5
Visualização

A representação discretizada de um objeto gráfico $\mathcal{O}$ é definida como $\mathcal{O}'$. Existem várias aplicações onde é necessário obter novamente o objeto gráfico contínuo $\mathcal{O}$ a partir de sua representação $\mathcal{O}'$. Esta operação é chamada de *reconstrução* [21].

Para visualizar os objetos gráficos nos dispositivos de saída é necessário reconstruí-los. Assim, o processo de visualização é visto como uma operação de reconstrução. Neste capítulo, será discutido como visualizar os objetos gráficos 2D a partir da representação utilizada neste trabalho. Da mesma forma que no capítulo anterior, os nomes dos algoritmos de visualização são apresentados em inglês seguindo o estilo de programação.

5.1
Visualização de Objetos Gráficos 2D

A visualização de um objeto gráfico em multi-resolução é realizada em duas fases. A primeira fase consiste em determinar a região do objeto gráfico que está sendo visualizada, essa região é chamada de *janela de visualização*. A outra consiste em determinar quais os níveis de resolução dessa região devem ser passados para o sistema de visualização. Esse cálculo é feito de acordo com o estado dos parâmetros da câmera virtual.

Dados os parâmetros da câmera, calcula-se a interseção entre o cone de visão com os níveis de resolução da estrutura em multi-resolução geométrica. A partir disso, obtém-se todos os ladrilhos de todos os níveis que estão sendo visualizados pela câmera virtual. Em seguida, escolhe-se apenas os ladrilhos que satisfazem algum tipo de critério. Esse critério pode ser definido em relação à representação do suporte geométrico ou em relação à representação dos atributos de textura do objeto gráfico. Para cada um dos casos, o critério é definido da seguinte forma:

– O critério baseado na representação do suporte geométrico consiste em dado um número máximo de ladrilhos, que devem ser passados

para o sistema de visualização, determinar quais ladrilhos devem ser escolhidos de tal forma que a diferença visual entre a superfície formada por esses ladrilhos e a superfície formada pelos ladrilhos do nível mais alto de resolução seja mínima. Em geral, os dados de terreno são visualizados utilizando esse tipo de critério.

– O critério baseado na representação dos atributos de textura consiste em determinar quais ladrilhos devem ser escolhidos de tal forma que a soma do número de pixels das texturas dos ladrilhos seja próxima do número de pixels da tela e que a diferença visual entre a textura reconstruída a partir desses ladrilhos e a textura reconstruída a partir dos ladrilhos do maior resolução seja mínima. A visualização de imagens de satélite e panoramas virtuais são exemplos do uso desse tipo de critério.

Observe que de uma forma geral, o critério de escolher os ladrilhos adequados está intimamente ligado com o problema de minimizar uma função de erro obedecendo um conjunto de restrições. Nas próximas seções serão mostradas as implementações do processo de visualização de dois tipos de objetos gráficos representados em multi-resolução. Esses objetos são as imagens de satélite e panoramas virtuais.

5.2 Imagens de Satélite

O modelo de câmera utilizado para imagens de satélite é bastante simples e possui as seguintes características:

– O tipo de projeção utilizado é o ortogonal.

– O centro de projeção da câmera só possui dois graus de liberdade e está num plano paralelo ao plano da imagem de satélite.

– A direção de visualização é sempre perpendicular ao plano da imagem de satélite.

– O vetor vertical é fixo. Em geral, é paralelo a um dos vetores que definem o plano da imagem de satélite.

– O centro de projeção é projetado no centro da tela.

– Os planos perto e longe são paralelos ao plano da imagem de satélite.

Os parâmetros da câmera serão definidos supondo que o plano da imagem de satélite é definido no plano $z = 0$ do espaço. Assim, o centro de

projeção da câmera virtual é definido como $c = (x, y, 1)$, ou seja, está no plano $z = 1$. A direção de visualização é definida como $v_e = (0, 0, -1)$ e o vetor vertical é dado por $v_{up} = (0, 1, 0)$. O plano perto e o plano longe são definidos por dois pontos. O plano perto é definido pelos pontos $Near_{left} = c - v_{zoom}$ e $Near_{right} = c + v_{zoom}$. O plano longe é definido por $Far_{left} = (x, y, -1) - v_{zoom}$ e $Far_{right} = (x, y, -1) + v_{zoom}$, onde $v_{zoom} = (K_v, K_h, 0)$, $K_v, K_h \in \mathbb{R}^+$. A Figura 5.1 ilustra o modelo de câmera virtual é utilizado para visualizar uma imagem de satélite.

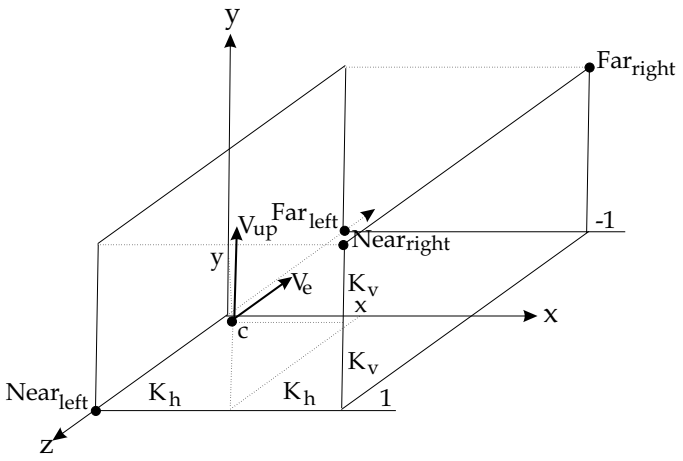

Figura 5.1: O modelo de câmera virtual utilizado para visualizar imagens de satélite.

Note que esse modelo de câmera permite realizar as seguintes operações de visualização:

- **Operação de translação**: Essa operação é realizada especificando os valores de x e y do centro de projeção da câmera. Assim, o feito dessa operação é percebido como a movimentação da câmera sobre a imagem de satélite.
- **Operação de Zoom**: Essa operação é realizada especificando os valores de K_v e K_h. As variações desses parâmetros se refletem em operações de aumento ou de diminuição do volume de visão. Quando o valores de K_h e K_v aumentam o volume de visão também aumenta e uma maior região da imagem de satélite é visualizada. Isso dá um feito de afastamento, ou seja, se realiza um operação de "ZoomOut". No caso inverso, quando o volume de visão diminui uma menor região da imagem de satélite é visualizada. Isso dá um feito de aproximação, ou seja, se realiza uma operação de "ZooIn". Para evitar distorções na projeção da imagem de satélite é necessário que a razão entre os

valores K_v e K_h seja a mesma que a razão entre as dimensões da tela. Assim, dado que (W_t, H_t) sejam as dimensões da tela em pixels, então $K_v = \frac{K_h H_t}{W_t}$. Devido a essa restrição, a operação de zoom é especificada com apenas um parâmetro.

A interseção entre o volume de visão e o suporte geométrico é uma operação simples. A janela de visualização será calculada no sistema de coordenadas do suporte paramétrico. Isto é feito em dois passos. Primeiro, projeta-se os pontos do plano perto ("Near") no suporte geométrico. E o segundo, aplica-se a função inversa que descreve o suporte geométrico nos pontos projetados, veja a Figura 5.2.

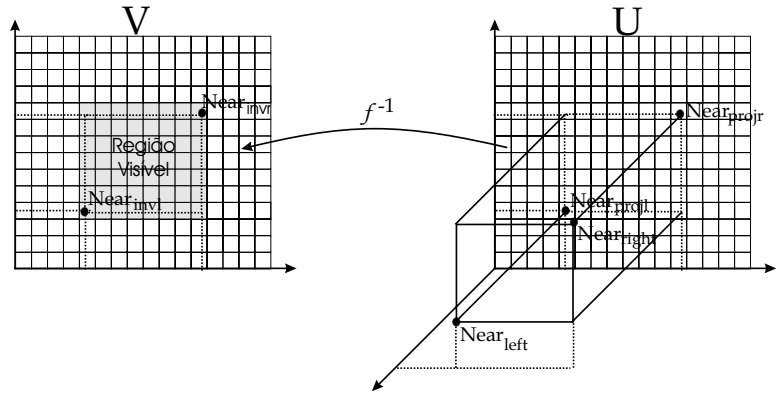

Figura 5.2: Cálculo da área de interseção entre o volume de visão e o suporte geométrico da imagem de satélite. Essa região corresponde área que está sendo visualizada pela câmera virtual.

Como o vetor de direção de visão é sempre perpendicular ao suporte geométrico, então a operação de projeção é realizada tomando $Near_{projl} = (x - k, y - k, 0)$ e $Near_{projr} = (x + k, y + k, 0)$. Suponha que o suporte paramétrico seja definido por $\mathbf{V} = [0, 1] \times [0, 1]$ e que a função paramétrica seja $f : \mathbf{V} \rightarrow \mathbf{U} \subset \mathbb{R}^3$, onde $\mathbf{U} = f(u, v) = (au + b, cv + d, 0)$. Desta forma, a função inversa é dada por $\mathbf{V} = f^{-1}(x, y, z) = (\frac{x-b}{a}, \frac{y-d}{c})$. Aplicando-se a função f^{-1} nos pontos $Near_{projl}$ e $Near_{projr}$, obtém-se dois pontos

$$Near_{invl} = \left(\frac{x-k-b}{a}, \frac{y-k-d}{c}\right) \text{ e } Near_{invr} = \left(\frac{x+k-b}{a}, \frac{y+k-d}{c}\right)$$

que demarcam os limites da região está sendo visualizada.

Neste trabalho, os suportes geométricos das imagens de satélites foram gerados considerando que $b = 0$ e $d = 0$. Para evitar distorções no mapeamento da textura sobre suporte geométrico, os parâmetros a e c foram definidos como $a = 1$ e $c = \frac{h}{w}$, onde w é a largura e h é altura em pixels da

textura. Esta operação faz com que o suporte geométrico tenha a mesma proporção que a textura. Logo, tem-se que

$$Near_{invl} = \left(x - k, \frac{w \cdot (y-k)}{h} \right) \text{ e } Near_{invr} = \left(x + k, \frac{w \cdot (y+k)}{h} \right).$$

A partir do cálculo da região de interseção é possível derteminar quais ladrilhos estão sendo visualizados. Esse processo será feito para cada nível de resolução. Em seguida, serão escolhidos os ladrilhos do nível de resolução que apresentarem melhor razão entre o somatório do número pixels das texturas visualizadas e o número de pixels da tela. O ideal seria escolher um nível onde essa razão fosse de $1:1$. Essa razão será simbolizada pela letra r. Como a região de interseção é retangular e a representação em multi-resolução da imagem de satélite é feita segundo uma escala diádica, então sempre será possível escolher um nível de resolução onde a relação é $0.5 \leq r \leq 2.0$. Observe que se r for maior que 2 escolhe-se um nível de menor resolução que tem $\frac{1}{4}$ do número de pixels. Utilizando esse nível, obtém-se uma nova razão $r > 0.5$. Da mesma forma, se $r < 0.5$, então escolhe-se um nível de maior resolução que possui quatro vezes mais pixels. Logo, a nova razão seria $r < 2.0$. Desta forma, sempre será possível manter uma razão entre 0.5 e 2.0, ou seja, próxima de 1.0.

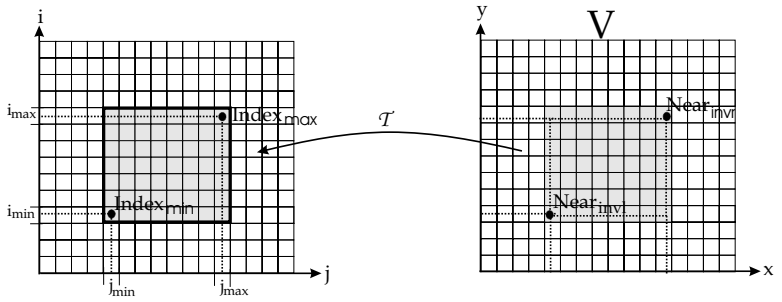

Figura 5.3: Os ladrilhos dentro da área visível são representados por uma sub-matriz. Os índices iniciais ($Index_{min}$) e finais ($Index_{max}$) dessa sub-matriz são obtidos a partir da transformação $T(Near_{invl})$ e $T(Near_{invr})$, respectivamente.

As informações contidas na estrutura de dados `GeoMultiResInfo`, apresentada na Seção 4.4, serão utilizadas para determinar os ladrilhos que são visualizados em cada nível de resolução. Como a região de inserção é retangular, então os ladrilhos que estão dentro dessa região representam uma sub-matriz dentro da matrix que representa o nível de resolução como um todo, veja a Figura 5.3. Os índices inicial e final dessa sub-matriz são representados por $Index_{mim} = (i_{mim}, j_{min}, k)$ e $Index_{max} = (i_{max}, j_{max}, k)$,

respectivamente. As coordenadas (i_{mim}, j_{min}) e (i_{max}, j_{max}) são obtidas a partir de uma transformação aplicada nas coordenadas $Near_{invl}$ e $Near_{invr}$. Essa transformação é definida por $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, onde

$$T(x, y, l) = \left(\frac{2^l \cdot (x - MinBox.x)}{GeoTileW}, \frac{2^l \cdot (y - MinBox.y)}{GeoTileH}, l \right).$$

Lembre-se que $MinBox$ contém as coordenadas mínimas do suporte paramétrico, `GeoTileW` e `GeoTileH` representam as dimensões de um ladrilho do primeiro nível de resolução em relação ao suporte geométrico. As coordenas (x, y) são definidas em relação ao sistema de coordenadas do suporte paramétrico e a coordenada l representa o nível de resolução onde se deseja obter os ladrilhos visíveis. A partir da transformação T, defini-se que $Index_{mim} = \lfloor T(Near_{invl}, k) \rfloor$ e que $Index_{max} = \lceil T(Near_{invr}, k) \rceil$.

Uma vez calculada a sub-matriz visível para cada nível de resolução, agora é necessário escolher uma dessas matrizes e passá-la para o sistema de visualização. Isto é feito calculando a razão r para cada sub-matriz. Suponha que a dimensão de cada sub-matriz seja (W_k, H_k) e que o ladrilhos são projetados numa região retangular da tela de dimensão (W_t, H_t) em pixels. Então a razão R_k é definida por

$$R_k = \frac{TexTileH \cdot TexTileW \cdot W_k \cdot H_k}{W_t H_t},$$

`TexTileH` e `TexTileW` são as dimensões em pixel da textura armazenada em cada ladrilho. Essa informação está na estrutura `TexMultiResInfo`. As dimensões das sub-matrizes é obtida a partir da diferença entre os índices $Index_{max}$ e $Index_{mim}$. O nível k que satisfaz a relação $0.5 \leq R_k \leq 2.0$ é o escolhido e enviado para o sistema de visualização.

O pseudo-código 5 mostra a implementação do processo de visualização de uma imagem de satélite. A funcão `ViewSatelliteImage` chama duas outras funções. A primeira determina o nível de resolução que deve ser visualizado (veja o pseudo-código 6). E a outra, determina quais ladrilhos devem ser passados para o sistema de visualização (pseudo-código 7).

Observe que primeiro foi obtido o nível de resolução. Em seguida, calcula-se a sub-matriz de ladrilhos visíveis para esse nível. O "truque" consiste em calcular a razão R_k sempre considerando uma matriz de ladrilhos capaz de preencher toda a tela. Isto é feito independentemente do que está sendo visualizado. Essa simplificação evita calcular a região da tela onde os ladrilhos serão projetados. Essa região deve ser calculada no

caso onde os ladrilhos não preencherem toda a tela. Essa simplificação torna o processo de visualização mais rápido.

Algoritmo 5 ViewSatelliteImage (**MultiRes** $\mathcal{MR}_{geo}$, **MultiRes** $\mathcal{MR}_{tex}$, **SatCamModel** $SatCam$, $ScreenW$, $ScreenH$)

CalculateLevel($\mathcal{MR}_{geo}$, $\mathcal{MR}_{tex}$, $SatCam$, $ScreenW$, $ScreenH$, $DrawLevel$)
CalculateIndex($\mathcal{MR}_{geo}$, $SatCam$, $DrawLevel$, $Index_{min}$, $Index_{max}$)
for ($i = Index_{min}.i$; $i \leq Index_{max}.i$; $i = i + 1$) **do**
  for ($j = Index_{min}.j$; $j \leq Index_{max}.j$; $j = i + 1$) **do**
    DrawTile ($\mathcal{MR}_{geo}.Level[DrawLevel, i, j].\mathcal{T}.GeoPointer$, $\mathcal{MR}_{geo}.Level[DrawLevel, i, j].\mathcal{T}.MapTexture[0]$)
  end for
end for
end function

Algoritmo 6 CalculateLevel (**MultiRes** $\mathcal{MR}_{geo}$, **MultiRes** $\mathcal{MR}_{tex}$, **SatCamModel** $SatCam$, $ScreenW$, $ScreenH$, $DrawLevel$)

$k = 0$;
$Near_{invl}$=$f^{-1}(SatCam.c.x - SatCam.k.x, SatCam.c.y - SatCam.k.y, 0)$
$Near_{invr}$=$f^{-1}(SatCam.c.x + SatCam.k.x, SatCam.c.y + SatCam.k.y, 0)$
$FrustumSize$=$|Near_{invr} - Near_{invl}|$
W_k=$FrustumSize.x/\mathcal{MR}_{geo}.GeoTileW$
H_k=$FrustumSize.y/\mathcal{MR}_{geo}.GeoTileH$
$NumOfTilePixels$=$\mathcal{MR}_{tex}.TexTileW * \mathcal{MR}_{tex}.TexTileH * W_k * H_k$
$NumOfScreenPixels$=$ScreenW * ScreenH$;
R_k=$NumOfTilePixels/NumOfScreenPixels$
if $R_k \leq 0.5$ **then**
  $DrawLevel$=k
else
  while $k < (\mathcal{MR}_{geo}.Level.nLevels - 1)$ **and** $(R_k > 2.0)$ **do**
    $k = k + 1$
    W_k=$2^k * FrustumSize.x/\mathcal{MR}_{geo}.GeoTileW$
    H_k=$2^k * FrustumSize.y/\mathcal{MR}_{geo}.GeoTileH$
    $NumOfTilePixels$=$\mathcal{MR}_{tex}.TexTileW * \mathcal{MR}_{tex}.TexTileH * W_k * H_k$
    R_k=$NumOfTilePixels/NumOfScreenPixels$
  end while
  $DrawLevel$=k
end if
end function

Algoritmo 7 CalculateIndex(**MultiRes** $\mathcal{MR}_{geo}$, **SatCamModel** $SatCam$, $DrawLevel$, $Index_{min}$, $Index_{max}$)

$Near_{invl}$=$f^{-1}(SatCam.c.x - SatCam.k.x, SatCam.c.y - SatCam.k.y, 0)$
$Near_{invr}$=$f^{-1}(SatCam.c.x + SatCam.k.x, SatCam.c.y + SatCam.k.y, 0)$
$Index_{min}.j$=$\lfloor 2^{DrawLevel} * (Near_{invl}.x - \mathcal{MR}_{geo}.MinBox.x)/\mathcal{MR}_{geo}.GeoTileW \rfloor$
$Index_{min}.i$=$\lfloor 2^{DrawLevel} * (Near_{invl}.y - \mathcal{MR}_{geo}.MinBox.y)/\mathcal{MR}_{geo}.GeoTileH \rfloor$
$Index_{max}.j$=$\lceil 2^{DrawLevel} * (Near_{invr}.x - \mathcal{MR}_{geo}.MinBox.x)/\mathcal{MR}_{geo}.GeoTileW \rceil$
$Index_{max}.i$=$\lceil 2^{DrawLevel} * (Near_{invr}.y - \mathcal{MR}_{geo}.MinBox.y)/\mathcal{MR}_{geo}.GeoTileH \rceil$
——Operações de Clipping————-
$nTilesW$=$2^{DrawLevel} * \mathcal{MR}_{geo}.nTilesW$
$nTilesH$=$2^{DrawLevel} * \mathcal{MR}_{geo}.nTilesH$
if $Index_{min}.j < 0$ **then**
  $Index_{min}.j = 0$
else if $Index_{min}.j > nTilesW$ **then**
  $Index_{min}.j = nTilesW - 1$
end if
if $Index_{min}.i < 0$ **then**
  $Index_{min}.i = 0$
else if $Index_{min}.i > nTilesH$ **then**
  $Index_{min}.i = nTilesH - 1$
end if
if $Index_{max}.j < 0$ **then**
  $Index_{max}.j = 0$
else if $Index_{max}.j > nTilesW$ **then**
  $Index_{max}.j = nTilesW - 1$
end if
if $Index_{min}.i < 0$ **then**
  $Index_{max}.i = 0$
else if $Index_{max}.i > nTilesH$ **then**
  $Index_{max}.i = nTilesH - 1$
end if
end function

5.3 Panoramas Virtuais

Para visualizar os panoramas virtuais é utilizado um modelo de câmera virtual que possui as seguintes características:

- O tipo de projeção utilizado é a projeção perspectiva.
- O centro de projeção é fixo e colocado no centro do cilindro.
- A direção de visualização é especificada a partir de coordenadas esféricas.
- O vetor vertical é fixo e paralelo ao eixo do cilindro.
- O centro de projeção é projetado no centro da tela.

– O volume de visão é especificado a partir dos ângulos de abertura vertical e horizonal.

– O plano perto ("Near") é sempre perpendicular ao vetor de direção e colocado a uma certa distância do centro de projeção.

Os parâmetros fixos da câmera serão definidos baseado no fato de que o cilindro é definido em torno do $eixoZ$ e centrado na origem. Suponha que a função paramétrica $f : \mathbf{V} \rightarrow \mathbf{U}$ que descreve o cilindro seja definida por $f(u, v) = (r\cos(u), r\sin(u), v)$, $\mathbf{V} = [0, 2\pi] \times [-h, h]$. Então o centro de projeção é definido na origem, ou seja, $c = (0, 0, 0)$ e o vetor vertical $v_{up} = (0, 0, 1)$. O único parâmetro que não depende da parametrização da superfície panorâmica é a distância do plano perto ao centro de projeção. Essa distância é determinada a partir de informações sobre o tipo de lente que é utilizado para capturar a imagem panorâmica.

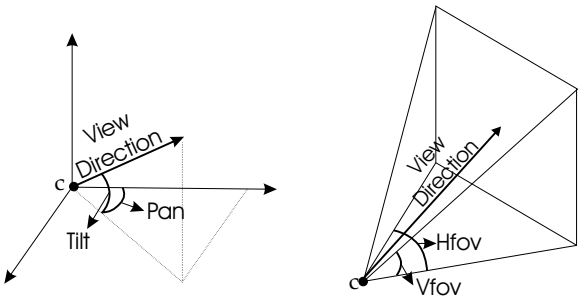

Figura 5.4: Modelo de camera utilizado para a visualização de panoramas virtuais.

Os únicos parâmetros da câmera que podem variar são o vetor de direção de visão e os ângulos de abertura. O vetor de direção de visão é especificado a partir do ângulo de inclinação vertical (ϕ) e do o ângulo de rotação horizontal (θ). O ângulo de inclinação vertical é chamada de *ângulo de tilt* e o ângulo de rotação horizontal é chamado de *ângulo de pan*. A operação de movimento de câmera no interior do cilindro é realizada a partir do controle desses dois ângulos. Os ângulos de aberturas são especificados por dois parâmetros: o ângulo de abertura horizontal ($hFOV$) e o ângulo de abertura vertical ($vFOV$). Esses ângulos controlam o tamanho do campo de visão da câmera virtual. A especificação desses ângulos resulta na operação de zoom. A operação de aproximação ("ZoomIn") é realizada a partir da diminuição dos ângulos de abertura e a operação de afastamento ("ZoomOut") é realizada a partir do seu aumento. A Figura 5.4 ilustra como o modelo de câmera virtual utilizado para visualizar panoramas.

Para evitar distorções no mapeamento da imagem panorâmica sobre a superfície é necessário que a razão entre o comprimento vertical e horizontal do cilindro seja igual razão entre a largura e a altura da imagem panorâmica. Neste trabalho, o suporte paramétrico do panorama virtual foi definido como $\mathbf{V} = [0, 2\pi] \times [-1, 1]$. Supondo que as dimensões da imagem panorâmica em pixels seja (W_p, H_p), então o raio do cilindro deve ser definido por $r = \frac{W_p}{H_p \pi}$ para evitar a distorção. Outro problema que também deve ser evitado são as distorções da projeção do cilindro na tela. Esse problema é resolvido mantendo a razão entre os ângulos de abertura sempre igual a razão entre as dimensões da tela. Ou seja, conhecido o ângulo de abertura vertical $vFOV$ e as dimensões da tela (W_t, H_t) em pixels, então ângulo de abertura horizontal é dado por $hFOV = \frac{W_t vFOV}{H_t}$. Observe que graças a essa restrição, a operação de zoom agora é especificada com apenas um parâmetro.

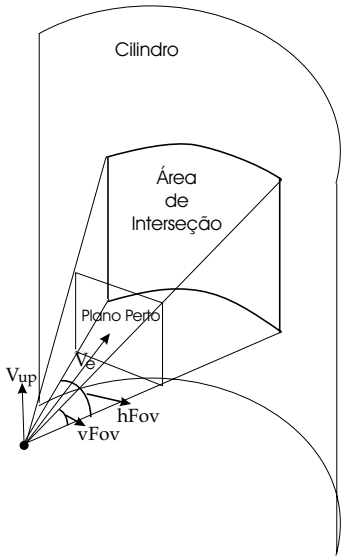

Figura 5.5: Região formada pela interseção entre o volume de visão e a superfície panorâmica.

Em geral, a região de interseção entre o volume de visão e a superfície panorâmica possui uma geometria complexa. Para simplificar os cálculos, essa região é aproximada por uma região retangular envolvente, como mostra a Figura 5.5. O cálculo da janela de visualização será feito da mesma forma como foi feito nas imagens de satélite. Isto é, as coordenadas da janela de visualização serão calculadas em relação ao sistema de coordenadas do suporte paramétrico do cilindro.

A região de inserção do volume de visão com o cilindro é especificada a partir de seis pontos. Esses pontos são ilustrados na Figura 5.6. As curvas superior e inferior, que delimitam a região de interseção, são quadráticas

no cilindro, logo são especificadas a partir de três pontos. Esses pontos são facilmente obtidos a partir dos parâmetros da câmera virtual, como é indicado na Figura 5.6.

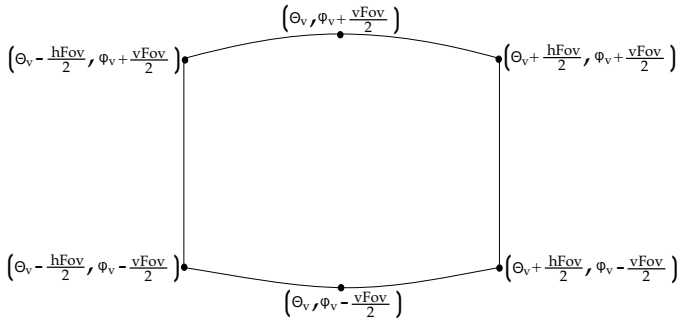

Figura 5.6: A região de interseção é determinada a partir de seis pontos. As coordenadas desses pontos são obtidas a partir dos parâmetros da camera virtual.

Após encontrar os seis pontos, utiliza-se uma transformação do sistema de coordenada esférico para sistema de coordenada do suporte paramétrico do cilindro. A transformação $H : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ é definida da seguinte forma: Sejam (θ_v, ϕ_v) o ângulo da direção de visão e um ponto p qualquer que pertença a curva de interseção entre um dos planos do volume de visão e o cilindro. Suponha que esse ponto faça um ângulo θ_p com o plano XY, onde $\theta_p \neq \theta_v$, e que o plano tenha uma inclinação ϕ_0 (Figura 5.7). Essa inclinação ϕ_0 pode ser $\phi_0 = \phi_v + \frac{vFOV}{2}$ ou $\phi_0 = \phi_v - \frac{vFOV}{2}$. Observe que $\tan(\phi_0) = \frac{h_p}{r\cos(\theta_p - \theta_v)}$, h_p é a altura do ponto p no domínio paramétrico. Daí tem-se que a transformação H é dada por

$$H(\theta, \phi) = (\theta, r\tan(\phi_0)\cos(\theta - \theta_v)).$$

Com a transformação H obtém-se a região retangular no suporte paramétrico que representa o que está sendo visualizado pela câmera. Essa região é determinada pelas coordenadas mínimas (θ_{min}, h_{min}) e máximas (θ_{max}, h_{max}) (Figura 5.7). Observe que as coordenadas mínimas e máximas são definidas por

$$(\theta_{min}, h_{min}) = \left(\theta_v - \frac{hFOV}{2}, r\tan(\phi_v - \frac{vFOV}{2}\right) \text{ e}$$

$$(\theta_{max}, h_{max}) = \left(\theta_v + \frac{hFOV}{2}, r\tan(\phi_v + \frac{vFOV}{2}\right).$$

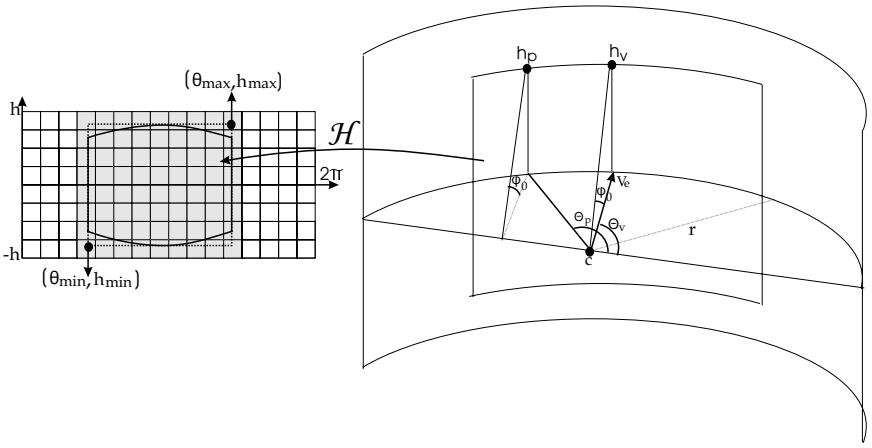

Figura 5.7: A transformação H converte as coordenadas dos pontos que delimitam a região de interseção em coordenadas do suporte paramétrico. A região pintada representa a sub-matriz de ladrilhos que estão dentro o campo de visão da câmera.

Da mesma forma que na imagem de satélite, os ladrilhos que estão dentro da região retangular formam uma sub-matriz dentro de cada nível de resolução. Assim, basta utilizar a transformação T, que foi mostrada na seção anterior, para se determinar os índices mínimos e máximos da sub-matriz visível para cada nível de resolução. Substituindo as coordenadas mínimas (θ_{min}, h_{min}) e máximas (θ_{max}, h_{max}) na transformação T obtem-se os índices mínimos e máximos das sub-matrizes, respectivamente. Fazendo uma analogia com o processo de visualização das imagens de satélite, tem-se que $Near_{projl} = (\theta_v, \phi_v - \frac{vFOV}{2})$, $Near_{projr} = (\theta_v, \phi_v + \frac{vFOV}{2})$, $Near_{invl} = H(Near_{projl})$ e $Near_{invr} = H(Near_{projr})$. Assim, aplicando a transformação T nos vértices $Near_{invl}$ e $Near_{invr}$ obtém-se os índices mínimos $(Index_{min})$ e máximos $(Index_{max})$, respectivamente.

A escolha da sub-matriz, que será passada para o sistema de visualização, é feita da mesma forma que na imagem de satélite. Ou seja, a sub-matriz do nível k que satisfizer a relação $0.5 \leq R_k \leq 2.0$ é enviada para o sistema de visualização.

Os pseudo-códigos 8, 9 e 10 mostram a implementação todo o processo de visualização. Observe que as fases de visualização das panoramas virtuais são praticamente as mesmas utilizadas para visualizar as imagens de satélites. As diferenças estão nas funções de transformação que delimitam a área de interseção e na forma de realizar a operação de "clipping". Essas operações estão intimamente relacionadas com a geometria dos objetos gráficos.

Algoritmo 8 ViewPanoramaVirtual (**MultiRes** $\mathcal{MR}_{geo}$, **MultiRes** $\mathcal{MR}_{tex}$, **PanCamModel** $PanCam$, $ScreenW$, $ScreenH$)

CalculateLevel($\mathcal{MR}_{geo}$, $\mathcal{MR}_{tex}$, $PanCam$, $ScreenW$, $ScreenH$, $DrawLevel$)
CalculateIndex($\mathcal{MR}_{geo}$, $PanCam$, $DrawLevel$, $Index_{min}$, $Index_{max}$)
for ($i = Index_{min}.i$; $i \leq Index_{max}.i$; $i = i + 1$) **do**
  for ($j = Index_{min}.j$; $j \leq Index_{max}.j$; $j = i + 1$) **do**
    if $j < 0$ **then**
      $Auxj$=$nTilesW + j$
    else
      $Auxj$=j**mod**$nTilesW$
    end if
    DrawTile ($\mathcal{MR}_{geo}.Level[DrawLevel, i, Auxj].\mathcal{T}.GeoPointer$, $\mathcal{MR}_{geo}.Level[DrawLevel, i, Auxj].\mathcal{T}.MapTexture[0]$)
  end for
end for
end function

Algoritmo 9 CalculateLevel (**MultiRes** $\mathcal{MR}_{geo}$, **MultiRes** $\mathcal{MR}_{tex}$, **PanCamModel** $PanCam$, $ScreenW$, $ScreenH$, $DrawLevel$)

$k = 0$;
$Near_{invl}.x$=$H(PanCam.teta - PanCam.hFov/2, PanCam.fi - PanCam.vFov/2).x$
$Near_{invl}.y$=$H(PanCam.teta, PanCam.fi - PanCam.vFov/2).y$
$Near_{invr}.x$=$H(PanCam.teta + PanCam.hFov/2, PanCam.fi + PanCam.vFov/2).x$
$Near_{invr}.y$=$H(PanCam.teta, PanCam.fi + PanCam.vFov/2).y$
$FrustumSize$=$|Near_{invr} - Near_{invl}|$
W_k=$FrustumSize.x/\mathcal{MR}_{geo}.GeoTileW$
H_k=$FrustumSize.y/\mathcal{MR}_{geo}.GeoTileH$
$NumOfTilePixels$=$\mathcal{MR}_{tex}.TexTileW * \mathcal{MR}_{tex}.TexTileH * W_k * H_k$
$NumOfScreenPixels$=$ScreenW * ScreenH$;
R_k=$NumOfTilePixels/NumOfScreenPixels$
if $R_k \leq 0.5$ **then**
  $DrawLevel$=k
else
  while $k < (\mathcal{MR}_{geo}.Level.nLevels - 1)$ **and** $(R_k > 2.0)$ **do**
    $k = k + 1$
    W_k=$2^k * FrustumSize.x/\mathcal{MR}_{geo}.GeoTileW$
    H_k=$2^k * FrustumSize.y/\mathcal{MR}_{geo}.GeoTileH$
    $NumOfTilePixels$=$\mathcal{MR}_{tex}.TexTileW * \mathcal{MR}_{tex}.TexTileH * W_k * H_k$
    R_k=$NumOfTilePixels/NumOfScreenPixels$
  end while
  $DrawLevel$=k
end if
end function

Algoritmo 10 CalculateIndex(**MultiRes** $\mathcal{MR}_{geo}$, **PanCamModel** $PanCam$, $DrawLevel$, $Index_{min}$, $Index_{max}$)

$Near_{invl}.x$=$H(PanCam.teta - PanCam.hFov/2, PanCam.fi - PanCam.vFov/2).x$
$Near_{invl}.y$=$H(PanCam.teta, PanCam.fi - PanCam.vFov/2).y$
$Near_{invr}.x$=$H(PanCam.teta + PanCam.hFov/2, PanCam.fi + PanCam.vFov/2).x$
$Near_{invr}.y$=$H(PanCam.teta, PanCam.fi + PanCam.vFov/2).y$
$Index_{min}.j$=$\lfloor 2^{DrawLevel} * (Near_{invl}.x - \mathcal{MR}_{geo}.MinBox.x)/\mathcal{MR}_{geo}.GeoTileW \rfloor$
$Index_{min}.i$=$\lfloor 2^{DrawLevel} * (Near_{invl}.y - \mathcal{MR}_{geo}.MinBox.y)/\mathcal{MR}_{geo}.GeoTileH \rfloor$
$Index_{max}.j$=$\lceil 2^{DrawLevel} * (Near_{invr}.x - \mathcal{MR}_{geo}.MinBox.x)/\mathcal{MR}_{geo}.GeoTileW \rceil$
$Index_{max}.i$=$\lceil 2^{DrawLevel} * (Near_{invr}.y - \mathcal{MR}_{geo}.MinBox.y)/\mathcal{MR}_{geo}.GeoTileH \rceil$
——Operações de Clipping———-
$nTilesW$=$2^{DrawLevel} * \mathcal{MR}_{geo}.nTilesW$
$nTilesH$=$2^{DrawLevel} * \mathcal{MR}_{geo}.nTilesH$
if $Index_{min}.i < 0$ **then**
  $Index_{min}.i = 0$
else if $Index_{min}.i > nTilesH$ **then**
  $Index_{min}.i = nTilesH - 1$
end if
if $Index_{min}.i < 0$ **then**
  $Index_{max}.i = 0$
else if $Index_{max}.i > nTilesH$ **then**
  $Index_{max}.i = nTilesH - 1$
end if
end function