Introdução ao Processamento de Imagens

Relatório Trabalho Final

Síntese e Transferência de Textura com *Image Quilting*

Felipe Oliveira Magno Neves
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
180016296@aluno.unb.br

***Resumo*—Neste trabalho, com objetivo de aplicar processos e técnicas aprendidos durante o semestre, será implementado o processo de transferência e síntese de textura *Image Quilting[1]*. Este é um processo de síntese e transferência de textura por *patches* completamente embasado em algoritmos de processamento de imagens – sem uso de aprendizado de máquina/visão computacional – com alta eficiência do ponto de vista de processamento, e com diversos pontos otimizáveis.**

I. Introdulção

A síntese e transferência de texturas é uma área do processamento de imagens e aprendizado de máquina que tem passado por grande crescimento nos últimos anos. Seja para aplicações artísticas, como na etapa de pós-produção de filmes, ou na otimização de renderizações em computação gráfica, a área tem despertado o interesse de um grande número de indivíduos e apresenta um problema ideal para aplicação de diversas técnicas estudadas no decorrer do semestre.

A motivação básica é a de, em vez de capturar toda a informação de uma imagem de grande tamanho, se use uma pequena quantidade de informação – uma subseção da imagem completa - para extrapolar (aproximar) uma imagem de tamanho desejado [2], seguindo os mesmos padrões de textura e extrutura.

Para tal, será utilizado um algoritmo símples que não faz uso de técnicas modernas de aprendizado de máquina. Aplicações mais novas, em geral, fazem uso de redes neurais convolucionais[3], pois estas tendem a gerar resultados mais generalizáveis e naturais. Por outro lado, foi escolhido o algoritmo *Image Quilting* por conta do número de oportunidades que o mesmo proporciona para uso dos conceitos estudados até então, e por não ser necessária uma etapa de treinamento, normalmente demorada e computacionalmente cara.

Será, também, abordada a forma com a qual algumas regras extras podem ser aplicadas à síntese de textura de forma a não somente criar uma imagem que siga os mesmos padrões da textura fonte, mas que também a disposição de elementos de uma imagem alvo, de forma a recirar uma imagem com as texturas presentes na outra.

Na seção II será explicado a fundo o funcionamento do algoritmo, assim como alguns detalhes da implementação que não são completamente explorados no artigo original [1].

Na seção III os resultados obtidos serão mostrados e compadados com os do artigo original. Além disso, serão mostrados alguns resultados inéditos.

Na seção IV o artigo será concluído ao discutir-se pontos fortes e limitações do algoritmo implementado, assim como possíveis otimizações a serem aplicadas.

II. Metodologia

Nesta seção será explicado o funcionamento interno do algoritmo *Image Quilting*, que consiste no processo de *quilting* propriamente dito, o corte de borda de erro mínimo e a adaptação do algoritmo de *quilting* para transferência de textura.

A. Quilting

Como explicado em mais detalhes no artigo *Image Quilting for Texture Synthesis and Transfer* [1], a motivação por trás deste algoritmo usar, diretamente, pedaços da imagem original para montar a imagem final vem do fato de muitas vezes, em algoritmos de síntese pixel a pixel, os pixels vizinhos serem completamente determinados pelo pixel inserido inicialmente, fazendo com que a avaliação pixel a pixel seja um gasto desnecessário de recursos.

Pedaços da textura original podem ser encaixados de diversas formas. Pode-se, por exemplo, repetir um mesmo pedaço de textura em toda a imagem de saída – que por motivos claros é uma péssima estratégia. É possível também escolher partes da textura original aleatoriamente e borrar o "encaixe" entre elas, o que ainda gera uma grade visível na imagem gerada.

A técnica de *quilting* consiste em se determinar uma borda para cada pedaço retangular recortado da textura original de forma que, ao encaixar pedaços na imagem de saída, estas bordas se sobreponham. Então, em vez de borrar – misturar – essas bordas, é escolhido um corte nesta área de sobreposição de forma a minimizar o erro na borda (um lado da borda será o mais parecido possível com o outro). Este é o processo de corte de borda de erro mínimo.

Uma vantagem gerada por esse processo é que agora teremos, quase sem gastar processamento extra, a medida da similaridade entre as bordas dos pedaços - *patches* – sendo avaliados. Essa medida é a base do algoritmo de *image quilting*: ao se encaixar um novo pedaço na imagem sendo gerada, em vez de escolhê-lo aleatoriamente é possível escolher o pedaço com o menor erro entre suas bordas e as bordas já existentes na imagem, garantindo melhores transições entre os *patches* escolhidos.

B. Corte de Borda de Erro Mínimo

O processo de geração do corte de borda de erro mínimo será explicado por meio de um exemplo. Para tal, vamos usar as seguintes imagens:

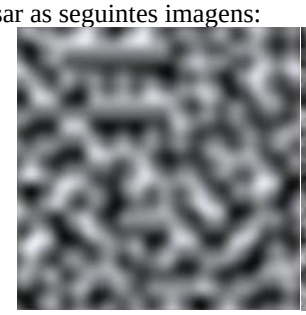

Figura 1

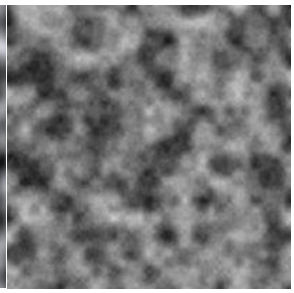

Figura 2

As imagens escolhidas são geradas por diferentes funções de ruído suave em 2D, encontradas online. Vamos supor que quer se encaixar as imagens na configuração acima (Figura 1 à esquerda da Figura 2).

Primeiro iremos isolar as bordas em questão:

Figura 3

Figura 4

Estas duas bordas devem ser usadas para gerar um mapa do erro entre elas. Para tal, basta fazer a diferença pixel a pixel entre elas. Foi feita a diferença em cada canal de cor separadamente, e a diferença total foi computada fazendo a raiz da soma dos quadrados das diferenças para cada canal. O resultado é, então, normalizado entre 0 e 255. O mapa gerado é o seguinte:

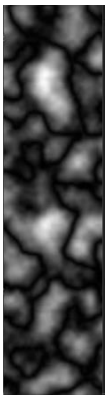

Figura 5 superfície de erro entre bordas

Agora quer-se calcular o caminho vertical que atravessa este mapa que passa pelas regiões de menor erro possível. Para o problema encontrado, força bruta é simplesmente inviável visto que este erro deverá ser calculado, normalmente, duas vezes por *patch* a ser colado na imagem final. Como mencionado no artigo original, é possível usar o algoritmo de Dijkstra para resolver esse problema. Ao implementar dito algoritmo percebeu-se que, mais que isso, esse problema pode ser reescrito de forma a ser resolvido por qualquer técnica que ache o menor caminho entre dois pontos em um grafo com pesos: basta interpretar a imagem de forma tal que cada pixel é um nó no grafo, pixels vizinhos estão conectados, e o peso da conexão para qualquer nó é o valor de brilho no pixel em questão. Não serão explicados muitos detalhes da adaptação feita do algoritmo de Dijkstra, mas vale a pena mencionar que foi feita uma otimização: em vez de poder-se andar para qualquer pixel vizinho, só é possível andar para os três pixels que se encontram na direção da borda oposta da imagem, isso diminui substancialmente o espaço de busca e, em geral, gera resultados bons o suficiente. Segue nas referências uma página web que provê a o processo de implementação do algoritmo básico em python, que serviu de base para a adaptação desenvolvida [5]. O resultado é o seguinte:

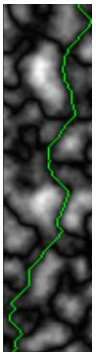

Figura 6 Corte de borda de erro mínimo

Vale a observação de que, na maioria das vezes, deve ser calculado o caminho corte de menor custo à esquerda e acima do bloco a ser inserido. Para tal, o mínimo entre os dois mapas de erro é escolhido para a parte superior da borda da esquerda e a parte esquerda da borda superior (a área de intersecção dos cortes). O caminho é então calculado separadamente para cada borda e, na etapa de corte do *patch*, se aplica os dois cortes (o que é equivalente a combiná-los até sua intersecção).

C. O Algoritmo

Agora temos todas as partes necessárias para aplicar o algoritmo de síntese de textura. Ele é composto pelos seguintes passos:

1) Percorra a imagem a ser sintetizada em ordem de rasterização em passos de tamanho de um patch – menos a borda.

2) Para cada local, procurar na textura fonte pedaços que satisfazem as condições de sobreposição.

3) Calcular o corte de menor custo entre o novo bloco e os existentes anteriormente, aplicar o corte e colar o novo bloco na imagem.

Como pode-se imaginar, as condições de sobreposição mencionadas na etapa 2 fazem grande diferença no resultado. Nesta implementação, em vez de se impor condições mínimas de sobreposição é simplesmente feita uma busca completa pelo melhor resultado – o resultado que gera o menor erro. Desta forma, podem ser explorados diversos critérios sem que sejam necessárias mudanças no algoritmo (como será explicado na seção D). Basta criar componentes

de erro limitadas entre 0 e 1, fazer a soma ponderada entre elas e escolher o patch que gera menor erro sobre essas condições. Vale a observação de que, na implementação feita, são salvas as opções que geram os K menores erros, de forma que seja possível escolher aleatoriamente um pedaço entre as K melhores opções.

Segue o exemplo do que ocorre no exemplo dado na seção B quando se aplica o corte na Figura 2 e depois se encaixa a mesma:

Figura 7 máscara de corte e aplicação de patch

Na seção III serão mostrados os resultados obtidos ao subdividir uma textura de entrada e preencher a saída como definido no algoritmo. Um detalhe importante é que, para a divisão da textura original em células de um tamanho específico, em vez de simplesmente cortar a imagem nessas partes, se percorre a imagem em intervalos de tamanho definido inferior ao tamanho de cada corte, salvando o conteúdo na área de tamanho do corte, de forma a ter vários cortes com regiões sobrepostas:

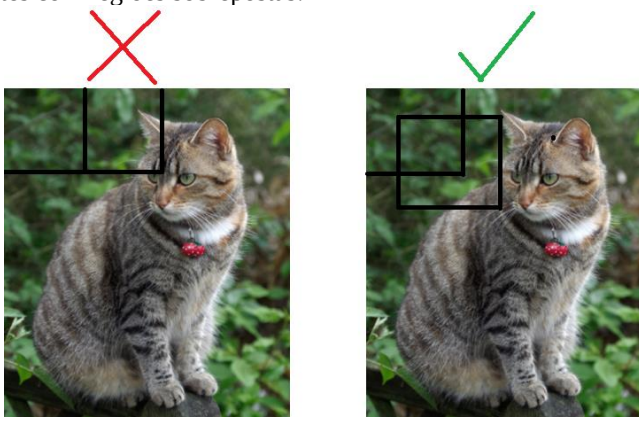

Figura 8 Ilustração da divisão da textura [4]

D. *Transferência de Textura*

Para adicionar transferência de textura ao algoritmo, basta criar uma componente de erro a ser inclusa no passo 2. Neste caso, criou-se mapas de transferência entre a imagem de textura e imagem alvo. Para gerar esses mapas, as imagens originais são passadas para tons de cinza e um filtro gaussiano é aplicada nas mesmas. Então, além da componente de erro de borda entre os *patches* aplicados, é calculada a diferença em nível de brilho nas áreas correspondentes dos mapas de transferência. O resultado é que se procura, na imagem de textura, áreas com o brilho similar ao presente na imagem que se quer recriar.

O erro total usado pelo algoritmo é, então, a soma ponderada dos dois erros, e é usado um termo alfa para controle da importância dada para cada componente – se importando mais com a continuidade entre os blocos aplicados ou com a fidelidade aos níveis de brilho da imagem original.

Como sugerido no artigo original, é aplicado um processo iterativo onde, a cada passo, se diminui o tamanho dos *patches* (blocos tirados da imagem de textura), e se aumenta o termo alfa de forma a inicialmente se importar mais com a fidelidade ao brilho, e gradualmente dar mais importância à continuidade das texturas. À cada passo o tamanho dos patches é reduzido em 1/3, e o termo alfa vai de 0.1 a 0.6 no decorrer dos passos (a variação em cada passo depende do número de repetições feitas).

III. Resultados

A. *Síntese de Textura:*

Seguem os resultados obtidos com algumas das texturas utilizadas no artigo original. Todos os resultados foram gerados usando 50 como o tamanho dos patches, dando passos de 1 pixel na textura original, com bordas de sobreposição com tamanho de ¼ do tamanho dos patches, escolhendo o melhor resultado encontrado (sem escolha aleatória entre os melhores).

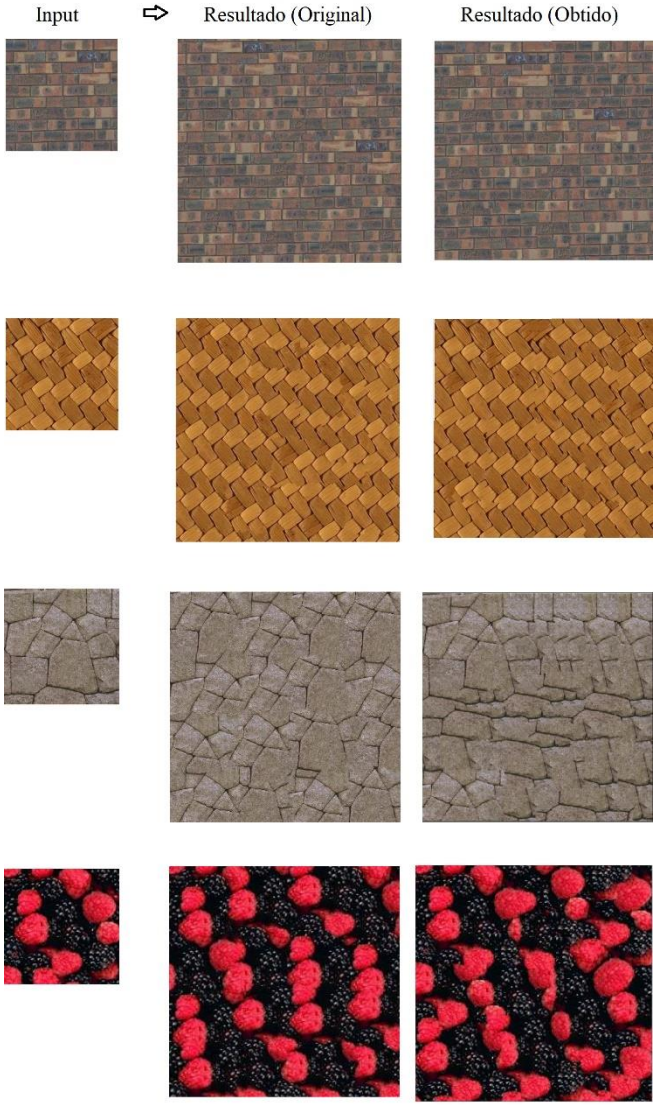

Figura 9 Comparação de resultados

Observa-se que, para síntese de textura, os resultados esperados e obtidos são extremamente similares. Seguem alguns exemplos executados com texturas não presentes no artigo original:

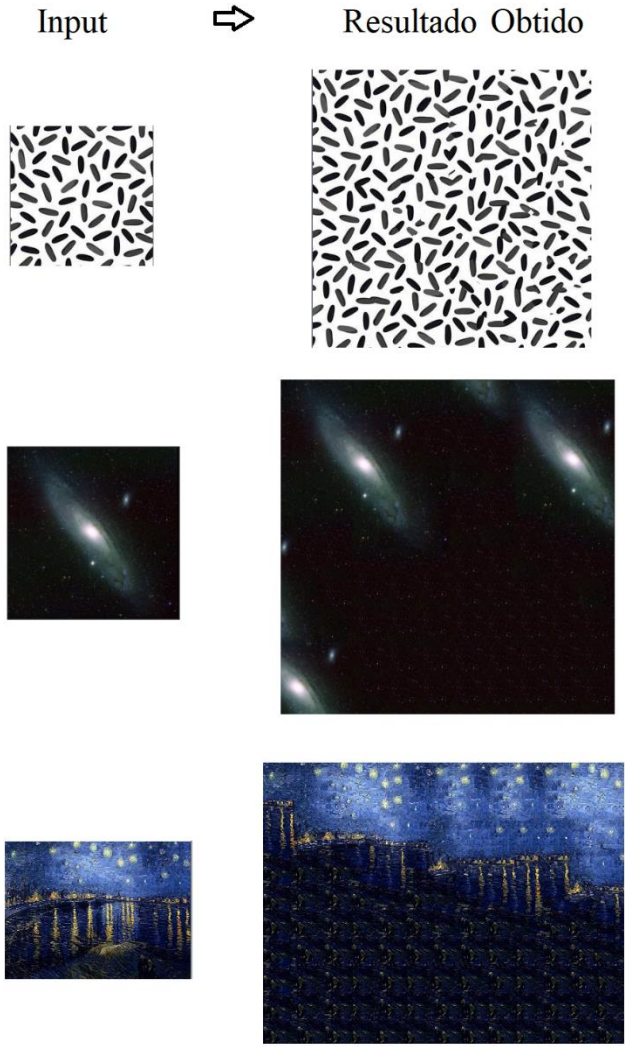

Figura 10 Resultados bônus em síntese de textura

B. Transferência de Textura:

No artigo original não existem muitas demonstrações diferentes de transferência de textura. Segue a comparação de resultados para uma delas:

Figura 11 Comparação de resultados para transferência de textura

Observa-se que, embora não tenha sido possível gerar um resultado exatamente como o original – principalmente pelo fato de os parâmetros exatos usados para a geração do resultado original não serem mencionados – a transferência de textura foi bem-sucedida (são visíveis linhas com poucas descontinuidades dispostas em formato similar à imagem original).

Seguem também alguns resultados obtidos com imagens escolhidas aparte:

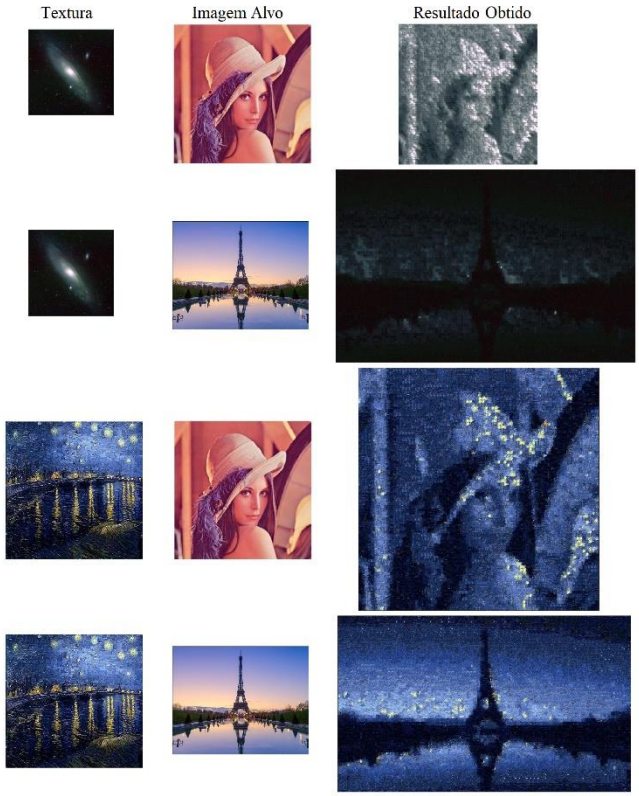

Figura 12 Resultados bônus em transferência de textura

IV. Conclusões

Os resultados encontrados foram surpreendentes para um algoritmo tão simples. Observa-se que a síntese de textura é excelente em texturas regulares e semi-regulares, e ao aplicar-se um pouco de aleatoriedade nas escolhas feitas, gera bons resultados até em texturas estocásticas.

O tempo de processamento pode vir a ser um problema ao se utilizar grandes imagens como fonte de textura. Por outro lado, é possível ajustar parâmetros (como o passo de divisão da textura e o tamanho dos patches) de fora a reduzir substancialmente o tempo de processamento sem que grandes pioras sejam vistas nos resultados.

Um problema encontrado é que, ao fazer transferência de textura, a textura escolhida deve ter grande variação de níveis de luminosidade de fora que possa ser encontrado um pedaço na mesma que possa ser "encaixado" na figura sendo gerada. Este problema pode, possivelmente, ser contornado caso seja feito algum processamento extra nas imagens de entrada ou nos mapas de transferência, como a equalização de histograma dos mesmos.

Boas sugestões de otimizações que podem ser aplicadas ao algoritmo são:

1) O uso de técnicas mais adequadas para encontrar o corte de erro mínimo – O algoritmo de Dijkstra modificado gerou bons resultados, mas é interessante investigar aplicação de algoritmos como A.*

2) A organização dos patches em ordem crescente de luminosidade de forma que seja possível fazer uma busca

binária ao procurar-se a melhor opção disponível para transferência de textura.

3) O uso de processamento paralelo na etapa encarregada de encontrar o patch com menor erro de inserção.

Em geral, os resultados encontrados foram extremamente parecidos com os esperados, e a implementação é considerada bem-sucedida, principalmente ao se levar em conta a otimização aplicada para encontrar-se o corte de borda de erro mínimo.

V. Referências

[1] Alexei A. Efros, and William T. Freeman, “Image Quilting for Texture Synthesis and Transfer”, 2001. Acessado em 25 de Outubro de 2021. Disponível em: https://en.wikipedia.org/wiki/Texture_synthesis#cite_note-1

[2] “Texture Synthesis”. Acessado em 25 de Outubro de 2021. Disponível em: http://graphics.cs.cmu.edu/people/efros/research/synthesis.html

[3] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[4] Kosma Kurlowicz, "Cat Pet Free Stock Image" Acessado em 26 de Outubro de 2021. Disponível em: https://stocksnap.io/photo/cat-pet-1CPSMUWHWX

[5] Ben Alex Keen, "IMPLEMENTING DJIKSTRA'S SHORTEST PATH ALGORITHM WITH PYTHON" Acessado em 20 de Outubro de 2021. Disponível em: https://benalexkeen.com/implementing-djikstras-shortest-path-algorithm-with-python/