3
Técnicas de visualização aplicadas à Escultura do Espaço

3.1
Introdução

No capítulo anterior apresentamos alguns dos principais conceitos sobre reconstrução volumétrica de cenas a partir de imagens, além de expor alguns dos principais métodos de coloração de voxels e escultura do espaço encontrados na literatura. Neste capítulo iremos descrever a aplicação de técnicas tradicionalmente utilizadas em síntese de imagens para o tratamento de algumas questões relevantes aos métodos de escultura do espaço. Especificamente, discutiremos a realização do processamento diretamente no espaço da cena, ao contrário da forma convencional, que trabalha no espaço das imagens.

Uma das vantagens em se utilizar técnicas de visualização para o tratamento de alguns dos subproblemas encontrados nos métodos de escultura do espaço é a de que estas técnicas, na maioria das vezes, se encontram implementadas em *hardware* altamente otimizado, permitido assim a construção de algoritmos concisos e eficientes.

Alguns dos trabalhos que também seguiram esta linha de abordagem foram o trabalho de Prock [52], o primeiro a sugerir a utilização de técnicas de síntese de imagens para o desenvolvimento de algoritmos eficientes para o problema de coloração de voxels e o trabalho de Sainz et al. [77] que, apesar de ter sido desenvolvido de forma completamente independente de nosso trabalho, possui com este vários pontos em comum, ainda que diferenças importantes possam ser encontradas, como veremos no próximo capítulo.

3.2
Alguns aspectos pouco explorados

Os primeiros trabalhos sobre a reconstrução de cenas baseados em escultura do espaço deram grande ênfase aos aspectos conceituais do proble–

ma, o que é natural em contribuições seminais [44, 50, 56]. O objetivo principal destes trabalhos era introduzir na literatura uma nova técnica, de fácil compreensão e de simples implementação, que pudesse se apresentar como uma nova alternativa ao conjunto de técnicas já existentes [37, 39, 40, 41, 42, 43] para a reconstrução de cenas a partir de imagens.

Em um segundo momento, enquanto alguns trabalhos continuaram a busca por um modelo teórico definitivo para o problema [55, 63], outros começaram a investigar questões um pouco mais específicas como, por exemplo, o uso de diferentes estatísticas na determinação da foto-consistência [71], o estudo de espaços volumétricos alternativos [58, 69, 70, 68] e a introdução de modelos probabilísticos [71, 72, 73]. Além disso, à medida em que a teoria sobre reconstrução a partir de imagens se tornou mais sólida, alguns problemas importantes do ponto de vista mais prático também começaram a ser focalizados, como o aumento de eficiência dos algoritmos [52], o uso de estruturas de dados específicas para manutenção das informações de visibilidade [57], a busca por robustez em face a erros de calibração [65] e o estudo de técnicas para o refinamento da reconstrução [67].

Entretanto, apesar de todo este esforço, alguns aspectos relevantes, até mesmo do ponto de vista conceitual, ainda permanecem pouco explorados. Como exemplos, podemos citar a relação existente entre o cálculo da foto-consistência e os problemas clássicos de reconstrução e amostragem de imagens, cuja compreensão é fundamental para que o resultado final do processo não seja afetado por erros provenientes de *aliasing*, o que muitas das vezes pode ser crítico quando somado aos erros inerentes à calibração e a ruído introduzido nas imagens pelos sistemas de aquisição.

Um outro aspecto não muito explorado é o de como integrar as informações sobre a segmentação do objeto nas imagens de entrada de modo coerente com as técnicas de síntese de imagens que aqui descreveremos. Mostraremos, no capítulo seguinte, que a integração destas informações pode ser extremamente útil ao processo de reconstrução, e, em alguns casos, fundamental, como no desenvolvimento de algoritmos adaptativos, o qual será abordado no capítulo 5.

Nas seções seguintes, iremos descrever alguns dos problemas encontrados nas implementações mais ingênuas do algoritmo de escultura do espaço, que se caracterizam basicamente por trabalhar no espaço das imagens, e discutir como estes problemas podem ser solucionados ao se trabalhar no espaço da cena, através da utilização de técnicas empregadas tradicionalmente em síntese de imagens.

3.3 Processamento no espaço das imagens

Como vimos no capítulo anterior, os algoritmos de escultura do espaço procuram determinar a forma de uma cena através da avaliação da foto-consistência de cada um dos voxels que compõem o volume de reconstrução inicial, o que é feito por meio do cálculo de uma medida estatística entre os pixels pertencentes às projeções destes mesmos voxels em cada uma das imagens. Logo, a projeção dos voxels, o registro e a rasterização de seus elementos constituem passos fundamentais para a determinação de quais são os elementos que fazem parte da cena a ser reconstruída.

Devido às transformações de câmera e de perspectiva, os voxels que compõem o volume de reconstrução inicial se projetam em forma de quadriláteros arbitrários, tornando não trivial tanto o registro quanto a rasterização dos pixels correspondentes.

Podemos contornar este problema ao assumir que os voxels são pequenos o suficiente para que possam ser aproximados por seus centróides; como resultado, a projeção de um voxel em uma imagem se reduz a um pixel, o que torna o registro praticamente imediato. Apesar de bastante simples, esta solução introduz problemas graves de amostragem, principalmente quando as imagens são geradas através de projeções perspectivas. Além disso, neste caso, o registro das informações de visibilidade com base em centróides não reflete corretamente a visibilidade dos voxels fazendo com que muitos dos que deveriam ser considerados não visíveis acabam sendo erroneamente tratados como visíveis.

O posicionamento das câmeras a distâncias arbitrárias também pode acrescentar uma nova variável ao problema, já que se torna difícil, neste caso, especificar uma resolução única para a discretização do espaço 3D de forma que a aproximação pontual dos voxels seja adequada para cada uma das câmeras utilizadas.

Como podemos perceber, o processamento no espaço das imagens desencadeia uma série de problemas que podem comprometer seriamente a qualidade da reconstrução obtida, e até mesmo sua corretude, se cuidados necessários não forem tomados e se certas condições não forem consideradas.

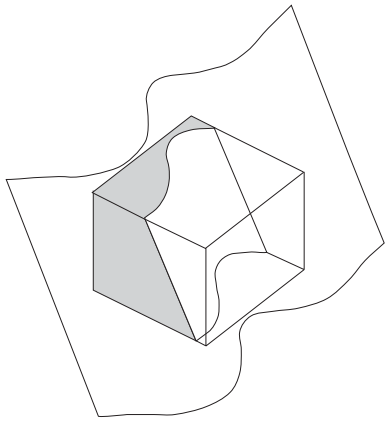


Figura 3.1: Superfície no interior de um voxel

3.4 Processamento no espaço da cena

Com o objetivo de evitar os problemas causados pelo processamento no espaço da imagem, Prock propôs que a escultura do espaço fosse realizada diretamente no espaço da cena.

Prock percebeu que da mesma forma como podemos trabalhar no espaço das imagens reprojetando voxels nas imagens de entrada, podemos trabalhar no espaço da cena através da projeção das imagens de entrada sobre superfícies passando pelo interior dos voxels (Figura 3.1). Entretanto, para que possamos adotar esta estratégia, precisamos primeiramente especificar quais são as superfícies sobre as quais as imagens de entrada devem ser projetadas.

Infelizmente não temos a menor idéia sobre quais são estas superfícies, o que nos obriga a verificar a foto-consistência em todas as possíveis superfícies que passam por cada um dos voxels do volume de reconstrução inicial, o que é obviamente impossível.

Na verdade, poderíamos estimar as superfícies do objeto passando pelo interior dos voxels com base na triangulação dual da envoltória visual [78], porém, por enquanto, não iremos explorar esta possibilidade, a qual levaria a uma nova classe de algoritmos de reconstrução. Na seção sobre trabalhos futuros discutiremos mais sobre esta possibilidade. Um solução mais simples pode ser obtida através da aproximação da geometria de um fragmento de superfície, potencialmente foto-consistente, por um plano cujo posicionamento e orientação devem ser especificados convenientemente. Obviamente, temos total liberdade para a escolha destes parâmetros em cada voxel. No entanto, o processamento pode tornar-se extremamente caro

se for necessário projetar as imagens de entrada em um plano distinto para cada voxel que vier a ser avaliado. Logo, lucraremos muito mais se pudermos avaliar a foto-consistência de um conjunto de voxels com base na projeção do conjunto de imagens em uma única superfície.

Podemos alcançar este objetivo tomando a direção de varredura como a orientação dos planos que irão compor o conjunto de superfícies candidatas. Desta forma, um único plano determina todos os fragmentos de superfícies candidatos para um conjunto de voxels que se encontram a uma distância fixa das câmeras de entrada. Desta forma, um plano com mesma orientação da direção de varredura, devidamente transladado, aproxima uma camada de voxels induzida pela partição do espaço e pela função de distância em relação ao conjunto de câmeras.

Várias escolhas podem ser feitas com relação ao posicionamento, sendo que nenhuma delas é significativamente mais vantajosa que as demais. Na maioria das vezes opta-se pela posição central (passando pelo centro do voxel), já que é a posição mais neutra, ou a posição frontal (a qual corresponde à face frontal do voxel), que por sua vez, é a superfície mais próxima ao conjunto de câmeras consideradas em uma varredura, no caso do algoritmo de escultura do espaço, ou do conjunto completo de câmeras, no caso do algoritmo de coloração de voxels (Figura 3.2).

Desta forma, definimos o conjunto de superfícies de projeção como um conjunto de planos π_k, os quais denominamos *planos de referência*, com mesma orientação da direção de varredura e com distância $dist(k)$ crescente em relação ao conjunto de câmeras consideradas.

A interseção de cada π_k com o conjunto de voxels V determina, no espaço da cena, um conjunto de regiões planares $RP = \{rp_{ijk} | rp_{ijk} \in \pi_k \cap V\}$ sobre as quais a etapa de determinação da foto-consistência deve ser efetuada. Por enquanto, não nos preocuparemos em como a amostragem deve ser feita sobre as imagens projetadas em cada uma destas regiões pois estamos interessados apenas, no momento, em como as imagens de entrada podem ser registradas no espaço da cena.

Ainda que não seja exata, esta solução é bastante superior à aproxi–mação pontual descrita anteriormente. Uma de suas vantagens é a de que, quando as imagens são projetadas em um plano de referência π_k, o regis–tro entre elas é automaticamente determinado em cada uma das regiões planares associadas, já que cada uma destas regiões recebe a contribuição fotométrica correspondente às projeções dos respectivos voxels no espaço da imagem. Uma outra vantagem é que o registro da visibilidade também pode ser feito de forma correta, ao contrário do que ocorre com o processamento

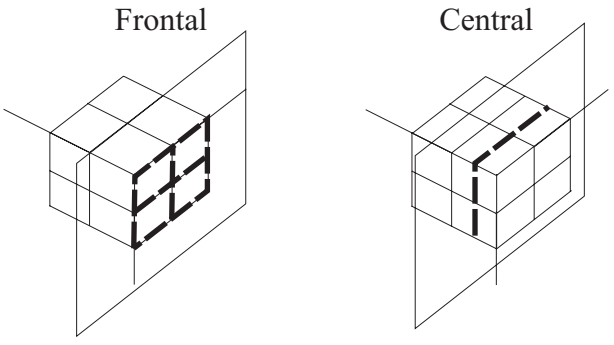


Figura 3.2: Planos de referência

no espaço das imagens.

As vantagens associadas à projeção das imagens no espaço da cena não se restringem somente à solução do problema de registro; ela também permite que um número grande de voxels possa ser processado de forma mais eficiente. Através desta estratégia podemos substituir a projeção e rasterização de milhões de voxels no espaço das imagens pela projeção e rasterização de apenas algumas centenas de imagens no espaço da cena, o que corresponde a um ganho considerável em eficiência.

Matematicamente, o mapeamento das imagens no espaço da cena corresponde a uma operação de *warping projetivo*, cuja complexidade pode ser proibitiva se considerarmos a utilização de técnicas de filtragem e anti-aliasing, mesmo quando implementada através de algoritmos que trabalham separadamente em cada uma das orientações naturais das imagens. Uma alternativa mais promissora consiste em efetuar o warping projetivo das imagens em hardware gráfico, através da projeção das mesmas como mapas de textura sobre a superfície determinada pelo plano de referência.

Esta técnica, conhecida como mapeamento projetivo de texturas, será utilizada neste trabalho para registrar, no espaço da cena, a maior parte das informações necessárias à reconstrução por escultura do espaço. Por este motivo, iremos dedicar algumas seções na descrição detalhada desta técnica, de seus principais problemas e de suas implicações diretas ou indiretas sobre os métodos de reconstrução que aqui serão apresentados.

3.4.1 Mapeamento projetivo de texturas

O *mapeamento projetivo de texturas*, introduzido por Segal et al. [28], é uma generalização do mapeamento de texturas clássico utilizado desde os primórdios da computação gráfica. Basicamente, sua idéia consiste em

projetar uma textura sobre uma superfície da cena, de forma análoga a uma projeção de slides (Figura 3.3). Várias são as suas aplicações, dentre as quais podemos destacar a produção de efeitos especiais de iluminação e a geração de sombras (*shadow maps*), os quais incrementam consideravelmente o realismo de cenas 3D produzidas sinteticamente (Figura 3.4).

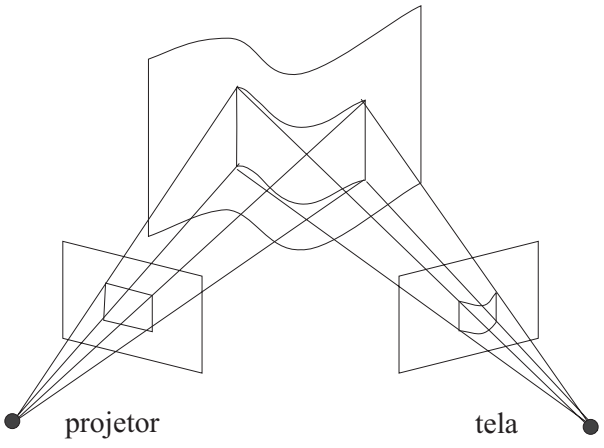


Figura 3.3: Mapeamento de textura sobre uma superfície

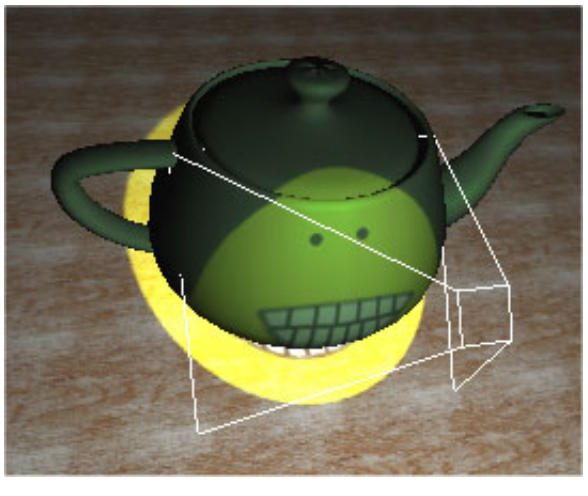
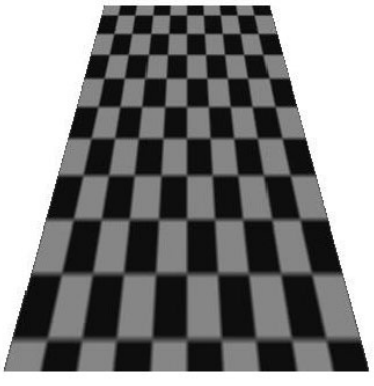


Figura 3.4: Mapeamento projetivo de textura

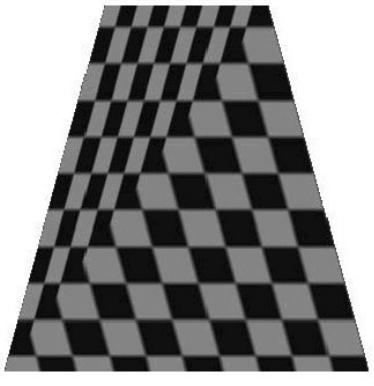
No mapeamento projetivo de textura, a relação entre as coordenadas no espaço de textura e as coordenadas no espaço do objeto são dadas por uma transformação projetiva, enquanto que no mapeamento de textura convencional esta relação é dada por uma transformação afim.

Consideremos, por exemplo, um mapeamento de textura projetivo bidimensional. Sejam (s, t, q) as coordenadas de textura homogêneas de um ponto p pertencente a uma região poligonal P, as quais são determinadas a partir da interpolação das coordenadas de textura homogêneas nos vértices de P. As coordenadas de textura reais correspondentes ao ponto p são obtidas então através da projeção das coordenadas de textura projetadas $(s/q, t/q, 1)$ no plano de suporte da textura. A coordenada q neste caso nada mais é que a distância do ponto em relação ao plano de suporte da textura.

As imagens abaixo ilustram as diferenças entre realizar a interpolação de coordenadas no espaço projetivo e no espaço real.



3.5(a): Mapeamento projetivo correto



3.5(b): Mapeamento projetivo incorreto

Para que o mapeamento projetivo possa ser realizado é necessário então atribuir coordenadas homogêneas de textura aos vértices que definem a primitiva a ser texturizada. A biblioteca OpenGL [1], por exemplo, adota o paradigma de projetor de slides sintético para representar o processo de mapeamento projetivo de textura. Isto significa que podemos definir uma matriz modelview e uma matriz de projeção a um projetor de slides que modela o efeito de um mapeamento projetivo de texturas, assim como fazemos na definição de uma câmera sintética. Desta forma, a relação entre o sistema de coordenadas do projetor e da câmera ficam determinadas de forma bastante consistente com a arquitetura da biblioteca.

Para que as coordenadas homogêneas fiquem completamente especificadas, a biblioteca OpenGL fornece um mecanismo para atribuir automaticamente coordenadas de textura aos vértices de uma determinada primitiva

com base em outros atributos de tais vértices. No caso de mapeamento projetivo de textura os dois modos que podem ser utilizados são o *eye linear texture generation* e o *object linear texture generation*, os quais atribuem coordenadas de textura aos vértices de um primitiva a partir de suas coordenadas no sistema de coordenadas da câmera e do objeto respectivamente. Em ambos os casos, cada uma das componentes das coordenadas de textura de um dado vértice são avaliadas através do valor fornecido por uma equação do plano no vértice em questão.

A atribuição automática de coordenadas de textura gera coordenadas de textura mesmo para pontos fora da região de projeção. Neste caso, como as coordenadas calculadas estão fora do domínio no qual a textura está definida, o processo de mapeamento de textura funcionará por replicação, o que significa que ele atribuirá as cores do elemento de textura mais próximo das coordenadas calculadas originalmente. Isto significa que a textura será extendida por todo a primitiva, o que não é adequado para os nossos propósitos. Para evitar este efeito é preciso criar uma borda transparente de modo que a textura projetada fique limita a área de projeção do projeto sintético.

Este foi o principal motivo que nos levou a abandonar o procedimento de atribuição automática de coordenadas de textura da biblioteca OpenGL, já que seria necessário adicionar bordas transparentes às imagens originais, durante a criação da textura, para que o mapeamento desejado pudesse ser obtido. Optamos então por utilizar o próprio mapeamento de textura convencional do OpenGL o que, no entanto, fez com que fosse necessário determinar explicitamente as coordenadas dos vértices que definem a área projetada, no sistema de coordenadas do mundo, juntamente com as suas coordenadas textura. As coordenadas de textura de x_t e y_t de cada vértice são determinadas de modo idêntico ao mapeamento de textura convencional, enquanto que a coordenada z_t deve ser igual a zero, uma vez que trabalhos com texturas bidimensionais. A coordenada w_t por sua vez é determinada calculando-se a distância do vértice em relação ao plano de suporte da textura, o que equivale a calcular uma equação do plano no vértice em questão, exatamente como é feito nos modos *eye linear* e *object linear* do processo de atribuição automática de coordenadas de textura da biblioteca OpenGL. Para maiores detalhes ver [28].

3.4.2 Reamostragem em mapeamentos de textura

A reamostragem de uma mapa de textura é necessária porque, em geral, não há uma associação direta entre um elemento de textura (*texel*) e as coordenadas de textura associadas a um pixel da tela, uma vez que os reticulados definidos no espaço da tela e no espaço de textura podem diferir tanto em resolução quanto em geometria.

A forma mais ingênua de se efetuar este procedimento consiste em determinar, para um determinado pixel da tela com coordenadas p_s, o elemento do mapa de textura cujas coordenadas se encontram mais próximas das coordenadas de textura p_t e em seguida atribuir os valores das suas componentes de cor e opacidade às componentes do pixel correspondente.

Esta solução, apesar de simples, pode produzir efeitos indesejáveis, principalmente quando o mapa de textura apresenta regiões com freqüências muito elevadas quando comparadas à taxa de amostragem. Este fenômeno, conhecido como *aliasing*, está intrinsecamente associado ao fato de precisarmos representar sinais contínuos de forma discreta em sistemas digitais.

Os aspectos relacionados à reconstrução e amostragem de imagens normalmente são estudadas dentro do contexto de *processamento de sinais*. De fato, o conceito de sinal é bastante adequado para o tratamento de imagens, já que estas podem ser vistas como grandezas fotométricas que variam em um subconjunto do espaço.

Por este motivo, iremos rever alguns dos principais conceitos e ferramentas utilizadas em processamento de sinais para que possamos efetuar uma análise sobre o processo de reamostragem que deve ser realizado durante o mapeamento projetivo de texturas.

Processamento de imagens como sinais

A grande maioria dos sinais provenientes do universo físico são de natureza contínua, como por exemplo, as imagens capturadas por uma câmera, o que nos obriga a encontrar métodos capazes de representá-los de forma adequada para que possamos processá-los em sistemas digitais, os quais, por definição, trabalham em domínios discretos.

Por esta razão, é fundamental, em processamento de sinais digitais, a análise dos procedimentos para obtenção de representações discretas de sinais contínuos, assim como dos procedimentos que buscam reconstituir um sinal contínuo a partir de representações discretas. Os primeiros procedimentos são conhecidos como *amostragem* e os últimos como *reconstrução*.

Dizemos que uma reconstrução é exata quando ela é capaz de reconstituir um sinal original, a partir de sua representação discreta, sem perdas de informação. Caso contrário dizemos que ela é uma reconstrução apenas aproximada.

A análise das relações entre a representação e a reconstrução de um sinal contínuo, e em que condições estes processos podem ser efetuados sem perda significativa de informação, é de suma importância para a área de processamento de sinais, e normalmente requer um tratamento mais rigoroso através de modelos e ferramentas matemáticas.

Modelos matemáticos para sinais

Um sinal pode ser modelado matematicamente através de uma função da forma $f : R^n \rightarrow V$, isto é, através de uma função que varia em um espaço n-dimensional tomando valores em um espaço vetorial.

Definição 3.4.1 *Um* espaço de sinais *S é um subespaço vetorial do espaço de funções $f : U \in R^n \rightarrow V^m$, no qual U,V e m são fixados.*

Os *modelos funcionais* são os modelos matemáticos mais adequados ao estudo de sinais, já que são capazes de capturar, com bastante naturalidade e precisão, o conceito de grandezas que variam ao longo de um determinado domínio. É importante ressaltar que isto é válido apenas nos casos em que os sinais correspondem a processos determinísticos; quando a variação da grandeza associada ao sinal ocorre de maneira não determinística, então o modelagem através de *processos estocásticos* é a mais adequada.

O modelo matemático de espaços de sinais é capaz de representar a maior parte dos sinais encontrados no universo físico. Entretanto, alguns sinais importantes não podem ser modelados através de funções, como é o caso dos *impulsos*, os quais precisam ser modelados através de formalismos mais sofisticados, como o das *distribuições*, que por sua vez, são generalizações matemáticas do conceito de função. Apesar desta dificuldade é possível, sem um envolvimento técnico muito rigoroso, aproximar um impulso através do limite de uma seqüência de sinais pulso [46].

Definição 3.4.2 *Um* sinal pulso *pode ser modelado como uma função*

$$p_a(t) = \begin{cases} 0 & t > |a|\,, \\ 1 & t \leq |a| \end{cases} \tag{3.4.1}$$

Assim, podemos definir um sinal impulso, também chamado δ de *Dirac* através do seguinte limite:

$$\delta(t) = \lim_{n\to\infty} \frac{1}{n} p_{1/n}(t) \tag{3.4.2}$$

O sinal impulso tem um papel fundamental na análise e descrição de sinais, pois podemos caracterizar qualquer função $f(t)$ através de uma seqüência de impulsos devidamente modulados e transladados.

$$f(x) = \int_{-\infty}^{+\infty} f(t)\delta(x-t)dt \tag{3.4.3}$$

Os modelos funcionais diferem entre si de acordo com a interpretação associada ao domínio e ao contradomínio das funções utilizadas na modelagem. Os modelos funcionais mais comumente utilizados em processamento de sinais são os modelos *espaciais* e modelos *espectrais*(ou de freqüências).

Nos *modelos funcionais espaciais*, como o próprio nome indica, o domínio representa a região do espaço ou intervalo de tempo em que a grandeza varia, enquanto que o contradomínio representa o conjunto de valores que ela pode assumir. Por exemplo, um modelo funcional para uma imagem na tela de um computador pode ser descrito através de uma função $f : U \subset R^2 \to R^3$, onde o domínio é um subconjunto do plano e o contradomínio é formado por um espaço vetorial tridimensional que caracteriza um espaço de cor formado pelas componentes vermelha, verde e azul.

Os *modelos funcionais espectrais*, apesar de serem um pouco menos intuitivos, estão diretamente relacionados à forma como percebemos a grande maioria dos sinais provenientes do universo físico, já que o sistema sensorial humano percebe sinais, como por exemplo, sons e cores, através de diferentes freqüências. Existem vários modelos espectrais dependendo dos operadores utilizados para detectar e analisar as freqüências dos sinais de interesse. O modelo espectral mais comum é o estabelecido com base na chamada *transformada de Fourier*.

Transformada de Fourier

Definição 3.4.3 *A Transformada de Fourier consiste em um operador unário, $F : S \to S'$, entre dois espaços de sinais, caracterizado por um núcleo periódico $e^{-2\pi iwx}$ com freqüência w, capaz de detectar as freqüências de um sinal não necessariamente periódico.*

$$\hat{f}(w) = (Ff)(x) = \int_{+\infty}^{-\infty} f(x)e^{-2\pi iwx}dx \tag{3.4.4}$$

Podemos entender a transformada de Fourier como um operador que mede a densidade de um determinada freqüência w em um sinal $f(x)$, visto que a integral do produto $f(x)e^{-2\pi iwx}$ retorna uma medida da ressonância entre as componentes exponenciais de $f(x)$, com freqüência w, e o núcleo exponencial da transformada.

A transforma de Fourier é inversível, sendo sua inversa dada por

$$f(x) = (F^{-1}f)(w) = \int_{+\infty}^{-\infty} \hat{f}(w)e^{2\pi iwx}dw, \tag{3.4.5}$$

o que significa que uma sinal pode ser reconstruído através de uma soma infinita de sinais exponenciais periódicos com freqüência w e amplitude $\hat{f}(w)$.

Uma das desvantagens da transformada de Fourier é sua incapacidade de indicar a localização espacial de uma determinada freqüência w existente em um sinal s. Quando isto é necessário precisamos adotar outros operadores como por exemplo, aqueles associados às *transformadas de Fourier com janelas* ou às *transformadas de wavelets* [54].

A Transformada de Fourier aqui apresentada é definida no domínio contínuo, porém sabemos que, na prática, os sinais devem ser representados discretamente para que possam ser processados em sistemas digitais, o que implica que, para analisar sinais desta natureza, precisamos trabalhar com versões discretas desta mesma transformada. Para a exposição que faremos a seguir, a transformada de Fourier contínua é suficiente, bastando para isso, termos em mente que estamos analisando sinais originalmente contínuos. Para maiores detalhes sobre transformadas de Fourier discretas ver [46].

Operações com sinais

Ao trabalharmos com espaços de sinais, obviamente estamos interessados em caracterizar as operações que sobre eles podem ser efetuadas.

Definição 3.4.4 *Uma operação em um espaço de sinais é um mapeamento* $T : R^m \times S^n \to S'$ *em um espaço de sinais* S'*, possivelmente distinto.*

Por definição, um espaço de sinais herda as operações fundamentais de um espaço vetorial: a operação de soma $S \times S \to S$, dada por $(f + g)(t) = f(t) + g(t)$ e a operação de multiplicação por escalar $R \times S \to S$, definida por $(\lambda f)(t) = \lambda f(t)$. Quando os sinais possuem estrutura de multiplicação,

como por exemplo, sinais que assumem valores em R^n ou C^n, podemos definir facilmente a operação de multiplicação de sinais $S \times S \rightarrow S$ como $(fg)(x) = f(x)g(x)$.

Uma operação importantíssima no estudo de sinais é a operação de *filtragem*.

Definição 3.4.5 *Um* filtro *é uma operação unária $L : S \rightarrow S$ que mapeia um sinal $s_0(x)$ em um sinal $s_1(x)$.*

Uma outra interpretação pode ser feita através do conceito de um sistema que recebe como entrada um sinal s_0 e retorna como saída um sinal s_1. Dentre os diversos tipos de filtros podemos destacar dois casos bastante importantes: os *filtros espacialmente invariantes* e os *filtros lineares*.

Definição 3.4.6 *Um filtro L é espacialmente invariante se, para todo sinal $s(x)$, $(Ls)(x-\alpha) = L(s(x-\alpha))$, o que significa que filtrar um sinal e então transladá-lo é equivalente a transladar um sinal e então filtrá-lo.*

Definição 3.4.7 *Um filtro L é linear se, para todo sinal $s(x)$, $L(\lambda s(x)) = \lambda L(s(x))$ e se para quaisquer sinais $s_0(x)$ e $s_1(x)$, $L(s_0+s_1)(x) = L(s_0(x))+ L(s_1(x))$.*

Os *filtros lineares espacialmente invariantes* são extremamente importantes pois podem ser caracterizados através de sua *resposta de impulso* ou *função de espalhamento* $h(t) = L(\delta)$, isto é, através da resposta produzida pelo filtro ao ser aplicado a um sinal impulso. A transformada de Fourier $\hat{h}(t)$ da resposta de impulso de um sinal é conhecida como *função de transferência*.

Como conseqüência, as operações de filtragem de um sinal envolvendo filtros lineares espacialmente invariantes podem ser representadas através da seguinte operação

$$L(f(x)) = \int_{-\infty}^{+\infty} f(t)L(\delta(x-t))dt = \int_{-\infty}^{+\infty} f(t)h(x-t)dt \qquad (3.4.6)$$

A segunda integral define uma operação $S \times S \rightarrow S$ denominada *produto de convolução* ou simplesmente convolução.

$$f * g = \int_{-\infty}^{+\infty} f(t)g(x-t)dt \qquad (3.4.7)$$

No caso dos filtros lineares espacialmente invariantes, a relação entre a operação de multiplicação no domínio espacial e a convolução no domínio da

freqüência é estabelecida pela transformada de Fourier. Esta característica é extremamente útil para que possamos relacionar o comportamento de certas operações nos domínios espacial e espectral.

$$f(x)g(x) \longleftrightarrow \hat{f}(w) * \hat{g}(w) \tag{3.4.8}$$

$$f(x) * g(x) \longleftrightarrow \hat{f}(w)\hat{g}(w) \tag{3.4.9}$$

Uma outra classe de filtros que devemos mencionar é a classe dos *filtros adaptativos*, que desempenham um papel bastante significativo no processo de reamostragem envolvendo mapeamentos genéricos, como por exemplo, os mapeamentos projetivos.

Finalmente, devemos lembrar que os filtros também podem ser classificados segundo a natureza do suporte de sua resposta de impulso. Neste caso, eles podem ser classificados em filtros com *resposta de impulso finita* ou com *resposta de impulso infinita*. Estes últimos precisam ser truncados para que possam ser utilizados em sistemas digitais, o que gera um problema importante, já que os filtros ideais de reconstrução, como veremos a seguir, não possuem resposta de impulso com suporte limitado.

Reconstrução e amostragem de sinais

A maneira mais comum de se efetuar a amostragem de um sinal s é através de uma *amostragem pontual uniforme*. Matematicamente, isto pode ser feito multiplicando-se s por um trem de impulsos

$$i_{\Delta t}(x) = \sum_{k \in Z} \delta(x - k\Delta t), \tag{3.4.10}$$

onde $\delta(t)$ são impulsos unitários, e Δt é o *intervalo de amostragem*. O trem de impulsos normalmente é conhecido como *função pente* ou *sinal de amostragem*. A transformada de Fourier de uma função pente é uma outra função pente

$$i_{1/\Delta t}(x) = \frac{1}{\Delta t} \sum_{k \in Z} \delta(x - \frac{k}{\Delta t}), \tag{3.4.11}$$

o que significa que o intervalo de amostragem de uma função pente no domínio do espectro é igual ao inverso do seu intervalo de amostragem no domínio do espaço (Figura 3.5(b)).

No domínio espectral a amostragem pontual de um sinal s corresponde ao produto de convolução da resposta espectral do sinal $\hat{s}$ com a função de transferência da função pente $\hat{i}(x)$. A convolução tem como efeito a replicação do espectro de freqüências do sinal original, ao longo do domínio do espectro, nas posições correspondentes a cada um dos impulsos que formam a função pente (Figura 3.5(c)).

Conseqüentemente, para reconstruirmos um sinal, precisamos remover as altas freqüências introduzidas pelo processo de amostragem que não correspondem às freqüências originais do sinal, o que pode ser feito através de um processo de filtragem.

A *filtragem ideal* é feita através da multiplicação do espectro de freqüências do sinal amostrado por um *filtro box* devidamente posicionado e modulado, o que no domínio espectral corresponde a um produto de convolução do sinal original com uma função $sinc(x) = \frac{sin(\pi x)}{\pi x}$, que por sua vez, corresponde à transformada de Fourier inversa do filtro box (Figura 3.5(d)).

Uma vez removidas as freqüências espúrias, introduzidas pelo processo de amostragem, podemos reconstruir o sinal original através da transformada de Fourier inversa do sinal filtrado.

Quando a amostragem causa uma mescla das informações de alta freqüência com as informações de baixa freqüência no espectro do sinal amostrado, então não é possível reconstruir um sinal, mesmo através da utilização de um filtro de reconstrução ideal; este fenômeno é o que conhecemos por *aliasing* (Figura 3.6(c)).

De fato, segundo o *teorema de amostragem*, só podemos reconstruir um sinal com exatidão, quando

$$\Delta t \geq \frac{1}{2\omega}, \tag{3.4.12}$$

ou de forma equivalente,

$$\frac{1}{\Delta t} \leq 2\omega, \tag{3.4.13}$$

isto é, quando a freqüência de amostragem é duas vezes maior que a freqüência máxima presente no sinal. Infelizmente, os sinais podem ter freqüências arbitrariamente altas, o que torna inviável e em alguns casos impossível, a amostragem acima do limite de Nyqvist. Nestes casos, o máximo que podemos fazer é tentar reduzir ao máximo os efeitos provocados por aliasing através de técnicas baseadas em filtragem conhecidas como técnicas *anti-aliasing*.

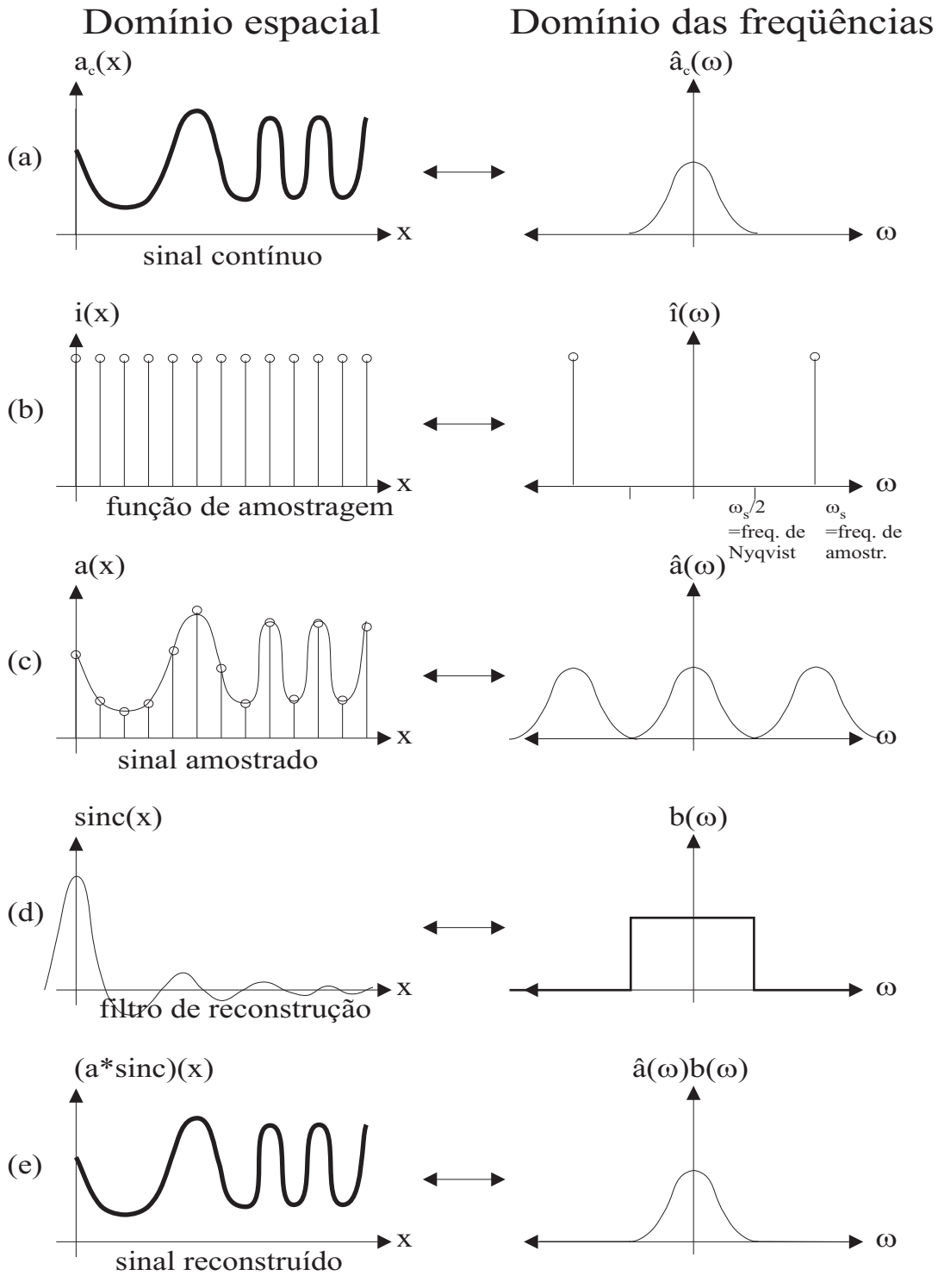


Figura 3.5: Amostragem acima do limite de Nyqvist

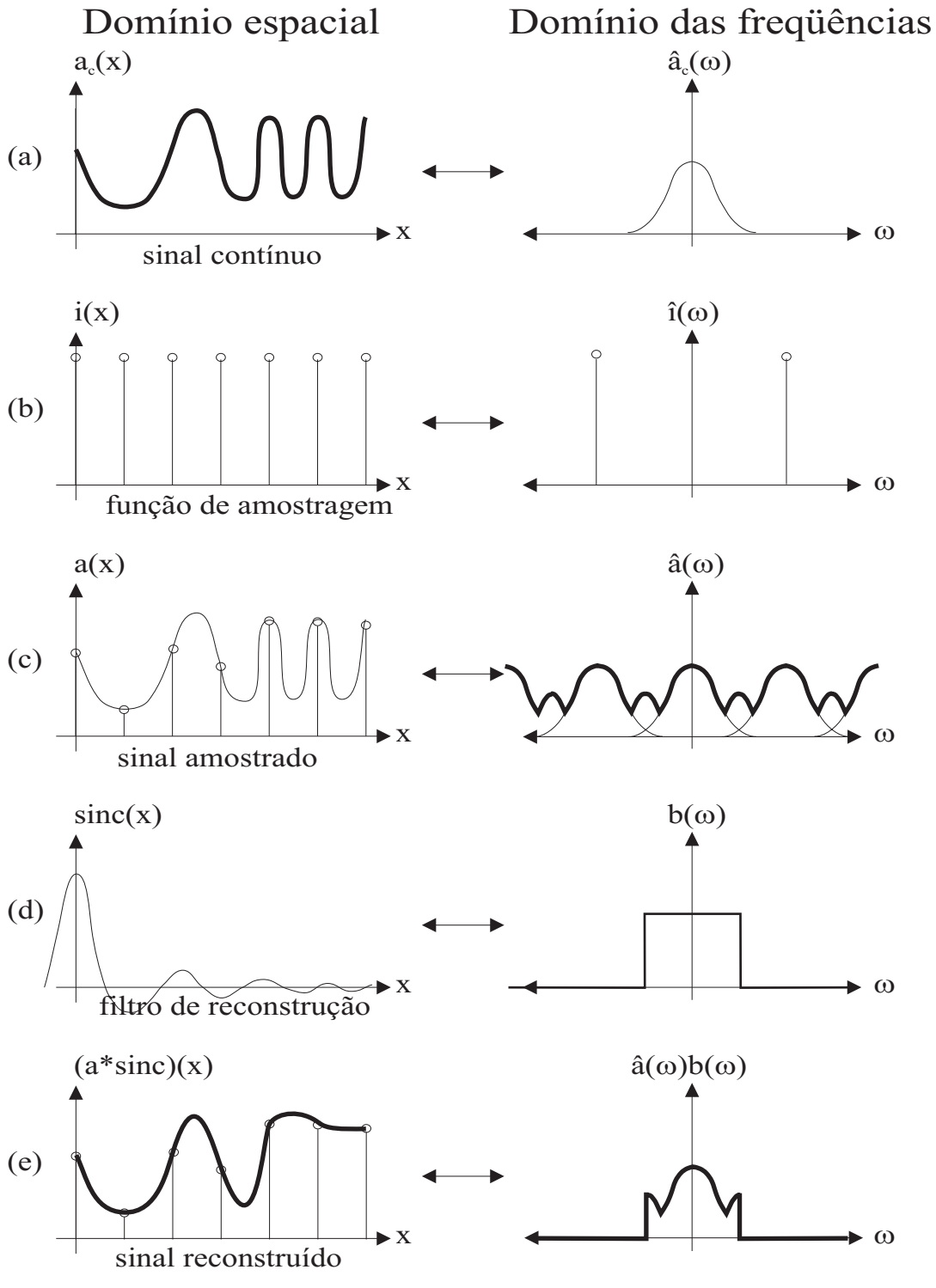


Figura 3.6: Amostragem abaixo do limite de Nyqvist

Técnicas anti-aliasing

Basicamente existem duas formas de se erradicar ou pelo menos reduzir o problema de aliasing: através do *aumento da freqüência de amostragem* ou através de *pré-filtragem* do sinal.

A primeira técnica consiste em aumentar a freqüência de amostragem de tal forma que ela se aproxime do limite de Nyqvist, permitindo assim que o sinal possa ser reconstruído, com o mínimo de aliasing possível, através de interpolação com filtros de reconstrução apropriados (no caso ideal isto significa efetuar uma convolução do sinal com uma função $sinc(x)$).

Ainda que seja capaz de reduzir a presença de aliasing na reconstrução de um sinal, o aumento da freqüência de amostragem por si próprio não é capaz de evitá-lo por completo, já que um sinal pode ter freqüências arbitrariamente altas. Além disso, o aumento excessivo da freqüência de amostragem pode ter conseqüências drásticas tanto em relação ao aumento do espaço em memória para o armazenamento da representação do sinal, quanto ao custo computacional para processá-lo.

A única técnica capaz de evitar aliasing por completo, ao menos em teoria, é a pré-filtragem, a qual consiste em aplicar um *filtro passa baixa* no sinal original com o objetivo de remover as altas freqüências antes de amostrá-lo (Figura 3.7).

Infelizmente, filtrar o sinal no domínio contínuo envolve a solução de integrais que podem ser arbitrariamente complexas dependendo do tipo de função associada, além de que, em alguns casos, o filtro utilizado pode não ser espacialmente invariante.

Por estes motivos, muitas das vezes, na prática, opta-se por resolver o problema de aliasing, através de *pós-filtragem*, na qual a filtragem ocorre após uma superamostragem do sinal.

Muitas das vezes, o sinal com o qual trabalhamos já é um sinal amostrado que, por algum motivo, precisa sofrer uma nova operação de amostragem. Este tipo de procedimento é muito comum em processamento digitais de sinais, estando diretamente relacionado aos procedimentos de mapeamento de texturas. Por esta razão, iremos investigá-lo em maiores detalhes na próxima seção.

Reamostragem de sinais

A *reamostragem* é o processo pelo qual um sinal contínuo é reconstruído a partir de um sinal discreto, podendo eventualmente sofrer uma transformação de um espaço de origem para um espaço destino, antes

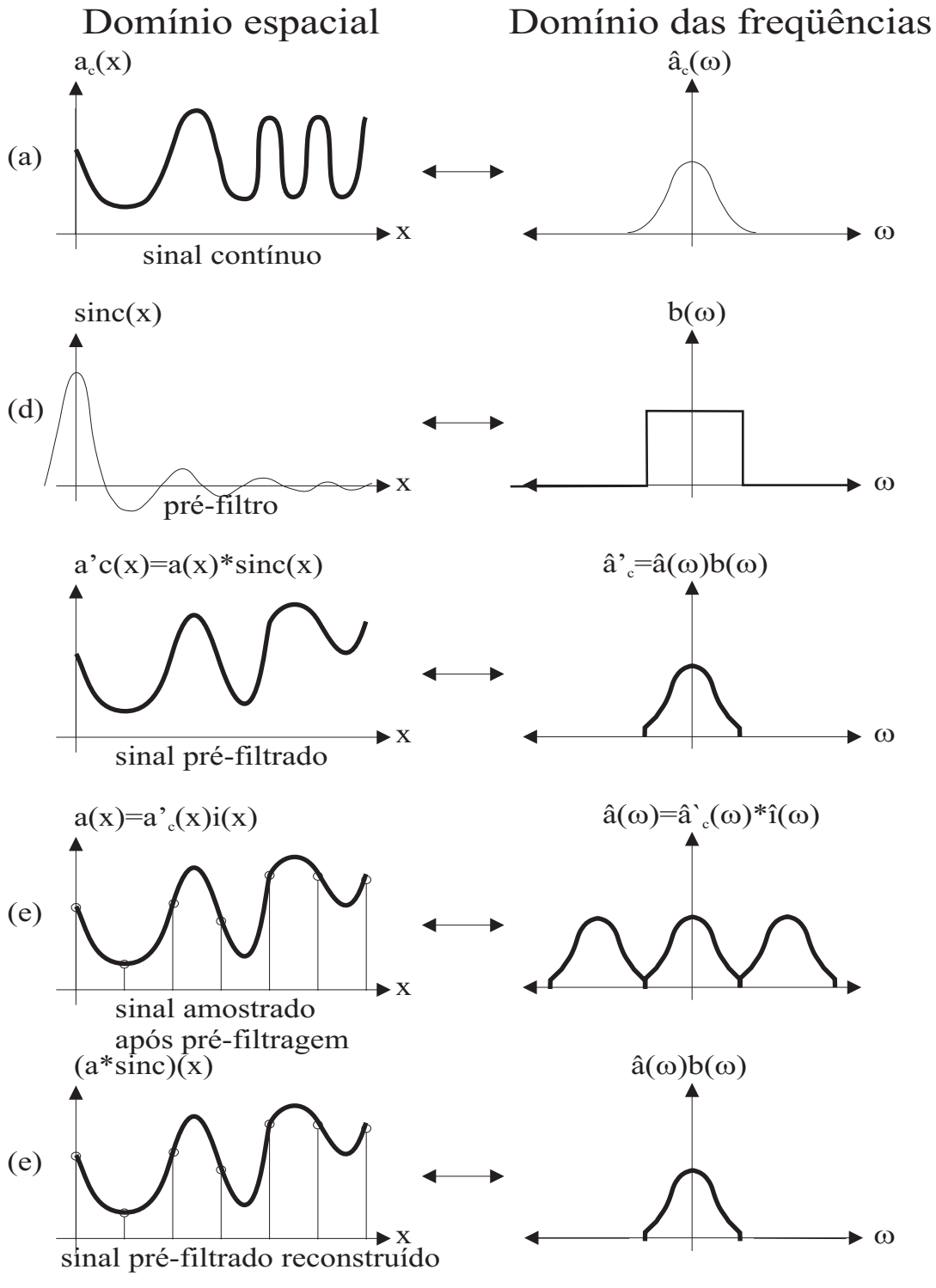


Figura 3.7: Pré-filtragem

de ser novamente amostrado, na maioria das vezes, com uma taxa de amostragem diferente da taxa de amostragem original. Um exemplo típico de reamostragem ocorre quando mapeamos uma textura em um polígono definido no espaço da tela.

O procedimento de reamostragem normalmente segue o seguinte esquema:

1. Reconstrução do sinal contínuo a partir do sinal discreto original.
2. Mapeamento do sinal reconstruído para o espaço de destino.
3. Pré-filtragem.
4. Amostragem.

Quando o mapeamento do espaço de origem para o espaço de destino é de natureza afim, então a teoria sobre *amostragem pontual uniforme*, apresentada anteriormente pode ser utilizada sem maiores dificuldades, já que nestes casos, a filtragem pode ser caracterizada através de *filtros lineares espacialmente invariantes*. Por outro lado, quando trabalhamos com mapeamentos mais genéricos, como por exemplo, mapeamentos projetivos, então os filtros em geral não são espacialmente invariantes, o que torna o processo um pouco mais sofisticado (Figura 3.8).

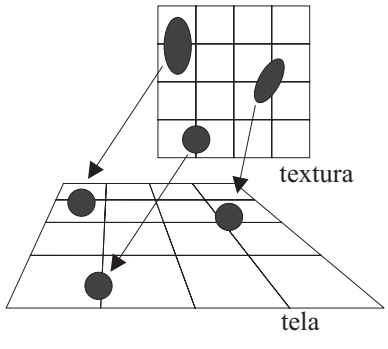


Figura 3.8: Mapeamentos não-afins

Seja $f(\mathbf{u})$ um sinal discreto definido em espaço de origem com coordenadas $\mathbf{u} = (u, v)$, o qual deve sofrer um processo de reamostragem para que possa ser mapeado em um sinal $g(\mathbf{x})$, também discreto, definido em um espaço de destino com coordenadas $\mathbf{x} = (x, y)$. O espaço de origem e o espaço de destino estão relacionados entre si através de um mapeamento direto $\mathbf{x} = m(\mathbf{u})$ e de um mapeamento inverso $\mathbf{u} = m^{-1}(\mathbf{x})$. Seja ainda $r(\mathbf{u})$ um filtro de reconstrução e $h(\mathbf{x})$ um pré-filtro a ser utilizado no processo

sinal discreto original	$f(\mathbf{u})$
reconstrução	$f_c(\mathbf{u}) = f(\mathbf{u}) * r(\mathbf{u}) = \sum_{\mathbf{k}\in Z^n} f(\mathbf{k}) r(\mathbf{u}-\mathbf{k})$
mapeamento	$g_c(\mathbf{x}) = f_c(m^{-1}(\mathbf{x}))$
pré-filtragem	$g'_c = g_c(\mathbf{x}) * h(\mathbf{x}) = \int_{R^n} g_c(\mathbf{t}) h(\mathbf{x}-\mathbf{t}) d\mathbf{t}$
amostragem final	$g(\mathbf{x}) = g'_c(\mathbf{x}) i(\mathbf{x})$

de reamostragem. Então, $g(\mathbf{x})$ pode ser obtido a partir de $f(u)$ através dos seguintes passos:

Expandindo a equação final $g(x)$ e reorganizando as operações de forma conveniente, podemos reduzir o conjunto de passos anterior a uma operação envolvendo o sinal original discreto $f(\mathbf{u})$ e um único filtro $\rho(\mathbf{x}, \mathbf{k})$, denominado *filtro de reamostragem*.

$$\begin{aligned} g(\mathbf{x}) &= g'_c(\mathbf{x}) \\ &= \int_{R^n} f_c(m^{-1}(\mathbf{t})) h(\mathbf{x}-\mathbf{t}) d\mathbf{t} \\ &= \int_{R^n} h(\mathbf{x}-\mathbf{t}) \sum_{\mathbf{k}\in Z^n} f(\mathbf{k}) r(m^{-1}(\mathbf{t})-\mathbf{k}) d\mathbf{t} \\ &= \sum_{\mathbf{k}\in Z^n} f(\mathbf{k}) \rho(\mathbf{x}, \mathbf{k}) \end{aligned}$$

$$onde : \rho(\mathbf{x}, \mathbf{k}) = \int_{R^n} h(\mathbf{x}-\mathbf{t}) r(m^{-1}(\mathbf{t})-\mathbf{k}) d\mathbf{t} \qquad (3.4.14)$$

O filtro de reamostragem $\rho(\mathbf{x}, \mathbf{k})$ varia espacialmente, determinando o peso de uma amostra na posição $\mathbf{k}$, no espaço de origem, correspondente a uma posição $\mathbf{x}$, no espaço de destino. Na equação anterior, $\rho(\mathbf{x}, \mathbf{k})$ é definido no espaço de destino, como uma integral do produto do pré-filtro com o filtro de reconstrução transformado para o espaço de destino. Porém, através de uma troca de variáveis, fazendo $\mathbf{t} = m(\mathbf{u})$, podemos defini-lo no espaço de origem como uma integral do produto do pré-filtro transformado para o espaço de origem com o filtro de reconstrução.

$$\rho(\mathbf{x}, \mathbf{k}) = \int_{R^n} h(\mathbf{x}-m(\mathbf{u})) r(\mathbf{u}-\mathbf{k}) \left| \frac{\partial m}{\partial \mathbf{u}} \right| d\mathbf{u} \qquad (3.4.15)$$

Como podemos ver através da equação 3.4.15, a reamostragem de sinais envolvendo mapeamentos não-afins pode ser relativamente complexa, o que significa que na prática precisamos efetuar algumas simplificações tanto no mapeamento do espaço de origem para o espaço de destino, quanto nos filtros de reconstrução.

O mapeamento projetivo $m(\mathbf{u})$ pode ser aproximado em uma vizinhança local $\mathbf{u_0}$ através de um mapeamento afim

$$m_{\mathbf{u}_0}(\mathbf{u}) = \mathbf{x}_0 + (\mathbf{u} - \mathbf{u}_0)J_{\mathbf{u}_0}, \tag{3.4.16}$$

onde

$$J_{\mathbf{u}_0} = \frac{\partial m}{\partial \mathbf{u}}(\mathbf{u}_0) \tag{3.4.17}$$

é a matriz jacobiana em $\mathbf{u_0}$.

Como o pré-filtro $h(\mathbf{x} - m^{-1}(\mathbf{x}))$ atribui um peso maior às regiões mais próximas a $\mathbf{u}_0$, onde a aproximação é mais precisa, e um peso menor às regiões mais distantes, onde a aproximação não é tão precisa, podemos utilizar a aproximação linear para o mapeamento sem causar maiores danos ao filtro de reamostragem.

Com efeito, o filtro de reamostragem pode ser expresso como

$$\rho_{\mathbf{u}_0}(\mathbf{x}_0, \mathbf{k}) = \int h(\mathbf{x} - m_{\mathbf{u}_0}(\mathbf{u}))r(\mathbf{u} - \mathbf{k})|J_{\mathbf{u}_0}|d\mathbf{u}. \tag{3.4.18}$$

Podemos também aproximar ou os filtros de reconstrução, ou os pré-filtros, através de impulsos, conforme o efeito causado pelo mapeamento envolvido no processo de reamostragem.

Quando o mapeamento causa uma ampliação da imagem no espaço de destino, então a forma e tamanho do filtro de reamostragem são dominados pelo filtro de reconstrução. Neste caso, podemos aproximar o pré-filtro através de um impulso, isto é, fazendo $h = \delta$. Desta forma, o filtro de reamostragem se reduz a

$$\rho(\mathbf{x}, \mathbf{k}) \approx \left|\frac{\partial m}{\partial \mathbf{u}}\right| r(m^{-1}(x) - \mathbf{k}) \tag{3.4.19}$$

o qual é um filtro de reconstrução não transformado, centrado em $m^{-1}(\mathbf{x})$, com um suporte independente do mapeamento e de tamanho em torno de 1 a 4 pixels. Estes filtros são conhecidos em processamento de imagens como *filtros de magnificação* ou *filtros de interpolação*, sendo exemplos clássicos os filtros box, os filtros bilineares e os filtros definidos por b-splines cúbicas.

Por outro lado, quando o mapeamento causa uma redução da imagem no espaço de destino, então a forma e tamanho do filtro de reamostragem são dominados pelo pré-filtro, o que nos permite aproximar o filtro de reconstrução por um impulso, isto é, fazendo $r = \delta$. Neste caso, o filtro

de reamostragem pode ser expresso da seguinte forma:

$$\rho(\mathbf{x}, \mathbf{k}) \approx \left| \frac{\partial m}{\partial \mathbf{u}} \right| h(\mathbf{x} - m(\mathbf{k})) \tag{3.4.20}$$

o qual é um pré-filtro centrado em $\mathbf{m}^{-1}(\mathbf{x})$, devidamente transformado segundo o mapeamento m. Estes filtros são conhecidos como *filtros de redução* ou *filtros de decimação*. O suporte deste tipo de filtro tem área inversamente proporcional ao fator de escala do mapeamento; como os fatores de escala podem assumir valores arbitrários, então o número de pixels no suporte pode variar de um a até alguns milhões.

As implementações de mapeamentos de textura normalmente determinam que sejam especificados filtros, possivelmente distintos, para os casos de magnificação e de redução, os quais são selecionados durante o processo de reamostragem de acordo com o fator de escala determinado pelo mapeamento.

Este tipo de procedimento pode causar alguns problemas em duas dimensões já que neste caso o fator de escala não é dado por um único escalar e sim por uma matriz 2×2, a matriz jacobiana da transformação. Por conseguinte, o mapeamento pode causar a ampliação em uma direção e uma redução em outra, o que em geral é resolvido, de maneira *ad hoc*, através da seguinte estratégia: utiliza-se o filtro de magnificação quando ambas as direções são ampliadas, caso alguma das direções seja reduzida, utiliza-se um filtro de redução. Quando uma das direções sofre uma redução e a outra sofre uma magnificação, a forma do filtro de reamostragem resultante pode ficar tão estreita que alguns pixels podem ser perdidos. Neste caso, costuma-se obrigar o filtro a amostrar ao menos um elemento.

Heckbert propôs a solução deste problema através da utilização de *filtros gaussianos elípticos* (Figura 3.9), os quais são capazes de fornecer um tratamento unificado para a operação de reamostragem pois conseguem transitar suavemente entre os casos de magnificação e redução [26]. Infelizmente, a maioria das implementações em placas gráficas não adota filtros tão sofisticados, pois a utilização destes, além de dificultar o projeto do hardware, implicaria em um custo computacional bastante alto, inviável para processamento em tempo real. A maioria dos métodos de filtragem disponíveis em placas gráficas ainda é baseada em filtros mais simples, como os filtros box e os filtros bilineares, juntamente com o uso de heurísticas para a escolha do tipo de filtragem utilizada, isto é, de redução ou de ampliação.

Em nossa implementação utilizamos os mecanismos de filtragem disponíveis nos procedimentos de mapeamento de textura fornecidos pela

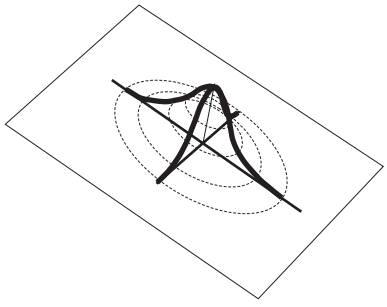


Figura 3.9: Filtros gaussianos elípticos

OpenGL, a qual determina que sejam especificados filtros para os casos de filtragem de magnificação e de redução. Para cada um destes casos é possível efetuar uma escolha entre dois tipos básicos de filtragem: um filtro baseado em vizinhança mais próxima e um filtro linear.

O filtro baseado em vizinhança mais próxima, na verdade, um filtro box, retorna o elemento de textura cujas coordenadas se encontram mais próximas das coordenadas de textura associadas a um determinado elemento da tela. Os resultados produzidos pela aplicação deste tipo de filtro são extremamente suscetíveis a aliasing, o que nos impede de utilizá-lo nos procedimentos de mapeamento projetivo de texturas necessários ao registro das imagens de entrada no espaço da cena.

O filtro linear, definido por uma máscara 2×2, efetua uma interpolação bilinear entre os elementos de textura mais próximos das coordenadas de textura correspondentes ao pixel desejado. Este filtro pode ser utilizado sem grandes problemas quando a imagem associada à textura é ampliada, porém, quando a imagem é reduzida, então os resultados provenientes de sua aplicação não são muito melhores que os obtidos através da aplicação de filtros box, sendo também bastante suscetíveis aos efeitos causados por aliasing. Isto ocorre porque os suportes dos filtros de redução devem variar de acordo com o fator de escala, não podendo ser especificados através de suportes de tamanho fixo, com por exemplo um suporte definido por uma máscara 2×2.

Um possível melhoramento pode ser obtido através do uso de *mipmapping*, o qual é um método de anti-aliasing que possibilita a aproximação da operação de filtragem, sobre um conjunto de texels que são mapeados em um determinado pixel, através de uma única operação de amostragem efetuada sobre uma pirâmide de mapas de textura pré-filtrados.

A estrutura piramidal utilizada pelo processo de mipmapping é formada por uma seqüência de n imagens em resolução decrescente, cada uma

Figura 3.10: Mipmap

delas determinando um nível de detalhe para o mapa de textura original (Figura 3.10). Cada um dos n níveis de detalhe são especificados da seguinte forma: o nível 0, que corresponde à base da pirâmide, é determinado pelo mapa de textura original; um nível i, tal que $0 < i < n$ é obtido através de um subamostragem do nível $i - 1$, onde uma combinação de texels vizinhos do nível $i - 1$ contribui para a geração de um único texel no nível i(normalmente esta combinação é dada por uma média de quatro texels vizinhos); o último nível, isto é, o nível n, corresponde ao mapa de textura no qual uma de suas dimensões possui apenas um elemento, obtido através da subamostragem do mapa de textura no nível $n - 1$.

O mecanismo subjacente ao processo de mipmapping procura determinar o nível de detalhe λ mais adequado para uma determinada operação de reamostragem, através de uma análise da região do mapa de textura que contribui para um determinado pixel no espaço da imagem. Um maneira simples de aproximar tal contribuição pode ser feita através da medida da maior aresta pertencente ao quadrilátero correspondente à projeção de um pixel no mapa de textura em questão. Uma outra estratégia se baseia no valor absoluto máximo entre as diferenciais $\partial u/\partial x$, $\partial v/\partial x$, $\partial w/\partial x$,$\partial u/\partial y$, $\partial v/\partial y$, $\partial w/\partial y$, que refletem as taxas de variação instantâneas das coordenadas de textura em função das variações instantâneas das coordenadas x e y no espaço da tela.

O valor de λ é utilizado para determinar o nível no qual deve ser realizada a amostragem na estrutura piramidal. O nível de detalhe mais adequado é aquele que possibilita uma taxa de amostragem mais próxima de um pixel para cada amostra, ou preferencialmente, de dois pixels para uma amostra, satisfazendo assim o limite de Nyqvist. Quando isto ocorre, a reamostragem é realizada de forma bastante eficiente ao mesmo tempo em

que os efeitos de aliasing são adequadamente atenuados.

A amostra desejada é especificada por uma tripla (u, v, λ). No entanto, como λ é um valor fracionário, precisamos efetuar alguns cálculos para determinar o nível apropriado na estrutura piramidal. A solução adotada consiste em amostrar os texels nos níveis de detalhe, acima e abaixo de λ, determinados por interpolações bilineares em torno das posições determinadas respectivamente por $(u, v, \lfloor\lambda\rfloor)$ e $(u, v, \lceil\lambda\rceil)$. A amostra resultante é então calculada através de uma interpolação linear entre as duas amostras obtidas, com base na distância de cada nível de textura a λ.

O processo de mipmapping tem a grande vantagem de realizar a amostragem em tempo constante, independente do grau de redução causado pelo mapeamento. No entanto, também apresenta alguns problemas, em especial, o excesso de suavização que pode ocorrer em determinadas regiões.

A causa deste problema está no fato de que, no mecanismo de mipmapping, somos obrigados a aproximar a área relativa à contribuição do mapa de textura para um determinado pixel através de um região quadrada. Esta aproximação normalmente não é adequada quando a área de contribuição real possui mais elementos de textura em uma direção do que na outra, o que é bastante comum quando o polígono texturizado é quase ortogonal ao plano de projeção, causando um borramento excessivo de certas características presentes no mapa de textura.

Existem alguns métodos que procuram evitar o fenômeno de suavização excessiva como por exemplo o *Ripmapping* [21, 30] e o método das *Tabelas de Áreas Somadas* (*Summed-area Tables*) [13]. Estes métodos são classificados como métodos de filtragem anisotrópicos, sendo capazes de recuperar valores de textura definidos em áreas não quadradas, o que os permite trabalhar com taxas de amostragem diferentes em direções distintas. A grande desvantagem associada a estes métodos é que eles só conseguem efetuar filtragens anisotrópicas, de forma eficiente, em direções preferencialmente horizontais e verticais. Uma outra característica inconveniente é a de que estes métodos requerem um espaço de memória bem maior que o necessário para o armazenamento da estrutura piramidal utilizada em mipmapping.

Um método mais robusto e, na verdade, mais eficiente também, é o *método de filtragem anisotrópica irrestrita* que trabalha sobre a própria estrutura utilizada em *mipmapping*. A diferença chave é a de que utilizamos várias amostras da estrutura do mipmap, ao invés de somente uma, para aproximar a área de contribuição do mapa de textura para um dado pixel, a qual corresponde à cobertura de tal pixel no mapa de textura. O valor

de λ, neste tipo de filtragem é dado pelo menor lado do quadrilátero associado à área de cobertura de um pixel no mapa de textura, o que faz com a região filtrada seja menor, causando, conseqüentemente, um menor borramento. O maior lado do quadrilátero determina o que chamamos *linha de anisotropia*, a qual passa pelo interior da região a ele associada, podendo assumir qualquer direção, sendo este o motivo pelo qual esse esquema de filtragem é denominado irrestrito. Um fator de anisotropia 2:1 significa que duas amostras são tomadas ao longo da direção por ela determinada. Já se encontram disponíveis filtragens com fator de anisotropia 8:1, como por exemplo, a que podemos encontrar nas GeForce 3 [2]. Em nosso trabalho não investigamos a utilização de filtragem anisotrópica.

Devido às limitações associadas à utilização de filtros de reamostragem bastante aquém dos filtros de reamostragem ideais, adicionamos um procedimento de *superamostragem* às técnicas de mipmapping e filtragem linear, com o objetivo de amenizar um pouco mais os efeitos causados por aliasing.

No próximo capítulo iremos aplicar as técnicas aqui descritas na construção de um algoritmo de escultura do espaço que trabalha diretamente no espaço da cena, em contraposição aos algoritmos de escultura do espaço tradicionais, que trabalham no espaço das imagens.