Leandro Tonietto

Uma abordagem baseada em *texels* para síntese de texturas que variam progressivamente

São Leopoldo

2005

Leandro Tonietto

Uma abordagem baseada em *texels* para síntese de texturas que variam progressivamente

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Orientador:
Marcelo Walter

São Leopoldo

2005

AGRADECIMENTOS

Agradeço a todos os amigos e colegas (principalmente o pessoal do laboratório CROMOS) que me ajudaram na realização deste trabalho e também aos que me incentivaram e me apoiaram durante o curso de Mestrado.

Em especial, agradeço ao Prof. Dr. Marcelo Walter, meu orientador, pela sua incomparável dedicação, pelas orientações nos trabalhos publicados e por tudo o que eu pude aprender com ele na área de Computação Gráfica e na vida acadêmica.

Faço um agradecimento especial também aos meus pais, por que acreditaram em mim, pelo incentivo ao estudo e à formação profissional e também pela ajuda financeira quando foi preciso durante o curso.

E por fim, agradeço a minha esposa Juliana e minha filhinha Luísa pelo apoio, pela paciência e por se privarem de alguns momentos importantes para que eu conseguisse cumprir os prazos regimentais do curso.

RESUMO

Este trabalho apresenta um modelo para síntese de texturas a partir de amostras, mais especificamente para síntese de texturas com elementos que variam progressivamente ao longo da textura, conhecidas como *Progressively-Variant Textures (PVT)*. Este tipo de textura é comum na Natureza, particularmente na pelagem de animais como leopardos. Os algoritmos de síntese existentes, na grande maioria, não permitem qualquer controle sobre o resultado final, ou então, os que permitem, são limitados ou tem problemas de performance inerente ao método de síntese utilizado (pixel-a-pixel). Para resolvermos estes problemas, propomos um algoritmo baseado numa nova unidade de síntese: o *texel*. O nosso algoritmo sintetiza uma nova textura agrupando texels que satisfazem um critério de similaridade numa vizinhança. As características de uma PVT são obtidas com transformações afim e operações morfológicas aplicadas sobre os texels e definidas pelo usuário. Os resultados obtidos apresentam uma ótima qualidade visual para uma grande variedade de texturas de amostra, sobretudo em casos onde alguns algoritmos falham, como em texturas naturais. Além disso, as muitas possibilidades de variação sobre os texels que compõe a textura, proporcionadas pelas transformações e operadores morfológicos, até então não exploradas nos trabalhos anteriores, mostram o poder do nosso algoritmo para síntese de texturas.

palavras-chave: síntese de texturas, síntese de texturas com variação progressiva, síntese de texturas por preenchimento de texels

ABSTRACT

We present a model for texture synthesis from samples, particularly for synthesis of textures with smooth variation of the texture elements, known as Progressively-Variant Textures (PVT). This type of textures is common in Nature, specially in animal coat markings such as leopards. Current solutions, for the most part, do not allow much control by the user or the ones which do allow control are limited or have problems due to the pixel-at-a-time nature of synthesis. We address these problems with the introduction of an algorithm based on a new building block for synthesis: the texel. Our solution builds a new texture grouping texels which satisfy a neighborhood similarity criterion. The PVT features are introduced with affine transformations and morphological operators defined by the user and applied to the texels. The results show a good visual quality for a large number of sample textures, including cases where current solutions for natural textures fail. Besides, the many possibilities for variation of texels as building blocks, provided by the affine transformations and morphological operators, so far not explored in current solutions, illustrate the synthesis power of our algorithm.

keywords: texture synthesis, progressively-variant texture synthesis, texel-based texture synthesis

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CG - *Computação Gráfica*
RGB - *Red, Green, Blue (vermelho, verde e azul)*
RGBA - *Red, Green, Blue and Alpha (vermelho, verde, azul e alfa)*
PVT - *Progressively-Variant Texture*
TVP - *Texturas que Variam Progressivamente*
MRF - *Markov Random Fields*
VQ - *Vector Quantization*
TSVQ - *Tree-Structured Vector Quantization*

SUMÁRIO

1 INTRODUÇÃO

A Computação Gráfica exerce um fascínio em seus seguidores: ela, audaciosamente, simula visualmente fenômenos naturais. Uma das principais tarefas nesta busca pela imitação, é o uso de texturas para incrementar o realismo visual das cenas sintetizadas. Neste contexto, a textura refere-se a uma propriedade visual intrínseca dos objetos e que define de que eles são feitos (WALTER, 1998).

O uso de texturas em Computação Gráfica iniciou em 1974 com Catmull em (CATMULL, 1974). A sua utilidade e necessidade se justificam pelo altíssimo custo computacional caso os detalhes visuais fossem ser modelos geometricamente. A idéia básica é incluir estes detalhes através de uma imagem denomida mapa de textura. A transferência desta imagem para o objeto a ser texturizado é denomida mapeamento de textura. Este trabalho situa-se no contexto de mapeamento de texturas e mais especificamente na síntese de mapas de textura para simular detalhes visuais ao invés de computá-los com modelagem geométrica. Com isso, procura-se garantir o realismo dos resultados, sem comprometer o modelo geométrico e assim, a responsabilidade do sucesso do realismo no resultado final fica transferida para as texturas a serem utilizadas.

Agora o problema passa a ser outro: como adquirir texturas de boa qualidade visual e que contenham todos os elementos ou características necessárias para garantir um resultado visual adequado?

Existem duas maneiras de se obter texturas: através da digitalização de imagens reais e através de síntese de texturas. Certamente a aquisição por digitalização de imagens

do mundo real é a forma que proporciona o maior realismo para uma textura. É a maneira mais trivial de buscar o realismo direto da fonte. Entretanto, imagens adquiridas do mundo real possuem diversos problemas (HECKBERT, 1986) (na figura 2 pode-se notar alguns destes problemas):

1. Aquisição, disponibilidade e variação. Conseguir bons modelos reais e que estejam em posição de favorecer uma boa captura de textura é um desejo. Precisamos ainda conseguir uma boa variedade de posições dos próprios modelos para gerar a maior diversidade possível de resultados.

2. Iluminação e efeitos tridimensionais (3D) intrínsecos. As imagens já carregam informação de iluminação e efeitos 3D, como perspectiva e oclusão de formas. Isto pode comprometer o resultado, pois conforme a posição da câmera, ou ponto de visão do observador, a visualização ficará distorcida. Pontos que antes coincidiam com a incidência de luz ou efeito 3D, numa outra posição de câmera, não ficam adequados.

3. A forma geométrica do modelo fotografado. Para algumas texturas, a forma do objeto pode afetar a captura da imagem, além disso o que não está se observando não aparece na foto.

4. A resolução também é algo importante, pois conforme o tamanho do objeto, a quantidade de informações (resolução da textura), pode não ser suficiente.

Obviamente que várias soluções têm sido propostas na literatura para os problemas acima citados, particularmente na comunidade de Processamento de Imagens e Visão Computacional, com modelos analíticos para análise e síntese de texturas tendo sido apresentados desde o final dos anos 70 (LU; FU, 1978).

As necessidades de Computação Gráfica entretanto são de outra natureza e uma nova classe de soluções para síntese que utilizam amostras do mundo real como ponto de partida para a síntese surgiram particularmente a partir de 1999 com o trabalho de

Efros e Leung (EFROS; LEUNG, 1999). Até recentemente os resultados tinham um custo computacional muito alto ou as técnicas não eram genéricas o suficiente (HEEGER; BERGEN, 1995; de Bonet, 1997; SIMONCELLI; PORTILLA, 1998). A idéia de utilizar uma amostra para guiar o processo de síntese também esteve presente, como por exemplo em (FU; LU, 1978; LU; FU, 1978; MONNE; SCHMITT; MASSALOUX, 1981; CROSS; JAIN, 1983; GAGALOWICZ; MA, 1985).

Um sistema ideal deveria ser capaz de produzir artificialmente (sintetizar) texturas que sigam as propriedades visuais existentes no mundo real, que possam ser geradas com qualquer tamanho, com grande diversidade de resultados que possam ser facilmente controlados e que contenham somente as informações que delas se precisa: elementos característicos que dão o efeito visual 2D, sem efeitos 3D ou de iluminação que possam dificultar o mapeamento em um modelo geométrico.

Podemos classificar texturas sintetizadas em dois tipos: procedurais e imagens. As procedurais são provenientes de implementações de modelos teóricos, que normalmente simulam modelos matemáticos, físicos ou biológicos que expressam a realidade. Já imagens são provenientes de síntese a partir de amostras bidimensionais de outras imagens que tanto podem ser do mundo real ou criadas em computador.

As simulações de padrões naturais por métodos procedurais têm a vantagem de serem corretas sob o ponto de vista de modelos teóricos; portanto, permitem uma validação quantitativa. Entretanto, visualmente, muitas vezes estas texturas não são convincentes, isso sob o ponto de vista da validação visual, qualitativa. A figura 1 é um exemplo deste tipo de textura.

O uso de imagens diretamente como foram capturadas não é uma boa solução pelas razões enumeradas acima; gerar texturas procedurais, que são corretas na teoria, porém com falhas visualmente, também não. Encontramos a resposta para este problema em síntese de texturas a partir de amostras do mundo real.

A síntese de texturas a partir de uma amostra pode ser definida como: dada uma

Figura 1: Exemplo de textura procedural, gerado com modelo MClone (WALTER; FOURNIER; MENEVAUX, 2001)

textura de amostra S, gerar uma nova textura R, de tamanho qualquer, que contenha o mesmo padrão visual de S. A idéia de conter o "mesmo padrão visual" não é formalmente definida e usualmente este critério é avaliado subjetivamente.

A vantagem da síntese de texturas a partir de amostras sobre as outras técnicas, é que ela, através de um simples conjunto finito de elementos numa amostra de textura, pode gerar uma grande diversidade de resultados, em qualquer tamanho, sem comprometer a qualidade visual, visto que a informação de entrada é proveniente do mundo real.

Nos últimos 5 anos, diversos trabalhos foram publicados para reprodução de texturas a partir de amostras (EFROS; LEUNG, 1999), (WEI; LEVOY, 2000a), (LIANG et al., 2001), (EFROS; FREEMAN, 2001), (TONIETTO; WALTER, 2002), (ZHANG et al., 2003). Esses trabalhos, no geral apresentam variações de técnicas para simples reprodução da amostra. Entretanto, existem casos no mundo real que contêm um padrão de textura mais complexo, como é caso do padrão de pelagem do leopardo. Existe uma diversidade muito grande de elementos característicos de textura entre as extremidades do animal e

os elementos do abdômen (figura 2). São texturas que apresentam diferentes elementos característicos dependentes da sua localização no objeto.

Portanto, além de sintetizar a textura, um bom modelo para este tipo de padrão deve fazer controle local de síntese, ou seja, conforme a localidade na imagem resultado, o elemento característico sintetizado varia de alguma forma, seguindo um padrão.

Recentemente, Zhang et al. (ZHANG et al., 2003) publicaram um modelo que é capaz de gerar imagens que são homogêneas numa pequena localidade, mas que variam progressivamente no todo a partir de uma simples amostra. Este trabalho dá um novo ímpeto da pesquisa em síntese de texturas, pois apresenta uma nova classe de texturas, aquelas cujos elementos característicos, que formam o padrão visual, variam progressivamente ao longo da imagem e que ficaram conhecidas como *Progressively-Variant Textures* (PVT), ou Texturas com Variação Progressiva (TVP). E isto abre as portas para novas contribuições e desenvolvimento de soluções mais sofisticadas que as anteriores. A figura 2 é um exemplo natural deste tipo de textura.

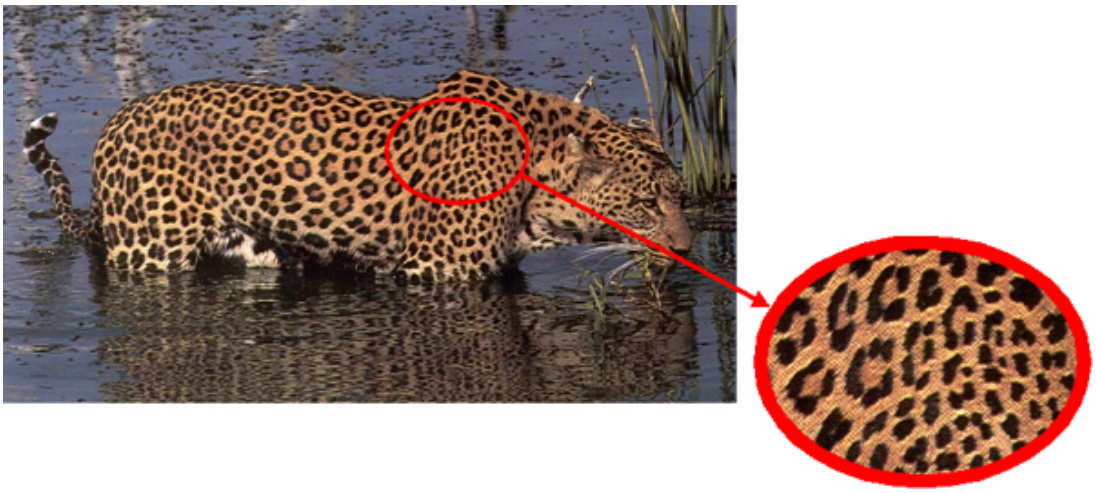

Figura 2: Em detalhe, um exemplo da diversidade de elementos característicos no padrão de pelagem do Leopardo (foto de (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991)).

A implementação de um modelo que reproduza texturas baseadas em síntese a partir de imagens amostras, com controle de localidade, é a realização de sonho antigo: resolver o problema de texturas como o padrão de pelagem do leopardo. Esta é a grande motivação para a realização desta pesquisa.

1.1 Objetivos

O objetivo principal deste estudo é a síntese de texturas que variam progressivamente no todo a partir de amostras de texturas que são visualmente homogêneas numa pequena vizinhança. Estas texturas são conhecidas com PVT (ZHANG et al., 2003) e ocorrem, por exemplo, em pelagem de animais.

A partir desta idéia desenvolvemos um modelo e, consequentemente, uma ferramenta que reproduz uma textura 2D capturando as diversidades de elementos característicos de uma amostra de textura da classe PVT.

O modelo apresentado nesta dissertação fundamenta-se nas idéias apresentadas em (ZHANG et al., 2003), propondo uma maneira alternativa de síntese de texturas PVT, apresentada no capítulo 4. A idéia principal de modelo é utilizar os texels [1] como bloco fundamental de síntese.

1.2 Organização da Dissertação

A dissertação está organizada da seguinte forma: no capítulo 2 serão descritos alguns conceitos importante sobre texturas e imagens; no capítulo 3 são apresentados alguns trabalhos relacionados e a contextualização do trabalho no estado da arte da síntese de texturas; no capítulo 4 será exposto nosso modelo de síntese de texturas PVT e também texturas simples (não PVT); no capítulo 5 são apresentados diversos resultados de síntese de texturas PVT e não PVT, obtidos com o nosso sintetizador; e, por fim, faremos uma discussão final sobre o modelo, os resultados e trabalhos futuros no capítulo 6.

[1] Texel é um elemento do padrão visual da textura. É composto por um conjunto de pixels que formam uma estrutura caracterísctca da textura.

2 CONCEITOS BÁSICOS

Este capítulo apresenta conceitos que são utilizados no desenvolvimento deste trabalho.

2.1 Texturas

Esta seção apresenta alguns conceitos relacionados à texturas. O que são? Quais são suas propriedades? Quais os tipos relevantes para este trabalho? E qual a diferença entre textura e imagem? São algumas questões que abordamos aqui.

Uma definição útil para os nossos propósitos diz que uma textura é a realização de um processo estocástico e estacionário (WEI; LEVOY, 2000a). Um conjunto de pontos randomicamente espalhados por um domínio bidimensional e que, numa determinada vizinhança, são estacionários (sem variação). Uma definição mais genérica é de que a textura é um caso particular de imagem com algum padrão de repetibilidade de elementos característicos. A figura 3 mostra um exemplo de textura.

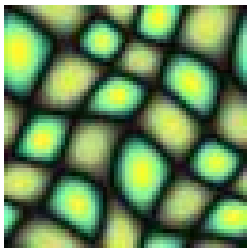

Figura 3: Exemplo de textura. Note que algumas estruturas se repetem no domínio.

Computacionalmente, uma imagem é definida como uma coleção de pontos arranjados num espaço bidimensional que carregam uma informação de cor associada a eles,

num determinado sistema de cores. Estes pontos recebem o nome de pixels. Normalmente, e para os casos utilizados neste trabalho, a informação que os pixels carregam é uma tupla (r, g, b) ou elemento do sistema de cores RGB (FOLEY et al., 1990).

O sistema de cores RGB é um espaço tridimensional composto por elementos que são uma combinação de três componentes básicos de cor: vermelho (*R - red*), verde (*G - green*) e azul (*B - blue*). Com estas três componentes é possível gerar um conjunto de cores útil para uso em tarefas de computação gráfica.

Quando estamos falando genericamente de imagens, sempre utilizamos a unidade mínima como pixels, pois é o tipo de registro que forma uma imagem. Entretanto, quando falamos em texturas, aparece um novo conceito de unidade, os elementos característicos que formam o padrão da textura. Em (EFROS; LEUNG, 1999) os autores sugerem o termo *texels*, como nomenclatura para elementos característicos que formam o padrão da textura. Assumiremos que os *texels* são o principal tipo de informação que formam as texturas. Para exemplificar o conceito de *texels* veja a região da imagem em destaque na figura 4.

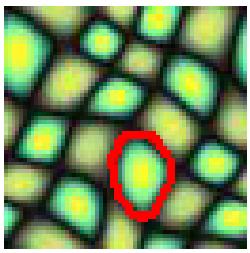

Figura 4: Exemplo de *texel* numa textura

Também de (EFROS; LEUNG, 1999) obtivemos uma classificação das texturas em dois tipos: estocásticas e regulares, de acordo com a aleatoriedade dos valores de seus pixels. As texturas regulares são aquelas mais estruturadas, com formas geométricas razoavelmente bem definidas, como linhas, retângulos, círculos em alta freqüência, e estocástica nos elementos com baixa freqüência. Já as texturas estocásticas são aquelas cujos elementos não são tão geometricamente bem definidos, ou seja, quanto mais ruído tem a textura mais estocástica ela é. O caso extremo para uma textura estocástica é uma

textura com apenas "*noise*" (ruído). A figura 5 exemplifica esses dois tipos de textura.

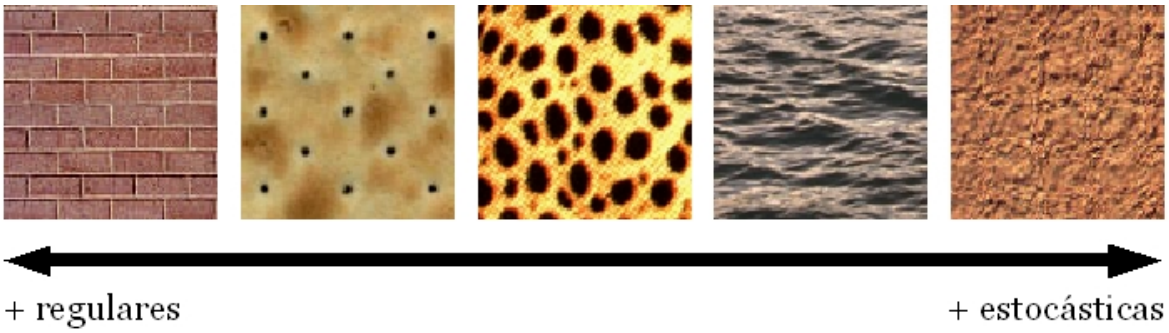

Figura 5: Exemplos de texturas regulares e estocásticas.

Outro ponto importante é a diferenciação entre texturas e imagens (figura 6 de (WEI; LEVOY, 2000a)). Vamos considerar duas imagens quaisquer como as apresentadas na figura 6. Se selecionarmos duas porções de cada imagem em diferentes posições, como ilustrado na figura, as duas porções (b1 e b2) em (b) são percebidas como similares, assim como em qualquer outra posição em (b). Entretanto, isto não acontece em (a). Isto é explicado pelas propriedades das texturas: estado estacionário e localidade.

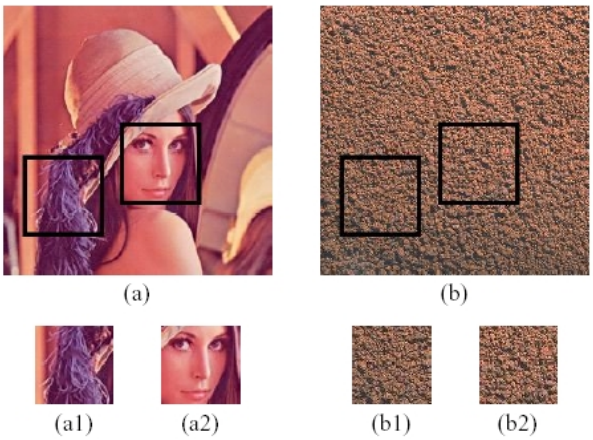

Figura 6: Ilustração de o quanto uma imagem qualquer se diferencia de uma textura em (WEI; LEVOY, 2000a).

Uma imagem, para ser uma textura, deve satisfazer estas duas propriedades. Uma imagem possui a característica de localidade se cada pixel na imagem é relacionado apenas ao pequeno conjunto de pixels ao seu redor (conceito de vizinhança). Uma imagem é dita estacionária se, para qualquer porção desta imagem, ela se apresenta similar a qualquer outra. Nota-se que o conceito de *ser similar* não é formalmente introduzido. Da mesma

forma o conceito de localidade faz mais sentido no momento de definir processos que possam gerar texturas, já que o conceito de um determinado pixel ser relacionado apenas ao conjunto de pixels ao seu redor é difícil de ser quantificado.

3 REVISÃO BIBLIOGRÁFICA

Desde o começo da utilização de texturas em computação gráfica em (CATMULL, 1974), diversas técnicas foram apresentadas para solucionar o problema de aquisição de texturas, desde síntese de texturas através de procedimentos (EBERT et al., 2002) até síntese a partir de amostras de imagens. Nestes últimos anos, a síntese de texturas a partir de amostras, em especial, ganhou bastante destaque em Computação Gráfica, devido ao grande número de publicações relacionadas ao assunto. Para mostrar a evolução destas técnicas, apresentamos na figura 7 uma linha do tempo com os principais trabalhos publicados nos últimos anos e suas contribuições. Em geral, os avanços ocorreram no sentido de melhorar a qualidade visual e performance dos algoritmos. Somente em (ZHANG et al., 2003) é que surge a definição de um novo conceito em texturas, a classe de texturas com variação progressiva, mudando o rumo das pesquisas em síntese de texturas e deixando alguns pontos em aberto.

Depois de revisar diversos trabalhos em síntese de texturas a partir de amostras, foi possível classificá-los em duas categorias ou abordagens: os que sintetizam pixel-a-pixel e os que sintetizam por blocos da amostra. Além destas categorias, podemos separar os trabalhos de acordo com as texturas sendo geradas: as homogêneas e as texturas que variam progressivamente, que podem resolver padrões visuais como o da pelagem de animais. A figura 7 mostra a evolução dos trabalhos desde (EFROS; LEUNG, 1999) (considerado como ponto inicial para dissertação).

O começo é com a abordagem pixel-a-pixel, onde Efros e Leung (EFROS; LEUNG, 1999) utilizam o conceito de MRF (*Markov Random Fields*) para geração de texturas. Wei

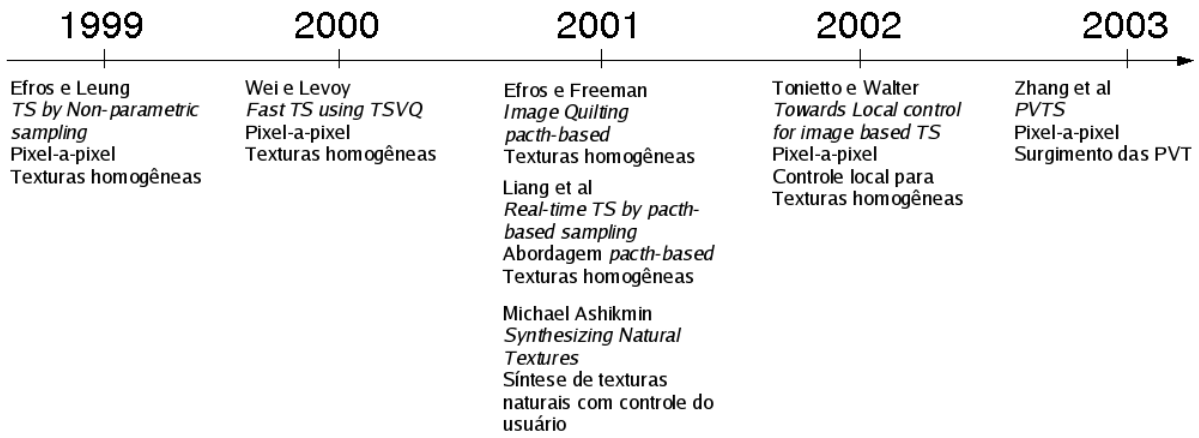

Figura 7: Com o tempo, as técnicas foram se dividindo em duas abordagens e ainda criou-se uma classe especial de texturas, as PVT.

e Levoy (WEI; LEVOY, 2000a) também utilizaram este conceito e propuseram otimizações para melhorar a performance de síntese.

Em 2001, Efros e Freeman (EFROS; FREEMAN, 2001) sugerem uma abordagem por blocos, tendo em vista a performance do algoritmo e explorando o fato de que a "mesma" informação era copiada, não fazendo sentido procurar por todos os pixels da imagem. Liang *et al.* (LIANG et al., 2001) conseguiram um algoritmo mais eficiente com o mesmo propósito.

Em 2001 Ashikhmin em (ASHIKHMIN, 2001) é o primeiro algoritmo a propor controle do resultado para o usuário. Tonietto e Walter (TONIETTO; WALTER, 2002) abordam o tema de controle local na síntese da textura, num algoritmo baseado em (WEI; LEVOY, 2000a) que modifica o tamanho da amostra conforme o local. A terceira e última abordagem, mais diretamente relacionada com o assunto da dissertação, surge em 2003 com Zhang *et al.* (ZHANG et al., 2003). Os autores propuseram um algoritmo para sintetizar texturas com variação progressiva, até então um problema em aberto.

A seguir apresentamos a revisão destes trabalhos dividida em três seções: métodos pixel-a-pixel, métodos de preenchimento por blocos e a nova classe de texturas, PVT. Aparecem também outros trabalhos considerados relevantes para a revisão, mas que não estão ligados diretamente aos objetivos deste trabalho. Por este motivo estão em menor

destaque.

3.1 Métodos Pixel-a-Pixel

Efros e Leung em 1999 (EFROS; LEUNG, 1999) propuseram um algoritmo de síntese de texturas baseado num modelo estatístico não paramétrico para síntese. O modelo utilizado foi o MRF (*Markov Random Field*) para determinar as possibilidades de combinação para um pixel, considerando a localidade espacial do mesmo como critério de seleção. A localidade espacial de um pixel é determinada pela sua vizinhança de pixels, que num determinado tamanho, captura um elemento característico local (*texel*) que representa a textura. Esta vizinhança é formada por uma janela de dimensões w ao redor do pixel corrente, onde w é um parâmetro informado pelo usuário.

A determinação do tamanho desta janela varia de acordo com o tamanho dos elementos mais característicos da textura de amostra (*texels*). Eles indicam que: "o tamanho da janela é um parâmetro livre que especifica o quão estocástica o usuário acredita ser esta amostra. Mais especificamente, se a textura é assumida como sendo principalmente regular, com freqüências espaciais altas; ou principalmente estocástica, com freqüências espaciais baixas, o tamanho da janela deveria ser na escala da maior característica regular" (EFROS; LEUNG, 1999).

Resumidamente, o algoritmo consiste em, para cada pixel a ser sintetizado, procurar e armazenar todas as possíveis combinações de vizinhança para este pixel que estejam dentro do limite de similaridade; depois uma destas combinações é aleatoriamente escolhida e o valor do pixel será igual ao deste sorteado. A combinação é determinada pela comparação da vizinhança do pixel a ser sintetizado com cada vizinhança de cada um dos pixels da amostra.

A métrica de distância ou similaridade entre vizinhanças é a seguinte:

$$d(w(p), w) < (1 + e) * d(w(p), wbest)$$

Onde a função d calcula a distância entre dois blocos de pixels, com a soma normalizada do quadrado das diferenças (norma L); $w(p)$ representa a vizinhança do pixel corrente, w é a vizinhança candidata na amostra, $wbest$ é a melhor vizinhança, previamente pesquisada, na amostra para o pixel corrente e "e" é o erro máximo permitido (utilizaram $e = 0.1$).

Para valorizar mais o aspecto de localidade na comparação, eles atribuíram pesos diferentes para cada posição da matriz de vizinhos, como kernel Gaussiano 2D, quanto mais próximo do centro, maior influência da diferença em relação às demais, quanto mais longe, menor a influência da distância no cálculo final.

Os resultados apresentados são bons, sobretudo para os casos de preenchimento de "buracos" em imagens. O algoritmo consegue gerar uma grande variedade de texturas naturais, entretanto, é muito lento e quanto maior a amostra o tempo de processamento aumenta numa progressão geométrica. Além disso, o algoritmo pode falhar com determinados tipos de texturas, convergindo para um local ruim da amostra e coletando apenas "lixo" para imagem, não conseguindo mais convergir para um bom resultado; ou então, os resultados apresentam muitas cópias de uma mesma região apenas. Este problema é atribuído à grande variedade de *texels* dentro da amostra, tornando difícil encontrar boas combinações, limitando o número de possibilidades para sorteio. Na figura 8 algumas texturas de amostra e seus respectivos resultados.

Wei e Levoy (WEI; LEVOY, 2000a) publicaram um algoritmo semelhante ao utilizado por Efros e Leung (EFROS; LEUNG, 1999), mas com algumas sofisticações como, por exemplo, não criam uma distribuição de probabilidade explícita para cada combinação, além de soluções para detectar estruturas maiores com maior facilidade e de aceleração do processo como um todo.

De acordo com as propriedades do modelo MRF, eles afirmam que o processo de síntese é uma realização de um processo randômico local e estacionário. Uma imagem é classificada como uma textura se ela for local e estacionária. Ela é estacionária se,

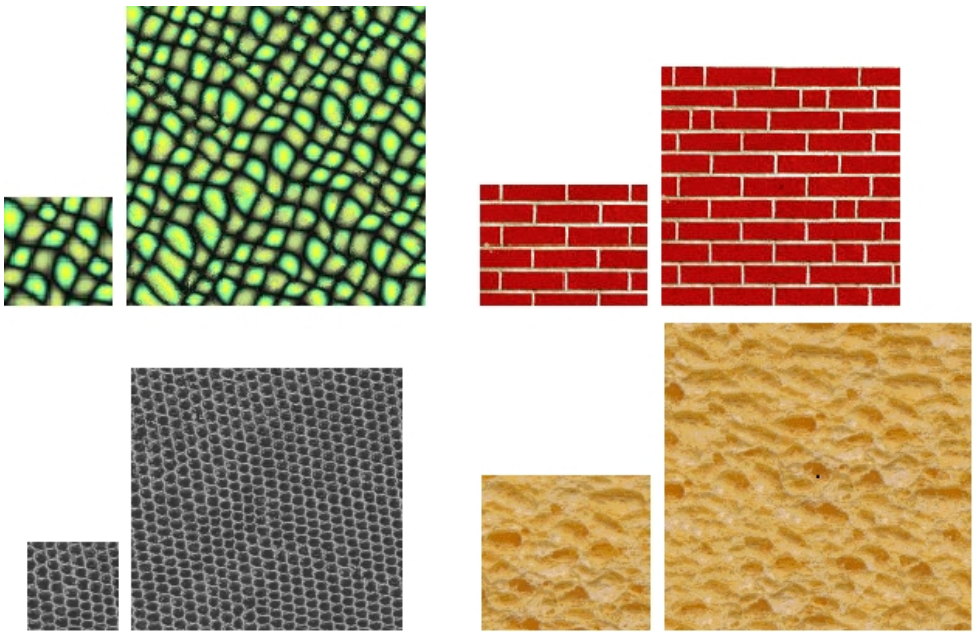

Figura 8: Resultados obtidos por Efros e Leung em (EFROS; LEUNG, 1999). Para todas as texturas, sempre à esquerda a amostra e à direita o resultado.

analisando-se uma porção qualquer da mesma, independentemente de posição, este pedaço é similar a qualquer outro dentro da imagem. Uma imagem é dita local, quando para cada pixel da imagem ele é relacionado apenas a um pequeno conjunto, ao seu redor, ou seja, sua vizinhança.

Baseado nestas definições eles criaram o algoritmo de resolução simples para sintetizar uma textura. O algoritmo pode ser resumido como: dada uma textura de amostra, uma imagem tipo "noise" para inicialização e um parâmetro de tamanho da vizinhança - um valor em que seja possível capturar as características da textura - o valor do pixel de saída será aquele que possuir a melhor combinação de vizinhança na amostra. Alguns pontos relevantes sobre o algoritmo:

- São utilizados apenas os pixels já sintetizados para formar a vizinhança do pixel, para evitar que pixels não sintetizados, com valores inválidos, possam introduzir informações errôneas na comparação e, consequentemente, produzir um resultado ruim;
- A imagem é sintetizada toroidalmente, de forma que se mantenha repetibilidade da

textura (*tieliable*);

- A imagem inicial, noise, é forçada para ter valores de pixel semelhantes aos da amostra, pela equalização de histogramas. Isso é importante para que as primeiras linhas, que não possuem valores sintetizados de pixels, contenham alguma informação mais próxima da amostra e assim, se consiga montar a vizinhança destes pixels;

- Métrica de comparação entre vizinhanças é a norma L2 dos valores (R,G,B);

- O tamanho da vizinhança ao redor do pixel é um parâmetro livre. Contudo, o tamanho da vizinhança não pode ser nem tão pequeno que não capture as estruturas relevantes da amostra, nem tão grande que possa produzir alta repetibilidade de padrões e prejudicar na aleatoriedade do resultado final. Como critério para definição da vizinhança, é indicado que seja do tamanho da maior estrutura regular da textura de amostra, de forma que se consiga capturar todas as suas estruturas básicas.

A figura 9 mostra o processo de síntese pelo algoritmo simples:

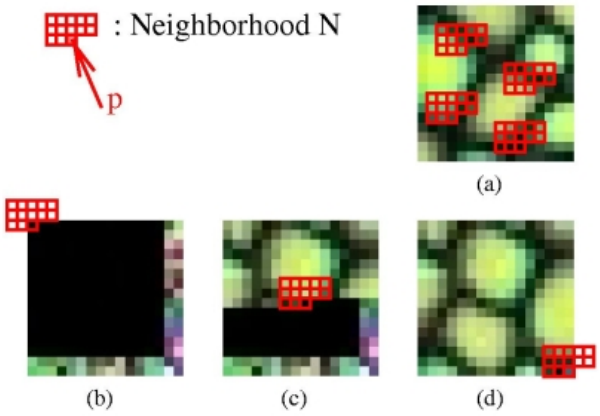

Figura 9: Processo de síntese em resolução simples apresentado em (WEI; LEVOY, 2000a). A imagem é sintetizada toroidalmente, por isso os pixels de vizinhança dos pixels das primeiras linhas e colunas são relativos nas últimas linhas e colunas (b). Para cada pixel na imagem de saída é feita uma verificação de vizinhança com todos os pixels da amostra (a). Note que apenas as duas últimas linhas de "*noise*" (inválidos ou ainda não processados) são utilizadas (b), para os demais pixels (c), somente pixels válidos são utilizados. Assim como ocorre nas primeiras linhas, para manter a propriedade toróide, os pixels das últimas colunas utilizam como vizinhos pixels das primeiras colunas (d).

O algoritmo apresenta bons resultados para uma grande variedade de texturas, porém o tempo de processamento é muito grande, podendo levar minutos, horas ou até dias, conforme o caso. Um dos motivos, além da busca exaustiva, inerente ao processo, é o fato de que para determinadas amostras o tamanho da vizinhança precisa ser muito grande, consequentemente, aumentando muito o tempo de processamento.

Como solução para este problema os autores propuseram um algoritmo que trabalha com multiresolução (*multiresolution synthesis*). A idéia básica é tratar as imagens em diversos níveis, onde cada nível contém as informações da imagem numa resolução diferente das demais, da mais alta até a mais baixa. Com isso a busca nos níveis de resolução mais baixos acelera a pesquisa, pois fornece uma pré-seleção de região a ser procurada para os níveis de resolução maior, conseguindo assim, que as estruturas maiores sejam encontradas mesmo com um tamanho de vizinhança menor que o esperado.

O algoritmo acelerado é quase o mesmo, exceto pela busca em níveis de resolução diferentes. Esses níveis são montados como uma pirâmide Gaussiana e todo pixel tem relacionamento com outro nos níveis.

Um dado interessante é que nos resultados apresentados no artigo eles utilizam, em geral, de 2 a 4 níveis de resolução apenas, isso para não prejudicar a qualidade do resultado final, pois a cada nível de resolução inferior, são perdidas muitas informações de pixels. Ainda assim, o modelo sofre com a busca exaustiva, pois para cada pixel a ser sintetizado é feito uma busca por todos os pixels da amostra (no caso do *single resolution*) ou pelo menos, de uma boa parte deles (no caso do *multi-resolution*).

Para aceleração propuseram ainda uma outra solução, que equilibra qualidade no resultado final com ganho de performance com o algoritmo de busca por quantização vetorial de árvore estruturada (*TSVQ - tree-structured vector quantization*). Os resultados são muito bons, principalmente sob o ponto de vista da performance do algoritmo. Com esta aceleração TSVQ ampliam-se os horizontes de aplicações que necessitam de síntese de texturas, mas ainda assim o algoritmo pode falhar em alguns tipos de texturas naturais e

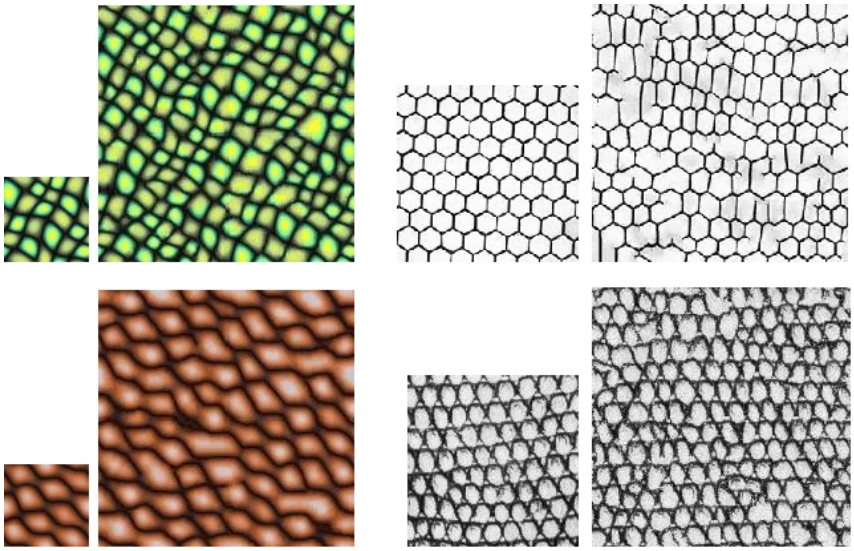

Figura 10: Resultados obtidos por Wei e Levoy em (WEI; LEVOY, 2000a). Para todas as texturas, sempre à esquerda a amostra e à direita o resultado.

até em algumas texturas regulares e o consumo de memória para a geração dos *codebooks* para aceleração também são problemas que ainda perduram.

Os dois algoritmos apresentados acima, apesar dos bons resultados visuais, não proporcionam nenhum tipo de controle sobre os resultados.

Em 2001, Ashikhmin (ASHIKHMIN, 2001) identificou que na classe de texturas naturais (como folhas, galhos, flores, chão de floresta, e pelagem de animais) o algoritmo WL (Wei e Levoy (WEI; LEVOY, 2000a)) não funciona bem. Estas texturas são caracterizadas como um arranjo de elementos distintos de formas e tamanhos irregulares, porém similares. O autor destaca que o problema para identificação destes elementos está principalmente relacionado com a métrica utilizada para critério de similaridade, a norma L_2. Este critério não se mostra adequado para identificar cantos, bordas e outras características em alto nível.

Ele propôs um novo algoritmo de síntese que estende o algoritmo de síntese proposto em (WEI; LEVOY, 2000a). O algoritmo antes de sintetizar um pixel, monta uma lista de candidatos baseados nos pixels da vizinhança deslocados. A partir destes candidatos o algoritmo elege um com a menor diferença (norma L_2). O resultado final é uma cópia de regiões irregulares da textura da amostra para a nova textura.

Entretanto, o motivo para colocamos este trabalho no escopo desta dissertação é outro: o controle à nível de usuário do resultado final. O algoritmo possibilita, através de uma imagem de destino (*target*), que o resultado da síntese seja reconhecido como tendo o mesmo padrão visual da amostra, que de alguma forma foi influenciado pela imagem *target*. O autor menciona que as imagens target podem ser quaisquer imagens desenhadas, por exemplo, utilizando um sistema de pintura simples.

Para conseguir a característica das duas imagens (textura de amostra e imagem de destino), o algoritmo preenche a vizinhança do pixel atual com informação das duas imagens. Para os pixels já sintetizados da vizinhança (parte superior da matriz), o sistema coloca pixels candidatos da textura de amostra. Para os pixels não válidos ou ainda não sintetizados da vizinhança (da parte inferior da matriz), o sistema coloca pixels provenientes da imagem destino seguindo o processo de síntese explicado acima. Caso necessário, novos passos de síntese podem ser realizados, utilizando como imagem *target* o resultado anterior. O resultado final fica com as características locais da amostra e com a aparência ou forma global da imagem de destino. A figura 11 de (ASHIKHMIN, 2001) exemplifica um resultado desta abordagem.

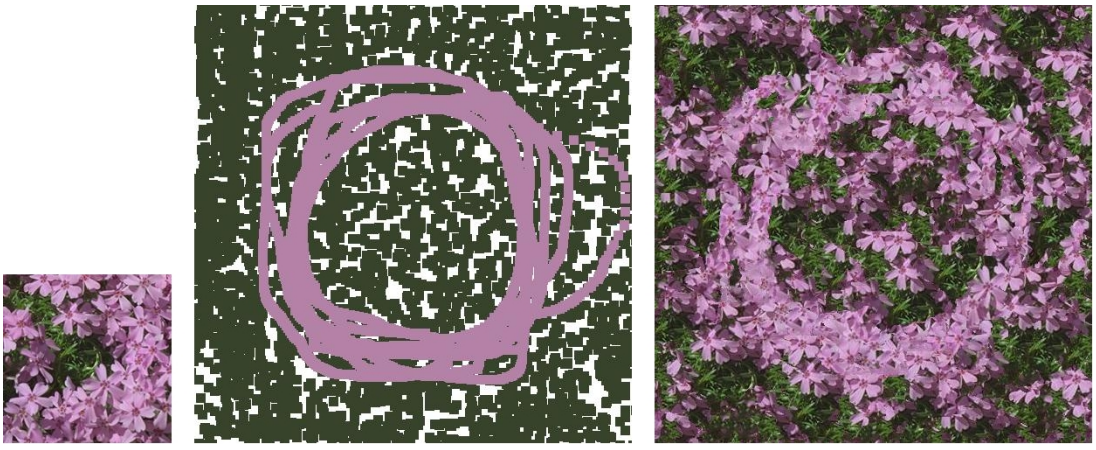

Figura 11: Exemplos de síntese com controle de usuário em (ASHIKHMIN, 2001). À esquerda a amostra, no centro a imagem de destino e à direita o resultado da síntese.

Afim de estabelecer um controle local de geração de textura através de síntese, Tonietto e Walter (TONIETTO; WALTER, 2002), propõem um algoritmo baseado em (WEI; LEVOY, 2000a) para sintetizar uma nova textura variando o tamanho das estruturas

conforme o local de síntese.

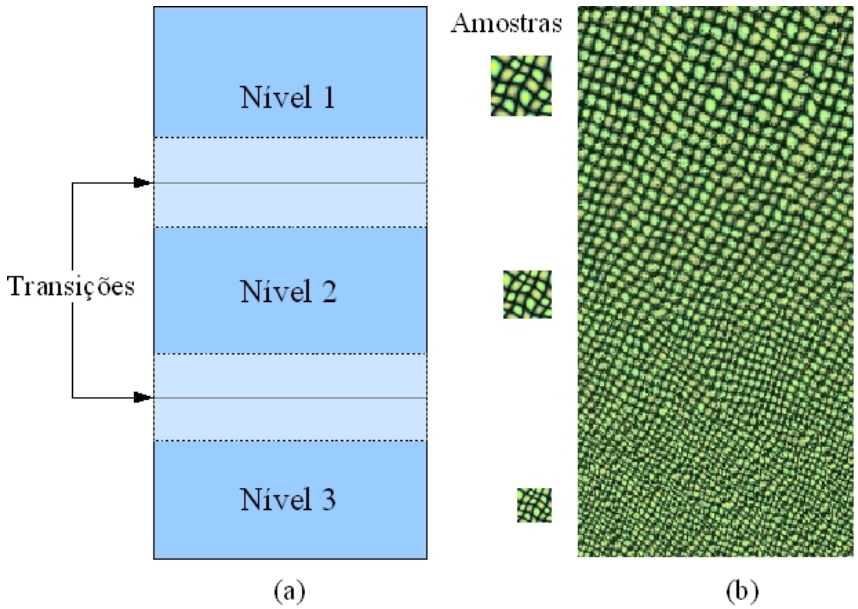

Figura 12: Ilustração do tratamento de transição entre faixas de síntese por escolha aleatória com probabilidade (TONIETTO; WALTER, 2002). Em (a) as faixas e as regiões de transição e em (b) um resultado e as amostras referentes a cada nível de síntese.

Para fazer este controle propuseram a divisão da imagem final em níveis de diferentes tamanhos de texturas. Para cada nível criado, é obtida uma instância da textura de amostra em resolução mais baixa que a do original. O usuário apenas especifica o número de níveis e o tamanho do último nível. O sistema calcula o tamanho de cada amostra para os demais. Por exemplo, se são três níveis e o último utilizará uma amostra de $0,3$ do tamanho original, considerando que a maior é sempre $1,0$, o segundo nível utilizara um tamanho $t = (1 - 0,3)/2 + 0,3$ do tamanho da amostra original.

O algoritmo sintetiza pixel-a-pixel em *raster scan-order*, da esquerda para a direita, de cima para baixo. Para estabelecer os níveis o algoritmo "divide" a imagem em n partes iguais, onde n é o número de níveis determinado pelo usuário. A cada linha sintetizada o algoritmo verifica em qual nível o pixel atual está e usa a amostra do nível apropriado para síntese.

Para evitar que o resultado mostre nitidamente uma mudança brusca de nível, fizeram também um controle de transição entre os níveis. Este controle de transição foi

testado de duas maneiras: por interpolação ou por escolha randômica de um pixel de acordo com uma probabilidade de estar mais próximo de um nível ou de outro.

O usuário determina o tamanho da área de transição entre os níveis (p.e. $t = 40$ pixels) e, quando o algoritmo está sintetizando pixels dentro de uma área de transição, o valor que será copiado é determinado de acordo a posição (linha) do pixel atual.

No caso da transição por interpolação de pixels, o algoritmo sintetiza o pixel em ambos os níveis e determina o valor do pixel com $(y/t) * P(L_i) + (1 - y/t) * P(L_{i+1})$. Desta forma, o valor final do pixel tem um contribuição de cada nível.

Já no controle por escolha randômica, o algoritmo faz um sorteio para determinar de qual nível colocar o valor para o pixel. Conforme a posição da linha sendo sintetizada o valor será copiado de acordo com a probabilidade de ele estar próximo de um nível da imagem $(1 - y/t)$ ou de outro (y/t, onde y é linha atual).

Quanto às duas abordagens de controle de transição entre níveis, o controle por escolha aleatória com probabilidade se mostrou mais eficiente do que o controle por interpolação. Neste último, na maioria dos casos, a transição resultou em um *blur* (borrado) indesejado. Enquanto que a abordagem por sorteio não apresentou o mesmo problema.

Apesar de atingir seu objetivo de sintetizar texturas com controle local, este algoritmo ainda não é suficiente para sintetizar texturas com variação progressiva, pois o controle apresentado se resumiu apenas ao tamanho de amostra que seria utilizado na síntese, consequentemente, diminuindo também o tamanho das estruturas sintetizadas. A figura 12 ilustra as faixas de transição e um resultado do algoritmo, que consegue transições visualmente contínuas e nos tamanhos dos pixels sintetizados.

3.2 Métodos de Preenchimento por Blocos

Os algoritmos apresentados até então utilizam como unidade de síntese o pixel. Esta é a menor unidade sendo sintetizada. Em geral, esses métodos utilizam o modelo

MRF e se valem da propriedade de localidade do pixel dentro da imagem, por isso utilizam o conceito de vizinhança do pixel para saber em qual local ele está inserido.

Efros e Freeman apresentaram em (EFROS; FREEMAN, 2001) um algoritmo de síntese por preenchimento com blocos. Neste trabalho observaram que os métodos baseados na abordagem pixel-a-pixel ("um-pixel-por-vez") procuram e copiam pixels já "conhecidos", ou seja, a cada pixel a ser sintetizado, valendo-se da propriedade de localidade do modelo, seus vizinhos também são conhecidos, portanto já determinados; entretanto somente o valor de um pixel é copiado, e os demais (da vizinhança) passam pelo mesmo processo de busca.

Há um desperdício de tempo de processamento muito grande, ao passo que a vizinhança também poderia ser copiada. Por causa disso propuseram um método que utiliza outra unidade de síntese ao invés de um pixel-por-vez: sintetizar blocos de pixels.

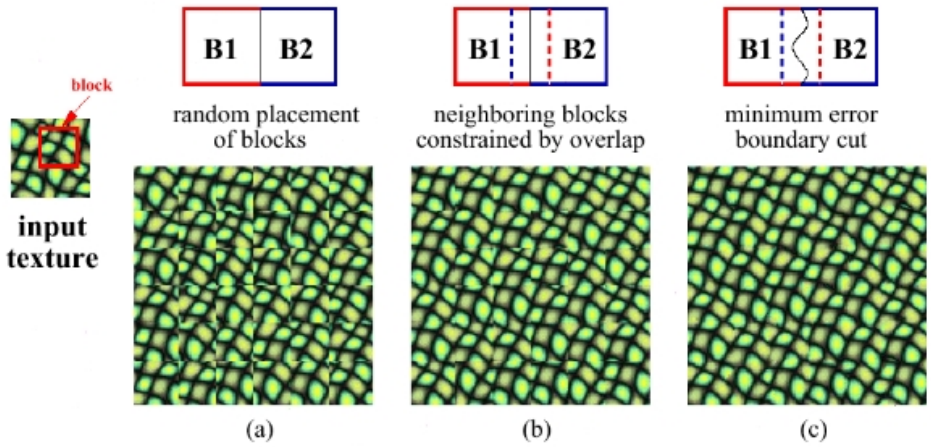

Figura 13: Ilustração do processo de síntese por blocos, feito em dois passos: cópia dos blocos que combinam (b) e ajuste do erro mínimo (c). Em (a) um exemplo da simples colocação de blocos aleatoriamente retirados da amostra. Figura de (EFROS; FREEMAN, 2001).

O algoritmo é muito simples e, realmente, é bem mais eficiente do que outras técnicas da abordagem por pixel-a-pixel. Dada uma imagem de entrada (amostra) I_i, B_{I_i} é um bloco qualquer contido nesta imagem, e S_B é o conjunto de blocos sobrepostos, contido em I_i. O algoritmo posiciona um bloco por vez, sendo que todos os blocos sintetizados são colocados sobrepostos numa determinada faixa de sobreposição ou borda. A escolha

de qual bloco será copiado é influenciada pela região de sobreposição dos blocos vizinhos, de forma a manter uma coerência no resultado. A métrica utilizada para escolha do bloco é a norma L2.

Ainda assim, os blocos apresentam erro nas suas bordas, a mudança de um bloco para outro pode ser percebida. Finalmente, para fechar a combinação dos blocos, foi implementado um algoritmo para computar a melhor forma de combinação para dois blocos. É o algoritmo do erro mínimo.

O erro mínimo é calculado por $e_j = (p_{ij}(B_1) - p_{ij}(B_2))^2$, onde j é a linha atual e i é a coluna, caso o erro esteja sendo calculado para combinação dois blocos na horizontal. Para o outro caso, j é coluna e i é a linha. No caso da combinação lateral, a menor diferença é encontrada para cada linha e os pixels à esquerda do "caminho" do erro mínimo são copiados de B_1 e os pixels à direita, são copiados de B_2. No caso da combinação vertical, os pixels abaixo do erro mínimo de cada coluna são copiados de B_1 e os que estão acima, são copiados de B_2. Veja a figura 13 para maior clareza do processo.

O tamanho dos blocos é um parâmetro de usuário, mas é indicado usar um tamanho que capture as características da textura de amostra (elementos mais característicos). Também não é interessante determinar blocos de tamanho muito grande, pois limita o número de blocos possíveis, podendo prejudicar o processo de busca. Para o tamanho da borda ou região de sobreposição, não é indicada nenhuma regra, contudo, eles utilizaram para todos os resultados publicados uma faixa de 1/6 do tamanho do bloco. Os resultados são muito bons e a performance do algoritmo é adequada, porém ainda pode falhar com estruturas mais regulares e não mantém nenhum controle local de síntese.

Outro trabalho analisado na abordagem de preenchimento por blocos é (LIANG et al., 2001) apresentado por Liang *et al.* Também baseado na idéia de copiar mais informações da amostra a cada passo de síntese, eles propuseram um algoritmo muito parecido com o publicado em (EFROS; FREEMAN, 2001). A principal diferença está na métrica para comparação dos blocos candidatos a uma combinação e na montagem dos

blocos escolhidos para gerar o resultado.

Para sintetizar uma textura I_{out} a partir de blocos existentes numa amostra I_{in}, a cada passo o algoritmo copia um bloco que tenha uma boa combinação com o bloco anterior já sintetizado.

O algoritmo em alto-nível pode ser apresentado como:

1. Escolher um bloco aleatoriamente em I_{in} e colocar no canto inferior esquerdo de I_{out}.

2. Para sintetizar o próximo bloco, busca em I_{in} todos os blocos que possibilitam um boa combinação com o(s) já sintetizado(s), segundo um critério de similaridade das bordas, formando um conjunto C de blocos candidatos.

3. Se o conjunto C estiver vazio, então o bloco que proporcionou a melhor combinação (ou menor diferença) é copiado.

4. Senão, selecionar aleatoriamente uma das possibilidades de combinação presente no conjunto C. Isto garante a diversidade de resultados possíveis.

5. Repetir os passos de 2 a 4 até que a imagem esteja completamente preenchida.

6. Forçar a combinação (*blending*) de todos os blocos copiados.

Para maior clareza do processo, veja a figura 14, que ilustra a colocação dos blocos em I_{out}.

O processo de escolha dos candidatos é o ponto chave do algoritmo. Primeiro, é determinado o erro máximo que é permitido a um bloco candidato para que ele possa fazer parte do conjunto de candidatos à combinação C do bloco anterior. Este erro é calculado com os pixels da borda do próprio bloco atual, seguindo a equação 3.1.

$$d_{max} = \epsilon \left[\frac{1}{A} \sum_{i=1}^{A} (p_{out}^{i})^2 \right]^{1/2} \tag{3.1}$$

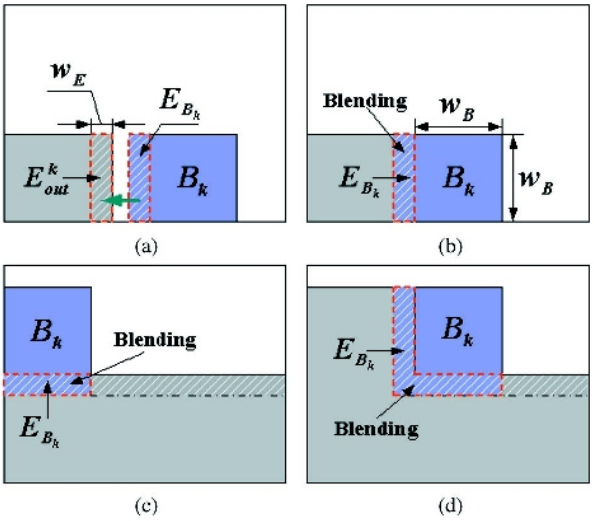

Figura 14: Ilustração do processo de síntese por blocos apresentada em (LIANG et al., 2001). Em (a) o início do processo, (b) exemplo de combinação de blocos somente nas bordas verticais, em (c) exemplo de combinação nas bordas horizontais e em (d) a combinação dos blocos do "meio" com região em forma de "L".

onde ϵ é um parâmetro de margem de erro, A são todos pixels da borda e p^i_{out} é o pixel atual.

A distância entre dois blocos (E_{B_k} atual e E^k_{out} anterior sintetizado), é o que determina se eles combinam ou não, e é calculada seguindo a equação 3.2. Se o resultado for menor ou igual a d_{max}, então o bloco é um candidato para combinação e é colocado em C.

$$d(E_{B_k}, E^k_{out}) = \left[\frac{1}{A}\sum_{i=1}^{A}(p^i_{B_k} - p^i_{out})^2\right]^{1/2} \tag{3.2}$$

onde A é quantidade de pixels da borda, $p^i_{B_k}$ é o pixel atual no bloco candidato (E_{B_k}) e p^i_{out} é o pixel atual no bloco já sintetizado (E^k_{out}).

A mistura (*blending*), ou combinação propriamente dita dos blocos, é feita com a técnica de *feathering* apresentada em (SZELISKI; SHUM, 1997).

A figura 15 ilustra um comparativo de resultados de síntese deste algoritmo com o equivalente gerado em (EFROS; FREEMAN, 2001). O algoritmo proposto gera muito bons resultados e é o mais rápido dos apresentados nesta revisão.

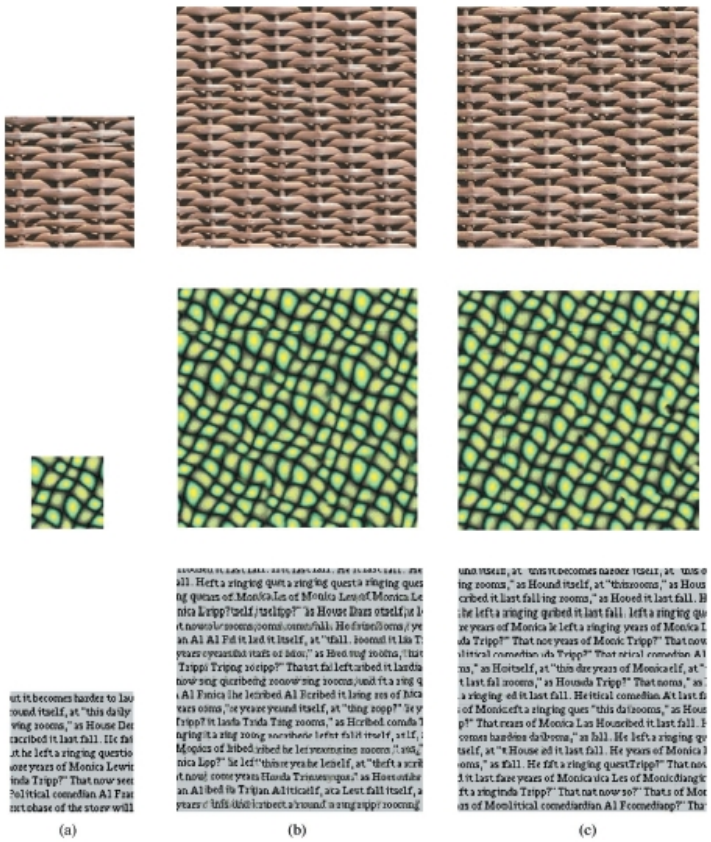

Figura 15: Comparativo de resultados entre (LIANG et al., 2001) e (EFROS; FREEMAN, 2001). Os resultados (b) em (LIANG et al., 2001) são visualmente mais fiéis à amostra (a) do que os resultados gerados em (EFROS; FREEMAN, 2001) (c).

3.3 Texturas com Variação Progressiva

Em geral os algoritmos de síntese de texturas revisados apresentam resultados visuais satisfatórios para texturas homogêneas. Texturas homogêneas são consideradas aquelas cujos elementos mais característicos não possuem nenhuma mutação de forma, cor ou tamanho (ou pelo menos não significante no todo), ao longo de toda a imagem de amostra. A figura 16 mostra exemplos de texturas homogêneas.

Entretanto, na natureza encontramos diversos tipos de texturas e em muitos casos, elas não são homogêneas. Existem também, texturas cujos elementos mais característicos possuem uma transformação progressiva ao longo de toda uma superfície. O padrão de

Figura 16: Exemplos de texturas homogêneas

pelagem de animais é um bom exemplo. No leopardo, como mostra a figura 2, percebe-se claramente que os elementos característicos da textura são as rosetas; elas variam em forma, tamanho e talvez até cor, ao longo do leopardo, mas ainda assim tem-se noção do mesmo elemento, a roseta. Assim são as texturas que variam progressivamente e este conceito surgiu em Zhang *et al* (ZHANG et al., 2003). São texturas que localmente são homogêneas, ou estacionárias, porém no todo elas são variantes.

No artigo os autores propõem um algoritmo para sintetizar texturas com variação progressiva. Primeiro, criaram um algoritmo (*field distortion*) para sintetizar PVT 2D a partir de amostras homogêneas, variando escala e orientação dos elementos estruturantes. Introduziram também o conceito de máscara de *textons* [1]. A máscara é um mapa com poucas cores que destaca os elementos mais característicos da amostra. Em geral, usam duas cores, onde o preto indica o "fundo" da textura e o branco indica a região do elemento característico.

A máscara exerce papel fundamental neste modelo, ela é responsável por manter a coerência do resultado final, pois evita que os elementos característicos "quebrem" ou saiam da forma original. Outra característica importante da máscara, é que ela é imune às falhas da métrica de diferença ou distância, não correndo o risco de coletar dados visualmente "errados".

Assim, o ponto-chave do algoritmo é sintetizar primeiro a máscara, depois a textura, pois ela manterá a coerência e guiará o resultado final (colorido) da melhor forma possível. A máscara também é um parâmetro especificado pelo usuário, onde o mesmo

[1] Texton é a representação de um texel na máscara de textons.

determina através da limiarização de cor, quais os elementos mais característicos e o que se pode considerar "fundo" da amostra.

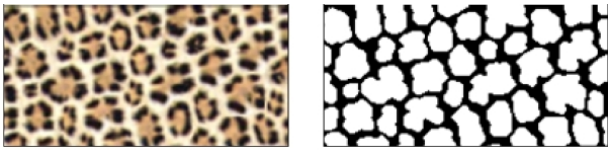

Figura 17: Amostra e sua respectiva máscara de *textons* em (ZHANG et al., 2003)

Para sintetizar uma textura PVT, o algoritmo necessita da amostra e sua respectiva máscara de *textons*, campos vetoriais de orientação e campos vetoriais de variação de escala. Os campos vetoriais são capturados numa interface para o usuário, onde o mesmo especifica algumas orientações ou escalas-chave e o programa calcula as demais (um para cada pixel) por interpolação de base-radial. A máscara de *textons* é um mapa que identifica onde e como é a forma de cada um dos elementos mais característicos da amostra. O *texton* é a representação destes elementos no mapa. Na figura 17 um exemplo de amostra e sua máscara de *textons*.

A base do algoritmo é mesma que o algoritmo de resolução simples apresentado em (WEI; LEVOY, 2000a), que copia o valor do pixel cuja vizinhança é a mais próxima ou parecida com a do pixel atual, somando ainda o controle de orientação e escala. O controle de escala e orientação é feito na construção da vizinhança do pixel ($N(p)$). Para trabalhar a escala, o algoritmo busca um fator de escala associado ao p, onde $d = F(p)$. Este d determina o novo tamanho dos pixels em $N(p)$. A $N(p)$ é então rotacionada de acordo com um ângulo associado ao pixel atual p, nos campos de orientação. Finalmente, o valor de cada pixel $N(p)$ é calculado pela interpolação bilinear dos quatro vizinhos mais próximos. Depois de construída a $N(p)$, o algoritmo procura na amostra a melhor combinação para esta vizinhança. A figura 18 exemplifica o processo.

O controle do algoritmo acima é classificado como a nível de pixel, pois ele trabalha de maneira uniforme todos os pixels e as mudanças refletem no todo. Entretanto, as PVT também variam a forma dos elementos característicos. Para realizar esta transformação, é preciso ter um controle a nível de elemento, para modificar a forma dos elementos carac-

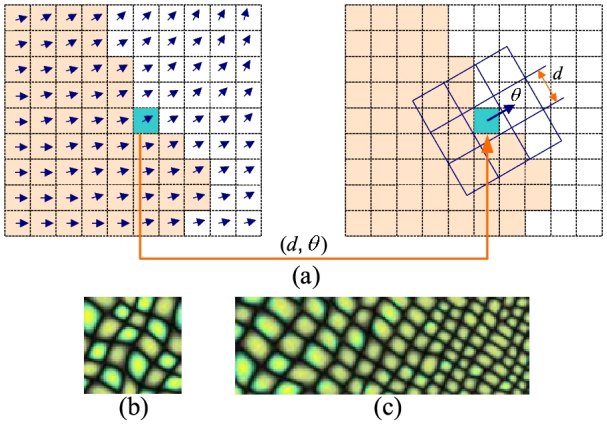

Figura 18: Ilustração do algoritmo para síntese de PVT em (ZHANG et al., 2003). Cada pixel na imagem de saída, tem uma orientação e uma escala associada a ele. O algoritmo aplica estas duas transformações direto na vizinhança do pixel (a). Abaixo textura de amostra (b) e o resultado da síntese para PVT (c)

terísticos. Eles solucionaram este problema com algoritmos baseados em características (*feature-based techniques*).

As duas técnicas baseadas em características são *warping* e *blending*. A técnica de *warping* realiza operações de modificação nos elementos característicos através da máscara de *textons* da amostra com alterações feitas pelo usuário. Assim, para gerar uma nova textura PVT, o algoritmo utiliza a amostra, máscara original e a máscara editada. No final o resultado obedecerá o controle estipulado na máscara editada pelo usuário. Veja a figura 19 que mostra como fica o resultado depois do *warping*.

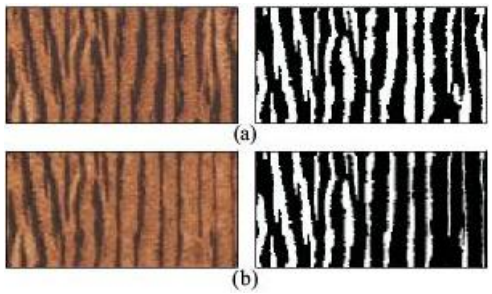

Figura 19: Técnica de *Warping* em (ZHANG et al., 2003). Em (a) textura e sua máscara original e (b) depois de aplicada a técnica de *warping*. Note que as listras da textura do padrão visual do tigre vão se estreitando para a direita.

O algoritmo de blending é capaz de misturar duas texturas homogêneas, conforme mostra a figura 20. Neste algoritmo o usuário determina as amostras e suas respectivas

máscaras. O sistema gera uma nova máscara para cada amostra com o resultado do blending das amostras.

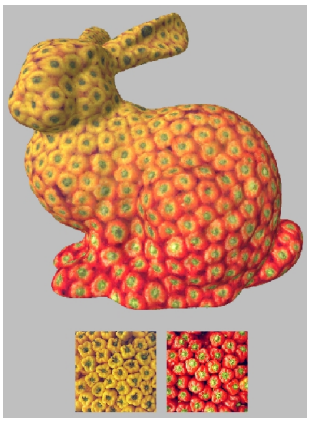

Figura 20: Resultado do algoritmo de *blending* em (ZHANG et al., 2003).

Os resultados apresentados no artigo são visualmente atraentes (veja figura 21) e o sistema consegue estabelecer controle local de síntese através dos algoritmos de distorção de campos e o *warping*, porém os resultados ainda podem ser melhorados e a performance do algoritmo ainda é muito lenta.

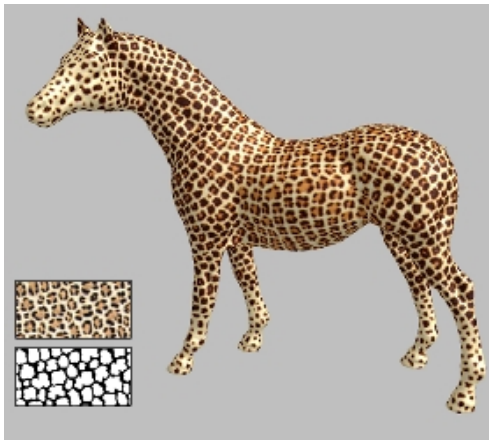

Figura 21: Resultado do algoritmo de síntese de PVT em objetos 3D em (ZHANG et al., 2003). Em destaque a amostra utilizada e a respectiva máscara de *textons*.

3.4 Contextualização do Trabalho no Estado da Arte

Os algoritmos estudados são eficientes para a geração de novas texturas a partir de amostras. Entretanto, a grande maioria apresenta apenas diferentes maneiras para

reprodução de amostras de texturas homogêneas, não abordando nenhuma forma de controle sobre o resultado gerado. Muitos tipos de texturas os padrões não são homogêneos, principalmente em padrões de texturas do mundo real, existem texturas com variação progressiva ao longo da amostra. Em (ZHANG et al., 2003) é apresentado um algoritmo para síntese de PVT. Este algoritmo consegue simular as variações dos *texels* da textura utilizando campos vetoriais de orientação e escala, e a introdução de novo conceito com a máscara de *textons*. Os autores afirmam que a máscara é imune às falhas da norma L2, que causam a "quebra" ou deformação da região dos *texels*, porque a norma L2 falha na detecção de bordas e cantos.

O problema também está relacionado em usá-la com o sistema de cores RGB, onde não é possível garantir que, a uma mesma distância d, uma cor será da região desejada em qualquer direção dentro do cubo RGB que medirmos esta distância. Como a máscara utiliza apenas algumas cores distintas, preto e branco, e em alguns casos tons de cinza, este problema não ocorre. Este é justamente o ponto onde os algoritmos anteriores apresentam falhas e que não ocorrem em (ZHANG et al., 2003). O avanço proporcionado por esta técnica foi tal que é possível sintetizar em objetos 3D texturas progressivamente variantes e obter resultados próximo à realidade, ou pelo menos, dentro do esperado, como na figura 21, onde a textura do Leopardo é mapeada num cavalo para demonstrar a variação dos elementos do abdômen e das extremidades.

Neste sentido o nosso trabalho parte da idéia das texturas com variação progressiva, mas diferentemente da abordagem pixel-a-pixel apresentada, estaremos propondo uma abordagem em blocos com formatos quaisquer. Estes blocos se aproximarão dos *texels* da textura e serão os blocos básicos para construção do resultado final. A textura final será uma combinação de blocos que podem ter sido *transformados* por operações geométricas básicas como rotação e escala, e ainda por operações morfológicas como dilatação e erosão. Este conjunto de blocos que podem ser transformados passa a ser um ferramental valioso para construção de novas texturas. No próximo capítulo detalhamos este procedimento.

4 MODELO

O objetivo principal do trabalho é sintetizar PVT de uma maneira alternativa à apresentada em (ZHANG et al., 2003). O conceito de texturas PVT consegue capturar a idéia de variação progressiva de elementos básicos. A proposta aqui sendo apresentada toma esta constatação como inspiração. A nova abordagem em alto nível consiste em (*i*) identificar os *texels* e dispor estes em um conjunto de texels de amostra; (*ii*) sintetizar a textura partindo de um *texel*-chave; (*iii*) aplicar transformações (orientação e escala) e; (*iv*) aplicar operações morfológicas (dilatação, erosão, abertura, fechamento, expansão e contração de área) em cada *texel* para montar uma PVT. Veja a ilustração do processo na figura 22.

Além de sintetizar PVT, o que também se apresenta neste trabalho, é propor uma variação da abordagem de síntese por preenchimento de blocos (retangulares), para a abordagem de síntese por preenchimento de *texels*. Assim, nossa unidade de composição de textura não é apenas um pixel (cuja síntese é mais lenta e mesmo com a máscara corremos o risco de quebra de elementos), nem um bloco retangular (que pode quebrar alguns *texels* e inviabiliza a aplicação de transformações), mas sim um bloco de pixels, de formato irregular, que contenha uma estrutura característica da textura, um texel.

A maior vantagem para se usar este novo tipo de abordagem é que os elementos característicos da textura, os *texels*, nunca vão quebrar. Outra vantagem é que nesta abordagem são trabalhadas formas ou estruturas e não pixels sem um forte relacionamento intrínseco. A figura 23 mostra como é a nossa unidade de síntese e a diferença em relação aos blocos como foram utilizados em (LIANG et al., 2001) e (EFROS; FREEMAN, 2001).

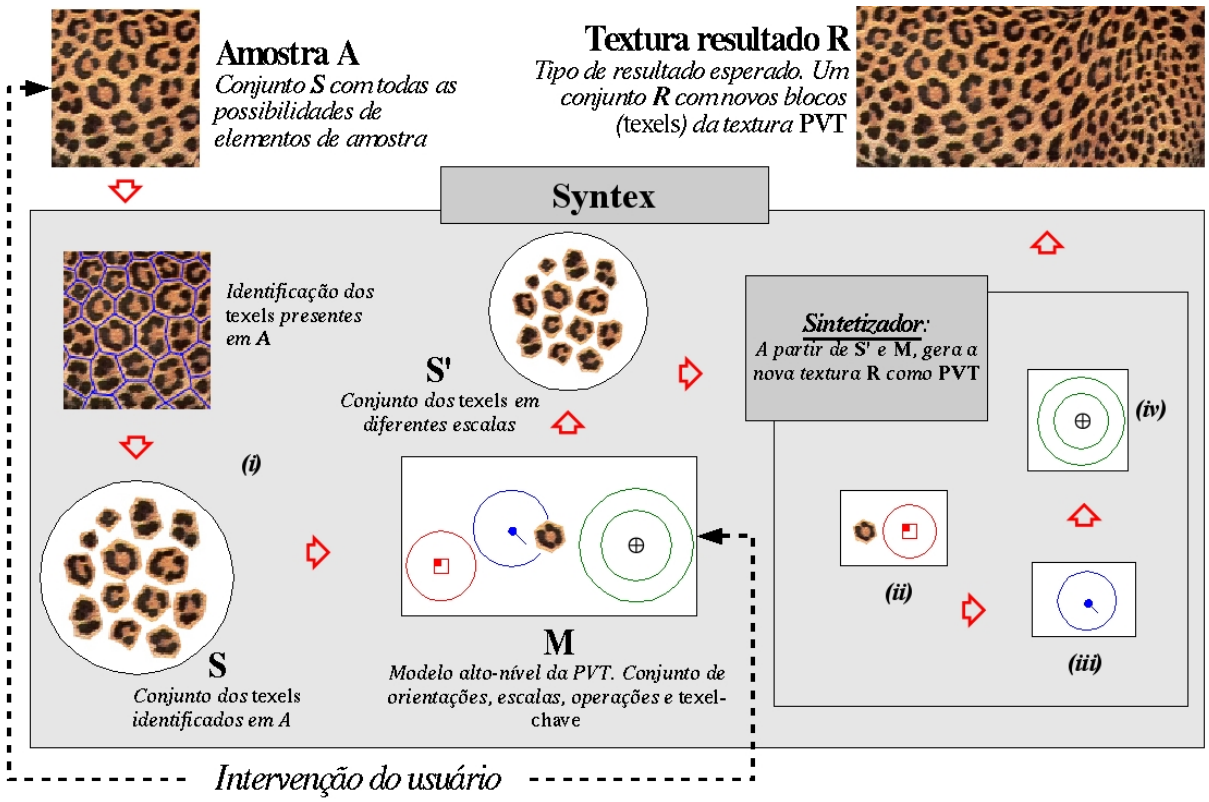

Figura 22: Diagrama do modelo para síntese de PVT. O usuário deve informar a amostra A e o modelo alto-nível M para sintetizar PVT. O sistema detecta os *texels* e os separa numa estrutura que armazena o conjunto S dos *texels* de A (i). O modelo em alto nível consiste do texel-chave, orientações-chave, escalas-chave e operadores-chave. O sistema recebe a amostra e um modelo e faz primeira a etapa de síntese com os texels originais e os escalados (ii). Depois, o sistema aplica as orientações sobre os texels (iii), para rotacioná-los e transladá-los, dando um efeito de movimento. Por fim, são aplicadas as operações morfológicas para proporcionar algum efeito de transformação sobre os texels (iv).

Como mostra a figura 23, um bloco, como proposto em (LIANG et al., 2001) e (EFROS; FREEMAN, 2001), não contém apenas pixels dos *texels* em questão, eles carregam também, pixels de outras estruturas. Isto ocorre porque estes blocos simplesmente não fazem nenhuma relação aos *texels*. Portanto, aplicar a abordagem de síntese por blocos para sintetizar PVT, não é tão trivial, pois os blocos rotacionados ou numa escala diferente, produzem resultados inconsistentes, não mantendo a coerência dos *texels*, conforme a amostra.

A síntese utilizando texels é feita combinando e ligando os texels nas suas chamadas "áreas de conexão". As áreas de conexão são regiões dos texels, dentro de um *bounding-*

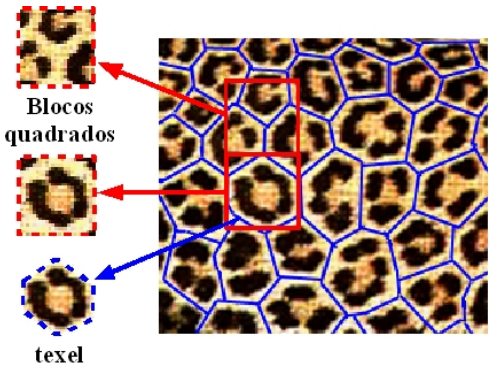

Figura 23: Exemplo de como são os *texels* de uma amostra no sistema. Note que os *texels* carregam uma parte do fundo para facilitar a combinação com outros *texels*. Em destaque também, blocos como utilizados em (LIANG et al., 2001) e (EFROS; FREEMAN, 2001).

box, que são transparentes, portanto, não fazem parte deste texel, mas sim de algum outro na amostra. É nesta definição que o trabalho está fortemente apoiado. A figura 26 mostra as áreas de conexão para um texel. O processo de síntese é descrito mais detalhadamente na seção 4.3.

O procedimento para geração de PVT pode ser dividido em seis etapas: formação do conjunto de *texels* de amostra, especificação do modelo para PVT, primeira etapa de síntese (já considerando escalas), aplicação das orientações, aplicação das operações morfológicas e preenchimento das falhas. As seções a seguir descrevem cada uma dessas etapas em detalhes.

4.1 Formação de Conjunto de *Texels* de Amostra

Nesta primeira etapa, a formação do conjunto de *texels*, o sistema separa os texels da amostra para compor um conjunto de texels de amostra S', que será a base de informação para a síntese. A formação do conjunto de texels é feita da seguinte forma: primeiro, partindo da textura de amostra e da cor-de-fundo da amostra, especificadas pelo usuário, o sistema monta uma máscara binária da amostra e depois separa os *texels* com base nesta máscara (veja figura 25).

A máscara binária é o resultado de uma operação de *thresholding* (limiarização)

sobre a textura de amostra (veja imagem do centro na figura 25). Partindo da uma cor RGB de referência e um limite de tolerância (entre 0 e 1), o sistema computa uma máscara com pixels em preto quando a cor pixel da amostra estiver abaixo do limite de tolerância estabelecido pelo usuário, e pixels em branco quando estiver acima deste limite. Para saber se a cor está abaixo do limite de tolerância, usamos a norma $L2$ para determinar uma distância de cor, entre a cor de referência e a cor do pixel atual. Este valor é normalizado pela distância máxima $(441,673)$, que é o resultado da norma $L2$ aplicada à diferença máxima em cada canal RGB (255).

Mais especificamente, a separação e identificação dos *texels* consiste em: identificar os pixels das regiões de primeiro-plano (regiões em branco na imagem do centro da figura 25), criando um registro de *texton* para cada região; e adicionar os pixels de segundo-plano ou fundo (região em preto na máscara) no *texton* mais próximo de cada ponto. Para determinar a qual texton um pixel de fundo pertence, o sistema traça um linha do pixel ao centro de cada texton e verifica o ponto de intersecção desta linha com a borda de cada texton. O ponto de intersecção que estiver mais próximo ao pixel de fundo atual, indica a qual texton ele pertence. Após identificados todos os textons, para cada registro de texton, o sistema gera um texel, copiando da amostra as informações de cor de cada pixel do mesmo. Veja a figura 24.

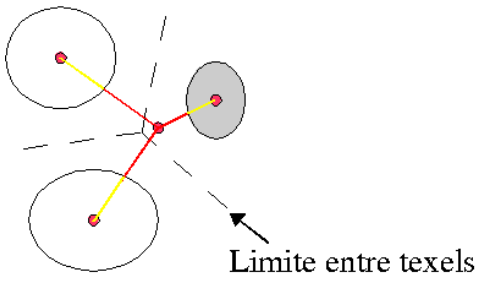

Figura 24: Exemplo de distribuição dos pontos de fundo. O texton em destaque é o que receberá o ponto do exemplo.

Para o processo de síntese são utilizados apenas os *texels* considerados válidos, pois os *texels* não válidos são elementos "quebrados", cortados pelas bordas da amostra e, naturalmente, introduzem combinações indesejadas ao resultado. Os texels inválidos

são facilmente identificados como contendo pelo menos um pixel no limite da resolução da amostra A.

Figura 25: Exemplo de como é feita a separação dos *texels* de uma amostra. Primeiro, é gerada uma máscara binária da amostra e depois ocorre a separação dos *texels* de acordo com a máscara. Note que o sistema desconsidera os *texels* das bordas.

Após a separação dos texels da amostra, é necessário computar as áreas de conexão de cada texel. As áreas de conexão são a parte mais importante do texel para o sintetizador, pois é nelas que o sistema conecta novos texels. O procedimento para detecção das áreas de conexão é simples. O sistema percorre todos os agrupamentos de pontos transparentes do texel (ou não pertinentes ao texel) e os demarca com uma área retangular (*bounding-box*). A figura 26 é um exemplo de um texel e as suas áreas de conexão.

Figura 26: Exemplo de como são as áreas de conexão de um *texel*. À direita uma representação esquemática do texel e suas áreas de conexão.

Uma outra vantagem da abordagem por preenchimento com *texels* é que o usuário não fica limitado ao conjunto inicial de *texels* da amostra. Ele pode gerar variações destes *texels* e aumentar a quantidade de *texels* amostras. Neste trabalho, por exemplo, nós acrescentamos ao conjunto inicial mais três versões de cada texel: espelhado verticalmente, espelhado horizontalmente e espelhado em ambos os sentidos. A figura 27 mostra um *texel* e suas variações com espelhamentos.

Como se pode notar este processo é feito antes da etapa de síntese, o que permite ao usuário utilizar outras rotinas para a formação do conjunto de *texels* de amostra.

Por fim, são gerados os conjuntos de texels com diferentes escalas S'. Estes conjuntos são pré-computados porque são os mesmos utilizados para todas as escalas-chave

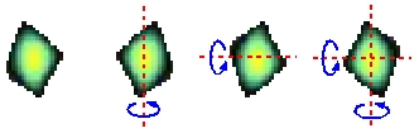

Figura 27: Exemplo de variações para um *texel*. Na seqüência da esquerda para a direita, o original, espelhado verticalmente, espelhado horizontalmente e espelhado em ambas as direções.

e para não onerar o processo no momento da síntese. Para gerar S' o sistema considera um conjunto de escalas s, que contém as possibilidades de escalas para cada texel. Atualmente, utilizamos $s = 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9$. O sistema gera para cada escala s uma nova versão do conjunto dos texels de S com tamanho diferente: $S' = S_{0,4}; S_{0,5}; S_{0,6}; S_{0,7}; S_{0,8}; S_{0,9}$. E em cada versão escalada do texel, é feito o mesmo processo de identificação das áreas de conexão.

4.1.1 Identificação de Texels com Imagem Auxiliar

A identificação de texels em texturas pode não ser possível para algumas amostras, devido a problemas de qualidade na digitalização das imagens ou artefatos 3D presentes na amostra (como iluminação, por exemplo), o que torna a tarefa de separação dos texels dos pixels de fundo muito difícil e em alguns casos impossível pelo método descrito anteriormente.

Um tipo de textura que geralmente causa este problema é o de pelagem de animais. Estas texturas normalmente são adquiridas através da digitalização de fotografias, perdendo qualidade neste processo (amostra da figura 28). Além disso, geralmente apresentam problemas de iluminação diferente em diversos pontos da amostra ou ainda, muitos texels possuem pixels cuja cor está dentro da região da cor de fundo, o que inviabiliza a separação dos mesmos do pixels de fundo.

Nestes casos não podemos usar diretamente o método de identificação dos texels mencionado anteriormente. É preciso a intervenção do usuário sobre a textura de amostra para tornar possível essa identificação. Assim, colocamos no sistema um recurso para

o usuário especificar, além da amostra, uma imagem auxiliar editada para facilitar a identificação dos texels. O sistema utiliza uma imagem para informar os texels de entrada e a outra apenas para auxiliar na separação dos texels. A figura 28 mostra um exemplo do problema de identificação de texels para uma textura com os problemas mencionados e também como é a separação dos texels com auxílio da imagem editada.

Figura 28: Processo de identificação de texels com o uso de imagem auxiliar. Da esquerda para direita: a textura de amostra, máscara da amostra cujo parâmetro de cor de fundo $bg = (247, 240, 196)$, imagem auxiliar editada a partir da amostra, máscara da imagem auxiliar e máscara com textons identificados com cores diferentes.

4.2 Especificação do Modelo para PVT

Após definido o conjunto de texels de amostra, o usuário começa a montar o modelo PVT. Ele determina as escalas, as orientações e as operações-chave e raio de abrangência de cada uma. No momento da síntese, o sistema constrói a PVT com base nesses parâmetros (conjunto M). Veja figura 29. A intervenção do usuário para especificação destas informações em alto-nível também foi utilizada em (ZHANG et al., 2003).

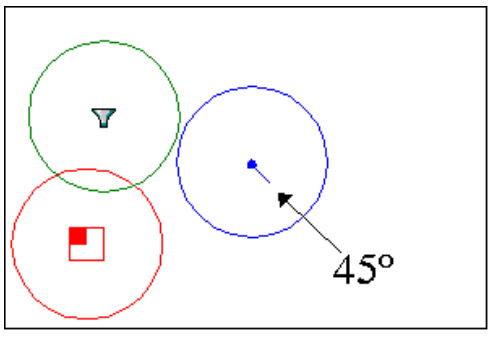

Figura 29: Exemplo de modelo de PVT. Em azul a especificação da escala-chave, em verde está a operação-chave e em vermelho a escala-chave.

Cada parâmetro do sistema (escalas ou orientações) é definido como um ponto de referência e um raio de ação associado. Desta forma, quando o sistema estiver sintetizando

um texel que esteja dentro do raio de ação de um parâmetro, ele deve calcular a influência desse parâmetro para o texel com interpolação simples.

Por exemplo, para o modelo ilustrado na figura 29, quando o sistema sintetiza um texel que coincide com a borda do círculo da orientação-chave (em azul), a orientação deste texel será mais próxima de 0° do que do valor da orientação-chave, no caso 45° em sentido horário. Desta maneira, o sistema faz uma transição suave entre os texels fora do raio de ação e dentro do raio de ação de cada parâmetro.

4.3 Primeira Etapa de Síntese

A primeira etapa da síntese gera a malha de texels conectados. Nesta etapa o sistema já considera também os registros de texels em escalas diferentes. Isto significa que quando o centro do texel estiver dentro do raio de ação de uma escala-chave, o sistema só procura dentro do conjunto do texels escalados S'.

O processo de síntese por preenchimento de texels na sua forma mais básica, é identificar no conjunto S' a melhor combinação para cada área de conexão do texel. Assim, cada texel estará ligado com um outro texel nas suas áreas de conexão e o sistema evolui o preenchimento até os limites de resolução da imagem de saída desejada.

A pesquisa pela melhor combinação pode ser feita de duas maneiras: ou comparando a parte transparente da área de conexão sobreposta nos texels candidatos, ou comparando também o valor RGB de cada pixel transparente mais os pixels não transparentes da área de conexão, com todos os pixels dos texels candidatos à combinação. A comparação é feita dentro dos limites da área de conexão do texel atual. Desses limites é computado um retângulo ou janela para pesquisa sobre todos os pontos de cada texel candidato à conexão. Repare nas figuras 30 e 31, a sobreposição do retângulo da área de conexão em diversos pontos dos texels candidatos. Em vermelho a área que produziu a melhor combinação.

Caso a pesquisa seja pelos pixels transparentes, a qual chamamos de comparação pelo canal alfa, o sistema fará uma sobreposição dos pixels da área de conexão com um determinado texel candidato. São comparados apenas os valores do canal alfa (que indica a opacidade da cor) dos pixels. Quando um pixel na área de conexão é transparente (0 no canal alfa), o pixel correspondente na área pesquisada do texel candidato deve ser o seu complemento, portanto não transparente (255 no canal alfa). Para efeitos de cálculo, o valor do ponto na área é invertido, se ele é 0 passa para 255 e vice-versa. Assim, quando os pixels combinam a diferença entre eles é zero. Por exemplo, $inv(0) - 255 = 0$ e $inv(255) - 255 = 255$ (valor absoluto). A diferença total entre texels é a soma das diferenças em cada pixel da área pesquisada no texel candidato.

A área que produzir a menor diferença total dos pixels é a melhor combinação dentro do candidato. E, consequentemente, o texel candidato que tiver a melhor combinação (menor diferença) é o eleito para a conexão com a área do texel atual. (Figura 30).

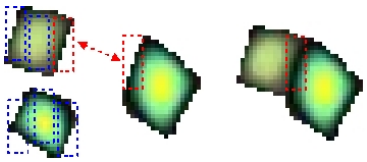

Figura 30: Escolha do melhor candidato pela combinação dos pixels transparentes. Ao centro o texel atual, à esquerda a comparação com todos os pixels dos dois texels candidatos e à direita o resultado da combinação. Em vermelho o retângulo da melhor combinação para a área.

Já no caso da combinação com canal alfa e mais o valor RGB, a qual chamamos comparação nos canais RGBA (ou alfa com RGB), o sistema executa a mesma comparação do canal alfa e verificará na sobreposição também se a cor de cada pixel (canais RGB) da área de conexão coincide com a cor de cada pixel correspondente na área testada do texel candidato. É computada a diferença RGB de cada pixel de acordo com a norma $L2$ e ainda, é somada a diferença no canal alfa. Finalmente, a diferença entre cada pixel é somada à diferença total da área testada. A área comparada no candidato que tiver a menor diferença em relação a área de conexão do texel atual é a melhor combinação para o candidato. O candidato que possui a menor diferença (melhor combinação) é o eleito

para conectar com a área do texel atual (Figura 31).

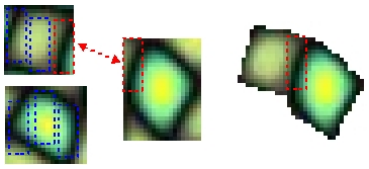

Figura 31: Escolha do melhor candidato pelo canal e mais a diferença da cor dos pixels. Ao centro o texel atual, à esquerda a comparação com todos os pixels dos dois texels candidatos e à direita o resultado da combinação. Em vermelho o retângulo da melhor combinação para a área de conexão.

A comparação pelo canal alfa pode falhar em algumas situações, pois não considera a relação das cores entre pixels dos texels, somente a forma dos texels. Isto prejudica na relação espacial dos texels e o resultado ainda apresenta muitas repetições de alguns texels nas direções determinadas pelas suas áreas de conexão. Na comparação pelo valor RGB esta relação de cores dos pixels é preservada e o número de falhas é menor. Além disso, o número de repetições de uma mesma estrutura é bem menor.

Pelos motivos já citados, na grande maioria dos testes realizados para este trabalho utilizamos a comparação RGBA. E todos os resultados (veja capítulo 5) utilizam esta métrica de combinação de texels.

A figura 32 mostra a síntese com os dois tipos de comparação. Note que na comparação apenas pelo canal alfa o sistema não preserva muito bem a relação de cores entre os texels e ocorre maior repetibilidade texels. Na comparação com todos os canais (alfa e RGB) o sistema preserva melhor a relação de cores, não apresenta tantas repetições e diminui muito a sobreposição dos texels.

A cada passo, o sistema sintetiza os texels novos do passo anterior, fazendo novas ligações e aumentando o que chamamos de *bloxel*. Um *bloxel* é um conjunto (ou bloco) de texels conectados. Para visualizar melhor o processo de síntese a cada passo, veja a figura 33 que identifica com cores distintas cada passo de síntese.

Este processo de síntese pode ser visto como uma relação de hierarquia de texels. Primeiro é colocado um texel-chave t (escolhido pelo usuário ou sorteado aleatoriamente

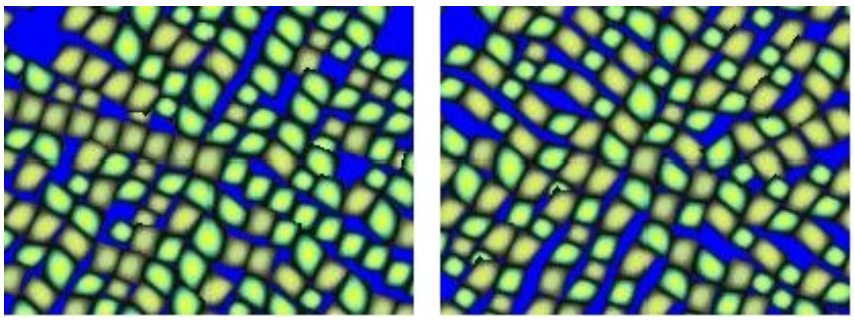

Figura 32: Exemplo partindo de um mesmo texel-chave, porém com diferentes formas de comparação. À esquerda a comparação pelo canal alfa e à direta a solução com comparação alfa e RGB.

pelo sistema), o qual é o pai dos novos texels conectados diretamente a ele (denominados filhos de t). E esta regra funciona para os demais texels sintetizados: os texels que serão sintetizados são filhos dos texels sintetizados no último passo de síntese. Na figura 33 os texels num mesmo nível hierárquico possuem a mesma cor. Devido ao fato de que os texels geralmente concorrem por um espaço vazio com algum "texel-primo" e como só um deles pode ocupar este espaço, eles são conectados a dois "texels-pais", o texel-pai que o gerou e ao texel do irmão concorrente; exceto os texels-chave (que não têm pai) e aqueles que estão ligados diretamente as chaves (com apenas um pai).

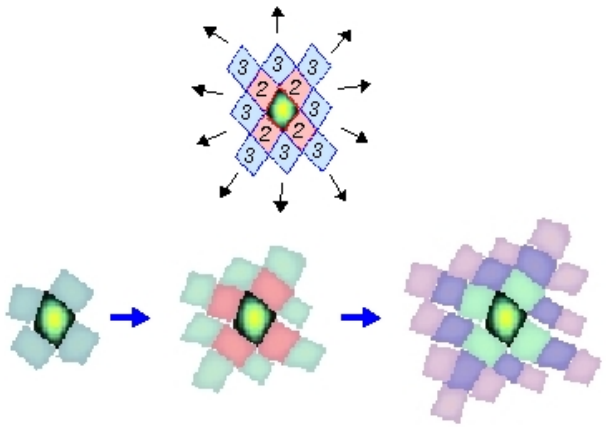

Figura 33: Ordem de síntese do algoritmo. O algoritmo começa a sintetizar a partir de um texel-chave e vai preenchendo novas conexões com novos texels de dentro para fora.

4.3.1 Síntese com Texels em Diferentes Escalas

Uma das características marcantes de uma PVT é a reprodução dos elementos característicos em diferentes escalas. A escala produz uma mudança no tamanho dos elementos característicos e, combinada com outras operações, pode promover a mutação do texel original para compor efeitos encontrados em algumas PVTs, como na figura 2. O algoritmo permite sintetizar texels de tamanhos diferentes de acordo com regiões associadas às escalas-chave.

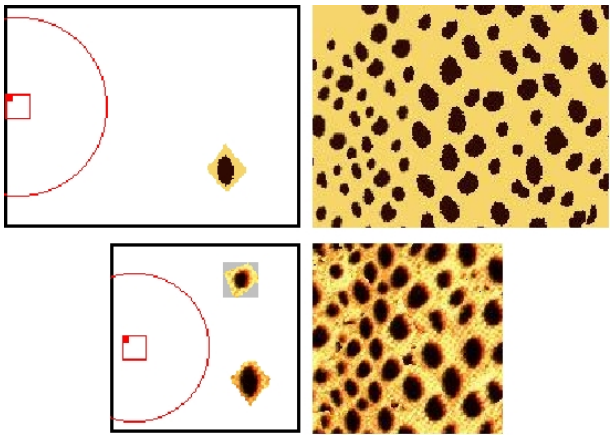

Figura 34: Exemplos da aplicação de escalas diferentes. À esquerda o modelo alto nível passado pelo usuário e à direita um resultado utilizando escalas diferentes. Note que no exemplo de baixo o modelo ainda apresenta mais de um texel-chave e o sistema ainda assim consegue tratar o encontro dos dois bloxels (um bloco de texels para cada texel-chave) com as novas estruturas escaladas.

O usuário determina uma ou mais escalas-chave e a região de abrangência dessa escala, conforme procedimento descrito na seção 4.2. No momento da síntese, o sistema faz a verificação da escala apropriada para o texel e busca a melhor combinação direto do subconjunto de texels apropriado.

Para saber se um texel está dentro de uma região de escala, o sistema verifica se o centro do texel coincide com a região de abrangência de alguma escala. A distância do centro do texel para o ponto de escala-chave determina o quão afetada é a escala do texel. Quanto mais próximo do centro da região da escala-chave, mais próximo do valor da mesma, quanto mais longe, mais de próximo de 1 (do original) é o valor da escala. A figura 34 mostra um exemplo de escala-chave e o comportamento do sintetizador no

momento da síntese.

A conexão de texels em escalas diferentes ocorre da mesma maneira que a combinação com texels da mesma escala, porém o conjunto de texels pesquisados para conectar é outro. Assim, o sistema elegerá a melhor combinação para o texel atual dentro do conjunto de escalas apropriado.

O sistema sintetiza as escalas durante a primeira etapa de síntese porque o conjunto S' já está previamente computado e porque é mais fácil promover as conexões em escalas diferentes neste momento do que a posteriori, pois a mudança de tamanho dos texels e, consequentemente, das suas áreas de conexão, poderia ocasionar em falhas no resultado. O sistema teria que movimentar e refazer as conexões entre os texels, isso é mais complicado e permitiria que ainda permanecessem algumas descontinuidades nos limites das regiões das escalas-chave.

4.4 Tratamento de Sobreposições de Texels

Um efeito que decorre do processo de combinação é a sobreposição de texels sintetizados. Isto ocorre porque é necessário permitir que os texels se sobreponham uns aos outros para que em casos onde não é possível um combinação perfeita (onde pixels transparentes se encaixem com pixels coloridos e vice-versa), o algoritmo coloque algum texel com a menor sobreposição.

Para permitir a sobreposição permitimos, no momento da comparação (tanto alfa quanto RGBA), que a área retangular no candidato ultrapasse os limites do retângulo que envolve o texel. Por exemplo, se a largura da área que está sendo comparada é 5 e a largura do texel candidato é de 15 pixels, a coordenada do primeiro ponto da área deveria chegar no máximo 10 ($10 + 5 = 15$) para não ultrapassar a área do texel e tentar combinação com pixels inexistentes. Contudo, pelo motivo já mencionado, colocamos uma tolerância para que a área que está sendo comparada ultrapasse os limites do retângulo do texel. Entretanto, quando isso ocorre, o algoritmo penaliza a comparação com a diferença

máxima entre os pixels sobrepostos fora dos limites do texel. Por exemplo, se a área ultrapassar 2 pixels na largura e 3 na altura, o sistema somará com a diferença dos pixels válidos o valor $(2 * 3 * MAX\_DIFF)$ onde MAX_DIFF é a diferença máxima nos três canais RGB $(441, 673)$. Esta penalização é importante para que estas comparações não tenham preferência sobre as comparações somente com pixels válidos. A figura 35 mostra como é uma comparação com pixels válidos e como é uma comparação excedendo os limites do texel.

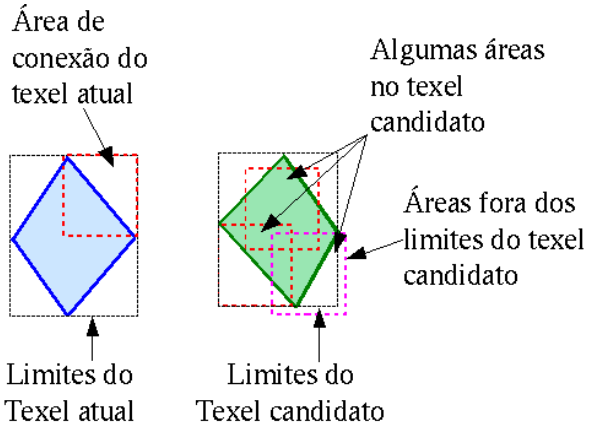

Figura 35: Exemplo do tratamento de sobreposições. Da esquerda para a direita o processo de tratamento de sobreposições.

Na interface o usuário determina uma tolerância à sobreposição dos texels com um parâmetro normalizado entre $(0, 1]$. De acordo com este parâmetro o resultado apresenta texels muito ou pouco sobrepostos. Na figura 36 é apresentado um comparativo entre diferentes níveis de tolerância a sobreposições. Nos nossos resultados os valores utilizados foram entre $0,05$ e $0,10$ para o limite de tolerância a sobreposições.

Ao final do processo de síntese, o algoritmo elimina os texels que ainda estão sobrepostos acima do limite de tolerância especificado pelo usuário. Estas sobreposições ainda existem em virtude do processo de seleção de texels que concorrem à um mesmo espaço vazio, que será explicado a seguir.

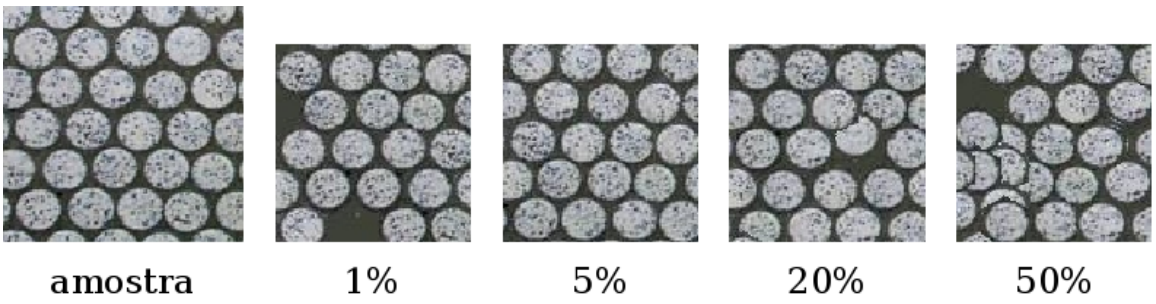

Figura 36: Exemplo de tolerância a sobreposições. Da esquerda para a direita a amostra (de (SIMONCELLI; PORTILLA, 2001)) e os diferentes níveis de tolerância. Note que quanto menor a tolerância mais preservada está a relação espacial dos texels. Entretanto, se este limite for muito baixo o sistema pode deixar falhas no resultado porque não conseguiu encaixar nenhum texel na área de conexão.

4.4.1 Seleção de texels que concorrem à um mesmo espaço vazio

A partir do segundo passo de síntese, alguns texels que ainda não foram conectados competem por um espaço vazio para conexão com outros texels sintetizados no mesmo passo. Ocorre que duas áreas podem conectar texels distintos num mesmo espaço vazio, gerando uma sobreposição de novos texels sintetizados. Nessas situações é preciso tomar uma decisão sobre qual é o melhor texel para ocupar aquele espaço.

Para decidir qual será o melhor texel dentre os dois novos sintetizados, nós verificamos qual deles combina melhor com o outro texel "pai" e assumimos este como o novo texel, mantendo as conexões atualizadas. Por exemplo, se o sistema conectou a um texel X um novo texel a, que sobrepõe um novo texel b conectado a um texel Y (do mesmo passo de X), o sistema compara a combinação de a com Y e de b com X. A menor diferença de conexão, ou melhor combinação, é a eleita para ocupar a vaga de novo texel. Digamos que o melhor texel é a, então o sistema deve conectar a com Y na área de conexão apropriada de a. O exemplo é ilustrado na figura 37.

4.5 Aplicação das Orientações

A outra característica marcante de uma PVT é a orientação (operações de rotação e translação) dos elementos característicos ao longo da textura. Com a orientação é possível

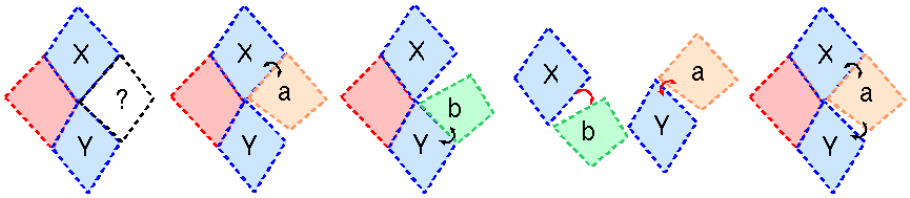

Figura 37: Exemplo de seleção de texels concorrentes. Da esquerda para a direita o processo de seleção de texels concorrentes.

simular a movimentação ou mudança de direção dos texels da textura.

Neste caso, também é esperado do usuário a indicação dos pontos-chave onde ocorrerão as orientações e uma região de abrangência associada. Depois, no momento da síntese o sistema verifica para cada texel se este deverá ou não ser orientado (rotacionado e transladado).

A orientação dos texels é aplicada depois da primeira etapa de síntese, para não tornar mais difícil o processo de conexão das áreas e porque fica fácil rotacionar os texels sem considerar suas ligações. A dificuldade de conectar texels rotacionados está no fato de que texels vizinhos possuem rotações diferentes, perdendo a relação original que os texels possuem, podendo deixar a busca mais ineficiente. Além disso, seria muito mais oneroso para o algoritmo rotacionar todos os texels candidatos e também todas as suas áreas de conexão a cada pesquisa. Ainda, a comparação de áreas rotacionadas seria muito mais complexa do que com áreas não rotacionadas.

Por fim, independente do momento onde é aplicada a orientação, considerando que as rotações não são bruscas (pois elas são interpoladas linearmente em todas as direções) a qualidade do resultado final em relação às conexões não fica comprometida em função da orientação dos texels. Em muitos casos, a rotação desloca alguns poucos pixels, não afetando muito, a nível local, o resultado, porém a nível global produz um efeito de movimentação.

Para efetuar as orientações, o sistema percorre todos os texels sintetizados e testa o centro do mesmo com a região de abrangência de cada orientação-chave. Se o texel deve sofrer alguma orientação, o sistema calcula o ângulo de rotação de acordo com a

posição do centro do texel, usando interpolação simples. Portanto, quanto mais próximo do ponto-chave da orientação, maior será a influência da orientação, quanto mais longe menor a influência. Desta forma, torna a movimentação suave e permite a rotação do texel sem maiores problemas com as regiões conectadas.

A figura 38 demonstra uma orientação-chave e o resultado com movimentação dos texels de acordo com esta orientação.

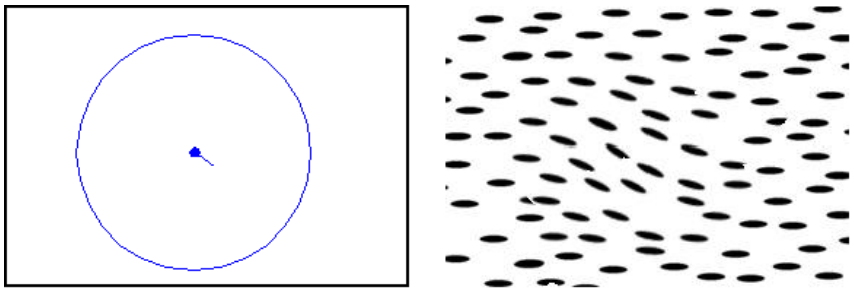

Figura 38: Exemplo de orientação dos texels. A orientação-chave deste exemplo é de 45° no sentido horário.

4.6 Aplicação de Operadores

A pura e simples aplicação de transformações afins (como rotação, escala e translação) pode não ser suficiente para chegar ao resultado de uma PVT. Em alguns casos as texturas apresentam também, modificações nas estruturas fundamentais, ou texels. Para simular essas modificações de texels nós optamos pelo uso de operadores de morfologia matemática e mais algumas propostas de variação deles (veja a figura 40 para ver o tipo de modificações proporcionadas por operadores).

A morfologia matemática é uma técnica muito boa para análise e transformações de formas geométricas em imagens. Ela é baseada em teorias como inclusão, união e intersecção de formas geométricas (D'ORNELLAS; BOOMGAARD, 1998). A aplicação de operadores morfológicos em imagens produz, visualmente, transformações conhecidas como dilatação e erosão de formas geométricas. Já a combinação destas duas operações consegue fazer as operações chamadas de abertura e fechamento morfológico (SERRA, 1982).

Esses operadores agem em conjunto com um elemento estruturante. O elemento estruturante é um mapa binário na forma de matriz, que aplica a operação em cada pixel da imagem. Se o pixel satisfaz a condição do operador, então ele muda de cor conforme o tipo de operador utilizado. O elemento estruturante recebe este nome porque ele faz com que a imagem do resultado da modificação com operadores morfológicos tenda para o mesmo formato da sua estrutura. Se executarmos sucessivas operações do mesmo tipo (operador com o mesmo elemento estruturante), a tendência é de que a forma alterada ganhe um aspecto mais parecido com o formato do elemento estruturante. A figura 39 é um exemplo de operações morfológicas sobre imagens.

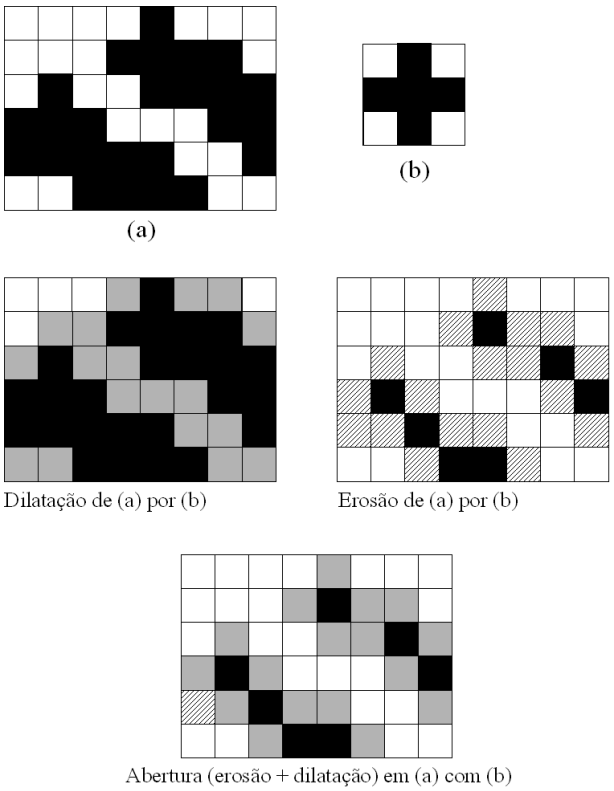

Figura 39: Exemplo de operações morfológicas sobre imagem (a), com elemento estruturante (b). Neste exemplo, os pixels em estão cinza são aqueles que serão modificados pela dilatação binária, portanto, eles estavam com a cor branca e ficarão com a cor preta. No caso da erosão binária, os pixels estão hachurados são os que serão modificados pela aplicação do operador, portanto, eles estavam com a cor preta e ficaram com a cor branca.

O operador de dilatação binária executa um aumento numa forma considerada de primeiro plano, alterando pixels de fundo por pixels de primeiro plano. Um pixel de fundo muda de valor quando, sobreposto pelo elemento estruturante B, possui pelo menos um

vizinho que seja de primeiro plano na mesma posição correspondente em B.

O operador de erosão binária executa uma diminuição numa forma considerada de primeiro plano, convertendo pixels de primeiro plano em pixels de segundo plano. Um pixel de primeiro plano troca de valor quando, sobreposto pelo elemento estruturante, não coincide em todas as posições com os pontos de primeiro plano em B.

Originalmente os operadores morfológicos foram criados para manipular imagens binárias (em preto e branco). Porém para imagens com mais cores como tons-de-cinza ou coloridas, este algoritmo sofre algumas modificações, pois não se tem mais uma idéia de primeiro plano e fundo, e sim várias tonalidades de cor. Primeiro, porque uma tonalidade não pode ser definida por apenas um elemento do espaço RGB. Por exemplo, existem diversas possibilidades de azul, e se considerarmos apenas os pixels iguais a uma determinada combinação RGB, como os nossos valores de primeiro plano, certamente muitas tonalidades de azul ficarão de fora da ação dos operadores morfológicos e também para manter as propriedades matemáticas do operador.

Os operadores morfológicos neste trabalho são implementados assim como em (LOUVERDIS et al., 2002). Resumidamente, para cada canal RGB de cada pixel do texel, o algoritmo sobrepõe o elemento estruturante e aplica a função correspondente, dilatação ou erosão. No caso da dilatação, o algoritmo obtém das posições válidas da matriz do elemento estruturante, qual é o maior valor em cada canal RGB e aplica para o pixel. Já no caso da dilatação, o algoritmo obtém o menor valor em cada canal RGB.

O efeito produzido pela dilatação é um aumento das áreas mais claras do texel e o efeito da erosão é um aumento das áreas mais escuras do texel. Veja a figura 40 para verificar o efeito dos operadores de dilatação e erosão.

Os operadores morfológicos tradicionais apenas produzem clareamento ou escurecimento de regiões da imagem, portanto não permitem uma modificação controlada por algum critério de seleção dos pixels a serem alterados. Para fazer esta modificação seletiva nós criamos algumas operações alternativas, as quais chamamos expansão de área e

Figura 40: Uso de operadores morfológicos. À esquerda o texel original, no centro o texel com uma e duas operações de dilatação; e na direita o texel com uma e duas operações de erosão.

contração de área, que assemelham-se, respectivamente, à dilatação e a erosão.

A expansão de área age sobre uma imagem dilatando ou aumentando uma área preenchida por uma determinada cor-chave dentro de um limite de tolerância. Mais especificamente, os pixels vizinhos aos pixels que satisfaçam o critério de seleção (distância RGB com norma $L2$) são afetados. Portanto, um pixel p modifica a sua cor original se:

1. a cor dele não está dentro da região de tolerância da cor-chave;
2. quando sobreposto pelo elemento estruturante, pelo menos um dos seus vizinhos coincide com uma posição do elemento estruturante.

Caso o pixel deva ser alterado ele recebe a cor média dos seus vizinhos que coincidem com posições do elemento estruturante, portanto ele recebe uma cor de primeiro plano. A figura 41 mostra numericamente o efeito da expansão de área em imagem em tons-de-cinza.

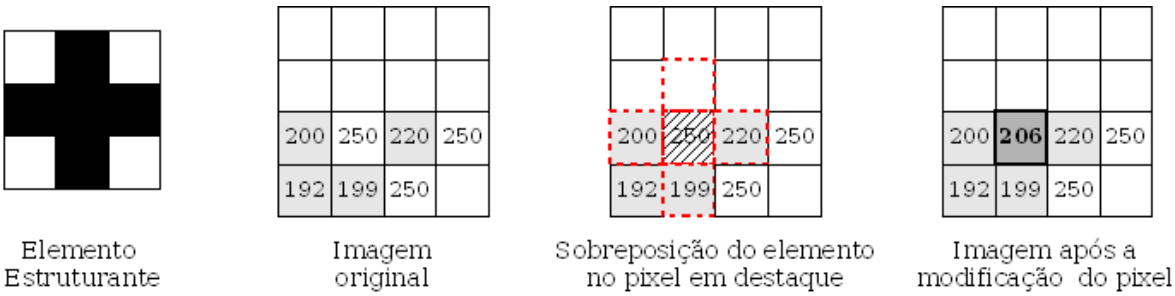

Figura 41: Exemplo com o operador de expansão. Da esquerda para a direita: o elemento estruturante, a imagem original, o pixel atual (que será afetado pelo operador) com a sobreposição do elemento estruturante (em vermelho os pixels da imagem que coincidem com posições do elemento) e a imagem modificada após esta sobreposição. O valor que o pixel recebe é a média aritmética dos valores dos vizinhos que coincidem com o elemento estruturante e são da região da cor-chave (pixels em cinza).

Já a contração faz o efeito contrário, diminui a área preenchida por uma determinada cor dentro de um limite de tolerância. Mais especificamente, somente os pixels que satisfaçam o critério de seleção (distância RGB com norma $L2$) são afetados. Portanto, um pixel p modifica a sua cor original se:

1. a cor dele está dentro da região de tolerância da cor-chave;
2. quando sobreposto pelo elemento estruturante, pelo menos um pixel vizinho não coincide com posições do elemento estruturante.

Caso todos os pixels vizinhos coincidam com as posições do elemento, o pixel permanece com a cor original. Caso o pixel deva ser alterado ele recebe a cor média dos seus vizinhos que não coincidem com posições do elemento estruturante, portanto ele recebe uma cor de "fundo". A figura 42 mostra numericamente o efeito da contração de área em imagem em tons-de-cinza.

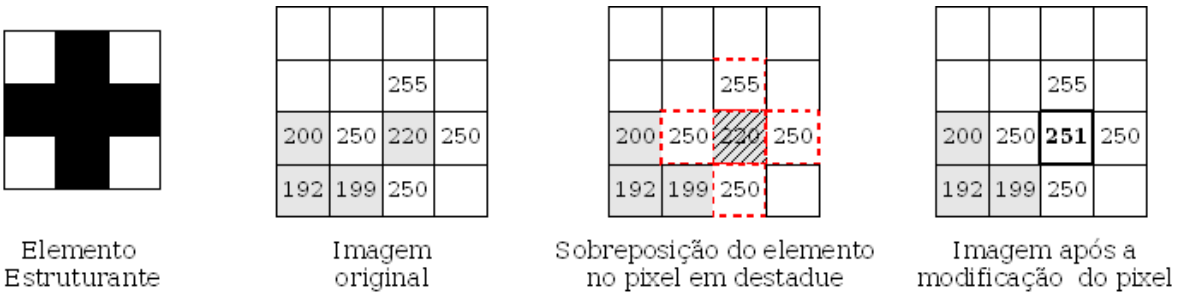

Figura 42: Exemplo com o operador de contração. Da esquerda para a direita: o elemento estruturante, a imagem original, o pixel atual (que será afetado pelo operador) com a sobreposição do elemento estruturante (em vermelho os pixels da imagem que coincidem com posições do elemento) e a imagem modificada após esta sobreposição. O valor que o pixel recebe é a média aritmética dos valores dos vizinhos que coincidem com o elemento e não são da região da cor-chave (pixels em branco). Os pixels em cinza estão dentro da região da cor-chave.

A figura 43 mostra os efeitos da aplicação dos dois operadores alternativos sobre imagens coloridas.

Estes operadores alternativos agem somente nos limites de tolerância da cor-chave, não afetando os demais pixels do texel. Isso ocorre ao contrário da ação dos operadores

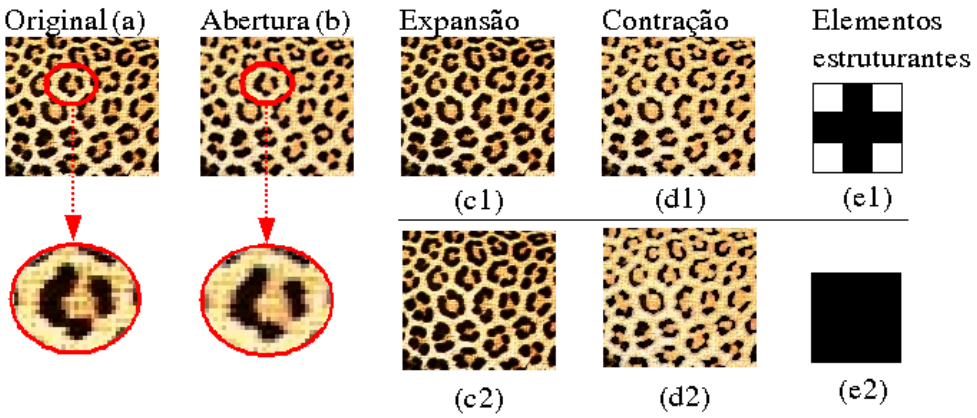

Figura 43: Uso de operadores alternativos. Em (c1) e (c2) exemplos de operadores de expansão de área, onde nota-se um aumento da área em preto (cor-chave). Em (d1) e (d2) exemplos de operadores de contração de área, onde nota-se a diminuição da área em preto (cor-chave). As imagens em (c1) e (d1) utilizam elemento estruturante (e1) em forma de "cruz" e as imagens (c2) e (d2) utilizam elemento estruturante (e2) em forma de "quadrado". A imagem (b) é resultado da aplicação de duas operações alternativas em cascata. À imagem original (a) foi aplicado primeiro a expansão e depois a contração de área. Em destaque dois texels ilustrando o efeito causando pela abertura num texel.

morfológicos tradicionais, que afetam todos os pixels do texel. A vantagem de trabalhar de forma seletiva é poder aumentar ou diminuir qualquer cor, o que é mais flexível do que apenas "clarear" ou "escurecer" os pixels.

A aplicação dos operadores no nosso algoritmo para síntese de texturas funciona da mesma maneira que a aplicação das orientações e escalas. O usuário especifica operadores-chave em determinadas posições e, depois da etapa de síntese, o sistema verifica quais texels estão dentro do raio de ação dos operadores e aplica as operações correspondentes nestes texels.

Para facilitar o uso de operadores nós permitimos que o usuário especifique mais de uma operação para cada "operador-chave". Assim ele pode configurar qualquer combinação de operações em cascata que ele desejar para aquele operador-chave. Por exemplo, ele pode aplicar duas dilatações numa região ou então combinar uma dilatação com uma expansão. A figura 44 mostra um resultado com o uso de operadores.

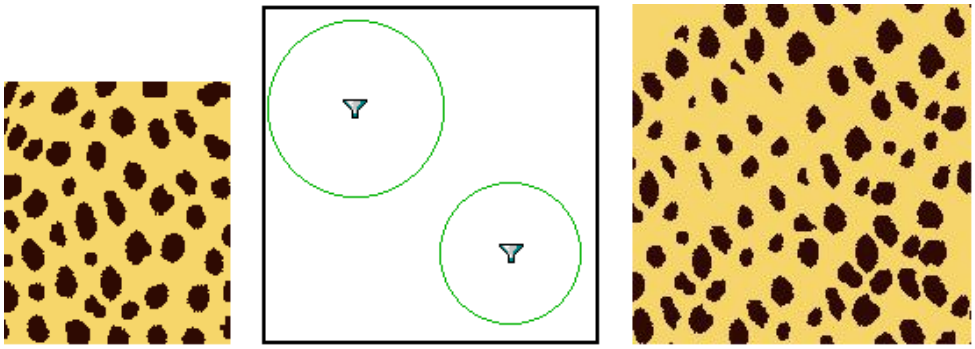

Figura 44: Resultado com uso de operadores morfológicos. À esquerda a amostra, no centro o modelo com os operadores-chave e à direita resultado com aplicação dos operadores dentro do raio de ação de cada operador-chave. Note que no canto superior esquerdo do resultado, por causa do efeito da dilatação, as estruturas diminuíram em função do aumento das áreas mais claras. Já no canto inferior direito, por causa do efeito da erosão, as estruturas aumentaram de tamanho, em função da erosão das áreas mais claras ou aumento das áreas mais escuras. Neste trabalho, em todos os resultados de síntese com uso de operadores foi utilizado o elemento estruturante em forma de "cruz" com 3 pixels de largura por 3 pixels de altura.

4.7 Preenchimento de Falhas

Ao final do processo de síntese alguns pixels permanecem sem cor. São falhas provenientes de combinações não perfeitas ou introduzidas pelos efeitos da orientação. Naturalmente estas falhas precisam ser tratadas e para isso estudamos duas possibilidades de preenchimento: utilizando a cor de fundo especificada pelo usuário ou fazer síntese pixel-a-pixel.

No nosso sistema é possível fazer o preenchimento das duas maneiras, entretanto após realizarmos diversos testes, percebemos que o simples preenchimento com a cor de fundo já produz resultados de boa qualidade e a um custo bem menor do que a síntese pixel-a-pixel. Portanto, na maioria dos resultados não é necessário apelar para um processo mais lento como a síntese pixel-a-pixel, exceto para situações onde as falhas entre texels são muito grandes, próximos ao tamanho médio de texel da amostra e o preenchimento com uma cor única apresente efeitos indesejados. Nestas situações recomendamos fortemente que o usuário utilize síntese de texturas para preenchimento das falhas. A seqüência de imagens da figura 45 mostra o resultado sem preenchimento e com preenchi-

mento com a cor de fundo.

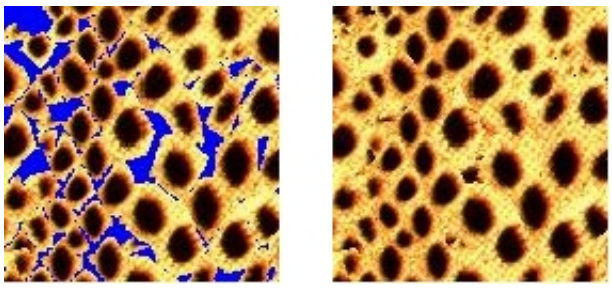

Figura 45: Resultados com e sem preenchimento de falhas.

O trabalho será considerado finalizado, quando conseguirmos gerar uma textura PVT que, visualmente, convença ao usuário que ele gerou uma imitação de textura que varia progressivamente (como exemplo da figura 2) e que esteja de acordo com o modelo alto-nível passado por parâmetro pelo mesmo.

5 RESULTADOS

Neste capítulo apresentaremos alguns resultados gerados com o nosso sintetizador e comparações com resultados apresentados por outros trabalhos, quando for o caso.

Os nossos resultados são bons para uma grande variedade de amostras, em geral para amostras que permitem uma fácil separação dos texels e que apresentam uma boa quantidade/diversidade dos mesmos. No capítulo 6 faremos uma discussão sobre algumas melhorias na interface que poderiam resolver o problema de segmentação dos texels de muitas amostras.

Este capítulo está dividido em duas seções: síntese de texturas não PVT e síntese de texturas PVT. O objetivo do trabalho é a geração das texturas tipo PVTs, entretanto, neste capítulo queremos mostrar que o modelo também é adequado para síntese de texturas homogêneas (como algumas da figura 16).

Na segunda seção, colocamos diversos modelos (M) e seus respectivos resultados para mostrar a fidelidade do sistema aos parâmetros especificados pelo usuário.

Cada seção está dividida em duas subseções por classe de textura para demonstrar a variedade de tipos de texturas que o algoritmo pode sintetizar com sucesso. Nós dividimos as texturas em artificiais e naturais. As artificiais são produzidas via modelos computacionais, enquanto que as naturais são imagens reais ou desenhos de texturas da natureza.

5.1 Síntese de Texturas Não PVT

5.1.1 Artificiais

O primeiro resultado apresentado (figura 46) é de uma textura de amostra comum em todos os trabalhos revisados sobre síntese de texturas. Apesar de permanecerem algumas falhas entre os texels sintetizados, o resultado consegue capturar o padrão da textura de amostra.

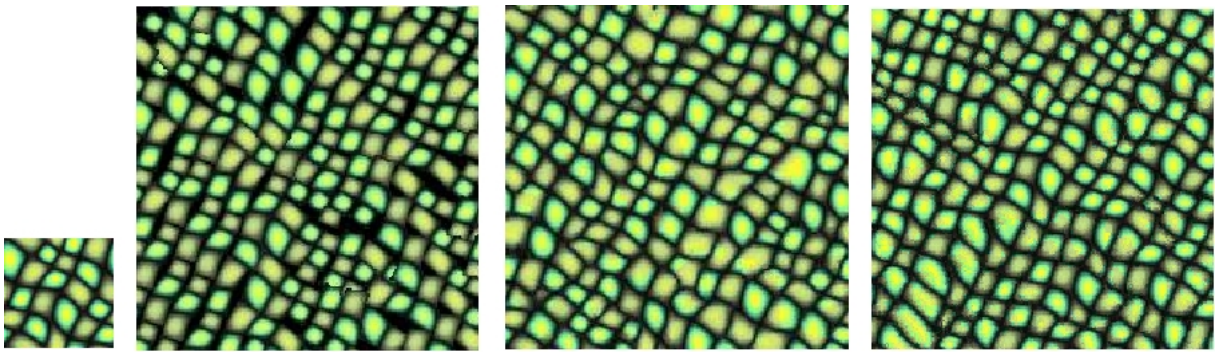

Figura 46: Resultado com a textura 161.*jpg* em (WEI; LEVOY, 2000b). Da esquerda para direita: a amostra, o resultado usando comparação com canais alfa e RGB, um resultado usando a mesma amostra com o algoritmo de (WEI; LEVOY, 2000a) e um resultado apresentado em (EFROS; LEUNG, 1999). A cor de fundo para identificação dos texels é $bg = (0, 0, 0)$, valor de limiarização para máscara é $t = 0,43$ e a tolerância de sobreposição é $ov = 0,08$.

O próximo resultado (figura 47) mostra maior fidelidade do nosso algoritmo que o algoritmo apresentado em (WEI; LEVOY, 2000a), pois conseguimos manter melhor a forma das estruturas originais.

Também para texturas bem estruturadas (ou regulares) como a amostra da figura 48 o sistema funciona bem. Repare na figura 48 que os texels mantiveram uma relação espacial muito próxima da amostra.

Para texturas artificiais que contenham alguma regularidade (figuras 49 e 50) também tivemos êxito. A textura de amostra da figura 49 foi feita com numa imagem real da pelagem do guepardo (*cheetah*). O objetivo foi mostrar que melhorando o processo de identificação dos texels é possível aumentar a quantidade de tipos de texturas que podem

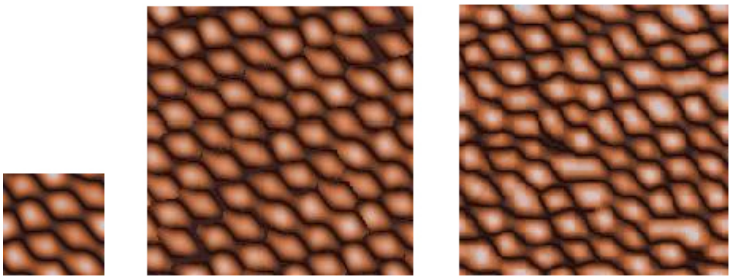

Figura 47: Resultado com a textura 295.*jpg* em (WEI; LEVOY, 2000b). Da esquerda para direita: a amostra, o nosso resultado usando comparação com canais alfa e RGB e um resultado com a mesma amostra em (WEI; LEVOY, 2000a). A cor de fundo para identificação dos texels é $bg = (58, 32, 31)$, valor de limiarização para máscara é $t = 0,25$ e a tolerância de sobreposição é $ov = 0,1$.

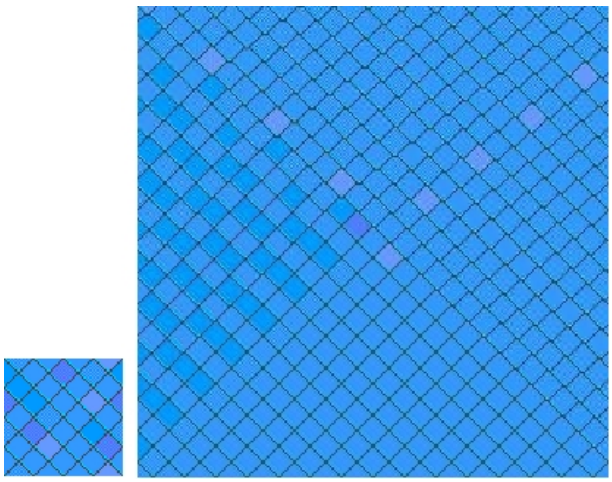

Figura 48: Resultado com a textura *blueangletile.gif* em (BOURKE, 2005). Da esquerda para direita: a amostra e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (68, 68, 68)$, valor de limiarização para máscara é $t = 0,17$ e a tolerância de sobreposição é $ov = 0,01$.

ser sintetizadas.

5.1.2 Naturais

É nesta classe de texturas que o nosso algoritmo apresenta os melhores resultados em relação a maioria dos algoritmos revisados. Isto ocorre para texturas que permitam uma fácil segmentação dos texels.

O primeiro resultado (figura 51) com textura natural é produzido com amostra proveniente de (SIMONCELLI; PORTILLA, 2001). Note que o nosso sintetizador consegue preservar melhor as estruturas características do padrão da amostra do que em (SIMON-

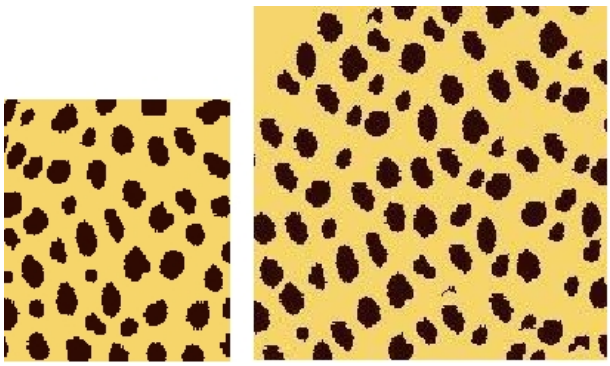

Figura 49: Resultado com a textura $cheetah2.gif$ em (WALTER, 1998) e (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). Da esquerda para direita: a amostra e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (246, 214, 106)$, valor de limiarização para máscara é $t = 0,43$ e a tolerância de sobreposição é $ov = 0,1$.

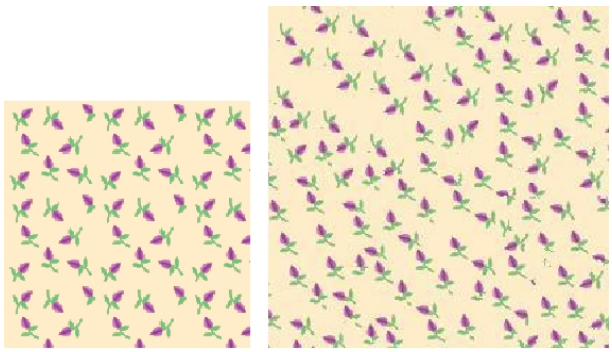

Figura 50: Resultado com a textura gera em software de edição de imagem $sample303.jpg$. Da esquerda para direita: a amostra e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (216, 130, 0)$, valor de limiarização para máscara é $t = 0,14$ e a tolerância de sobreposição é $ov = 0,17$.

CELLI; PORTILLA, 2001).

A figura 52 também mostra um comparativo com outros algoritmos e novamente o nosso algoritmo preservou melhor as estruturas características do padrão da amostra, enquanto que os outros resultados apresentam quebras de alguns texels.

Outro tipo de resultado que o nosso sistema permite realizar é a síntese partindo de um grupo de texels já "sintetizados". Neste tipo de síntese o usuário agrupa os texels-chave formando um bloxel para que o sistema, ao iniciar o processo, preencha o restante da nova textura a partir das bordas deste grupo. Algumas texturas de amostra possuem texels organizados de forma mais complexa, como por exemplo, a amostra da figura 53,

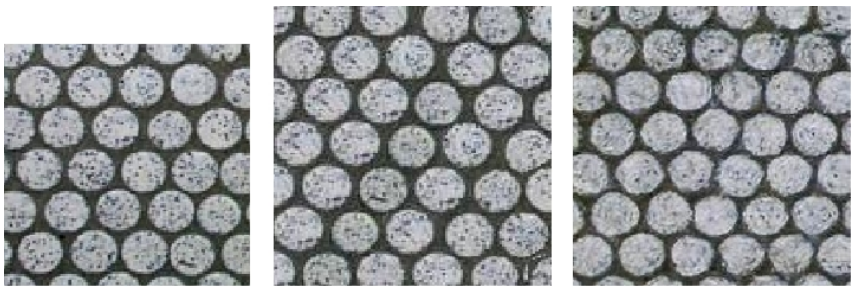

Figura 51: Resultado com a textura $d30\_1923.o.jpg$ em (SIMONCELLI; PORTILLA, 2001). Da esquerda para direita: a amostra, o resultado usando comparação com canais alfa e RGB e um resultado com a mesma amostra em (SIMONCELLI; PORTILLA, 2001). A cor de fundo para identificação dos texels é $bg = (69, 69, 59)$, valor de limiarização para máscara é $t = 0,21$ e a tolerância de sobreposição é $ov = 0,1$.

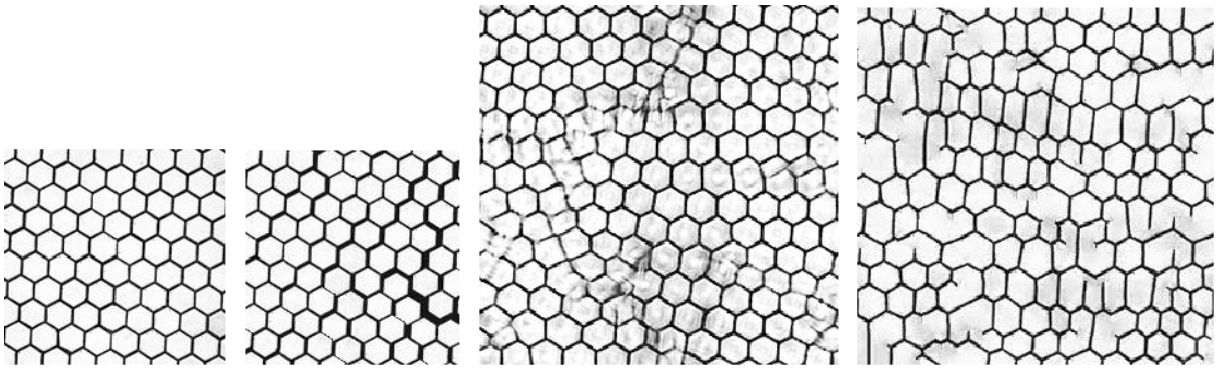

Figura 52: Resultado com a textura $tile - floor2\_c.o.jpg$ em (SIMONCELLI; PORTILLA, 2001). Da esquerda para direita: a amostra, o resultado usando comparação com canais alfa e RGB, o resultado em (SIMONCELLI; PORTILLA, 2001) e o resultado em (WEI; LEVOY, 2000a). A cor de fundo para identificação dos texels é $bg = (24, 24, 24)$, valor de limiarização para máscara é $t = 0,51$ e a tolerância de sobreposição é $ov = 0,1$.

onde os texels "invadem" a área retangular que envolve outros texels. Nestes casos, a definição de um grupo inicial auxilia o sintetizador para um melhor preenchimento da nova textura, pois o grupo de texels-chave informa ao algoritmo de síntese como e onde o ele deve conectar os próximos, aumentando o bloxel inicial a cada passo de síntese até os limites da nova textura.

Entre os tipos de texturas mais difíceis de conseguir bons resultados estão as texturas de pelagem de animais. Esta dificuldade está relacionada, principalmente, com a qualidade da amostra que, geralmente, contém com poucos texels, não é homogênea e contém pixels de "fundo" com cores muito próximas as dos texels em alguns locais da textura (veja figura 28). Em alguns casos deste tipo de textura, foi necessária a utilização

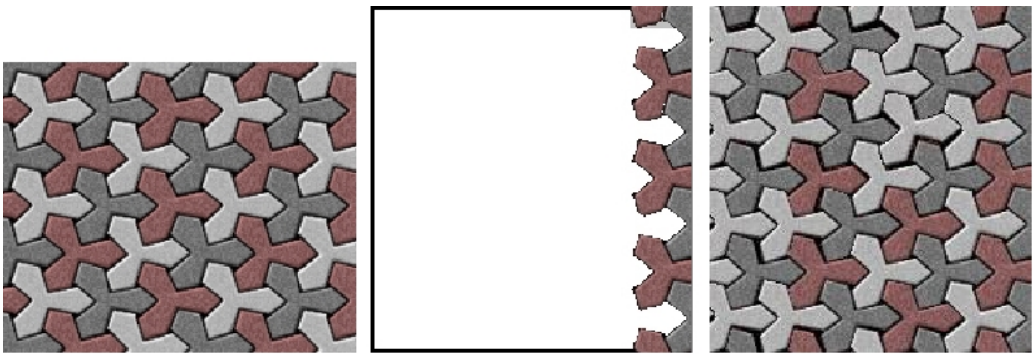

Figura 53: Resultado com a textura *tile*3.*jpg* em (BOURKE, 2005). Da esquerda para direita: a amostra, o modelo M com grupo texel-chave de início e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (0, 0, 0)$, valor de limiarização para máscara é $t = 0,28$ e a tolerância de sobreposição é $ov = 0,1$.

de uma outra imagem para identificar os texels da textura de amostra original.

As figuras 54, 55, 56, 59, 61 e 60 são exemplos de síntese com texturas de pelagem de animais. E em 56, 61 e 60 foi necessário o de imagem auxiliar.

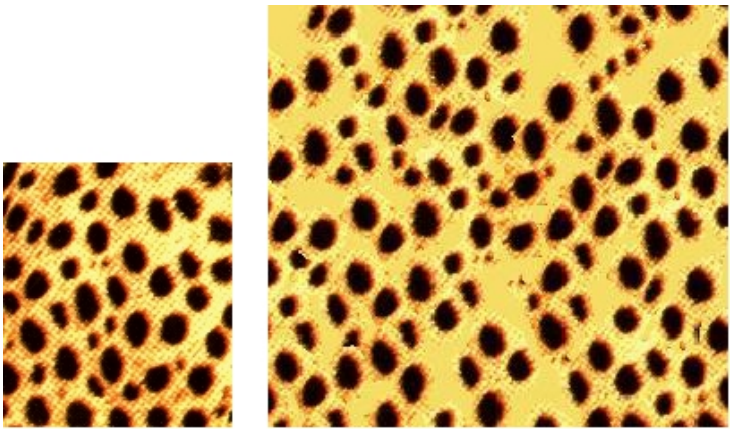

Figura 54: Resultado com a textura *cheetah.jpg* em (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). À esquerda a amostra e a direita o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (246, 221, 102)$, valor de limiarização para máscara é $t = 0,48$ e a tolerância de sobreposição é $ov = 0,05$.

5.2 Síntese de Texturas PVT

Nesta seção mostraremos alguns resultados obtidos com o nosso sintetizador usando os parâmetros de texturas PVT (orientação, escala e operadores morfológicos).

Figura 55: Resultado com a textura *skintile6.jpg* em (BOURKE, 2005). À esquerda a amostra e à direita o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (193, 143, 118)$, valor de limiarização para máscara é $t = 0,50$ e a tolerância de sobreposição é $ov = 0,1$.

5.2.1 Artificiais

No primeiro resultado 57 com texturas da classe PVT e do tipo artificial (ou gerada por computador), podemos ver a capacidade do modelo para gerar texturas PVT a partir de amostras com fácil identificação dos texels.

Na figura 58 mostra o efeito da aplicação em cascata do operador morfológico de erosão. O resultado ficou bom, entretanto, devido ao formato do texel, algumas conexões não foram feitas pelo algoritmo. Um dos motivos é que precisamos colocar um nível de tolerância à sobreposições muito rígido para que os texels não se sobrepusessem de forma indesejada.

5.2.2 Naturais

Nesta seção serão apresentados alguns resultados de síntese PVT (com transformações afim e aplicação de operadores) a partir de amostras de texturas da natureza.

Os primeiros resultados mostram como a aplicação de orientação (rotação e translação) nos texels permite produzir um efeito de "movimentação" gradual dos elementos característicos ao longo da textura (figuras 59 e 58).

A figura 60 mostra um exemplo de variação de escala em determinados locais da

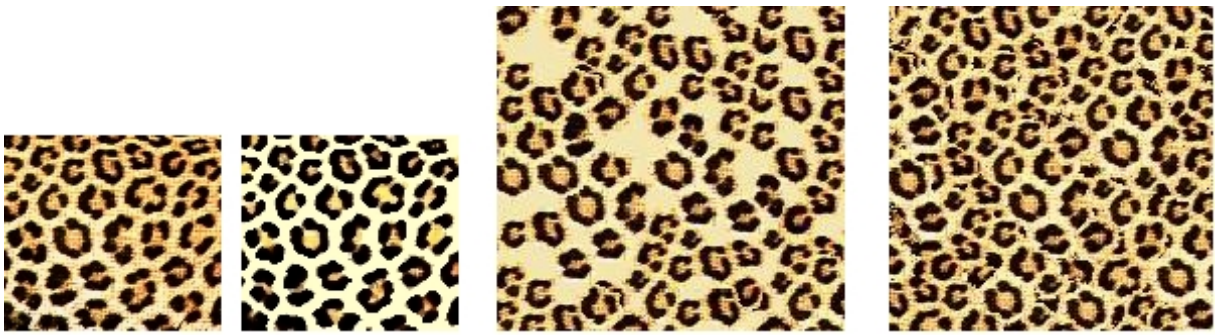

Figura 56: Resultado com a textura *realleop.jpg* em (WALTER, 1998) e (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). Da esquerda para direita: a amostra, a imagem editada da amostra, um resultado usando comparação com canais alfa e RGB e com preenchimento de cor única $(243, 231, 177)$ para falhas e um resultado com os mesmos parâmetros, porém com síntese pixel-a-pixel para preenchimento de falhas. A cor de fundo para identificação dos texels é $bg = (255, 253, 210)$, valor de limiarização para máscara é $t = 0,04$ e a tolerância de sobreposição é $ov = 0,1$. Note que no resultado com preenchimento por síntese pixel-a-pixel alguns texels estão quebrados, isto é uma conseqüência do algoritmo utilizado (WL de (WEI; LEVOY, 2000a)).

amostra, como aparece em texturas de guepardo (*cheetah*). Note que com um simples parâmetro PVT é possível simular uma textura PVT.

O último resultado ilustra um caso interessante 61. Apesar da amostra conter uma grande quantidade e variedade de texels, o que indicaria uma vantagem na síntese, o resultado não foi satisfatório na nossa opinião. O motivo é que, como a textura de amostra já carrega informações PVT no próprio padrão, o sintetizador utiliza os texels com informações "PVT" em locais não adequados e, conseqüentemente, o resultado não representa uma boa textura PVT. Este resultado abre possibilidades de investigação futura.

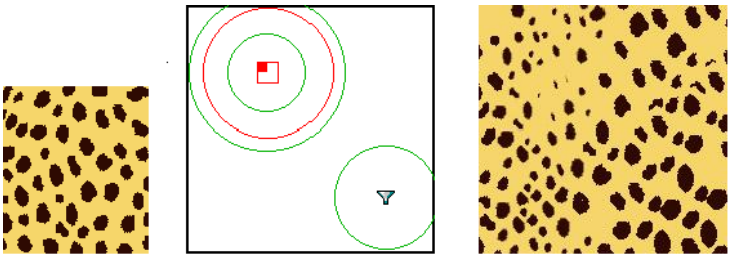

Figura 57: Resultado PVT com a textura *cheetah2.gif* em (WALTER, 1998) e (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). Da esquerda para direita: a amostra, o modelo utilizado e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (246, 214, 106)$, valor de limiarização para máscara é $t = 0,43$ e a tolerância de sobreposição é $ov = 0,1$. Os parâmetros PVT são: no canto esquerdo superior, uma escala-chave de 30%, um operador-chave de dilatação (circunferência verde maior) e um operador-chave com duas dilatações em cascata (circunferência verde menor); no canto direito inferior: um operador-chave de erosão.

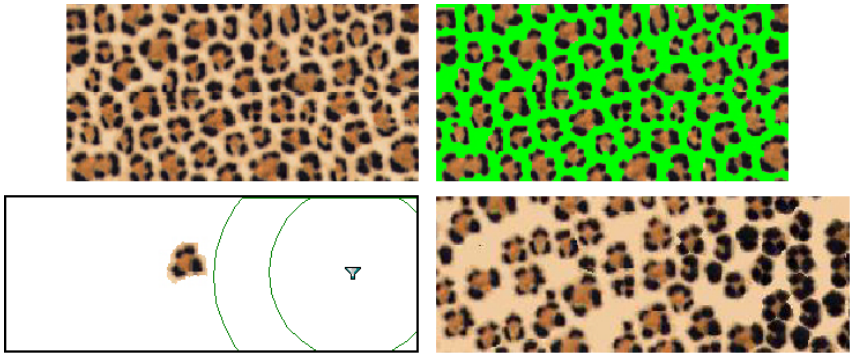

Figura 58: Resultado PVT com a textura *kingdon.jpg* em (ZHANG et al., 2003). Da esquerda para direita de cima para baixo: a amostra, a imagem editada da amostra para deixar um fundo homogêneo, o modelo utilizado e o resultado usando comparação com canais alfa e RGB e o resultado. A cor de fundo para identificação dos texels é $bg = (0, 255, 0)$, valor de limiarização para máscara é $t = 0,10$ e a tolerância de sobreposição é $ov = 0,04$. Os parâmetros PVT são: no canto direito, um operador-chave com erosão (circunferência maior) e um operador-chave com duas erosões em cascata (circunferência menor).

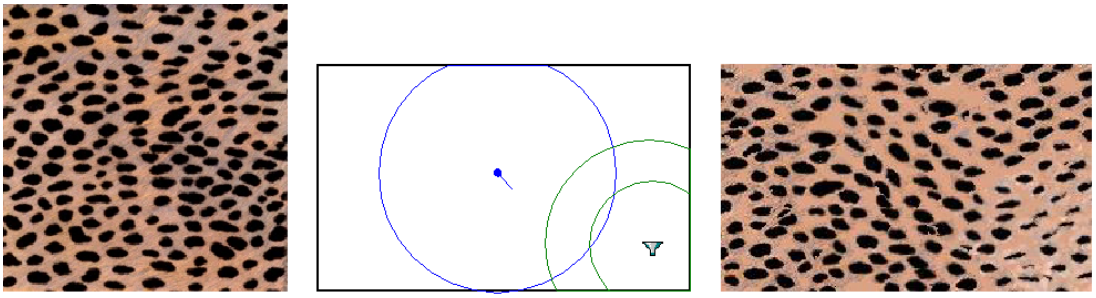

Figura 59: Resultado PVT com a textura *skintile6.jpg* em (BOURKE, 2005). Da esquerda para direita: a amostra, o modelo utilizado e o resultado usando comparação com canais alfa e RGB. A cor de fundo para identificação dos texels é $bg = (0, 255, 0)$, valor de limiarização para máscara é $t = 0,10$ e a tolerância de sobreposição é $ov = 0,04$. Os parâmetros PVT são: ao centro uma orientação-chave de 45°, no canto inferior direito um operador-chave com erosão (circunferência maior) e um operador-chave com duas erosões em cascata (circunferência menor).

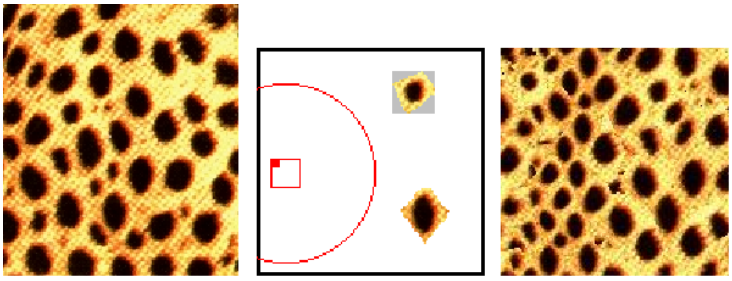

Figura 60: Resultado PVT com a textura *cheetah.jpg* em (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). Da esquerda para direita de cima para baixo: a amostra, o modelo utilizado e o resultado usando comparação com canais alfa e RGB e o resultado. A cor de fundo para identificação dos texels é $bg = (246, 221, 102)$, valor de limiarização para máscara é $t = 0,48$ e a tolerância de sobreposição é $ov = 0,05$. O parâmetro PVT utilizado foi um operador-chave com escala ($s = 30\%$).

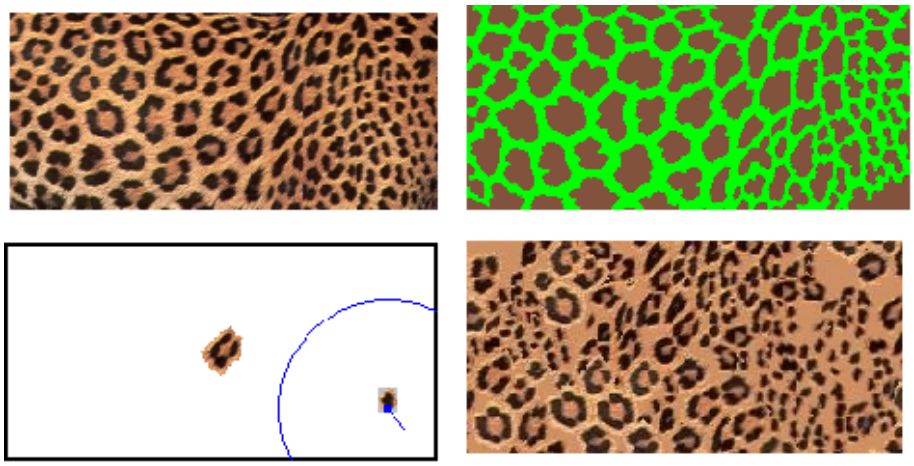

Figura 61: Resultado PVT com a textura *realleopard2.jpg* em (WALTER, 1998) e (SEIDENSTICKER; LUMPKIN; KNIGHT, 1991). Da esquerda para direita de cima para baixo: a amostra, a imagem editada da amostra para deixar um fundo homogêneo, o modelo utilizado e o resultado usando comparação com canais alfa e RGB. A cor de preenchimento do fundo é $(243, 231, 177)$. A cor de fundo para identificação dos texels é $bg = (0, 255, 0)$, valor de limiarização para máscara é $t = 0,10$ e a tolerância de sobreposição é $ov = 0,06$. Foi utilizado o parâmetro PVT orientação-chave de 45°.

6 CONCLUSÃO

Este trabalho apresentou um modelo para síntese de texturas a partir de amostras, mais especificamente para síntese de texturas com elementos que variam progressivamente ao longo da textura, conhecidas como *Progressively-Variant Textures (PVT)*. Neste modelo é proposta uma nova abordagem para síntese de texturas a partir de amostras: a síntese com preenchimento por texels. O sistema sintetiza uma nova textura partindo de um texel inicial (randômico ou definido pelo usuário) conectando novos texels até que a texura esteja completa. Esta união de texels produz uma textura homogênea, ou não-PVT. Para obter as características de uma textura PVT, podem ser aplicadas transformações afim (escala e rotação) e operadores morfológicos sobre os texels. Estas operações são definidas pelo usuário, que desta forma consegue estabelecer um certo controle sobre o resultado e consiga simular os efeitos existentes nas texturas PVT.

O modelo se mostrou eficaz para realizar as tarefas que se propôe, sintetizando tanto texturas simples como texturas do tipo PVT, como podemos verificar nos resultados. O sistema permite ainda vários tipos de controle ao usuário que pode usar da sua criatividade para gerar uma grande diversidade de resultados a partir de uma mesma amostra. Os demais sistemas de síntese revisados neste trabalho também geram bons resultados de síntese de texturas, entretanto, na maioria dos casos, estão limitados a repetição do mesmo padrão da textura de amostra. Já aqueles que são capazes de permitir ao usuário algum tipo de controle sobre o resultado final, deixam algumas pontos em aberto, como: problemas de qualidade, no resultado visual, problemas de performance inerentes ao processo pixel-a-pixel e quebra dos elementos característicos da textura, ine-

rente à fragilidade do algoritmo-base de Wey & Levoy para texturas naturais. O algoritmo apresentado neste trabalho consegue sintetizar texturas naturais com muito boa qualidade visual para vários tipos de textura e a abordagem de preenchimento por texels evita quebra (ou descaracterização) dos elementos característicos da textura de amostra.

Foi possível verificar ainda, que o algoritmo se comportou melhor com texturas cujos os texels são facilmente identificados e segmentados na máscara de textons. Para texturas onde não é possível segmentar os texels ou nos casos onde as texturas são mais próximas do limite das texturas do tipo estocásticas (veja figura 5) os resultados visuais não são tão satisfatórios. Para alguns casos de difícil segmentação dos texels permitimos ao usuário selecionar manualmente os texels ou regiões irregulares da textura, para que o sistema consiga formar um conjunto inicial de texels e, consequentemente, execute o processo normal de síntese.

As principais contribuições do trabalho no contexto de síntese de texuras são:

1. Uma nova abordagem para síntese de texturas a partir de amostras por texels, que garante a integridade dos texels e também permite ao modelo explorar modificações sobre o conjunto inicial de amostra (como espelhamentos dos texels iniciais). A abordagem permite ainda modificar o padrão visual da textura criando novos padrões (por exemplo, aumentando a tolerância à sobreposições entre texels);

2. Aplicação de transformações afim e operadores morfológicos sobre os texels, que permitem facilmente gerar resultados de texturas do tipo PVT;

3. Maior flexibilidade no controle do resultado final, com diversos tipos de controle de alto-nível para usuário;

4. Possibilidade de gerar uma grande diversidade visual de resultados a partir de uma mesma amostra, apenas modificando os parâmetros iniciais, como texels-chave e parâmetros para gerar PVT. Esta possibilidade soluciona o problema de gerar variações individualizadas de uma mesma textura, necessária em algumas tarefas de

computação gráfica.

6.1 Trabalhos Futuros

Através da análise dos diversos teste realizados com o nosso sintetizador de texturas, percebemos que algumas funcionalidades poderiam ser melhor exploradas, como a segmentação dos texels da amostra, outros tipos de operações além da morfologia matemática e um algoritmo mais eficiente para preenchimento das falhas.

A principal dificuldade encontrada para obter bons resultados foi a identificação e separação de texels da amostra. Em muitas exemplos de amostra, simplesmente não foi possível identificar os texels por causa de problemas relacionados a aquisição das imagens, geralmente, de fotos digitalizadas. Estas imagens apresentam muitas variedades de cor de fundo, problemas de iluminação e em alguns casos, devidos a alguns artefatos incluídos pelo equipamento/software de digitalização alguns pixels dos texels recebem uma cor parecida com a cor do fundo, inviabilizando a correta separação dos mesmos.

Uma solução encontrada para amenizar este problema foi carregar duas imagens de entrada, a amostra original e uma outra imagem modificada num editor de imagens externo para facilitar a identificação dos texels. Neste caso, seria possível adicionar ao sistema ferramentas de edição para produzir alterações diretamente na textura amostra dentro do sistema, para facilitar a identificação dos texels.

Quanto ao algoritmo de segmentação dos textons na máscara, seria interessante o uso de alguma outra técnica para segmentação. Uma possibilidade seria o uso da técnica de *watershed*, que é uma técnica conhecida para a segmentação de imagens.

O uso de outras operações ou filtros de imagem sobre os texels poderia produzir uma diversidade de resultados anda maior, podendo inclusive permitir a confecção de um novo padrão visual da textura.

Por fim, a alteração do algoritmo de preenchimento de falhas poderia produzir

resultados melhores a um custo computacional bem menor. Uma possibilidade seria o uso do algoritmo proposto em (ASHIKHMIN, 2001), que é bom para a classe de texturas naturais. Ou ainda, o que seria melhor, a inclusão de mais uma etapa de síntese, onde se identifica as regiões das falhas, calcula-se a densidade de pontos dessas (número de pixels transparentes dividido pelo tamanho da área) e procura-se, no conjunto de texels, um texel que preencha este espaço, levando em consideração o parâmetro de tolerância à sobreposições.

REFERÊNCIAS

ASHIKHMIN, M. Synthesizing natural textures. In: *Symposium on Interactive 3D Graphics*. [S.l.: s.n.], 2001. p. 217–226.

BOURKE, P. *Textures*. 2005. http://astronomy.swin.edu.au/~ pbourke/texture/.

CATMULL, E. E. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. 77 p. Tese (Ph.D. Thesis) — University of Utah, December 1974.

CROSS, G.; JAIN, A. K. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 5, n. 1, p. 25–39, jan. 1983.

de Bonet, J. S. Multiresolution sampling procedure for analysis and synthesis of texture images. In: WHITTED, T. (Ed.). *SIGGRAPH 97 Conference Proceedings*. [S.l.]: Addison Wesley, 1997. (Annual Conference Series), p. 361–368. ISBN 0-89791-896-7.

D'ORNELLAS, M.; BOOMGAARD, R. V. D. *Generic Algorithms for Morphological Image Operations - A Case Study Using Watersheds*. 1998.

EBERT, D. S. et al. *Texturing and Modeling: A Procedural Approach*. Third. [S.l.]: Morgan Kaufmann Publishers, 2002. ISBN 1558608486.

EFROS, A. A.; FREEMAN, W. T. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, p. 341–346, August 2001.

EFROS, A. A.; LEUNG, T. K. Texture synthesis by non-parametric sampling. In: *IEEE International Conference on Computer Vision*. Corfu, Greece: [s.n.], 1999. p. 1033–1038.

FOLEY, J. et al. *Computer Graphics: Principles and Practice*. Second. [S.l.]: Addison-Wesley Publishing Company, 1990. (The systems Programming Series). ISBN 02001121107.

FU, K. S.; LU, S. Y. Computer generation of texture using a syntactic approach. *Computer Graphics (SIGGRAPH '78 Proceedings)*, v. 12, n. 3, p. 147–152, ago. 1978.

GAGALOWICZ, A.; MA, S. Model driven synthesis of natural textures for 3-D scenes. In: *Eurographics '85*. [S.l.: s.n.], 1985. p. 91–108.

HECKBERT, P. S. Survey of texture mapping. In: *Graphics Interface '86*. [S.l.: s.n.], 1986. p. 207–212.

HEEGER, D. J.; BERGEN, J. R. Pyramid-based texture analysis/synthesis. In: COOK, R. (Ed.). *Computer Graphics (SIGGRAPH '95 Proceedings)*. [S.l.: s.n.], 1995. p. 229–238.

LIANG, L. et al. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG)*, ACM Press, v. 20, n. 3, p. 127–150, 2001. ISSN 0730-0301.

LOUVERDIS, G. et al. A new approach to morphological color image processing. *Pattern Recognition*, v. 35, n. 8, p. 1733–1741, august 2002.

LU, S.; FU, K. A syntactic approach to texture analysis. *Computer Graphics and Image Processing*, v. 7, n. 3, p. 303–30, jun. 1978.

MONNE, J.; SCHMITT, F.; MASSALOUX, D. Bidimensional texture synthesis by markov chains. *Computer Graphics and Image Processing*, v. 17, n. 1, p. 1–23, set. 1981.

SEIDENSTICKER, J.; LUMPKIN, S.; KNIGHT, F. *Great Cats*. First. [S.l.]: St Martins Pr, 1991. ISBN 0878579656.

SERRA, J. *Image Analysis and Mathematical Morphology*. First. [S.l.]: Academic Press, 1982. ISBN 012637242X.

SIMONCELLI, E.; PORTILLA, J. Texture characterization via joint statistics of wavelet coefficient magnitudes. In: *Fifth IEEE International Conf on Image Processing*. Chicago, Illinois: IEEE Computer Society, 1998. I, p. 62–66.

SIMONCELLI, E.; PORTILLA, J. *Parametric Model for Visual Texture Representation and Synthesis*. April 2001. http://www.cns.nyu.edu/ eero/texture/.

SZELISKI, R.; SHUM, H.-Y. Creating full view panoramic image mosaics and environment maps. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. [S.l.]: ACM Press/Addison-Wesley Publishing Co., 1997. p. 251–258. ISBN 0-89791-896-7.

TONIETTO, L.; WALTER, M. Towards local control for image-based texture synthesis. In: SBC - BRAZILIAN COMPUTER SOCIETY. *Proceedings of Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI 2002*. IEEE Computer Society, 2002. Disponível em: <http://iris.sid.inpe.br:1906/rep/sid.inpe.br/banon/2002/10.23.11.47>.

WALTER, M. *Integration of Complex Shapes and Natural Patterns*. Tese (Doutorado) — The University of British Columbia, 1998.

WALTER, M.; FOURNIER, A.; MENEVAUX, D. Integrating shape and pattern in mammalian models. In: *Proceedings of ACM SIGGRAPH 2001*. [S.l.]: ACM Press, 2001. p. 317–326.

WEI, L.-Y.; LEVOY, M. Fast texture synthesis using tree-structured vector quantization. In: *Proceedings of ACM SIGGRAPH 2000*. [S.l.: s.n.], 2000. (Computer Graphics Proceedings, Annual Conference Series), p. 479–488.

WEI, L.-Y.; LEVOY, M. *Texture Analysis and Synthesis*. 2000. http://www.graphics.stanford.edu/projects/texture/.

ZHANG, J. et al. Synthesis of progressively variant textures on arbitrary surfaces. *ACM Transactions on Graphics*, v. 22, n. 3, p. 295–302, jul. 2003.