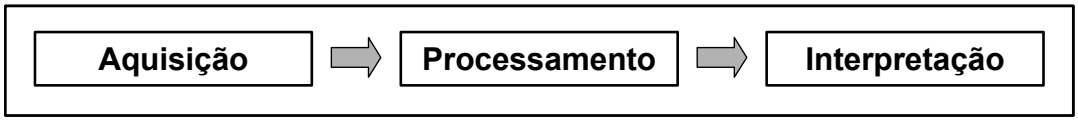
2
Conceitos básicos

O caráter interdisciplinar deste trabalho torna necessária a apresentação de conceitos básicos da área de exploração sísmica e de Computação Gráfica. Na próxima seção faremos uma breve descrição do processo de obtenção de dados sísmicos 3D. Em seguida introduziremos os atributos sísmicos instantâneos, que têm um papel importante no nosso estudo.

As discussões e os resultados relativos à visualização volumétrica direta de dados sísmicos contidos neste documento (capítulos 3 e 4) são independentes de implementação, ou seja, são pertinentes às implementações de sistemas de visualização volumétrica de dados sísmicos em geral. Porém, para validar nossas idéias, desenvolvemos um sistema de visualização volumétrica baseado na utilização de texturas 3D. Escolhemos esta opção por permitir um bom nível de interatividade com o modelo. Atualmente existe uma vasta literatura sobre o uso de texturas 3D na visualização volumétrica direta. Na seção 2.2 faremos um apanhado das técnicas usadas na nossa implementação.

2.1.
Dados sísmicos 3D

A sísmica de reflexão é um método indireto de exploração da subsuperfície da terra. Este método vem sendo largamente utilizado pelo fato de ser capaz de cobrir grandes áreas e ser extremamente mais econômico quando comparado com um método direto, como, por exemplo, a perfuração de poços. Segundo Robinson e Treitel (1980) a exploração de hidrocarbonetos, óleo e gás baseada em sísmica pode ser dividida em três etapas: aquisição, processamento e interpretação.



Exploração (Óleo e Gás) baseada em Sísmica

Figura 1 – Etapas da exploração baseada na sísmica de reflexão.

Nas subseções seguintes faremos uma breve descrição de cada etapa. Para uma visão mais aprofundada sugerimos consultar os trabalhos de Gerhardt (1998), Machado (2000), Robinson e Treitel (1980) e Yilmaz (1987).

2.1.1. Aquisição

A aquisição é feita usando uma fonte para gerar ondas sísmicas que se propagam abaixo da superfície da terra. Em aquisições terrestres é comum usar explosões de dinamite como fonte; em aquisições marinhas são usados normalmente dispositivos pneumáticos como canhões de ar. Quando a onda sísmica alcança uma interface entre duas camadas de rocha com valores de impedância acústica diferentes, parte da onda é refratada e continua viajando para baixo; outra parte é refletida e retorna a superfície. A porção da energia refletida é proporcional à diferença de impedância acústica entre os dois meios. A parte refletida da onda que retorna à superfície é captada nos receptores (geofones em aquisições terrestres ou hidrofones em marinhas) e gravada nos sismógrafos. O sismógrafo armazena tanto o tempo de chegada da onda quanto a intensidade medida neste momento. Após várias detonações variando a posição da fonte e dos receptores, todos os dados armazenados são enviados para serem processados. A Figura 2 ilustra os processos de aquisição terrestre e marinha.

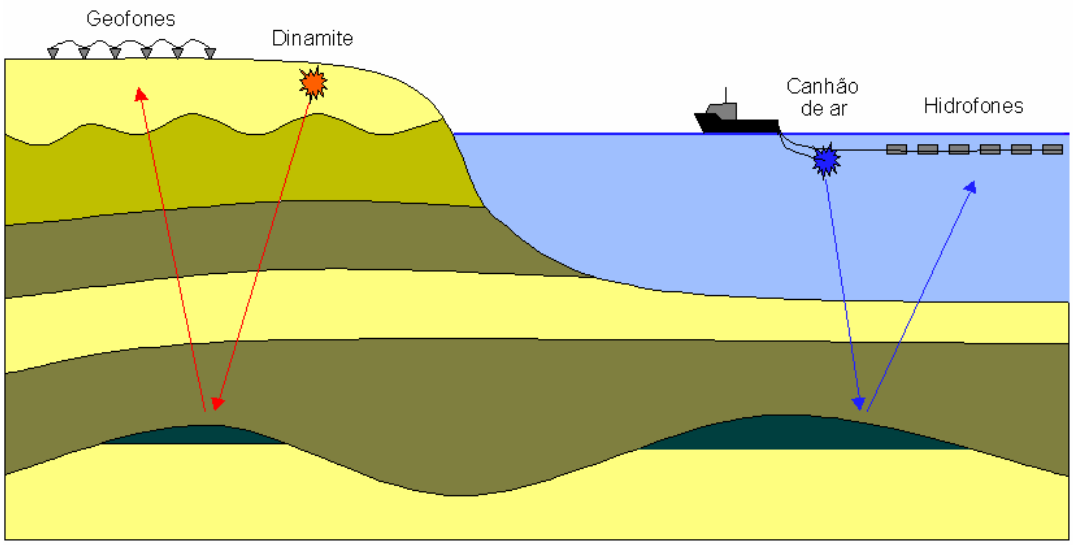


Figura 2 – Aquisição sísmica. Adaptada de Gerhardt (1998).

2.1.2. Processamento

Na etapa de processamento, alguns erros inerentes ao levantamento sísmico são corrigidos. Além disso, os dados são reorganizados para formarem uma grade tridimensional com uma amostra de amplitude sísmica em cada vértice da grade (*voxel*). Duas das dimensões do conjunto de dados são direções espaciais e estão relacionadas com as posições das fontes e dos receptores. Uma das transformações realizadas nos dados durante o processamento faz com que as posições da fonte e do receptor sejam a mesma. Também graças a esta transformação podemos considerar a terceira dimensão do conjunto de dados como sendo o temporal e que a propagação da onda é feita apenas na direção vertical. Como podemos considerar que a fonte e o receptor estão na mesma posição na superfície, o tempo de cada amostra corresponde ao tempo que a onda leva para viajar até uma interface mais o tempo da volta à superfície. Uma coluna de amostras com as mesmas coordenadas espaciais, variando apenas o tempo, é chamada de *traço sísmico*. Os máximos e mínimos da função de amplitude sísmica do traço são chamados de eventos sísmicos.

A organização das amostras em um dado sísmico é mostrada na Figura 3. Do lado esquerdo temos a função de amplitudes sísmicas do traço sísmico, na qual a única dimensão é a temporal (1D). No centro temos uma seção vertical do conjunto de dados formada por um conjunto de traços sísmicos, que é chamada de *linha sísmica* (2D), com uma dimensão espacial e a outra temporal. No caso dos dados sísmicos 3D (volume sísmico), formados por várias linhas sísmicas, temos duas direções espaciais, que são chamadas de *inline* (direção das linhas sísmicas) e *crossline* (direção perpendicular às linhas sísmicas), além de uma direção temporal.

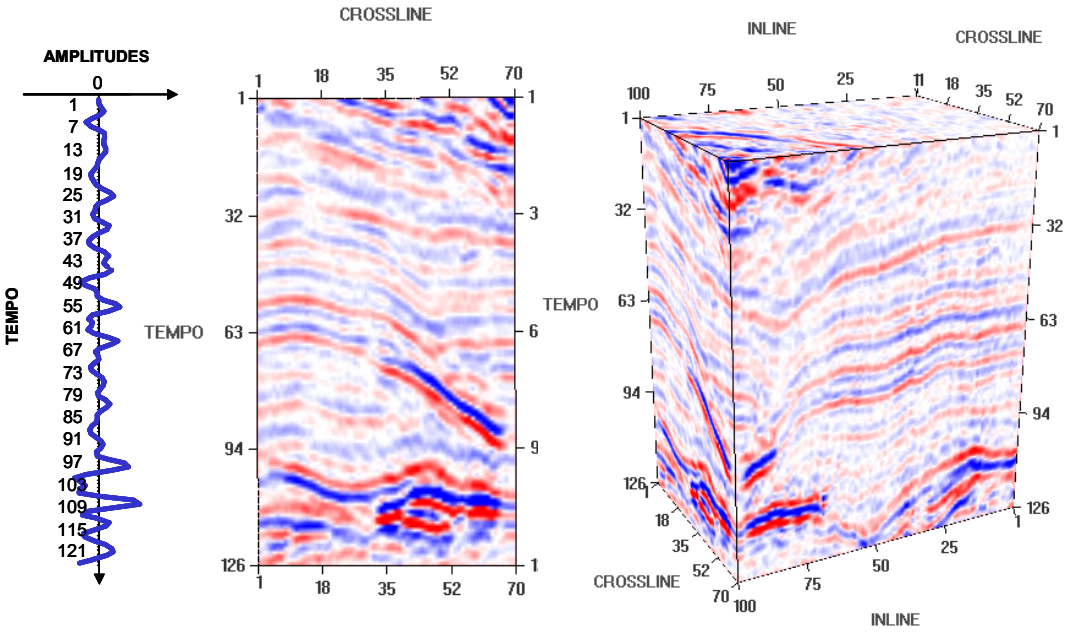


Figura 3 - Traço sísmico (esquerda), linha sísmica (centro) e volume sísmico (direita).

Um modelo matemático interessante que descreve bem o efeito do processamento sísmico realizado sobre o dado é o modelo de convolução, ilustrado na Figura 4. Neste modelo consideramos a função de amplitude sísmica de cada traço do conjunto de dados como sendo o resultado da convolução de um impulso sísmico com uma função refletividade - a rigor, a função refletividade é uma distribuição de coeficientes de reflexão. Os coeficientes de reflexão são proporcionais à diferença de impedância acústica entre camadas geológicas adjacentes.

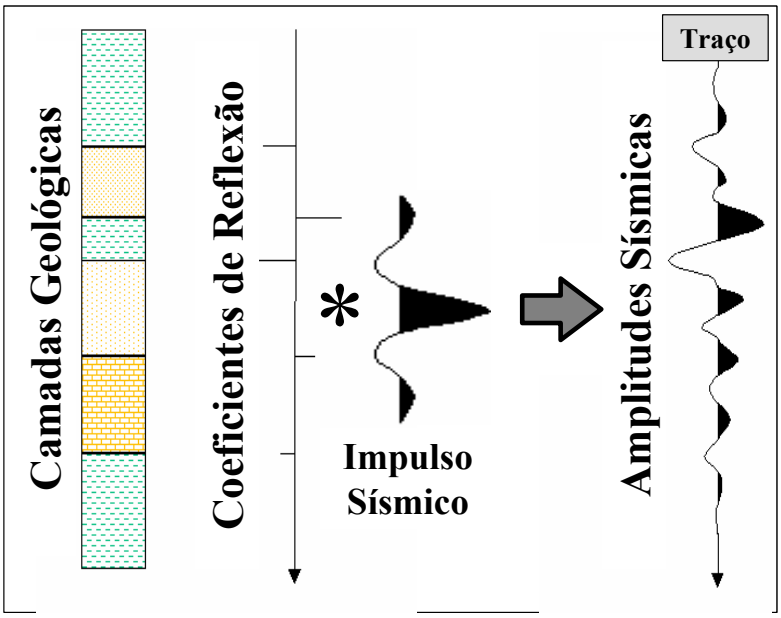


Figura 4 – Modelo de convolução. Adaptado de Gerhardt (1998).

2.1.3.
Interpretação

Na etapa de interpretação o intérprete, em geral um geólogo ou geofísico, analisa os dados sísmicos e tenta criar um modelo que represente a geologia contida na área do levantamento. A Figura 5 mostra um modelo geológico que poderia ser resultante da interpretação de uma linha sísmica. A interpretação sísmica pode ser classificada, de acordo com o foco, em dois tipos: estrutural e estratigráfica. A interpretação estrutural basicamente tenta identificar as camadas geológicas ou, de forma equivalente, as interfaces entre as camadas, bem como as falhas geológicas que recortam as camadas. Na interpretação estratigráfica o foco do trabalho está em entender a maneira como as camadas foram se formando ao longo do tempo.

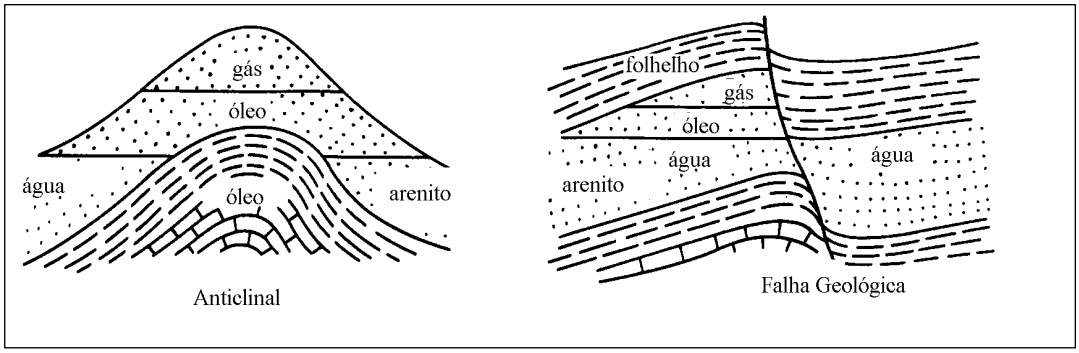


Figura 5 – Modelo geológico. Adaptada de Robinson e Treitel (1980).

Sheriff (1991) define um horizonte sísmico como sendo a superfície que separa duas camadas diferentes de rocha, onde tal superfície (mesmo sem ter sido identificada) está associada com uma reflexão que se estende por uma grande área. Um horizonte sísmico se manifesta em um dado sísmico como uma série de eventos (picos ou vales de amplitudes sísmicas) que aparecem de forma consistente traço a traço. O mapeamento dos horizontes do conjunto de dados é uma das tarefas mais importantes da interpretação sísmica. Os horizontes sísmicos também são chamados de refletores.

2.2.
Visualização volumétrica direta com texturas 3D

A visualização volumétrica é um processo de síntese de imagens no qual os dados de entrada são modelados matematicamente como um campo escalar

tridimensional. A versão discreta deste modelo implícito é um conjunto de amostras tomadas em uma grade regular tridimensional. Cada amostra armazena um número real correspondente ao valor do campo em um dado ponto da grade. A grande maioria das aplicações quantiza as amostras para armazená-las como inteiros de 8 *bits*. Como exemplos de modelos volumétricos podemos citar os dados obtidos por tomografia computadorizada, ressonância magnética, resultados de simulações físicas, equações matemáticas implícitas e dados sísmicos.

Existem duas estratégias distintas na visualização volumétrica. Na estratégia indireta visualizamos superfícies extraídas do modelo implícito. Na abordagem direta utilizamos transparências para conseguir visualizar estruturas tridimensionais complexas dentro do dado. Esta abordagem é indicada para modelos nos quais a extração de superfícies é difícil.

Os algoritmos de visualização volumétrica direta produzem imagens associando cor e opacidade a cada *voxel*. Esta associação é feita indiretamente, isto é, uma tabela de cor e opacidade é criada para os valores armazenados nos *voxels*. Por exemplo, em um algoritmo que usa uma representação de 8 *bits* para os valores armazenados, as tabelas de cor e opacidade possuem 256 entradas. A função que associa a cada valor do campo escalar uma cor e uma opacidade é chamada de *função de transferência*.

Existem vários algoritmos de visualização volumétrica direta. Entre eles podemos citar *ray casting* (Levoy, 1988)*, shear-warp* (Lacroute and Levoy, 1994) e *splatting* (Westover, 1990) como alguns dos mais importantes.

O algoritmo de visualização volumétrica baseado em texturas é equivalente ao algoritmo de *ray casting* e produz resultados semelhantes. Contudo, enquanto no algoritmo de *ray casting* a imagem é construída lançando um raio para cada *pixel*, na abordagem por textura vários raios são processados simultaneamente, uma fatia 2D por vez. O dado é armazenado em uma textura 3D e a fatia é utilizada para amostrar a textura. As fatias com textura aplicada são combinadas para formar a imagem final. Como esta abordagem pode ser acelerada por *hardware*, o algoritmo usando textura é mais rápido do que o *ray casting.* Em sistemas em que o *hardware* gráfico não implementa texturas 3D, uma versão mais limitada pode ser implementada com texturas 2D. Porém, a implementação com textura 3D é mais simples e produz imagens com qualidade melhor.

2.2.1.
Amostragem da textura

A textura 3D em geral é amostrada usando planos perpendiculares à direção de visualização. Porém, quando o observador está muito próximo ao volume, ou mesmo dentro do volume, uma amostragem mais correta da textura pode ser obtida utilizando conchas esféricas centradas na posição do observador. A Figura 6 ilustra as geometrias de amostragem formadas por fatias planas e por conchas esféricas. A geometria de amostragem escolhida é decomposta em polígonos recortados pelos limites do volume e, então, a textura 3D contendo os dados volumétricos é aplicada aos polígonos recortados. As fatias texturizadas são compostas de trás para a frente na direção do observador utilizando o operador *over* (Drebin et al, 1988).

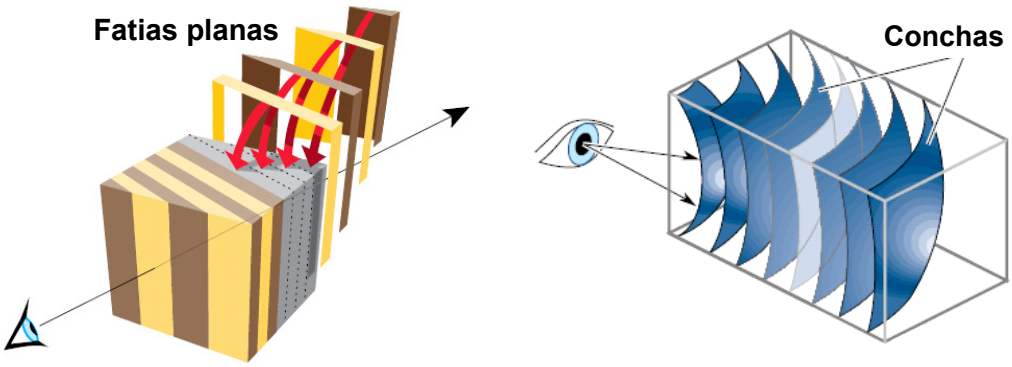


Figura 6 – Geometrias de amostragem. Adaptadas de Blythe e McReynolds (2000).

2.2.2.
Composição

O operador *over* é a maneira mais comum de compor as amostras transparentes na visualização volumétrica. Este operador também é utilizado em outros algoritmos de visualização volumétrica direta, tais como o *ray casting*. O operador *over* aproxima a integral de iluminação definida pelo transporte da luz através do volume transparente. A opacidade (transparência) de uma amostra está armazenada no valor do canal *alfa* do *texel*[1]. *Texels* com valores de *alfa* alto (quase opacos) obstruem a visualização dos *texels* que estão atrás.

[1]. A palavra *texel* deriva da expressão em inglês *texture element*.

Segundo Wittenbrink (1998), para evitar artefatos de interpolação, as cores das amostras devem ser multiplicadas pela opacidade antes de serem interpoladas. Portanto, o operador *over* pode ser implementado definindo a função de *blending* do OpenGL da seguinte forma:

```
glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA);
```

2.2.3. Freqüência de amostragem

Segundo Blythe e McReynolds (2000), existem vários fatores que devem ser levados em conta na hora de escolher o número de fatias (planos ou conchas esféricas) usadas na amostragem da textura.

O desempenho que se deseja obter é um dos fatores mais importantes: quanto maior é o número de fatias, menor o desempenho. Em geral os sistemas de visualização volumétrica baseados em textura operam em dois modos: um modo 'interativo' em que são usadas poucas fatias, para garantir maior desempenho; e um modo 'detalhado' no qual são usadas mais fatias, para obter uma imagem com maior qualidade.

Para evitar artefatos de re-amostragem é importante considerar os *texels* como sendo cúbicos, ou seja, termos uma amostragem uniforme em todas as direções. Portanto, é interessante ter uma freqüência de amostragem entre as fatias compatível com a freqüência de amostragem dos dados. Isto pode ser conseguido definindo o número de fatias igual à soma das dimensões do conjunto de dados (em *texels*).

Aumentar exageradamente o número de fatias pode tornar difícil a visualização da porção mais afastada do conjunto de dados. O operador *over* é não-linear, por isso é difícil compensar o aumento do número de fatias com uma atenuação da sua opacidade.

Quando a visualização é feita com projeção em perspectiva, a freqüência de amostragem deve aumentar com a distância ao observador. Devido à distorção da perspectiva, a parte mais afastada do dado deve parecer mais densa.

2.2.4.
Gerenciando a memória de textura

O algoritmo de visualização volumétrica baseada em texturas 3D é capaz de tratar volumes grandes que não cabem na memória de textura. Para isto o volume é dividido em vários pequenos volumes, chamados de *tijolos*. A Figura 7 ilustra a divisão de um volume em tijolos. O tamanho dos tijolos é escolhido para que cada um caiba na memória de textura.

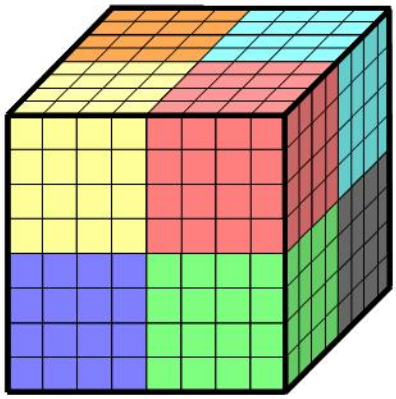


Figura 7 – Volume de dados dividido em tijolos.

Durante a visualização cada tijolo é carregado na memória de textura amostrado por fatias e o resultado é composto, como descrito anteriormente. Assim como as fatias, os tijolos são processados de trás para a frente na direção do observador.

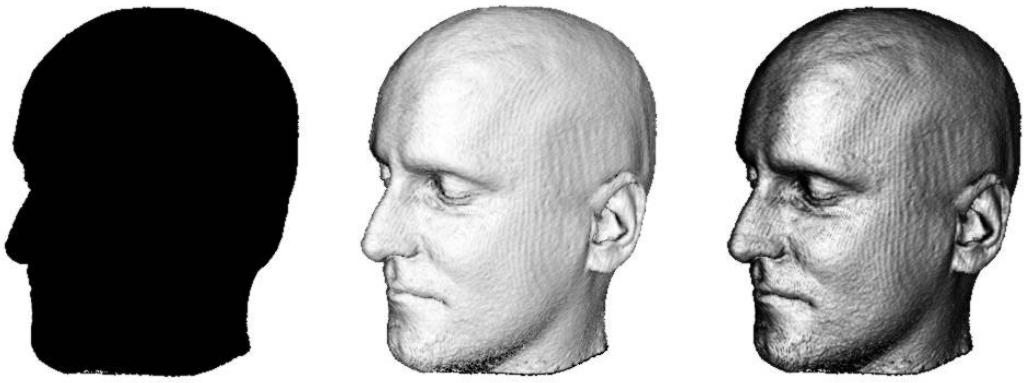
Para evitar erros na re-amostragem do conjunto de dados nas emendas dos tijolos, o recorte dos polígonos e suas coordenadas de textura devem ser ajustados para não usarem os *texels* das faces dos tijolos. Além disso, cada tijolo é construído de forma que os *texels* de suas faces sejam comuns, respectivamente, a cada um dos seus 6 vizinhos. Com isso garantimos que os tijolos encaixem corretamente na imagem resultante.

Podemos encontrar mais informações sobre paginação (*paging)* de texturas em geral na seção 6.8 do trabalho de Blythe e McReynolds (2000).

2.2.5.
Iluminação

Os modelos de iluminação local aproximam pontualmente a interação entre a luz e a superfície de um objeto. Em um modelo local, como o modelo de Phong

(1975), a intensidade da luz refletida em um ponto é calculada em função da orientação da superfície, da posição e direção da fonte de iluminação e de propriedades óticas do material do objeto. Nestes modelos a orientação local da superfície é estimada pelo vetor normal em cada ponto. Portanto, para aplicar um modelo de iluminação local à visualização volumétrica é necessário estimar o vetor normal em cada *voxel*. Para a maioria dos campos escalares, o vetor gradiente pode ser um substituto apropriado para o vetor normal da superfície, uma vez que o gradiente é o próprio vetor normal da superfície de nível em cada *voxel*. A Figura 8 mostra o modelo de Phong aplicado à visualização de uma superfície de nível de um volume de dados médicos. Na figura temos: (a) apenas a componente ambiente; (b) componentes ambiente e difusa; (c) componentes ambiente, difusa e especular.



(a) (b) (c)

Figura 8 – Iluminação de uma superfície de nível de um volume. Adaptado de Hadwiger et al (2002)

Existem diversas maneiras de calcular o gradiente em um dado volumétrico. Em nossa implementação utilizamos o método diferenças centrais, que é derivado dos primeiros termos da série de Taylor. Na visualização volumétrica baseada em textura o campo gradiente é pré-computado e armazenado como uma textura 3D. As três componentes do gradiente normalizado são transformadas do intervalo $[-1,1]$ para o intervalo $[0,1]$ e armazenadas como triplas RGB.

$$I(x,y,z) \tag{1}$$

$$\nabla I = \begin{pmatrix} I_x \\ I_y \\ I_z \end{pmatrix} = \begin{pmatrix} I(x+1,y,z) - I(x-1,y,z) \\ I(x,y+1,z) - I(x,y-1,z) \\ I(x,y,z+1) - I(x,y,z-1) \end{pmatrix} \quad (2)$$

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \frac{\nabla I}{\|\nabla I\|} \quad (3)$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1+n_x \\ 1+n_y \\ 1+n_z \end{pmatrix} \quad (4)$$

Na Figura 9 temos uma visualização do campo vetorial do gradiente em uma superfície de nível. As cores, definidas pelo mapeamento da eq. (4), representam os vetores do campo gradiente do volume.

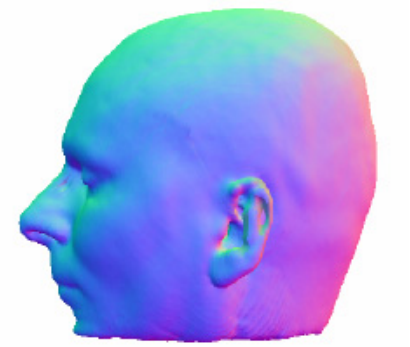


Figura 9 – Gradiente de uma superfície de nível. Adaptado de Westermann e Ertl (1998).

Westermann e Ertl (1998) apresentaram uma implementação para a visualização de superfícies de nível, com o cálculo de iluminação feito dentro da placa gráfica. Ao desenhar os polígonos da geometria de amostragem, o teste de *alfa* do OpenGL é usado para filtrar os *voxels* da superfície de nível desejada. A matriz de cores do OpenGL é utilizada para armazenar a direção da fonte de iluminação. Após a projeção dos polígonos, os *pixels* da imagem são copiados do *frame buffer* para ele mesmo. A cor, contendo os vetores normais, dos *pixels* copiados é multiplicada pela matriz de cores, resultando no cálculo da equação de iluminação.

A arquitetura da placa de vídeo GeForce3 da NVIDIA implementa em *hardware* os combinadores de registros (*register combiners*) além da textura 3D

(Spitzer, 2001). Os combinadores de registros permitem que se façam operações sobre texturas. Uma implementação do cálculo da iluminação feita em *hardware* baseada nesta arquitetura é apresentada no trabalho de Rezk-Salama (2000). A implementação do nosso sistema é baseada nesta abordagem. As placas de vídeo da ATI a partir da RADEON 8500 implementam um recurso de *hardware* chamado *fragment shaders*, que é similar aos combinadores de registros da NVIDIA.

Atualmente as placas gráficas programáveis permitem uma implementação simples do cálculo de iluminação. Esta implementação pode ser feita usando a linguagem Cg (Fernando e Kilgard, 2003).

2.2.6. Função de transferência

A função de transferência tem um papel fundamental na visualização volumétrica direta. É através da função de transferência que podemos determinar as porções visíveis do volume e definir a cor e a transparência com que estas irão aparecer na imagem. A Figura 10 mostra a visualização volumétrica de um dado obtido por tomografia computadorizada de uma peça mecânica. Na imagem à esquerda foi aplicada uma função de transferência que associa uma cor ciano e opacidade alta (igual a 1) aos valores de baixa densidade, tornando-os totalmente opacos, e associa opacidade baixa (0) aos valores de densidade alta, tornando-os invisíveis. A imagem do centro foi obtida tornando invisíveis os *voxels* de densidade baixa associando a estes opacidade 0, e opacidade igual a 1 para os valores de densidade alta, que também receberam a cor vermelha. Na imagem da direita os valores de densidade baixa tornaram-se transparentes recebendo um valor de opacidade intermediário (entre 0 e 1).

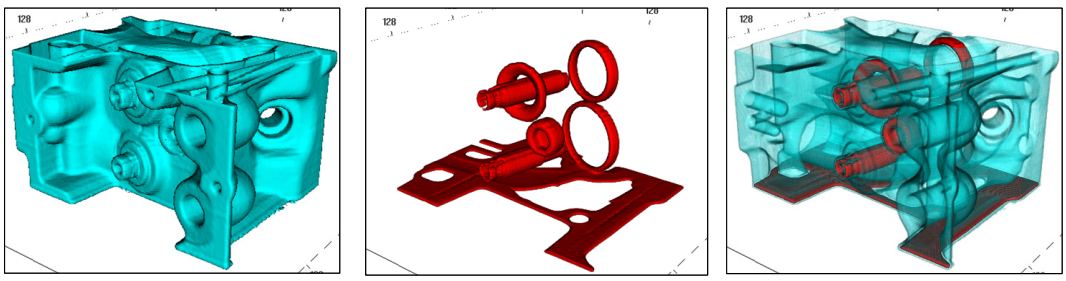


Figura 10 – *Voxels* com densidade baixa (esquerda), *voxels* com densidade alta (centro), *voxels* com densidade baixa e alta (direita).

A função de transferência pode ser vista como uma tabela que associa cor e transparência (RGBA) a cada valor do volume. A função de transferência pode ser implementada utilizando uma das tabelas (*look-up tables*) do OpenGL. A extensão `SGI_texture_color_table` consulta a tabela durante a aplicação da textura, após o valor do fragmento ter sido interpolado. Através da extensão `EXT_color_table` podemos usar uma tabela para converter os valores do volume para RGBA durante o carregamento da textura, porém a textura será uma textura RGBA. Se a extensão `EXT_paletted_texture` estiver disponível podemos armazenar o volume como uma textura de índices. Esta é uma opção interessante, pois podemos ajustar a imagem rapidamente a cada modificação na função de transferência. Atualmente, com as placas de vídeo programáveis podemos implementar a função de transferência usando um *Fragment Program*.

2.2.7. Apresentação do algoritmo

O quadro a seguir apresenta, em linhas gerais, os passos básicos de um algoritmo de visualização volumétrica direta baseada em texturas 3D (Blythe e McReynolds, 2000). Aqui consideramos que a função de transferência é implementada usando a extensão `EXT_paletted_texture` e que a iluminação é feita por meio de combinadores de registros. Consideramos também que existe uma única fonte de luz, direcional e branca (monocromática). Apenas as componentes ambiente e difusa do modelo de Phong serão levadas em conta no cálculo da iluminação.

1. Carregar os dados volumétricos em uma textura 3D.
2. Carregar o campo de vetores normais em uma textura 3D.
3. Definir o número de fatias.
4. Habilitar o operador de composição (*over*).
5. Carregar a direção da iluminação na cor primária.
6. Configurar os combinadores de registros (iluminação).
7. Carregar a função de transferência.
8. Encontrar a posição do observador e a direção de visualização.
9. Computar os polígonos da geometria de amostragem. Usar geração automática de coordenadas de textura.
10. Usar a matriz de transformação de texturas para orientar corretamente a aplicação da textura 3D nas fatias.
11. Visualizar os polígonos de trás para a frente na direção do observador.

A Figura 11 ilustra como os dados que descrevem a visualização volumétrica são mapeados para a placa de vídeo. O dado volumétrico é carregado em uma textura 3D de índices, ou seja, o valor de cada amostra deve ser mapeado para um inteiro de 8 *bits*. O campo de vetores normais deve ser mapeado em uma textura 3D de valores RGB e cada componente, após ter sido transformada usando a eq. (4), deve ser convertida para um inteiro de 8 *bits*. A função de transferência, definida pelo usuário, é mapeada em uma tabela com 256 entradas, cada uma contendo um quádrupla RGBA. Esta tabela é definida como uma paleta da textura de índices usando funções da extensão `EXT_paletted_texture`. O vetor da direção da fonte de luz também deve ser transformado usando a eq. (4) e em seguida armazenada em RGB como cor primária. Desta forma os combinadores de registros podem acessar a direção de iluminação para usá-la no cálculo da componente difusa.

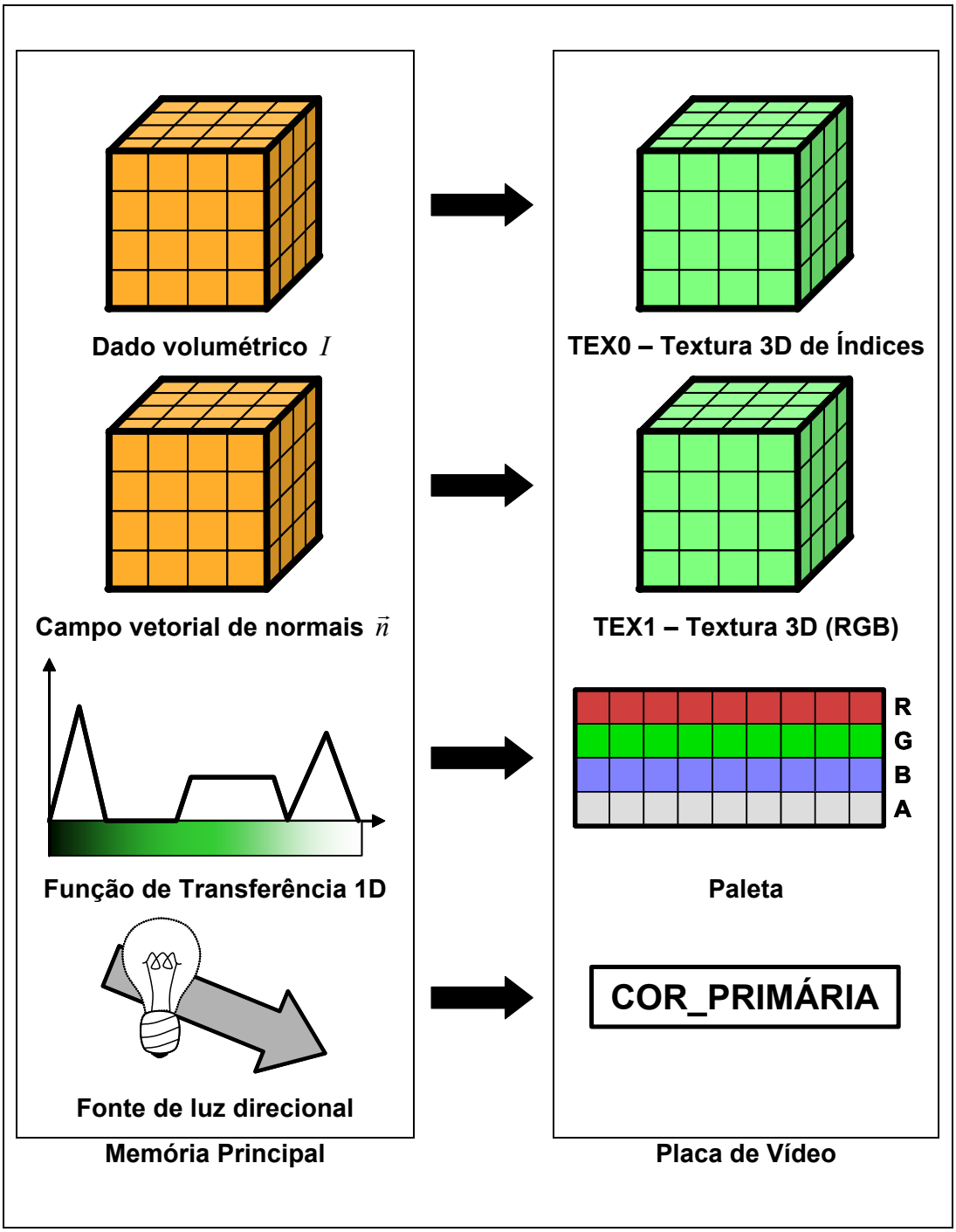


Figura 11 – Mapeamento de dados da memória principal para a placa de vídeo.

O cálculo da iluminação é feito por *pixel* no momento em que cada polígono é desenhado. A Figura 12 apresenta um esboço do processamento que é feito pela placa de vídeo durante a visualização. A geração automática de coordenadas de textura determina para cada vértice do polígono uma tripla (s_i, t_i, r_i) de coordenadas. Durante a rasterização, para cada *pixel* do polígono é calculada uma tripla de coordenadas de textura (s, t, r) interpolando as

coordenadas de textura dos vértices. As coordenadas de textura interpoladas definem um *texel* em cada unidade de textura.

No caso da Figura 12 temos duas unidades de textura. O *texel* da unidade de textura TEX0 determinado pelas coordenadas de textura (s,t,r) é um índice que, por sua vez, determina uma entrada RGBA na tabela da função de transferência. Essa entrada RGBA contendo a cor e a opacidade da amostra é então mandada para o combinador final. Na unidade de textura TEX1, o *texel* determinado pelas coordenadas de textura (s,t,r) contém uma tripla RGB que armazena o vetor normal da amostra. Dentro do combinador de registros, o vetor normal e a cor primária são convertidos de valores RGB novamente para coordenas e é calculado o produto interno entre eles. O resultado do produto interno é armazenado em cada componente de outra tripla RGB que também segue para o combinador final.

No combinador final, a cor vinda da função de transferência (paleta) é multiplicada pelo resultado do produto interno vindo do combinador de registros. Esta multiplicação é feita coordenada por coordenada, por isso o combinador de registros repete o valor do produto interno nas três componentes da sua saída RGB. O combinador final também pode ser usado para adicionar a componente de cor ambiente do modelo de Phong. O combinador final não altera o valor da opacidade da amostra. Finalmente, a saída RGBA do combinador final é combinada com o RGBA que está no *Frame Buffer* pelo operador de composição (*over*).

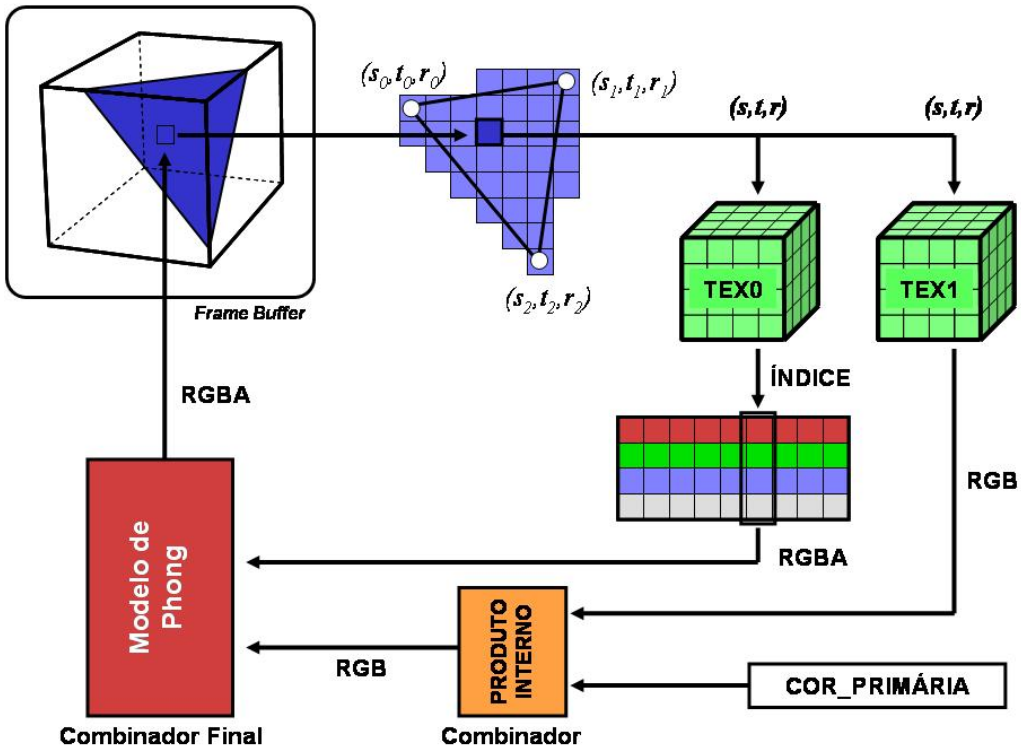


Figura 12 – Cálculo de iluminação feito por *pixel* durante a visualização.

2.2.8. Funções de transferência multidimensionais

As funções de transferência multidimensionais são usadas em dados volumétricos em que temos mais de um valor por amostra. Estes dados podem ser campos vetoriais propriamente ditos, uma reunião de diferentes medidas feitas nas amostras ou uma reunião de diferentes grandezas calculadas nas amostras. A dimensão da função de transferência é igual ao número de valores por amostra. O uso de funções de transferência multidimensionais pode aumentar a capacidade do usuário de selecionar a porção visível do dado, porém a complexidade da tarefa de especificar uma função de transferência de dimensão alta pode tornar sua aplicação inviável (Kindlmann, 1999).

O modelo de visualização volumétrica descrito por Levoy (1988) já incluía uma função de transferência 2D. Nesse modelo cada amostra possui dois valores: o próprio valor do campo escalar como primeira dimensão e o módulo do gradiente como segunda dimensão. Essa abordagem usa uma função de transferência fixa que serve para realçar as bordas entre regiões homogêneas do dado. Extensões para essa abordagem são apresentadas nos trabalhos de Kindlmann e Durkin (1998) e Kniss (2002), nas quais o usuário pode manipular a função de transferência. Na Figura 13 e na Figura 14 temos a aplicação da

função de transferência 2D a um dado de tomografia computadorizada de um dente. Na Figura 13a vemos a distribuição das amostras no plano definido pelos valores do campo escalar e pelo módulo do gradiente do campo. Cada *pixel* da imagem corresponde a um par de valores do dado. Com uma ferramenta similar a um sistema de edição de imagens, o usuário pode definir cores para os pares de valores, como na Figura 13b. Nesta imagem cada região colorida representa os *voxels* da fronteira entre duas regiões homogêneas do volume.

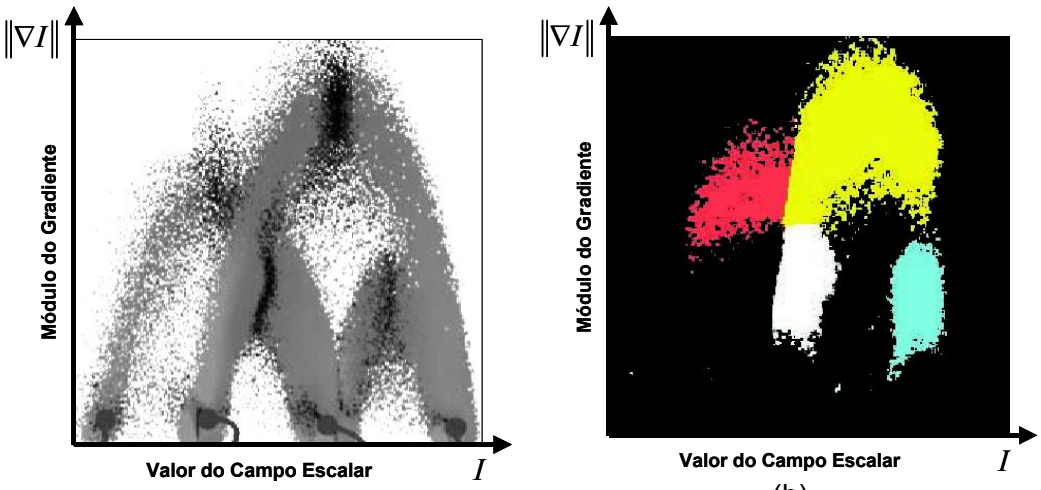


Figura 13 – Função de Transferência Bidimensional. Adaptada de Pfister et al (2001)

Na Figura 14 temos o resultado da visualização volumétrica com função de transferência 2D aplicada ao dado do dente. A função de cor utilizada foi a da Figura 13b. No canto superior esquerdo de cada imagem temos a opacidade 2D utilizada. Os tons de cinza representam a opacidade da amostra, sendo branco para mais opaco e preto para menos opaco. Podemos ver que a função de transferência bidimensional torna possível o realce das fronteiras entre as regiões homogêneas do campo escalar.

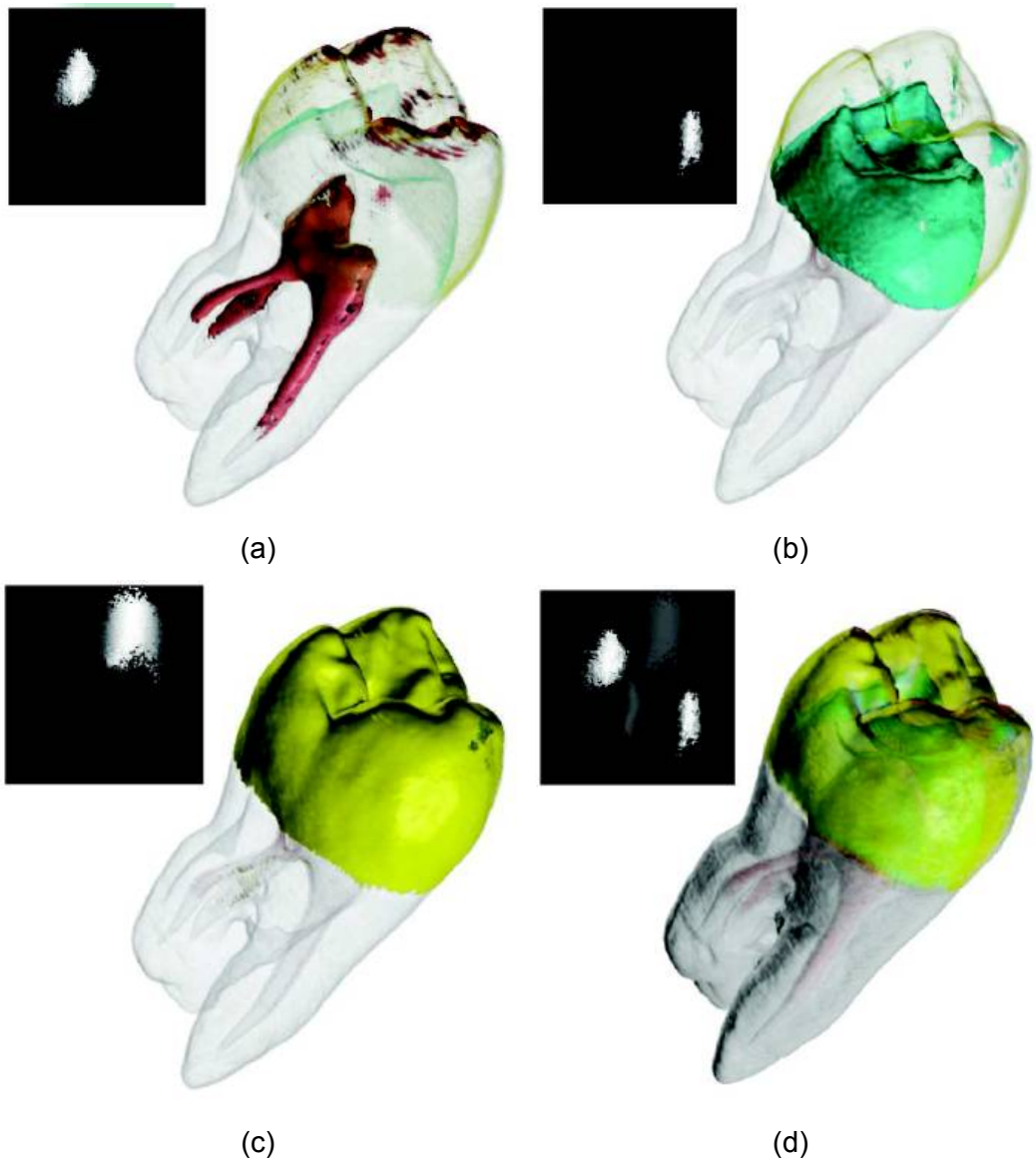


Figura 14 – Realce de fronteiras. Adaptado de Pfister et al (2001)

A função de transferência bidimensional pode ser implementada utilizando o recurso de textura dependente disponível nas placas de vídeo da NVIDIA (Spitzer, 2001) ou fazendo uso da possibilidade de programação das placas de vídeo mais modernas. A Figura 15 mostra como é feito o mapeamento dos dados da memória principal para a placa de vídeo na implementação da função de transferência 2D. Comparando a Figura 15 com a Figura 11, vemos o que muda da implementação da função de transferência 1D para a 2D. O dado volumétrico e o módulo do gradiente do dado são mapeados para uma única textura 3D de valores RGB, o dado volumétrico para o canal **G** e o módulo do gradiente para o canal **B**, o canal **R** não é utilizado. O campo de normais e a direção da iluminação são mapeados da mesma forma para RGB. A função de transferência 2D é mapeada para uma textura 2D de valores RGBA.

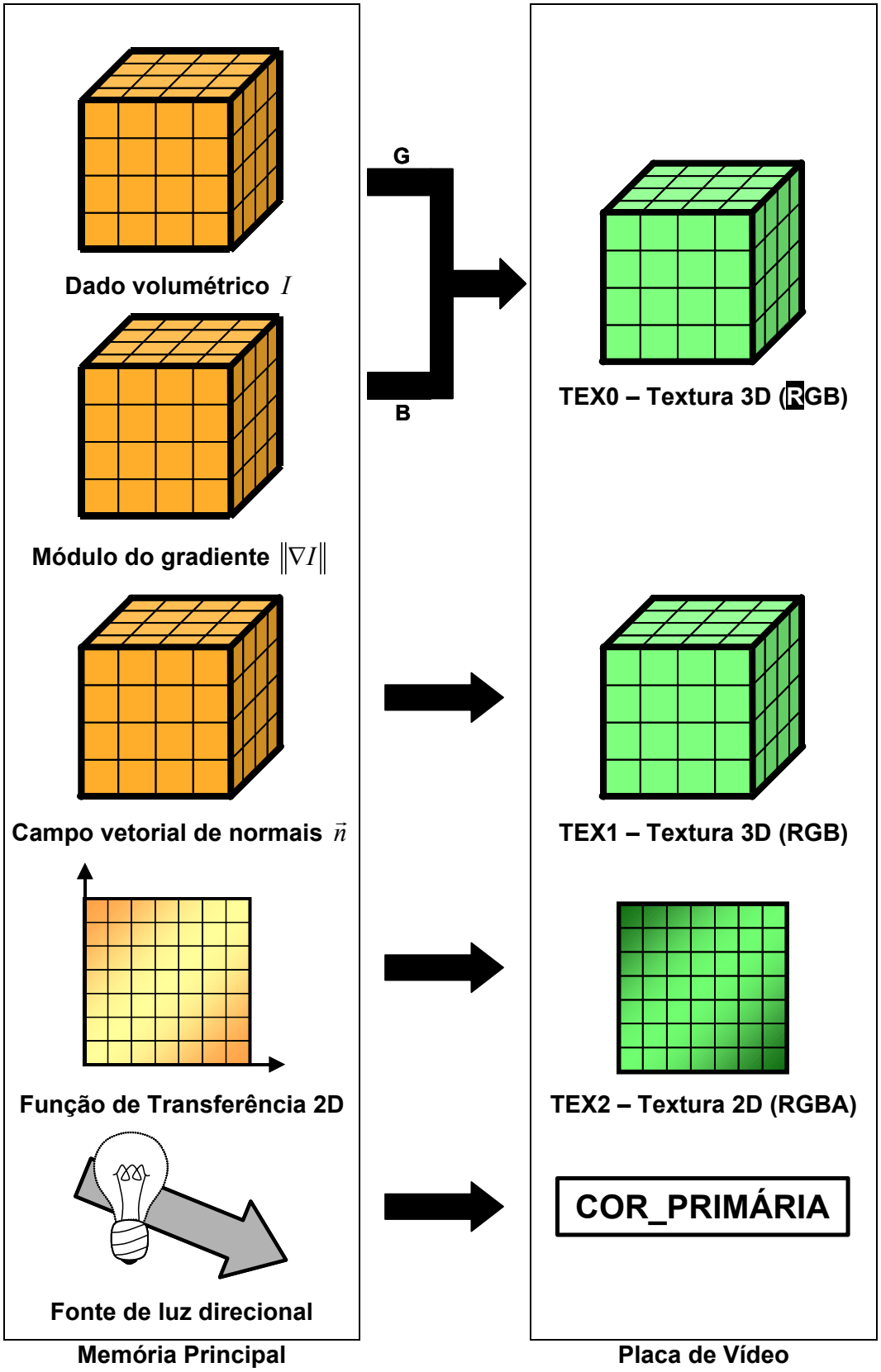


Figura 15 – Mapeamento de dados. Função de transferência 2D.

Na nossa implementação da função de transferência 2D utilizamos três unidades de textura. A tabela da função de transferência 1D é substituída por uma textura dependente. O recurso de textura dependente, disponível nas placas de vídeo mais recentes da NVIDIA, permite que os canais de cor de um *texel* de uma textura sejam utilizados como índices para a textura dependente. Na nossa implementação a unidade de textura TEX2, contendo a função de transferência, é dependente da unidade de textura TEX0, que contém o dado e o módulo do gradiente. Portanto, os canais **G** e **B** da unidade TEX0 são usados como índices na unidade dependente TEX2, ou seja, a cor e a opacidade da amostra dependem tanto do valor da amostra quanto do módulo do gradiente na mesma amostra.

Comparando a Figura 16 com a Figura 12, vemos que na implementação da função de transferência 2D o cálculo da iluminação é o mesmo. Porém, a maneira de determinar a cor e a opacidade da amostra é diferente. A unidade de textura TEX0, que era uma textura 3D de índices, passa a ser uma textura 3D de valores RGB. As coordenadas de textura (s,t,r) definem o valor RGB do *texel* correspondente. Os canais **G** e **B** são convertidos em coordenadas de textura (s,t), que são usadas para determinar um *texel* na unidade de textura 2D dependente TEX2. O valor RGBA correspondendo ao *texel* da unidade de textura TEX2 representa a cor e a opacidade da amostra que segue para o combinador final.

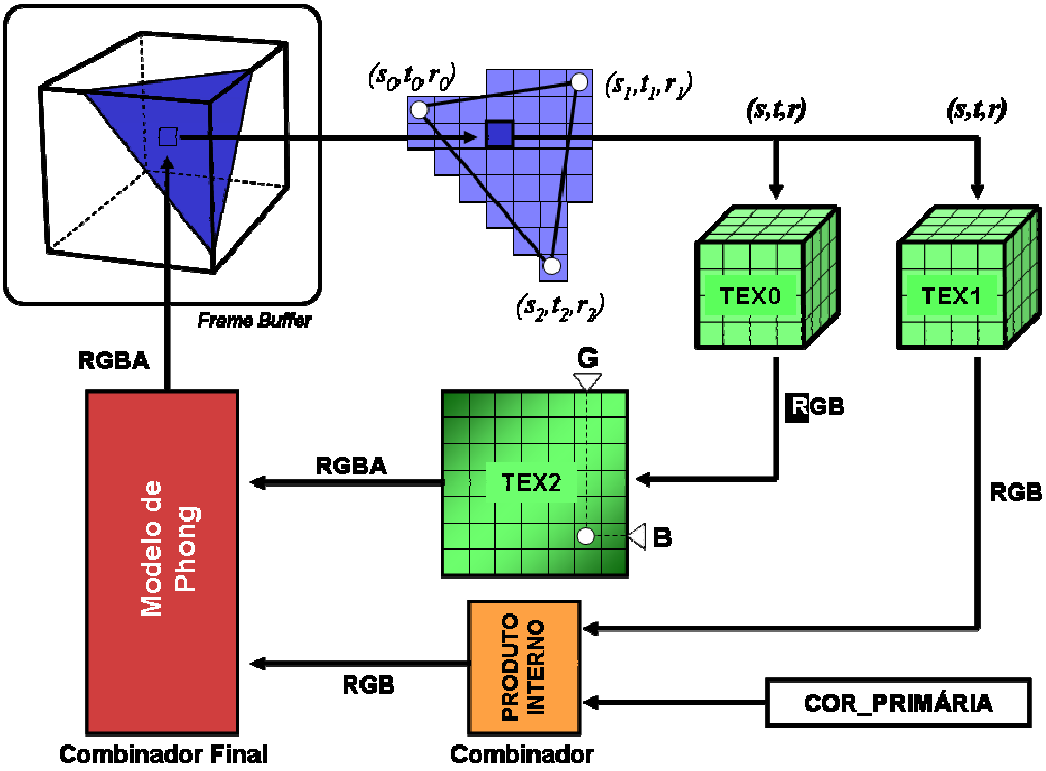


Figura 16 – Função de transferência 2D.