2 Renderização Baseada em Imagens

What is real? How do you define real? If you're talking about what you can feel, what you can smell, what you can taste and see, then real is simply electrical signals interpreted by your brain. (The Matrix)

2.1 Introdução

O trabalho apresentado nesta pesquisa pertence, em parte, à área da computação gráfica denominada de renderização baseada em imagens. Esta área tem como objetivo representar um cenário completo ou objetos que façam parte dele a partir de imagens, ao invés de elementos geométricos. Neste capítulo descreve-se inicialmente o conceito de função plenóptica, juntamente com os trabalhos mais significativos na área. Em seguida discute-se, resumidamente, um sistema para classificação e comparação dos métodos existentes. Finalmente, expõe-se com detalhes o *3D Image Warping*, que é fundamental para desenvolver o conceito dos impostores com relevo.

2.2 A Função Plenóptica

A área de renderização baseada em imagens surgiu a partir de uma combinação entre as técnicas de Computação Gráfica 3D e Visão Computacional, tendo em vista adquirir resultados foto-realistas e ao mesmo tempo rápidos na visualização de cenas. Atualmente, com o avanço do poder computacional das placas gráficas, esta técnica ganhou grande importância em sistemas de visualização de tempo real, tais como jogos, aplicações de realidade virtual e simulações.

Para uma definição mais formal do que vem a ser o *ibr* – abreviatura comumente utilizada para referir-se a renderização baseada em imagens, extraída da expressão em inglês *image-based rendering* -, deve-se recorrer ao conceito de

função plenóptica[1], definida inicialmente por Adelson (1991). Esta é uma função parametrizada que descreve tudo o que é possível de ser visto a partir de qualquer posição e orientação do espaço, a qualquer momento e em qualquer comprimento de onda da luz. Em (McMillan, 1995) a função é definida como:

$$cor = \mathrm{P}\,(\theta, \phi, \lambda, V_x, V_y, V_z, t) \tag{2-1}$$

onde (V_x, V_y, V_z) é a posição do observador, θ e ϕ são os ângulos de azimute e elevação da direção do vetor do observador (ver figura 2.1) e λ é o comprimento da onde de luz capaz de ser vista. No caso de uma cena dinâmica, a variável *t* descreve o tempo.

Pode-se entender a função plenóptica como sendo uma função capaz de descrever todas as possíveis vistas de uma determinada cena. Diz-se que esta função está completa se, para um ponto coberto pela função, toda a abóboda que está à sua volta pode ser vista; diz-se incompleta quando apenas uma parte desta abóbada é visível. Vários autores afirmam que todos os algoritmos de *ibr* se resumem a uma abordagem particular da função plenóptica.

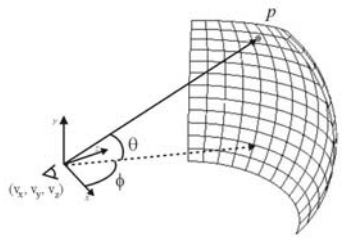


figura 2.1 – Qualquer ponto visível para qualquer posição de um observador pode ser descrito pela função plenóptica.

Desta maneira, um possível enunciado para a maioria dos problemas relacionados a *ibr* pode ser descrito da seguinte forma: “dado um conjunto de amostras (completas ou incompletas) da função plenóptica, o objetivo da solução proposta consiste em gerar uma representação contínua desta função” (McMillan, 1995).

A seguir descrevem-se os trabalhos de maior destaque na área de *ibr*,

[1] *plenus* = completa e *optic* = visão

classificados de acordo com o que se propõem a representar: cenários completos (tudo o que é visível é modelado por imagens), panoramas (objetos que estão em volta e infinitamente afastados do observador são imagens), transformação de elementos da cena em imagens com o objetivo de realizar *culling* ou otimização da visualização e objetos específicos (apenas alguns elementos da cena são descritos por imagens).

2.2.1 Modelagem de cenários completos

Os trabalhos mais próximos da definição de função plenóptica e que procuram representar um cenário completo são o *Ligth Field* (Levoy, 1996) e o *Lumigraph* (Gortler, 1996), embora antes destes Lippman (1980) tenha desenvolvido o *MovieMap*, que pode ser considerado como o precursor de todas as implementações de *ibr* e da definição da função plenóptica. O *Ligth Field* e o *Lumigraph* poderiam também ser chamados de técnicas “força bruta”, no sentido de que criam um banco de dados com todos os possíveis raios de visão existentes dentro de uma região. Para tanto, estas técnicas sugerem descrever cada raio de luz através das duas coordenadas de interseção do mesmo com dois planos de suporte distintos. Desta maneira, é possível descrever um sub-conjunto da função plenóptica através de dois pares de coordenadas bidimensionais. Deve-se ressaltar que apenas 2 planos não são suficientes para representar todos os raios do interior de uma cena, sendo necessário portanto definir vários pares de planos distintos e não paralelos. Não é difícil concluir que ambos requerem uma densidade de amostras enorme para poder haver um mínimo de continuidade nos movimentos.

O *Unstructural Lumigraph Rendering* (Buheler, 2001) tem uma idéia semelhante aos dois trabalhos citados, porém não precisa de uma seqüência de imagens de entrada com uma ordem pré-estabelecida, como ocorre no *Lumigraph* ou no *Light Field*, embora ambos possuam soluções de pré-processamento que solucionam, em parte, este problema. Isto possibilita que a modelagem da cena seja mais simples, bastando capturar imagens do local através de aparelhos convencionais. Entretanto, o sistema requer que seja feito uma estimativa da posição da câmera, bem como uma aproximação geométrica da cena. Quanto melhor é esta aproximação, menos imagens são necessárias. O algoritmo baseia-se no conceito de *camera blending field*, que consiste numa tabela de distribuição de

pesos para cada ponto visto pela câmera, indicando qual é a imagem que deve ser usada para gerá-lo e quanto é a sua contribuição (um mesmo ponto é descrito por várias imagens, desde que a soma de todos os pesos seja igual a 1). Para construir esta distribuição de pesos, realiza-se o cálculo das imagens que possuem o menor ângulo formado entre o seu centro de projeção e o ponto que está sendo observado com a posição real da câmera com o mesmo ponto. Embora a aproximação da geometria das imagens possa ser um processo complexo, e não ser conveniente para representar ambientes extensos, o algoritmo demonstra ter bom comportamento quando se utilizam aproximações grosseiras da geometria, como por exemplo criando uma malha de polígonos coincidente com o plano de projeção da câmera e com alguns poucos vértices deslocados. À aproximação geométrica é possível somar-se a geometria de outros objetos, possibilitando que haja uma mistura de *ibr* com modelagem tradicional.

Procurar extrair informações geométricas da cena tem sido uma abordagem muito utilizada para reduzir o número de imagens necessárias para modelar a função plenóptica. Em (Debvec, 1996) propõe-se que a partir de um conjunto de imagens arquitetônicas de um ambiente real, seja gerado um modelo geométrico aproximado e a seguir se aplique as imagens sobre os modelos, com técnicas tradicionais de projeção de texturas. O sistema auxilia o usuário, detectando linhas horizontais e verticais para posteriormente indicar onde colocar primitivas geométricas, cujos vértices possuirão correspondência com coordenadas de texturas referentes às imagens fonte. Esta técnica introduz o conceito de view-dependent texture mapping e prevê ainda que detalhes da modelagem sejam estimados utilizando métodos de visão estéreo para calcular um mapa de profundidade de cada imagem original.

Apesar desta solução possuir um grau de liberdade alto (5 dimensões: θ, ϕ, V_x, V_y e V_z) o espaço de navegação é pequeno e restrito. Quanto mais se deseje ampliar esta região, maior será o trabalho de modelagem, tornando-se em alguns casos impraticável. Além disso, sua ferramenta de autoria prevê que os objetos das imagens tenham um estilo arquitetônico (linhas retas), o que limita a utilização do sistema para cenas particulares. Borshukov (1997) estendeu o sistema para algumas primitivas curvas.

2.2.2 Modelagem de panoramas

Uma forma mais simples e mais genérica para representar cenas com um suporte geométrico consiste na técnica de panoramas, inicialmente proposto por Chen (1995). Esta técnica, que se tornou conhecida devido à sua aplicação comercial *Quicktime VR*, utiliza a idéia básica de *enviroment-maps*[*].

O *QuickTime VR* utiliza um panorama cilíndrico como suporte geométrico, tendo em vista a facilidade no processo de autoria (gerar a biblioteca de imagens que serão necessárias). No espaço da função plenóptica este método é classificado como bidimensional, pois o movimento da câmera não é contínuo e está restrito a um conjunto finito de pontos do espaço, sendo as únicas variáveis que podem variar continuamente θ e ϕ - ângulos de azimute e elevação do observador. No caso do panorama ser cilíndrico o ângulo de elevação está restrito. Isto é resolvido criando-se suportes cúbicos ou esféricos, do panorama.

Esta técnica vem sendo amplamente utilizada em jogos, e é conhecida também como *skyboxes*. Neste caso, ao invés de um suporte cilíndrico utiliza-se um cubo, onde o observador está sempre no seu centro. Através deste método, representam-se objetos afastados e sem interação com o jogador, como o céu, paisagens de fundo, estrelas, etc.

Na tentativa de se poder realizar mudanças contínuas entre diversas imagens panorâmicas, um dos trabalhos de mais destaque é o proposto em (Aliaga, 2001), chamado de *Plenoptic Stitching*.

O método consiste inicialmente em capturar imagens de um ambiente com um pequeno carro rádio-controlado que percorre caminhos dentro de um determinado ambiente. Para o bom funcionamento do sistema, de acordo com os autores, são necessárias em torno de 30 imagens por metro, utilizando uma câmera omnidirecional (capaz de adquirir fotos de 360^o de latitude por 180^o de longitude). Estes caminhos percorridos devem cruzar-se, de maneira a formar uma malha irregular, sendo cada célula da malha denominada de ciclo de imagem. Pode-se entender estes ciclos como diversos panoramas justapostos, não havendo necessidade de uma forma pré-definida para cada um. Dada a grande quantidade de imagens necessárias, o sistema foi previsto apenas para ambientes internos.

[*] Imagens mapeadas sobre uma esfera que engloba completamente o objeto (Blinn, 76)

Feita a captura dos panoramas, o sistema calcula uma aproximação da posição da câmera para cada imagem. Sugere-se para isto o algoritmo chamado de *bacon-based*, capaz de estimar o movimento feito pela câmera partindo de uma imagem inicial. De maneira a realizar esta estimativa com mais precisão, os autores sugerem que se coloquem duas lâmpadas no ambiente, indicando-se suas posições na imagem inicial.

A interpolação de imagens será necessária sempre que o observador estiver no interior de uma célula. Neste caso, a composição é feita utilizando-se colunas de pixels das imagens que estiverem nas bordas dos ciclos de imagens mais próximos da câmera.

Para que o sistema possa ser tempo real, é necessário haver uma fase de pré-processamento, onde sobretudo se cria uma estrutura de dados para que se possa ter rápido acesso às linhas radiais e às suas funções de mapeamento requeridas na reconstrução.

2.2.3 Aplicações de *ibr* para *cache* e *culling*

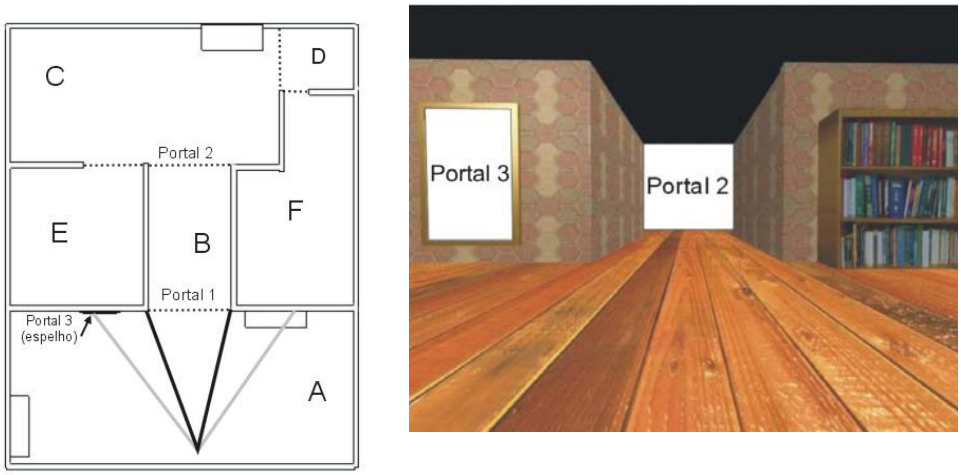
Algoritmos de *ibr* também podem ser utilizados para otimização do processo de visualização de uma cena. Enquadram-se dentro desta área, os algoritmos de portais e impostores.

Os impostores são discutidos com mais detalhes na seção 3.3, mas visam sobretudo minimizar o número de cálculos de visualização de um objeto, reaproveitando resultados já obtidos anteriormente. Neste sentido é que funcionam como um *cache*.

Os portais (Airey 1990, Teller 1991, Eberly 2000) possuem um conceito que em parte é semelhante ao dos impostores, porém para regiões visíveis de uma cena ao invés de objetos individuais. São comumente utilizados como um método eficiente para *culling*[*] de ambientes que podem ser divididos em células. Este método procura otimizar a visualização, fazendo com que apenas as células visíveis num determinado instante sejam processadas. Aliaga (1997) propõem que esta idéia seja usada para substituir parte da geometria de uma cena por um

[*] Processo de eliminação de polígonos não necessários para a visualização da imagem correspondente.

polígono especial, que é o portal, onde se mapeia a imagem resultante da visualização de uma câmera posicionada no mesmo local do observador inicial, mas com um campo de visão menor que o original e limitado pelo próprio polígono corretamente recortado (figura 2.2).



(a) (b)

figura 2.2 – Em (a) pode-se ver um mapa de um ambiente fechado. Cada célula está indicada por uma letra maiúscula e os portais estão representados com uma linha tracejada. Em (b) pode-se ver o resultado parcial da visualização da cena: o portal 1 já está sendo mostrado e os portais 2 e 3 ainda estão em branco.

O algoritmo de portais inicia-se dividindo a cena em células convexas, o que garante a propriedade de que um observador posicionado em qualquer ponto do interior da célula pode ver qualquer outra parte da mesma (não se está considerando a existência de objetos oclusivos não pertencentes à estrutura). Estando-se dentro de uma célula, apenas se podem ver outras através dos polígonos especiais, que são os portais (figura 2.2). Assim, diz-se que todas as células estão separadas por polígonos normais (o que impede que o observador veja a célula adjacente) ou por portais. Para a representação de células não convexas, criam-se paredes invisíveis, onde os portais correspondem a estas paredes.

O *pipeline* de renderização que utiliza os portais deve garantir que os polígonos sejam renderizados na ordem de traz para frente, de maneira a garantir a ordem correta de polígonos visíveis. A visualização pode ser resumida pelo seguinte algoritmo:

Fazer o Clipping e Culling da célula onde se encontra o observador
Visualizar a cena utilizando o observador na sua posição original
Se um polígono da cena é um portal então
Recalcular o campo de visão do observador
Visualizar a sala que está sendo vista pelo portal
Chamar recursivamente o algoritmo de visualização

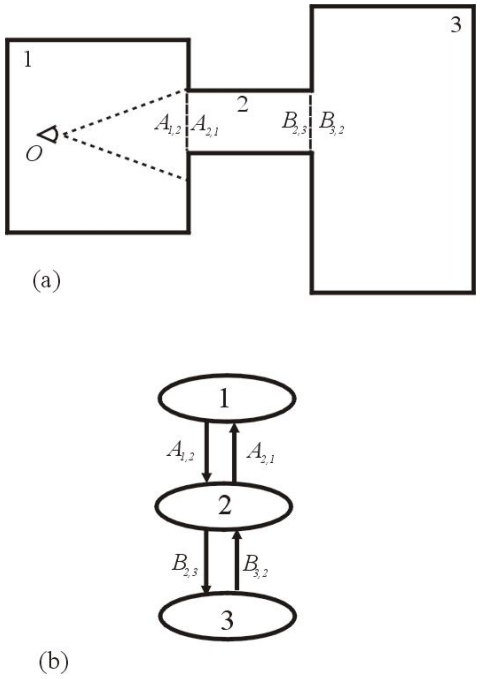


figura 2.3 - (a) Exemplo de um conjunto de células e seus respectivos portais bidirecionais. (b) grafo direcionado que representa a estrutura do exemplo da figura.

A relação dos portais com a área de *ibr* é explorada em (Aliaga, 1997), onde se apresenta uma otimização, fazendo com que os portais correspondam a texturas pré-calculadas (figura 2.4). Quando o observador se aproxima do portal, este passa a ser tratado como geometria, utilizando-se um algoritmo de *warping* para a transição. Com esta idéia torna-se necessário calcular a visualização apenas da célula onde o observador se encontra, uma vez que as demais células serão tratadas como imagens. Se por um lado este método traz uma grande aceleração, por outro lado traz problemas relacionados à falta de sensação espacial, já que as imagens das células anexas ficam estáticas, problema semelhante ao que ocorre com os *sprites*, conforme se verá na seção 3.2. Em (Rafferty, 1998a) o autor procura solucionar este problema sugerindo fornecer a cada portal um conjunto de

imagens pré-calculadas, com vistas da célula anexa, a partir de ângulos diferentes (figura 2.4). Dependendo da posição do observador, escolhe-se a vista que lhe é mais adequada. Este método aumenta a sensação de imersão, mas ainda gera transições descontínuas ao intercalar as imagens. Neste mesmo trabalho, assim como em (Rafferty, 1998b), apresenta-se como uma possível solução para este problema a utilização de algoritmos de 3D *image warping*, baseado na equação de *warping* de McMillan, conforme será apresentado na seção 2.4. Desta maneira, além de eliminar o efeito de transição descontínua, possibilita-se que haja um número menor de imagens para cada portal. Surgem, no entanto, alguns problemas relacionados à falta de informação para algumas posições em que o observador se encontra (buracos). Em (Popescu, 1998) apresenta-se uma solução para este mesmo problema utilizando a técnica de *Layered Depth Images*, que consiste basicamente numa imagem com várias camadas de cores para cada *pixel* (Gortler 1997, Max 1995).

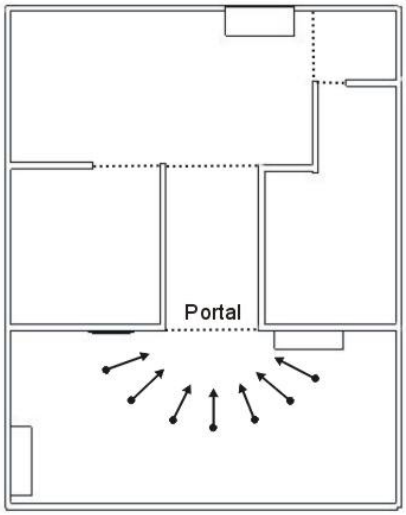


figura 2.4 – Em (Aliaga, 1997) sugere-se que várias possíveis vistas de um portal sejam pré-renderizadas e armazenadas na forma de textura. Dependendo da posição do observador, escolhe-se a imagem mais adequada deste conjunto.

Na pesquisa desta dissertação, procura-se de certa forma uma otimização semelhante à obtida nestes trabalhos de portais: reduzir cenas de natureza geométrica complexa num conjunto de imagens. Em (Rafferty, 1998a) e (Aliaga, 1999) uma cena é dividida em diversas células. Apenas a geometria da célula onde o observador se encontra é visualizada a cada frame. Para as outras células

são geradas imagens separadamente e utilizadas como texturas projetadas nas faces adjacentes à célula onde o observador se encontra, utilizando-se técnicas de *warping* de imagens para corrigir erros de paralaxe* que podem surgir.

Imagens com centros de projeção múltipla (Rademacher, 1998) podem ser vistas como uma adaptação otimizada do *Lumigraph* e *Light Field* para objetos baseados em imagens: os objetos são representados como conjuntos de imagens unidimensionais, tomadas cada uma a partir de uma direção específica.

2.2.4 Modelagem de objetos por imagens

Várias abordagens de *ibr* foram desenvolvidas com o objetivo de modelar objetos individuais, ao invés de cenários ou panoramas. Como os impostores com relevo se enquadram dentro deste tema, apesar de serem também uma forma de *cache* e *culling*, todo o capítulo 3 é dedicado a uma abordagem mais detalhada sobre o tema. Neste capítulo, depois de realizar uma descrição dos principais trabalhos existentes, introduz-se o conceito de impostores com relevo.

2.3 Componentes de classificação para os métodos de *ibr*

Como foi visto, a impraticabilidade da implementação da função plenóptica no seu caso geral faz com que os sistemas de *ibr* procurem criar amostras da mesma, a partir de um conjunto finito e discreto de dados. Isto implica em que toda abordagem possui restrições inerentes à solução proposta, apesar dos grandes progressos que se vêm obtendo recentemente. Não existe - pelo menos até agora - uma solução ótima para todos os casos, o que obriga que sejam elaborados métodos classificativos capazes de analisá-los e compará-los entre si, possibilitando uma escolha adequada para necessidades específicas. A seguir expõem-se resumidamente alguns tópicos que permitem realizar uma análise comparativa para as diversas técnicas existentes. No capítulo 3 os impostores com relevo são discutidos de acordo com estes critérios.

a) Processo de autoria: A viabilidade de um sistema de *ibr* pode ser ditada em

* Deslocamento da posição aparente de um corpo devido à mudança de ponto de vista do observador.

muitos casos pela dificuldade ou facilidade que o seu processo de autoria possui. Um sistema ideal seria aquele onde qualquer usuário poderia construir o ambiente ou os objetos, sem a necessidade de equipamentos especiais ou sofisticados. Algumas aplicações necessitam, por exemplo, a profundidade de cada pixel. Esta informação pode ser obtida em alguns casos na fase de captura da imagem, sendo necessário para tal a utilização de equipamentos especiais, como o que está descrito em (Nyland, 2001). No caso de imagens sintéticas, esta informação é trivial de ser obtida, bastando armazenar o *Z-buffer* correspondente a cada *pixel*. Em alguns casos, pode ser que a aplicação apresente soluções para extrair estes dados, como em (Oh, 2001) em que se realiza uma aproximação geométrica da imagem.

Caso o sistema consista em criar um panorama para modelar um ambiente, é importante distinguir qual a geometria do objeto que serve de suporte. Para aplicações de panoramas cilíndricos é necessária uma ferramenta de autoria, como apresentado em (Szeliski, 1996) para compor uma seqüência de fotos. Já para panoramas esféricos ou cúbicos serão necessários aparelhos de captura ou métodos de composição mais sofisticados, como os sugeridos em (Aliaga, 2001). Pode ser interessante indicar qual o método utilizado para estimar a posição de câmera, caso isto seja necessário para o algoritmo. Em (Takahashi, 2000), por exemplo, o método indica a utilização de um sistema de *Global Positioning System* (GPS), o que inviabiliza sua implementação para muitos casos.

Para o *Light Field* e o *Lumigraph*, bem como os trabalhos que se derivam destes, o critério de autoria revela-os como métodos custosos - pois é necessário um conjunto grande de amostras. No processo de aquisição das imagens pode-se imaginar que ambos os planos serão malhas com um espaçamento de tamanho fixo entre os nós. A câmera será colocada numa coordenada (*s*, *t*) e são tiradas *M* x *N* fotos, onde *M* e *N* é a resolução do outro plano (alvo da câmera). Para cada posição da câmera o seu alvo percorre cada uma das coordenadas (*u*, *v*) do outro plano. Embora, a rigor, não seja necessário nenhum equipamento especial, Gortler (1996) sugere a montagem de uma plataforma baseada em padrões de cores para determinar a posição da câmera durante a captura de amostras. Já Levoy (1996) utiliza um aparelho controlado por computador para posicionar uma câmera corretamente.

b) Número de variáveis da função plenóptica que podem ser manipuladas continuamente e em Tempo Real (graus de liberdade): Na prática, todos os métodos de visualização baseados em *ibr* possibilitam a manipulação de um sub-conjunto de variáveis (menor ou igual a 7) da função plenóptica. Quanto maior o número de variáveis livres, maior é o grau de liberdade permitido na interatividade do sistema implementado.

Por exemplo, restringindo-se o problema para cenas estáticas e para um comprimento de onda fixo, reduz-se a função plenóptica para 5 dimensões. McMillan (1995) usa imagens com valores de profundidade para os *pixels* de forma a reconstruir uma função de 5 dimensões. Outros exemplos de trabalhos que reduzem a função plenóptica para esta dimensão são (Chen 1993, Kang, 1996).

Para espaços sem obstrução pode-se reduzir a função para 4 dimensões, fazendo com que a cena ou o observador estejam presos a uma caixa ou a um cilindro. São exemplos desta redução os trabalhos de Levoy (1996) e Gortler (1996).

No trabalho de Shum (1999) os autores capturam imagens e fazem com que o movimento da câmera esteja preso a círculos concêntricos e paralelos ao chão, sendo um exemplo de redução da função para 3 dimensões.

Finalmente, para o caso de se fixar o observador e se permitir alterar apenas a direção do observador e o fator de *zoom*, chega-se a uma função plenóptica bidimensional. Exemplos típicos para estes casos são os panoramas esféricos e cilíndricos, onde se destacam os trabalhos de Chen (1995), Szeliski (1996) e Szeliski (1997), bem como a técnica de *skyboxes*.

c) Continuidade e limitações impostas na mudança do valor dos parâmetros V_x, V_y, V_z da função plenóptica: Estes parâmetros descrevem a posição onde se encontra o observador. Alterar o valor de alguma destas variáveis corresponde a caminhar com o observador numa cena. Alguns sistemas apenas permitem alterar estes valores de forma discreta, o que faz o observador dar saltos de um local para outro, tais como (Chen 1995, Szeliski 1997). A vantagem destes sistemas é que na maioria das vezes, o processo de autoria é mais fácil e não é necessário obter informações de profundidade dos *pixels*.

São exemplos de sistemas onde esta mudança é contínua (Buehlere 2001, Aliaga 2001). Para fazer esta mudança do observador ser continua, é necessário deformar e interpolar um conjunto de imagens, para pontos de vistas distintos. Estas interpolações em geral trazem alguns problemas, tais como *image fold* (mais de um *pixel* da imagem de referência são mapeados para um mesmo *pixel* da imagem resultante) e surgimento de buracos (informações que estavam oclusas na imagem inicial passam a ser necessárias na imagem gerada).

d) Densidade de amostras para situações ideais: Cada sistema tem sua base de dados ideal. Em muitos casos, para que a interatividade e imersão dentro de um ambiente sejam completas, é necessário um grande volume de amostras de imagens. Em geral, sistemas que levam em conta a profundidade do *pixel* têm a vantagem de solicitar amostras mais esparsas de imagens (Debvec, 1996). Para se fazer uma medida objetiva deste valor, pode-se criar uma relação entre o tamanho da região que se permite navegar com o número de imagens necessárias para montar este cenário. Em (Chai, 2000), por exemplo, realiza-se um estudo detalhado da quantidade mínima de imagens necessárias para métodos baseados no *Light-Field* (Levoy, 1996).

e) Extração de dados geométricos a partir das imagens: Costuma-se dividir os sistemas de *ibr* em dois conjuntos distintos: aqueles que requerem informações da geometria da cena, tendo como um extremo algoritmos denominados *View Dependent Texture Mapping* e aqueles que não requerem nenhuma informação da geometria, tendo como entrada apenas conjuntos de imagens. Alguns autores preferem inclusive separar os métodos, chamando o primeiro de modelagem baseada em imagens (Debvec, 1996) e o segundo de renderização baseada em imagens. Para os casos onde se requer informações de geometria devem ser desenvolvidas ferramentas especiais de autoria, que permitam esta extração manual ou automática. Possuem em geral a vantagem de necessitar menos imagens de entrada. Entretanto, por ser necessário realizar a estimativa da geometria da cena, o sistema depende da complexidade do local ou do objeto que está sendo criado.

f) Inclusão da variável tempo da função plenóptica: Poucos trabalhos foram

feitos até agora tendo em vista este propósito. Grande parte das aplicações de *ibr* consistem em caminhar dentro de cenários ou realizar a visualização de alguns objetos, mas não prevêem que o tempo transcorra linearmente, permitindo que diversos objetos estejam movendo-se individualmente. Isto porque a densidade de amostras se torna extremamente grande ao armazenar sub-conjuntos de imagens animadas.

Para um estudo classificatório das técnicas de *ibr* mais aprofundado veja-se (Buehlere, 2001) e (Clua, 2003b).

2.4 3D *Image Warping*

Para as técnicas que requerem uma alteração contínua em pelo menos uma das variáveis da função plenóptica, tendo-se em conta que é impraticável ter um conjunto infinitamente grande de imagens da cena ou do objeto, uma solução frequentemente adotada consiste na deformação (*warping*) das imagens fonte.

Assim, uma deformação sobre uma imagem pode ser feita para, por exemplo, dar a impressão de que o observador andou para frente ou olhou para outra direção.

Existem diversas abordagens matemáticas para se realizar esta deformação bidimensional, tais como as apresentadas detalhadamente em (Gomes, 1997). Nas sessões 2.4.2 e 2.4.3 apresentam-se respectivamente dois métodos convenientes para a elaboração do conceito estendido de impostores: O *view-morphing* (Seitz, 1996) e o 3D *Image Warping* (McMillan, 1997), que é o modelo escolhido para a implementação deste trabalho.

2.4.1 Definição de *Warping* em Imagens

O *warping* de uma imagem consiste numa função $w: U \rightarrow W$, onde $U, W \subset R^2$. Esta função transforma a posição de um ponto pertencente a uma imagem entrada ou fonte U, produzindo uma nova imagem, que será chamada de imagem destino W.

Intuitivamente, o *warping* de imagens pode ser entendido como o deslocamento de um *pixel* de uma imagem fonte para outra posição. O resultado obtido ao deslocar todos os *pixels* da imagem fonte gera uma imagem destino.

Um exemplo de aplicação direta do processo de *warping* consiste no *morphing* de imagens: transições suaves de uma imagem para outra.

2.4.2 View-Morphing

Chen (1993) e Sauer (2002) descrevem uma técnica capaz de criar transições suaves de imagens, de maneira a poder gerar situações intermediárias entre elas, denominada de *optical flow*. Esta técnica requer que sejam fornecidos ou calculados vetores com a direção e a intensidade do movimento que cada *pixel* deverá sofrer para alterar corretamente a imagem fonte. Entretanto, esta técnica tem uma série de limitações com relação à preparação das imagens para serem utilizadas, já que fornecer ou calcular estes vetores pode em alguns casos ser impraticável. Uma série de trabalhos mais recentes aborda este problema, tais como (Horry 1997, Szeliski 1997, Kanade 1997). A técnica do *view morphing* (Seitz, 1996) requer um conjunto de imagens de diversas vistas do objeto sendo representado e é capaz de gerar vistas do elemento para ângulos não fornecidos pelo conjunto de figuras. Isto é feito sem a necessidade de vetores de direção de movimento para cada *pixel*, mas sim realizando uma interpolação suave entre duas imagens (figura 2.5).

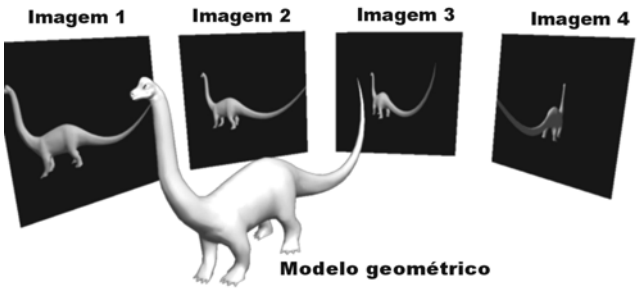


figura 2.5 – Para criar um objeto baseado na técnica de *view morphing*, deve-se ter um conjunto de imagens tiradas ou geradas ao redor do objeto, de maneira a poder interpolar as vistas intermediárias.

O *morphing* tradicional é determinado por duas imagens I_0 e I_1 e pela função de transição C_0: $I_0 \rightarrow I_1$ e C_1: $I_1 \rightarrow I_0$. Existe uma correspondência entre pontos de

uma imagem e outra. A correspondência para alguns destes pontos chaves é dada pelo usuário, sendo que os demais pontos são calculados automaticamente por uma função de interpolação. Fornecer estes pontos chaves é um dos principais inconvenientes do *view-morphing* em relação ao 3D *image warping*, pois requer um processo de autoria mais sofisticado.

Uma função de *warping* será construída a partir da função de transição:

$$W_0(p_0, s) = (1-s)\,p_0 + sC_0\,p_0$$
$$W_1(p_1, s) = (1-s)C_1\,p_1 + sp_1 \qquad (2\text{-}2)$$

W_0 e W_1 são assim funções que retornam o valor de deslocamento de cada ponto $p_0 \in I_0$ e $p_1 \in I_1$ em função de $s \in [0, 1]$. Assim, uma imagem intermediária I_s é criada aplicando a função de *warping* às duas images I_0 e I_1 e calculando a média da cor dos pixels das imagens resultantes do *warping*.

O problema maior que se tem ao aplicar este cálculo de *warping* sobre duas imagens tiradas de um mesmo objeto com uma posição de câmera ligeiramente alterada consiste em que as retas (contornos, por exemplo) do objeto não serão consistentes ao longo da transição. Isto faz que, por exemplo, uma reta possa temporariamente converter-se numa curva, como mostra a figura 2.6.

Diz-se que uma transformação é *shape preserving* no caso em que dadas duas imagens de um mesmo objeto visto de ângulos diferentes, esta transformação seja capaz de gerar uma imagem representando uma vista intermediária do mesmo objeto, sem sofrer nenhuma deformação.

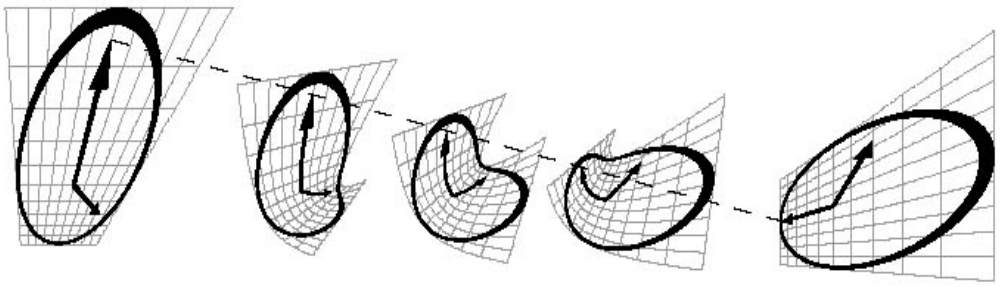


figura 2.6 – O *warping* que ocorre nestas duas imagens não é *shape preserving*, pois ocorre uma deformação nas imagens intermediárias. (Seitz, 1996)

Os algoritmos tradicionais de *morphing* (Wolberg 1990, Lee 1992, Beier 1992) não podem ser utilizados para esta finalidade, pelo fato de não serem *shape*

preserving. Já o algoritmo de *view morphing* descrito por Seitz (1996) é capaz de garantir esta propriedade.

Para calcular o *morphing* são necessárias, além das duas imagens representando vistas diferentes do mesmo objeto I_0 e I_1, suas respectivas matrizes de projeção Π_0 e Π_1 e um conjunto de *pixels* de correspondência de uma imagem para outra. A princípio, esta correspondência deve ser fornecida pelo usuário.

Estes dados são em geral necessários para qualquer técnica de *morphing.* Em geral, o *morphing* é correto quando a correspondência for correta e completa (quando isto não ocorrer, surgem "buracos" nas imagens intermediárias, devido à falta de informações). No caso do *view morphing*, esta correspondência correta garante também que as imagens geradas sejam *shape preserving*.

Resumidamente, o algoritmo funciona da seguinte maneira: sejam I_0 e I_1 as imagens a serem interpoladas e $\Pi_0 = [H_0 \mid -H_0C_0]$, $\Pi_1 = [H_1 \mid -H_1C_1]$ suas respectivas matrizes de projeção. De maneira a simplificar os cálculos, convém escolher um sistema de coordenadas, cujo eixo X coincide com a reta em que o observador se moveu. Isto permite ter-se $C_0 = (X_0, 0, 0)$ e $C_1 = (X_1, 0, 0)$. O eixo Y deste sistema pode ser obtido pelo produto vetorial entre as normais das duas imagens. Feito isto, imagens interpoladas de I_0 e I_1, conforme ilustra a figura 2.7, podem ser obtidas pela posição do observador C_s, dada pela equação de interpolação e pela matriz de projeção $\Pi_s = [H_s \mid -H_sC_s]$. Assim, o processo de *view morphing* se resume a três etapas:

1) Aplicar a matriz de transformação de projeção H_0^{-1} a I_0 e H_1^{-1} a I_1, produzindo as imagens intermediárias I^*_0 e I^*_1. Esta primeira etapa faz com que os planos das duas imagens estejam paralelos, sem alterar o centro de cada uma. É interessante notar que I^*_0 e I^*_1 representam vistas dadas pelas matrizes de projeção $\Pi^*_0 = [I_d \mid -C_0]$ e $\Pi^*_1 = [I_d \mid -C_1]$, sendo I_d a matriz identidade de dimensão 3. Estas duas imagens intermediárias possuem uma propriedade importante que é o fato de os pontos que se correspondem estarem na mesma linha da imagem, o que permite que a interpolação necessária para gerar I_s seja feita numa única dimensão.
2) Gerar I^*_s através de uma interpolação linear da posição dos pontos e das cores de pontos correspondentes entre I^*_0 e I^*_1 utilizando alguma equação de

interpolação. Esta etapa corresponde ao *morphing* propriamente dito e move o centro da imagem para C_s.

3) Aplicar H_s a I^*_s de maneira a obter a imagem interpolada I_s. Aqui finalmente se transforma o plano da imagem para a sua posição e orientação corretas.

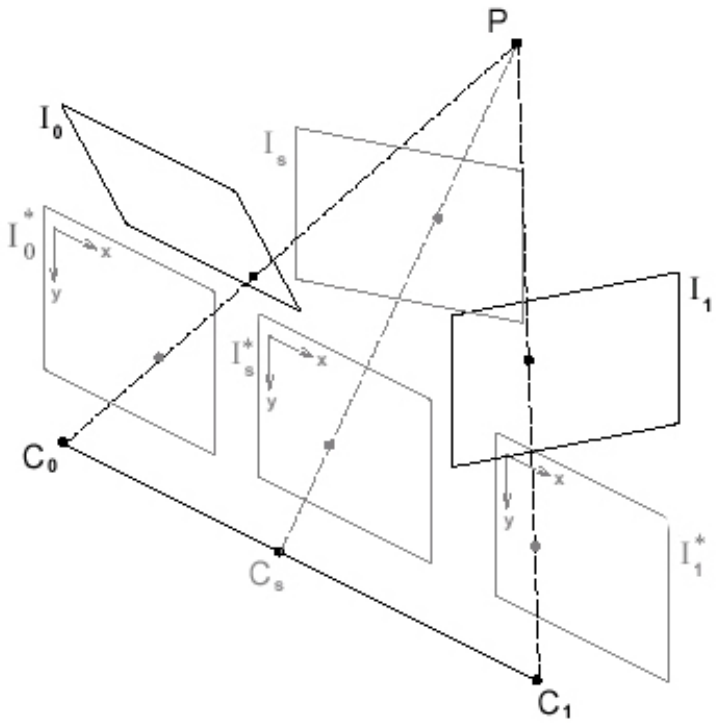


figura 2.7 – I_0 e I_1 correspondem às imagens originais. I^*_0 e I^*_1 são as imagens intermediárias obtidas a partir da multiplicação da matriz de projeção com H_0^{-1} e H_1^{-1}. I^*_s consiste na interpolação entre as imagens intermediárias e I_s será o resultado final do *view morphing* (Seitz, 1996).

2.4.3 3D Image Warping

O *3D image warping* definido por McMillan (1997) consiste numa função de transformação geométrica $w:U' \rightarrow W \subset \boldsymbol{R^2}$ capaz de mapear uma imagem fonte i_f para uma imagem destino i_d. A imagem fonte deve conter, além das cores dos *pixels*, a profundidade correspondente à superfície que está sendo representada pelo respectivo *pixel*. Além disso, conhecem-se os dados relacionados à câmera desta imagem fonte (a posição $\dot{C}_f$ bem como o seu plano de projeção). Utilizando o modelo de câmera perspectiva tem-se que um ponto $\dot{x}$ (figura 2.8) pertencente à cena geométrica pode ser descrito pela equação 2-3:

$$\dot{x} = \dot{C}_f + (\vec{c} + u_f \vec{a} + v_f \vec{b}).t_f(u_f, v_f) \quad (2\text{-}3)$$

Onde (u_f, v_f) são as coordenadas da projeção deste ponto sobre o plano de projeção da câmera, os vetores $\vec{a}$ e $\vec{b}$ formam a base para o plano da imagem e os comprimentos correspondem às medidas de cada *pixel* no espaço euclidiano, $\vec{c}$ é o vetor da direção dada pelo centro de projeção à origem do plano da imagem e o coeficiente $t_f(u_f, v_f)$ é dado pela razão da distância de $\dot{C}_f$ até $\dot{x}$ pela distância de $\dot{C}_f$ até a projeção de $\dot{x}$ no plano, dado pela coordenada (u_f, v_f). A equação 2-4 corresponde ao cálculo deste coeficiente:

$$t_f(u_f, v_f) = \frac{|\dot{x} - \dot{C}_f|}{|\vec{c} + u_f \vec{a} + v_f \vec{b}|} \quad (2\text{-}4)$$

A equação (2-3) pode ser rescrita utilizando operação de matrizes da seguinte maneira:

$$\dot{x} = \dot{C}_f + \begin{bmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{bmatrix} \begin{bmatrix} u_f \\ v_f \\ 1 \end{bmatrix} t_f(u_f, v_f) = \dot{C}_f + P.\vec{x}.t_f(u_f, v_f) \quad (2\text{-}5)$$

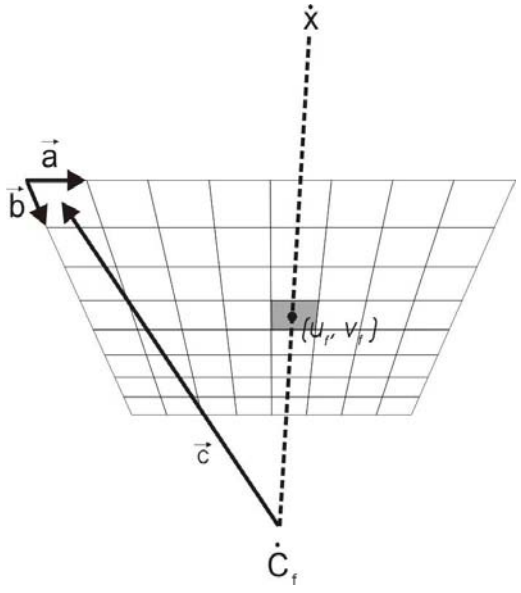


figura 2.8 – O ponto (u_f, v_f) é a projeção de $\dot{x}$ para o modelo da câmera fonte $\dot{C}_f$.

De forma semelhante, para uma outra posição do observador dada por $\dot{C}_d$ (figura 2.9), o mesmo $\dot{x}$ pode ser descrito da seguinte maneira:

$$\dot{x} = \dot{C}_d + P_d.\vec{x}_d.t_d(u_d, v_d) \tag{2-6}$$

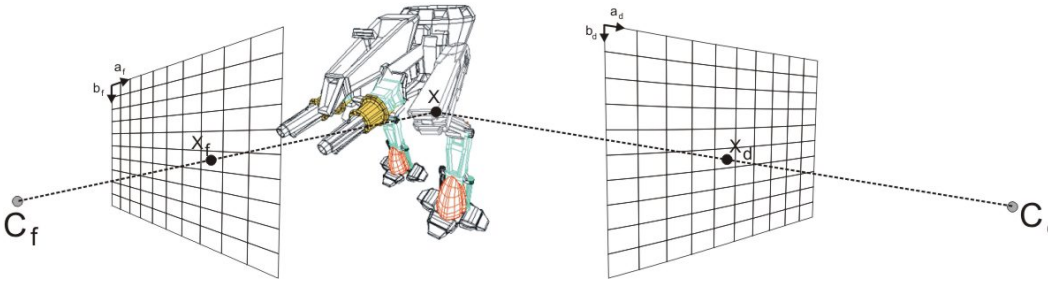


figura 2.9 – x_f e x_d são as projeções do ponto x sobre os planos de projeção das câmeras C_f (fonte) e C_d (destino), respectivamente.

Como $\dot{x}$ corresponde ao mesmo ponto, visto por observadores colocados em posições diferentes, pode-se criar a relação de equivalência entre as equações (2-5) e (2-6):

$$\dot{x} = \dot{C}_d + P_d.\vec{x}_d.t_d(u_d, v_d) = \dot{C}_f + P.\vec{x}.t_f(u_f, v_f) \tag{2-7}$$

Esta igualdade pode ser interpretada como sendo duas retas que se interceptam no ponto $\dot{x}$. Desenvolvendo a equação (2-7) tem-se:

$$P_d.\vec{x}_d.t_d(u_d, v_d) = (\dot{C}_f - \dot{C}_d) + P.\vec{x}.t_f(u_f, v_f)$$

$$\vec{x}_d.t_d(u_d, v_d) = P_d^{-1}[(\dot{C}_f - \dot{C}_d) + P.\vec{x}.t_f(u_f, v_f)] \tag{2-8}$$

Desta igualdade pode-se concluir que vale a seguinte equivalência projetiva[*]:

[*] O vetor resultante possui a mesma direção e sentido, podendo ser apenas o módulo diferente.

$$\vec{x} \doteq P_d^{-1}[P_f.\vec{x}_f.t_f(u_d,v_d) + (\dot{C}_f - \dot{C}_d)] \quad (2\text{-}9)$$

Esta equação explicita uma propriedade importante em relação à imagem que se deseja gerar: qualquer nova posição do observador não requer a informação de profundidade dos *pixels* da imagem a ser gerada (que na verdade é uma informação que não se dispõe), sendo suficiente apenas a profundidade dos *pixels* da imagem fonte.

Para chegar à formulação final da equação de 3D *image warping* de McMillan (equação 2-10), divide-se a equação (2-8) por $t_f(u_f, v_f)$ e distribui-se a matriz inversa de projeção P_d^{-1}:

$$\vec{x} \doteq P_d^{-1}P_f.\vec{x}_f + P_d^{-1}(\dot{C}_f - \dot{C}_d).\partial_f(u_f,v_f) \quad (2\text{-}10)$$

Onde $\partial_f(u_f,v_f) = 1/t_f(u_f,v_f)$ é denominado de disparidade generalizada do *pixel* (u_f, v_f) da imagem fonte.

Esta equação pode ser vista como a composição de duas transformações bidimensionais: A primeira parcela representa uma transformação perspectiva planar homográfica sobre a imagem fonte (que pode ser vista como uma projeção de textura) e a segunda equivale a uma transformação *pixel* a *pixel*[**], proporcional ao valor da disparidade generalizada (dada pelo termo $\partial_f(u_f,v_f)$) na direção do ponto epipolar do plano da imagem destino (Oliveira, 2000).

A parcela de transformação perspectiva pode ser resolvida pelas implementações convencionais de projeção de textura por hardware. Quanto à transformação *pixel* a *pixel*, é possível reescrevê-la através de uma estrutura unidimensional simples, o que permite uma eficiente implementação por *software*.

O capítulo 3, após apresentar diversos métodos existentes para modelar elementos por imagens, discute com mais detalhes a técnica de texturas com relevo proposta por Oliveira (2000), que consiste numa implementação da equação de 3D *image warping* apresentada. Os impostores com relevo são uma

[**] Também conhecido com o nome de operação *per-pixel*

extensão desta abordagem, procurando aumentar sua capacidade para modelar objetos quaisquer.

2.5 Discussão

As duas formas de *warping* apresentadas podem servir para se elaborar os impostores com relevo. Como foi apresentado, o *view morphing* apresenta uma deficiência que consiste em necessitar dos pontos de correspondência entre as duas imagens. Entretanto, dado que os impostores com relevo trabalham com imagens sintetizadas dinamicamente pela aplicação, este problema pode ser resolvido, da seguinte forma: cada vértice do modelo recebe um identificador. Ao renderizar cada imagem, associa-se à estrutura do *pixel* que representa a projeção de cada vértice o identificador correspondente. Ao gerar outra imagem, os *pixels* dos mesmos vértices receberão um identificador equivalente ao da primeira imagem. Os pontos de correspondência serão aqueles com os mesmos identificadores.

A principal vantagem do *warping* apresentado por McMillan (1997) consiste em que, para cada posição de câmera, apenas é necessário uma imagem fonte. Esta abordagem requer, no entanto, que seja fornecida a profundidade de cada *pixel*, o que é trivial para o caso de imagens sintetizadas. No próximo capítulo, após uma síntese dos principais métodos para modelar objetos através de imagens, detalha-se o funcionamento do mapeamento de texturas com relevo e introduz-se o conceito de impostores com relevo.