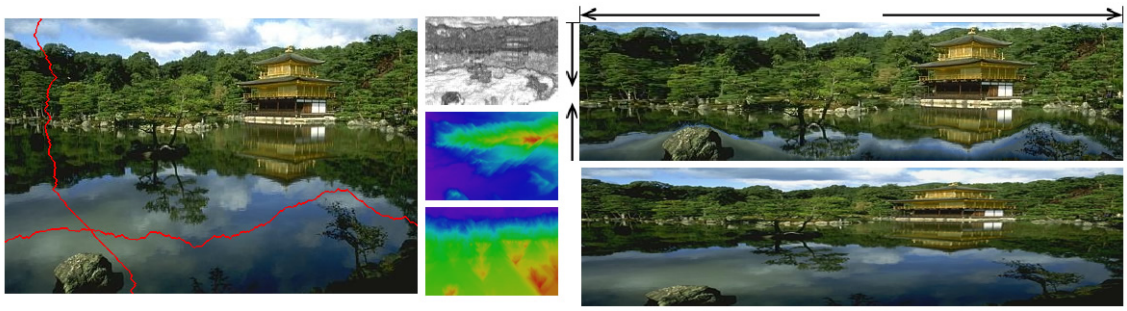


Figura 2.3: Redimensionamento automático por *Seam Carving*. À esquerda, caminhos selecionados na imagem original. Ao centro, o resultado das etapas intermediárias do algoritmo. À direita, a imagem resultante depois da aplicação de ampliação horizontal e redução vertical.

Entre as técnicas mais recentes de redimensionamento automático de imagens preservando o conteúdo, a que mais se destaca é o *Seam Carving* (AVIDAN; SHAMIR, 2007). A execução da técnica inicia pelo cálculo da importância de cada pixel da imagem utilizando uma função de energia (Figura 2.3). O resultado do cálculo da função de energia

pode ser combinado com marcações do usuário ou com funções adicionais, como detectores de faces. Utilizando programação dinâmica, encontra-se um caminho de pixels que corta a image verticalmente ou horizontalmente e cuja energia acumulada seja mínima. O caminho, ou *seam*, pode ser removido, para a redução da imagem, ou servir de base para a inserção de um novo *seam*. O trabalho apresenta ainda uma comparação entre diferentes funções de energia e a especificação de uma estrutura para armazenar uma imagem dinâmica, que pode ser redimensionada em tempo real.

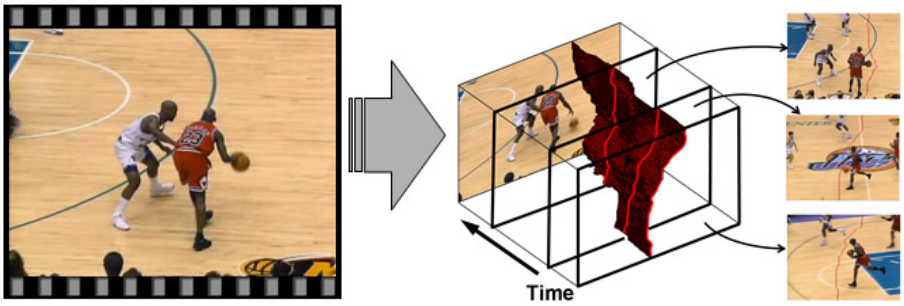


Figura 2.4: *Seam Carving* para vídeos. Combina os frames do vídeo para formar um cubo 3D e utiliza cortes em grafos para encontrar espaços conectados entre os frames.

RUBINSTEIN; SHAMIR; AVIDAN (2008) aperfeiçoaram a técnica de *Seam Carving*, propondo uma nova maneira de encontrar o *seam* ótimo. Ao invés de selecionar o *seam* com menor energia acumulada, seleciona-se aquele que irá inserir a menor quantidade de energia no resultado final. Esta abordagem proporciona melhores resultados por que ao remover um *seam* da imagem, duas regiões que não eram adjacentes passam a ser, e há a criação de uma borda entre elas. Além disso, RUBINSTEIN; SHAMIR; AVIDAN propuseram o cálculo do *seam* ótimo utilizando cortes em grafos. Com esse método é possível estender a técnica para o redimensionamento automático de vídeos, removendo ou inserindo espaços conectados que passam por todos os frames do video (Figura 2.4).

(a) Original

(b) Escala

(c) *Seam Carving*

(d) Multi-operador

Figura 2.5: Multi-operador para redimensionamento de imagens. Comparação entre os resultados obtidos no redimensionamento de uma imagem por diferentes técnicas. O multi-operador (d), que combina diferentes técnicas, proporciona um resultado melhor do que a escala (b) ou *Seam Carving* (c) aplicados individualmente.

Em seu trabalho mais recente, os autores do *Seam Carving* exploram o fato de os usuários da técnica combinarem-na com outros métodos de redimensionamento de imagem para obter melhores resultados (RUBINSTEIN; SHAMIR; AVIDAN, 2009). Considerando que a qualidade do redimensionamento por *Seam Carving* depende da imagem em questão, pode ser melhor utilizar mais de um método na mesma imagem. O trabalho consiste em um algoritmo automático que encontra a combinação entre escala, recorte e *Seam Carving* que oferece o melhor resultado final (Figura 2.5).

3 SEAM CARVING

AVIDAN; SHAMIR propuseram um operador para redimensionamento de imagens, preservando seu conteúdo, chamado *Seam Carving*, que suporta tanto redução como expansão. *Seams* são caminhos ótimos de pixels 8-conexos que cortam a imagem da esquerda à direita ou de cima a baixo. Para encontrar os caminhos ótimos aplica-se uma função de energia que determina quais regiões da imagem têm maior importância visual (alta energia) e quais regiões têm menor importância visual (baixa energia). As regiões de alta energia são preservadas e as regiões de baixa energia são removidas. Através da remoção ou inserção de *seams* em ambas as direções da imagem, podemos alterar seu tamanho e proporção, preservando as informações relevantes.

Consideremos o caso onde queremos reduzir a largura de uma imagem. Assumindo que temos uma função de energia que determina o nível de importância de cada pixel desta imagem, podemos adotar diversas estratégias para fazer a redução. A remoção dos pixels em ordem crescente de importância proporciona um resultado ótimo em termos de conservação de energia na imagem. No entanto, isto faz com que o formato da imagem seja destruído, pois não há garantias que o mesmo número de pixels será removido de cada linha. Podemos preservar o formato da imagem removendo de cada linha o mesmo número de pixels de baixa importância, porém isto destrói o conteúdo da imagem, causando um efeito de zigue-zague. O recorte automático, ou seja, a escolha de uma região da imagem que concentre a maior energia, preserva tanto o conteúdo como a coerência visual da imagem. Esta estratégia propicia bons resultados para imagens com somente uma região de importância, mas pode excluir regiões adicionais. A remoção de colunas inteiras introduz artefatos, como serrilhados em linhas diagonais.

3.1 Seam

Buscando uma alternativa menos restritiva do que recortes ou remoção de colunas, mas que ainda preserve o conteúdo da imagem, AVIDAN; SHAMIR (2007) propuseram a remoção de caminhos que atravessam a imagem de uma extremidade à outra, chamados *seams*. Estes caminhos são importantes também para a solução de outros problemas de processamento de imagens, como a concatenação de partes de imagens de um conjunto para formar uma imagem única com as partes escolhidas pelo usuário (AGARWALA et al., 2004) ou para criar automaticamente uma imagem que sumarize os aspectos importantes de um conjunto de imagens (ROTHER et al., 2006).

Considerando uma imagem $\mathbf{I}$ de dimensões $n \times m$, define-se um *seam* vertical como

sendo:

$$\mathbf{s^x} = \{s_i^x\}_{i=1}^n = \{(x(i),i)\}_{i=1}^n, \text{ i. e. } \forall i, |x(i) - x(i-1)| \leq 1 \tag{3.1}$$

onde x é um mapeamento $x : [1,...,n] \rightarrow [1,...,m]$. Ou seja, um *seam vertical* é um caminho 8-conexo de pixels do topo até a base da imagem, sendo que cada linha da imagem contém um e somente um pixel.

Sendo que y é um mapeamento $y : [1,...,m] \rightarrow [1,...,n]$, um *seam horizontal* é um caminho com as mesmas propriedades, porém da esquerda até a direita da imagem:

$$\mathbf{s^y} = \left\{s_j^y\right\}_{j=1}^m = \{(j,y(j))\}_{j=1}^m, \text{ i. e. } \forall j, |y(j) - y(j-1)| \leq 1 \tag{3.2}$$

A remoção de *seams* introduz somente efeitos locais na imagem, uma vez que todos os outros pixels são deslocados em um pixel para cobrir o espaço onde o *seam* foi removido, e as diferenças são perceptíveis somente ao longo do caminho removido.

3.2 Preservação de energia

Para avaliar a eficácia de diferentes estratégias de redimensionamento de imagens, utiliza-se a medida da média de energia entre os pixels da imagem em questão:

$$\frac{1}{|\mathbf{I}|} \sum_{p \in \mathbf{I}} e(p) \tag{3.3}$$

onde e é o resultado do cálculo da energia da imagem $\mathbf{I}$, e p é o índice do pixel sendo analisado.

A remoção aleatória de pixels da imagem mantém a média de energia constante. Porém, o redimensionamento preservando o conteúdo eleva a média de energia, uma vez que os pixels de menor importância são removidos e os de alta importância permanecem.

3.3 Função de energia

Para redimensionar uma imagem causando o menor impacto visual possível, é necessário determinar quais regiões dela têm maior importância visual. O sistema visual humano tende a perceber melhor diferenças presentes nos limites entre regiões, ou seja, as bordas presentes na imagem. Estas regiões deverão ser preservadas, para que a energia da imagem como um todo seja preservada. *Seam Carving* pode utilizar diferentes métodos para determinar a energia da imagem. Referimo-nos a elas como *funções de energia*. Pode-se combinar diferentes funções de energia entre si ou com o resultado da intervenção do usuário, que pode assinalar regiões que deseja preservar durante o redimensionamento, ou regiões que devem ser eliminadas preferencialmente.

Em AVIDAN; SHAMIR (2007) são estudadas diversas funções de energia, entre elas:

- Norma L_1 do gradiente;
- Norma L_2 do gradiente;

- Medida de saliência (ITTI; KOCH; NIEBUR, 1998);
- Detector de faces;
- e_1 (Equação 3.4), uma aproximação da magnitude do gradiente e
- e_{HoG}, uma combinação entre e_1 e histograma de gradientes orientados (DALAL; TRIGGS, 2005).

Dentre estas funções, e_1 e e_{HoG} oferecem os melhores resultados.

A magnitude do gradiente é um filtro baseado em derivadas bastante utilizado em processamento de imagens para realce de detalhes e detecção de bordas. Para cada pixel da imagem, calcula-se o valor da magnitude do vetor gradiente naquele pixel somando os módulos das derivadas na direção horizontal e vertical. O filtro da magnitude do gradiente pode ser aproximado utilizando a seguinte função (AVIDAN; SHAMIR, 2007):

$$e_1(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \tag{3.4}$$

Os termos da equação 3.4 são obtidos pelas equações:

$$\frac{\partial}{\partial x}\mathbf{I}(x,y) = \mathbf{I}(x+1,y) - \mathbf{I}(x,y) \tag{3.5}$$

$$\frac{\partial}{\partial y}\mathbf{I}(x,y) = \mathbf{I}(x,y+1) - \mathbf{I}(x,y) \tag{3.6}$$

3.4 Mapa de caminhos

A partir do resultado da função de energia, queremos encontrar o *seam* ótimo, ou seja, cujo custo é mínimo. O custo de um *seam* é definido pela soma da energia de todos os pixels que o constituem:

$$E(\mathbf{s}) = E(\mathbf{I_s}) = \sum_{i=1}^{n} e(\mathbf{I}(s_i)) \tag{3.7}$$

onde e é a função de energia descrita na seção 3.3 e s é um *seam* na imagem $\mathbf{I}$. O *seam* ótimo é definido por:

$$s^* = \min_s E(\mathbf{s}) = \min_s \sum_{i=1}^{n} e(\mathbf{I}(s_i)) \tag{3.8}$$

O cálculo de *seams* pode ser feito através de diversos métodos, como o algoritmo para caminho mínimo de Djikstra (CORMEN et al., 2001a), programação dinâmica (CORMEN et al., 2001b), ou cortes em grafos (CORMEN et al., 2001c). Neste trabalho utilizamos programação dinâmica. Percorremos a imagem da segunda à última linha, computando a energia cumulativa de todos os possíveis *seams*. Obtemos então a matriz $\mathbf{M}$, que será o mapa de caminhos:

$$M(i,j) \;=\; e(i,j) \;+\; min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \tag{3.9}$$

onde e é o resultado do cálculo da função de energia.

O valor mínimo encontrado na última linha de $\mathbf{M}$ indicará o fim do caminho vertical de menor custo. Para recuperar todos os pixels que fazem parte do caminho, realizamos um *backtracking* através do mapa de caminhos.

3.5 Remoção e inserção de seams

Considerando uma imagem $\mathbf{I}$ de dimensões $n \times m$, se quisermos alterar a sua proporção para $n \times m'$, onde $m - m' = c$, podemos remover sucessivamente c seams verticais da imagem $\mathbf{I}$. Este processo, ao contrário de uma ampliação convencional, não alterará as partes importantes da imagem, definidas pela aplicação da função de energia. Similarmente, para alterar a proporção da imagem de $n \times m$ para $n \times m'$, onde $m' - m = c$, podemos inserir c seams em $\mathbf{I}$.

3.6 Pipeline

Podemos enxergar o processo de redimensionamento por *Seam Carving* como um *pipeline* composto das seguintes operações:

- Cálculo da energia da imagem
- Cálculo da energia acumulada dos *seams*
- Escolha do *seam* de menor energia
- Remoção/Inserção do *seam* escolhido

O seguinte diagrama ilustra a sequência de operações:

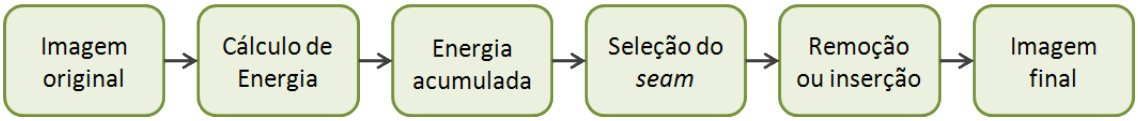


Figura 3.1: *Pipeline* do processo de *Seam Carving*.

3.7 Redução

Para reduzir a largura de uma imagem por uma coluna, identificaremos o seam ótimo, definido pelo mapa de caminhos, e removeremos todos os pixels da imagem que fazem parte deste seam. Os pixels restantes são então deslocados para cobrir o espaço deixado, gerando uma imagem final com uma coluna a menos.
Para remover k *seams*, repete-se o processo utilizando a imagem gerada com uma coluna a menos, k vezes, conforme a figura 3.2.

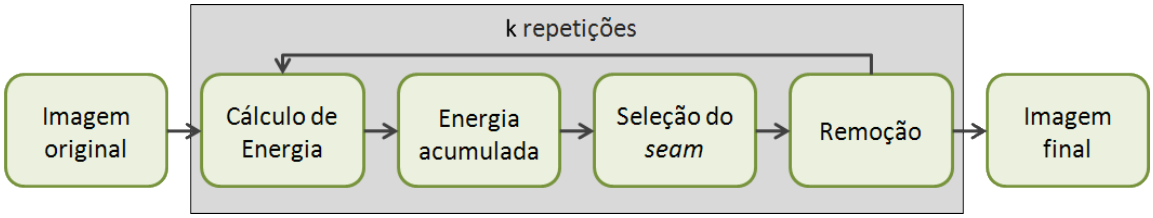


Figura 3.2: *Pipeline* do processo de remoção sucessiva de *seams* da imagem.

3.8 Ampliação

O processo de aumentar a largura de uma imagem por uma coluna, utiliza o mesmo *seam* ótimo definido pelo mapa de transporte e duplica-o calculando a média entre os pixels à esquerda e à direita de cada pixel do seam. O restante dos pixels da imagem é deslocado em uma coluna, e a imagem resultante terá uma coluna a mais. Analogamente ao processo de redução, para inserir *k seams*, repete-se a inserção na imagem com uma coluna a mais, conforme mostra a figura 3.3.

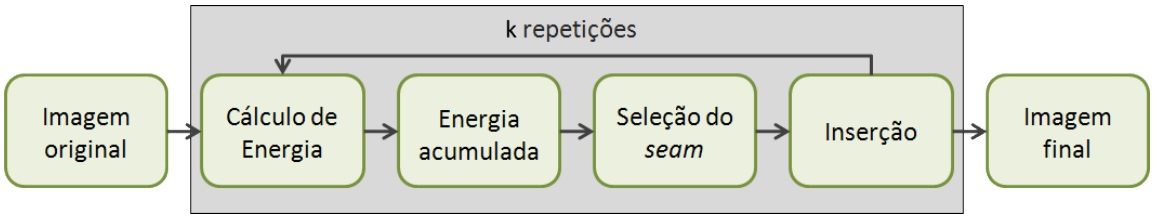


Figura 3.3: *Pipeline* do processo de inserção sucessiva de *seams* da imagem.

No entanto, selecionar o *seam* ótimo a cada iteração faz com que o mesmo *seam* seja selecionado repetidamente. Isto introduz um efeito de *stretching* no local onde o *seam* é inserido (Figura 3.4). Para evitar esse efeito, é necessário escolher os *k* melhores *seams* e duplicá-los. O processo de ampliação tem um *pipeline* diferenciado do processo de redução. A energia da imagem e a energia acumulada dos *seams* não é recalculada a cada iteração. Os *k seams* são selecionados de uma vez só, no mesmo mapa de caminhos. A Figura 3.5 mostra a mudança.

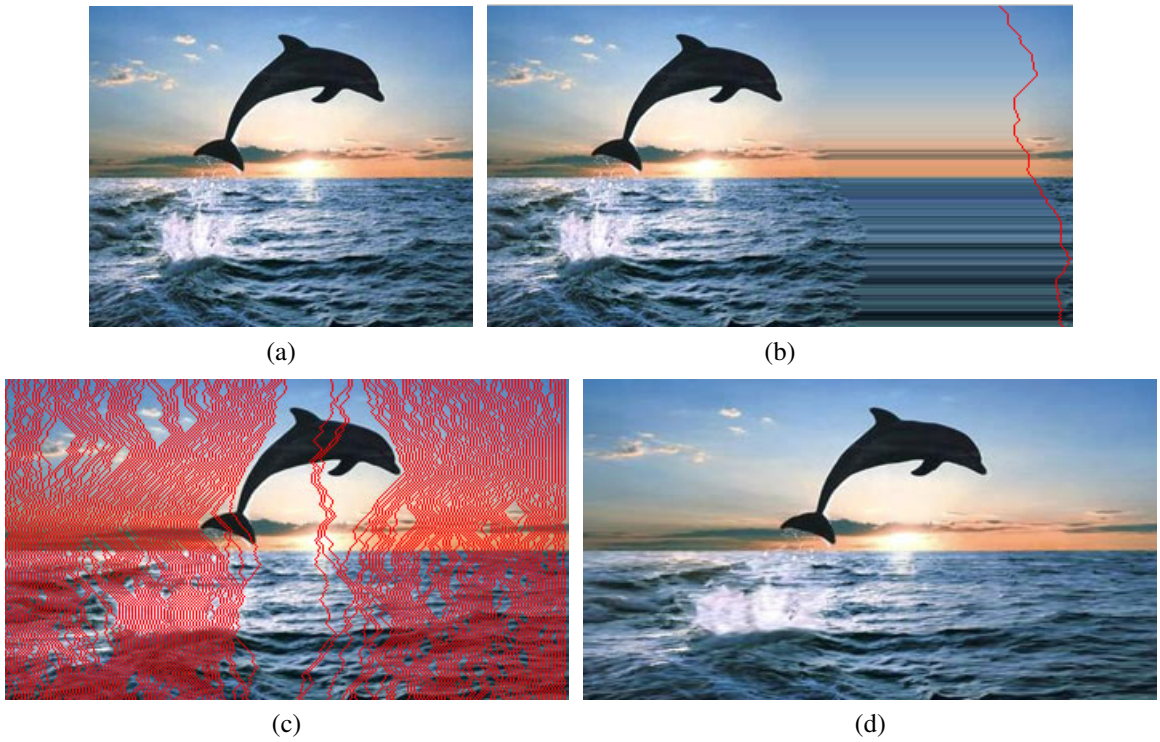


Figura 3.4: Efeito indesejado na inserção do *seam* ótimo. Ao escolher e inserir o *seam* ótimo repetidamente, causa-se um efeito de *stretching* na imagem final (b). Selecionando os *k seams* de menor energia (c), consegue-se um efeito melhor (d) (AVIDAN; SHAMIR, 2007).

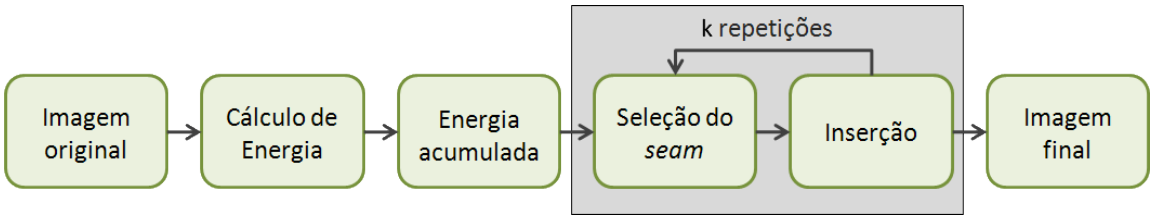
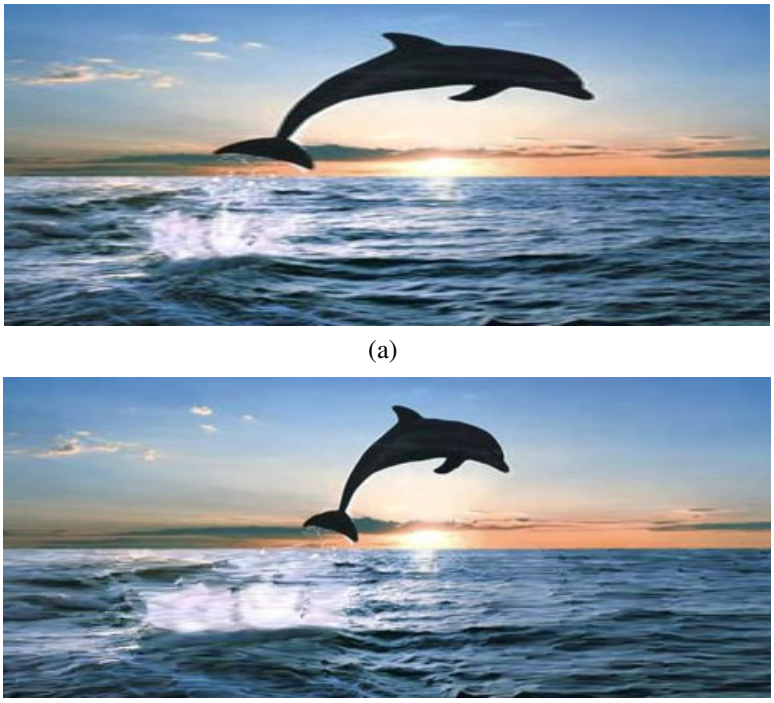


Figura 3.5: *Pipeline* do processo modificado de ampliação da imagem pela inserção de *seams*.

3.9 Restrição de alterações por região da imagem

A duplicação de todos os *seams* de uma imagem é equivalente à aplicação de uma escala de 200%. Para evitar que todo o conteúdo da imagem seja duplicado, é necessário dividir o processo de redimensionamento para grandes proporções da imagem em diversas etapas. Cada etapa aumenta a imagem em apenas uma fração do seu tamanho original.



(a)

(b)

Figura 3.6: Ampliação de grandes porções da imagem. A duplicação de todos os *seams* é equivalente à operação de escala (a). O redimensionamento em diversas etapas (b) preserva a região importante (AVIDAN; SHAMIR, 2007).

3.10 Seleção de seams considerando a energia inserida

Apesar de oferecer bons resultados para grande parte das imagens, a remoção de *seams* com baixa energia não leva em conta a energia que é inserida na imagem após o processo. Ao remover *seams*, pixels que não eram adjacentes anteriormente passam a ser, o que pode introduzir artefatos na imagem. A fim de contornar este problema, pode-se incorporar ao *Seam Carving* um método mais robusto para a seleção dos *seams*, chamado *forward energy* (RUBINSTEIN; SHAMIR; AVIDAN, 2008).

Assumindo o redimensionamento de uma imagem $I = I_{t-1}$ pela remoção de k *seams* (t = 1, ..., k), avaliamos a mudança de energia na imagem após a remoção de cada *seam* pela diferença entre a imagem após a remoção ($I_{t=i+1}$) e a energia das partes que não foram removidas na imagem anterior I_{t-1}. A equação 3.10 determina a diferença de energia entre a imagem original dada como entrada para o *Seam Carving* e a imagem resultante após a remoção de k *seams*.

$$\Delta E_{t=i+1} = E(I_{t=i+1}) - [E(I_{t=i}) - E(S_i)] \tag{3.10}$$

Para prevenir a introdução de artefatos devido à remoção de *seams*, passa-se a escolher como caminho ótimo aquele que introduzirá a menor quantidade de energia após o processo de remoção. Embora a energia deste caminho possa não ser mínima, ele introduzirá menos artefatos na imagem final.

O cálculo dos melhores caminhos passa a ser feito considerando a diferença de energia entre os pixels que passarão a ser vizinhos após a remoção. Há três direções possíveis que

o caminho pode tomar, e o custo para cada uma delas é calculado pelas equações:

$$(a)\ C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)|$$
$$(b)\ C_U(i,j) = |I(i,j+1) - I(i,j-1)|$$
$$(c)\ C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$$

Define-se então um novo mapa de caminhos **MF**, calculado através de programação dinâmica usando estes novos custos:

$$MF(i,j) = P(i,j) + min \begin{cases} MF(i-1,j-1) + C_L(i,j), \\ MF(i-1,j) + C_U(i,j), \\ MF(i-1,j+1 + C_R(i,j) \end{cases} \tag{3.11}$$

onde P(i,j) é uma medida de energia adicional, como o resultado de um filtro de detecção de faces ou pesos informados pelo usuário, que podem ser utilizados em conjunto com o custo da *forward energy*.

3.11 Imagem Multi-tamanho

Como nem sempre é possível saber qual será o tamanho que uma imagem precisa ter para ser exibida em um contexto, Avidan e Shamir propõe *imagens multi-tamanho*, que armazenam as informações sobre os *seams* de uma imagem e permitem seu redimensionamento em tempo real, tanto para dimensões menores do que a original como para dimensões maiores. A estrutura proposta armazena uma pequena quantidade de informações e pode ser pré-computada em alguns segundos.

Considerando que queremos mudar a largura de uma imagem, definimos um *mapa de índices* **V**, de tamanho $m \times n$ que armazena, para cada pixel da imagem, o índice do *seam* vertical que o remove. Desta forma, para obter uma imagem com largura *m'*, basta recuperar em cada linha todos os pixels cujo índice seja maior ou igual a $m - m'$.

Esta representação suporta também a ampliação da imagem através da inserção de *seams*, similarmente ao processo descrito na seção 3.8. Os seams inseridos recebem índices negativos, a partir de -1. Para aumentar a imagem original em k pixels, pega-se os pixels da imagem ampliada cujo indíce é maior do que $m - (m+k) = -k$, obtendo uma imagem de com tamanho $m+k$.

Para redimensionar uma imagem verticalmente é necessário gerar outro mapa de índices **H** para armazenar os índices dos *seams* horizontais. Porém, o redimensionamento em ambas as direções não pode ser feito, pois os *seams* verticais e horizontais podem colidir em mais de um ponto. A remoção de um *seam* vertical que colide com um *seam* horizontal em mais de um ponto destrói a estrutura do mapa de índices horizontal.

3.11.1 Mapas de índices consistentes

É necessário definir mapas de índices consistentes para evitar que eles sejam destruídos durante o redimensionamento. Dizemos que **H** e **V** são consistentes se cada *seam*

horizontal intersecciona todos os *seams* verticais uma e somente uma vez. A consistência garante que a remoção de um *seam* em uma direção fará com que exatamente um pixel seja removido de cada *seam* na outra direção, preservando a estrutura.

Podemos manter a consistência entre os mapas de índices utilizando somente *seams* que são conectados na imagem original. Só é possível ocorrer violação de consistência entre estes *seams* em passos diagonais. Portanto, calculamos primeiramente o mapa de índices de uma direção, marcando todas as diagonais utilizadas por ele. Em seguida, calculamos o mapa de índices na outra direção, utilizando somente as diagonais que ainda não foram utilizadas. Desta forma, não haverá diagonais utilizadas por *seams* em ambas as direções, garantindo a consistência dos mapas.

4 IMPLEMENTAÇÃO E RESULTADOS

A fim de utilizar a técnica de *Seam Carving* e melhor avaliar seus resultados, desenvolvemos uma aplicação que permite redimensionar uma imagem visualizando todos os estágios intermediários do processamento. A aplicação permite carregar uma imagem e determinar quais são as dimensões que queremos que ela assuma. O usuário pode então visualizar a função de energia calculada para aquela imagem, os mapas de transporte vertical e horizontal e a imagem final.

4.1 Ambiente de desenvolvimento

O desenvolvimento da aplicação foi feita utilizando a linguagem C++ no ambiente de desenvolvimento *Microsoft Visual Studio 2008*. Para modelar a interface utilizou-se a biblioteca Qt. Esta biblioteca fornece ainda diversas funcionalidades adicionais. Entre elas, o suporte para carregamento de imagens em diversos formatos, que foi bastante útil para o trabalho. Utilizou-se ainda um plugin de integração da biblioteca *Qt* com o *Visual Studio*.

4.2 Aplicação

A aplicação permite o carregamento de uma imagem digital em diferentes formatos. A partir da imagem carregada, é possível visualizar o resultado cálculo da energia para ela, os mapas de caminhos verticais e horizontais e configurar a utilização de *Forward Energy*.

Para redimensionar a imagem, escolhe-se o tamanho final desejado e inicia-se o cálculo. Pode-se visualizar o resultado passo-a-passo (processo mais lento, pois envolve a conversão das etapas intermediárias para um formato que possa ser exibido), ou somente após o processamento estar completo.

Além disso, é possível gerar imagens multi-tamanho a partir da imagem carregada. Maiores detalhes são discutidos na Seção 4.3.6.

4.3 Redimensionamento

Conforme descrito na seção 3.7, o processo de redimensionamento de uma imagem por *Seam Carving* envolve basicamente 4 passos, que são repetidos para cada *seam* remo-

vido ou inserido:

- Cálculo da energia da imagem;
- Cálculo da energia acumulada, ou mapa de caminhos;
- Seleção do *seam* e
- Remoção/inserção do *seam* selecionado.

O cálculo da energia da imagem e o cálculo do mapa de caminhos são implementados da mesma forma, tanto para a ampliação como para a redução da imagem. No entanto, a seleção do *seam* e remoção ou inserção são feitos de formas específicas para cada caso.

4.3.1 Cálculo da energia

Para o cálculo da função de energia e_1 (Equação 3.4), percorremos os pixels da imagem, computando a diferença entre o pixel atual e seu vizinho da direita e entre o pixel atual e seu vizinho de baixo. A soma entre o valor absoluto destas diferenças resulta na aproximação da magnitude do gradiente. O cálculo das diferenças é computado para cada canal dos pixels (R, G e B) e o valor final da energia para o pixel é dado pelo valor de luminância resultante da equação:

$$l = (R \times 0,299) + (G \times 0,587) + (B \times 0,114) \tag{4.1}$$

A fim de melhor ilustrar os valores computados para as imagens neste trabalho, convertemos os valores de luminância representados em 8 bits para uma escala de cores. A Figura 4.1 mostra a escala utilizada:

Figura 4.1: Escala de cores utilizada na exibição da energia.

A Figura 4.2 mostra um exemplo da aplicação da função de energia exibido com a escala de cores:

4.3.2 Mapa de caminhos

Depois de obter a energia da imagem, calculamos a estrutura que determina o peso total de cada *seam* na imagem. Esta estrutura é uma matriz com as mesmas dimensões da imagem. Cada posição da matriz armazena o peso do caminho de menor custo que termina no pixel de mesmo índice.

Assumindo que estamos redimensionando a imagem verticalmente, inicializamos a estrutura preenchendo a primeira linha do topo com 0. A partir a segunda linha, calculamos o peso acumulado para cada pixel, conforme a Equação 3.9. A Figura 4.3, mostra o resultado do cálculo dos mapas de caminhos para a imagem mostrada na Figura 4.2.

Figura 4.2: Exemplo do cálculo da função de energia e_1. A imagem original à esquerda e o resultado do cálculo da energia à direita, exibido na escala de cores.

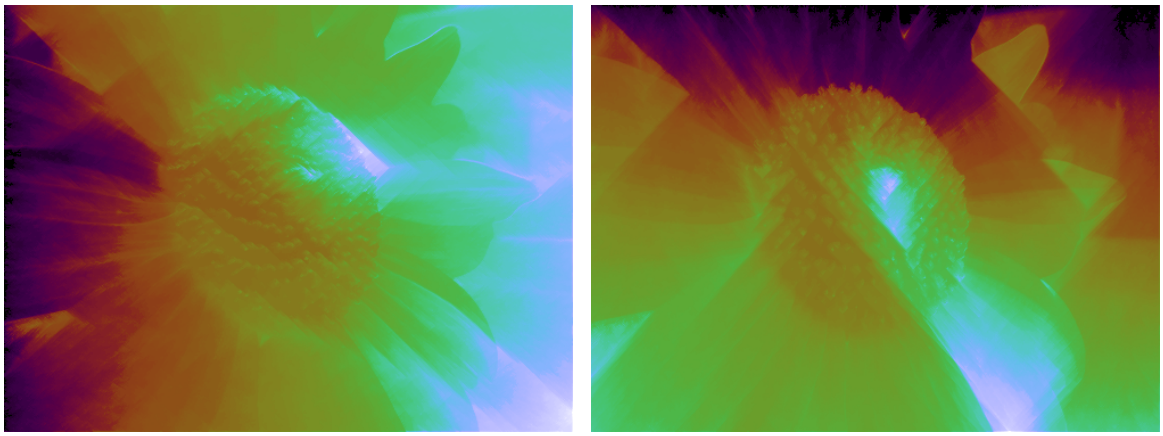


Figura 4.3: Mapas de caminhos. À esquerda, o mapa de caminhos horizontais, utilizado para o redimensionamento vertical da imagem. À direita, o mapa de caminhos verticais, utilizado para o redimensionamento horizontal.

4.3.3 Redução

O processo de redução de uma imagem por *Seam Carving* envolve a repetição dos 4 passos básicos relacionados na Seção 4.3. A seguir, descrevemos os detalhes da implementação dos dois últimos passos para a remoção de *seams*.

4.3.3.1 Seleção de seams *para remoção*

A cada iteração do algoritmo escolhemos o *seam* a ser removido. Ele será o *seam* com a menor energia acumulada, calculada no mapa de caminhos. Para selecionar o *seam* a ser removido, procuramos o menor valor da última linha do mapa de caminhos verticais, se estamos redimensionando a imagem horizontalmente, ou a última coluna do mapa de caminhos horizontais, se estamos redimensionando-a verticalmente.

O índice do valor encontrado é correspondente ao último pixel do *seam* ótimo na imagem. Depois de encontrado o menor valor, efetuamos um *backtraking* na matriz, recuperando o restante dos índices do *seam* mínimo através da seleção do menor valor entre as 3 posições da linha ou coluna anterior (dependendo do sentido do redimensionamento) da matriz que estão 8-conectadas à posição anterior. Depois de percorrer toda a imagem,

obtemos o conjunto dos índices que compõe o *seam* ótimo.

A Figura 4.4 mostra os *seams* selecionados para os processos de redução horizontal de vertical. Eles passam pela região de menor energia acumulada, conforme é possível constatar olhando a Figura 4.3.

Figura 4.4: *Seams* selecionados para remoção. Na imagem à esquerda, o *seam* horizontal ótimo está selecionado, e na imagem à direita, o *seam* vertical ótimo está selecionado.

4.3.3.2 Remoção de seams

Para remover o *seam* ótimo, criamos uma imagem de tamanho correspondente à imagem original, com uma linha ou uma coluna a menos. Então, copiamos os pixels da imagem original, exceto os pixels pertencentes ao *seam* selecionado.

4.3.4 Ampliação

O processo de seleção de um *seam* para inclusão difere do processo de seleção para remoção, pois conforme visto na Seção 3.8, a seleção sucessiva do *seam* ótimo para a ampliação da imagem provoca o efeito de *stretching* na imagem final. Para contornar este problema, selecionamos de uma vez só todos os *seams* que serão inseridos na imagem.

4.3.4.1 Seleção de seams *para inclusão*

A seleção de um *seam* para inclusão envolve o mesmo processo de *backtracking* discutido na Seção 4.3.3.1, porém, ao invés de gerar uma lista de índices de pixels, utilizamos uma matriz adicional que é usada como uma máscara. Ao selecionar um *seam*, marcamos na máscara todos os pixels que fazem parte dele.

Como o *seam* não é removido da imagem, o mapa de caminhos ainda contém a energia acumulada para ele. A máscara é utilizada para que possamos ignorar os pixels de *seam* já selecionado. Ao fazer o *backtracking*, comparamos os 3 valores mais próximos ao pixel previamente selecionado. Ao contrário do processo de seleção de *seams* para remoção, os 3 valores podem não estar adjacentes ao pixel anterior.

Na seleção de um *seam* vertical, encontramos o menor valor na última linha do mapa de caminhos e depois percorremos a linha superior até encontrar um valor à esquerda

e um valor à direita que não estejam marcados na máscara, ou seja, não pertençam a nenhum outro *seam* já escolhido. O terceiro valor é o do pixel imediamente acima do pixel anterior. Caso este pixel já esteja selecionado, adotamos o valor limite da representação para comparação, para evita que ele seja selecionado entre os 3 pixels candidatos.

4.3.4.2 Inclusão de seams

Para incluir os *seams* selecionados, criamos uma nova imagem, adicionando linhas ou colunas extras, e percorremos a imagem original copiando os seus pixels para a nova imagem. Quando encontramos um pixel marcado na máscara, inserimos um novo pixel na imagem redimensionada tomando a média entre o pixel marcado e o seu vizinho.

A fim de melhor proteger o conteúdo da imagem, dividimos o processo de ampliação em etapas menores. Caso o número de *seams* a serem inseridos ultrapasse um certo limite, repetimos toda a operação para os *seams* restantes. Na aplicação desenvolvida, um máximo de 50% da imagem é alterada a cada iteração.

A figura 4.5 mostra um conjunto de *seams* selecionados para a ampliação da imagem, em ambas as direções.

Figura 4.5: *Seams* selecionados para inserção. Na imagem à esquerda, todos os *seams* horizontais selecionados para o processo de ampliação da imagem, e na imagem à direita, todos os *seams* verticais selecionados.

4.3.5 *Forward energy*

Para obter melhores resultados no redimensionamento das imagens, implementamos o mapa de caminhos utilizando *Forward Energy*, processo que está descrito em detalhes na Seção 3.10. A construção deste mapa de caminhos leva em conta a energia que será inserida na imagem após a inserção ou remoção de um *seam*, e permite selecionar o *seam* que causará menos impacto visual no resultado final. A Figura 4.6 mostra os novos mapas de caminhos.

Nas Figuras 4.7 e 4.8, mostramos alguns exemplos comparativos entre o redimensionamento da imagem com o mapa de caminhos comum e com o mapa de caminhos utilizando *Forward Energy*.

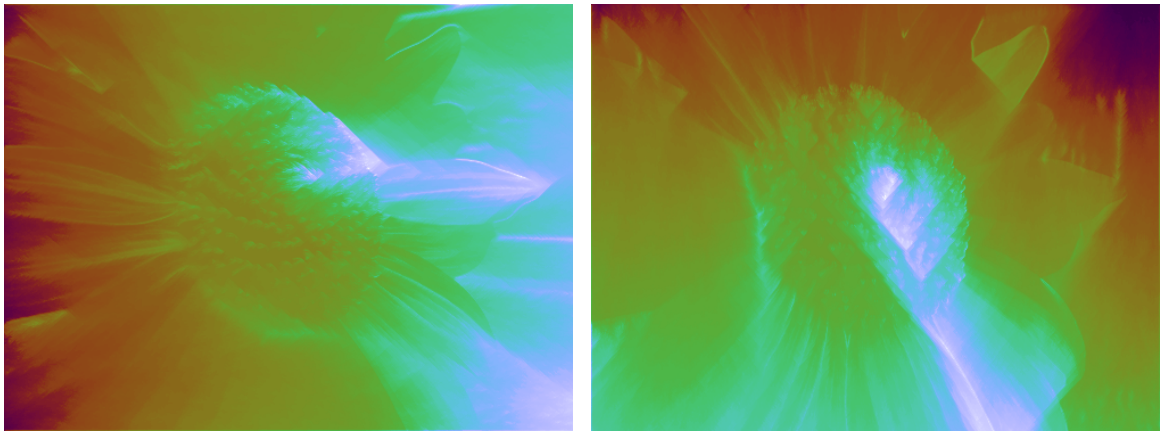


Figura 4.6: Mapas de caminhos utilizando *Forward Energy*. Mapa de caminhos horizontais à esquerda e mapa de caminhos verticais à direita.

4.3.6 Imagem multi-tamanho

Como uma alternativa para o uso do *Seam Carving* como uma ferramenta de pré-processamento de imagens, implementamos as *imagens multi-tamanho*, descritas na Seção 3.11. A aplicação permite a geração de uma imagem com largura variável (Figura 4.9), de uma imagem com altura variável (Figura 4.10), ou de uma imagem variável em ambas as direções.

Depois de gerar a imagem, pode-se redimensioná-la através de uma interface específica da aplicação, em tempo real. Seria possível armazenar a estrutura pré-calculada em formato de arquivo, para poder incorporá-la a alguma aplicação que possa suportar o redimensionamento em tempo real.

4.3.6.1 Mapas de índices consistentes

Em um computador com processador Inter Core 2 Duo de 2,3GHz, a geração de uma imagem variável em apenas uma direção é realizada em menos de 1 segundo, para uma imagem de dimensões 500 x 400. Porém, para gerar uma imagem redimensionável nas duas direções, é necessário o emprego de uma técnica mais robusta, para preservar a integridade dos mapas de índices, as estruturas intermediárias pré-calculadas na geração da imagem.

Utilizamos a técnica referenciada por AVIDAN; SHAMIR (2007) para a construção dos mapas de índices consistentes, conhecida como o algoritmo húngaro KUHN (1955). Porém, a implementação é baseada em uma adaptação do algoritmo (DOTY).

Primeiro, calculamos o mapa de índice na direção horizontal, criando as conexões entre dos pixels entre duas colunas da imagem. A conexão dos pixels é modelada como um problema de atribuição de grafos, onde encontramos o conjunto de arestas de peso mínimo que liga dois grafos bipartidos. Cada coluna da imagem é um grafo bipartido, os pixels da coluna são os vértices do grafo e as arestas são as adjacências entre os pixels. Os pesos das arestas são os valores de energia cumulativa dos pixels, computados no mapa de caminhos.

O algoritmo executa com o grafo representado por uma matriz de adjacência. Logo, cada linha da matriz contém os pesos de 3 arestas que conectam um pixel de uma coluna aos pixels de outra. O restante da linha é preenchido com o valor infinito. Para uma imagem de largura 500, é necessário executar o algoritmo 499 vezes, criando uma matriz de 500 x 500 e processando-a. Depois é necessário computar o mapa de índices na outra direção, desta vez omitindo as arestas que já foram utilizadas no mapa de índices da primeira direção.

Ao contrário das imagens variáveis em somente uma direção, a geração da imagem com mapas de índices consistentes para o redimensionamento em ambas as direções demanda muito tempo, chegando a dezenas de minutos para uma imagem de dimensões 500×400.

4.3.7 Implementação em GPU

A fim de melhorar o desempenho da técnica de *Seam Carving*, tentamos implementar os algoritmos que a compõe utilizando hardware gráfico programável. A linguagem escolhida para a implementação foi CUDA, e o hardware utilizado foi uma GPU NVIDIA GeForce 8500 GT. Com a tecnologia CUDA é possível dividir os dados em porções e processá-las em paralelo na GPU, acelerando a execução do algoritmo.

Implementamos as funções de energia utilizadas e constatamos bons ganhos de desempenho, porém o desenvolvimento não pode ser concluído devido a dificuldades na implementação do cálculo da energia acumulada.

Primeiramente, encontramos dificuldades para depurar o código desenvolvido em CUDA. Ao executar o programa, bugs presentes no código corrompiam o desenho da tela tanto na aplicação como no restante do sistema operacional, impedindo o uso do computador e forçando sua reinicialização. Este cenário fez com que muito tempo fosse gasto na tentativa de depurar o programa, e inviabilizou a implementação. Para evitar o efeito de corrupção do desenho da tela, é necessário utilizar duas placas gráficas, deixando uma para o desenho e a outra para o processamento paralelo. Desta forma, quando houver um problema durante o processamento paralelo, o desenho não será comprometido.

Além disso, as mensagens de erro exibidas pelo *runtime* do CUDA não são esclarecedoras, dificultando a localização dos bugs e a compreensão necessária para consertá-los.

Para completar o algoritmo em GPU, é necessário finalizar a implementação do cálculo de energia acumulada, que foi interrompido pela dificuldade de depuração, implementar a seleção dos *seams* e a geração da imagem final.

O processo de seleção dos *seams* não é ideal para ser paralelizado, pois a escolha de um pixel do *seam* depende da escolha dos pixels anteriores. Por outro lado, o cálculo da energia dos pixels, o cálculo da energia acumulada e a geração da imagem final são processos que se beneficiam da aceleração proporcionada pela GPU.

Figura 4.7: Redimensionamento horizontal com *Forward Energy*. Na imagem do canto inferior esquerdo, com redimensionamento sem *Forward Energy*, há a introdução de bordas indesejadas nas laterais da imagem. Além disso, o formato do banco não foi bem conservado. Na imagem do canto inferior direito, com *Forward Energy*, não há a inserção de bordas e o formato do banco foi melhor mantido.

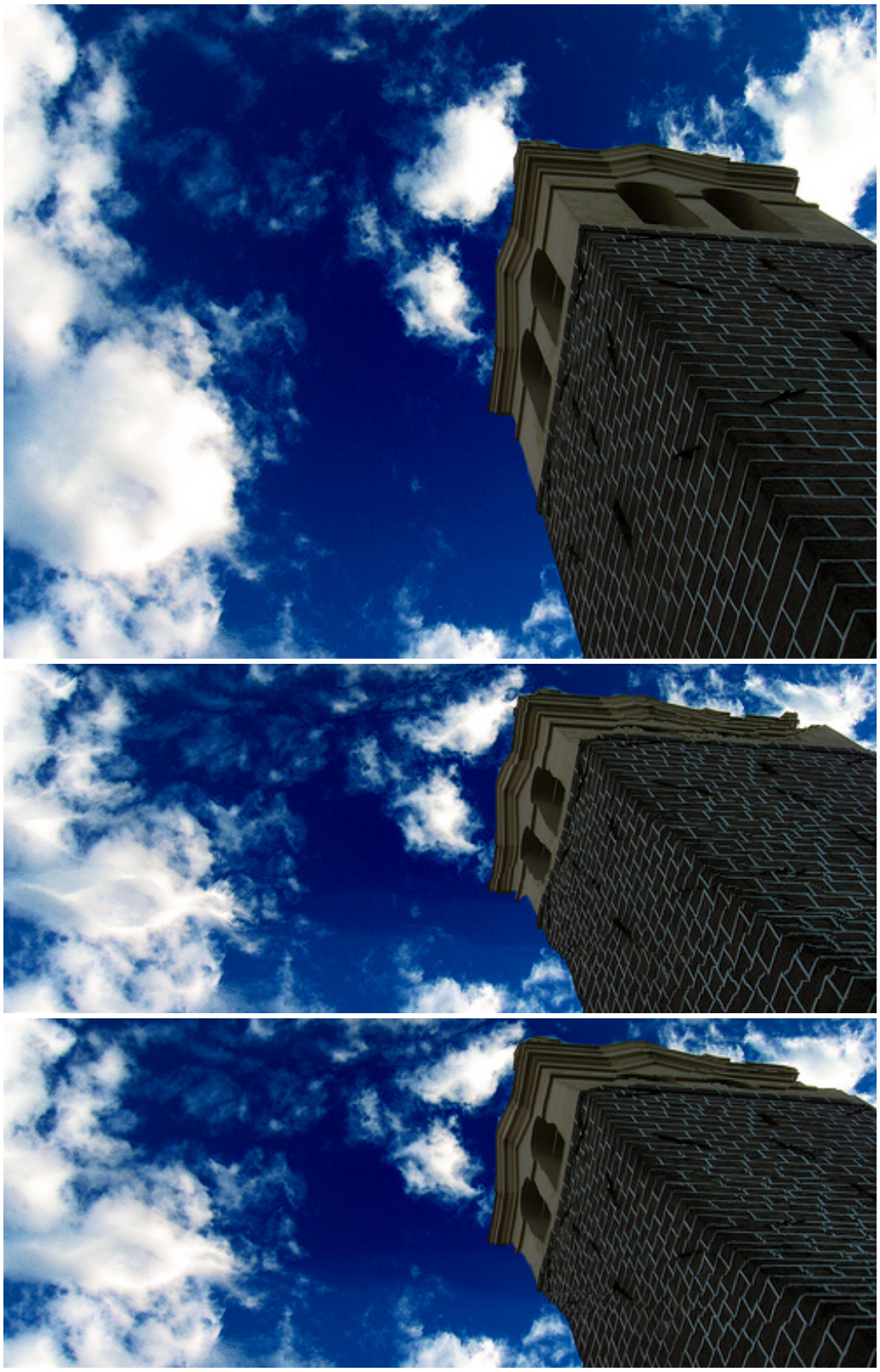


Figura 4.8: Redimensionamento vertical com *Forward Energy*. A imagem original está no topo. Na imagem do centro, redimensionada com o uso de um mapa de caminhos simples, é possível notar a introdução de artefatos mais sérios do que os introduzidos com a utilização de *Forward Energy*, na imagem de baixo. A diferença na deformação do telhado da torre é bem visível.

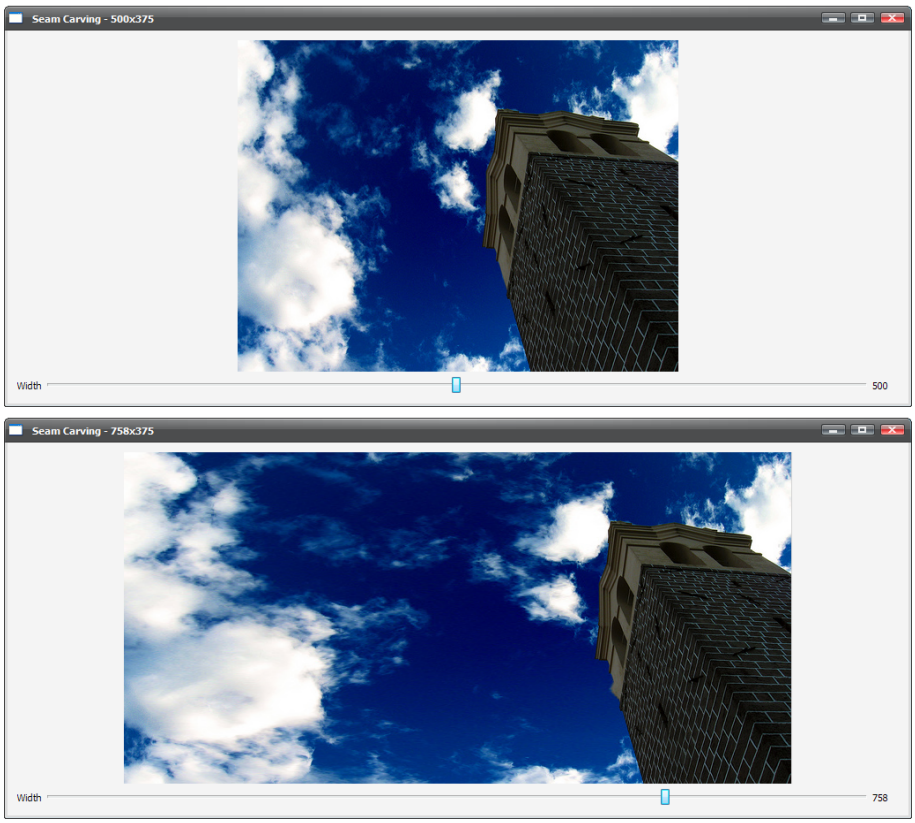


Figura 4.9: Interface para redimensionamento de imagens multi-tamanho horizontais. No topo, a imagem original. Embaixo, a imagem ampliada horizontalmente.

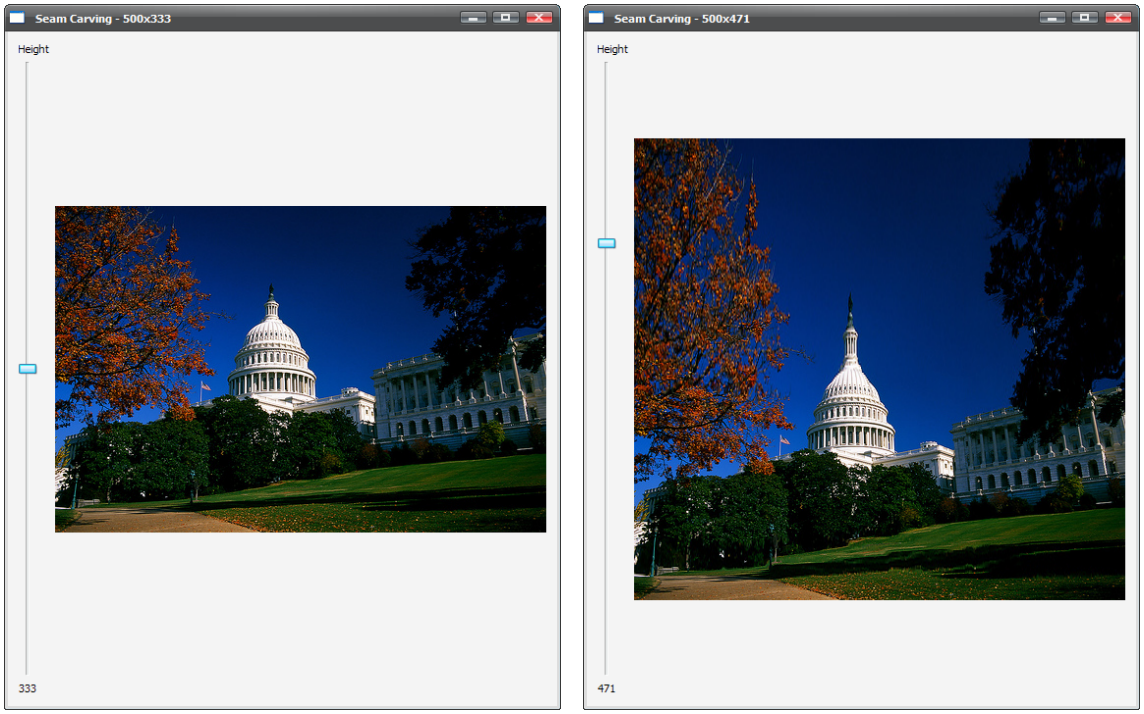


Figura 4.10: Interface para redimensionamento de imagens multi-tamanho verticais. À esquerda, a imagem original. À direita, a imagem ampliada verticalmente.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho implementamos a técnica de redimensionamento de imagens por *Seam Carving*. No Capítulo 2, relatamos diversas técnicas recentes para o redimensionamento de imagens preservando as regiões de interesse (ROI), bem como o propósito a que cada uma delas se destina. Também apresentamos os trabalhos relacionados ao redimensionamento de imagens por *Seam Carving*, que apresentam as técnicas descritas em detalhes no Capítulo 3.

No Capítulo 4 apresentamos os resultados da implementação, com diversos exemplos ilustrando o resultado de cada etapa da aplicação da técnica de *Seam Carving*. Além disso, apresentamos a implementação das imagens multi-tamanho, que permitem o redimensionamento de imagens em tempo real, tendo calculado previamente a estrutura que permite a realização desta operação.

Para complementar este trabalho, o passo natural é a implementação dos mapas de caminhos utilizando cortes em grafos. A utilização de cortes em grafos (CORMEN et al., 2001c), ao invés da programação dinâmica (CORMEN et al., 2001b), permite que a técnica seja estendida para o redimensionamento de videos. Existem técnicas específicas de otimizações para cortes em grafos e trabalhos que propõe implementações desta técnica utilizando o hardware gráfico (VINEET; NARAYANAN, 2008).

A utilização do *Seam Carving* para o redimensionamento de vídeos em tempo real sem o uso de uma estrutura dinâmica como a discutida em (RUBINSTEIN; SHAMIR; AVIDAN, 2008), é um grande desafio. Encontrar caminhos que passem por todos os frames de um vídeo de longa duração demanda muito processamento e seria inviável. É necessário estudar formas de reduzir o número de frames considerados, como dividir o vídeo em diversas cenas e processar somente a cena sendo visualizada.

O hardware gráfico vem evoluindo rapidamente e incorporando mais módulos programáveis, permitindo que se utilize sua tecnologia nos mais variados domínios de aplicação (OWENS et al., 2007). Devido ao seu baixo custo e alto grau de paralelismo, as placas gráficas estão sendo cada vez mais utilizadas para acelerar implementações que exigem grande quantidade de operações. A implementação do *Seam Carving* no hardware gráfico, tanto para redimensionamento de imagens, como para vídeos, pode proporcionar grandes ganhos de desempenho e facilitar o uso e experimentação da ténica, facilitando sua adoção como uma ferramenta padrão para edição de imagens e vídeos.

REFERÊNCIAS

AGARWALA, A.; DONTCHEVA, M.; AGRAWALA, M.; DRUCKER, S.; COLBURN, A.; CURLESS, B.; SALESIN, D.; COHEN, M. Interactive digital photomontage. **ACM Trans. Graph.**, New York, NY, USA, v.23, n.3, p.294–302, 2004.

AVIDAN, S.; SHAMIR, A. Seam Carving for Content-Aware Image Resizing. **ACM Transactions on Graphics, (Proceedings SIGGRAPH 2007)**, [S.l.], v.26, n.3, 2007.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 2.ed. Cambridge, MA: MIT Press, 2001. 595-601p.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 2.ed. Cambridge, MA: MIT Press, 2001. 323-369p.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 2.ed. Cambridge, MA: MIT Press, 2001. 563p.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. **CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1**, Washington, DC, USA, p.886–893, 2005.

DOTY, D. **Munkres Assignment Algorithm**. http://www.itl.nist.gov/div897/sqg/dads/HTML/ munkresAssignment.html, Website.

ITTI, L.; KOCH, C.; NIEBUR, E. A model of saliency-based visual attention for rapid scene analysis. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.20, n.11, p.1254–1259, 1998.

KUHN, H. W. The Hungarian method for the assignment problem. **Naval Research Logistic Quarterly**, [S.l.], v.2, p.83–97, 1955.

LIU, F.; GLEICHER, M. Automatic image retargeting with fisheye-view warping. **UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology**, New York, NY, USA, p.153–162, 2005.

OWENS, J. D.; LUEBKE, D.; GOVINDARAJU, N.; HARRIS, M.; KRüGER, J.; LEFOHN, A. E.; PURCELL, T. J. A Survey of General-Purpose Computation on Graphics Hardware. **Computer Graphics Forum**, [S.l.], v.26, n.1, 2007.

ROTHER, C.; BORDEAUX, L.; HAMADI, Y.; BLAKE, A. AutoCollage. **ACM Trans. Graph.**, New York, NY, USA, v.25, n.3, p.847–852, 2006.

RUBINSTEIN, M.; SHAMIR, A.; AVIDAN, S. Improved Seam Carving for Video Retargeting. **ACM Transactions on Graphics, (Proceedings SIGGRAPH 2008)**, [S.l.], v.27, n.3, 2008.

RUBINSTEIN, M.; SHAMIR, A.; AVIDAN, S. Multi-operator media retargeting. **SIGGRAPH '09: ACM SIGGRAPH 2009 papers**, New York, NY, USA, p.1–11, 2009.

SANTELLA, A.; AGRAWALA, M.; DECARLO, D.; SALESIN, D.; COHEN, M. Gaze-based interaction for semi-automatic photo cropping. **CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems**, New York, NY, USA, p.771–780, 2006.

SETLUR, V.; TAKAGI, S.; RASKAR, R.; GLEICHER, M.; GOOCH, B. Automatic image retargeting. **MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia**, New York, NY, USA, p.59–68, 2005.

VINEET, V.; NARAYANAN, P. CUDA cuts: fast graph cuts on the gpu. **Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on**, [S.l.], 2008.