MAIKON CISMOSKI DOS SANTOS

Renderização de Cenas Tridimensionais Interativas em Computadores com Recursos Gráficos Limitados

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Hélio Pedrini

CURITIBA

2009

AGRADECIMENTOS

Aos meus pais, pelo apoio e incentivo, sem vocês não teria chegado até aqui.

Ao Hélio Pedrini pela excelente orientação concedida a este trabalho, sugestões, ensinamentos e oportunidades. Estando sempre à disposição para ajudar nos problemas encontrados, dúvidas e definições necessárias, muito obrigado!

Ao André Luiz Battaiola e toda a equipe do projeto EEHouse pelo amparo e concessão de recursos para o desenvolvimento deste trabalho.

Ao Sergio Scheer pelo fornecimento dos recursos computacionais, como o servidor e laboratórios para os testes do protótipo.

Ao Jorge Luis Salvi pela modelagem dos objetos tridimensionais usados neste trabalho.

Ao Bruno Silva de Oliveira pelas inúmeras dicas e sugestões.

À Financiadora de Estudos e Projetos (FINEP) pela ajuda financeira.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

2D	*Two-Dimensional*	Bidimensional
3D	*Three-Dimensional*	Tridimensional
COP	*Center of Projection*	Centro de Projeção
CPU	*Central Processing Unit*	Unidade Central de Processamento
FOV	*Field of View*	Campo de Visão
FPS	*Frames Per Second*	Quadros Por Segundo
GLX	*OpenGL Extension to the X Window System*	Extensão do OpenGL para o Sistema de Janela X
GPU	*Graphics Processing Unit*	Unidade de Processamento Gráfico
HTML	*HyperText Markup Language*	Linguagem de Marcação de Hipertexto
IBR	*Image-Based Rendering*	Renderização Baseada em Imagens
IP	*Internet Protocol*	Protocolo de Internet
JPEG	*Joint Photographic Experts Group*	Grupo de Especialistas em Fotografia
LAN	*Local Area Network*	Rede de Área Local
LDI	*Layered Depth Image*	Imagem de Profundidade em Camadas
LTSP	*Linux Terminal Server Project*	Projeto de Servidor de Terminal Linux
OpenGL	*Open Graphics Library*	Biblioteca OpenGL
PDA	*Personal Digital Assistant*	Assistente Pessoal Digital
RAM	*Random Access Memory*	Memória de Acesso Aleatório
RGB	*Red Green Blue*	Vermelho Verde Azul
TCP	*Transmission Control Protocol*	Protocolo de Controle de Transmissão

UDP	*User Datagram Protocol*	Protocolo de Datagrama do Usuário
VDTM	*View-Dependent Texture Mapping*	Mapeamento de Textura Dependente da Visão
VRML	*Virtual Reality Modeling Language*	Linguagem para Modelagem de Realidade Virtual
X3D	*Extensible 3D*	3D Extensível

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

RESUMO

Este trabalho propõe e implementa quatro abordagens diferentes para renderização remota e sob demanda da cena 3D destinada a computadores com recursos gráficos limitados, que não dispõem de uma unidade de processamento gráfico ou computadores que possuam placas aceleradoras gráficas, mas essas sejam limitadas em termos de processamento. Para aprimorar o processo de renderização, a técnica de Renderização Baseada em Imagens 3D *warping* foi empregada, permitindo que o cliente remoto renderize localmente novas vistas do ambiente virtual, enquanto espera por dados requisitados ao servidor ou até mesmo quando há problemas em sua transmissão ou atraso no recebimento.

O conceito de imagens de resíduo foi utilizado, permitindo que menos dados sejam transmitidos entre servidor e cliente em comparação com os quadros completos renderizados a partir do modelo 3D. Um método para o aproveitamento das imagens de resíduo recebidas pelos clientes foi proposto e implementado, possibilitando que as imagens recebidas pelos clientes sejam utilizadas para o refinamento da qualidade das vistas exibidas, ao invés de serem descartadas. Uma abordagem para o descarte de requisições no servidor foi proposta, buscando reduzir sua sobrecarga devido à grande demanda de requisições feitas pelos clientes.

O desempenho do protótipo foi avaliado com base em dois experimentos realizados, nos quais foram analisados os tempos obtidos na renderização remota e local da cena 3D. Os testes de desempenho contemplaram computadores com capacidade de processamento gráfico limitada.

CAPÍTULO 1

INTRODUÇÃO

A Renderização é a atividade de geração de imagens ou vídeos por meio de um computador [3]. A renderização de cenas tridimensionais (3D) é uma tarefa bem conhecida em diferentes tipos de aplicação, como jogos e ambientes virtuais em geral. A partir de um modelo 3D são produzidas imagens baseadas nas informações tridimensionais deste modelo, como geometria, ponto de vista, texturas e efeitos de iluminação.

O constante avanço tecnológico proporciona o surgimento de unidades de processamento gráfico (GPU, do inglês, *Graphics Processing Unit*) cada vez mais poderosas, capazes de renderizar modelos 3D complexos, contendo uma grande quantidade de polígonos e representando um alto grau de realismo. Entretanto, repartições públicas, tais como escolas, nem sempre dispõem de um parque computacional com esses recursos gráficos, normalmente formado por computadores sem GPU, restringindo a renderização de cenas 3D a ambientes virtuais que contenham poucos detalhes. Nesta dissertação, o termo *recursos gráficos limitados* refere-se somente à capacidade de processamento gráfico limitada, não englobando monitores ou qualquer outro dispositivo.

Um paradigma que pode ser utilizado para tentar resolver esse problema é a renderização remota da cena, a qual consiste em renderizar, sob demanda, o conteúdo 3D em um servidor robusto com grande capacidade de processamento gráfico e transmitir para os clientes as imagens ou vídeo codificado com poucos quadros resultantes do processo de renderização. Quando o cliente recebe as imagens renderizadas ou o vídeo codificado, ele apenas exibe esse conteúdo na tela, deixando que a maior carga de processamento seja realizada no servidor. Quando o observador deseja movimentar-se no ambiente virtual, uma nova renderização relativa ao ponto de vista requerido é solicitado ao servidor.

O problema desse paradigma é sua suscetibilidade à latência de rede e a possíveis falhas que venham a ocorrer na transmissão dos dados do servidor para o cliente, como perda

de pacotes. Sempre quando um pacote enviado sofre um atraso ou é perdido, isso afeta diretamente a taxa de exibição dos quadros (imagens) no cliente, tardando a visualização do conteúdo transmitido pelo servidor.

Técnicas denominadas Renderização Baseada em Imagens, conhecidas pela sigla IBR (do inglês, *Image-Based Rendering*), têm sido muito utilizadas nestes últimos anos como mecanismo para renderização de novas vistas a partir de imagens, ao invés de utilizar as tradicionais primitivas geométricas.

Este trabalho utiliza uma renderização remota da cena e emprega a técnica de IBR 3D *warping* para renderizar vistas localmente no cliente, usando somente o último quadro recebido. Assim, o cliente é capaz de renderizar novas vistas enquanto espera por dados requisitados ao servidor ou até mesmo quando há problemas em sua transmissão ou atraso no recebimento. Além disso, a centralização da capacidade de renderização em um servidor robusto permite que, mesmo os clientes que possuam placas aceleradoras gráficas, mas essas sejam limitadas em termos de processamento, poderão renderizar cenas complexas.

1.1 Objetivos e Contribuições

Este trabalho tem como principal objetivo construir um protótipo para renderização de cenas 3D interativas em computadores com recursos gráficos limitados, que não dispõem de uma GPU para o processamento gráfico ou computadores que possuam placas aceleradoras gráficas, mas essas sejam limitadas em termos de processamento. A tarefa de renderização do ambiente virtual 3D foi feita remotamente, juntamente com o uso da técnica de IBR 3D *warping* para aprimorar esse processo, renderizando as vistas localmente no cliente, buscando manter boas taxas de visualização dos quadros mesmo em casos onde há atraso no recebimento das imagens.

Para demonstrar e comparar o desempenho do protótipo na renderização do ambiente virtual, quatro arquiteturas diferentes foram propostas e implementadas: três arquiteturas foram baseadas na técnica de IBR 3D *warping* e uma baseada no paradigma de renderização remota tradicional, que não emprega nenhuma técnica de IBR. Dessa forma, uma avaliação detalhada do protótipo foi realizada com relação às diferentes formas de

utilização da técnica 3D *warping* na renderização das vistas e uma comparação foi feita entre os processos de renderização da cena com e sem o uso de técnicas IBR.

Um método para o aproveitamento das imagens transmitidas do servidor para o cliente foi proposto e implementado. Quando o cliente recebe imagens referentes a uma movimentação mais antiga do que a atual exibida pelo cliente, essas imagens recebidas não são descartadas e sim aproveitadas para o refinamento da qualidade das vistas exibidas no cliente. Uma abordagem foi proposta para o descarte de requisições no servidor, reduzindo sua sobrecarga devido à grande demanda de requisições feitas pelos clientes. Um estudo foi realizado entre as diferentes técnicas de IBR definindo uma técnica apropriada para auxiliar o processo de renderização da cena.

Apenas pacotes livres e gratuitos foram utilizados no desenvolvimento do protótipo. Na implementação do módulo cliente, utilizou-se de uma linguagem de programação que gera um código portável para múltiplas plataformas, facilitando a distribuição e execução do programa.

1.2 Estrutura da Dissertação

Este trabalho está organizado como segue. O capítulo 2 apresenta as principais técnicas de IBR e trabalhos previamente propostos para renderização e visualização de conteúdos 3D em dispositivos com capacidade de processamento gráfico limitada.

A metodologia de pesquisa é descrita no capítulo 3, onde são apresentadas as diferentes arquiteturas propostas, o método de aproveitamento das imagens recebidas pelos clientes para o refinamento das vistas exibidas, a abordagem utilizada para a redução da sobrecarga no servidor, os detalhes da compressão e transmissão dos dados entre servidor e cliente, as métricas para a avaliação do desempenho e as ferramentas utilizadas na implementação do protótipo.

No capítulo 4 são descritos os resultados obtidos nos experimentos realizados com relação à renderização remota e local do ambiente virtual nas diferentes arquiteturas propostas. Finalmente, as conclusões e trabalhos futuros são apresentados no capítulo 5.

CAPÍTULO 2

TRABALHOS RELACIONADOS

Este capítulo é dividido em duas seções. Na seção 2.1 são apresentadas as principais técnicas de IBR destacando suas vantagens e desvantagens e definindo uma técnica de IBR apropriada para aprimorar o processo de renderização do ambiente virtual. Na seção 2.2 são descritos os principais trabalhos que abordam a renderização de cena 3D em dispositivos com recursos gráficos limitados.

2.1 Renderização Baseada em Imagens

A combinação entre as áreas de visão computacional e computação gráfica proporcionou a criação de um conjunto de técnicas denominadas Renderização Baseada em Imagens (IBR). O surgimento de IBR ocorreu a partir da definição da função plenóptica, detalhada na subseção 2.1.1.

Atualmente, muitos pesquisadores utilizam IBR como mecanismo para renderizar cenas tridimensionais com alto grau de realismo em dispositivos como celulares e PDAs (*Personal Digital Assistant*). Esses tipos de equipamentos, geralmente, possuem baixa capacidade de processamento gráfico.

IBR tem o objetivo de acelerar o processo de renderização, obter uma renderização mais realista e simplificar a tarefa de modelagem [41]. A qualidade das novas vistas renderizadas empregando IBR podem ser fotorrealistas. Esse conjunto de técnicas utiliza-se de imagens (amostras) como primitiva de renderização ao invés da primitiva geométrica tradicional, fazendo com que o custo para renderizar uma vista empregando IBR seja independente da complexidade da cena e condicionado à resolução da imagem.

As amostras podem conter algum tipo de informação geométrica associada. A classificação das técnicas de IBR neste trabalho foi determinada de acordo com a informação geométrica associada em cada uma: *renderização sem geometria* onde somente imagens

são utilizadas para renderizar novas vistas e a *renderização com geometria* que, além das amostras, utiliza algum tipo de informação geométrica como, por exemplo, a informação de profundidade dos pixels.

A seguir, na subseção 2.1.1 são descritos os conceitos da função plenóptica, na subseção 2.1.2 são apresentadas as técnicas de IBR que utilizam somente imagens para renderizar novas vistas, na subseção 2.1.3 são descritas as técnicas de IBR que empregam imagens e algum tipo de informação geométrica na renderização dos quadros e, por fim, na subseção 2.1.4 as técnicas de IBR são discutidas.

2.1.1 Função Plenóptica

A função plenóptica [2] (*plenus* = completa e *optic* = visão) é uma função parametrizada que descreve o que pode ser visto no espaço. Ela registra a intensidade dos raios de luz visíveis que passam pelo centro de projeção da câmera, em qualquer ponto do espaço, orientação, tempo e comprimento de onda da luz.

A função plenóptica consiste em $f : P \rightarrow U$, onde $P \subset \mathbb{R}^7$ e $U \subset \mathbb{N}$, definida em [37] como:

$$\mu = \text{Plenóptica}(\theta, \phi, \lambda, Vx, Vy, Vz, t) \tag{2.1}$$

em que μ é a intensidade de cor resultante da função plenóptica, dados os parâmetros (Vx, Vy, Vz) que definem a posição do observador, a direção em que o observador está focando, composta pelos ângulos de azimute e elevação (θ, ϕ) respectivamente, o comprimento de onda da luz dado por λ e em cenas dinâmicas, t determina o tempo. Os parâmetros da função plenóptica descritos são ilustrados na Figura 2.1.

Esta função pode ser considerada com um conjunto de todos os possíveis mapas de ambiente[1] (*environment maps*) que podem ser definidos para uma cena arbitrária [37], por este motivo, raramente ela é disponível. As imagens que compõem os mapas de ambiente são chamadas de amostras da função plenóptica.

Para a renderização de novas vistas, inicialmente é feita a especificação dos parâmetros

[1] Projeção de um ambiente completo em um cubo através do mapeamento de uma imagem para cada face [25].

da função plenóptica com a definição da posição do observador (Vx, Vy, Vz), o comprimento da onda da luz λ, o tempo t da visualização e o conjunto de valores (θ, ϕ) que correspondem ao campo de visão desejado.

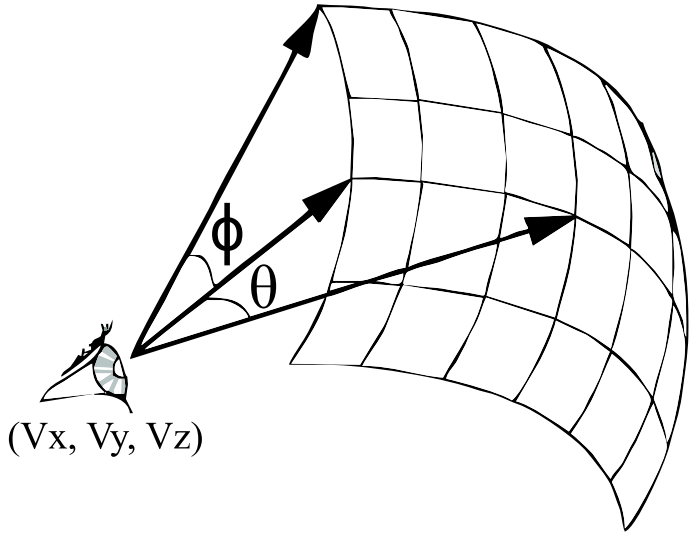


Figura 2.1: Parâmetros da função plenóptica. Fonte [37].

Com os parâmetros especificados, a imagem referente à vista desejada é sintetizada integrando todas as intensidades das cores computadas pela função plenóptica com a discretização das coordenadas dos pixels contidos nas amostras.

Cada técnica de IBR possui uma abordagem específica com relação à função plenóptica. Em [38, 53], a IBR é definida como um conjunto de técnicas para reconstruir uma representação contínua da função plenóptica a partir de amostras discretas completas ou incompletas.

2.1.2 Renderização sem Geometria

Nesta seção são descritas as técnicas que utilizam apenas imagens para renderizar novas vistas. As amostras são um conjunto de imagens (fotografias) tiradas do mundo real ou renderizadas por computador.

Imagens do mundo real proporcionam ao ambiente um aspecto mais realista, comparada com imagens renderizadas. Dificilmente as imagens renderizadas conseguem reproduzir todos os detalhes como geometria e efeitos de luz de uma cena do mundo real.

A seguir, na subseção 2.1.2.1 é apresentada a técnica panoramas cilíndricos, que utiliza imagens panorâmicas para representar o ambiente virtual, na subseção 2.1.2.2 é descrita a técnica mosaicos concêntricos, que permite a exploração do ambiente virtual dentro de uma região circular, na subseção 2.1.2.3 são apresentadas as abordagens *light field* e *lumigraph* que possibilitam a visualização de uma área ou objeto 3D delimitada por uma envoltória convexa e na subseção 2.1.2.4 é descrita a técnica *plenoptic stitching*, que proporciona a exploração de ambientes 3D formados por caminhos pré-definidos.

2.1.2.1 Panoramas Cilíndricos

Inicialmente propostos por Chen [17], os panoramas cilíndricos utilizam imagens panorâmicas para representar o ambiente virtual. Tornaram-se muito conhecidos pela implementação da técnica em um produto comercial chamado QuickTime® VR [5].

A cena é formada por uma imagem de panorama capturada por câmeras panorâmicas especiais [1], renderizadas por computador ou por uma câmera convencional através da junção de várias imagens para formar o panorama.

O conceito de mapas de ambiente é utilizado por esta técnica. A cena (imagem de panorama) é mapeada em uma projeção cilíndrica contendo um ponto de vista fixo e permitindo movimentos de rotação e aproximação ou afastamento (*zooming in/out*) do campo de visão no espaço da imagem.

Em um panorama cilíndrico, os movimentos de rotação são de 360 graus horizontalmente e limitados na direção vertical, variando geralmente de 50° [41] a 180° [17].

A orientação do ponto de vista atual, juntamente com o campo de visão e a distância focal da câmera definem a região visível da imagem de panorama. Uma nova vista é renderizada através do *warping*[2] da região visível para uma vista planar, criando uma nova imagem que representa o correto ponto de vista.

Uma única imagem de panorama é necessária para visualizar uma cena em um ponto de vista fixo. O observador geralmente fica posicionado no centro da cena e realiza mo-

[2] É uma transformação em que os pixels de uma imagem fonte são movidos para uma imagem destino de acordo com um mapeamento específico [23].

vimentos de rotação e *zooming* para a exploração do ambiente. Quando o usuário deseja avançar sua posição para outro ponto de vista, um novo panorama é carregado.

Um sistema baseado em panoramas cilíndricos para navegação nas ruas da cidade de Vancouver [56] utiliza-se de um observador posicionado na esquina de um quarteirão e, para se locomover pelas ruas, o usuário vai de um quarteirão (panorama) para outro.

Esse tipo de navegação não é suave, oferecendo uma navegação precária, onde o observador é limitado em girar em torno de um ponto para explorar o ambiente ou se locomover para frente ou para trás com a utilização do efeito de *zooming* na imagem, acarretando perda da qualidade devido à interpolação, além deste ser limitado.

Além dos panoramas cilíndricos, outras abordagens seguindo os mesmos princípios, podem ser criadas apenas mudando o mapeamento da projeção cilíndrica para esférica, cúbica, entre outros [17]. O tipo de projeção utilizado interfere diretamente na quantidade de graus de liberdade usados na movimentação de rotação horizontal e vertical e o número de imagens requeridas para formar o panorama.

2.1.2.2 Mosaicos Concêntricos

O mosaico de imagens tem como objetivo combinar uma ou mais imagens de entrada, tendo como saída uma imagem maior, que representa um campo de visão maior em comparação com o de uma única imagem [45]. Mosaicos concêntricos [52] é uma técnica que deriva do conceito de panoramas cilíndricos, permitindo que o observador tenha liberdade de movimentação dentro de uma região circular, além de usufruir de efeitos de luz. O ambiente virtual é representado por um conjunto de mosaicos panorâmicos [45]. Cada mosaico é construído pela junção de diversas imagens capturadas de pontos de vistas próximos.

Para a captura dos mosaicos concêntricos são utilizadas múltiplas câmeras, C0, C1, ..., Cn, como mostra a Figura 2.2. As câmeras são fixadas em uma superfície e cada uma posicionada em um ponto de vista diferente e fixo. Esta superfície é apoiada por somente um tripé colocado na primeira câmera.

A formação dos mosaicos concêntricos, CM0, CM1, ..., CMn é ilustrada na Figura 2.3.

As câmeras sofrem rotação ao longo de círculos concêntricos planares de forma uniforme e contínua e, durante esse movimento, várias imagens são capturadas. O posicionamento das câmeras durante o processo de captura é representado na Figura 2.3 pelas setas contidas em cada círculo. Como a movimentação da câmera é pequena entre cada captura, as imagens capturadas possuem sobreposição com a antecedente e a subsequente.

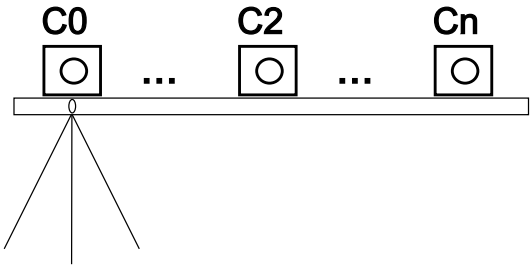


Figura 2.2: Organização das câmeras em mosaicos concêntricos.

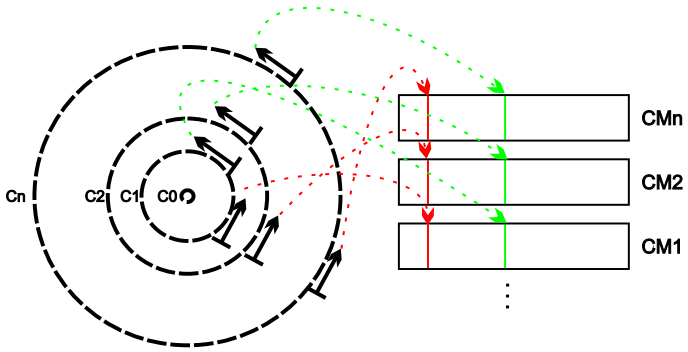


Figura 2.3: Criação dos mosaicos concêntricos. Adaptada de [52].

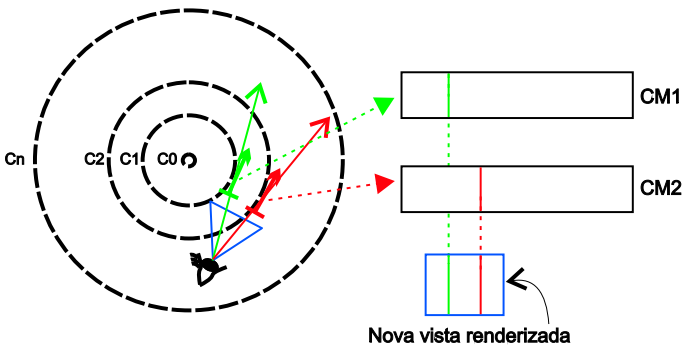


Figura 2.4: Renderização com mosaicos concêntricos. Adaptada de [52].

Por esse motivo, para a criação dos mosaicos, somente as *linhas verticais centrais*[3] das imagens capturadas são utilizadas, por representarem os pixel distintos entre cada captura,

[3]Região de uma imagem composta por uma ou mais linhas verticais localizadas no centro da imagem, também conhecida como *slit image* [65].

como mostram as linhas tracejadas vermelhas e verdes na Figura 2.3, que informam o instante de cada imagem capturada e onde os pixels contidos em cada imagem compõem os mosaicos correspondentes.

O processo de renderização dessa abordagem é mostrado na Figura 2.4. O observador está localizado dentro da região circular e seu campo de visão é ilustrado pelo triângulo de cor azul. Os raios de luz (linha de cor verde e vermelha), que passam através do observador e são tangentes ao círculos, determinam as linhas verticais contidas nos mosaicos (os pixels referentes às linhas verde e vermelha dos mosaicos CM1 e CM2) que compõem a nova vista renderizada.

As vistas renderizadas por esta abordagem possuem uma boa qualidade quando a distancia radial é pequena entre um círculo (mosaico) e outro adjacente. Quanto maior for a distância, mais interpolação é necessária para gerar as vistas, comprometendo a qualidade.

Na técnica apresentada, apesar do observador não estar fixo em um ponto, comparado com panoramas cilíndricos, ele ainda fica limitado em uma pequena região circular. A dimensão da cena determina a quantidade de mosaicos (círculos) necessários para ter uma renderização das vistas com qualidade. Uma cena com grande dimensão pode requerer uma grande quantidade de mosaicos concêntricos (imagens) para formar o ambiente virtual.

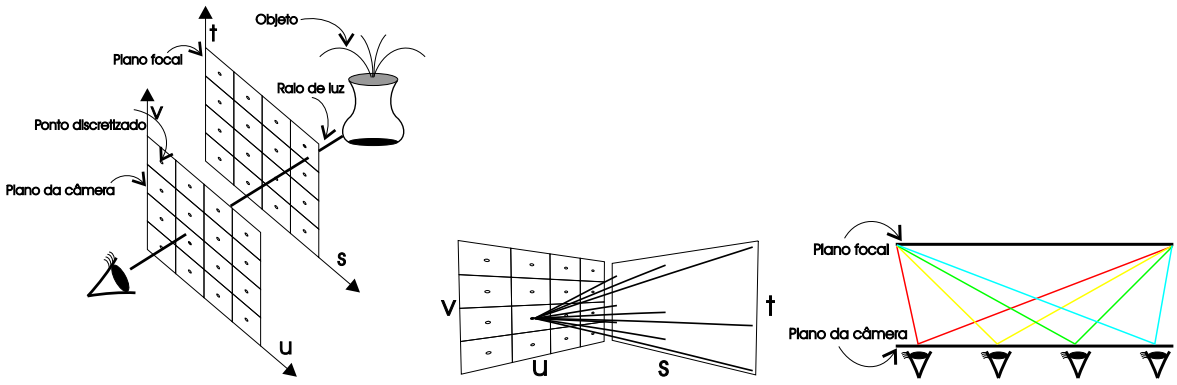
2.1.2.3 *Light Field* e *Lumigraph*

Light field [34] e *lumigraph* [24] são técnicas que possibilitam o observador explorar uma determinada área de um ponto de vista arbitrário. Esta área é delimitada por uma envoltória convexa, na forma de um cubo que cobre minimamente o objeto ou cena.

Acerca desta envoltória, os raios de luz em todas as posições e direções são parametrizados. A parametrização do espaço se dá com a interseção dos raios com dois planos, o da câmera *uv* e o focal *st*, ilustrado na Figura 2.5 (a). Cada face do cubo é formada por esses dois planos paralelos, sendo necessárias 6 faces para representar uma cena completa.

Os planos *uv* e *st* são subdivididos em diversas partes, criando uma representação

discreta em forma de uma grade. Cada subdivisão do plano *uv* é uma amostra (imagem) que é criada através dos raios que passam por um ponto do plano *uv* sobre todos os pontos do plano *st*, como mostra a Figura 2.5 (b). Quanto maior for a subdivisão, maior a resolução e a quantidade de amostras.



(a) (b) (c)

Figura 2.5: Representação dos raios de luz nos planos da câmera *uv* e focal *st*.

A criação das amostras de cada ponto do plano *uv* é denominada amostragem. Neste processo, o centro de projeção da câmera é posicionado em cada ponto do plano *uv* focando em direção ao plano *st*, apresentado na Figura 2.5 (c), cada par de linha da mesma cor representa o campo de visão do observador em pontos de vista distintos. Cada amostra é capturada de um ponto de vista diferente e armazenada em uma grande base de dados, contendo todas as amostras capturadas.

Para renderizar novas vistas, realiza-se o processo de reamostragem. Dada uma posição e direção do ponto de vista desejado, as intersecções dos raios formados para cada pixel sobre o plano *uv* são calculadas. Cada intersecção é relacionada a um pixel de uma determinada amostra, armazenada na base de dados. Finalmente, a imagem final do processo de renderização é construída através da interpolação dos pixels contidos nas amostras.

O processo de renderização pode ser otimizado, utilizando um mapeamento de textura sobre o plano *uv* para criar uma nova vista, sendo cada ponto do plano uma textura, ao invés de calcular as intersecções dos raios formados para cada pixel sobre o plano *uv*.

Os princípios de *light field* e *lumigraph* são semelhantes. A maior diferença entre as técnicas é que *lumigraph* usa uma aproximação geométrica para aprimorar o processo de

renderização, diminuindo o número de amostras necessárias para compor a nova vista [63] e utiliza correção de profundidade [24], melhorando a qualidade da imagem.

Estas abordagens fazem uso de uma grande base de dados de amostras, usada no processo de reamostragem para renderizar novas vistas. Quanto maior for a cena a ser representada, maior será a dimensão do cubo que cerca o ambiente e, consequentemente, aumentando o número de amostras, limitando a representação a pequenas áreas, não a ambientes completos [4].

2.1.2.4 *Plenoptic Stitching*

Plenoptic stitching [4] é uma técnica de IBR que proporciona a exploração de ambientes interativos 3D que não possuam obstruções, com uma movimentação suave, utilizando imagens do mundo real. A cena é constituída por caminhos pré-definidos, criando uma grade de forma irregular sobre um plano, onde ao longo de cada caminho são capturadas várias imagens omnidirecionais[4], como mostra a Figura 2.6.

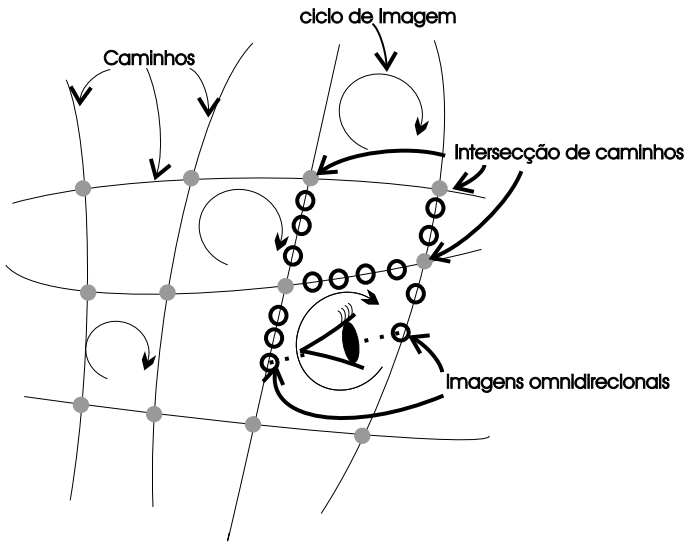


Figura 2.6: Construção de uma cena utilizando *plenoptic stitching*.

Além da grade, o ambiente é composto por *ciclos de imagens* [4], que são áreas definidas pelas intersecções dos caminhos na grade. Estas áreas são utilizadas pelo observador para gerar novas vistas referentes a sua interação no ambiente. Dentro do ambiente, o

[4]Imagem panorâmica com um grande campo de visão, chegando até 360° [57].

observador é posicionado em um ciclo de imagem específico, e sua movimentação de translação na cena é realizada de uma área para outra próxima. Movimentos de rotação são permitidos somente na horizontal, sendo computados dentro do mesmo ciclo de imagem.

No processo de renderização, inicialmente são determinadas duas imagens omnidirecionais de entrada (em áreas convexas), posicionadas na frente e atrás do observador, com relação ao atual ponto de vista do observador, como mostra a Figura 2.6. Em áreas não convexas, mais de duas imagens omnidirecionais de entrada poderão ser necessárias.

Então, novas vistas são construídas através de um *warping* de cada imagem de entrada para uma reprojeção planar e cada pixel da imagem resultante do processo de renderização é colorido com a combinação entre os pixels das imagens planares, coluna por coluna. Qual coluna de cada imagem planar é combinada é determinada por uma interpolação [4].

As vantagens desta abordagem são o uso de imagens reais, ao invés de imagens renderizadas e a criação de uma cena completa com movimentação suave sobre ela, além de poder ser utilizada em ambientes de diferentes formas e tamanhos.

Em contrapartida, para a construção de cada ambiente há a necessidade da intervenção humana para determinar quantos e por onde os caminhos vão ser posicionados para formar a grade, não sendo automatizado. Devido à utilização de interpolação e filtros de suavização no processo de renderização das vistas, quando o observador está interagindo no ambiente, a qualidade da imagem final é severamente prejudicada. A principal desvantagem deste método é a enorme quantidade de dados que um simples ambiente necessita para sua construção. Para exemplificar, em uma cena com 8 caminhos (totalizando 49 metros de extensão) e 9 ciclos de imagens, 1600 imagens omnidirecionais são tiradas, totalizando 365MB [4].

2.1.3 Renderização com Geometria

Neste conjunto de técnicas, utiliza-se algum tipo de informação geométrica para renderizar novas vistas, seja ela implícita, com a utilização de cálculos de projeção, ou explícita, com o uso de imagens de profundidade[5] ou coordenadas 3D.

[5]Uma imagem que, além das coordenadas xy referente à cor de cada pixel, possui um valor escalar z, associado a cada ponto (pixel) amostrado, representando a distância entre um ponto e uma entidade

A seguir, nas subseções 2.1.3.1 e 2.1.3.2 são apresentadas, respectivamente, as técnicas *view interpolation* e *view morphing*, que estendem o conceito da técnica *morphing* para a renderização dos quadros, na subseção 2.1.3.3 é descrita a técnica 3D *warping*, que utiliza uma ou mais imagens de entrada juntamente com a informação da profundidade dos pixels para renderizar novas vistas, e por fim, na subseção 2.1.3.4 é apresentada a abordagem *view-dependent texture mapping*, que permite a modelagem e renderização de cenas arquitetônicas.

2.1.3.1 *View Interpolation*

View interpolation [18] estende o conceito da técnica *morphing*, realizando a interpolação entre duas imagens fontes para gerar uma imagem intermediária, aproximando uma transformação 3D de um objeto ou cena, utilizando imagens sintéticas.

Morphing ou metamorfose de imagem é uma técnica que produz a transição entre duas imagens fontes [50]. Esta transição é feita por uma interpolação das posições e cores dos pixels de cada imagem fonte, para uma imagem intermediária, através de um mapeamento. Este mapeamento é a correspondência dos pixels[6] ou segmentos de linhas entre as duas imagens fontes. Tradicionalmente, esta correspondência é realizada manualmente [18].

A imagem intermediária representa uma nova vista gerada através do *morphing* de duas imagens fontes, de acordo com a posição e a direção da câmera. Para representar um ambiente (Figura 2.7 (a)), pode-se utilizar uma malha de retângulos, onde cada retângulo é cercado por diversas imagens omnidirecionais. O observador é posicionado no centro de um retângulo arbitrário e sua movimentação é feita de um retângulo para outro próximo, semelhante à construção de um ambiente utilizando a técnica *plenoptic stitching*.

Para renderizar novas vistas, duas imagens planares são reprojetadas, de duas imagens omnidirecionais próximas, de acordo com a orientação atual do observador no interior de um retângulo, como mostra a Figura 2.7 (b). Cada imagem planar é uma imagem fonte

de referência [41]. Em imagens sintéticas, a informação de profundidade é facilmente obtida como, por exemplo, através da leitura do *buffer* de profundidade (*z-buffer*) disponibilizado pela biblioteca gráfica OpenGL.

[6] Pontos (pixels) de duas ou mais imagens projetados em um mesmo ponto no mundo [20].

utilizada no *morphing*, gerando a vista intermediária.

O *morphing* aplicado por esta técnica utiliza-se de imagens de profundidade e a transformação da câmera[7] para determinar a correspondência entre as imagens fontes de forma automática [18]. Esta correspondência cria um mapeamento, que descreve quais pixels de cada imagem fonte são interpolados para compor a imagem intermediária.

Uma nova vista é criada com a interpolação das coordenadas dos pixels de cada imagem com profundidade, determinando o local onde cada pixel mapeado será movido, compondo a imagem intermediária.

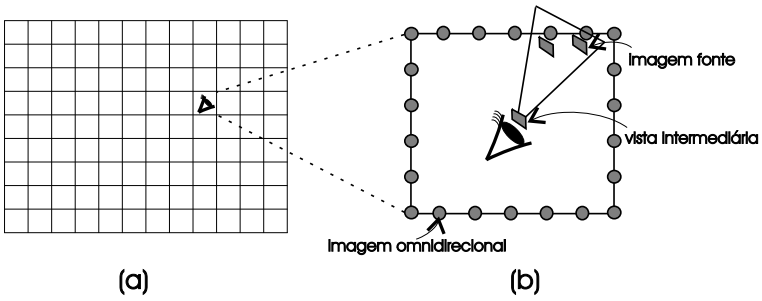


Figura 2.7: Construção de uma cena utilizando *view interpolation*.

Durante o processo de interpolação, muitos pixels adjacentes das imagens fontes podem ser movidos para um mesmo pixel da imagem intermediária. Isso ocorre quando uma imagem fonte sobrepõe a outra. Este problema pode ser resolvido através da comparação da coordenada z dos pixels, sendo que o pixel mais próximo será movido para a imagem intermediária.

Outro problema que pode ocorrer com esta técnica é a ocorrência de buracos (pixels que não foram coloridos) na vista renderizada. Para solucioná-lo, em cada pixel que não tiver nenhum valor atribuído será realizada uma interpolação com pixels adjacentes. O número de buracos será menor quanto mais próximas as duas imagens omnidirecionais envolvidas no processo de renderização estiverem posicionadas no ambiente.

Esta abordagem, além da exploração de ambientes virtuais, pode ser utilizada para computar o efeito *motion blur* e sombra [18]. *Motion blur* é um efeito que aparenta em uma imagem a presença de movimento em grande velocidade.

[7]Matriz formada pela posição e orientação da câmera, alinhados com o sistema de coordenadas do mundo [43].

Para Seitz e Dyer [50], técnicas baseadas em *morphing* utilizadas para gerar vistas 3D podem gerar vistas geometricamente incorretas (torções na forma do objeto ou cena), em casos onde as imagens fontes não estão paralelas com relação ao plano da imagem da câmera. Este problema pode comprometer a qualidade da imagem renderizada.

Para não haver um número excessivo de buracos nas vistas renderizadas, o ambiente deve ser denso de imagens omnidirecionais, para que as imagens fontes sejam capturadas de ponto de vista próximos, evitando o aparecimento de buracos [18].

2.1.3.2 *View Morphing*

Semelhante à *view interpolation*, *view morphing* [50] renderiza novas vistas aproximando uma transformação natural de um objeto 3D, a partir da transição de duas imagens fontes, utilizando imagens reais ou sintéticas. Esta técnica usa informação geométrica implícita, uma vez que apenas uma matriz da transformação da câmera referente a cada imagem fonte é necessária para computar o *morphing*.

A transformação natural ou geometricamente correta é possível porque *view morphing* realiza uma efetiva transição entre as imagens, evitando possíveis torções que possam ocorrer no objeto, preservando sua forma. Estas torções são comuns utilizando métodos de *morphing* convencionais [50], por não considerar mudanças no ponto de vista da câmera ou posição do objeto, resultando em imagens matematicamente incorretas.

Com o objetivo de produzir vistas geometricamente corretas, *view morphing* renderiza novas vistas através de três passos básicos: *pré-warping* de duas imagens fontes, *morphing* entre as imagens resultantes do primeiro passo e, por fim, o *pós-warping*, criando a nova vista. No estágio do *pré-warping*, as imagens fontes são reprojetadas com o uso de uma transformação de projeção [50], realizando uma projeção de perspectiva, fazendo com que as imagens fontes fiquem paralelas ao plano da imagem da câmera. Quando as imagens fontes estão paralelas, o *morphing* destas imagens resulta em uma transição sem torções no objeto, preservando sua forma [50]. O segundo passo é o *morphing* das imagens resultantes do *pré-warping*. Por último é realizado o *pós-warping*, que consiste em transformar o plano da imagem da nova vista gerada pelo segundo passo, para a posição e orientação desejada.

Problemas como buracos e sobreposição de pixels, encontrados em *view interpolation* também aparecem nesta abordagem. A solução destes problemas em ambas as técnicas é semelhante, porém, no caso da sobreposição de pixels utilizando imagens reais é computada a disparidade dos pontos[8], substituindo a profundidade dos pixels, necessária para a solução deste problema.

Dos três estágios que compõem *view morphing*, *pré-warping* é executado automaticamente, embora o *pós-warping* requer a especificação manual de pontos de controle (pontos que formam uma região arbitrária na imagem), utilizados na transformação da imagem final, na posição e direção desejadas. Durante a exploração de um ambiente pelo observador, o processo de renderização das vistas é feito automaticamente, onde o usuário se preocupa apenas com a interação na cena. A especificação manual desses pontos de controle, necessários para gerar novas vistas com *view morphing*, torna seu uso impraticável neste contexto.

Outras abordagens com princípios semelhantes às de *mophing* empregadas na síntese de vistas são encontradas na literatura, em vez de duas imagens fontes, existem métodos que utilizam três [6] ou pequeno conjunto de imagens [32], associadas à transformação da câmera de cada uma, cálculo dos pontos correspondentes entre as imagens fontes e a reprojeção dos pixels para a nova vista. *Plenoptic modeling* [38] é uma abordagem similar às técnicas que empregam duas imagens fontes para computar uma vista. A diferença é que as imagens são projeções cilíndricas panorâmicas. Para gerar novas vistas, um mapeamento é calculado com a correspondência dos pontos das duas imagens e é realizado um *warping* destes pontos para uma imagem cilíndrica arbitrária ou uma vista planar.

2.1.3.3 3D *Warping*

3D *warping* de imagens [37] utiliza uma ou mais imagens com profundidade (imagens fontes) e a informação da transformação da câmera associada a cada uma para criar novas vistas, com a projeção dos pixels das imagens fontes para uma imagem destino, relativo

[8] É inversamente proporcional à profundidade dos pixels, computado pela diferença entre as coordenadas x de pontos correspondentes, em vistas paralelas [50].

ao ponto de vista desejado.

Esta projeção (*warping*) realiza uma transformação geométrica mapeando uma imagem fonte com profundidade para uma vista destino arbitrária. Uma nova vista é criada através da projeção dos pontos tridimensionais X, referente a todas as interseções dos raios visíveis de origem em $C1$ e $C2$ passando pelos planos fonte e destino, ilustrado na Figura 2.8. Um ponto X do espaço Euclidiano 3D é dado por uma intersecção entre os raios visíveis de origens nos centros de projeções (COPs) $C1$ e $C2$ e passam pelas coordenadas $x1$ e $x2$ dos planos das imagens fonte e destino, respectivamente.

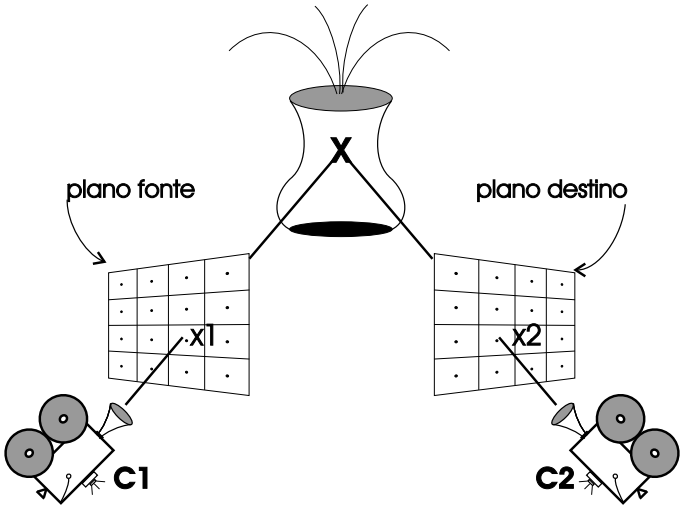
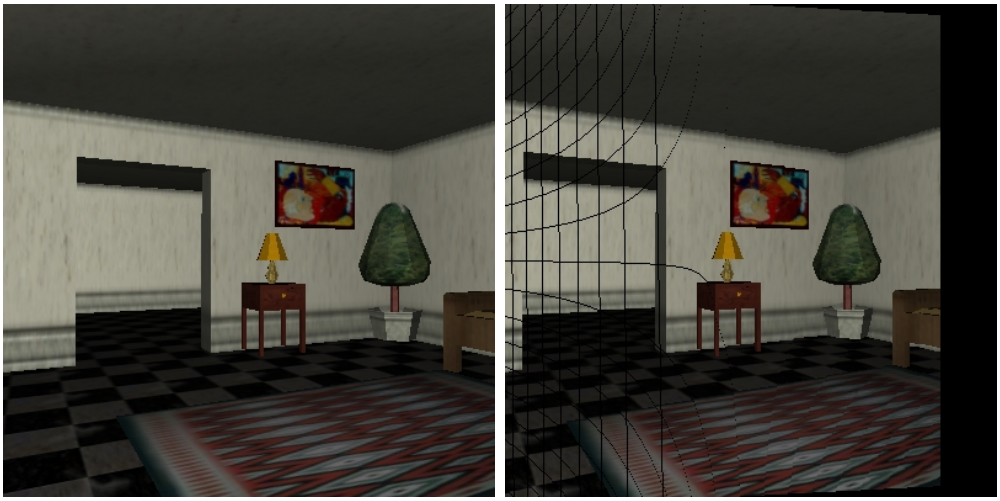


Figura 2.8: Ponto tridimensional X visualizado pelos planos fonte e destino.



(a) Imagem de entrada (b) Vista renderizada via 3D *warping*

Figura 2.9: Artefatos presentes na vista renderizada empregando 3D *warping*.

Esta técnica permite que o *warping* para obter a vista desejada seja feito sem a neces-

sidade da comparação do valor de profundidade dos pixels da imagem fonte com a destino. As coordenadas dos pixel da imagem destino podem ser obtidas diretamente expressando a coordenada X em ambos os sistemas de câmera (fonte e destino) [37].

Problemas como buracos e desoclusão, denominados artefatos, podem surgir na imagem destino. Buracos aparecem devido à diferença entre a resolução de amostragem entre a imagem fonte e a destino. Desoclusão são regiões na imagem destino que não receberam valores durante o processo de amostragem dos pixels. Os artefatos são ilustrados na Figura 2.9 (b). A região escura do lado direito da imagem representa a desoclusão e os buracos são observados do lado esquerdo da imagem. A partir de uma imagem de entrada (Figura 2.9 (a)) é renderizada uma nova vista utilizando 3D *warping* (Figura 2.9 (b)), que representa um movimento de rotação para a direita.

Para preencher os buracos, a forma mais comum utilizada é a expansão dos pixels da imagem fonte para a destino durante o mapeamento, fazendo com que um pixel da imagem fonte seja mapeado para vários da imagem destino. Este método é denominado *splatting* [9, 36, 51].

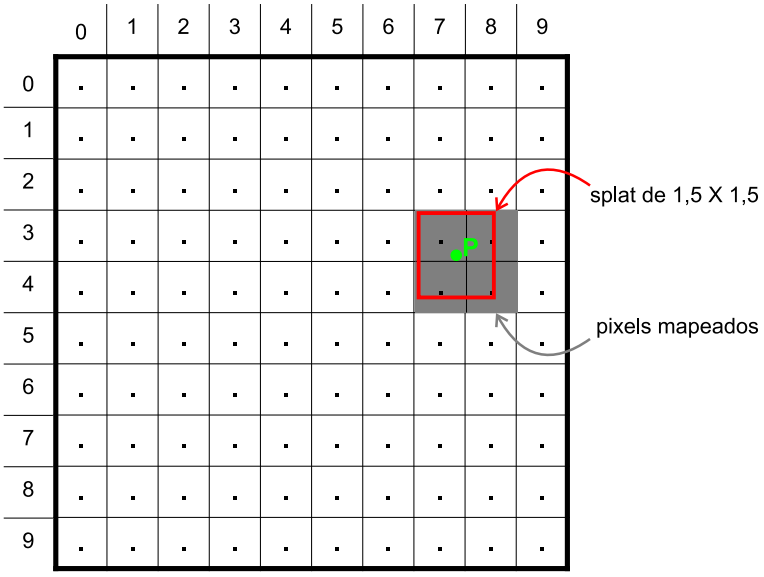


Figura 2.10: Implementação da técnica de *splatting* com dimensão de $1,5 \times 1,5$ pixels.

Uma forma de implementar a técnica de *splatting* é mostrada na Figura 2.10. Um ponto (pixel) da imagem fonte é mapeado em um ponto P na imagem destino (ponto marcado na cor verde). Baseado na coordenada de P é posicionado o *splat* (quadro de cor

vermelha) de dimensão $1,5 \times 1,5$ pixels, fazendo com que o ponto P fique centralizado com relação ao *splat*. Os pixels da imagem destino cobertos pela área em que o *splat* abrange (pixels de cor cinza) serão mapeados por um pixel da imagem fonte, ou seja, o pixel da imagem fonte referente à projeção do ponto P é copiado para os pixels (3,7), (3,8), (4,7), (4,8) na imagem destino.

Outro problema que pode surgir nas vistas renderizadas por esta técnica é a desoclusão, a qual ocorre quando parte da cena que está sendo visualizada na imagem destino não está na imagem fonte. Apresentado na Figura 2.11 (a), dado o plano fonte referente ao COP $C1$ e o plano destino referente ao COP $C2$, representando um movimento de rotação para a esquerda, a desoclusão é visualizada na região escura do plano destino (Figura 2.11 (b)).

Uma forma de minimizar esse problema é a utilização de várias imagens fontes no *warping* para a imagem destino. Porém, isso pode gerar problemas de visibilidade, porque a ordem em que os pixels serão projetados não pode ser estabelecida [41]. Ao invés do *warping* de múltiplas imagens fontes, Shade et al. [51] propuseram uma técnica chamada *Layered Depth Image* (LDI), que realiza a união de múltiplas imagens fontes em uma única. Cada pixel da imagem fonte é composto por múltiplos valores de cor e profundidade, minimizando ou quase resolvendo a desoclusão.

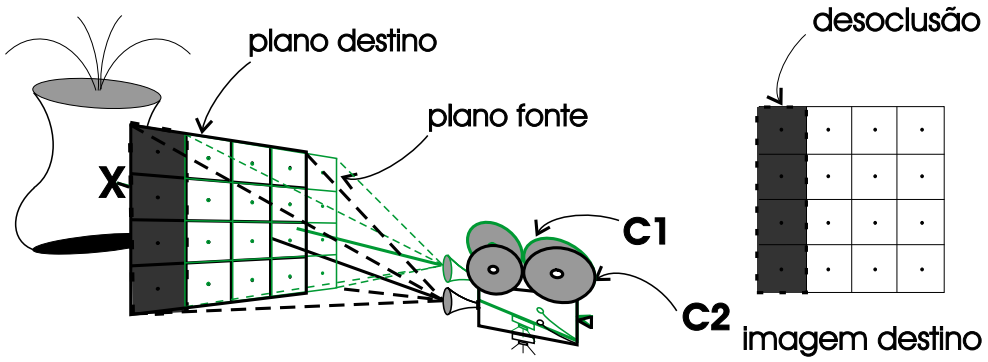


Figura 2.11: Desoclusão na imagem destino produzida pelo 3D *warping*.

LDI possui uma resolução fixa, podendo não prover uma taxa de amostragem adequada para todas as imagens fontes [15]. *LDI Tree* [15] resolve este problema combinando uma representação hierárquica com o conceito da LDI, selecionando uma LDI na hierarquia para cada pixel das imagens fontes.

2.1.3.4 *View-Dependent Texture Mapping*

Mapeamentos de textura são largamente utilizados em computação gráfica para produzir ambientes fotorrealistas [53]. Diferente dos métodos de mapeamento de texturas convencionais, em que imagens são utilizadas para colorir faces de um modelo, *view-dependent texture mapping* (VDTM) interpola duas ou mais imagens de entrada referente à cena, dependendo do ponto de vista do usuário [20].

VDTM foi inicialmente proposta por Debevec et al. [20], eles apresentam uma abordagem para modelagem e renderização de cenas arquitetônicas utilizando imagens reais. Inicialmente, um modelo geométrico aproximado é criado a partir de um conjunto de imagens reais e aplicado VDTM neste modelo para renderizar novas vistas.

O processo de modelagem é realizado de forma manual. A partir de um conjunto de fotografias arquitetônicas, arestas são marcadas e primitivas geométricas são instanciadas, então, um modelo geométrico aproximado é criado através da correspondência entre as arestas marcadas nas imagens e as arestas pertencentes às primitivas geométricas.

Para renderizar novas vistas é aplicado o método VDTM, que projeta as imagens de entrada sobre o modelo, com relação ao ponto de vista desejado. Quando um pixel arbitrário da vista renderizada é projetado por mais de uma imagem de entrada (sobreposição das imagens de entrada) é aplicada uma *função de pesagem* [20], que realiza uma ponderação entre os pixel projetados.

Esta função consiste em atribuir pesos às imagens de entrada sobrepostas. Quanto mais próximo for o ângulo de visão da vista renderizada, maior será o peso atribuído. Assim, quando há sobreposição, os pixels são compostos pela interpolação das imagens de entrada de acordo com os pesos especificados.

Esta abordagem, através de um conjunto de imagens de entrada, constrói um modelo aproximado e aplica as mesmas imagens como textura. O processo de modelagem é manual e árduo, podendo se tornar impraticável em cenas com grande dimensão. Além disso, a modelagem é baseada em arestas e detalhes geométricos que podem ser perdidos nos modelos construídos.

Debevec et al. [21] aprimoram VDTM reduzindo seu custo computacional com o pré-

processamento dos polígonos das imagens de entrada, referente à visão atual, criando um mapeamento da visão e o emprego de mapeamento de texturas projetivas [49].

2.1.4 Discussão

Entre as técnicas descritas que utilizam alguma informação geométrica, *view-dependent texture mapping* é capaz de renderizar um ambiente completo, embora se limite a cenas arquitetônicas e detalhes geométricos possam ser perdidos nos modelos construídos.

View interpolation e *view morphing* renderizam novas vistas a partir da transição de duas imagens fontes, relativo ao ponto de vista desejado, enquanto 3D *warping* requer pelo menos uma imagem fonte. A renderização de um ambiente completo utilizando estas técnicas pode ser obtida com o emprego de uma grande quantidade de amostras.

Entre as abordagens que empregam somente imagens para renderizar novas vistas, *plenoptic stitching* é capaz de renderizar um ambiente completo. Porém, o grande problema é o número excessivo de amostras requeridas.

Mosaicos concêntricos, *light field* e *lumigraph* também possuem o mesmo problema encontrado em *plenoptic stitching*, além de limitar a exploração a pequenas áreas, ao invés de um ambiente completo.

A utilização de panoramas cilíndricos tem uma grande vantagem sobre as demais técnicas, por necessitar apenas de uma única imagem de panorama para renderizar novas vistas. Porém, o observador é fixo em um ponto de vista arbitrário e a exploração do ambiente é limitada a movimentos de rotação em torno deste ponto.

Com o emprego de poucas amostras, a renderização de um ambiente completo que ofereça uma navegação suave se torna um grande desafio. Baseados nesta premissa, este trabalho utiliza uma arquitetura cliente-servidor, renderizando o ambiente virtual 3D remotamente. O servidor, dotado de um grande poder de processamento gráfico, renderiza os modelos 3D sob demanda e posteriormente transmite as imagens resultantes deste processo para o cliente, que apenas exibe as imagens recebidas.

Para a aprimorar a renderização da cena, a técnica de IBR escolhida foi a 3D *warping*, por ser a única técnica de todas as descritas na seção 2.1 que é capaz de renderizar novas

vistas a partir de uma única amostra, no caso é utilizado o último quadro recebido pelo cliente para a renderização de novas vistas localmente.

2.2 Renderização em Dispositivos com Recursos Gráficos Limitados

A renderização e visualização de conteúdos 3D em dispositivos com capacidade de processamento gráfico limitada foi previamente proposta. Muitos trabalhos encontrados na literatura relacionados a esse contexto empregam uma arquitetura cliente-servidor, renderizando o conteúdo 3D remotamente. O servidor, dotado de um grande poder de processamento gráfico, renderiza os modelos 3D sob demanda e posteriormente transmite as imagens resultantes deste processo para o cliente, que apenas exibe as imagens recebidas.

Brachtl et al. [14] apresentam um sistema de navegação[9] baseado em PDA para ambientes 3D. Inicialmente, o usuário entra com o ponto inicial e final do caminho que deseja explorar, então, estas informações são enviadas para um servidor, que renderiza todo o caminho a ser percorrido na forma de um vídeo e transmite-o para o cliente exibi-lo.

O vídeo é formado de quadros renderizados a partir de um modelo 3D, de acordo com a exploração do ambiente no caminho especificado. Os quadros sofrem uma redução no número de *bits* com o emprego da técnica de *dithering*[10] e posteriormente são comprimidos. O uso de *dithering* proporciona uma menor representação dos quadros renderizados devido à redução de profundidade, porém, a qualidade da imagem resultante pode ficar comprometida, conforme o limite de profundidade adotado.

As principais desvantagens deste método são o fato da exploração do ambiente ser limitada apenas ao caminho gerado pelo servidor e o usuário não ter uma interação direta no ambiente virtual, ele somente especifica pontos de entrada. O vídeo referente a todo o percurso é criado no servidor e transmitido para o cliente, que apenas exibe o vídeo

[9] Este tipo de sistema geralmente é utilizado para orientar os usuários de como ir de um lugar para outro dentro de um ambiente [14].

[10] Técnica que cria a ilusão de profundidade de cor em imagens de profundidade limitada.

recebido, dando uma noção por onde o usuário do sistema deve-se locomover para chegar ao destino.

Quax et al. [46] propõem um sistema de navegação baseado em dispositivos móveis usado como guia de turistas e visitantes em um meio urbano. O sistema utiliza uma representação tridimensional do ambiente e a cena é renderizada sob demanda (interação do usuário na cena) através de um conjunto de servidores, que codifica vídeos com os quadros renderizados e posteriormente transmite-os para os clientes, que exibe na tela o conteúdo recebido.

A arquitetura cliente-servidor desta abordagem é composta pelos clientes e quatro servidores principais: o de autenticação, conteúdo, mundo e de renderização. Quando determinado cliente deseja se conectar ao mundo virtual, o servidor responsável por esta tarefa é o de autenticação, além de controlar se uma determinada conexão pode ou não receber dados de áudio ou vídeo requisitados. Os dados multimídia são fornecidos pelo servidor de conteúdo de acordo com a interação do usuário na cena.

Sempre quando o usuário realiza uma interação no mundo virtual, em que a posição ou orientação da câmera é alterada, o servidor do mundo é invocado, que requisita ao servidor de renderização para codificar um vídeo referente à movimentação e, em seguida, transmiti-lo para o cliente.

O emprego de diversos servidores que compõem a infra-estrutura do servidor como um todo tem como vantagem a divisão das tarefas, mas obviamente esta abordagem requer um parque computacional maior, comparado com um único servidor. Uma grande desvantagem deste tipo de abordagem é sua suscetibilidade a problemas que possam ocorrer na transmissão dos dados do servidor para os clientes, como perda de pacotes ou atraso no recebimento dos dados por parte do cliente. Se houver atraso no recebimento do conteúdo renderizado, travamentos podem ocorrer durante a interação do usuário no ambiente virtual pelo fato de que o quadro requerido não está disponível para a visualização em um intervalo de tempo adequado.

Outra abordagem que sofre deste mesmo problema é apresentada por Noimark e Cohen-Or [40], contendo uma arquitetura semelhante à detalhada anteriormente, em que

o modelo 3D é renderizado no servidor sob demanda, de acordo com a ponto de vista do usuário e vídeos são codificados e transmitidos para os clientes. Entretanto, este método apresenta algumas vantagens. O processo de codificação do vídeo é acelerado através de um algoritmo de estimação de movimento [40] baseado nos parâmetros da câmera conhecidos.

Além disso, cada quadro é codificado com vários níveis de quantização[11], ou seja, regiões da imagem mais distantes do plano da câmera recebem um valor de quantização maior, enquanto os mais próximos recebem valores menores. Assim, os objetos mais próximos possuem uma qualidade visual melhor que os mais distantes. A utilização de coeficientes de quantização corretos produz uma imagem com melhor taxa de compressão e qualidade [40].

Park et al. [44] propõem um sistema para a visualização de dados médicos baseado em PDA, suportando um trabalho colaborativo entre os usuários remotos conectados e permitindo a exploração de um grande conjunto de dados médicos de forma compartilhada. A arquitetura do sistema é composta por três módulos principais: cliente, *gateway* e o servidor de renderização paralelo. O cliente escolhe um conjunto de dados e especifica os parâmetros de visualização deste conteúdo de forma interativa e cooperativa, então envia ao *gateway* uma requisição dos dados que deseja renderizar, que gerencia a transmissão de dados entre os clientes e o servidor de renderização paralelo.

As requisições recebidas pelo *gateway* são entregues para o servidor de renderização paralelo, que renderiza o conteúdo 3D de forma eficiente através de um *cluster* de alto desempenho e entrega o resultado na forma de imagens ou vídeos para o *gateway*, que transmite para todos os usuários conectados. A renderização remota empregada por esta arquitetura requer um grande parque computacional, no caso, 15 nodos são utilizados no servidor de renderização paralelo, sendo impraticável o uso deste modelo em projetos que não dispõem desses recursos.

Chim et al. [19] e Yang e Kuo [62] empregam o conceito de malhas progressivas[12] para

[11] O nível de quantização determina a taxa de compressão e controla o grau da compressão com perdas [40]. Quanto maior for o nível de quantização, maior será a compressão e a degradação da qualidade da imagem.

[12] Inicialmente proposta por Hoppe [28], esta técnica é baseada em duas transformações aplicadas na

reduzir o tempo de transmissão e renderização de modelos 3D utilizando redes de largura de banda limitada. O servidor armazena dados geométricos e transmite informações geométricas sobre os modelos 3D para o cliente, que constrói e renderiza estes modelos progressivamente. Primeiramente, o servidor envia o modelo contendo uma malha triangular simplificada para o cliente e progressivamente são transmitidas instruções para criar novas faces na malha, até que esta seja reconstruída por completo, representando a geometria original do modelo.

O emprego de malhas progressivas permite transmitir os modelos 3D do servidor para os clientes otimizando o uso da rede, já que menos dados são transmitidos, comparado com uma transmissão que replica o modelo completo. Porém, malhas progressivas são impraticáveis para grandes modelos devido ao enorme tempo gasto para construi-las [9].

Quillet et al. [47] propõem um método para renderização remota de modelos 3D de grandes cidades. O processo de renderização é baseado em linhas, com o objetivo de reduzir o tamanho dos modelos e conseguir uma transmissão mais eficiente dos dados do servidor para o cliente. O servidor é o responsável por enviar, sob demanda, dados geométricos para o cliente renderizar. Estes dados são linhas armazenadas de uma forma vetorial, criadas a partir de um conjunto de fotografias, que representam a aparência externa de prédios e construções das ruas de uma cidade. As linhas são geradas com o uso de um algoritmo de detecção de bordas e a conexão dos segmentos de retas identificados, formandos as linhas.

Nesta abordagem, os dados transmitidos para os clientes são arquivos binários contendo a representação das linhas, sendo que cada linha é representada por quatro valores do tipo real referente às coordenadas de início e fim de uma linha, reduzindo significativamente o tamanho do modelo representado [47]. As principais desvantagens desta abordagem são o emprego de uma renderização que não é fotorrealista e a restrição no uso de modelos que apenas representam uma aparência externa de edifícios e construções de ruas de uma cidade.

---

malha de triângulos do modelo, o *colapso de arestas*, em que faces da malha são removidas, e a *divisão de arestas*, adicionando novas faces. Dada uma malha de triângulos arbitrária, esta técnica é capaz de reduzir e aumentar sua resolução progressivamente.

Hachet et al. [26] propõem um sistema de navegação para dispositivos móveis utilizados como guia em caminhadas ou passeios turísticos, visualizando o terreno 3D em sua posição atual e informações complementares (nome de montanhas, altura) sobre o local. A arquitetura do sistema é composta por cinco componentes: cliente, servidor principal, servidor de informações geográficas, servidor de informações complementares e o servidor de renderização.

Esta abordagem utiliza panoramas cúbicos para renderizar o ambiente virtual, semelhante à técnica de IBR panoramas cilíndricos, mudando somente a projeção, ao invés de cilíndrica é cúbica. Inicialmente, o cliente requisita ao servidor principal a visualização do terreno de acordo com sua localização, que coleta informações geográficas e complementares sobre a localização fornecida com os servidores específicos na arquitetura do dado em questão. Por fim, estas informações são enviadas ao servidor de renderização, que gera o panorama e envia para o cliente visualizar o ambiente virtual. Quando o usuário deseja visualizar um outro local, ele deve requisitar ao sistema manualmente, que repete o processo de renderização, caracterizando uma navegação que não é suave no ambiente.

Boukerche e Pazzi [11, 12, 13] e Lei et al. [33] apresentam abordagens para a renderização do ambiente virtual utilizando a técnica de IBR panoramas cilíndricos, provendo uma movimentação suave no ambiente virtual. O servidor renderiza sob demanda e progressivamente as imagens de panorama, que são transmitidas para o cliente, que as utiliza para compor o panorama relativo à posição atual do observador. O cliente, enquanto espera a renderização e transmissão dos dados requisitados ao servidor, pode realizar movimentos de rotação horizontal, renderizando as vistas com o panorama parcial ou completamente construído ou mover-se para frente ou para trás com a utilização do efeito de *zoom* na imagem.

Além disso, um mecanismo é utilizado para a predição dos movimentos do usuário no ambiente virtual, determinando a prioridade da renderização e transmissão das imagens. As imagens renderizadas no servidor são armazenadas em um *buffer* no cliente. Antes de requisitar ao servidor a renderização de um fragmento relativo a um panorama arbitrário, o cliente verifica se esta imagem encontra-se em seu *buffer*; caso positivo, a requisição é

cancelada e o processo de renderização se torna mais eficiente.

A grande desvantagem das abordagens baseadas em panoramas cilíndricos é a navegação precária no ambiente virtual, o usuário não pode mover-se para a esquerda ou para a direita, os movimentos de translação são limitados para frente e para trás na direção atual do observador. O movimento de rotação na vertical também não é permitido. Este movimento é ausente com o objetivo de decrescer o tamanho das imagens a serem transmitidas e diminuir a quantidade de movimentos a serem preditos pelo mecanismo de predição.

Chang e Ger [16] propõem uma renderização remota da cena sob demanda para dispositivos móveis utilizando a técnica 3D *warping* para auxiliar no processo de renderização. O cliente requisita as vistas ao servidor e quando há atraso ou algum problema no recebimento de um quadro requerido, ou mesmo enquanto espera sua chegada, o cliente aplica a técnica 3D *warping* no último quadro recebido, renderizando a vista desejada localmente. Ao chegar o quadro transmitido pelo servidor, este é exibido na tela, substituindo o quadro renderizado através de 3D *warping*, uma vez que as vistas renderizadas pela técnica 3D *warping* podem apresentar artefatos e a vista transmitida pelo servidor não possui, porque foi renderizada a partir do modelo 3D.

Outra abordagem semelhante e mais eficiente que a descrita acima é proposta por Bao e Gourlay [8, 9] e por Bao et al. [10], que visa reduzir o tamanho das imagens transmitidas e minimizar o problema de desoclusão nas vistas renderizadas através do 3D *warping*. Inicialmente, o servidor renderiza o primeiro quadro e o envia para o cliente; cada quadro é uma imagem de profundidade. Sempre que o cliente requisita ao servidor um novo quadro relativo à mudança em seu ponto de vista, enquanto espera os dados do servidor, o próprio cliente renderiza sua vistas, aplicando o 3D *warping* no quadro atual (imagem fonte).

Durante esse tempo de espera, o cliente pode requisitar outras mudanças em seu ponto de vista, sem que os dados da primeira requisição tenham chegado. Quanto maior for a mudança no ponto de vista desejado, mais artefatos podem aparecer na imagem resultante do 3D *warping*, quando se utiliza a mesma imagem fonte.

Quando o servidor recebe a requisição de um novo quadro, inicialmente é computado

o 3D *warping* do último quadro renderizado e, posteriormente, renderiza o novo quadro relativo ao ponto de vista requisitado. Então, duas imagens são produzidas, uma referente ao *warping* e a outra da renderização da cena.

Posteriormente, o servidor calcula a diferença entre essas duas imagens, criando uma imagem denominada resíduo, que é comprimida e enviada para o cliente. Esta imagem armazena uma informação esparsa e, quando comprimida, seu tamanho é reduzido significativamente. O cliente, ao receber o resíduo, gera a vista compensada, concatenando a imagem de resíduo com o quadro atual renderizado pelo 3D *warping*. A vista criada é livre de artefatos e representa a projeção correspondente ao ponto de vista requisitado.

Para minimizar o aparecimento da desoclusão nas imagens renderizadas pelo 3D *warping*, a imagem renderizada pelo servidor possui um tamanho maior que o campo de visão da câmera utilizado no cliente.

Embora, nesta abordagem o próprio cliente renderize suas vistas enquanto espera os resíduos transmitidos pelo servidor, obrigatoriamente o cliente depende desse dado para criar a vista compensada da cena e para que as futuras renderizações não apresentem desoclusão.

Uma grande desvantagem é o fato do cliente ter que esperar o resíduo correspondente à última vista renderizada pelo 3D *warping*. Se o cliente, enquanto espera os dados de uma requisição feita ao servidor, renderizou vistas relativas a três movimentos, por exemplo, caso receba os resíduos referentes aos movimentos do primeiro e do segundo movimentos, estes serão descartados. Apenas quando chegar o resíduo relativo ao último movimento é que a vista compensada será criada.

Além disso, a transmissão dos resíduos realizada do servidor para os clientes exige um protocolo confiável, que ofereça garantia de entrega. Caso o cliente não receba um resíduo gerado pelo servidor, o próximo resíduo computado não será correspondente à vista renderizada pelo cliente, não sendo possível renderizar a vista compensada da cena. O uso de um protocolo de transporte que provê uma conexão confiável como o TCP possui uma latência potencial [62], não sendo adequado a aplicações que são suscetíveis a esses atrasos.

CAPÍTULO 3

METODOLOGIA

Neste capítulo é descrita a metodologia empregada na construção do protótipo e as abordagens utilizadas na renderização do ambiente virtual 3D. Quatro arquiteturas cliente-servidor utilizadas na renderização da cena 3D foram propostas, estendendo o trabalho apresentado em [22]. Três dessas arquiteturas são baseadas em 3D *warping*, que são elas: arquitetura com imagens de resíduo, arquitetura com quadros completos e arquitetura combinada. Outra arquitetura implementada é a convencional, que não emprega nenhuma técnica de IBR, utilizada para a comparação dos resultados com as demais.

As bordagens propostas são formadas por uma arquitetura cliente-servidor usada na renderização remota e sob demanda da cena 3D. O cliente, ao realizar uma interação na cena (movimentar-se), requisita ao servidor a renderização da cena referente ao ponto de vista desejado, então a imagem gerada nesse processo é transmitida para o cliente a exibir.

Um ponto crucial para se ter um bom desempenho utilizando um paradigma de renderização remota é a transmissão dos dados de forma eficiente. Por conta disso, optou-se pela utilização de um protocolo que ofereça mais rapidez na transmissão dos dados, embora não garanta a entrega nem a ordem de recebimento desses dados. Baseado nisso, a arquitetura cliente-servidor é preparada para que, caso ocorra problema na transmissão dos dados, isso não afete o bom funcionamento do sistema.

Este trabalho possui semelhanças aos trabalhos descritos na seção 2.2 de Bao e Gourlay [8, 9] e Chang e Ger [16], entretanto, com a adição de novas contribuições. Um método para o aproveitamento das imagens transmitidas do servidor para o cliente foi proposto. Quando o cliente recebe imagens referentes a uma movimentação mais antiga do que a atual exibida pelo cliente, essas imagens recebidas não são descartadas e sim aproveitadas para o refinamento da qualidade das vistas exibidas no cliente. Uma abordagem foi proposta para o descarte de requisições no servidor, reduzindo sua sobrecarga devido à grande

demanda de requisições feitas pelos clientes. Um estudo foi realizado entre as diferentes técnicas de IBR definindo uma técnica apropriada para auxiliar o processo de renderização da cena. Além disso, para demonstrar e comparar o desempenho do protótipo na renderização do ambiente virtual de forma detalhada, quatro arquiteturas diferentes foram propostas.

Este capítulo está organizado como segue. Na seção 3.1 são apresentadas as arquiteturas cliente-servidor propostas para a renderização do ambiente virtual, o método para geração da informação de profundidade dos pixels na seção 3.2, as técnicas empregadas para a redução dos artefatos contidos nas vistas renderizadas via 3D *warping* na seção 3.3, a seção 3.4 descreve a abordagem utilizada na redução da sobrecarga do servidor devido ao grande número de requisições, o detalhamento de como é feita a compressão e transmissão dos dados na seção 3.5, as métricas para a avaliação de desempenho do protótipo na seção 3.6 e, por fim, na seção 3.7 são descritas as ferramentas utilizadas para a implementação do protótipo.

3.1 Arquiteturas

Esta seção apresenta as quatro arquiteturas cliente-servidor empregadas na renderização do ambiente virtual. A seguir, a subseção 3.1.1 apresenta a arquitetura que utiliza o conceito de imagens resíduo e o método de aproveitamento das imagens recebidas pelo cliente, a subseção 3.1.2 descreve a arquitetura que renderiza somente quadros completos, a subseção 3.1.3 relata a arquitetura que combina imagens de resíduo e quadros completos na renderização do ambiente virtual e, por fim, a subseção 3.1.4 descreve a arquitetura convencional.

3.1.1 Arquitetura com Imagens de Resíduo

A arquitetura com imagens de resíduo é proposta buscando realizar a transferência dos quadros renderizados no servidor e transmitidos para os clientes de forma mais eficiente. Ao invés de transmitir quadros completos renderizados a partir do modelo 3D, são

utilizadas imagens de resíduo, as quais armazenam somente os pixels utilizados para preencher os artefatos contidos nas vistas renderizadas localmente no cliente via 3D *warping*. Por armazenar uma informação esparsa, esta imagem quando comprimida consegue boas taxas de compressão comparada ao quadro completo.

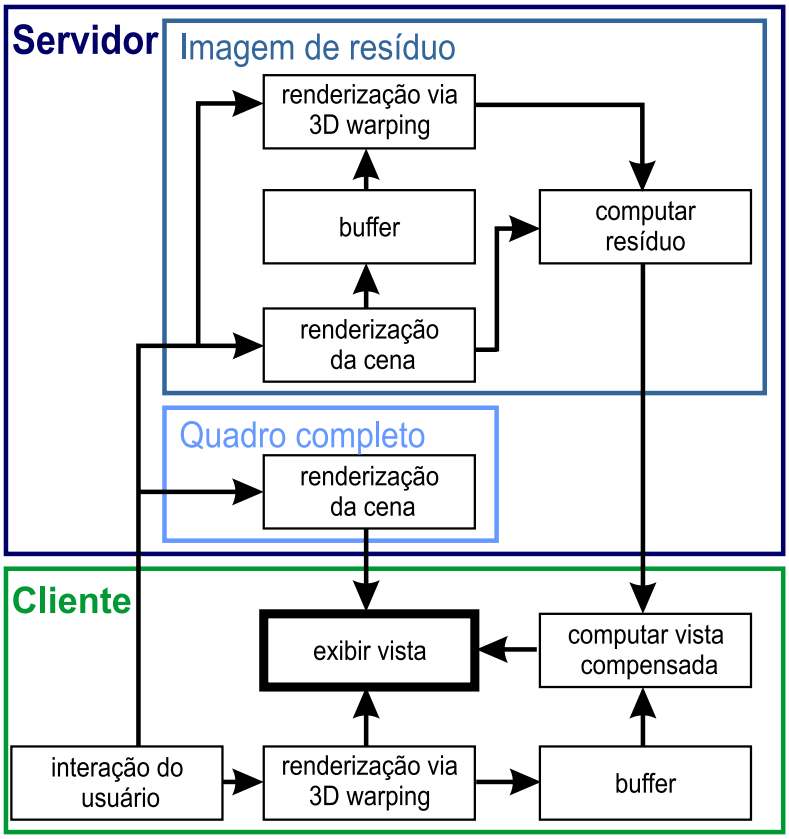


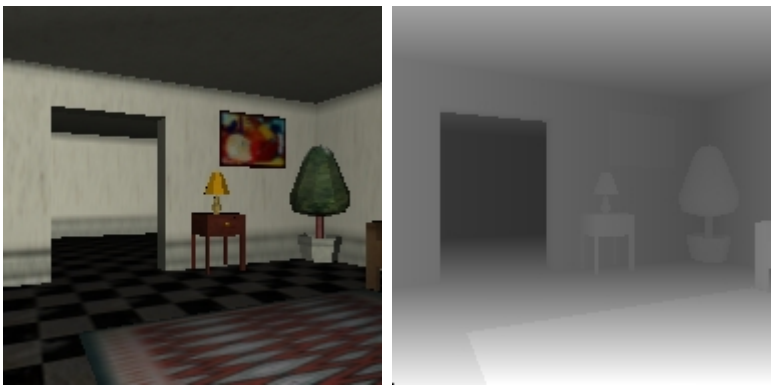
Figura 3.1: Arquitetura com Imagens de Resíduo.

A arquitetura com imagens de resíduo é ilustrada na Figura 3.1. Inicialmente, o servidor renderiza e transmite para o cliente o primeiro quadro, relativo ao ponto de vista inicial. O primeiro quadro é completo, ou seja, a imagem exata da renderização do modelo 3D. Somente o primeiro quadro é completo, posteriormente, todas as vistas renderizadas serão computadas as imagens de resíduo.

Cada quadro (imagem de resíduo ou quadro completo) renderizado e transmitido pelo servidor é composto por duas imagens, uma referente à cor dos pixels e a outra que representa a informação de profundidade correspondente, ilustradas na Figura 3.2. As áreas mais claras da imagem de profundidade representam as regiões do ambiente virtual mais próximas do plano da câmera, enquanto as áreas escuras, as regiões mais distantes.

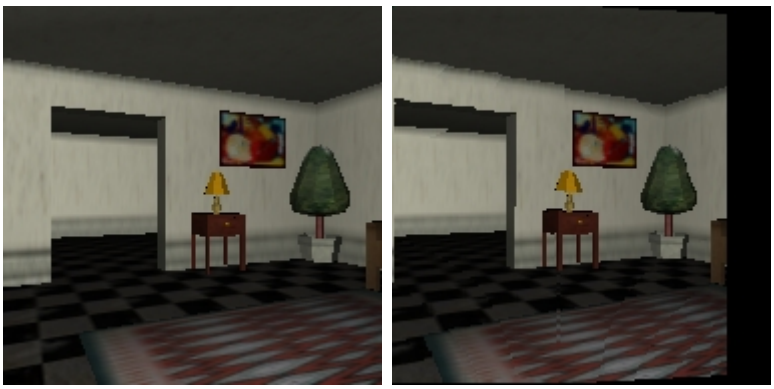
A descrição de como a informação de profundidade é gerada é detalhada na seção 3.2.

O cliente ao receber o primeiro quadro (quadro completo) apenas exibe-o na tela. Com o primeiro quadro recebido, o cliente está apto a explorar o ambiente virtual. Quando há uma interação no ambiente, primeiramente o cliente requisita ao servidor o quadro correspondente ao movimento pretendido, informando o seu ponto de vista atual e o desejado, então, o próprio cliente renderiza localmente esta requisição empregando a técnica 3D *warping*, tendo como imagem de entrada o último quadro recebido (Figura 3.3 (a)) e de saída a vista renderizada referente ao ponto de vista requerido (Figura 3.3 (b)).



(a) Imagem renderizada (b) Informação de profundidade

Figura 3.2: Quadro renderizado a partir do modelo 3D.



(a) Imagem de entrada (b) Imagem renderizada

Figura 3.3: Renderização empregando a técnica 3D *warping* relativa a um movimento de rotação para a direita.

A vista renderizada localmente é exibida na tela e copiada para um *buffer*. Assim, o cliente é capaz de renderizar vistas enquanto espera dados vindos do servidor, embora a vista exibida possa apresentar artefatos (buracos e desoclusão). Os artefatos são eli-

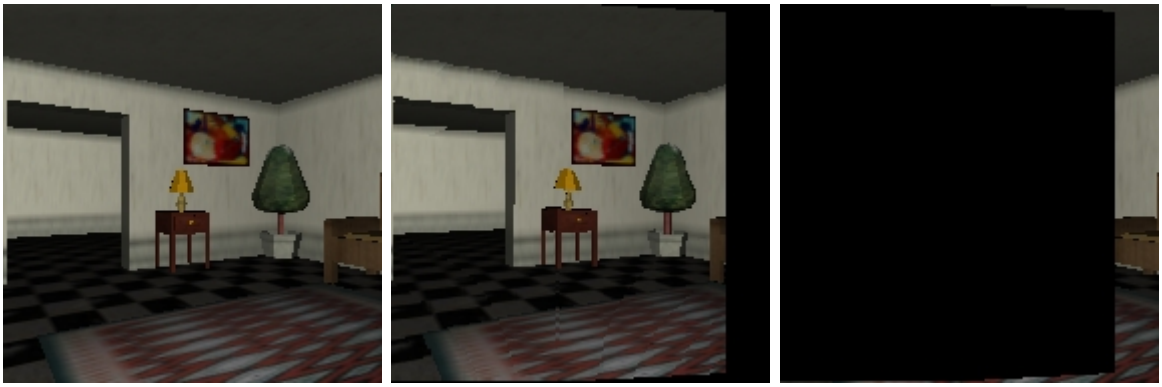
minados com a chegada da imagem de resíduo transmitida pelo servidor, criando a vista compensada.

O servidor, ao receber uma requisição, realiza a renderização da vista desejada empregando dois métodos: renderização da cena e a renderização via 3D *warping*.

A renderização da cena gera o quadro representando a vista exata com relação ao ponto de vista desejado especificado. Uma cópia desse quadro é armazenada no *buffer*, o qual será utilizado na próxima renderização via 3D *warping* e outra cópia é usada para computar o resíduo. O *buffer* local do servidor armazena somente o último quadro renderizado para cada cliente.

A renderização via 3D *warping* computa a vista idêntica a que foi renderizada localmente pelo cliente (que contém artefatos), de acordo com o ponto de vista desejado, através do uso de uma imagem fonte igual à usada na renderização pelo cliente remoto. Esta imagem é encontrada no *buffer* do próprio servidor relativo ao atual ponto de vista do cliente.

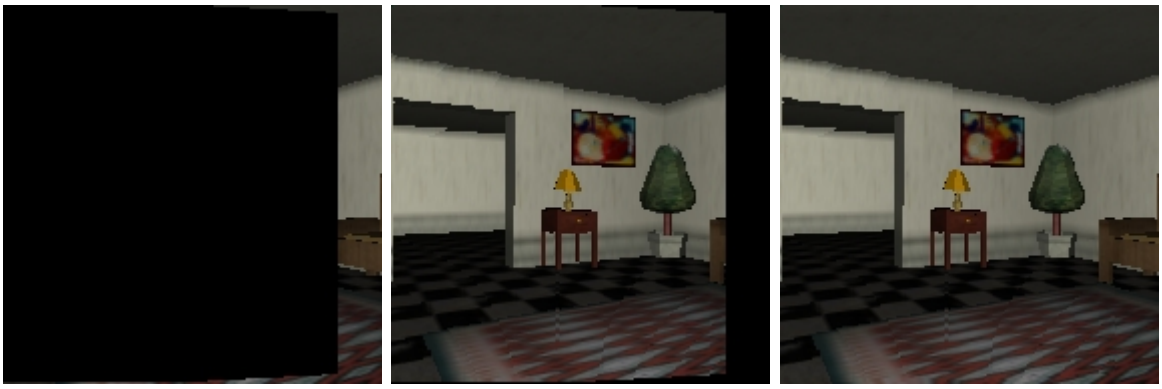
Os dois quadros renderizados por esses dois métodos são usados na geração da imagem de resíduo, como mostra a Figura 3.4. A geração desta imagem consiste em copiar os pixels que não foram projetados na imagem renderizada via 3D *warping* da imagem referente à renderização da cena (vista exata) para uma imagem nova, assim, a imagem de resíduo contém somente os pixels que não foram valorados durante o *warping*, representando os buracos e a desoclusão da vista renderizada localmente pelo cliente.



(a) Vista exata (b) Vista renderizada pelo 3D *warping* (c) Resíduo computado

Figura 3.4: Geração da imagem de resíduo.

O resíduo gerado é comprimido e enviado para o cliente, que ao receber computa a vista compensada, concatenando o resíduo (Figura 3.5 (a)) com a vista renderizada via 3D *warping* correspondente (Figura 3.5 (b)), encontrada no *buffer* local do cliente e, finalmente, a vista compensada (Figura 3.5 (c)) da cena é exibida. Quando nova vista é requerida, todo o processo descrito é realizado novamente.



(a) Resíduo (b) Vista renderizada pelo 3D *warping* (c) Vista compensada

Figura 3.5: Geração da vista compensada.

As tarefas executadas no processo de renderização do ambiente virtual a partir dessa arquitetura com resíduos são apresentadas na forma de pseudocódigo. Os algoritmos 3.1 e 3.2 referem-se às etapas de renderização local dos quadros e renderização de vista compensada no cliente, respectivamente, enquanto o algoritmo 3.3 refere-se às etapas realizadas no servidor.

```
1  Imagem imCor, imProfundidade;
2  PontoVista v_atual;
3  int id_quadro_atual;
4  void renderizar_localmente(PontoVista v_desejado)
5  {
6     Imagem imWarp;
7
8     id_quadro_atual++;
9     Requisitar_Quadro_Servidor(id_quadro_atual, v_atual, v_desejado);
10    imWarp = 3D_Warp(v_atual, v_desejado, imCor, imProfundidade);
11    Exibir_Vista(imWarp);
12    Gravar_IMG_Buffer(id_quadro_atual, imWarp, v_desejado);
13    v_atual = v_desejado;
14 }
```

Algoritmo 3.1: Algoritmo para a renderização local dos quadros no cliente na arquitetura com resíduos.

```
Imagem imCor, imProfundidade;
PontoVista v_atual;
int id_quadro_atual;
void quadro_recebido(int id_quadro, Imagem imCorRecebida,
                     Imagem imProfRecebida)
{
   Imagem imWarp, imCompensada;
   PontoVista v_img;
   if(! Le_IMG_Buffer(id_quadro, &imWarp, &v_img))
      return; //quadro recebido muito antigo

   Remove_IMG_Buffer(id_quadro); //remove este quadro e os mais antigos
   imCompensada = Computar_Vista_Compensada(imWarp, imCorRecebida);
   imCor = imCompensada;
   imProfundidade = imProfRecebida;

   if(id_quadro == id_quadro_atual)
      Exibir_Vista(imCompensada);
   else //aproveitamento dos resíduos, resíduo recebido referente
   { //ao um ponto de vista mais antigo que o atual
      imWarp = 3D_Warp(v_img, v_atual, imCompensada, imProfRecebida);
      Exibir_Vista(imWarp);
   }
}
```

Algoritmo 3.2: Algoritmo para a renderização de vista compensada no cliente na arquitetura com resíduos.

```
void quadro_requisitado(int id_cliente, int id_quadro, PontoVista v_atual,
                        PontoVista v_desejado)
{
   Imagem imCor, imProfundidade, imExataCor, imExataProf;
   Imagem imWarp, imResiduo;
   char *bytesCor, *bytesProf;
   if(! Le_IMG_Buffer(id_cliente, v_atual, &imCor, &imProfundidade))
      Renderizar(v_atual, &imCor, &imProfundidade);

   imWarp = 3D_Warp(v_atual, v_desejado, imCor, imProfundidade);
   Renderizar(v_desejado, &imExataCor, &imExataProf);
   imResiduo = Computar_Residuo(imExataCor, imWarp);
   Comprimir_Imagens_JPEG(imResiduo, imExataProf, &bytesCor, &bytesProf);
   Enviar_Dados(id_cliente, id_quadro, bytesCor, bytesProf);
   Gravar_IMG_Buffer(id_cliente, imExataCor, imExataProf);
}
```

Algoritmo 3.3: Algoritmo para a renderização dos quadros no servidor na arquitetura com resíduos.

Aproveitamento das Imagens de Resíduo

O *buffer* no cliente tem uma função importante na arquitetura. Caso o resíduo recebido seja relativo a uma movimentação mais antiga do que a atual exibida pelo cliente

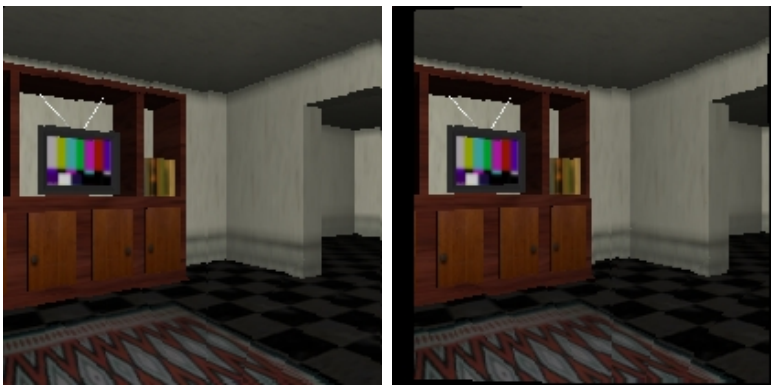
(enquanto o servidor renderiza e transmite a vista, o cliente pode renderizar várias vistas localmente). Esse resíduo não é descartado e sim utilizado na criação da vista compensada correspondente a essa movimentação.

Uma nova vista é localmente renderizada para o atual ponto de vista do cliente empregando esta imagem, refinando a qualidade da vista exibida conforme os resíduos são recebidos.

A Figura 3.6 mostra a renderização de vistas baseadas em uma única imagem de entrada referente a três movimentos de rotação para a esquerda (*Warp* 1, 2 e 3). Nas vistas renderizadas é observado que a presença de artefatos aumenta ou diminui conforme o grau de rotação de cada movimento.

(a) *Warp* 1 (b) *Warp* 2 (c) *Warp* 3

Figura 3.6: Vistas renderizadas via 3D *warping* referente a três movimentos de rotação para a esquerda.



(a) Vista Compensada *Warp* 2 (b) *Warp* 3

Figura 3.7: Aproveitamento de resíduos referentes a movimentos mais antigos que o atual ponto de vista do cliente.

Supondo que o cliente renderizou esses três movimentos localmente (Figura 3.6) e

recebeu o resíduo referente ao quadro *Warp* 2, este não é descartado, sendo utilizado para gerar a vista compensada desse quadro (Figura 3.7 (a)). Então, a vista relativa ao movimento *Warp* 3 é renderizada (Figura 3.7 (b)) empregando como imagem de entrada a imagem compensada processada. A diferença entre as vistas exibidas pelo cliente sem e com o aproveitamento de resíduos, são observadas nas Figura 3.6 (c) e Figura 3.7 (b) respectivamente.

3.1.2 Arquitetura com Quadros Completos

A arquitetura com imagens de resíduo busca realizar a transmissão dos quadros pela rede de forma mais eficiente e a visualização dos quadros no cliente é dada pela renderização das vistas através da técnica 3D *warping* ou pelas imagens compensadas geradas com a chegada do resíduo. O problema dessa abordagem é que a qualidade das vistas exibidas no cliente vai se degradando conforme o número de quadros renderizados, pelo fato de que as vistas renderizadas pela técnica 3D *warping* permitem a projeção (*warping*) dos pixels de uma imagem de entrada para imagem de saída, mas não representam a vista exata da cena.

Além disso, é empregada a técnica de *splatting*, que expande os pixels mapeados da imagem fonte para a destino, reduzindo o número de artefatos, de acordo com uma interpolação. Obviamente, o uso de interpolação contribui para a degradação da vista gerada.

(a) Vista exata (b) Vista compensada (c) Diferença

Figura 3.8: Diferença entre a vista compensada e vista exata.

A diferença entre uma vista exata renderizada do modelo 3D e a vista compensada criada a partir da renderização via 3D *warping* e a concatenação do resíduo é ilustrada na Figura 3.8. A imagem de entrada utilizada para computar a vista compensada é um quadro completo (vista exata renderizada a partir do modelo 3D), ou seja, essa diferença tende a ficar maior, degradando a qualidade, quando as imagens de entrada para renderização via 3D *warping* são imagens compensadas, por já possuírem uma qualidade inferior, comparada com um quadro completo.

Com base nessas premissas é proposta a arquitetura com quadros completos, apresentada na Figura 3.9. O cliente ao movimentar-se no ambiente virtual requisita a vista desejada ao servidor remoto e renderiza localmente o ponto de vista desejando empregando 3D *warping*. A imagem produzida localmente é exibida na tela. O servidor, ao invés de gerar as imagens de resíduo, apenas renderiza o quadro a partir do modelo 3D, juntamente com a informação de profundidade dos pixels e envia para o cliente, que ao receber exibe este quadro na tela, substituindo o quadro renderizado localmente, uma vez que as vistas renderizadas pela técnica 3D *warping* podem apresentar artefatos e a vista transmitida pelo servidor não possui, porque foi renderizada a partir do modelo 3D.

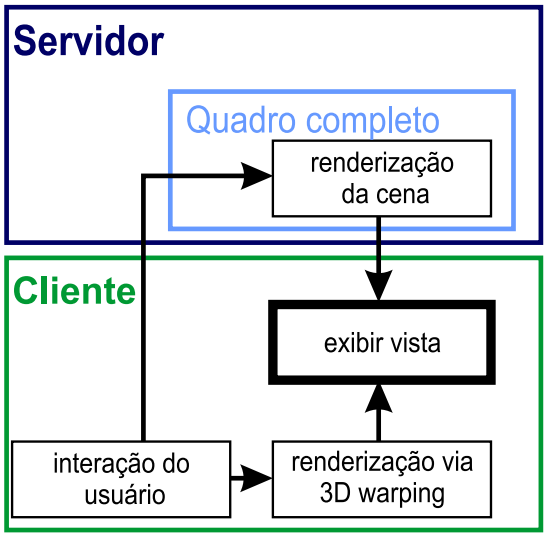


Figura 3.9: Arquitetura com Quadros Completos.

Essa abordagem minimiza consideravelmente o problema de degradação da qualidade das vistas renderizadas encontrado na arquitetura baseada em imagens de resíduos, pois poucas vistas são renderizadas localmente até que um quadro completo livre de artefatos

seja recebido. Entretanto, com o uso de quadros completos há um maior tráfego de rede, pois o quadro completo possui um tamanho maior em comparação com as imagens de resíduo.

3.1.3 Arquitetura Combinada

O emprego da arquitetura com imagens de resíduo maximiza o uso da rede, uma vez que as imagens de resíduo possuem tamanhos menores em comparação com os quadros completos, mas apresenta o problema de degradação da qualidade das vistas. Em oposição, a arquitetura com quadros completos reduz o problema de degradação da qualidade das vistas, mas os quadros possuem tamanhos maiores.

Com base nessas informações é proposta a arquitetura combinada, que combina as duas arquiteturas, quadros completos e imagens de resíduo, fazendo com que, a cada determinado número de vistas renderizadas empregando a abordagem de resíduo, seja requisitado um quadro completo, minimizando o problema da degradação da qualidade da vista exibida e menos dados são transmitidos entre servidor e cliente.

Esta arquitetura é apresentada na Figura 3.1 da subseção 3.1.1 referente à arquitetura com imagens de resíduo. Embora, a mesma figura represente ambas as arquiteturas, o comportamento é diferente. Na arquitetura com imagens de resíduo, somente o primeiro quadro é completo, posteriormente, todas as renderizações são computadas com imagens de resíduo. Na arquitetura combinada, quadros completos e imagens de resíduo são renderizadas durante a renderização do ambiente virtual conforme a requisição do cliente.

3.1.4 Arquitetura Convencional

A arquitetura convencional é ilustrada na Figura 3.10. Esta abordagem utiliza o paradigma tradicional de renderização remota. O cliente ao movimentar-se no ambiente virtual requisita a vista ao servidor remoto e aguarda sua chegada ociosamente, para então realizar uma nova interação no ambiente virtual. O servidor ao receber a requisição, renderiza o quadro a partir do modelo 3D e transmite para o cliente, que ao receber exibe-o na tela.

Nesta arquitetura, nenhuma técnica de IBR é aplicada. Ela é proposta com o objetivo de comparar com as demais arquiteturas que empregam a técnica 3D *warping*, buscando uma análise detalhada dos resultados na renderização do ambiente virtual com e sem o uso de IBR.

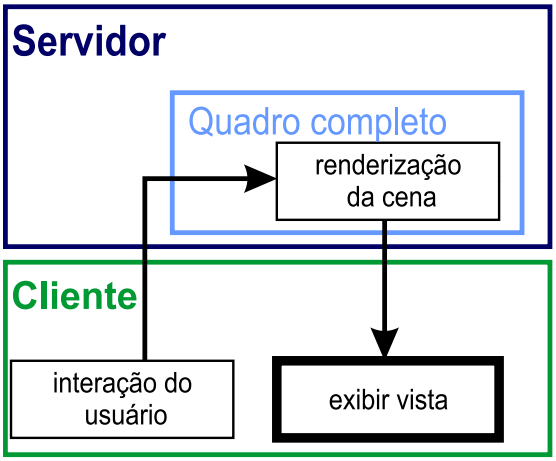


Figura 3.10: Arquitetura Convencional.

3.2 Geração da Informação de Profundidade dos Pixels

Cada quadro renderizado no servidor é composto por duas imagens, uma referente à cor dos pixels e outra representando informação de profundidade correspondente, como mostra a Figura 3.2 (seção 3.1.1). Na Figura 3.2 (b), os pixels com cores mais claras representam pontos mais próximos, enquanto os mais escuros os mais distantes.

Os valores de profundidade são lidos do *buffer* de profundidade computado pela biblioteca gráfica OpenGL e calculados pela seguinte fórmula, definida em [48] como:

$$\text{profundidade}\,(i) = 1 - \text{zBuffer}\,(i) \;*\; \frac{\text{zFar} - \text{zNear}}{\text{zFar}} \tag{3.1}$$

em que *profundidade* é *buffer* que armazena os valores de profundidade computados, i é o índice do *buffer*, tal que $0 \leq i \leq$ tamanho do *buffer* -1, *zBuffer* é o *buffer* de profundidade computado pelo OpenGL, que armazena valores no intervalo de 0 a 1, *zFar* e *zNear* são valores constantes que representam a distância do observador até o plano de corte mais afastado e mais próximo, respectivamente, com relação ao eixo z no sistema

de coordenadas do mundo.

Após o cálculo dos valores de profundidade, a imagem final é gerada com a escala desses valores para o intervalo numérico de 0 a 255 através da seguinte fórmula:

$$\mathrm{img}\,(x,\ y) = \frac{(\mathrm{profundidade}\,(i)\ -\ \mathrm{pMin})\ *\ 255}{\mathrm{pMax}\ -\ \mathrm{pMin}} \tag{3.2}$$

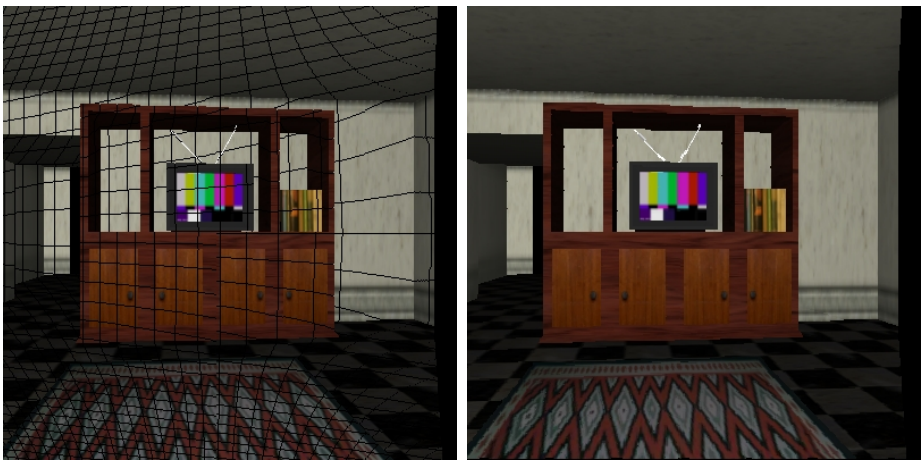
em que *img* é a imagem onde a informação de profundidade é armazenada, x e y são as coordenadas da imagem, *pMin* e *pMax* são os valores mínimo e máximo de profundidade calculados pela equação 3.1, respectivamente. Os valores *pMin* e *pMax* podem ser variáveis, ou seja, a cada imagem de profundidade computada podem ser diferentes, uma vez que o valores de profundidade variaram de acordo com o ponto de vista.

3.3 Redução de Artefatos

Uma desvantagem da técnica 3D *warping* são os buracos e a desoclusão presentes nas vistas renderizadas, denominados de artefatos. Os buracos ocorrem devido à diferença de amostragem dos pixels durante o *warping* e a desoclusão são regiões da imagem destino onde os pixels não foram mapeados, como mostra a região escura do lado direito das imagens apresentadas na Figura 3.11. Este problema é detalhado na subseção 2.1.3.3.

Para reduzir a presença de buracos foi utilizada da técnica de *splatting* (o funcionamento desta técnica foi descrito na subseção 2.1.3.3) que consiste em expandir os pixels durante o *warping* da imagem fonte para a destino. O resultado da aplicação dessa técnica pode ser observado na Figura 3.11, em que foi utilizado um *splat* de dimensão 1,0×1,0 pixels.

Para reduzir a presença da desoclusão é empregada a abordagem apresentada em [7], que consiste em utilizar uma imagem de entrada (imagem fonte) maior que a exibida pelo cliente durante a renderização das vistas utilizando 3D *warping*. A Figura 3.12 ilustra o funcionamento dessa técnica. A imagem exibida pelo cliente possui dimensão de 400×400 pixels (Figura 3.12 (b)). Para renderizar as vistas empregando 3D *wapring* são usadas imagens de entrada de 500×500 pixels (Figura 3.12 (a)).



(a) Sem o uso da técnica de *splatting* (b) Com o uso da técnica de *splatting*

Figura 3.11: Aplicação da técnica de *splatting* nas vistas renderizadas via 3D *warping*.

(a) Vista renderizada pela técnica 3D *warping* (b) Vista exibida no cliente

Figura 3.12: Renderização via 3D *warping* empregando uma imagem fonte maior que a área de exibição dos quadros no cliente.

O cliente ao receber a imagem com dimensão maior do que a área de exibição dos quadros, nesse caso, apenas desloca 50 pixels nas coordenadas x e y no momento da exibição desse quadro. Assim, somente a região delimitada pelo quadro tracejado observado na Figura 3.12 (a) é mostrada na tela e a desoclusão ilustrada no lado direito da imagem renderizada por 3D *wapring* não está presente na imagem exibida pelo cliente. Entretanto, dependendo do movimento realizado, esta abordagem pode não eliminar por completo a desoclusão, por exemplo, caso o cliente renderize 3 movimentos de rotação para esquerda, o grau de rotação é maior e, consequentemente, a desoclusão vai ser maior e essa abordagem vai minimizar o problema mas não eliminá-lo.

3.4 Redução da Sobrecarga do Servidor

Um problema que pode ocorrer utilizando um paradigma de renderização remota é a sobrecarga do servidor devido a um grande número de requisições de quadros a serem renderizados, em um curto intervalo de tempo.

Esse problema surge quando o tempo de renderização de cada quadro no servidor é maior que o intervalo de tempo entre cada requisição, ou seja, se o servidor possui um tempo médio de renderização de 200 milissegundos e a cada 190 milissegundos novas requisições são feitas pelos clientes, então, os quadros renderizados terão um atraso médio de 10 milissegundos, multiplicado pelo número de renderizações relativas as requisições que se acumularam nos *buffers* de requisições do servidor devido à espera para serem processadas.

O acúmulo de requisições nos *buffers* do servidor é ilustrado na Figura 3.13. O servidor contém um *buffer* de requisições para cada cliente $(C1, C2, ..., Cn)$. Neste exemplo, enquanto o servidor está renderizando o quadro referente à requisição do cliente $C1$, $C2$ requisitou 7 novas vistas e Cn requisitou 5 vistas.

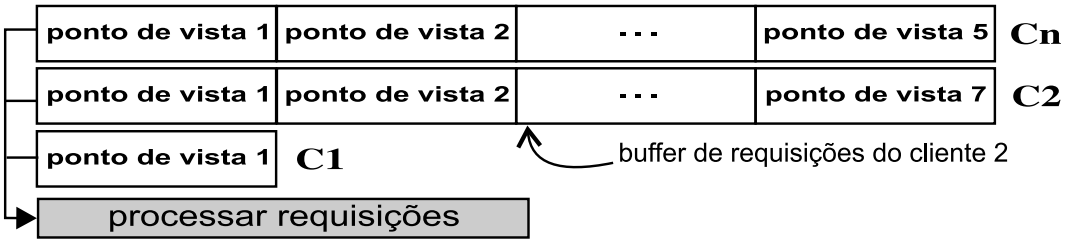


Figura 3.13: *Buffers* de requisições do servidor.

O atraso na renderização da requisição referente ao ponto de vista 7 do cliente $C2$ corresponde à quantidade de requisições feitas anteriormente a esta e que estão acumuladas no *buffer* (nesse caso 6) multiplicado pelo tempo de renderização de cada quadro.

Para minimizar esse problema pode ser renderizada somente a última vista requisitada de cada cliente. Uma vez que as vistas foram renderizadas localmente pelo cliente via 3D *warping* enquanto ele esperava os quadros requisitados ao servidor, o cliente necessita somente do resíduo para gerar a vista compensada ou o quadro completo para ser exibido, referente à última vista.

Assim, no exemplo mostrado na Figura 3.13, somente os quadros referentes aos pontos de vista 7 do cliente $C2$ e 5 do cliente Cn são renderizados, as demais requisições são descartadas, acelerando o processo de renderização e não prejudicando a qualidade de exibição das vistas no cliente e permitindo que o servidor possa atender um maior número de clientes simultaneamente.

3.5 Compressão e Transmissão dos Dados entre Cliente e Servidor

Para a compressão das imagens renderizadas no servidor e posteriormente transmitidas para o cliente foi adotado o método de compressão JPEG (*Joint Photographic Experts Group*) [31]. Esse método de compressão é muito utilizado em diferentes tipos de imagens e possui o grau de compressão ajustável, tal que quanto maior o grau de compressão pior será a qualidade da imagem comprimida e vice-versa.

A maioria das aplicações multimídias em tempo real baseada em Internet adota como protocolo de transporte o UDP (*User Datagram Protocol*) [64]. Comparado com o protocolo TCP (*Transmission Control Protocol*) [62], o UDP não possui atraso com relação à retransmissão de pacotes, por isso, seu uso se torna mais adequado em aplicações que exigem a transmissão de dados em tempo real ou são suscetíveis a esses atrasos [64].

Neste trabalho, o protocolo UDP foi adotado por permitir a transmissão mais rápida dos pacotes que o TCP. Entretanto, o protocolo TCP foi também utilizado na comparação dos resultados.

Inicialmente, as imagens geradas no servidor são fragmentadas em pacotes com tamanho fixo e, posteriormente, transmitidas para o cliente. O cliente, conforme recebe os dados, armazena-os ordenadamente e quando todos pacotes referentes à imagem arbitrária são recebidos, os dados são concatenados formando a imagem, que é exibida na tela. Caso um determinado pacote não seja recebido de uma determinada imagem, esta imagem não é exibida e os pacotes recebidos referente a este quadro são descartados.

Como os pacotes podem chegar de forma desordenada, cada pacote contém um pequeno cabeçalho que orienta o processo de geração das imagens a partir dos dados recebidos no cliente, informando a ordem em que os dados recebidos devem ser concatenados

e classificando-os nas imagens que estão sendo transmitidas durante o processo de renderização dos quadros que eles pertencem.

Segundo Stevens [55], o tamanho máximo do *buffer* que armazena os dados de um datagrama UDP é de 65507 *bytes*. Entretanto, muitas aplicações adotam um limite muito inferior do valor máximo, devido a limitações especificadas nas bibliotecas de *socket*[1] e limitações na implementação do *kernel* do TCP/IP [55].

O tamanho máximo de um datagrama que um *host* é obrigado a receber é de 576 *bytes* e muitas aplicações que utilizam o protocolo UDP restringem o tamanho do datagrama UDP para 512 *bytes* ou menos [55]. Com base nessas premissas, o tamanho adotado do *buffer* que armazena os dados do datagrama UDP neste trabalho é de 500 *bytes*.

3.6 Métricas para Avaliação de Desempenho

Nesta seção são descritas as métricas utilizadas para avaliar o desempenho do protótipo. A avaliação do desempenho da versão cliente do protótipo implementado é baseada no tempo de renderização local e no tempo de renderização remota, enquanto no servidor é baseado no tempo de renderização das vistas requisitadas. O tempo, em todas as etapas do processo de avalização do protótipo, é medido em milissegundos.

Na renderização local é avaliado o tempo gasto para produzir as vistas empregando a técnica 3D *warping*. Na renderização remota é avaliado o tempo gasto desde a requisição da vista ao servidor até a chegada das imagens que foram renderizadas remotamente.

Para avaliar tanto o servidor quanto o cliente, o tempo de renderização das vistas é medido nas diferentes arquiteturas propostas neste trabalho, porque as tarefas executadas no processo de renderização são diferentes em cada arquitetura, conforme descrito nas seções 3.1.1 e 3.1.2, gerando-se um tempo de renderização específico para cada uma.

O cálculo do tempo de renderização local utilizando 3D *warping* é dado pela equação

$$T_{local} = T_{wf} - T_{wi} \tag{3.3}$$

[1] *Socket* pode ser utilizado em ligações de redes de computadores para estabelecer uma conexão bidirecional de comunicação entre dois programas [54].

em que T_{local} é o tempo gasto resultante do processo de renderização local, T_{wf} é o tempo final da execução da técnica 3D *warping* e T_{wi} é o tempo inicial.

Para computar o tempo de renderização remota é utilizada a fórmula

$$T_{remoto} = T_c - T_r \tag{3.4}$$

em que T_{remoto} é o tempo gasto resultante do processo de renderização remota, T_c é o tempo de chagada do quadro requisitado ao servidor e T_r é o tempo no momento da requisição do quadro.

No cálculo do valor de T_{remoto}, de forma implícita são computados os tempos da transferência da requisição do cliente até o servidor, o tempo de espera que a requisição pode sofrer até que esta seja processada (devido ao problema de acúmulo de requisições, detalhado na seção 3.4), tempo de processamento do quadro pelo servidor e, finalmente, o tempo de transferência das imagens renderizadas do servidor para o cliente, formando o T_{remoto}, que representa o tempo desde a requisição até a chegada do quadro.

A avaliação do desempenho do servidor no processo de renderização das vistas requisitadas pelos clientes é dada pela equação

$$T_{servidor} = T_{rf} - T_{ri} \tag{3.5}$$

em que $T_{servidor}$ é o tempo gasto resultante do processo de renderização do quadro no servidor, T_{rf} é o tempo final do processo de renderização e T_{ri} é o tempo inicial.

De forma semelhante ao cálculo do tempo de renderização remota, o tempo de renderização no servidor é formado implicitamente pelos tempos das tarefas executadas para computar o quadro, como renderização da cena e informação de profundidade, renderização via 3D *warping*, computação da imagem de resíduo e compressão das imagens. Algumas dessas tarefas, dependendo da arquitetura utilizada, podem estar presentes ou não.

3.7 Ferramentas Utilizadas na Implementação do Protótipo

Nesta seção são descritas as ferramentas utilizadas para a implementação do protótipo. Um dos critérios para a definição das ferramentas foi utilizar apenas pacotes livres e gratuitos.

Na subseção 3.7.1 são descritas as linguagens de programação utilizadas na implementação do protótipo, as ferramentas empregadas na renderização da cena remota e local na subseção 3.7.2 e, por fim, na subseção 3.7.3 são detalhadas as ferramentas utilizadas na compressão e transmissão dos dados.

3.7.1 Linguagens de Programação

Para o desenvolvimento do protótipo foram utilizadas duas linguagens de programação, o cliente foi construído usando Java (versão 1.6) e o servidor, C++ (g++ versão 4.3.2). Ambas as linguagens suportam o paradigma de orientação a objetos.

C++ é uma linguagem de programação de alto nível que suporta multiparadigmas, utilizada para codificar diferentes tipos de aplicações. Seu uso é popular por gerar executáveis que possuem um grande desempenho, adequado para o servidor que necessita renderizar a cena e atender a múltiplos clientes.

A linguagem de programação orientada a objetos Java tem uma grande vantagem por ser independente de plataforma. Os códigos escritos utilizando esta linguagem são executados em uma máquina virtual, diferente de C++ que gera um executável nativo da plataforma em que foi compilado. A máquina virtual Java é disponível em várias plataformas.

Outras facilidades que a linguagem Java oferece são seus recursos voltados à Internet, sendo que os programas podem ser executados diretamente nos navegadores, como o Applet Java que permite ser incluído em uma página HTML.

Tradicionalmente, as linguagens interpretadas como Java não oferecem um grande desempenho, mas seus recursos destinados à Internet, como a possibilidade de executar os programas desenvolvidos no navegador, agregado com sua portabilidade, são propriedades

adequadas para o desenvolvimento do cliente.

3.7.2 Ferramentas Utilizadas na Renderização Remota e Local

Na renderização da cena no servidor (renderização remota), os modelos tridimensionais são carregados utilizando a biblioteca *Open ActiveWrl* [58] (versão 0.9.8) e renderizados utilizando OpenGL [42] (versão 2.1).

Open ActiveWrl é uma biblioteca de código aberto e multiplataforma desenvolvida na linguagem C++ para manipulação de modelos 3D no formato VRML e X3D. Esta biblioteca carrega o grafo de cena especificado no modelo VRML ou X3D que contém as informações utilizadas na renderização do ambiente virtual, tais como geometria, texturas, iluminação e câmera, então, a cena é renderizada baseada nessas informações carregadas utilizando OpenGL.

A biblioteca gráfica OpenGL é uma interface de *software* para o *hardware* gráfico [60]. Com esta biblioteca é possível criar (desenhar) objetos 2D e 3D e ambientes virtuais a partir de primitivas geométricas (como triângulos, cubos). Esta biblioteca é multilinguagem, multiplataforma e livre.

OpenGL é utilizada em diferentes tipo de aplicações, como científicas, jogos, simuladores e realidade virtual, possuindo uma vasta documentação encontrada em livros, exemplos disponíveis na Internet e tutoriais.

Como o servidor somente precisa renderizar os quadros e a visualização destes é feita pelo cliente, utilizou-se GLX [59] (*OpenGL Extension to the X Window System*, versão 1.3) por permitir que a renderização da cena via OpenGL seja feita sem uso de janelas (*Off-screen Rendering*) e acelerada via *hardware* gráfico, agilizando a tarefa de renderização.

GLX permite a conexão do OpenGL com o sistema de janelas X[2]. Um dos propósitos de GLX é criar um contexto de renderização do OpenGL e associá-lo com uma área de desenho, que pode ser uma janela (*onscreen*) ou um *buffer* na memória (*offscreen*).

GLX dispõe do *GLXPbuffer* [59] que permite que os quadros renderizados sejam alocados em um *buffer* na memória, ao invés de serem exibidos na tela, como nos métodos

[2]Protocolo para interface gráfica do usuário nos sistemas Unix e assemelhados, como o Linux [61].

tradicionais, além do processo de renderização por esse método ser acelerado via *hardware* [59].

No cliente, a renderização local dos quadros utilizando a técnica de IBR 3D *warping* foi feita através da adaptação do código fonte disponível em [37] para o protótipo desenvolvido.

3.7.3 Ferramentas Utilizadas na Compressão e Transmissão dos Dados

Os quadros renderizados no servidor são comprimidos utilizando o método de compressão JPEG com o uso da biblioteca disponível em [29] (versão 6b). Esta biblioteca possui uma boa documentação, além de ser código aberto e multiplataforma, fatos que justificam seu uso.

Depois de comprimidos, os quadros são transmitidos do servidor para o cliente utilizando-se a biblioteca de *socket* C++ Sockets Library [27] (versão 2.3.2). Esta biblioteca permite a comunicação entre *sockets* TCP e UDP, escrita em C++ e que suporta as plataformas Windows e Linux. Além disso, possui exemplos de sua utilização e completa documentação disponível em [27].

No cliente, a decodificação das imagens JPEG e a transmissão dos dados são feitas utilizando-se os pacotes nativos da linguagem Java, já dispondo de exemplos de utilização na própria documentação da linguagem.

CAPÍTULO 4

RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir da avaliação do protótipo desenvolvido baseado na metodologia proposta. O protótipo possibilita a renderização do ambiente virtual empregando as quatro arquiteturas descritas na seção 3.1. Dois experimentos foram realizados e em ambos três testes foram abordadas: a avaliação do desempenho das arquiteturas baseadas em 3D *warping*, a comparação entre uma arquitetura que emprega o método de descarte de requisições (seção 3.4) e outra que não descarta requisições e a avaliação da arquitetura convencional.

O teste referente às arquiteturas baseadas em 3D *warping* compreendeu as três arquiteturas, a com imagens de resíduo, a com quadros completos e a combinada, em que foram apresentados e comparados os resultados da renderização remota e local do ambiente virtual. No teste relativo ao descarte de requisições, a arquitetura com imagens de resíduo foi avaliada, em que foram comparados os resultados obtidos com e sem a utilização do método de descarte. O teste da arquitetura convencional avaliou o desempenho dessa arquitetura na renderização remota dos quadros e, além disso, comparou o emprego de dois protocolos de transporte diferentes, o UDP e o TCP na transmissão dos quadros.

Os três testes realizados nos dois experimentos foram iguais, mudando apenas os recursos computacionais dos clientes para cada experimento, permitindo uma comparação dos resultados obtidos na renderização do ambiente virtual em cada teste com relação à capacidade computacional disponível para a execução das tarefas no cliente.

A avaliação do protótipo foi realizada com base em uma simulação da interação dos usuários no ambiente virtual. Os clientes (usuários) percorrem um caminho pré-definido no ambiente virtual, formado por uma sequência pré-estabelecida de movimentos (esquerda, direita, etc) realizados pelo observador no ambiente, requisitando as vistas referentes a este caminho ao servidor. A definição do caminho é detalhada na seção 4.1. O

tempo entre cada requisição é um parâmetro definido no início da simulação. Os clientes iniciam a simulação praticamente ao mesmo tempo através de uma mensagem enviada do servidor para todos os clientes informando-os o início da simulação.

Nos experimentos realizados, utilizou-se um modelo 3D de 1334 faces poligonais, o qual representa o ambiente virtual. A rede utilizada nos experimentos foi uma rede local (LAN) com largura de banda de 100 Mbps (*megabits* por segundo). O nível de compressão das imagens JPEG renderizadas no servidor foi definido em 50% de qualidade. O formato de cor das imagens coloridas foi o RGB e as imagens que contêm a informação de profundidade foram representadas em níveis de cinza.

Por padrão, em todos os testes foi utilizado o protocolo de transporte UDP, com exceção do teste da arquitetura convencional, em que foi usado o TCP para comparar os resultados entre os dois protocolos. Além disso, o método de descarte de requisição (detalhado na seção 3.4) também foi utilizado em todos os testes, com exceção dos testes em que foi comparado as arquiteturas que empregam ou não este método e na arquitetura convencional.

A dimensão das imagens foi de 400×400 pixels na arquitetura convencional e, nas demais baseadas em 3D *warping*, foi de 500×500 pixels. A diferença na dimensão dos quadros entre a arquitetura convencional e as baseadas em 3D *warping* está relacionada à redução de artefatos descrita na seção 3.3.

Para a ilustração do desempenho da renderização remota ou local do ambiente virtual nas diferentes arquiteturas propostas foram utilizados gráficos, que mostram o tempo de renderização obtido pelos clientes com relação a um instante de tempo durante a simulação. Nesses gráficos, o tempo de renderização exibido relativo a cada arquitetura avaliada refere-se ao tempo médio de renderização de todos os cliente envolvidos na simulação.

A seguir, a seção 4.1 apresenta o módulo cliente do protótipo e a especificação da simulação empregada nos experimentos, a seção 4.2 descreve a configuração e o desempenho do servidor, as seções 4.3 e 4.4 apresentam os experimentos realizados e, por fim, na seção 4.5 são discutidos os resultados obtidos nos experimentos.

4.1 Módulo Cliente do Protótipo e Especificação da Execução da Simulação

O módulo cliente do protótipo é apresentado na Figura 4.1. A interface é dividida em cinco partes. A parte superior possui três botões, dois para requisitar a conexão e desconexão com o servidor remoto e outro botão para sair do programa.

Figura 4.1: Interface gráfica do módulo cliente do protótipo.

O protocolo de transporte adotado foi o UDP e este não é orientado à conexão. O termo conexão utilizado neste trabalho refere-se à autenticação do cliente no servidor. Quando o cliente clica no botão conectar, uma mensagem é envida ao servidor, que aloca os *buffers* utilizados para este cliente. Quando o servidor recebe a mensagem de desconexão, ele apenas desaloca esses *buffers*.

A parte inferior da interface informa ao usuário o estado da conexão e os tempos de renderização remota e local dos últimos quadros renderizados. A região do lado esquerdo da interface exibe as imagens renderizadas e possui os botões de direção (esquerda, direita,

frente, etc.) que possibilitam a interação no ambiente virtual. A dimensão da área em que os quadros renderizados são exibidos é de 400×400 pixels.

O lado direito da interface do usuário é composta pelos componentes que permitem a escolha da arquitetura desejada para a renderização da cena e os parâmetros da simulação. Esses dados devem ser especificados antes da conexão do cliente.

Para especificar a execução de uma simulação arbitrária, os parâmetros são iguais em todos os clientes, com exceção do identificador do cliente e o caminho onde o arquivo contendo os tempos de renderização dos quadros durante a simulação é gravado, que podem ser diferentes para cada cliente.

Para a simulação de uma arquitetura são especificados os parâmetros da arquitetura desejada[1], se o servidor deve ou não utilizar o método de descarte de requisições e o tempo em milissegundos entre cada requisição.

Com o objetivo de que a simulação inicie e termine ao mesmo tempo para todos os clientes conectados, implementou-se um esquema em que o servidor envia uma mensagem para todos os clientes informando-os o início e término de uma simulação. Primeiramente, todos os clientes se conectam ao servidor, então, apenas um dos clientes conectados faz as requisições de início ou de término da simulação (botões presentes na interface) ao servidor, que ao receber esta mensagem comunica todos os clientes do término ou início da simulação.

Na simulação, para todas as vistas renderizadas, primeiramente, o cliente requisita a renderização da vista ao servidor, renderiza localmente e, posteriormente, aguarda o intervalo de tempo especificado nos parâmetros da simulação para a próxima requisição referente ao caminho que está sendo percorrido no ambiente virtual.

Então, o tempo de intervalo entres as requisições das vistas realizadas na simulação é o tempo especificado nos parâmetros da simulação somado com o tempo de renderização local. Em experimentos em que o tempo de renderização local é maior, as requisições terão intervalos maiores comparado com outro experimento que renderize localmente mais

[1] Caso seja selecionada a arquitetura combinada também deve ser especificado o valor que informa a cada quantos quadros renderizados, utilizando a abordagem de resíduo, deve-se renderizar um quadro completo.

rápido e que especifique o mesmo tempo entre as requisições nos parâmetros da simulação.

As requisições das vistas feitas pelos clientes são referentes ao caminho em que eles percorrem no ambiente virtual. Este caminho é apresentado na Tabela 4.1. Na tabela, por exemplo, *r_direita(5)* e *t_frente(4)* informam que serão realizados movimentos de rotação para a direita e de translação para frente, respectivamente. A letra *r* antes do denominador do movimento indica movimento de rotação e *t* de translação. O número entre parênteses informa a quantidade de movimentos a serem realizados.

No caminho percorrido pelos clientes, os primeiros 5 movimentos são de rotação para direita, especificados na linha 1 e coluna *Sequência 1* na Tabela 4.1. Posteriormente, mais 4 movimentos de translação para frente são realizados dadas a linha 2 e a coluna *Sequência 1*. Os movimentos realizados no caminho especificado na tabela seguem a seguinte ordem: primeiramente, todos os movimentos contidos na primeira coluna são realizados, depois, os da segunda coluna e assim por diante, seguindo esta ordem. O primeiro movimento é o da linha 1, coluna *Sequência 1* e o último é o da linha 11, coluna *Sequência 4*.

Tabela 4.1: Caminho percorrido pelos clientes no ambiente virtual.

	Sequência 1	**Sequência 2**	**Sequência 3**	**Sequência 4**
1	r_direita(5)	t_frente(53)	r_direita(19)	t_frente(54)
2	t_frente(4)	r_direita(75)	t_frente(18)	r_esquerda(24)
3	r_direita(5)	t_tras(22)	r_esquerda(3)	r_direita(1)
4	t_frente(9)	r_esquerda(11)	t_frente(50)	r_esquerda(5)
5	r_direita(17)	r_direita(2)	r_direita(32)	t_direita(6)
6	t_frente(22)	t_frente(56)	t_frente(26)	t_frente(4)
7	r_esquerda(42)	r_direita(56)	r_direita(82)	r_direita(46)
8	t_frente(27)	t_tras(24)	t_frente(19)	r_esquerda(10)
9	r_esquerda(20)	t_direita(6)	r_esquerda(37)	t_frente(9)
10	t_frente(8)	r_esquerda(45)	t_frente(22)	r_esquerda(42)
11	r_direita(27)	t_frente(29)	r_esquerda(4)	r_direita(6)

4.2 Configuração e Desempenho do Servidor de Renderização

Para a renderização remota dos quadros utilizou-se um servidor com uma CPU Intel® Pentium® 4, 3.0 GHz, 1 GBytes de RAM, *hard disk* de 80 GBytes e placa de vídeo NVidia GeForce 6200 com 256 MBytes de memória de vídeo. O servidor foi executado no

sistema operacional Linux Ubuntu Server 8.10. Este servidor ficou dedicado às tarefas de renderização, ou seja, nenhum outro serviço foi executado nele.

A avaliação do desempenho do servidor foi realizada de acordo com os três tipos de quadros renderizados relativos às arquiteturas propostas, o quadro completo, a imagem de resíduo e o quadro referente à arquitetura convencional.

Para avaliar o desempenho do servidor, coletou-se os tempos das tarefas executadas na renderização dos quadros em cada arquitetura relativos às simulações realizadas em cada uma, então, calculou-se o tempo médio gasto em cada tarefa a partir dos dados coletados, permitindo uma análise detalhada do desempenho do servidor na renderização dos quadros.

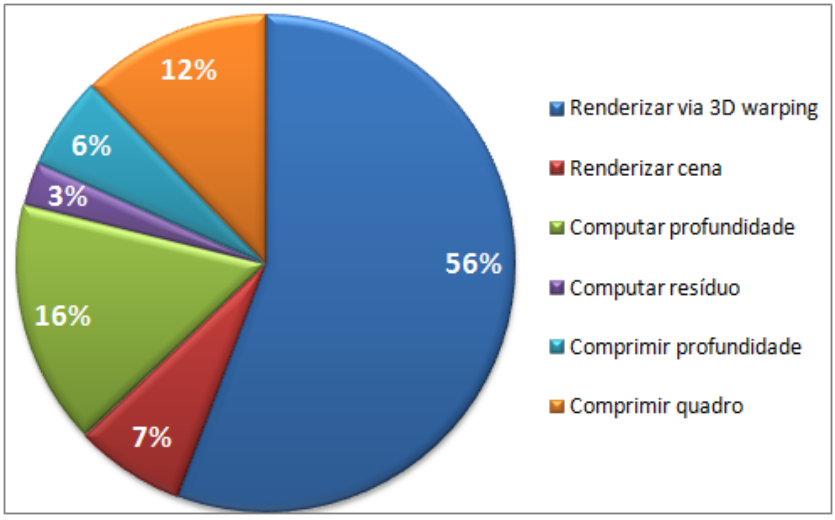


Figura 4.2: Percentual dos tempos gastos nas tarefas que compõem o processo de renderização dos quadros no servidor utilizando a arquitetura com imagens de resíduo.

Tabela 4.2: Tempos médios gastos nas tarefas que compõem o processo de renderização dos quadros no servidor utilizando a arquitetura com imagens de resíduo.

	Tempo (milissegundos)
Renderizar via 3D *warping*	54,81
Renderizar cena	7,00
Computar profundidade	15,65
Computar resíduo	2,74
Comprimir profundidade	5,96
Comprimir quadro	12,13
Total	98,30

O percentual dos tempos gastos para cada tarefa na renderização dos quadros no

servidor relativo à arquitetura com imagens de resíduo pode ser observado na Figura 4.2. As tarefas que mais demandaram processamento foram a renderização pela técnica 3D *warping* com 56%, seguido da computação da informação de profundidade com 16% e a compressão do quadro com 12%. O restante das tarefas somaram 16%. A grande diferença entre o percentual dos tempos de renderização da cena e a renderização via 3D *warping* ocorreu porque a renderização da cena foi executada pela GPU e 3D *warping* pela CPU.

Os tempos médios gastos em cada tarefa na renderização dos quadros no servidor referente à arquitetura com imagens de resíduo são ilustrados na Tabela 4.2. O tempo total médio gasto para renderizar as vistas nessa arquitetura resulta numa taxa média de 10,17 quadros por segundo (FPS).

O baixo desempenho do servidor (baixo FPS) resultante nessa arquitetura foi atribuído principalmente à tarefa de renderização via 3D *warping* que consumiu cerca de 55 milissegundos e também à grande quantidade de tarefas que foram executadas para computar a imagem de resíduo.

Retirando duas tarefas do processo de renderização da arquitetura com imagens de resíduo, que são a renderização via 3D *warping* e a computação do resíduo, constitui-se a renderização da cena através da arquitetura com quadros completos. O percentual dos tempos gastos nas tarefas executadas para renderizar as vistas utilizando essa arquitetura é ilustrado na Figura 4.3.

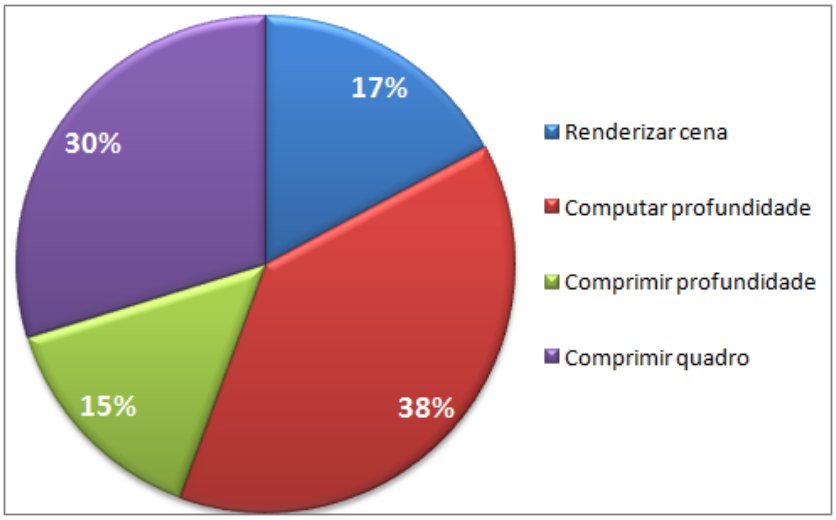


Figura 4.3: Percentual dos tempos médios gastos nas tarefas que compõem o processo de renderização dos quadros no servidor utilizando a arquitetura com quadros completos.

Observa-se um equilíbrio entre os tempos gastos nas tarefas executadas no processo de renderização das vistas. As tarefas de computar a informação de profundidade e a compressão do quadro foram as que demandaram mais tempo, consumindo 68% e as demais praticamente dividiram o restante do tempo.

Os tempos de cada tarefa são os mesmos apresentados na Tabela 4.2 referente à arquitetura com imagens de resíduo. Como duas tarefas a menos foram executadas na arquitetura com quadros completos em comparação com a de imagens de resíduo, o tempo total médio da renderização dessa arquitetura foi de 40,74 milissegundos, dado pela subtração do tempo total apresentado na Tabela 4.2 pelos tempos referentes às tarefas que foram excluídas (renderização via 3D *warping* e computação do resíduo).

Obviamente, o desempenho na renderização dos quadros no servidor empregando a arquitetura com quadros completos foi muito melhor em comparação com a arquitetura com imagens de resíduo, cerca de 58,5% mais rápida na renderização dos quadros. Isso se deve à redução do número das tarefas executadas e porque a tarefa de renderização via 3D *warping* não foi realizada, por ser uma tarefa considerada lenta comparada com as demais. O tempo médio obtido na renderização dos quadros utilizando a arquitetura com quadros completos resulta em uma taxa de renderização de 24 FPS.

O percentual dos tempos gastos nas tarefas executadas para renderizar as vistas no servidor utilizando a arquitetura convencional é ilustrado na Figura 4.4. A diferença entre os tempos das tarefas foi bem acentuada. Isto ocorreu porque a tarefa de renderização da cena foi executada pela GPU e a tarefa de compressão pela CPU do servidor.

O desempenho do servidor na renderização dos quadros relativo à arquitetura convencional é ilustrado na Tabela 4.3. Se comparar as tarefas de renderização da cena e compressão do quadro dessa arquitetura com as outras arquiteturas baseadas em 3D *warping*, observa-se uma grande diferença nos tempos gastos pelas tarefas.

Esse fato ocorreu porque na arquitetura convencional a dimensão das imagens renderizadas foi de 400×400 pixels e nas arquiteturas baseadas em 3D *warping* foi de 500×500 pixels. Essa diferença contribuiu consideravelmente para o aumento do desempenho na arquitetura convencional.

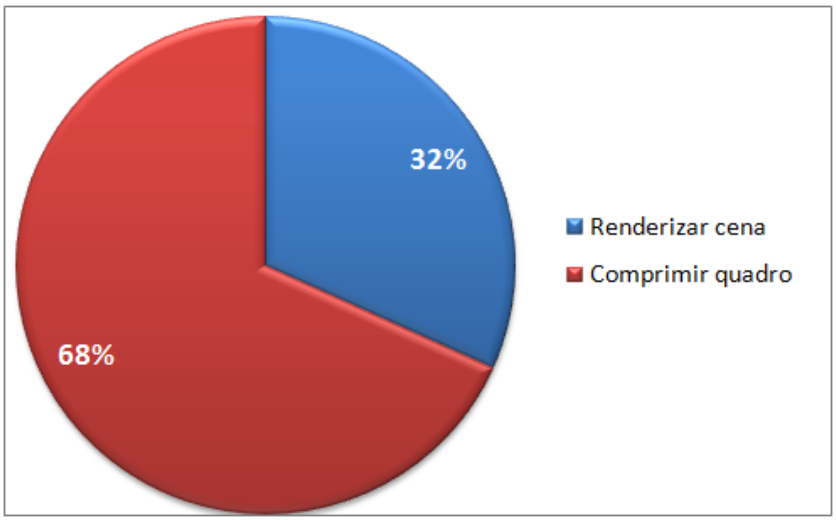


Figura 4.4: Percentual dos tempos médios gastos nas tarefas que compõem o processo de renderização dos quadros no servidor utilizando a arquitetura convencional.

Tabela 4.3: Tempos médios gastos nas tarefas que compõem o processo de renderização dos quadros no servidor utilizando a arquitetura convencional.

	Tempo (milissegundos)
Renderizar cena	4,86
Comprimir quadro	10,45
Total	15,31

O tempo médio gasto para renderizar as vistas no servidor empregando a arquitetura convencional resultou em uma taxa de 65,3 FPS. Com relação ao tempo de renderização dos quadros no servidor, esta arquitetura foi a que obteve melhor desempenho, por realizar o menor número de tarefas, além de que as imagens renderizadas possuem dimensão menor que nas demais arquiteturas.

Os tempos médios de renderização obtidos nos diferentes quadros renderizados pelos servidor podem variar em torno de 2% a 3%. Essa variação foi calculada baseada na relação entre o desvio padrão e média dos tempos obtidos.

O desempenho obtido na renderização das vistas no servidor em cada arquitetura influencia no desempenho conseguido no processo de renderização remota no cliente, pois o tempo de renderização remota inclui o tempo de renderização dos quadros no servidor (seção 3.6).

4.3 Experimento I

Nesta seção é descrito o experimento I que avaliou o desempenho do protótipo nas diferentes arquiteturas propostas. Este experimento utilizou 6 clientes que interagiram no ambiente virtual simultaneamente através da simulação. O tempo de intervalo entre as requisições foi definido em 150 milissegundos. Na arquitetura combinada foi especificado que, a cada 10 quadros renderizados utilizando a arquitetura com imagens de resíduos, um quadro foi renderizado empregando a arquitetura com quadros completos e assim sucessivamente.

O ambiente computacional dos clientes foi formado por computadores homogêneos, que utilizam o conceito do LTSP (*Linux Terminal Server Project*) [35]. LTSP permite a criação de terminais utilizando Linux, possibilitando que estes executem os aplicativos instalados no servidor e inicialize o sistema operacional via rede, baixando todos os *softwares* necessários do servidor.

Os terminais não necessitam de *hard disk*, já que o servidor é o responsável por executar todas as tarefas (programas) dos terminais e armazenar os dados [39]. Além disso, os terminais, apenas informam ao servidor os dados de entrada como as teclas digitadas ou o botão do *mouse* pressionado, então, o servidor processa as informações recebidas e transmite para os terminais apenas as instruções para gerar as janelas que serão exibidas [39].

LTSP permite utilizar-se de computadores (terminais) de baixo desempenho, uma vez que a única tarefa do terminal é atualizar a tela com instruções enviadas pelo servidor [39]. Os terminais empregados neste experimento não dispõem de placas aceleradoras gráficas.

A organização computacional é ilustrada na Figura 4.5. Os terminais requisitam as vistas ao servidor LTSP, caso a requisição seja uma renderização local do cliente empregando 3D *warping*, o próprio servidor LTSP renderiza esta vista e entrega ao terminal e caso a requisição seja relativa a uma renderização remota, o servidor LTSP transmite essa requisição para o servidor de renderização, que renderiza a imagem e transmite via rede para o servidor LTSP, que entrega ao terminal que requisitou a vista.

O servidor dos terminais conta com 2 processadores Intel® Xeon® *quad core* modelo

E5345 de 2.33 GHz, memória disponível de 11 Gbytes, limitada em 1,1 GBytes por processo do usuário[2] e espaço em disco disponível é de 2 GBytes para o usuário utilizado. Utilizou-se de um único usuário para os 6 terminais usados, permitindo a execução de 6 clientes na simulação deste experimento, um em cada terminal.

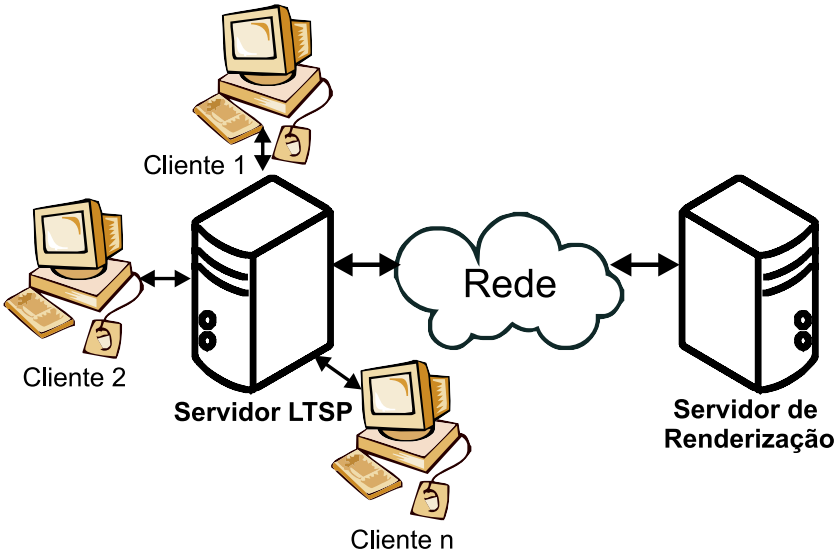


Figura 4.5: Organização computacional.

A seguir, a subseção 4.3.1 descreve os resultados obtidos na renderização do ambiente virtual empregando as arquiteturas baseadas na técnica 3D *warping*, a subseção 4.3.2 apresenta a comparação entre a renderização da cena utilizando uma arquitetura com e sem descarte de requisições e, por fim, na subseção 4.3.3 é descrita a avaliação do desempenho do protótipo com relação à arquitetura convencional UDP e TCP.

4.3.1 Arquiteturas Baseadas em 3D *Warping*

Os resultados obtidos na renderização remota da cena utilizando as arquiteturas baseadas em 3D *warping* para o experimento I são descritos a seguir na subseção 4.3.1.1 e na renderização local na subseção 4.3.1.2.

[2]Para a utilização de um terminal é necessário um usuário cadastrado e este deve estar autenticado no servidor. O servidor limita os recursos de memória e armazenamento por usuário.

4.3.1.1 Renderização Remota

Os tempos médios da renderização remota obtidos nas arquiteturas baseadas na técnica 3D *warping* são ilustrados na Figura 4.6. Os tempos exibidos nessa figura representam a média dos tempos obtidos na renderização remota dos 6 clientes utilizados na simulação em cada arquitetura. As medidas estatísticas dos tempos de renderização remota da cena de cada cliente relativas às arquiteturas com imagens de resíduo, combinada e quadros completos, podem ser observadas respectivamente nas Tabelas 4.4, 4.5 e 4.6.

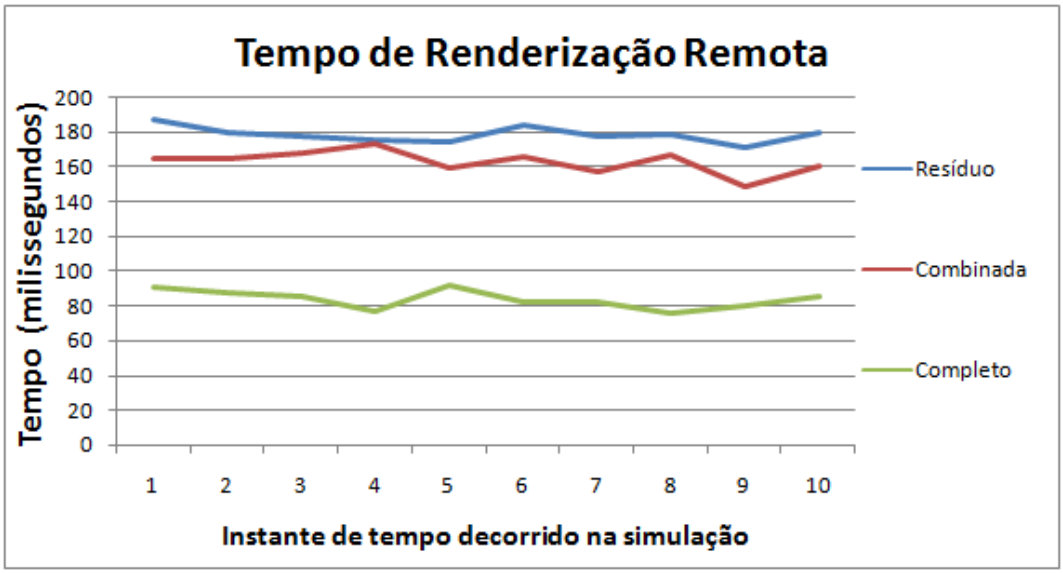


Figura 4.6: Tempo médio do processo de renderização remota nas arquiteturas baseadas na técnica 3D *warping* para o experimento I.

A arquitetura que teve melhor desempenho na renderização remota foi a com quadros completos, com tempo médio de 80ms, seguido da arquitetura combinada que ficou em torno de 160ms. O pior desempenho foi obtido pela arquitetura com imagens de resíduo com tempo médio de 180ms. O gráfico também mostra que a diferença dos tempos de renderização entre as arquiteturas que empregaram as imagens de resíduo foi pequena.

Nota-se que as arquiteturas com imagens de resíduo e combinada que utilizam imagens de resíduo tiveram tempos de renderização altos em comparação com a arquitetura com quadros completos. Este fato ocorreu porque para computar as imagens de resíduo no servidor de renderização foi necessário a renderização via 3D *warping*, que foi executada pela CPU do servidor. Em oposição, a arquitetura com quadros completos não computa a

imagem de resíduo, e a cena foi renderizada somente pela GPU do servidor de renderização.

A limitação do desempenho na renderização remota dos quadros nesse experimento foi determinada pelo desempenho do servidor. Os valores de mínimo da arquitetura com quadros completos e a arquitetura combinada, observados nas Tabelas 4.5 e 4.6, foram semelhantes, porque ambas renderizaram quadros completos e este tipo de quadro no servidor foi renderizado com o tempo médio de 40,74ms.

Tabela 4.4: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura com imagens de resíduo.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	109,09	92,92	98,11	106,03	110,14	85,42
Máximo	257,67	288,42	259,48	234,56	291,70	286,76
Média	183,58	175,37	177,00	178,40	175,20	180,73
Desvio padrão	25,92	35,81	29,80	26,03	30,07	32,21
Variação	14%	20%	17%	15%	17%	18%

Tabela 4.5: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura combinada.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	57,85	59,35	72,32	61,36	59,92	59,16
Máximo	228,36	249,28	223,74	246,19	225,73	232,39
Média	159,94	158,99	165,19	167,42	160,69	162,34
Desvio padrão	37,40	36,71	35,48	38,69	37,40	38,64
Variação	23%	23%	21%	23%	23%	24%

Tabela 4.6: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura com quadros completos.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	58,04	58,42	58,99	58,00	57,34	58,16
Máximo	145,07	140,97	112,33	114,67	116,23	115,60
Média	78,91	85,69	86,15	77,36	81,24	81,53
Desvio padrão	19,33	17,54	15,78	18,03	19,53	17,49
Variação	25%	20%	18%	23%	24%	21%

Na arquitetura com imagens de resíduo, os valores de mínimos de cada cliente também foram próximos do tempo médio de renderização desses quadros no servidor que fica em torno de 98,30ms. Nota-se que os valores de máximo entre a arquitetura combinada e a arquitetura com imagens de resíduo foram similares, porque ambas utilizaram imagens

de resíduo e esse tipo de quadro demanda mais tempo para sua renderização no servidor em comparação com quadros completos. Além disso, a diferença dos valores de máximo entre a arquitetura com quadros completos e as demais arquiteturas baseadas em 3D *warping* foi dada por essa diferença entre o tempo de renderização dos quadros completos e imagens de resíduo no servidor.

A medida de variação apresentada nas tabelas é a relação entre o desvio padrão e a média dos tempos de renderização obtidos pelos clientes durante a simulação. O tempo de renderização remota pode sofrer variação com relação ao tempo de renderização dos quadros no servidor, o tempo de transferência das imagens renderizadas no servidor e transmitidas para os clientes e o tempo de espera para que a requisição da vista desejada pelo cliente seja processada pelo servidor.

A renderização dos quadros no servidor, descrita na seção 4.2, possui uma variação pequena em torno de 2 a 3%. O tempo de transferência das imagens do servidor para o cliente pode variar em situações de instabilidade da rede, como a largura de banda real disponível para o momento exato da transferência, que pode variar de acordo com o tráfego da rede.

O tempo de espera para que uma requisição de um quadro feita pelo cliente ao servidor seja renderizada pode variar de acordo com a concorrência dos clientes na renderização remota das vistas. Os clientes iniciam a simulação praticamente ao mesmo tempo e então começam a requisitar as vistas ao servidor. O servidor armazena as requisições em seu *buffer* de requisições e a ordem de renderização das vistas é a ordem de chegada das requisições.

Nas arquiteturas baseadas em 3D *warping* o acúmulo de requisições não interferiu no desempenho da renderização remota. As requisições acumuladas foram descartadas pelo servidor e somente a última requisição de cada cliente foi processada, como descrito no método proposto para a redução da sobrecarga do servidor na seção 3.4.

Entretanto, as requisições dos clientes envolvidos na simulação foram feitas em tempos próximos, gerando uma concorrência no processamento dessas requisições. A primeira a chegar foi a primeira a ser renderizada e as demais tiveram que esperar a renderização das

requisições dos outros clientes de acordo com a ordem armazenada no *buffer* de requisições do servidor, variando o tempo de renderização remota das vistas por conta disso.

A variação implica na qualidade das vistas renderizadas localmente pelos clientes e, consequentemente, a qualidade das vistas exibidas. Quando há muita variação, as imagens renderizadas remotamente podem ter tempos de renderização remota altos e baixos. Nos tempos altos, a demora no recebimento pode implicar na falta de amostra e as vistas renderizadas localmente apresentarão uma quantidade maior de artefatos e nos tempos baixos, as vistas renderizadas tendem a apresentar menos artefatos.

A arquitetura com menor medida de variação foi a arquitetura com imagens de resíduo, em que os clientes obtiveram variação nos tempos de renderização remota com relação aos tempos médios de 14% para a menor variação relativa ao cliente C1 chegando até a 20% na maior variação medida. Nas demais arquiteturas, os clientes obtiveram medidas de variação similares: na arquitetura combinada, a variação foi de no máximo 24%, enquanto a arquitetura com quadros completos teve variação máxima de 25%.

Os valores de máximos estão relacionados a momentos de grande variação na renderização remota, por exemplo, momentos em que uma requisição de uma vista feita pelo cliente ao servidor foi a última a ser renderizada, com relação às demais requisições armazenadas no *buffer* de requisições do servidor. Assim, o tempo de espera foi alto, gerando os tempos máximos durante a renderização remota do ambiente virtual.

Os tempos de mínimo representam momentos, em que o tempo de renderização remota praticamente não foi afetado pela concorrência na renderização das vistas no servidor e qualquer outro tipo de variação. Os valores de mínimos foram muito próximos do tempo médio de renderização dos quadros no servidor, indicando momentos de bom desempenho na renderização remota dos quadros.

Os percentuais de requisições de vistas renderizadas e descartadas no servidor relativos às arquiteturas com imagens de resíduo, combinada e quadros completos podem ser observados respectivamente nas Tabelas 4.7, 4.8 e 4.9.

Os percentuais de requisições descartadas nas arquiteturas que utilizam imagens de resíduo foram maiores em comparação a arquitetura com quadros completos. Isso ocorreu

porque, na renderização dos quadros utilizando a arquitetura com quadros completos, o servidor de renderização apresentou maior desempenho comparado com a renderização das imagens de resíduo, assim, o servidor renderizou mais rápido e despachou o quadros mais rapidamente, reduzindo o percentual de requisições acumuladas e obviamente de requisições descartadas.

Tabela 4.7: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento I utilizando a arquitetura com imagens de resíduo.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartadas	69%	73%	70%	72%	63%	72%
Renderizadas	31%	27%	30%	28%	37%	28%

Tabela 4.8: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento I utilizando a arquitetura combinada.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartadas	68%	66%	70%	71%	70%	67%
Renderizadas	32%	34%	30%	29%	30%	33%

Tabela 4.9: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento I utilizando a arquitetura com quadros completos.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartadas	31%	48%	64%	36%	48%	50%
Renderizadas	69%	52%	36%	64%	52%	50%

Essa diferença no desempenho da renderização dos quadros no servidor ocorreu porque nas arquiteturas que empregam imagens de resíduo, estas imagens necessitam da execução da técnica 3D *warping*, que foi executada pela CPU. Já quadros completos foram renderizados somente pela GPU, resultando em um desempenho melhor.

O percentual de requisições descartadas implica na qualidade das vistas exibidas pelos clientes. Quanto maior o percentual de requisições descartadas, menos amostras (imagens) foram enviadas para o cliente que, por sua vez, renderizou mais vistas a partir de uma única amostra até que outra fosse recebida, produzindo vistas com um maior número de artefatos.

Entretanto, por outro lado, o descarte de requisições faz com que o servidor não fique sobrecarregado, permitindo a renderização dos quadros a atendendo vários clientes. Para

reduzir o percentual de requisições descartadas, deve-se diminuir o número de clientes ou aumentar o tempo de intervalo entre as requisições definido nos parâmetros da simulação, fazendo com que menos requisições acumulem-se no *buffer* do servidor.

4.3.1.2 Renderização Local

Os tempos médios de renderização local nas arquiteturas baseadas na técnica 3D *warping* são ilustrados na Figura 4.7. O gráfico mostra o tempo médio resultante da renderização local dos 6 clientes envolvidos na simulação deste experimento em cada arquitetura.

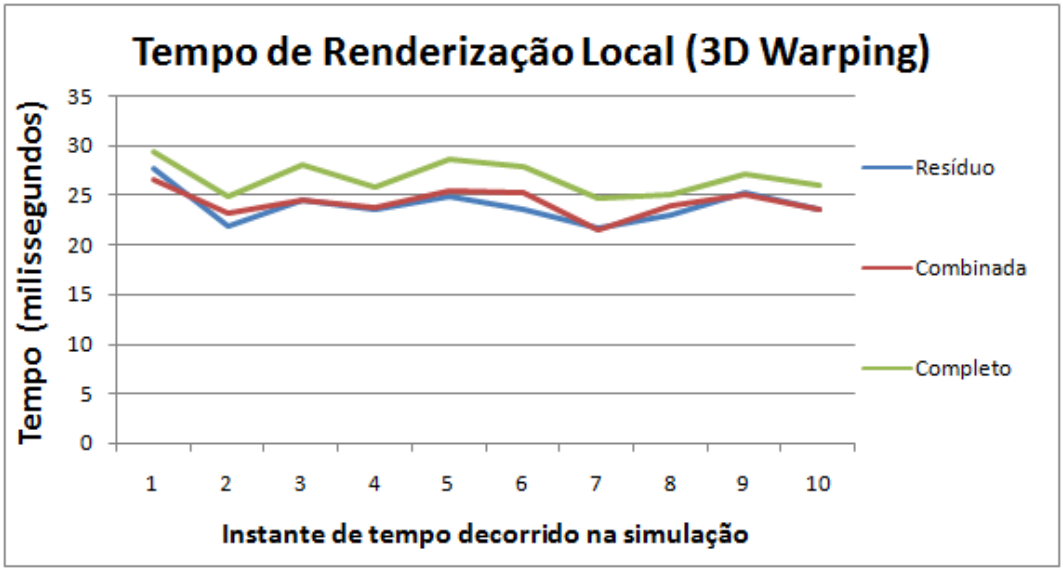


Figura 4.7: Tempo médio do processo de renderização local nas arquiteturas baseadas na técnica 3D *warping* para o experimento I.

O tempo de renderização resultante nas arquiteturas foram bem similares. Na arquitetura com quadros completos observa-se uma pequena diferença, tendo os valores de tempos mais altos em comparação com as demais.

As curvas ilustradas no gráfico mostram que, em todas as arquiteturas, obteve-se o mesmo padrão de comportamento. Os instantes de tempo 2 e 7 foram os pontos de melhor desempenho e os pontos 1, 5, 6 e 9 foram os instantes de tempo na simulação que resultaram em um maior tempo gasto para renderizar as vistas localmente.

Apesar da diferença dos tempos entre as arquiteturas, esse padrão indicou que, em

determinadas vistas, o tempo gasto para processar foi maior e em outras menor. Essa variação no tempo de renderização das vistas utilizando 3D *warping* ocorreu porque, quanto maior o número de pixels mapeados da imagem fonte para a destino, maior foi o tempo de renderização da vista.

No caso dos movimentos de rotação, tradicionalmente uma região vertical em um dos lados da imagem não é mapeada (Figura 3.3 da seção 3.1.1) e nos movimentos de translação para frente quase todos os pixels são mapeados, gerando a diferença nos tempos de renderização.

Essa diferença pode ser observada no gráfico. No início da simulação, uma sequência de movimentos de rotação foi realizada, gerando a queda no tempo de renderização, que vai do instante de tempo 1 até o 2, posteriormente, vários movimentos de translação para frente foram feitos, indicando o aumento do tempo de renderização.

As medidas estatísticas dos tempos de renderização local da cena de cada cliente relativas às arquiteturas com imagens de resíduo, combinada e quadros completos, podem ser observadas respectivamente nas Tabelas 4.10, 4.11 e 4.12.

A diferença entre os tempos médios na renderização local em cada arquitetura está associada à qualidade das amostras utilizadas na renderização das vistas empregando 3D *warping*. Na arquitetura com quadros completos, a renderização remota do quadros (subseção 4.3.1.1) foi mais rápida que nas demais, então as amostras transmitidas pelo servidor estavam disponíveis para a renderização local mais rapidamente em comparação com as demais arquiteturas.

Isso faz com que menos vistas sejam renderizadas localmente baseadas em uma única amostra, até que uma nova amostra referente ao atual ponto de vista ou o ponto de vista mais próximo seja recebida. Quanto mais vistas são renderizadas a partir de uma única amostra, mais artefatos estarão presentes nas vistas renderizadas e, consequentemente, menos pixels são mapeados durante o *warping*, como observado na Figura 3.6 da subseção 3.1.1.

Assim, as vistas renderizadas localmente pela arquitetura de quadros completos possuem qualidade superior em comparação com as demais arquiteturas, por mapear mais

pixels para a imagem destino durante o *warping* e, consequentemente, requer um tempo maior gasto para renderizar os quadros.

Pelos mesmos motivos, os tempos de renderização da arquitetura combinada foram maiores que os da arquitetura com imagens de resíduo, porque a cada 10 quadros renderizados um foi completo e, por isso, a diferença nos tempos de renderização entre essas duas arquiteturas foi pequena.

Tabela 4.10: Medidas estatísticas referentes ao tempo de renderização local para o experimento I utilizando a arquitetura com imagens de resíduo.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	9,14	9,27	10,27	7,64	7,18	8,52
Máximo	94,36	107,23	94,39	78,25	82,07	123,35
Média	24,59	24,19	23,51	23,63	23,88	23,73
Desvio padrão	8,08	9,19	8,24	7,61	7,61	9,51
Variação	33%	38%	35%	32%	32%	40%

Tabela 4.11: Medidas estatísticas referentes ao tempo de renderização local para o experimento I utilizando a arquitetura combinada.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	8,76	6,32	3,74	9,78	6,08	10,77
Máximo	64,40	56,38	64,47	59,05	87,07	66,66
Média	24,39	24,15	24,01	24,15	23,51	24,91
Desvio padrão	7,11	7,28	7,13	6,88	6,59	6,97
Variação	29%	30%	30%	28%	28%	28%

Tabela 4.12: Medidas estatísticas referentes ao tempo de renderização local para o experimento I utilizando a arquitetura com quadros completos.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	15,30	11,30	10,26	11,37	8,88	12,97
Máximo	124,79	88,80	86,25	78,52	89,92	70,41
Média	27,30	26,90	26,00	27,58	26,53	26,44
Desvio padrão	7,99	7,35	7,11	6,25	7,14	6,27
Variação	29%	27%	27%	23%	27%	24%

Os valores de mínimos ilustrados nas tabelas mostram dois comportamentos diferentes. Se o valor é muito baixo com relação à média, como no caso do cliente 3 (C3) na Tabela 4.11, isto indica que a vista renderizada teve pouquíssimos pixels mapeados. Outro comportamento é quando o valor de mínimo não apresenta tanta diferença, como no caso

do cliente 6 na Tabela 4.12, representando uma vista que foi renderizada rapidamente como, por exemplo, relativo a um movimento de rotação.

Nota-se que os valores de mínimos da arquitetura com quadros completos foram melhores que nas demais arquiteturas, indicando que as vistas renderizadas tiveram um bom mapeamento durante o *warping*.

A medida de variação apresentada nas tabelas pode ser usada para medir a qualidade da vistas renderizadas localmente. Quando a variação é grande indica que, durante a renderização dos quadros, muitas vistas foram renderizadas rapidamente e outras em tempo médio, caracterizando momentos em que o cliente renderizou vistas com poucos pixels mapeados e, quando recebia as amostras renderizadas pelo servidor, voltava a renderizar em tempo médio e assim sucessivamente. Já, se a variação é pequena, isto indica que as vistas renderizadas tiveram um bom mapeamento dos pixels.

Nas três arquiteturas, a medida de variação neste teste acompanhou a qualidade das amostras utilizadas na renderização local das vistas nos clientes. A arquitetura com imagens de resíduo foi a que obteve maior variação, com 32% para o cliente que teve menos variação chegando a até 40% para o cliente C6. Em seguida, a arquitetura combinada obteve variação de 28% para a menor medida e 30% para a maior. A arquitetura com quadros completos obteve a menor variação que foi de 23% para C4 chegando a até 29%.

A medida de variação mostra que a arquitetura combinada apresentou uma variação menor comparada com a arquitetura com imagens de resíduo e semelhante à arquitetura com quadros completos. Isso indica que o quadro completo recebido pelo cliente a cada 10 vistas renderizadas foi fundamental para manter a qualidade das vistas renderizadas, dado o valor de variação obtido pelos clientes dessa arquitetura.

Os computadores utilizados para esse experimento utilizam a abordagem LTSP. Os tempos de renderização local considerados altos, como os valores de máximos, podem ocorrer em momentos em que o poder de processamento do servidor LTSP é limitado ou reduzido durante o processamento (renderização) das vistas requisitadas pelos terminais.

Esse fato pode vir a acontecer em momentos de instabilidade do servidor LTSP, que não consegue processar as requisições dos terminais em tempo adequado por estar so-

brecarregado ou processando tarefas para outros terminais, demandando mais tempo na renderização local das vistas.

4.3.2 Comparação entre Arquitetura com Descarte de Requisições e Arquitetura sem Descarte de Requisições

Este teste apresenta a comparação entre a renderização do ambiente virtual empregando a arquitetura com imagens de resíduo com e sem descarte de requisições. Neste teste, o número de clientes utilizados foi reduzido para 2. Como as imagens de resíduo obtiveram pior desempenho entre os diferentes quadros renderizados no servidor (seção 4.2), a tendência é que um maior número de requisições acumule-se no *buffer* do servidor empregando a arquitetura com imagens de resíduo, justificando sua escolha. O número de clientes foi reduzido porque em um dos testes não é empregado o método de descarte de requisições e a utilização de vários clientes pode sobrecarregar o servidor devido ao acúmulo de requisições.

A seguir, na subseção 4.3.2.1 são apresentados os resultados da comparação entre a arquitetura com imagens de resíduo com e sem descarte de requisições referente à renderização remota e na subseção 4.3.2.2 referente à renderização local.

4.3.2.1 Renderização Remota

Os tempos médios obtidos na renderização remota entre as arquiteturas com imagens de resíduo com e sem descarte de requisições são ilustrados na Figura 4.8. As medidas estatísticas dos tempos de renderização remota da cena são ilustradas nas Tabelas 4.13 e 4.14.

A Figura 4.8 exibe uma grande discrepância entre os tempos de renderização das duas arquiteturas. A arquitetura com descarte de requisições se manteve estável durante a simulação com tempo médio que variou entre 167 a 172 milissegundos. Assim como os tempos médios dessa arquitetura, os valores de máximo e mínimo obtidos foram semelhantes aos da renderização remota dessa arquitetura neste experimento descritos na

subseção 4.3.1.1.

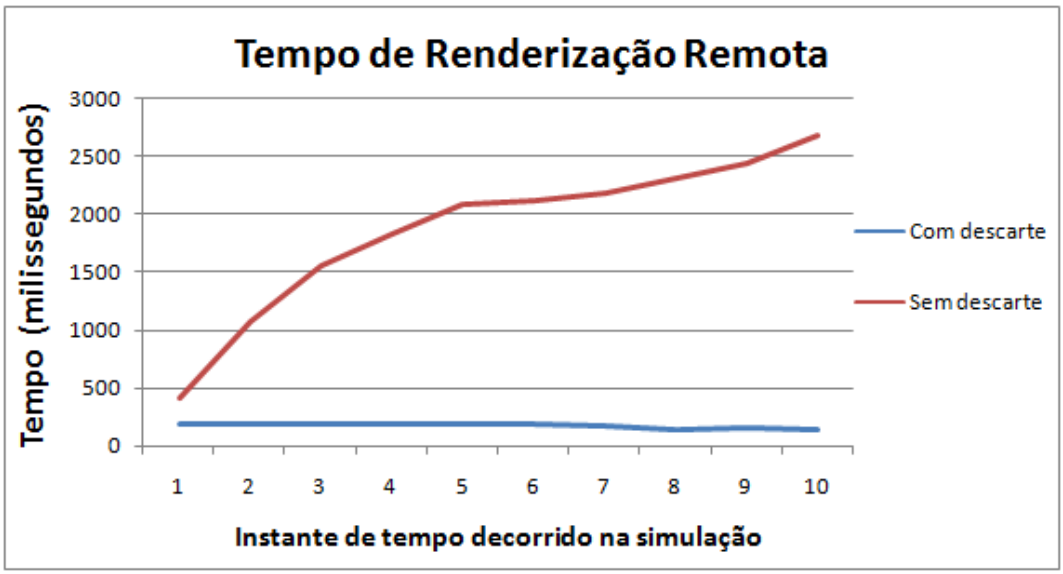


Figura 4.8: Tempo médio do processo de renderização remota entre as arquiteturas com imagens de resíduo com e sem descarte de requisições para o experimento I.

Tabela 4.13: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	C2
Mínimo	105,99	103,70
Máximo	251,20	260,52
Média	167,75	172,87
Desvio padrão	32,62	33,58
Variação	19%	19%

Tabela 4.14: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura com imagens de resíduo sem descarte de requisições.

	C1	C2
Mínimo	114,42	134,01
Máximo	3184,25	3166,76
Média	2138,92	2128,53
Desvio padrão	742,28	756,94
Variação	35%	36%

A arquitetura sem descarte de requisições apresentou um comportamento com baixo desempenho. Os valores de mínimo indicam que, no início na simulação, os tempos de renderização obtidos foram semelhantes aos da arquitetura com descarte de requisições.

Assim que começou o acúmulo de requisições no servidor, o tempo de renderização remota obtido pelos clientes começou a subir.

Nota-se que, desde o início da simulação até o final, o tempo de renderização só aumentou, indicando que o servidor não conseguiu renderizar as vistas em tempo adequado e as requisições acumularam-se gerando um atraso maior nas renderização dos quadros a cada instante de tempo que decorreu a simulação.

No instante de tempo 1 até o instante de tempo 5, a curva que mostra o tempo de renderização da arquitetura sem descarte de requisições cresce abruptamente. Posteriormente, no instante de tempo 5 até o 8, observa-se que o servidor conseguiu renderizar as vistas mais rapidamente, embora o acúmulo das requisições ainda estivesse presente.

Este fato pode ocorrer devido à variação do intervalo de tempo entre as requisições. Em momentos que a renderização local é rápida, o intervalo entre as requisições é menor aumentando o acúmulo de vistas requisitadas no servidor e quando a renderização local é mais lenta, o intervalo é maior diminuindo o número de requisições acumuladas.

Como mostra a Figura 4.9 da subseção 4.3.2.2, em que foram descritos os resultados da renderização local deste teste, nos instantes de tempo 5 a 9 da arquitetura sem descarte de requisições, a renderização local foi mais lenta comparada com outros momentos da simulação, assim, os clientes demoraram mais tempo para requisitar as vistas ao servidor, refletindo em um número menor de requisições acumuladas no servidor nesse intervalo de tempo e, consequentemente, melhorando a taxa de renderização remota.

Os valores dos tempos máximos, médios, desvio padrão e variação obtidos pelos clientes da arquitetura sem descarte de requisições foram bem discrepantes da arquitetura com descarte de requisições, devido ao acúmulo de requisições no servidor. A arquitetura com descarte teve um tempo médio de renderização remota de aproximadamente 92% mais rápido que a arquitetura sem descarte, devido ao grande acúmulo de requisições, gerando tempos de renderização remota altos.

Para o servidor manter um equilíbrio no tempo de renderização remota dos quadros entre os clientes e evitar a sua sobrecarga, observa-se o percentual de requisições descartadas e renderizadas pelo servidor na arquitetura com descarte na Tabela 4.15.

Em comparação com a mesma arquitetura utilizada no teste da subseção 4.3.1.1, ocorreu uma grande diferença entre o percentual obtido de requisições descartadas, porque o número de clientes neste teste foi igual a 2 e no teste anterior igual a 6. Quanto maior o número de clientes utilizados na simulação, a tendência é um maior acúmulo de requisições no servidor e, consequentemente, um número maior de requisições descartadas.

Tabela 4.15: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento I utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	**C2**
Descartados	9%	21%
Renderizados	91%	79%

A diferença entre o percentual de quadros descartados dos clientes C1 e C2 pode ter ocorrido devido à ordem em que as requisições foram recebidas pelo servidor e armazenadas no *buffer* de requisições.

4.3.2.2 Renderização Local

Os tempos médios da renderização local obtidos entre as arquiteturas com imagens de resíduo com e sem descarte de requisições são ilustrados na Figura 4.9. Os tempos exibidos nessa figura representam a média dos tempos obtidos na renderização local dos 2 clientes utilizados na simulação em cada arquitetura.

A Figura 4.9 mostra que os clientes que renderizaram localmente utilizando a arquitetura com descarte tiveram tempos médios de 25 milissegundos, enquanto os clientes que utilizaram a arquitetura sem descarte obtiveram uma queda nos tempos de renderização acumulada no decorrer da simulação, tento um comportamento oposto do resultado na renderização remota sem descarte de requisições.

O baixo tempo de renderização na arquitetura sem descarte de requisições indica que poucos pixels foram mapeados, resultando em vistas com uma presença grande de artefatos. Alguns casos, como nos tempos abaixo de 10 milissegundos, indicam que praticamente nenhum pixel foi mapeado.

Este problema ocorreu porque a renderização remota resultou em tempos de rende-

rização muito elevados, em que várias vistas foram renderizadas localmente no cliente a partir de uma única amostra, renderizando rapidamente por mapear poucos pixels na imagem destino, assim, resultando em uma visualização das vistas renderizadas no cliente de baixa qualidade com a presença de muitos artefatos.

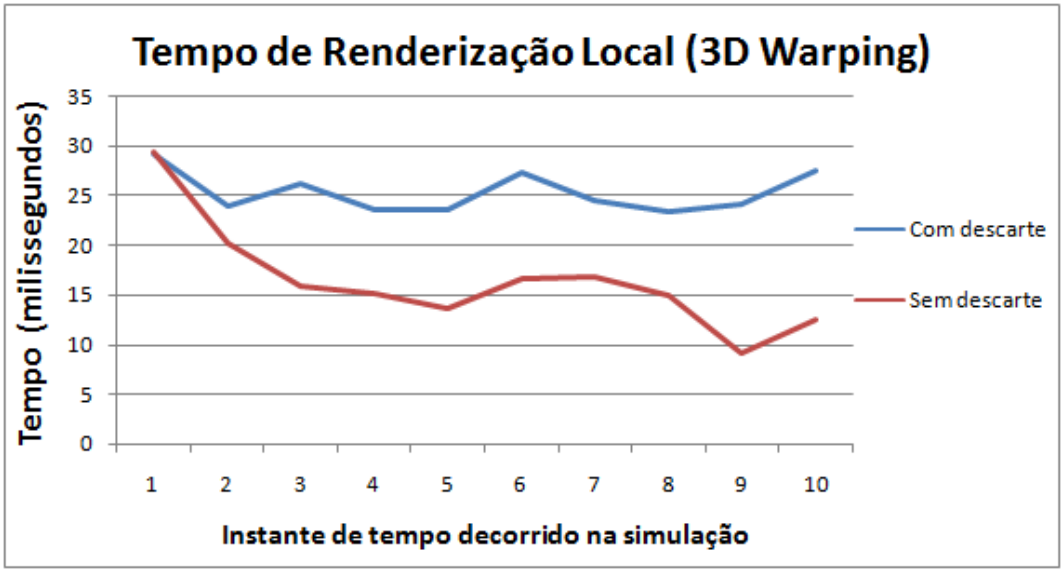


Figura 4.9: Tempo médio do processo de renderização local entre as arquiteturas com imagens de resíduo com e sem descarte de requisições para o experimento I.

A queda contínua no tempo de renderização local da arquitetura sem descarte indica que, quando o quadro foi recebido pelo cliente, este já era muito antigo com relação ao seu atual ponto de vista, então, as vistas renderizadas mesmo com a chegada da amostra apresentam artefatos e poucos pixels foram mapeados. Esse problema se agrava no decorrer da simulação acompanhando os elevados tempos de renderização remota.

As curvas nos instantes de tempo 3 a 6 e 6 a 9 nos clientes que renderizaram utilizando a arquitetura com descarte de requisições foram semelhantes às obtidas no teste da subseção 4.3.1.2, indicando que essas curvas representaram as variações nos tempos de renderização pela técnica 3D *warping* por conta do tipo de vista que foi renderizada.

Como descrito na subseção anterior, a renderização local mais lenta, como no caso das curvas da arquitetura sem descarte, fez com que o servidor renderizasse mais rapidamente por ter menos acúmulo de requisições, então, as amostras foram recebidas mais rapidamente nesses instantes de tempo pelo cliente, mapeando mais pixels para a vista renderizada, porque menos vistas foram renderizadas a partir de uma única amostra.

As medidas estatísticas dos tempos de renderização local da cena de cada cliente relativa à arquitetura com imagens de resíduo com descarte de requisições e a arquitetura com imagens de resíduo sem descarte de requisições são ilustradas nas Tabelas 4.16 e 4.17, respectivamente.

Tabela 4.16: Medidas estatísticas referentes ao tempo de renderização local para o experimento I utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	**C2**
Mínimo	20,02	19,72
Máximo	73,88	52,29
Média	25,62	25,19
Desvio padrão	4,79	3,93
Variação	19%	16%

Tabela 4.17: Medidas estatísticas referentes ao tempo de renderização local para o experimento I utilizando a arquitetura com imagens de resíduo sem descarte de requisições.

	C1	**C2**
Mínimo	6,05	6,08
Máximo	88,05	43,85
Média	16,56	15,05
Desvio padrão	9,09	5,72
Variação	55%	38%

Os tempos médios de renderização obtidos pelos clientes na arquitetura sem descarte mostram que as vistas renderizadas localmente apresentaram um grande número de artefatos, por conta do baixo tempo de renderização. As medidas de variação foram maiores na arquitetura sem descarte. Essa diferença é facilmente visualizada no gráfico devido à variação discrepante nos tempos de renderização obtidos pelos clientes nessa arquitetura.

Os tempos mínimos e máximos em ambas as arquiteturas denotam os mesmos comportamentos descritos na subseção 4.3.1.2, em que os valores de mínimos indicam as vistas que foram renderizadas mais rapidamente, em que poucos pixels foram mapeados, e os de máximo, o maior tempo gasto na renderização de uma vista, geralmente, em momentos de instabilidade do servidor dos terminais, que não processou a requisição do terminal em tempo adequado.

4.3.3 Arquitetura Convencional UDP e TCP

Neste teste foi avaliado o desempenho da arquitetura convencional na renderização do ambiente virtual. Além disso, uma comparação da arquitetura convencional utilizando dois protocolos de transporte, o UDP e o TCP foi realizada, buscando avaliar o desempenho do protótipo utilizando os diferentes protocolos.

Os tempos obtidos na renderização remota do ambiente virtual nas arquiteturas convencionais que utilizaram o protocolo de transporte TCP e o UDP são ilustrados na Figura 4.10. O gráfico mostra o tempo médio resultante da renderização remota dos 6 clientes envolvidos na simulação deste experimento em cada arquitetura.

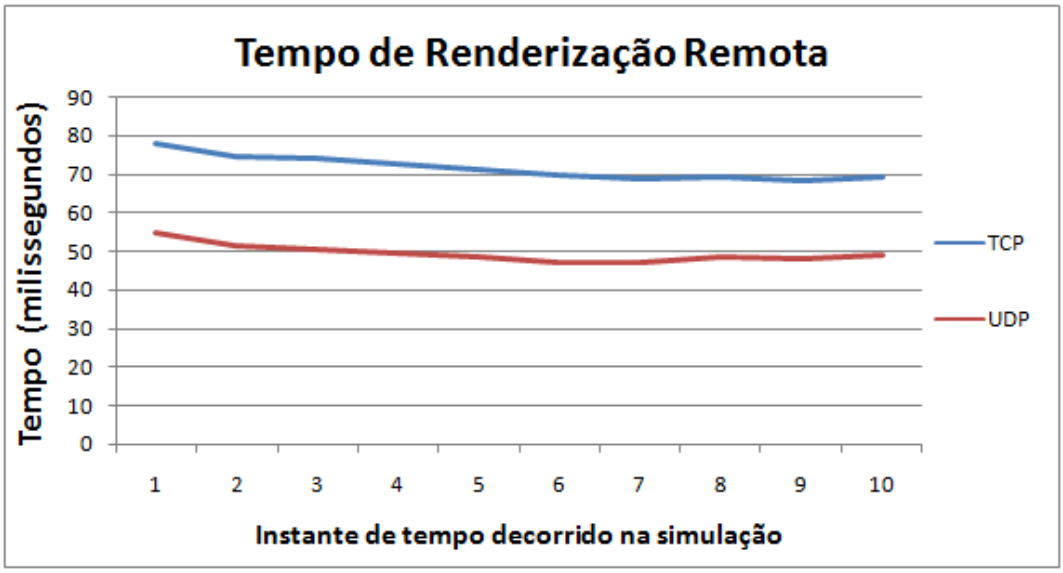


Figura 4.10: Tempo médio do processo de renderização remota entre as arquiteturas convencionais TCP e UDP para o experimento I.

A arquitetura convencional UDP obteve melhor desempenho comparado com a TCP, sendo aproximadamente 28% mais rápida. O tempo de renderização na UDP variou de 47 a 54 milissegundos e na TCP a variação foi maior de 68 a 78 milissegundos.

Na arquitetura convencional, as vistas somente são requisitadas ao servidor (renderização remota), não há renderização local. Os intervalos das requisições nas arquiteturas baseadas em 3D *warping* são de 150 milissegundos para este experimento, somando-se com o tempo da renderização local.

Como o tempo de renderização local utilizando 3D *warping* tem grande variação con-

forme a vista que é renderizada, as requisições chegam no servidor em tempos distintos. No entanto, o tempo de renderização do servidor nessas arquiteturas é lento comparado com a arquitetura convencional, gerando a concorrência para o processamento das requisições dos clientes no servidor por conta disso, como descrito na subseção 4.3.1.1.

Na arquitetura convencional UDP e TCP não há descarte de requisições por parte do servidor, porque os clientes não dispõem de um mecanismo para a renderização local das vistas, podendo gerar acúmulos de mensagens.

Embora, o servidor renderize os quadros mais rapidamente na arquitetura convencional do que nas arquiteturas baseadas em 3D *warping*, despachando as vistas requisitadas mais rapidamente, a concorrência para a renderização das vistas requisitadas ao servidor é maior, porque os clientes praticamente iniciam juntos a simulação e, a cada exatamente 150 milissegundos, 6 novas requisições são feitas ao servidor, podendo gerar acúmulo momentâneo de requisições por conta disso.

Tabela 4.18: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura convencional TCP.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	16,01	15,34	27,23	15,56	15,41	201,59
Máximo	76,99	61,03	78,92	64,52	96,19	249,68
Média	47,01	28,38	54,35	40,08	40,80	218,10
Desvio padrão	17,78	9,41	17,07	16,65	26,42	12,92
Variação	38%	33%	31%	42%	65%	6%

Tabela 4.19: Medidas estatísticas referentes ao tempo de renderização remota para o experimento I utilizando a arquitetura convencional UDP.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	29,84	15,70	30,79	16,00	15,88	42,41
Máximo	90,12	78,34	94,16	99,88	95,20	92,99
Média	70,84	58,60	50,08	30,75	29,87	57,77
Desvio padrão	14,33	17,92	9,62	15,25	19,69	10,83
Variação	20%	31%	19%	50%	66%	19%

As medidas estatísticas dos tempos de renderização remota da cena de cada cliente relativas à arquitetura convencional TCP e à arquitetura convencional UDP são ilustradas nas Tabelas 4.18 e 4.19, respectivamente.

Os valores de média e de variação foram semelhantes entre as arquiteturas, mas discrepantes entre os clientes de cada arquitetura. A discrepância entre os clientes de uma mesma arquitetura ocorreu, porque as requisições que não sofreram com o problema da concorrência no processamento das vistas no servidor, por serem as que foram primeiramente requisitadas e permaneceram no início da fila (*buffer* de requisições do servidor), foram renderizadas mais rapidamente, enquanto as demais foram prejudicadas, esperando para serem renderizadas.

No início da simulação, o servidor envia uma mensagem para cada cliente informando que a simulação começou. O servidor envia essa mensagem ordenadamente, primeiro para o cliente 1 (C1), até o último (C6). Na arquitetura que utiliza TCP é garantido que essa ordem seja respeitada, então C1 inicia a simulação primeiro que os demais, depois C2 e assim por diante. Já na UDP, isso não é garantido, uma vez que as mensagens podem chegar desordenadas.

A diferença entre o início da simulação de um cliente para outro é mínima, questão de poucos milissegundos. Como na arquitetura convencional não há descarte de requisições, a tendência de que os primeiros a iniciarem a simulação se beneficiem do processamento das vistas no servidor, uma vez que o processamento das requisições segue a ordem de chegada.

Assim, se C1 foi o primeiro a renderizar a vista não sendo prejudicado pelo tempo de espera causado pela concorrência no processamento das requisições no servidor. A tendencia é que na próxima requisição isso se repita. Entretanto, essa ordem pode mudar quando há instabilidades na rede. As requisições de alguns clientes podem chegar no servidor mais atrasadas do que as enviadas por outros clientes, então o cliente C1 que estava renderizando as vistas em tempo baixo pode começar a ter tempos de renderização mais altos e outro cliente se beneficia de estar no início da fila de requisições.

Essa variação pode ser observada nos valores de média dos clientes. Aqueles com valores de média mais baixos foram os menos afetados pela concorrência no processamento das requisições no servidor, enquanto os clientes com valores de média mais altos foram os mais prejudicados, que esperam mais tempo para a renderização das vistas.

Um caso específico é ilustrado pelos tempos de renderização do cliente C6 na arquitetura convencional TCP. O tempo médio de renderização desse cliente foi bem discrepante dos demais e o tempo mínimo indica que esse cliente teve um tempo de espera bem alto em todos os quadros renderizados. Isso ocorreu porque a biblioteca de *socket* utilizada dispõe de um *buffer* de dados de entrada para cada cliente.

Então, o servidor renderiza todas as vistas requisitadas por C1, não importando se tem uma ou mais requisições a serem renderizadas, para depois renderizar as vistas do próximo cliente. Assim, C6 deve esperar as renderizações de todos os clientes para depois renderizar suas requisições, tendo um tempo elevado na renderização por conta do tempo de espera.

Já na arquitetura convencional UDP não existe esse problema, porque o *buffer* de entrada de dados do servidor é único para todos os clientes, assim, a disputa entre os clientes para a renderização das vistas no servidor é igual para todos.

Os valores mínimos apresentaram dois comportamentos. Por exemplo, os clientes C2, C4 e C5 na Tabela 4.19 obtiveram os tempos mínimos semelhantes ao tempo de renderização dos quadros no servidor, indicando que esses clientes em algum momento da simulação foram beneficiados na renderização dos quadros no servidor, não sofrendo os efeitos causados pela concorrência no processamento das vistas por estarem no início da fila e renderizando as vistas rapidamente, enquanto os clientes C1, C3 e C6 nunca conseguiram esse benefício, esperando mais tempo para renderizar as vistas.

Os valores de máximos podem ocorrer pelo grande acúmulo de mensagens, em casos onde o cliente está no final da fila de requisições e deve esperar a renderização de várias vistas. Os valores máximos muito discrepantes do tempo médio também podem estar relacionados à demora na transferência dos dados do servidor para os clientes.

4.4 Experimento II

Nesta seção são descritos os resultados obtidos na avaliação do desempenho do protótipo para o experimento II. Na simulação deste experimento foram utilizados 6 clientes, o intervalo de tempo entre as requisições foi definido em 150 milissegundos e na arquite-

tura combinada, a cada 10 quadros renderizados utilizando o conceito de resíduos, 1 foi renderizado empregando a arquitetura com quadros completos.

Os computadores utilizados neste experimento foram semelhantes aos do experimento I, utilizando o conceito de LTSP. Entretanto, o servidor dos terminais neste experimento tem poder de processamento inferior comparado com o do experimento I.

Cada cliente foi executado em um terminal. O servidor dos terminais possui 2 processadores AMD Opteron™ modelo 242 de 1.6 GHz, memória disponível de 3 Gbytes, limitada em 1,1 GBytes por processo do usuário e espaço em disco disponível de 2 GBytes para o usuário utilizado.

Esta seção está organizada como segue. Na subseção 4.4.1 são descritos os resultados obtidos na renderização do ambiente virtual empregando as arquiteturas baseadas na técnica 3D *warping*, na subseção 4.4.2 são apresentados os resultados entre a renderização da cena utilizando uma arquitetura com e sem descarte de requisições e, por fim, na subseção 4.4.3 é apresentada a avalização do desempenho do protótipo nas arquiteturas convencional UDP e TCP.

4.4.1 Arquiteturas Baseadas em 3D *Warping*

Esta avaliação está dividida em duas partes. Na subseção 4.4.1.1 são apresentados os resultados da renderização remota nas arquiteturas baseadas em 3D *warping* e na subseção 4.4.1.2 referente à renderização local.

4.4.1.1 Renderização Remota

O desempenho obtido na renderização remota do ambiente virtual nas arquiteturas baseadas na técnica 3D *warping* é apresentado no gráfico da Figura 4.11. Os tempos em cada arquitetura foram computados com a média dos tempos obtidos na renderização remota dos 6 clientes utilizados na simulação.

O melhor desempenho foi obtido pela arquitetura com quadros completos, com tempo médio de renderização de 95 milissegundos durante a simulação. As arquiteturas baseadas em imagens de resíduo obtiveram tempos de renderização remota próximos, a arquitetura

combinada teve tempo médio que variou de 165 a 178 milissegundos e a arquitetura com imagens de resíduo foi a que teve pior desempenho, com tempo médio de 180 milissegundos.

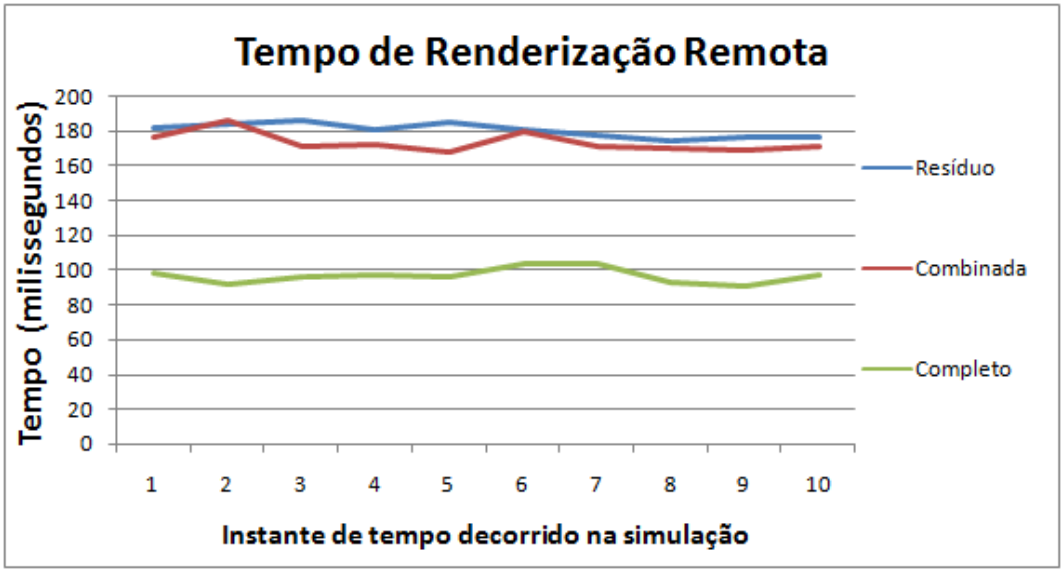


Figura 4.11: Tempo médio do processo de renderização remota nas arquiteturas baseadas na técnica 3D *warping* para o experimento II.

Os resultados obtidos mostraram que o principal delimitador dos tempos de renderização remota em cada arquitetura foi o tempo gasto para renderizar os quadros no servidor. Os quadros completos foram renderizados pelo servidor aproximadamente 58% mais rapidamente do que as imagens de resíduo, justificando a diferença desta arquitetura com as demais.

Nas abordagens que utilizaram imagens de resíduo, a arquitetura combinada obteve melhor desempenho porque, a cada 10 quadros renderizados com a arquitetura baseada em resíduo, 1 foi quadro completo, resultando em melhor desempenho comparado com a arquitetura com imagens de resíduo e justificando o fato de ambas obterem tempos de renderização remota semelhantes.

As medidas estatísticas dos tempos obtidos pelos clientes na renderização remota da cena relativas às arquiteturas com imagens de resíduo, combinada e quadros completos são ilustradas nas Tabelas 4.20, 4.21 e 4.22, respectivamente.

Os tempos mínimos obtidos na arquitetura combinada e a arquitetura com quadros completos foram semelhantes e discrepantes dos valores de mínimos da arquitetura com

imagens de resíduo. Na arquitetura com imagens de resíduo, somente imagens de resíduo foram renderizadas. Os tempos mínimos obtidos por essa arquitetura foram semelhantes ao tempo médio que o servidor demandou para renderizar este tipo de quadro e nas demais arquiteturas, que renderizaram quadros completos, seus valores de mínimo também acompanharam o tempo médio da renderização destes quadros no servidor.

Tabela 4.20: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura com imagens de resíduo.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	112,07	111,04	112,39	118,03	110,20	114,02
Máximo	495,97	379,59	458,27	429,62	471,40	414,67
Média	179,67	177,58	182,39	180,21	183,51	181,19
Desvio padrão	46,47	41,24	47,27	42,74	45,93	40,50
Variação	26%	23%	26%	24%	25%	22%

Tabela 4.21: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura combinada.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	72,66	72,86	71,89	72,84	70,69	72,13
Máximo	401,82	684,66	403,78	460,81	418,51	339,66
Média	173,83	178,98	176,68	170,69	174,44	165,28
Desvio padrão	45,38	74,37	49,90	42,61	45,45	40,52
Variação	26%	42%	28%	25%	26%	25%

Tabela 4.22: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura com quadros completos.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Mínimo	70,12	70,98	71,00	70,87	70,89	71,12
Máximo	262,97	242,30	423,31	222,73	304,76	247,68
Média	93,79	94,06	95,14	93,16	96,16	96,26
Desvio padrão	29,48	26,51	30,20	25,52	32,99	28,76
Variação	31%	28%	32%	27%	34%	30%

A medida de variação foi similar nas arquiteturas baseadas em resíduo, que obtiveram variação em torno de 25%, com exceção do cliente C2 da arquitetura combinada, que obteve medida de variação de 42%. A arquitetura com quadros completos foi a que obteve maior variação na renderização remota das vistas, a menor medida de variação foi do cliente C4 com 27% e a maior foi de C5 com 34%.

A discrepância na medida de variação obtida pelo cliente C2 da arquitetura com imagens de resíduo comparada com os demais clientes indica que, em alguns instantes de tempo na simulação, este cliente foi fortemente prejudicado pela concorrência na renderização das vistas no servidor e, em outros momentos, teve pouca concorrência na renderização das vistas, obtendo grande percentual de variação durante a renderização remota. Nos momentos de grande concorrência, os tempos de renderização remota foram altos por conta do tempo de espera na renderização das vistas e, quando houve pouca concorrência, as vistas foram renderizadas mais rapidamente.

Os percentuais de requisições renderizadas e descartadas no servidor relativos às arquiteturas com imagens de resíduo, combinada e com quadros completos podem ser observados respectivamente nas Tabelas 4.23, 4.24 e 4.25.

Tabela 4.23: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento II utilizando a arquitetura com imagens de resíduo.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartados	49%	45%	45%	48%	44%	47%
Renderizados	51%	55%	55%	52%	56%	53%

Tabela 4.24: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento II utilizando a arquitetura combinada.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartados	44%	51%	44%	46%	51%	49%
Renderizados	56%	49%	56%	54%	49%	51%

Tabela 4.25: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento II utilizando a arquitetura com quadros completos.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
Descartados	14%	12%	11%	9%	10%	10%
Renderizados	86%	88%	89%	91%	90%	90%

As arquiteturas baseadas em resíduo tiveram um percentual de requisições descartadas semelhantes. Comparando as arquiteturas baseadas em resíduo com a arquitetura com quadros completos, o percentual de requisições descartadas foram discrepantes. Essa discrepância está relacionada com o tempo de renderização dos quadros no servidor. Os quadros completos foram renderizados mais rapidamente pelo servidor em comparação

com as imagens de resíduo, então as requisições acumularam-se menos na arquitetura com quadros completos e, consequentemente, menos requisições foram descartadas.

4.4.1.2 Renderização Local

Os tempos médios na renderização local do ambiente virtual nas arquiteturas baseadas na técnica 3D *warping* são ilustrados na Figura 4.12. O gráfico mostra o tempo médio resultante da renderização local dos 6 clientes envolvidos na simulação deste experimento em cada arquitetura. As medidas estatísticas dos tempos obtidos pelos clientes na renderização local da cena relativas às arquiteturas com imagens de resíduo, combinada e com quadros completos são ilustradas nas Tabelas 4.26, 4.27 e 4.28, respectivamente.

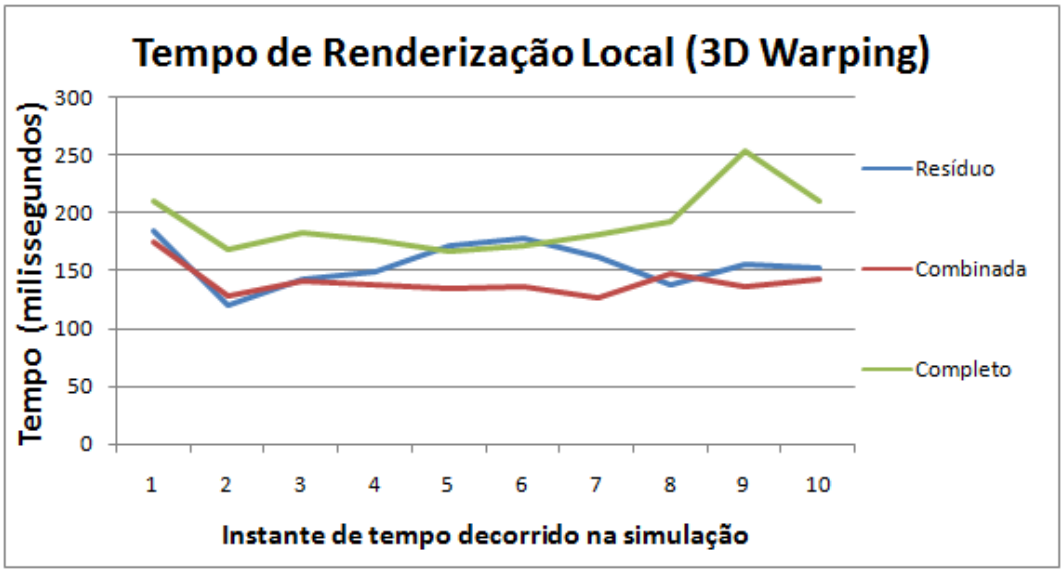


Figura 4.12: Tempo médio do processo de renderização local nas arquiteturas baseadas na técnica 3D *warping* para o experimento II.

A arquitetura com menor tempo de renderização foi a combinada, em que os clientes obtiveram tempos médios de 138 a 144 milissegundos. Na arquitetura com imagens de resíduo, os tempos médios de renderização local alcançados pelos clientes variaram de 147 a 161 milissegundos. Os tempos mais elevados de renderização local foram obtidos na arquitetura com quadros completos que variaram de 182 a 198 milissegundos.

O gráfico mostra um bom comportamento da simulação nos instantes de tempo 1 até 3, em que as curvas referentes aos tempos de renderização de cada arquitetura tiveram

um comportamento semelhante. A partir do instante de tempo 3 até o final da simulação houve uma grande discrepância entre as curvas ilustradas, sinalizando grande variação nos tempos de renderização local.

Tabela 4.26: Medidas estatísticas referentes ao tempo de renderização local para o experimento II utilizando a arquitetura com imagens de resíduo.

	C1	C2	C3	C4	C5	C6
Mínimo	29,80	38,25	34,29	31,79	39,05	36,56
Máximo	505,00	482,62	462,41	583,29	467,69	474,46
Média	153,54	147,47	150,24	159,70	158,04	161,40
Desvio padrão	76,04	75,42	70,70	76,96	73,04	71,71
Variação	50%	51%	47%	48%	46%	44%

Tabela 4.27: Medidas estatísticas referentes ao tempo de renderização local para o experimento II utilizando a arquitetura combinada.

	C1	C2	C3	C4	C5	C6
Mínimo	31,67	36,60	30,34	33,65	34,19	40,47
Máximo	620,84	517,43	553,42	358,65	360,64	480,41
Média	139,71	141,86	140,59	142,58	144,18	138,10
Desvio padrão	70,13	66,18	67,80	60,50	63,06	66,61
Variação	50%	47%	48%	42%	44%	48%

Tabela 4.28: Medidas estatísticas referentes ao tempo de renderização local para o experimento II utilizando a arquitetura com quadros completos.

	C1	C2	C3	C4	C5	C6
Mínimo	46,28	46,27	40,49	50,26	48,73	49,23
Máximo	749,76	678,71	607,91	439,92	634,13	443,81
Média	182,61	187,49	198,78	194,11	191,29	191,83
Desvio padrão	89,15	81,53	84,55	79,95	81,86	78,84
Variação	49%	43%	43%	41%	43%	41%

Essa variação pode ser observada nas tabelas de medidas estatísticas referente a cada arquitetura. A variação nas três arquiteturas foi alta, entre 41 a 51%. Nas arquiteturas baseadas em imagens de resíduo, as medidas de variação foram semelhantes e na arquitetura com quadros completos apresentou uma pequena diferença, sendo a que obteve menor variação nos tempos de renderização local.

O grande percentual de variação obtido está relacionado ao poder de processamento do servidor LTSP (servidor dos terminais) que foi limitado para atender à demanda de

requisições (renderização das vistas) dos terminais. Então, essa sobrecarga do servidor LTSP fez com que ele executasse algumas requisições rapidamente e outras lentamente. Essa instabilidade no servidor LTSP contribuiu para o aumento nos tempos médios e máximos de renderização local e o aumento na medida de variação. O gráfico ilustra momentos de instabilidade na renderização local das vistas, observados nos instantes de tempo 3 até 8 na arquitetura com imagens de resíduo e do 8 ao 10 na arquitetura com quadros completos. Nesses períodos da simulação, os tempos de renderização local obtidos sofreram elevação.

A tendência é que, quanto menor for a demanda das requisições, menor será a instabilidade do servidor dos terminais. Um exemplo da relação entre a instabilidade do servidor LTSP com a demanda das requisições pode ser observado na comparação dos tempos de renderização obtidos pela arquitetura com imagens de resíduo deste teste que utilizou 6 clientes e do teste da subseção 4.4.2.2, em que foram utilizados somente 2 clientes. Com a redução do número de clientes para 2, a renderização local da arquitetura com imagens de resíduo se tornou mais estável, com medida de variação de aproximadamente 30% menor e tempos médios de renderização cerca de 65% mais rápidos.

Embora o servidor LTSP neste teste apresentou baixo desempenho para a renderização local, as vistas renderizadas localmente obtiveram boa qualidade (não apresentaram um grande número de artefatos). Isto ocorreu porque a diferença entre os tempos de renderização local e remota (subseção 4.4.1.1) deste teste foi pequena, porque o tempo gasto para renderizar as vistas localmente foi alto, aproximando os tempos de renderização local e remota. Quando isso ocorre, as amostras (imagens) transmitidas pelo servidor estão disponíveis rapidamente para os clientes, em alguns casos antes mesmo da renderização local terminar, permitindo que poucas vistas sejam renderizadas a partir de uma única amostra até que uma nova seja recebida, produzindo vistas com boa qualidade.

Porém, o baixo desempenho do servidor LTSP fez com que a velocidade de navegação do observador (cliente) no ambiente virtual fosse lenta, porque o tempo gasto para renderizar as vistas localmente foi alto, e o intervalo entre as requisições também foi alto, uma vez que o intervalo das requisições foi calculado pelo tempo especificado nos parâmetros

da simulação somado com o tempo de renderização local de cada vista.

Os valores de máximos nas três arquiteturas avaliadas denotam momentos em que as vistas foram renderizadas com tempos altos, com grande discrepância com relação aos tempos médios obtidos, devido às instabilidades do servidor LTSP na renderização das vistas. Os valores de mínimos indicam os melhores tempos obtidos na renderização das vistas. Observa-se que, na arquitetura com imagens de resíduo, os clientes obtiveram os valores de mínimos menores em comparação com as demais. Na arquitetura combinada, os valores de mínimos tiveram um pequeno acréscimo com relação à arquitetura com imagens de resíduo e a arquitetura com quadros completos foi a que os clientes obtiveram os valores de mínimos mais altos.

A diferença nos valores de mínimos entre as arquiteturas denota a qualidade das vistas renderizadas em cada arquitetura. Nas arquiteturas que utilizam imagens de resíduo, as amostras possuem qualidade inferior (maior número de artefatos) em comparação com os quadros completos, devido à degradação da qualidade das amostras, como descrito no início da subseção 3.1.2. Assim, as vistas renderizadas na arquitetura com quadros completos mapeiam mais pixels para a imagem destino, demandando mais tempo na renderização e obtendo tempos mínimos mais altos. A arquitetura combinada obteve tempos mínimos maiores que a arquitetura com imagens de resíduo porque, a cada 10 amostras recebidas, 1 foi quadro completo.

4.4.2 Comparação entre Arquitetura com Descarte de Requisições e Arquitetura sem Descarte de Requisições

Nesta avaliação são apresentados os resultados obtidos na renderização da cena utilizando a arquitetura com imagens de resíduo com e sem descarte de requisições. O número de clientes utilizados na simulação foi reduzido para 2. A arquitetura com imagens de resíduo foi escolhida por ser a que tem maior probabilidade de que as requisições acumulem-se no servidor, uma vez que a imagem de resíduo foi o quadro que demandou mais tempo entre os quadros renderizados pelo servidor. O número de clientes foi reduzido porque, em um dos testes, não é empregado o método de descarte de requisições e a utilização de

vários clientes pode sobrecarregar o servidor devido ao acúmulo de requisições.

A seguir, na subseção 4.4.2.1 são apresentados os resultados da renderização remota e na subseção 4.4.2.2 referente à renderização local.

4.4.2.1 Renderização Remota

O desempenho dos clientes na renderização remota da cena referente às arquiteturas com imagens de resíduo com e sem descarte de requisições são ilustrados na Figura 4.13. Os tempos exibidos representam a média dos tempos obtidos na renderização remota dos 2 clientes utilizados na simulação em cada arquitetura. O percentual de requisições descartadas e renderizadas na arquitetura com descarte podem ser observados na Tabela 4.29.

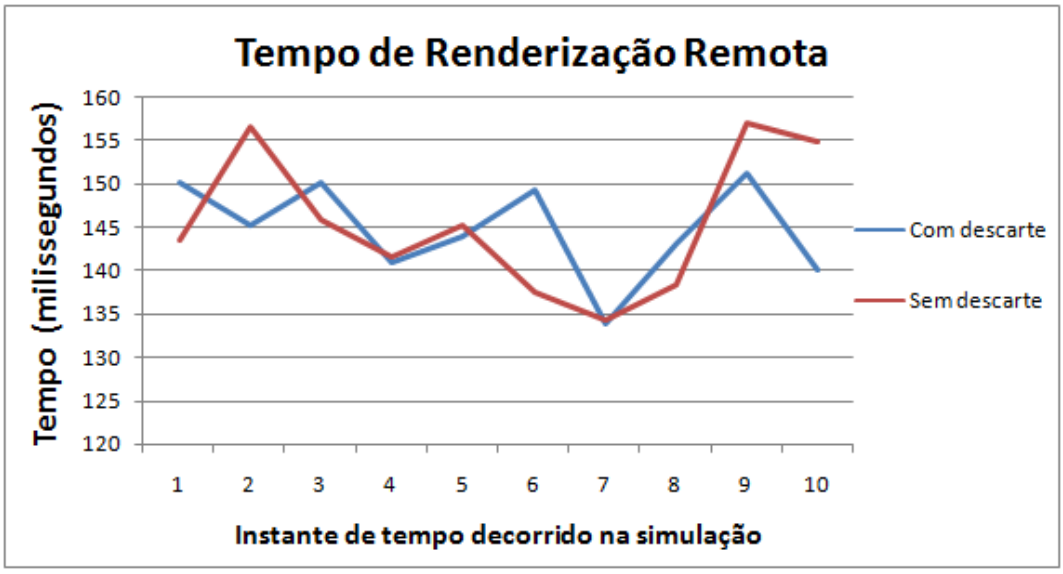


Figura 4.13: Tempo médio do processo de renderização remota entre as arquiteturas com imagens de resíduo com e sem descarte de requisições para o experimento II.

Tabela 4.29: Percentual de requisições descartadas e renderizadas pelo servidor para o experimento II utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	C2
Descartados	1%	0%
Renderizados	99%	100%

Os tempos médios de renderização remota obtidos pelos clientes em ambas as arquiteturas foram semelhantes, variando de 142 a 149 milissegundos. O percentual de descarte

de requisições foi extremamente baixo, de 1% para o cliente C1 e 0% para C2.

A semelhança nos tempos médios de renderização e o pequeno percentual de quadros descartados indica que a renderização remota em ambas as arquiteturas foi semelhante e o uso de descarte não influenciou nos resultados, demonstrando que praticamente não ocorreu acúmulo de requisições no servidor.

O fato do acúmulo de requisições ter sido baixo está relacionado à demanda de requisições. Neste teste praticamente não houve acúmulo de requisições porque o intervalo de tempo entre as requisições foi longo e o número de clientes foi de apenas 2, então o servidor conseguiu renderizar as vistas requisitadas em tempo adequado e, quando novas vistas eram requisitadas, o *buffer* de requisições estava vazio, não havendo acúmulo de requisições.

O tempo de intervalo entre as requisições é dado pela soma do tempo de intervalo especificado nos parâmetros da simulação com o tempo de renderização local. Neste experimento, o tempo de intervalo definido na simulação foi de 150 milissegundos e os tempos de renderização local neste experimento foram altos, então o intervalo de tempo entre as requisições foram longos e, por conta disso, o percentual de requisições descartadas foi baixo, porque poucas requisições ficaram acumuladas no *buffer* do servidor.

O gráfico mostra que, nos instantes de tempo 2, 6 e 9 na simulação, os momentos em que os tempos de renderização remota foram mais discrepantes entre as arquiteturas avaliadas. A variação no tempo de renderização remota obtidos pelas médias dos tempos dos clientes em cada arquitetura está relacionada com a variação nos tempos de transferência das imagens do servidor para os clientes e a concorrência na renderização das vistas.

As medidas estatísticas dos tempos de renderização remota da cena de cada cliente relativa à arquitetura com imagens de resíduo com e sem descarte de requisições são ilustradas nas Tabelas 4.30 e 4.31, respectivamente.

As medidas de mínimo, média e variação foram semelhantes em ambas as arquiteturas. Os valores de máximo foram maiores na arquitetura sem descarte de requisições, especialmente no cliente C1 que teve 1% dos quadros descartados indicando que, nos momentos de acúmulo das requisições, o tempo de espera sofrido na renderização das vistas resultou

em tempos de renderização remota altos.

Tabela 4.30: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	C2
Mínimo	115,53	114,96
Máximo	261,37	268,58
Média	146,28	142,86
Desvio padrão	32,45	27,91
Variação	22%	20%

Tabela 4.31: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura com imagens de resíduo sem descarte de requisições.

	C1	C2
Mínimo	115,93	115,46
Máximo	365,16	302,95
Média	149,01	142,09
Desvio padrão	33,35	35,25
Variação	22%	25%

4.4.2.2 Renderização Local

Os tempos médios da renderização local obtido entre as arquiteturas com imagens de resíduo com e sem descarte de requisições são ilustrados na Figura 4.14. Os tempos exibidos representam a média dos tempos obtidos na renderização local dos 2 clientes utilizados na simulação em cada arquitetura. As medidas estatísticas relativas às arquiteturas com imagens de resíduo com e sem descarte de requisições são ilustradas nas Tabelas 4.32 e 4.33, respectivamente.

Os tempos de renderização local dos quadros obtidos pelos clientes em ambas as arquiteturas foram semelhantes, embora a arquitetura com descarte apresentou resultados melhores na renderização, com tempos médios de renderização e medida de variação menores, mas essa diferença foi pequena em comparação com a arquitetura sem descarte de requisições.

Os tempos médios na arquitetura com descarte foram de 63ms para o cliente C1 e 65ms para C2 e a medida de variação foi de 25% e 28% para os respectivos clientes.

Na arquitetura sem descarte, os tempos médios foram de 69ms para ambos os clientes e a medida de variação foi de 35% para C1 e de 31% para C2. Os valores de mínimos obtidos foram praticamente iguais em todos os clientes de ambas as arquiteturas, em torno de 46 milissegundos. Os tempos máximos foram maiores na arquitetura sem descarte, acompanhando os valores de variação que também foram maiores nesta arquitetura.

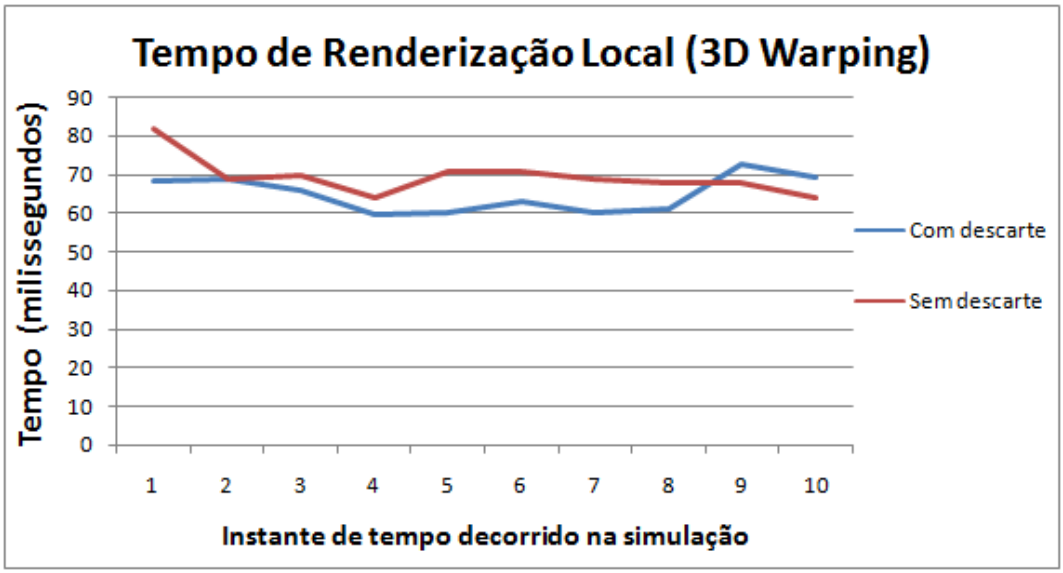


Figura 4.14: Tempo médio do processo de renderização local entre as arquiteturas com imagens de resíduo com e sem descarte de requisições para o experimento II.

Tabela 4.32: Medidas estatísticas referentes ao tempo de renderização local para o experimento II utilizando a arquitetura com imagens de resíduo com descarte de requisições.

	C1	C2
Mínimo	46,22	46,73
Máximo	147,55	172,81
Média	63,96	65,66
Desvio padrão	16,14	18,37
Variação	25%	28%

Tabela 4.33: Medidas estatísticas referentes ao tempo de renderização local para o experimento II utilizando a arquitetura com imagens de resíduo sem descarte de requisições.

	C1	C2
Mínimo	46,73	46,48
Máximo	237,99	223,85
Média	69,68	69,03
Desvio padrão	24,17	21,64
Variação	35%	31%

Os resultados na renderização remota dos quadros para este teste, descrito na subseção 4.4.2.1, mostraram que a diferença entre as duas arquiteturas foi mínima, de acordo com os valores das medidas estatísticas analisadas. Então, a renderização remota não contribuiu para que houvesse diferenças na medida de variação e valores máximos entre as arquiteturas com e sem descarte de requisições na renderização local.

Esta variação, embora pequena, está associada a momentos de instabilidade no servidor LTSP, como descrito na subseção 4.4.1.2 em que foi testada a renderização local nas arquiteturas baseadas em 3D *warping*.

Nota-se que os valores de média e variação obtidos pelos clientes neste teste foram bem menores que os resultados apresentados na subseção 4.4.1.2 referente à arquitetura com imagens de resíduo, devido à redução no número de clientes para 2, diminuindo a demanda de requisições ao servidor LTSP para a renderização das vistas e, consequentemente, apresentando melhores resultados na renderização local.

4.4.3 Arquitetura Convencional UDP e TCP

O desempenho resultante na renderização do ambiente virtual utilizando as arquiteturas convencionais que empregam o protocolo de transporte TCP e UDP é ilustrado na Figura 4.15 e as medidas estatísticas relativas a essas arquiteturas nas Tabelas 4.34 e 4.35, respectivamente. O gráfico mostra o tempo médio resultante da renderização remota dos 6 clientes envolvidos na simulação deste experimento em cada arquitetura.

O tempo médio de renderização remota obtido pela arquitetura convencional TCP variou de 74 a 100 milissegundos e na arquitetura convencional UDP de 21 a 43 milissegundos. Nas arquiteturas convencionais, como não há descarte de requisições e nem renderização local, a variação dos tempos de renderização observados no gráfico e pelos valores de variação de cada cliente está relacionada com a variação nos tempos de transferência dos quadros do servidor para os clientes e no tempo de espera de cada requisição sofrido pela concorrência na renderização das vistas requisitadas pelos clientes ao servidor ou pelo acúmulo das requisições no *buffer* do servidor.

A Figura 4.15 mostra que a maior variação no tempo de renderização remota foi regis-

trada no início da simulação nos instantes de tempo 1 até o 4, em ambas as arquiteturas. Na arquitetura convencional UDP, os tempos de renderização iniciais foram de 43 milissegundos, baixando até 23, que se manteve até o final da simulação.

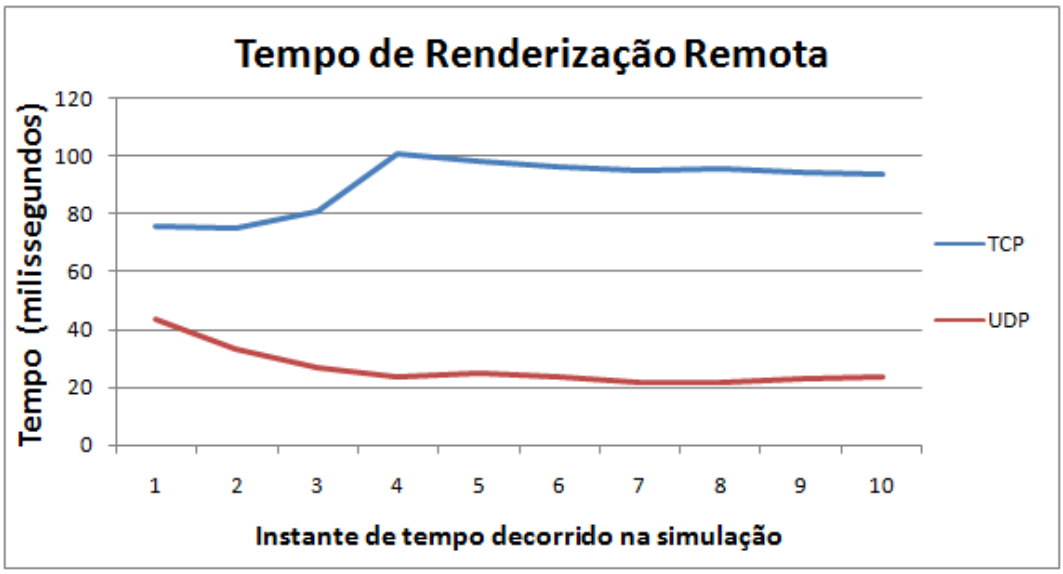


Figura 4.15: Tempo médio do processo de renderização remota entre as arquiteturas convencionais TCP e UDP para o experimento II.

Tabela 4.34: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura convencional TCP.

	C1	C2	C3	C4	C5	C6
Mínimo	16,08	16,19	15,34	15,34	30,39	229,12
Máximo	84,96	152,58	98,96	127,93	148,86	396,93
Média	33,90	41,78	28,32	41,94	62,74	333,60
Desvio padrão	18,07	17,18	19,59	19,83	15,60	63,52
Variação	53%	41%	69%	47%	25%	19%

Tabela 4.35: Medidas estatísticas referentes ao tempo de renderização remota para o experimento II utilizando a arquitetura convencional UDP.

	C1	C2	C3	C4	C5	C6
Mínimo	15,66	15,81	15,96	15,66	15,93	15,77
Máximo	95,82	81,14	55,40	73,00	80,98	143,03
Média	33,10	24,51	20,46	31,01	31,13	20,10
Desvio padrão	9,64	8,57	5,26	14,42	14,29	12,38
Variação	29%	35%	26%	46%	46%	62%

Na arquitetura convencional TCP, o tempo inicial foi de 75ms, posteriormente, sobe para 100ms indicando um momento em que a transferência das imagens do servidor para os

clientes foi lenta e as requisições nesse período se acumularam no servidor. No instante de tempo 4 até o final da simulação observa-se que a velocidade de transferência das imagens pela rede foi normalizada e as requisições foram diminuindo no servidor e o tempo de renderização foi baixando no decorrer da simulação chegando ao final com 93ms.

Os tempos médios obtidos pelos clientes na arquitetura convencional UDP variou de 20 a 33ms e na arquitetura convencional TCP foi de 28 a 333ms. A grande discrepância entre os tempos de renderização obtidos pelos clientes da arquitetura convencional TCP está relacionada com a estrutura dos *buffers* de entrada de dados da biblioteca de *socket* utilizada, que possui um *buffer* para cada cliente e na UDP é um *buffer* de entrada de dados único para todos os clientes. Assim, a concorrência na renderização das vistas no servidor na arquitetura convencional TCP não foi igual para todos os clientes como na UDP.

A maioria dos clientes obteve tempos mínimos próximos ao do tempo médio de renderização dos quadros no servidor, indicando bom desempenho nesses momentos. Para os clientes C5 e C6 da arquitetura TCP, os tempos mínimos indicam que estes clientes foram mais afetados com a concorrência na renderização das vistas no servidor e sempre tiveram que esperar a renderização das requisições de outros clientes.

Os tempos máximos indicam momentos em que as requisições dos clientes tiveram que esperar bastante tempo na fila de requisições do servidor ou que a transferência dos dados do servidor para os clientes foi lenta, gerando tempos de renderização remota altos. Os valores de máximos, assim como as medidas de variação obtidas entre os clientes em ambas as arquiteturas, apresentaram grande discrepância. Essa discrepância está associada à grande concorrência na renderização das vistas no servidor porque, nas arquiteturas convencionais, os tempos em que as requisições das vistas foram realizadas pelos clientes ao servidor foram próximos.

4.5 Discussão

O tempo de renderização remota da cena nas arquiteturas baseadas em 3D *warping* foi similar nos dois experimentos, ilustrados nas Figuras 4.6 e 4.11. As maiores discrepâncias

observadas foram as diferenças na medida de variação e nos tempos médios obtidos na arquitetura com quadros completos, que foram maiores no experimento II. A medida de variação mais elevada está associada à variação na renderização local dos quadros, que foi maior no experimento II.

Em comparação com o experimento I, os tempos médios de renderização remota obtidos na arquitetura com quadros completos foram maiores no experimento II demostrando que a transferência dos quadros do servidor para os clientes foi mais lenta neste teste. Além disso, nota-se que nos testes com renderização remota, em que foram utilizadas as arquiteturas baseadas em imagens de resíduo, os tempos médios de renderização foram semelhantes em ambos os experimentos, sinalizando que o tempo gasto na transferência dos dados foi similar nestes testes.

O melhor desempenho alcançado na renderização remota baseada em 3D *warping* foi na arquitetura com quadros completos. Em segundo lugar ficou a arquitetura combinada e os piores resultados foram obtidos na arquitetura com imagens de resíduo. A diferença entre os tempos médios obtidos nas arquiteturas baseadas em imagens de resíduo foi pequena, cerca de 10%. Já a diferença entre as arquiteturas baseadas em imagens de resíduo e a arquitetura com quadros completos foi grande, em torno de 50%. O desempenho obtido na renderização remota nas arquiteturas baseadas em 3D *warping* está associado aos tempos gastos para renderizar os quadros no servidor em cada arquitetura. Os quadros completos foram renderizados cerca de 58% mais rápidos do que as imagens de resíduo, justificando as diferenças nos tempos de renderização remota obtidos entre as arquiteturas.

Os resultados obtidos na renderização local do ambiente virtual nas arquiteturas baseadas em 3D *warping* foram discrepantes entre os dois experimentos realizados, observados nas Figuras 4.7 e 4.12. No experimento II, a medida de variação foi alta, assim como os tempos médios, desvio padrão e tempos máximos. Os valores altos estão relacionados ao poder de processamento limitado do servidor LTSP utilizado, que ficou sobrecarregado devido à grande demanda de requisições feitas pelos clientes para o processamento das vistas. No experimento I, a renderização local foi mais estável, com medida de variação, tempos médios, desvio padrão e tempos máximos menores.

As velocidades de navegação dos clientes (observador) no ambiente virtual durante a simulação foram diferentes para cada experimento. No experimento I, os clientes movimentaram-se no ambiente virtual mais rapidamente, enquanto no experimento II mais lentamente. Essa diferença está relacionada ao tempo de renderização local obtido em cada experimento. O tempo de intervalo entre as requisições definido nos parâmetros da simulação foi de 150ms em ambos experimentos, somado com o tempo de renderização local. Então, como no experimento I a renderização local foi mais rápida, as requisições tiveram intervalos de tempo menores que no experimento II, e a navegação dos clientes no ambiente virtual foi mais rápida.

Outro resultado discrepante entre os resultados da renderização local nos experimentos realizados foi a qualidade das vistas renderizadas. As vistas renderizadas no experimento II apresentaram maior qualidade (vistas com menor presença de artefatos) em comparação com o experimento I. Embora o servidor LTSP no experimentou II apresentou ser de baixo desempenho em comparação com o experimento I, as vistas renderizadas localmente obtiveram boa qualidade. Isto ocorreu porque a diferença entre os tempos de renderização local e remota foi pequena no experimento II, porque o tempo gasto para renderizar as vistas localmente foi alto, aproximando os tempos de renderização local e remota. Quando isso ocorre, as amostras (imagens) transmitidas pelo servidor estão disponíveis rapidamente para os clientes, em alguns casos antes mesmo da renderização local terminar, permitindo que poucas vistas sejam renderizadas a partir de uma única amostra até que uma nova seja recebida, produzindo vistas com boa qualidade.

Outro fator que comprova que no experimento II as vistas renderizadas localmente obtiveram maior qualidade foram os percentuais de quadros descartados pelo servidor devido ao acúmulo de requisições. No experimento II, o percentual de descarte foi menor em comparação com o experimento I, cerca de 30% nas arquiteturas baseadas em resíduo e 55% na arquitetura com quadros completos. Quanto menor o percentual de descarte, menos vistas foram renderizadas localmente a partir de uma única amostra.

A qualidade das vistas e o tempo gasto para renderizar os quadros localmente nas arquiteturas baseadas em 3D *warping* foram diretamente proporcionais à quantidade de

pixels mapeados. A diferença entre os tempos gastos na renderização local das vistas foi pequena, em torno de 10 a 12% entre as arquiteturas baseadas em imagens de resíduo para a de quadros completos.

Assim, a arquitetura que apresentou melhores resultados na renderização local foi a com quadros completos, porque as amostras utilizadas foram de qualidade superior em comparação com as imagens de resíduo. A arquitetura combinada fica em segundo lugar entre as que obtiveram os melhores resultados na renderização local, porque a cada 10 quadros renderizados um foi completo, minimizando o problema da degradação de qualidade das imagens de resíduo. A arquitetura com imagens de resíduo foi a que obteve os piores resultados, por ser a que renderizou as vistas apresentando um maior número de artefatos.

O teste referente à arquitetura com imagens de resíduos com e sem descarte de requisições foi discrepante entre os experimentos. No experimento I (Figuras 4.8 e 4.9), a arquitetura sem descarte obteve tempos de renderização remota extremamente altos, em comparação com a arquitetura com descarte e na renderização local, os clientes obtiveram tempos médios muito baixos, sinalizando que poucos pixels foram mapeados da imagem fonte para a destino durante toda a simulação, resultando em uma visualização das vistas de baixa qualidade.

O baixo desempenho obtido por essa arquitetura está relacionado ao acúmulo de requisições no servidor, uma vez que elas não foram descartadas. As requisições acumulavam-se no *buffer* de requisições do servidor e esse acúmulo aumentava com o decorrer da simulação porque mais requisições eram feitas.

Já na arquitetura com descarte, os clientes obtiveram tempos médios de renderização remota e local baixos em comparação com a arquitetura sem descarte. Para isso, foram descartados cerca de 9% dos quadros para o cliente C1 e 21% para C2, eliminando a sobrecarga do servidor com relação ao acúmulo de requisições e os clientes renderizaram as vistas descartadas localmente empregando 3D *warping*.

No experimento II (Figuras 4.13 e 4.14), as arquiteturas com e sem descarte de requisições foram similares na renderização local e remota do ambiente virtual. O percentual

de quadros descartados foi mínimo, com 1% dos quadros descartados para somente um dos clientes envolvido na simulação, sinalizando que praticamente não houve acúmulo de requisições e as poucas requisições acumuladas não influenciaram nos tempos de renderização obtidos. O baixo percentual de descarte foi obtido porque o intervalo de tempo entre as requisições no experimento II foi maior que no experimento I, gerando menos acúmulo de requisições no servidor.

O descarte de requisições no servidor demostrou ser um recurso eficiente na redução de sua sobrecarga e, assim, permitiu o atendimento a um número maior de clientes. Além disso, os quadros descartados no servidor foram renderizados localmente pelos clientes, que necessitavam somente do resíduo ou quadro completo relativo à última vista requisitada para gerar a vista compensada.

Os resultados obtidos da avaliação do processo de renderização remota do ambiente virtual nas arquiteturas convencionais TCP e UDP (Figuras 4.10 e 4.15) foram semelhantes, com exceção que, no experimento II, houve uma variação maior no tempo de transferência das imagens do servidor para os clientes na arquitetura convencional TCP.

A renderização remota dos quadros na arquitetura convencional UDP foi mais rápida que na TCP, cerca de 28% no experimento I e 75% no experimento II. Essa diferença entre os experimentos ocorreu porque na arquitetura TCP, no experimento II, a transferência das imagens do servidor para os clientes foi mais lenta no início da simulação e as requisições acumularam-se no servidor nesse período. Então, os tempos médios de renderização obtidos foram afetados durante a simulação pelo tempo de espera na renderização das vistas no servidor causado pelo acúmulo das requisições.

Em ambos os experimentos, a arquitetura convencional TCP apresentou problemas com relação aos *buffers* de entrada de dados da biblioteca de *socket* utilizada, que disponibilizou um *buffer* para cada cliente, fazendo com que a concorrência na renderização das vistas no servidor fosse desigual entre os clientes. A arquitetura convencional UDP não apresentou esse problema nos experimentos realizados por dispor de um *buffer* de entrada de dados único para todos os clientes, tonando a concorrência na renderização das vistas igual para todos os clientes.

A concorrência na renderização das vistas requisitadas ao servidor foi maior nas arquiteturas convencionais UDP e TCP em comparação com as arquiteturas baseadas em 3D *warping*, pois o intervalo entre as requisições foi menor nas arquiteturas convencionais, uma vez que não houve renderização local. Essa concorrência fez com que os tempos médios obtidos fossem discrepantes entre os clientes, ao contrário das arquiteturas baseadas em 3D *warping*, em que os tempos médios obtidos pelos clientes foram semelhantes.

O tamanho médio das imagens produzidas no processo de renderização no servidor nas diferentes arquiteturas empregadas é mostrado na Tabela 4.36. Os tamanhos das imagens exibidas são referentes às imagens comprimidas utilizando o método JPEG com 50% de qualidade.

Tabela 4.36: Tamanho médio das imagens renderizadas no servidor.

	Dimensão	**Tamanho (Kbytes)**
Quadro resíduo	500×500	6,33
Quadro completo	500×500	15,92
Quadro convencional	400×400	10,40
Profundidade	500×500	5,36

Entre os parâmetros pré-definidos para a execução dos experimentos, o protocolo de transporte UDP demonstrou ser mais eficiente que o TCP, transmitindo os quadros mais rapidamente. O intervalo de tempo entre as requisições definido em 150 milissegundos, no experimento I, indicou ser muito pequeno devido ao grande percentual de requisições descartadas. Na arquitetura combinada, a especificação de que a cada 10 quadros renderizados utilizando a abordagem de resíduo 1 quadro completo seja renderizado, reduziu o problema de degradação da qualidade das vistas causado pela abordagem de resíduo. O emprego de uma imagem com dimensão maior do que a área de exibição dos quadros nos clientes foi útil para prevenir a presença de artefatos nas vistas renderizadas localmente, embora o quadro transmitido teve tamanho maior em comparação com quadro da arquitetura convencional, que possui dimensão igual à da área de exibição dos quadros no cliente.

Comparando-se a arquitetura convencional UDP com a arquitetura com quadros completos (a arquitetura baseada em 3D *warping* que obteve melhor desempenho), a arquite-

tura convencional obteve melhor desempenho na renderização remota dos quadros. Como o desempenho na renderização do ambiente virtual foi delimitado pelo tempo gasto na renderização das vistas no servidor em todos os testes realizados, a arquitetura convencional foi aproximadamente 62% mais rápida do que a arquitetura com quadros completos, porque esse percentual foi a diferença entre os tempos de renderização dos quadros convencional e completo no servidor.

Embora, a imagem de resíduo seja o menor quadro de todos os renderizados (Tabela 4.36), a técnica 3D *warping* requer a imagem referente à informação de profundidade, então, com base na soma dessas duas imagens, além da arquitetura convencional ser mais eficiente na renderização dos quadros no servidor, ela também foi mais eficiente na transmissão das informações dos dados do servidor para os clientes, por transmitir menos informações.

Para que a abordagem baseada em imagens de resíduo se torne mais eficiente na transmissão dos dados, seria necessária a redução da dimensão das imagens renderizadas para a mesma dimensão da arquitetura convencional. Assim, a soma do tamanho da imagem de resíduo com a informação de profundidade torna-se menor que o quadro renderizado na arquitetura convencional. Porém, a utilização de imagens com dimensão igual à da área de visualização no cliente pode apresentar um maior número de artefatos nas vistas renderizadas via 3D *warping*, como descrito na seção 3.3.

Observa-se que a diferença entre as imagens resíduo e quadros completos é grande com relação ao tamanho dos dados que foram transmitidos do servidor para os clientes. Embora, os quadros completos demandem mais tempo na sua transmissão pela rede por possuírem tamanhos maiores que a imagem de resíduo, essa diferença não afetou o desempenho obtido na renderização remota na mesma proporção que o tempo gasto pelo servidor na renderização das vistas. Este fato ocorreu porque para computar as imagens de resíduo no servidor de renderização foi necessária a renderização via 3D *warping*, que foi executada pela CPU do servidor. Em oposição, a arquitetura com quadros completos não computa a imagem de resíduo, e a cena foi renderizada somente pela GPU do servidor de renderização. Por isso, o melhor desempenho obtido na renderização remota das ar-

quiteturas baseadas em 3D *warping* foi o da arquitetura com quadros completos, porque os quadros completos foram renderizados pelo servidor mais rapidamente em comparação com as imagens de resíduo.

Em contrapartida, levando-se em conta a arquitetura que possui o melhor desempenho na transferência dos dados pela rede e que mantém um bom percentual de qualidade nas vistas renderizadas, a arquitetura combinada foi a melhor, porque transmitiu os quadros rapidamente e a cada 10 quadros renderizados recebeu um completo, reduzindo o problema da degradação da qualidade das vistas renderizadas.

A arquitetura convencional UDP, embora foi a mais eficiente na renderização remota e na transmissão dos quadros, não dispõe de um mecanismo para a renderização local das vistas e o cliente fica ocioso esperando os quadros do servidor. Quando há atrasos no recebimento ou problemas na transmissão dos dados, a visualização dos quadros nos clientes pode ser afetada.

Por outro lado, as arquiteturas baseadas em 3D *warping* demandaram muito tempo para a renderização das vistas no servidor, por necessitar da execução de um maior número de tarefas em comparação com a arquitetura convencional. Uma forma de aprimorar a renderização dos quadros seria a paralelização da execução das tarefas, que atualmente são executadas sequencialmente.

As tarefas de renderização da cena e renderização utilizando a técnica 3D *warping* podem ser executadas em paralelo, assim como a compressão das imagens referente ao quadro e a informação de profundidade. Outra possibilidade, relacionada às tarefas que envolvem processamento de imagens, seria dividir as imagens em regiões e processar essas regiões em paralelo. Essa abordagem poderia ser usada nas tarefas de compressão, geração da imagem de resíduo e renderização via 3D *warping*. Uma biblioteca de código aberto que poderia ser utilizada seria a Intel® *Threading Building Blocks* [30], que provê o desenvolvimento de aplicações com computação paralela, maximizando o desempenho dos múltiplos núcleos (*core*) de processamento disponíveis em um computador [30].

CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou quatro abordagens diferentes para renderização remota e sob demanda da cena 3D destinada a computadores com recursos gráficos limitados, que não dispõem de uma GPU ou computadores que possuam placas aceleradoras gráficas, mas essas sejam limitadas em termos de processamento.

A renderização da cena 3D é feita remotamente, sob demanda e emprega a técnica de Renderização Baseada em Imagens 3D *warping* para renderizar vistas localmente no cliente, usando somente o último quadro recebido. Assim, o cliente é capaz de renderizar novas vistas enquanto espera por dados requisitados ao servidor ou até mesmo quando há problemas em sua transmissão ou atraso no recebimento.

O protótipo desenvolvido permite a renderização do ambiente virtual empregando quatro arquiteturas diferentes, a arquitetura convencional, em que a renderização é feita somente remotamente, não há renderização local, a arquitetura com imagens de resíduo, que busca a otimização na transferências dos dados entre servidor e cliente e permite a renderização local da cena, a arquitetura com quadros completos, que possibilita a renderização local reduzindo a presença de artefatos nas vistas produzidas e a arquitetura combinada, formada pelas arquiteturas com quadros completos e imagens de resíduo, buscando transmitir os dados de forma eficiente e manter a qualidade das vistas renderizadas.

A avaliação do protótipo foi realizada com base em uma simulação da interação dos usuários no ambiente virtual. O desempenho do protótipo foi avaliado com base em dois experimentos realizados, nos quais foram analisados os tempos obtidos na renderização remota e local da cena 3D. Os testes de desempenho contemplaram computadores com capacidade de processamento gráfico limitada.

O desempenho na renderização do ambiente virtual foi delimitado pelo tempo gasto na renderização das vistas no servidor em todos os testes realizados. A arquitetura convenci-

onal foi a mais rápida na renderização remota e transmissão dos dados, em segundo lugar, a arquitetura com quadros completos, em terceiro lugar, a arquitetura com imagens de resíduo e quadros completos e o pior desempenho foi obtido pela arquitetura com imagens de resíduo.

A arquitetura convencional foi a mais eficiente na renderização remota e na transmissão dos quadros, entretanto não dispõe de um mecanismo para a renderização local das vistas e o cliente fica ocioso esperando os quadros do servidor. Quando há atrasos no recebimento ou problemas na transmissão dos dados, a visualização dos quadros nos clientes pode ser afetada. Por outro lado, as demais arquiteturas baseadas em 3D *warping* que dispõem de um método para a renderização local das vistas demandaram muito tempo para a renderização dos quadros no servidor, por necessitar da execução de um maior número de tarefas em comparação com a arquitetura convencional.

Algumas propostas para estender o trabalho desenvolvido são indicadas a seguir. A renderização remota do ambiente virtual poderia ser aprimorada através da paralelização da execução das tarefas que compõem o processo de renderização das vistas no servidor, que atualmente são executadas sequencialmente.

A avaliação do desempenho do protótipo poderia ser realizada com novos experimentos, utilizando-se um número maior de clientes, com computadores heterogêneos. Além disso, a análise dos resultados da renderização dos quadros no servidor poderia levar em conta outros modelos 3D, com um maior número de polígonos, representando um ambiente virtual com mais realismo.

Pretende-se também realizar uma pesquisa mais aprofundada sobre os métodos de compressão de imagens, buscando um método que reduza o tempo gasto na execução dessa tarefa e que diminua o tamanho das imagens transmitidas pela rede e uma análise quantitativa da qualidade das vistas renderizadas poderia ser realizada. Finalmente, planeja-se uma investigação mais detalhada para avaliar o impacto do tráfego de dados na rede com relação aos resultados obtidos na renderização remota da cena.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] 0-360.com. 0-360 Panoramic Optic^{TM}. Lente que permite ser anexada em uma câmera fotográfica para capturar imagens de panorama com campo de visão de 360°. `http://www.0-360.com/camera.asp`, acesso em: 09 de março de 2008.

[2] Adelson, E. H. e Bergen, J. R. The Plenoptic Function and the Elements of Early Vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing* (1991).

[3] Akenine-Möller, T., Haines, E. e Hoffman, N. *Real-Time Rendering*, third ed. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[4] Aliaga, D. G. e Carlbom, I. Plenoptic stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 443–450.

[5] Apple Inc. QuickTime VR. Aplicativo que permite a visualização de ambientes virtuais a partir de uma imagem de panorama. `http://www.apple.com/quicktime/technologies/qtvr/`, acesso em: 26 de maio de 2008.

[6] Avidan, S. e Shashua, A. Novel View Synthesis in Tensor Space. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 1034–1040.

[7] Bao, P. e Gourlay, D. Superview 3d image warping for visibility gap errors. *IEEE Transactions on Consumer Electronics 49*, 1 (Feb. 2003), 177–182.

[8] Bao, P. e Gourlay, D. Remote Walkthrough over Mobile Networks Using 3-D Image Warping and Streaming. *IEE Proceedings - Vision, Image and Signal Processing 151*, 4 (Aug. 2004), 329–336.

[9] Bao, P. e Gourlay, D. A Framework for Remote Rendering of 3-D Scenes on Limited Mobile Devices. *IEEE Transactions on Multimedia 8*, 2 (April 2006), 382–389.

[10] Bao, P., Gourlay, D. e Li, Y. Mobile Collaborative Walkthrough Using 3-D Image Warping and Streaming. In *ITI 2004 - Proceedings of the 26th International Conference on Information Technology Interfaces* (June 2004), vol. 1, pp. 279–285.

[11] Boukerche, A. e Pazzi, R. W. N. Enhancing Remote Walkthrough for E-learning Environments on Mobile Devices over Heterogeneous Networks. In *Proceedings of 2006 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems* (July 2006), pp. 13–18.

[12] Boukerche, A. e Pazzi, R. W. N. Remote Rendering and Streaming of Progressive Panoramas for Mobile Devices. In *MULTIMEDIA '06: Proceedings of the 14th Annual ACM International Conference on Multimedia* (New York, NY, USA, 2006), ACM, pp. 691–694.

[13] Boukerche, A. e Pazzi, R. W. N. A Peer-to-Peer Approach for Remote Rendering and Image Streaming in Walkthrough Applications. In *Proceedings of 2007 IEEE International Conference on Communications - ICC '07* (June 2007), pp. 1692–1697.

[14] Brachtl, M., Slajs, J. e Slavík, P. PDA based navigation system for a 3D environment. *Computers & Graphics 25* (Aug 2001), 627–634.

[15] Chang, C.-F., Bishop, G. e Lastra, A. LDI Tree: A Hierarchical Representation for Image-Based Rendering. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 291–298.

[16] Chang, C.-F. e Ger, S.-H. Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering. In *Proceedings of 2002 Third IEEE Pacific Rim Conference on Multimedia - PCM '02* (London, UK, 2002), Springer-Verlag, pp. 1105–1111.

[17] Chen, S. E. QuickTime VR - An Image-Based Approach to Virtual Environment Navigation. *Computer Graphics 29*, Annual Conference Series (1995), 29–38.

[18] Chen, S. E. e Williams, L. View Interpolation for Image Synthesis. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 279–288.

[19] Chim, J., Lau, R., Leong, H. e Si, A. CyberWalk: A Web-based Distributed Virtual Walkthrough Environment. *IEEE Transactions on Multimedia 5*, 4 (Dec. 2003), 503–515.

[20] Debevec, P. E., Taylor, C. J. e Malik, J. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry and Image-Based Approach. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 11–20.

[21] Debevec, P. E., Yu, Y. e Borshukov, G. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *Proceedings of the 9th Eurographics Workshop on Rendering* (Vienna, Austria, June 1998), pp. 105–116.

[22] dos Santos, M. C. e Pedrini, H. Renderização de Cenas Tridimensionais Interativas em Computadores de Baixo Desempenho. In *Anais do V Workshop de Realidade Virtual e Aumentada (WRVA'2008)* (Bauru, SP, 2008), pp. 1–8.

[23] Funkhouser, T. Image Warping, 2000. Nota de aula sobre warping de imagens. http://www.cs.princeton.edu/courses/archive/fall00/cs426/lectures/warp/warp.pdf, acesso em: 23 de junho de 2008.

[24] Gortler, S. J., Grzeszczuk, R., Szeliski, R. e Cohen, M. F. The Lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 43–54.

[25] Greene, N. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications 6*, 11 (1986), 21–29.

[26] Hachet, M., Pouderoux, J., Knödel, S. e Guitton, P. 3D Panorama Service on Mobile Device for Hiking. In *CHI Workshop on Mobile Spatial Interaction* (2007), pp. 106–109.

[27] Hedström, A. C++ Sockets Library, 2008. `http://www.alhem.net/Sockets/`, acesso em: 30 de julho de 2008.

[28] Hoppe, H. Progressive Meshes. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.

[29] IJG. IJG - Independent JPEG Group, 2008. Biblioteca para codificação e decodificação de arquivos JPEG. `http://www.ijg.org/`, acesso em: 30 de julho de 2008.

[30] Intel. Intel® Threading Building Blocks, 2007. Biblioteca que provê o desenvolvimento de aplicações com computação paralela. `http://www.threadingbuildingblocks.org/`, acesso em: 15 de janeiro de 2009.

[31] JPEG. Joint Photographic Experts Group, 2008. Método de compressão de imagens com perda. `http://www.jpeg.org/`, acesso em: 30 de julho de 2008.

[32] Laveau, S. e Faugeras, O. 3-D Scene Representation as a Collection of Images. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition* (Oct 1994), vol. 1, pp. 689–691.

[33] Lei, Y., Jiang, Z., Chen, D. e Bao, H. Image-Based Walkthrough over Internet on Mobile Devices. In *Proceedings of the Grid and Cooperative Computing - GCC 2004 Workshops* (2004), Springer Berlin / Heidelberg, pp. 728–735.

[34] Levoy, M. e Hanrahan, P. Light Field Rendering. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 31–42.

[35] LTSP.org. Linux Terminal Server Project, 2008. `http://www.ltsp.org/`, acesso em: 30 de julho de 2008.

[36] Mark, W. *Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping.* PhD thesis, University of North Carolina, Chapel Hill, NC, USA, Apr 1999.

[37] McMillan, L. *An Image-Based Approach to Three-Dimensional Computer Graphics.* PhD thesis, University of North Carolina, Apr 1997.

[38] McMillan, L. e Bishop, G. Plenoptic Modeling: An Image-Based Rendering System. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM, pp. 39–46.

[39] Morimoto, C. E. *Redes e Servidores Linux*, Segunda ed. GDH Press e Sul Editores, 2006.

[40] Noimark, Y. e Cohen-Or, D. Streaming Scenes to MPEG-4 Video-Enabled Devices. *IEEE Computer Graphics and Applications 23*, 1 (Jan/Feb 2003), 58–64.

[41] Oliveira, M. M. Image-Based Modeling and Rendering Techniques: A Survey. *Revista de Informática Teórica e Aplicada 9*, 2 (2002), 37–66.

[42] OpenGL.org. Open Graphics Library, 2008. `http://www.opengl.org/`, acesso em: 30 de julho de 2008.

[43] Owen, G. S. 3D Camera Transformation, 1998. http://www.siggraph.org/education/materials/HyperGraph/viewing/view3d/3dview1.htm, acesso em: 23 de junho de 2008.

[44] Park, S., Kim, W. e Ihm, I. Mobile Collaborative Medical Display System. *Computer Methods and Programs in Biomedicine 89* (Mar 2008), 248–260.

[45] Peleg, S. e Herman, J. Panoramic Mosaics by Manifold Projection. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 338–343.

[46] Quax, P., Geuns, B., Jehaes, T., Lamotte, W. e Vansichem, G. On the Applicability of Remote Rendering of Networked Virtual Environments on Mobile Devices. In *ICSNC '06: Proceedings of the International Conference on Systems and Networks Communication* (Washington, DC, USA, 2006), IEEE Computer Society, p. 16.

[47] Quillet, J.-C., Thomas, G., Granier, X., Guitton, P. e Marvie, J.-E. Using Expressive Rendering for Remote Visualization of Large City Models. In *Web3D '06: Proceedings of the Eleventh International Conference on 3D Web Technology* (New York, NY, USA, 2006), ACM, pp. 27–35.

[48] Rafferty, M. M., Aliaga, D. G. e Lastra, A. A. 3D Image Warping in Architectural Walkthroughs. In *Proceedings of the Virtual Reality Annual International Symposium - VRAIS '98* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 228–233.

[49] Segal, M., Korobkin, C., van Widenfelt, R., Foran, J. e Haeberli, P. Fast shadows and lighting effects using texture mapping. In *SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), ACM, pp. 249–252.

[50] Seitz, S. M. e Dyer, C. R. View Morphing. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 21–30.

[51] Shade, J., Gortler, S., wei He, L. e Szeliski, R. Layered Depth Images. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 231–242.

[52] Shum, H.-Y. e He, L.-W. Rendering with Concentric Mosaics. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 299–306.

[53] Shum, H.-Y., Kang, S. B. e Chan, S.-C. Survey of Image-Based Representations and Compression Techniques. *IEEE Transactions on Circuits and Systems for Video Technology 13*, 11 (Nov. 2003), 1020–1037.

[54] Starlin, G. *TCP/IP - Redes de Computadores e Comunicação de Dados*. Editora Alta Books, 2004.

[55] Stevens, W. R. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional, 1993.

[56] Vancouver. A Virtual Visit of Vancouver, Canada. Sistema que possibilita um passeio virtual nas ruas da cidade de Vancouver. `http://www.virtuallyvancouver.com/indextour.html`, acesso em: 09 de março de 2008.

[57] Vanijja, V. e Horiguchi, S. Omni-Directional Stereoscopic Images from One Omni-Directional Camera. *The Journal of VLSI Signal Processing 42* (Jan 2006), 91–101.

[58] Winkelholz, C., Weiss, M., Cakuli, P., Schreiber, M. e Renkewitz, H. Open ActiveWrl, 2009. Biblioteca para a manipulação de objetos tridimensionais no formato VRML e X3D. `http://open-activewrl.sourceforge.net/`, acesso em: 15 de janeiro de 2009.

[59] Womack, P. e Leech, J. OpenGL® Graphics with the X Window System®, 1998. `http://www.opengl.org/documentation/specs/glx/glx1.3.pdf`, acesso em: 15 de janeiro de 2009.

[60] Woo, M., Neider, J. e Davis, T. *Opengl Programming Guide: The Official Guide to Learning Opengl, Version 1.1*, second ed. Addison-Wesley Pub (Sd), 1997.

[61] X.Org. X Window System, 2009. Protocolo que provê uma interface gráfica do usuário. `http://www.x.org/wiki/`, acesso em: 15 de janeiro de 2009.

[62] Yang, S. e Kuo, C.-C. J. Robust Graphics Streaming in Walkthrough Virtual Environments via Wireless Channels. In *GLOBECOM '03: Proceedings of the IEEE Global Telecommunications Conference* (Dec. 2003), vol. 6, pp. 3191–3195.

[63] Zhang, C. e Chen, T. A Survey on Image-Based Rendering–Representation, Sampling and Compression. *Signal Processing: Image Communication 19* (Jan 2004), 1–28.

[64] Zheng, H. e Boyce, J. An improved udp protocol for video transmission over internet-to-wireless networks. *IEEE Transactions on Multimedia 3*, 3 (Sep 2001), 356–365.

[65] Zheng, J. e Tsuji, S. Panoramic Representation of Scenes for Route Understanding. *10th International Conference on Pattern Recognition 1* (Jun 1990), 161–167.